TUM School of Computation, Information and Technology
Technische Universität München

Uncertainty Estimation for Independent and Non-Independent Data

Bertrand P.A.H. Charpentier

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr. Jens Großklags

Prüfer*innen der Dissertation:

1. Prof. Dr. Stephan Günnemann
2. Prof. Dr. Eyke Hüllermeier
3. Prof. Dr. Eric Nalisnick

Die Dissertation wurde am 13.04.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 22.01.2024 angenommen.

Abstract

Both *practical* and *theoretical* reasons justify why we need uncertainty estimation to build reliable machine learning models. While *uncertainty estimation* is expected to provide *trust*, *safety*, *fairness* and facilitate *maintenance* in real-world applications, uncertainty estimation is also highly required to represent the real physical world which is inherently *non-deterministic* and only *partially observable*.

Furthermore, machine learning models must deal with various types of input data (e.g. tabular, images, graph data, sequential data) and output data (classes, real values, counts, time events) which can be assumed either *independent* or *non-independent*. While the independence assumption is convenient to represent various data types, the non-independence assumption is particularly useful to represent complex data types with network effects or time effects.

In this dissertation, we look at uncertainty estimation for both independent and non-independent data. To this end, we elaborate on three main aspects. **(1)** We propose *desiderata* capturing the desired behavior of uncertainty estimation. These desiderata cover both aleatoric and epistemic uncertainty in the presence of perturbations – in particular adversarial perturbations –, as well as network effects, or time effects. Further, we analyze the desired behavior for uncertainty estimates at both training and testing time. **(2)** We present a large family of new Bayesian *models* which provide uncertainty estimates at a low practical cost. These models demonstrate strong empirical performance and have theoretical guarantees for different data types. **(3)** We develop extensive *metrics* to evaluate uncertainty estimates for practical tasks. These experimental setups cover correct/wrong prediction detection, Out-Of-Distribution (OOD) detection, dataset shifts, and calibration metrics in the presence of (adversarial) perturbations, network effects, or time effects. Finally, we analyze the benefit of using uncertainty estimates to achieve good exploration/exploitation trade-off with high sample efficiency.

Zusammenfassung

Sowohl *praktische* als auch *theoretische* Überlegeungen zeigen auf, dass Unsicherheitsschätzung benötigen für zuverlässige Modell des maschinellen Lernens benötigt werden. Während *Unsicherheitsschätzung* dazu beitragen soll *Vertrauen*, *Sicherheit*, und *Fairness* zu schaffen, sowie die *Wartung* in realen Anwendungen zu erleichtern, ist Unsicherheitsschätzung außerdem erforderlich, um die echte Welt, welche *nicht-deterministisch* und nur *teilweise beobachtbar* ist, adäquat abbilden zu können.

Modelle des maschinellen Lernens müssen darüber hinaus mit verschiedenen Arten von Eingabedaten – Tabellen, Bildern, Graphen, oder sequentiellen Daten – und Ausgabedaten – Klassen, reale Werte, Anzahl, Zeitereignisse – umgehen können, wobei diese sowohl als *unabhängige* oder *nicht unabhängige* Instanzen auftreten können. Während die Unabhängigkeitsannahme dabei hilft, verschiedene Datentypen abzubilden, ist die Nicht-Unabhängigkeitsannahme besonders im Kontext von komplexen Datentypen mit Netzwerkeffekten oder Zeiteffekten nützlich.

In dieser Dissertation betrachten wir die Unsicherheitsschätzung sowohl für unabhängige als auch für nicht unabhängige Daten und gehen hierfür auf drei Hauptaspekte ein. **(1)** Wir schlagen *Desiderata* vor und diskutieren diese, um das gewünschte Verhalten der Unsicherheitsschätzung zu umreißen. Diese Desiderata umfassen sowohl aleatorische als auch epistemische Ungewissheit in Gegenwart von Störungen – insbesondere kontradiktorischer Störungen – sowie Netzwerk- oder Zeiteffekten. Außerdem analysieren wir das gewünschte Verhalten für Unsicherheitsschätzungen sowohl zur Trainings- als auch zur Testzeit. **(2)** Wir stellen eine Familie neuer bayesscher *Modelle* vor, die Unsicherheitsschätzungen zu geringen praktischen Kosten liefern. Diese Modelle zeigen eine starke empirische Leistung und liefern theoretische Garantien für verschiedene Datentypen. **(3)** Wir entwickeln umfangreiche *Metriken* zur Bewertung von Unsicherheitsschätzungen für praktische Aufgaben. Diese Versuchsaufbauten decken die Erkennung richtiger und falscher Vorhersagen, Out-Of-Distribution (OOD)-Erkennung, Datensatzverschiebungen und Kalibrierungsmetriken in Gegenwart von (gegensätzlichen) Störungen, Netzwerkeffekten oder Zeiteffekten ab. Zuletzt analysieren wir den Nutzen der Verwendung von Unsicherheitsschätzungen, um einen guten Kompromiss zwischen Exploration und Ausbeutung mit hoher Probeneffizienz zu erreichen.

Acknowledgements

I would first like to thank my supervisor Prof. Stephan Günnemann. I feel incredibly lucky for the opportunity to pursue a Ph.D. under your guidance. Thank you for showing me how to become a better researcher, teacher, presenter, and supervisor. I enjoyed our brainstorming sessions where you always provided helpful and precise feedback. Thank you for your understanding and great support which was always motivating to achieve great work.

I am very grateful to all DAML group. I could not have imagined a better environment for my Ph.D. study. I would like to thank everyone in the group for their readiness to help, discuss ideas, and share research progress. It was a lot of fun during and outside working hours. I would like to particularly thank all my colleagues, collaborators, and co-authors I had the chance to work with at the Technical University of Munich. Thank you Simon Geisler, Anna-Kathrin Kopetzki, Tom Wollshläger, and Nicholas Gao. Thank you also Marin Biloš for your support, help, and your great sense of humor. It was a lot of fun to share our office. Special thanks to Daniel Zügner for all your support and openness. Beyond all the great research collaborations, I would like to thank you for all the friendly discussions and advice.

While pursuing my Ph.D., I had the chance to work with bright, curious, and motivated students. Thank you Oliver, Maximilian, Daniel, Johannes, Sascha, Igor, Franz, Raphael, Niklas, Stephan, Simon, Sven, Chenxiang, Xun, Morgane, and John.

I am very grateful to Prof. Mykel Kochenderfer from Stanford University for hosting me during my research stay. I would like to thank everyone from SISL lab with special thanks to Ransalu Senanayake. You always make me feel welcome and provided me with great advice during my stay.

I am also very grateful to the members of the Twitter Cortex team for your warm welcome and support despite the intense time at the company. I would like to particularly thank Nils Hammerla, Emanuele Rossi, Francesco Di Giovianni, and Prof. Michael Bronstein for the great discussions and research collaborations.

I would like to also sincerely thank Prof. Eyke Hüllermeier and Prof. Eric Nalisnick for carefully reading, examining, and questioning this thesis. It is an honor to have established experts in the field reviewing my work.

I would also like to express my very profound gratitude to my family, my friends, and my partner for providing me with continuous encouragement, joy, and happiness throughout these years. I am particularly thankful to my parents who provided me with unconditional support. I love you.

This thesis is also dedicated to my beloved grandfather Alain, godfather Bertrand, and cousin Quentin who passed away. Thank you for all the time that I could share with you. You remain a great source of inspiration for me.

Contents

Part I

Introduction

1 Introduction

As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.

Albert Einstein

Madness is the consequence not of uncertainty but of certainty.

Friedrich W. Nietzsche

Artificial Intelligence (AI) consists in the intelligence demonstrated by machines in contrast with intelligence demonstrated by humans or animals. Hence, AI models are usually computing systems able to perform tasks which would normally require human intelligence, such as visual perception, speech recognition, decision-making, translation between languages, or even art generation. AI includes Machine Learning (ML) which consists in the study of computer programs which automatically learn from experience [290], and Deep Learning (DL) which is a subfield of ML and can be defined as models which hierarchically learn complex representations based on simpler ones [164]. AI is a very important technology with high potential *economic* and *ethical* impacts.

Economically, AI has a high momentum. Indeed, 48% of companies claim that they use AI [326] and 83% companies have AI is their priority [136]. Overall, the market value of AI is $136 billion in 2022 [357] and had a Compound Annual Growth Rate (CAGR) of 38.5% between 2022-2030 [357], thus indicating a very fast growth. Hence, AI finds many applications in industry and science. Industry applications include e.g. agriculture (e.g. Cropin, Iron Ox, FarmWise), manufacturing (e.g. Exotec, Bright Machines, Fieldbox), automotive industry (e.g. Tesla, Waymo), medicine (e.g. Exscientia, Valence Discovery, Radium), construction (Procore Technologies, OpenSpace), finance (e.g. Affirm, Kreditech, Kayrros), education (e.g. Duolingo, Age of Learning), or even AI for art (e.g. Bytedance, OpenAI). Scientific applications include e.g. physics (e.g. [145]), chemistry (e.g. [198]), biology (e.g. [186]), or even mathematics (e.g. [88]). In all these applications, AI have shown to be applicable with various data types (e.g. images, tabular, text, graphs) and at various scales from very small (e.g. quantum systems or molecules) to very large systems (e.g. social networks or climate).

Ethically, AI systems raises multiple major concerns [24]. It does not guarantee a safe behavior and can create accidents [11]. It can generate fake data, thus potentially to fake news [317]. It does not provide clear explanations for its predictions or provide unreliable explanations [304, 297, 236]. It can be racist and discriminate minorities, e.g., due to

biased data or manipulation of learning by users [284]. It can partially replace human jobs by stealing human creations which leaves the question of intellectual property unclear [294, 155]. The replaced jobs include activities like art which can also be enjoyable for humans. It might reduce individuals' control over their lives and diminish individuals' cognitive, social and survival skills as they become dependent on AI [12]. It might lead to persistent surveillance and violate data privacy [23]. Finally, it might have goals which are not aligned with human goals. This turns out to be particularly problematic when AI systems achieve super-intelligence [50] which is realistic in many tasks where AI systems are already better than humans [134].

Hence, the fast AI economic growth and the multiple AI ethical concerns urge the development of reliable AI models which is the main subject of this thesis.

1.1 Why do we need uncertainty estimation?

Uncertainty estimation (a.k.a. *uncertainty quantification*) consists in evaluating the confidence of models in their predictions. This task is crucial for both *practical* and *theoretical* reasons summarized in Fig. 1.1.

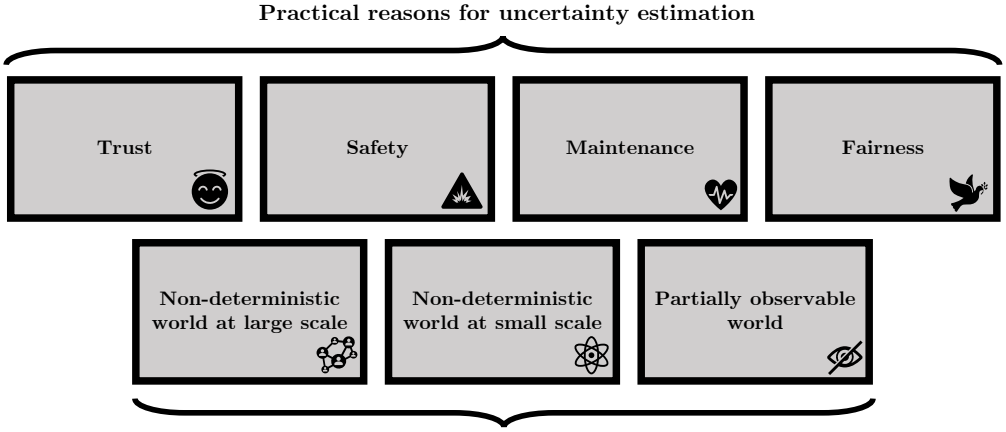


Figure 1.1: Overview of the *practical* and *theoretical* reasons for uncertainty estimation.

Practical reasons. The Dunning-Kruger effect [240] describes a psychological cognitive bias in which people lacking knowledge in a particular domain overestimate their abilities. Interestingly, a similar phenomenon also applies to machine learning models. Traditional neural networks show overconfident predictions, in particular on data that are different from the data utilized during training [182]. The illusion of knowledge of machine learning models highly impacts the reliability of such models in safety-critical domains. First, it affects the *trust* in ML model predictions. Ideally, we expect ML models to be confident when predicting correctly and uncertain when predicting wrongly. Second, it affects the *safety* of ML predictions in unfamiliar situations. Ideally, we expect ML models to flag

predictions on unknown domains corresponding to anomaly detection. Third, it affects the ease of *development* and *maintenance* of ML models. Ideally, we expect ML models to assign high uncertainty to data worth to train on or become more uncertain when the testing data has drifted away from the training data, thus indicating the need of retraining the model. This is particularly important in application where ML models need to efficiently explore and learn all life-long. And finally, it affects the *fairness* of ML models. Ideally, we expect ML models to provide calibrated predictions on all input regions including underrepresented data. We relate each of this practical motivation for uncertainty estimation to concrete uncertainty metrics in Section 2.3.

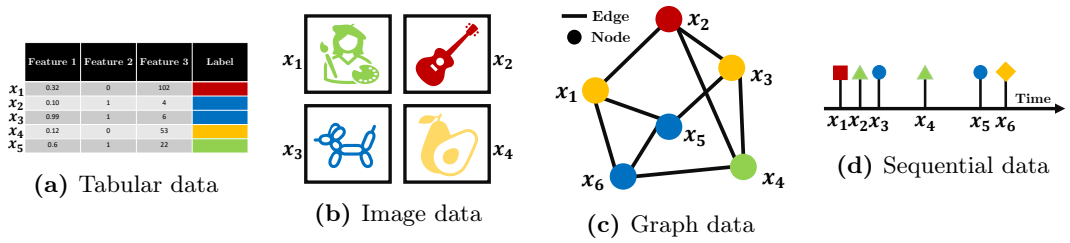
Theoretical reasons. ML models aim at precisely representing the flow of information in the world which is inherently uncertain. First, the world is *non-deterministic at a large scale*. Indeed, uncertainty emerges when small individual events contribute to macro phenomena like GDP growth, micro phenomena like the growth rate of firms, and non-economic events like war and climate change [41]. Second, the world is *non-deterministic at a small scale*. A prominent evidence of this is the uncertainty principle in (quantum) physics. Indeed, the uncertainty principle implies that it is in general not possible to predict the value of a particle quantity (like position and speed) with arbitrary certainty, even if all initial conditions are specified [187]. Eventually, the world is also only *partially observable*. On one hand, any agent (e.g. human or robot) has to deal with incomplete information on the environment in which it evolves [212]. Indeed, its available information is limited by its internal sensors which cannot capture any signal type at any resolution. On the other hand, the accessible information is restricted to the observable world which is, at the extreme, limited by the speed of light [325], thus making the world inherently uncertain for any agent (even if it is equipped with perfect sensors).

All these reasons underline the need of accurate uncertainty estimation methods in ML. Specifically, a reliable ML model should provide high-quality estimates of aleatoric and epistemic uncertainty [140]. The *aleatoric uncertainty* allows a model to account for the irreducible data uncertainty (e.g. the inherent sensor noise, or the inherent environment stochasticity). The *epistemic uncertainty* allows a model to account for its lack of knowledge about unseen data regions (e.g. testing data differs significantly from training data). Aleatoric and epistemic uncertainty levels can eventually be combined into an overall *predictive uncertainty* [140]. Hence, this thesis studies the usage of different types of uncertainty for ML methods.

1.2 Why do we need to handle independent and non-independent data?

In this thesis, we consider ML models which process input data $\boldsymbol{x}_i$ to accurately predict output targets y_i. More specifically, we expect ML models to be able to process *any type of input data* (e.g. tabular, images, graph or time series) to predict *any type of*

output data (e.g. classes, continuous values, counts). To this end, *independence* and *non-independence* are key assumptions to create *practical* and *accurate* models which precisely describe the real-world.



(a) Tabular data **(b)** Image data **(c)** Graph data **(d)** Sequential data

Figure 1.2: Overview of different data types covering independent $\boldsymbol{x}_i$ inputs like tabular and image data, and dependent $\boldsymbol{x}_i$ inputs like graph nodes and sequential time events.

Independent Data. The independence assumption means that, given the knowledge of the underlying mechanism that generates the data, different data samples do not have any internal dependency amongst themselves. In other words, the knowledge of one data sample does not bring any information on another data sample given the true data generating process. The independence assumption is particularly useful when representing tabular data [355] (e.g. a group of unrelated persons for a medicine trial, defects on multiple different devices) or image data [264, 257] (e.g. disease detection on medical images of different patients, object detection in different self-driving cars or robots). Visualizations of such data are represented in Fig. 1.2a and Fig. 1.2b. In this case, representing the interaction between data samples does not bring much information to perform the predictions. Hence, the key advantage of the independence assumption is that it allows mathematical factorization for practical modelling simplifications [37] without important information loss.

Non-Independent Data. The non-independence assumption means that, given the knowledge of the underlying mechanism that generates the data, different data samples might still have some internal dependencies amongst themselves. In other words, observing a data sample gives some information on the value of another data sample even when knowing the true data generating process. The non-independence assumption is particularly useful when representing graph data [447] (e.g. social networks, citation networks) or sequential data [383, 107] (e.g. financial time series, interaction history of a user). Visualizations of such data are represented in Fig. 1.2c and Fig. 1.2d. In this case, neighboring nodes of a graph are expected to share important information and past events are expected to give important information on future events. Hence, the key advantage of the non-independence assumption is that it retains the information contained in graph and sequential interactions to model the world more precisely.

Beyond the *(non-)independence assumption*, another common assumption in ML is that data are *identically distributed*. This assumption assumes that all input data comes

from the same data distribution which also has strong limitations related to the reasons motivating the usage of uncertainty estimates for ML predictions. While the training data are assumed to come from the same data distribution, the testing data might come from different distributions and suffer from *Distribution Shifts* [350, 335]. Indeed, the testing data might come from the *In-Distribution (ID)* similar to the training data, or a *Out-Of-Distribution (OOD)* which could be any distribution different from the training distribution [453, 385]. This scenario frequently happens in *safety* and *maintenance* use-cases where the data distribution observed by the model continuously drifts at testing time.

Hence, this thesis acknowledges the limitation of the standard *independent and identically distributed (i.i.d.)* assumptions by studying the application of uncertainty estimation to independent and non-independent data.

1.3 Contributions and outline

This thesis studies the application of uncertainty estimation for independent and non-independent data via three main components:

- Explicit *desiderata* capturing the desired behavior of uncertainty estimation.
- Accurate and efficient *models* for uncertainty estimation with low practical overhead.
- Practical *metrics* evaluating uncertainty estimation in real-world applications including worst-case scenarios.

In Chapter 2, we start by establishing the background knowledge on uncertainty estimation around which this thesis is articulated. In particular, we first present the desiderata related to the Bayesian properties, the different types of uncertainty estimates, and the practical requirements in Section 2.1. Second, we present an overview of the important families of methods for uncertainty estimation in Section 2.2. Third, we present metrics used in experimental setups for uncertainty estimation which answer practical questions on reliability for ML models in Section 2.3.

In Part II, we present a study of uncertainty estimation for *independent data*: In Chapter 3, we construct a new Bayesian model for uncertainty estimation for classification called Posterior Network (PostNet). PostNet requires a *single-forward pass*, does need *no OOD data during training*, and adapt to *many core architectures*. In Chapter 4, we construct a new Bayesian model for uncertainty estimation for regression called Natural Posterior Network (NatPN). Beyond regression, NatPN applies to a *large variety of supervised tasks* (incl. classification and count prediction) with *very low computational overhead*. In Chapter 5, we study the practicality of efficient uncertainty estimation methods by analyzing the role of the *training*, the *architecture* and the *prior* in their final performances. In particular, we show a fundamental trade-off between OOD generalization and OOD detection performance in the presence of feature collapse. In Chapter 6,

Table 1.1: List of own publications that this thesis is based on. Code and datasets for the respective publications are available at `www.cs.cit.tum.de/daml/[project]`.

Ch.	Ref.	Title	Conference	Repository
3	[67]	Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts	NeurIPS 2020	`/postnet/`
4	[68]	Natural posterior network: Deep bayesian predictive uncertainty for exponential family distributions.	ICLR 2021	`/natpn/`
5	[72]	Training, Architecture, and Prior for Deterministic Uncertainty Methods?	TrustML - ICLR 2023	`/training-architecture-prior-dum/`
6	[235]	Evaluating robustness of predictive uncertainty estimation: Are dirichlet-based models reliable?	ICML 2020	`/dbu-robustness/`
8	[398]	Graph posterior network: Bayesian predictive uncertainty for node classification.	NeurIPS 2021	`/graph-postnet/`
9	[36]	Uncertainty on asynchronous time event prediction	NeurIPS 2019	`/uncertainty-event-prediction/`
10	[71]	Disentangling epistemic and aleatoric uncertainty in reinforcement learning	DFUQ - ICML 2022	`/aleatoric-epistemic-uncertainty-rl/`

we present the first study of the robustness of uncertainty models in the worst-case scenario of adversarial attacks. We show that uncertainty estimates of an important family of single-forward pass models are *not robust* for many important real-world tasks (incl. correct/wrong prediction detection, adversarial examples detection, and ID/OOD detection). Further, we explore first approaches methods to improve uncertainty robustness by using *adversarial training* and *randomized smoothing*. In Chapter 7, we present a retrospective on the evolution of the research field since our first study on independent data.

In Part III, we present a study of uncertainty estimation for *non-independent data*: In Chapter 8, we present the first framework for uncertainty estimation for node classification on graph data. This framework proposes explicit desiderata, a new Bayesian model, and an exhaustive evaluation setup which covers aleatoric and epistemic uncertainty estimation *without* and *with network effects*. In Chapter 9, we present new models for uncertainty estimation in sequential data with asynchronous time events. The two proposed models are able of accurate *event types* and *event time* predictions while capturing *uncertainty with rich temporal evolution*. In Chapter 10, we present the first framework to disentangle aleatoric and epistemic uncertainty estimation in reinforcement learning. This framework proposes explicit desiderata, four models inspired from supervised learning, and a detailed experimental setup which cover aleatoric and epistemic uncertainty estimation at both *training time* and *testing time*. In Chapter 11, we present a retrospective on the evolution of the research field since our first study on non-independent data.

In Part IV, we present a conclusion including open questions as suggestion to future research directions (see Chapter 12).

1.4 Own publications

The content of Chapters 3 to 10 is mostly based on papers previously published at international peer-reviewed conferences. We list these papers in Table 1.1. We also provide the full list of publications that the author was involved in during the PhD

studies below. In case of multiple equal contributions, the name of the first authors are starred with "*". These publications focus on three main topics: *uncertainty estimation* including Bayesian models, energy-based models and robustness [67, 68, 235, 72, 398, 36, 71, 120, 21], *structure learning* including hierarchical and directed acyclic graphs [70, 66, 481, 48], and *efficient ML* including pruning methods and sparse neural networks [351, 21, 157].

[1] Morgane Ayle, Bertrand Charpentier, John Rachwan, Daniel Zügner, Simon Geisler, and Stephan Günnemann. On the robustness and anomaly detection of sparse neural networks. In *Sparsity in Neural Networks Workshop, SNN*, 2022.

[2] *Marin Biloš, *Bertrand Charpentier, and Stephan Günnemann. Uncertainty on asynchronous time event prediction. *Neural Information Processing Systems, NeurIPS*, 2019.

[3] Thomas Bonald, Nathan de Lara, Quentin Lutz, and Bertrand Charpentier. Scikit-network: Graph analysis in python. *Journal of Machine Learning Research, JMLR*, 2020.

[4] Bertrand Charpentier and Thomas Bonald. Tree sampling divergence: An information-theoretic metric for hierarchical graph clustering. In *International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

[5] *Bertrand Charpentier, *Oliver Borchert, Daniel Zügner, Simon Geisler, and Stephan Günnemann. Natural posterior network: Deep bayesian predictive uncertainty for exponential family distributions. In *International Conference on Learning Representations, ICLR*, 2021.

[6] Bertrand Charpentier, Simon Kibler, and Stephan Günnemann. Differentiable dag sampling. In *International Conference on Learning Representations, ICLR*, 2022.

[7] Bertrand Charpentier, Ransalu Senanayake, Mykel Kochenderfer, and Stephan Günnemann. Disentangling epistemic and aleatoric uncertainty in reinforcement learning. In *ICML Workshop on Distribution-Free Uncertainty Quantification, ICML - DFUQ*, 2022.

[8] Bertrand Charpentier, Chenxiang Zhang, and Stephan Günnemann. Training, architecture, and prior for deterministic uncertainty methods. In *ICLR 2023 Workshop on Pitfalls of limited data and computation for Trustworthy ML*, 2023.

[9] Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. *Neural Information Processing Systems, NeurIPS*, 2020.

[10] Sven Elflein, Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. On out-of-distribution detection with energy-based models. In *ICML workshop on Uncertainty and Robustness in Deep Learning Workshop, UDL - ICML*, 2021.

[11] Johannes Getzner, Bertrand Charpentier, and Stephan Günnemann. Accuracy is not the only metric that matters: Estimating the energy consumption of deep learning models. In *ICLR 2023 Workshop on Tackling Climate Change with Machine Learning: Global Perspectives and Local Challenges*, 2023.

[12] *Anna-Kathrin Kopetzki, *Bertrand Charpentier Daniel Zügner, Sandhya Giri, and Stephan Günnemann. Evaluating robustness of predictive uncertainty estimation: Are dirichlet-based models reliable? In *International Conference on Machine Learning, ICML*, 2020.

[13] John Rachwan, Daniel Zügner, Bertrand Charpentier, Simon Geisler, Morgane Ayle, and Stephan Günnemann. Winning the lottery ahead of time: Efficient early network pruning. In *International Conference on Machine Learning, ICML*, 2022.

[14] *Maximilian Stadler, *Bertrand Charpentier, Simon Geisler, Daniel Zügner, and Stephan Günnemann. Graph posterior network: Bayesian predictive uncertainty for node classification. In *Advances in Neural Information Processing Systems*, 2021.

[15] Daniel Zügner, Bertrand Charpentier, Morgane Ayle, Sascha Geringer, and Stephan Günnemann. End-to-end learning of probabilistic hierarchies on graphs. In *International Conference on Learning Representations, ICLR*, 2022.

2 Background

Science is the outcome of being prepared to live without certainty and therefore a mark of maturity. It embraces doubt and loose ends.

A.C. Grayling

In this chapter, we introduce the main background on the content of this thesis. It covers background on the desiderata, models, and metrics for uncertainty estimation in Machine Learning. In order to preserve the original storyline of the original publications, we kept the background sections in Chapters 3, 4, 6 and 8 to 10. Beyond this chapter, we also recommend existing surveys on uncertainty estimation for further details on uncertainty estimation in deep learning [150, 3, 348, 13, 202].

2.1 Uncertainty desiderata

In this section, we review important desiderata for uncertainty estimation. We provide a summary of these desiderata in Table 2.1.

Bayesian distributions	Uncertainty types	Practical requirements
$\mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$ where $\boldsymbol{\phi}$ are model weights	Aleatoric uncertainty	Efficiency: Data & Time
$\mathbb{Q}(\boldsymbol{a} \mid \mathcal{D}, \boldsymbol{x})$ where $\boldsymbol{a}$ are values of the activations	Epistemic uncertainty	Flexibility: Architecture & Optimization
$\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{x})$ where $\boldsymbol{\theta}$ are parameters of the target distribution	Predictive uncertainty	Robustness: Natural & Adversarial

Table 2.1: Overview of desiderata for models for uncertainty estimation. Important desiderata involve modelling Bayesian distributions, distinguishing between uncertainty types, and fulfilling practical requirements.

2.1.1 Bayesian Distributions

Bayesian distributions offer a convenient framework to express beliefs in the realization of an event. In particular, the Bayesian framework allows to easily incorporate prior knowledge and update our beliefs given additional new observations in a principled way. Hence, we first recall the key concept of the Bayesian framework since it is a core concept in many uncertainty estimation approaches.

Unsupervised learning. We define the distribution $\mathbb{P}(y \mid \boldsymbol{\theta})$ over the target variable $y \in \mathbb{R}^K$ given the parameter $\boldsymbol{\theta}$. Given a dataset $\mathcal{D} = \{y^{(1)}, ..., y^{(N)}\}$, the Bayes formula is:

$$\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{\mathbb{P}(\mathcal{D} \mid \boldsymbol{\theta}) \times \mathbb{Q}(\boldsymbol{\theta})}{\mathbb{P}(\mathcal{D})} \tag{2.1}$$

where $\mathbb{P}(\mathcal{D} \mid \boldsymbol{\theta})$ is the *likelihood*, $\mathbb{Q}(\boldsymbol{\theta})$ is the *prior*, $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D})$ is the *posterior*, and $\mathbb{P}(\mathcal{D})$ is the *evidence*. Intuitively, the Bayesian formula updates the prior belief represented by $\mathbb{Q}(\boldsymbol{\theta})$ into the posterior belief represented by $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D})$ after observing a dataset $\mathcal{D}$ [37]. The choice of prior is crucial. A common choice is to follow the *principle of maximum entropy* [364] and enforce high entropy for the prior distribution which is usually considered less informative. However, note that many works studied different choices of priors [209, 388]. The evidence term $\mathbb{P}(\mathcal{D})$ corresponds to a normalization constant which can sometimes be ignored [37].

After observing a dataset $\mathcal{D}$, we can update the distribution over the target variable y in two ways. A first option is to use a point-wise estimate of the target distribution parameter, i.e.:

$$\mathbb{P}(y \mid \boldsymbol{\theta}^*) \tag{2.2}$$

where $\boldsymbol{\theta}^* = \arg\max \mathbb{P}(\mathcal{D} \mid \boldsymbol{\theta})$ would be the maximum likelihood estimate, or $\boldsymbol{\theta}^* = \arg\max \mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D})$ would be the maximum a posteriori estimate. A second option is to integrate over all possible values for the target distribution parameter, i.e.:

$$\mathbb{P}(y \mid \mathcal{D}) = \int \mathbb{P}(y \mid \boldsymbol{\theta})\, \mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D}) d\boldsymbol{\theta} \tag{2.3}$$

where $\mathbb{P}(y \mid \mathcal{D})$ is called the posterior predictive distribution. This second approach is often considered to be more Bayesian since it depends on the full posterior distribution $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D})$. However, it might be costly to compute the posterior distribution since it requires integration.

Supervised learning. In supervised learning, the goal is to predict the value of a target output variable y given an input $\boldsymbol{x}$ after observing a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), ..., (\boldsymbol{x}^{(N)}, y^{(N)})\}$. In this case the posterior predictive requires to be adapted with three main options: compute the posterior distribution over the weights $\mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$, compute the posterior over the activations $\mathbb{Q}(\boldsymbol{a} \mid \mathcal{D}, \boldsymbol{x})$, and compute the posterior over the parameters of the target distribution $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{x})$ (see Table 2.1 left). The first option is:

$$\mathbb{P}(y \mid \mathcal{D}, \boldsymbol{x}) = \int \mathbb{P}(y \mid \boldsymbol{\phi}, \boldsymbol{x})\, \mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D}) d\boldsymbol{\phi} \tag{2.4}$$

where $\boldsymbol{\phi}$ denote the model weights. In this case, the parameter distribution $\mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$ is not conditioned on the input $\boldsymbol{x}$ and only accounts for an uncertainty dependent on the dataset $\mathcal{D}$. Examples of such models are Bayesian neural networks which learn Bayesian distribution $\mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$ where $\boldsymbol{\phi}$ denote the model weights [42]. The second option is:

$$\mathbb{P}(y \mid \mathcal{D}, \boldsymbol{x}) = \int \mathbb{P}(y \mid \boldsymbol{a})\, \mathbb{Q}(\boldsymbol{a} \mid \mathcal{D}, \boldsymbol{x}) d\boldsymbol{a} \tag{2.5}$$

where $\boldsymbol{a}$ denote intermediate representations of $\boldsymbol{x}$. In this case, the parameter distribution $\mathbb{Q}(\boldsymbol{a} \mid \mathcal{D}, \boldsymbol{x})$ is conditioned on the new input $\boldsymbol{x}$ and accounts for an uncertainty dependent on the input $\boldsymbol{x}$. Examples of such models are Bayesian neural networks which learn Bayesian distribution $\mathbb{Q}(\boldsymbol{a} \mid \mathcal{D}, \boldsymbol{x})$ where $\boldsymbol{a}$ denote the values of the activations [300, 425]. Alternatively, the third option is:

$$\mathbb{P}(y \mid \mathcal{D}, \boldsymbol{x}) = \int \mathbb{P}(y \mid \boldsymbol{\theta})\, \mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{x}) d\boldsymbol{\theta} \tag{2.6}$$

where $\boldsymbol{\theta}$ denote the direct parameters of the distribution over the target y. In this case, the parameter distribution $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{x})$ is conditioned on the new input $\boldsymbol{x}$ and also accounts for an uncertainty dependent on the input $\boldsymbol{x}$. Examples of such models are the Bayesian neural networks proposed in this thesis [67, 69, 398, 71, 36] which learn Bayesian distributions $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{x})$ where $\boldsymbol{\theta}$ denote the parameters of the distribution over the target labels y. In this thesis, we focus on learning Bayesian distributions on the (generally) low dimensional target parameters $\boldsymbol{\theta}$, in contrast to the (generally) high dimensional activation $\boldsymbol{a}$ or weights $\boldsymbol{\phi}$, thus advantageously allowing to reduce the computation complexity of the posterior distribution.

2.1.2 Aleatoric, Epistemic & Predictive Uncertainty

The Bayesian formula in Eq. (2.3) involves three types of distribution (i.e. $\mathbb{P}(y \mid \boldsymbol{\theta})$, $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D})$, and $\mathbb{P}(y \mid \mathcal{D})$) which cover the three main sources of uncertainty: the aleatoric uncertainty represented by the distribution $\mathbb{P}(y \mid \boldsymbol{\theta})$, the epistemic uncertainty represented by the distribution $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D})$, and the predictive uncertainty represented by the distribution $\mathbb{P}(y \mid \mathcal{D})$ (see Table 2.1 middle).

Aleatoric uncertainty. The *aleatoric uncertainty* is sometimes also called data uncertainty, stochastic uncertainty or risk [202, 231, 271]. The aleatoric uncertainty is represented by the distribution $\mathbb{P}(y \mid \boldsymbol{\theta})$. The aleatoric uncertainty should be high when *the model does not know because of inherent noise in a given context* (e.g. noisy environment, noisy sensors, low computation resources, model mispecification) [437, 202]. Given a specific context, the aleatoric uncertainty is *irreducible* since additional observations cannot resolve the information loss due noisy measurements or misspecifications. However, aleatoric uncertainty can be reduced by using higher measurement resolution or improving the model specification.

Epistemic uncertainty. The *epistemic uncertainty* is sometimes also called knowledge uncertainty, systematic uncertainty, or knightian uncertainty [202, 231, 271]. The epistemic uncertainty is represented by the distribution $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D})$. The epistemic uncertainty should be high when *the model does not know because of a lack of observed data* in the dataset $\mathcal{D}$. Hence, the epistemic uncertainty is *reducible* since it should decrease when collecting additional data.

Predictive uncertainty. The *predictive uncertainty* is sometimes also called total uncertainty [202, 271]. The predictive uncertainty is represented by the distribution $\mathbb{P}(y \mid \mathcal{D})$. Intuitively, the predictive uncertainty aggregates the effect of the aleatoric and epistemic uncertainty by integrating jointly the distributions $\mathbb{P}(y \mid \boldsymbol{\theta})$ and $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D})$.

In practice, each uncertainty type can be measured by computing the spread of its respective distribution with the (differential) entropy which represents its variability [270, 140], i.e.:

$$u_{\text{alea}} = \mathbb{H}[\mathbb{P}(y \mid \boldsymbol{\theta})], \quad u_{\text{epist}} = \mathbb{H}[\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D})], \quad u_{\text{pred}} = \mathbb{H}[\mathbb{P}(y \mid \mathcal{D})] \tag{2.7}$$

Apart from the entropy, other uncertainty metrics can be commonly used. E.g. the variance of the distributions can indicate different uncertainty types, the max probability for classification [271] can indicate the aleatoric uncertainty, or the concentration parameters can indicate the epistemic uncertainty if they exist [67].

In this thesis, we focus on methods capable to estimates the three types of uncertainty types.

2.1.3 Efficiency, Flexibility & Robustness

Uncertainty methods are also expected to have practical characteristics (see Table 2.1 right).

First, an uncertainty method is expected to be *efficient* at both training and testing time. A first aspect is *time efficiency* meaning that the method should be fast with low computational overhead. E.g. uncertainty methods requiring multiple forward passes are usually more expensive than method require a single forward pass. As second aspect is *data efficiency* meaning that the method should require as few data as possible to train.

Second, an uncertainty method is expected to be *flexible*. A first aspect is *architecture flexibility* meaning that the method should easily adapt to any architectures in order to easily adapt to different input types (e.g. tabular, images, graphs, and sequential data) and output types (e.g. classification and regression). A second aspect is *optimization flexibility* meaning that the method should easily adapt to different training schemes including end-to-end training or fine-tuning based on pretrained models.

Finally, an uncertainty method is expected to be *robust*. A first aspect is *natural robustness* meaning that the method should be performant even if there is some natural drift in the data. Natural drifts could be due to time evolution or location variability [233, 314, 278]. A second aspect is *adversarial robustness* meaning that the method should be performant even against adversarial perturbations which are specifically designed to fool the model. Adversarial perturbations can be viewed as the worst-case scenario for the model. Different methods to compute adversarial perturbations including white box attacks which do not have information about the model, black box attacks which have full access to the model information, and gray box which have partial information about the model [451].

In this thesis, we focus on proposing practical methods which are efficient in terms of data and time, flexible in terms of architecture and optimization schemes, and robust in terms of natural and adversarial perturbations.

2.2 Uncertainty models

In this section, we review important families of models for uncertainty estimation. We provide a summary of these models in Fig. 2.1. We refer the reader to seed papers and surveys provided in the following paragraphs presenting each family of methods for a detailed presentation of their concepts. Overall, we observed that these families of methods could be approximately be classified in two groups. On one hand, ensembles, Monte Carlo dropout, Markov chain Monte Carlo, variational inference, Laplace approximation, and Gaussian process methods form a first group of methods with strong Bayesian guarantees but often requiring expensive computations. On the other hand, evidential, calibration, density-based, energy-based, and distance-based methods form a second group of methods sometimes capable of efficient uncertainty predictions to the cost of weaker Bayesian guarantees or additional practical constraints like the need of additional training or validation data.

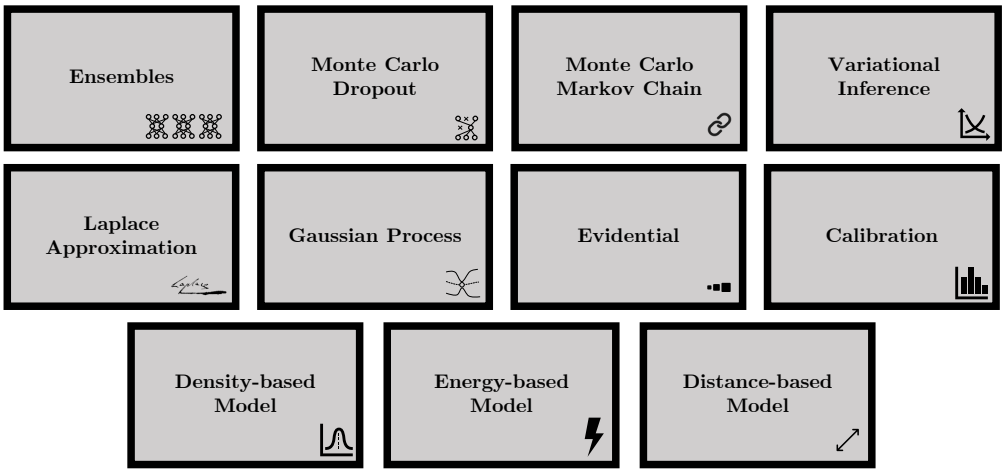


Figure 2.1: Overview of the different families of methods for uncertainty estimation.

Ensembles. This family of methods consists in combining the predictions of a set of multiple models $f_{\boldsymbol{\phi}_1}, \cdot, f_{\boldsymbol{\phi}_K}$ by using stacking, bagging, boosting, Bayesian averaging or other Bayesian combinations [298]. The variance of the different model predictions can be viewed as uncertainty estimates. These uncertainty estimates can be considered Bayesian where the member weights are sampled from the posterior weight distribution, i.e. $\boldsymbol{\phi}_k \sim \mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$. The ensembles can be *homogeneous ensembles*, i.e. the models share the same architecture, be *heterogeneous ensembles*, i.e. the models have different architectures, or *implicit ensembles*, i.e. a single model approximated ensembles parameter sampling [4]. Further, the different models can be trained sequentially (see e.g. [370, 74]) or independently (see e.g. [246]). Ensembles are considered as strong baselines in terms of uncertainty estimation and predictive performances [335]. Ensembles also benefit from a simple implementation [246]. Ensembles are often considered to be expensive as they require multiple forward passes. Many approaches [434, 178] proposed to make faster

ensembles. Further, while ensembles improve predictive performance, this performance gain can actually be replicated through the use of (larger) single models [4].

Variational Inference. This family of methods consists in approximating a (generally intractable) posterior distribution with a variational distribution belonging to a tractable family of distributions [40]. The approximate posterior distribution is often optimized using an Evidence Lower Bound (ELBO) loss. Many methods (see e.g. [170, 42, 328]) are interested in the posterior distribution over the model weights, i.e. $\mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$ where $\boldsymbol{\phi}$ are the model weights. Thus, similarly to ensembles, uncertainty estimates can be obtained by sampling from the posterior distribution on the weights and computing the variance of their predictions. These methods generally needs to trade off the expressivity of the posterior distribution with computational complexity. To approximate the posterior distribution over the high dimensional weights, many works proposed to use techniques like mean-field approximations [170, 42, 328], normalizing flows [358, 263], or matrix decomposition with low rank or Kronecker products [289, 22, 466]. In contrast with methods which are interested in posterior distributions $\mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$ over model weights $\boldsymbol{\phi}$, we present in this thesis variational methods which approximate the posterior distribution $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{x})$ over the target distribution parameters $\boldsymbol{\theta}$ (see Chapters 3 and 4).

Monte Carlo Dropout. Monte Carlo (MC) dropout consists in randomly dropping neurons to form different models $f_{\boldsymbol{\phi}_1}, \cdot, f_{\boldsymbol{\phi}_K}$ and combine their predictions [141] at both training and testing time. This approach can be considered as an ensemble of implicit models [4] but also as performing variational inference [141]. Hence, similarly to ensembles, the variance of the predictions can be viewed as Bayesian uncertainty estimates the member weights are sampled from the posterior weight distribution, i.e. $\boldsymbol{\phi}_k \sim \mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$. Many works extended this approach for e.g. convolutional layers [322] or dropping connection instead of activations [424].

Markov Chain Monte Carlo. Markov Chain Monte Carlo (MCMC) methods, consists in building a Markov Chain which will converge to samples from the posterior distribution. A key example is the Metropolis-Hastings algorithm which iteratively draw a sample from a transition rule and then decide to accept or reject it [360]. Many works focus on sampling from the approximate posterior distribution $\mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$ over model weights $\boldsymbol{\phi}$ (see e.g. [112, 432]). In this case, the Markov Chain builds a sequence $\boldsymbol{\phi}, \cdot, \boldsymbol{\phi}_t$ by following e.g. Hamiltonian dynamics [112] or Stochastic Gradient Langevin Dynamics [432, 148, 267] which converge to sample from the posterior distribution, i.e. $\boldsymbol{\phi}_t \sim \mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$ when $t \to \infty$. On one hand, while full batch MCMC has managed to scale to modern tasks and models, they often come at a heavy computational cost [206]. On the other hand, while stochastic gradient MCMC improve the computational efficiency [432, 148, 267], it might introduce bias in the stationary distribution by omitting the Metropolis-Hasting rejection or subsampling the data [34, 206].

Laplace Approximation. The Laplace approximation consists in approximating the posterior distribution with a Gaussian distribution centered around a mode of the posterior distribution. Many works focus on approximating the posterior distribution $\mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D})$ over model weights $\boldsymbol{\phi}$. In this case the Laplace approximation gives:

$$\mathbb{Q}(\boldsymbol{\phi} \mid \mathcal{D}) \approx \mathcal{N}(\boldsymbol{\phi}_{\text{MAP}}, \boldsymbol{\Sigma})$$

where $\boldsymbol{\phi}_{\text{MAP}}$ is the maximum a posteriori estimate, and $\boldsymbol{\Sigma} = -(\nabla^2_{\boldsymbol{\phi}} \mathcal{L}(\mathcal{D}; \boldsymbol{\phi})|_{\boldsymbol{\phi}_{\text{MAP}}})^{-1}$ is the Laplace approximation of the covariance based on the Hessian of the loss. The posterior distribution can be defined on all model weights, subnetworks [100], or only the last layer [237]. The Hessian computation can be costly and can be approximated using the different techniques like the Fisher information matrix, the generalized Gauss-Newton matrix, diagonal factorization, Kronecker-factored approximate curvature, or low-rank approximation [99]. Finally, the computation of the predictive distribution needs further approximation like linearizing the neural network or using Monte-Carlo approximation [99].

Gaussian Process. Gaussian Processes (GPs) are a family of Bayesian methods which, given a dataset $\mathcal{D}$, associates to each new input $\boldsymbol{x}$ a predictive Gaussian distribution over the output $y \sim \mathbb{P}(y \mid \mathcal{D}, \boldsymbol{x}) = \mathcal{N}(m(\boldsymbol{x}), \sigma^2(\boldsymbol{x}))$. The mean function $m(\boldsymbol{x})$ and the variance function $\sigma^2(\boldsymbol{x})$ depends on the kernel function $\kappa(.,.)$ which encodes the similarity between data samples. Intuitively, uncertainty estimates can be obtained by computing the variance or the entropy of the predicted Gaussian distribution. On one hand, standard GPs have computational and storage limitations [208]. E.g. the computation of the variance function is cubic in the number of data samples which does not scale well to large datasets. To mitigate this issue, previous works proposed sparse Gaussian process which introduce variational pseudo-points can be optimized with stochastic gradient descent (see e.g. [393, 412, 421]). On the other hand, standard GPs does not naturally extract hierarchical representations from structured data. To mitigate this issue, previous works proposed e.g. to use multilayer GPs [95] or deep neural networks as the kernel function [442]. We refer to existing survey on Gaussian process in deep learning for further details [356, 95, 208].

Evidential. Evidential methods are derived from the theory of evidence which can be seen as generalization of the Bayesian theory to subjective probables [102, 376]. Instead of predicting directly the parameters $\boldsymbol{\theta}$ of the target distribution $\mathbb{P}(y \mid \boldsymbol{\theta})$, evidential methods consist in predicting the parameters of the distribution $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{x})$ defined over the parameters $\boldsymbol{\theta}$, thus following the factorization in Eq. (2.6). In this case, uncertainty estimates can be obtained by computing the variance or the entropy of the predicted target or evidential distributions. This family of models is generally effective since it only requires a single forward pass for uncertainty estimation. To improve performances of evidential models, other works proposed to use OOD data during training [270, 273], knowledge distillation [277], or contrastive learning [375]. In this thesis, we introduce a

new family of evidential models called *Posterior networks* with a clear Bayesian interpretation and low practical overhead (see Chapters 3, 4, 6 and 8 to 10). We refer to existing survey on evidential deep learning for uncertainty quantification for further details [419].

Calibration. Calibration metrics consists at evaluating if the probabilities predicted by a model correspond to the true probabilities of the model to be correct (see Section 2.3.1 for further details). Hence, in order to achieve high performance w.r.t. calibration metrics, calibration methods aim at predicting probabilities which are good approximations of their true probability of correctness. Beyond uncertainty-aware models which are expected to be well-calibrated, we distinguish between two other groups of calibration methods: during-training calibration methods and post-training calibration methods. During-training calibration methods apply regularization techniques in the loss objective to create inherently calibrated models (see e.g. [252, 90, 288]). Post-training calibration methods recalibrate the model predictions after training (see e.g. [175, 436]). While these methods can adapt well to different architectures types, they usually require additional held-out validation data and assume that validation and test distribution are similar. Prominent classes of post-training calibration method are histogram binning [458], Temperature scaling [175], isotonic calibration [459], Dirichlet calibration [243], or conformal predictions [13, 279].

Density-based models. This family of models assigned to every data sample $\boldsymbol{x}$ a probability density estimate $\mathbb{P}(\boldsymbol{x} \mid \boldsymbol{\phi})$. In this case, a data sample considered uncertain by the model should be assigned a low density value while a data sample considered certain for the model should be assigned a high density value. The density estimator can be of different nature like a mixture of Gaussian [253, 110] or a normalizing flow [232, 224, 345]. The choice of space on which the density estimator operates is crucial. While density estimation on the input space might be difficult due to the curse of dimensionality [77, 307, 308], other works [67, 224, 302, 443] improved the performance of density-based methods on uncertainty tasks by leveraging a task-induced bias or low-dimensional statistics. In particular, density-based models methods have achieved impressive success in OOD detection tasks [453].

Energy-based models. Energy-based models (EBMs) associate to every combination of input $\boldsymbol{x}$ and output y, a scalar energy value $E_{\boldsymbol{\phi}}(y, \boldsymbol{x})$ [250]. Interestingly, EBMs can often be viewed as density-based models. Indeed, the energy function $E_{\boldsymbol{\phi}}(y, \boldsymbol{x})$ can be transformed into Gibbs distributions $\mathbb{P}(y \mid \boldsymbol{\phi}, \boldsymbol{x})$, $\mathbb{P}(y, \boldsymbol{x} \mid \boldsymbol{\phi})$, or even $\mathbb{P}(\boldsymbol{x} \mid \boldsymbol{\phi})$ given some integrations constraints on $E_{\boldsymbol{\phi}}(y, \boldsymbol{x})$, thus assigning uncertainty estimates on for different combinations of variables $y, \boldsymbol{x}$ [167]. In this case, low predicted energy values correspond to high uncertainty estimates. EBMs are flexible models capable to achieve great performances in OOD detection in many tasks [259, 429] as long as they can capture semantic features of the data [120].

Distance-based models. The core idea of distance-based methods is that the model should assign high uncertainty to testing data samples which are far away from training data samples. Hence, distance-based models can e.g. use distance to class centroids [296], nearest-neighbor [401], or inducing points [421] to estimate uncertainty. Further, they can also use Mahalanobis distance [296], euclidean distance [200], or geodesic distance [162]. Similarly to density-based models, distance-based models can also achieve great performance in OOD detection tasks [453]. Further, distance awareness has been shown to be an important component for uncertainty estimation tasks [247, 421].

2.3 Uncertainty metrics

While ML models are primarily expected to provide accurate predictions, we present in this section an exhaustive summary of the main metrics used to evaluate the quality of uncertainty estimation. It covers correct/wrong predictions detection, OOD & dataset shifts detection, calibration, and sample efficiency. We provide a collection of evaluation setups covering various tasks to benchmark uncertainty models in Table 2.2. Beyond these metrics, note that Bayesian methods have been also used in other tasks like model pruning, model selection, or hyper-parameter tuning [42, 99].

Uncertainty metrics	Existing evaluation setups	Practical reason
Uncertainty calibration	[335, 81, 305, 414]	*Fairness*, *Trust*
Correct/wrong pred. detection	[235, 184, 278, 414]	*Trust*
OOD detection	[235, 71, 454, 59, 223, 184, 71, 414]	*Safety*
Robustness to dataset shifts	[235, 233, 314, 278, 183, 406, 335, 92, 71, 414]	*Maintenance*
Sample efficiency	[71, 194, 256, 14, 414]	*Development, Maintenance*

Table 2.2: Overview of metrics for uncertainty estimation. We relate each of the uncertainty metric to existing evaluation setups and practical reason for uncertainty estimation presented in Section 1.1

2.3.1 Uncertainty Calibration

It is crucial to provide confidence intervals accurately reflecting the true chance of an event to happen. This allows to increase *fairness* and *trust* of the ML prediction even on under-represented data regions. Intuitively, if the model predict 80% chance for a class to be the correct one, we would expect the model to be 80% of the time correct. Hence, uncertainty estimates are important to answer the following practical question:

Do probabilities predicted by ML models correspond to the true probabilities?

In practice, the confidence intervals provided by the models can be used to estimate risks when making decisions. Appropriate metrics to evaluate calibration involve

(strictly) proper scoring rules like Brier scores for classification and quantiles scores for regression [161].

2.3.2 Correct & Wrong Predictions

It is also crucial to detect when ML models are likely to provide correct or wrong predictions. This allows to increase *trust* in the model predictions, especially when the predictions are used to make important decisions. Intuitively, while the model predictions should be accurate, uncertainty estimates should also be good indicators of the prediction errors. Indeed, high uncertainty should indicate likely wrong prediction while low uncertainty should indicate likely correct predictions. Hence, uncertainty estimates are important to answer the following practical question:

Can we detect prediction errors of ML models?

In practice, each application would require to set a threshold on the uncertainty estimates. Ideally, while predictions associated with uncertainty estimates below this threshold should be correct, the predictions associated with uncertainty estimates above this threshold should be wrong. Hence, we can use evaluation metrics which compare scores (i.e. uncertainty estimates) with binary classes (i.e. correct/wrong predictions). Common metrics are based on false and true positive and negative rates given a specific threshold like precision, recall, or F1 score [347]. However, these metrics have the important limitation to depend on a specific choice of threshold. Instead, there exist other evaluations like receiving operator curves (ROC) and precision-recall curves (PR) which can compare the prediction correctness and the predicted uncertainty scores for any choice of threshold. In particular, the area under the ROC curve (AUC-ROC) and the area under the PR curve (AUC-PR) are appropriate metrics to evaluate the overall performance of the uncertainty scores independently of the choice of threshold [98]. All the data used for the evaluation of the correct/wrong predictions should be relevant to the task meaning that every input has a corresponding output label. This requirement contrasts data used for out-of-distribution detection data where inputs might not have corresponding labels.

2.3.3 Out-Of-Distribution

It is crucial to detect when incoming data are anomalous to increase the *safety* of model predictions. The anomalous data are often considered out-of-distribution (OOD) in contrast with normal data similar to data observed during training which are considered in-distribution (ID). Intuitively, uncertainty estimates should be good indicators of anomalous data. Indeed, high uncertainty should indicate likely abnormal data while low uncertainty should indicate likely normal data. Hence, uncertainty estimates are important to answer the following practical question:

Can uncertainty estimates detect anomalous data?

Similarly to the detection of correct and wrong predictions, the detection of anomalous data would also require to set a threshold on the uncertainty estimates. In this case, while predictions associated with uncertainty estimates below this threshold should be normal ID data, predictions associated with uncertainty estimates above this threshold should ideally be abnormal OOD data. Hence, we can also use evaluation metrics like precision, recall, AUC-ROC, and AUC-PR which compare scores (i.e. uncertainty estimates) with binary classes (i.e. ID/OOD data). While ID data should be relevant to the task (i.e. ID inputs have output labels), the OOD data should be clear anomalies (e.g. OOD data come from a different dataset) and might not be relevant to the task (e.g. OOD data are noisy inputs without output labels).

2.3.4 Dataset Shifts

It is crucial to detect and be robust against shifts in the data to primarily increase the ease of *maintenance* of ML models. Intuitively, while the predictions should be robust to dataset shifts, the uncertainty estimates should increase under dataset shifts. Hence, uncertainty estimates are able to indicate when the incoming data drifts away from the training data before that the model breaks. Hence, uncertainty estimates are important to answer the following practical question:

Are model predictions robust to data shift?

In practice, the model would require to *jointly* look at the evolution of the accuracy and the uncertainty estimates under different magnitudes of perturbations. Ideally, while the model should maintain high accuracy on shifted data (a.k.a. OOD generalization performance [385]), the uncertainty estimates of the model should increase on shifted data (a.k.a. OOD detection performance [453]). Further, other expectations on the predictions under dataset shifts involve maintaining good calibration [335], high correct/wrong prediction detection performance (see Chapter 6), and high OOD detection performances (see Chapter 6). In this case, the shifted dataset is still relevant to the original task (i.e. inputs in the shifted dataset have output labels) but differ from the original ID dataset. We distinguish between natural and adversarial dataset shifts. Natural shifts correspond to natural perturbations which could occur in real world scenarios like time shifts [233, 314, 278], location shifts [233, 314, 278], or corrupted data [183, 406]. In contrast, adversarial perturbations shifts actions correspond to the worst-case scenario where the perturbations are designed to fool the model.

2.3.5 Sample efficiency

It is crucial to wisely select data samples to learn efficiently while avoiding failures. This allows to ease the *development* and *maintenance* of the ML models by reducing training time, number of model failures, or enabling to continually learn from the environment. Intuitively, samples with high confidence might not be interesting since already learned, while samples with too high uncertainty might not be irrelevant outliers Hence, uncertainty estimates are important to answer the following practical question:

How can uncertainty estimates efficiently select data samples to learn from?

In practice, the sample selection process is particularly relevant in reinforcement learning (see Chapter 10), active learning (e.g. [143, 227]), continual learning [194, 256], or few-shots learning [14]. Appropriate metrics would be e.g. to measure the training speed by counting the required number of training samples or the time to train.

Part II

Uncertainty Estimation for Independent Data

3 Uncertainty Estimation for Classification

To know what you know and what you do not know, that is true knowledge.

Confucius

The only true wisdom is in knowing you know nothing.

Socrates

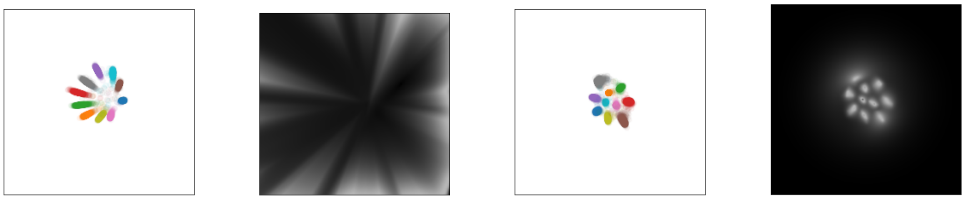
3.1 Introduction

In Chapters 1 and 2, we have reviewed the motivation and background for uncertainty estimation for independent and non-independent data in ML. In this part, we first focus on uncertainty estimation on independent data (e.g. tabular or images). To this end, we start in this chapter to focus on uncertainty estimation for the well-established classification tasks. Despite the necessity for accurate uncertainty estimation, traditional neural networks for classification tasks cannot distinguish between aleatoric and epistemic uncertainty and show overconfident predictions, even for data that is significantly different from the training data [246] [175].

As discussed in Section 2.2, multiple uncertainty methods have been proposed to mitigate this problem. In particular, many methods like ensembles, variational Inference, MC Dropout, MCMC have demonstrated remarkable performance but only describe implicit distributions for predictions which require a costly sampling phase for uncertainty estimation.

Recently, a new class of evidential models aim to directly predict the parameters of a prior distribution on the categorical probability predictions, accounting for the different types of uncertainty [270, 273, 374, 36]. However, these methods require (i) the definition of arbitrary target prior distributions [270, 273, 374], and most importantly, (ii) out-of-distribution (OOD) samples during training time, which is an unrealistic assumption in most applications [270, 273].

Classically, these models would use both ID and OOD samples (e.g. MNIST and FashionMNIST) during training to detect similar OOD samples (i.e. FashionMNIST) at inference time. We show that preventing access to explicit OOD data during training leads to poor results using these approaches (see Fig. 3.1 for MNIST; or Appendix A.4



(a) Data labels - PriorNet **(b)** Uncertainty - PriorNet **(c)** Data labels - PostNet **(d)** Uncertainty - PostNet

Figure 3.1: PriorNet has a pre-ultimate layer of dimension 2 and was trained with Reverse KL and uniform noise on $[0, 255]^{28\times28}$ as OOD data. PostNet has a latent space of dimension 2 and was trained without OOD data. (a) and (c) show the learned latent positions of data with colored labels. (b) and (d) show uncertainty estimates in the latent spaces where darker regions indicate high uncertainty. PostNet correctly assigns high uncertainty to OOD regions contrary to PriorNet.

for toy datasets). Contrary to the expected results, these models produce increasingly confident predictions for samples far from observed data.

Contributions. In contrast, we propose in this chapter, Posterior Network (PostNet), which assigns high epistemic uncertainty to out-of-distribution samples, low overall uncertainty to regions nearby observed data of a single class, and high aleatoric and low epistemic uncertainty to regions nearby observed data of different classes. PostNet uses normalizing flows to learn a distribution over Dirichlet parameters in latent space. We enforce the densities of the individual classes to integrate to the number of training samples in that class, which matches well with the intuition of Dirichlet parameters corresponding to the number of observations per class. PostNet does not require any OOD samples for training, the (arbitrary) specification of target prior distributions, or costly sampling for uncertainty estimation at test time.

3.2 Posterior Network

In classification, we can distinguish between two types of uncertainty for a given input $\boldsymbol{x}^{(i)}$: the uncertainty on the class prediction $y^{(i)} \in \{1, \dots, C\}$ (i.e. aleatoric unceratinty), and the uncertainty on the categorical distribution prediction $\boldsymbol{p}^{(i)} = [p_1^{(i)}, \dots, p_C^{(i)}]$ (i.e. epistemic uncertainty). A convenient way to model both is to describe the *epistemic distribution* $q^{(i)}$ of the categorical distribution prediction $\boldsymbol{p}^{(i)}$, i.e. $\boldsymbol{p}^{(i)} \sim q^{(i)}$. From the epistemic distribution follows naturally an estimate of the *aleatoric distribution* of the class prediction $y^{(i)} \sim \text{Cat}(\bar{\boldsymbol{p}}^{(i)})$ where $\mathbb{E}_{q^{(i)}}[\boldsymbol{p}^{(i)}] = \bar{\boldsymbol{p}}^{(i)}$.

Approaches like ensembles [246] and dropout [141] model $q^{(i)}$ implicitly, which only allows them to estimate statistics at the cost of S samples (e.g. $\mathbb{E}_{q^{(i)}}[\boldsymbol{p}^{(i)}] \approx \frac{1}{S}\sum_{s=1}^{S} \tilde{\boldsymbol{p}}^{(i)}$ where $\tilde{\boldsymbol{p}}^{(i)}$ is sampled from $q^{(i)}$). Another class of models [270, 273, 36, 374] explicitly parametrizes the epistemic distribution with a Dirichlet distribution (i.e. $q^{(i)} = \text{Dir}(\boldsymbol{\alpha}^{(i)})$ where $f_\theta(x^{(i)}) = \boldsymbol{\alpha}^{(i)} \in \mathbb{R}_+^C$), which is the natural prior for categorical distributions. This

parametrization is convenient since it requires only one pass to compute epistemic distribution, aleatoric distribution and class prediction:

$$q^{(i)} = \text{Dir}(\boldsymbol{\alpha}^{(i)}), \qquad \bar{p}_c^{(i)} = \frac{\alpha_c}{\alpha_0} \text{ with } \alpha_0 = \sum_{c=1}^{C} \alpha_c, \qquad y^{(i)} = \arg\max\, [\bar{p}_1, ..., \bar{p}_C] \tag{3.1}$$

The concentration parameters $\alpha_c^{(i)}$ can be interpreted as the number of observed samples of class c and, thus, are a good indicator of epistemic uncertainty for non-degenerate Dirichlet distributions (i.e. $\alpha_c^{(i)} \geq 1$). To learn these parameters, Prior Networks [270, 273] use OOD samples for training and define different target values for ID and OOD data. For ID data, $\alpha_c^{(i)}$ is set to an arbitrary, large number if c is the correct class and 1 otherwise. For OOD data, $\alpha_c^{(i)}$ is set to 1 for all classes.

This approach has four issues: **(1)** The knowledge of OOD data for training is unrealistic. In practice, we might not have these data, since OOD samples are by definition not likely to be observed. **(2)** Discriminating in- from out-of-distribution data by providing an explicit set of OOD samples is hopeless. Since any data not from the data distribution is OOD, it is therefore impossible to characterize the infinitely large OOD distribution with an explicit data set. **(3)** The predicted Dirichlet parameters can take any value, especially for new OOD samples which were not seen during training. In the same way, the sum of the total fictitious prior observations over the full input domain $\int \alpha_0(\boldsymbol{x})d\boldsymbol{x}$ is not bounded and in particular can be much larger than the number of ground-truth observations N. This can result in undesired behavior and assign arbitrarily high epistemic certainty for OOD data not seen during training. **(4)** Besides producing such arbitrarily high prior confidence, PNs can also produce degenerate concentration parameters (i.e. $\alpha_c < 1$). While [273] tried to fix this issue by using a different loss, nothing intrinsically prevents Prior Networks from predicting degenerate prior distributions. In the following section we describe how Posterior Network solves these drawbacks.

3.2.1 An input-dependent Bayesian posterior

First, recall the Bayesian update of a single categorical distribution $y \sim \text{Cat}(\boldsymbol{p})$. It consists in (1) introducing a prior Dirichlet distribution over its parameters i.e. $\mathbb{P}(\boldsymbol{p}) = \text{Dir}(\boldsymbol{\beta}^{\text{prior}})$ where $\boldsymbol{\beta}^{\text{prior}} \in \mathbb{R}_+^C$, and (2) using N given observations $y^{(1)}, ..., y^{(N)}$ to form the posterior distribution $\mathbb{P}(\boldsymbol{p}|\{y^{(j)}\}_{j=1}^N) = \text{Dir}(\boldsymbol{\beta}^{\text{prior}} + \boldsymbol{\beta}^{\text{data}})$ where $\beta_c^{\text{data}} = \sum_j \mathbb{1}_{y^{(j)}=c}$ are the class counts. That is, the Bayesian update is

$$\mathbb{P}(\boldsymbol{p}|\{y^{(j)}\}_{j=1}^n) \propto \mathbb{P}(\{y^{(j)}\}_{j=1}^n|\boldsymbol{p}) \times \mathbb{P}(\boldsymbol{p}). \tag{3.2}$$

Observing no data (i.e. $\beta_c^{\text{data}} \to 0$) would lead to flat categorical distribution (i.e. $p_c = \beta_c^{\text{prior}} \cdot (\sum_{c'} \beta_{c'}^{\text{prior}})^{-1}$), while observing many samples (i.e. β_c^{data} is large) would converge to the true data distribution (i.e. $p_c \approx \frac{\beta_c}{\sum_i \beta_i}$). Furthermore, we remark that N behaves like a certainty budget distributed over all classes i.e. $N = \sum_c \beta_c^{\text{data}}$.

Classification is more complex. Generally, we predict the class label $y^{(i)}$ from a different categorical distribution $\text{Cat}(\boldsymbol{p}^{(i)})$ for each input $\boldsymbol{x}^{(i)}$. PostNet extends the Bayesian

treatment of a single categorical distribution to classification by predicting an individual posterior update for any possible input. To this end, it distinguishes between a fixed prior parameter $\boldsymbol{\beta}^{\text{prior}}$ and the additional learned (pseudo) counts $\boldsymbol{\beta}^{(i)}$ to form the parameters of the posterior Dirichlet distribution $\boldsymbol{\alpha}^{(i)} = \boldsymbol{\beta}^{\text{prior}} + \boldsymbol{\beta}^{(i)}$. Hence, PostNet's posterior update is equivalent to predicting a set of pseudo observations $\{\tilde{y}^{(j)}\}_j^{(i)}$ per input $\boldsymbol{x}^{(i)}$, accordingly $\beta_c^{(i)} = \sum_j \mathbb{1}_{\tilde{y}^{(j)}=c}$ and

$$\mathbb{P}(\boldsymbol{p}^{(i)}|\{\tilde{y}^{(j)}\}_j^{(i)}) \propto \mathbb{P}(\{\tilde{y}^{(j)}\}_j^{(i)}|\boldsymbol{p}^{(i)}) \times \mathbb{P}(\boldsymbol{p}^{(i)}). \tag{3.3}$$

In practice, we set $\boldsymbol{\beta}^{\text{prior}} = \mathbf{1}$ leading to a flat equiprobable prior when the model brings no additional evidence, i.e. when $\boldsymbol{\beta}^{(i)} = \mathbf{0}$.

The parametrization of $\beta_c^{(i)}$ is crucial and based on two main components. The first component is an encoder neural network, f_θ that maps a data point $\boldsymbol{x}^{(i)}$ onto a low-dimensional latent vector $\mathbf{z}^{(i)} = f_\theta(\boldsymbol{x}^{(i)}) \in \mathbb{R}^H$. The second component is to learn a *normalized* probability density $\mathbb{P}(\mathbf{z}|c;\phi)$ per class on this latent space; intuitively acting as class conditionals in the latent space. Given these and the number of ground-truth observations N_c in class c, we define:

$$\beta_c^{(i)} = N_c \cdot \mathbb{P}(\mathbf{z}^{(i)}|c;\phi) = N \cdot \mathbb{P}(\mathbf{z}^{(i)}|c;\phi) \cdot \mathbb{P}(c), \tag{3.4}$$

which corresponds to the number of (pseudo) observations of class c at $\mathbf{z}^{(i)}$. Note that it is crucial that $\mathbb{P}(\mathbf{z}|c;\phi)$ corresponds to a proper normalized density function since this will ensure the model's epistemic uncertainty to increase outside the known distribution. Indeed, the core idea of our approach is to parameterize these distributions by a flexible, yet tractable family: normalizing flows (e.g. radial flow [358] or IAF [221]). Note that normalizing flows are theoretically capable of modeling any continuous distribution given an expressive and deep enough model [199, 221].

In practice, we observed that various architectures can be used for the encoder. Also, similarly to GAN training [352], we observed that adding a batch normalization after the encoder made the training more stable. It facilitates the match between the latent positions output by the encoder and non-zero density regions learned by the normalizing flows. Remark that we can theoretically use any density estimator on the latent space. We experimentally compared Mixtures of Gaussians (MoG), radial flow [358] and IAF [221]. While all density types performed reasonably well (see Table 3.1 and Appendix A.4), we observed a better performance of flow-based density estimation in general. We decided to use radial flow for its good trade-off between flexibility, stability, and compactness (only few parameters).

Model discussion. The Eq. (3.4) exhibits a set of interesting properties which ensure reasonable uncertainty estimates for ID and OOD samples. To highlight the properties of the Dirichlet distributions learned by PostNet, we assume in this paragraph that we have a fixed encoder f_θ and normalizing flow model parameterized by ϕ. Writing the mean of the Dirichlet distribution parametrized by Eq. (3.4) and using Bayes' theorem gives:

$$\mathbb{E}_{\boldsymbol{p}\sim\text{Dir}(\alpha^{(i)})}[p_c] = \frac{\beta_c^{\text{prior}} + N \cdot \mathbb{P}(c|\mathbf{z}^{(i)};\phi) \cdot \mathbb{P}(\mathbf{z}^{(i)};\phi)}{\sum_c \beta_c^{\text{prior}} + N \cdot \mathbb{P}(\mathbf{z}^{(i)};\phi)} \tag{3.5}$$

For very likely in-distribution data (i.e. $\mathbb{P}(\mathbf{z}^{(i)};\phi) \to \infty$), the aleatoric distribution estimate $\bar{\boldsymbol{p}}^{(i)} = \mathbb{E}_{\boldsymbol{p}\sim\text{Dir}(\alpha^{(i)})}[p_c]$ converges to the true categorical distribution $\mathbb{P}(c|\mathbf{z}^{(i)};\phi)$. Thus, predictions are more accurate and calibrated for likely samples. Conversely, for out-of-distribution samples (i.e. $\mathbb{P}(\mathbf{z}^{(i)};\phi) \to 0$), the aleatoric distribution estimate $\bar{\boldsymbol{p}}^{(i)} = \mathbb{E}_{\boldsymbol{p}\sim\text{Dir}(\alpha^{(i)})}[p_c]$ converges to the flat prior distribution (e.g. $p_c = \frac{1}{C}$ if $\boldsymbol{\beta}^{prior} = \mathbf{1}$). In the same way, we show in the appendix that the covariance of the epistemic distribution converges to $\mathbf{0}$ for very likely in-distribution data, meaning no epistemic uncertainty. Thus, uncertainty for in-distribution data is reduced to the (inherent) aleatoric uncertainty and zero epistemic uncertainty. Similarly to the single categorical distribution case, increasing the training dataset size (i.e. $N \to \infty$) also leads to the mean prediction $\mathbb{E}_{\boldsymbol{p}\sim\text{Dir}(\alpha)}[p_c]$ converging to the class posterior $\mathbb{P}(c|\mathbf{z}^{(i)};\phi)$. On the contrary, no observed data (i.e $N = 0$) again leads to the model reverting to a flat prior.

Posterior Network also handles limited certainty budgets at different levels. At the sample level, the certainty budget $\alpha_0^{(i)} = \sum_c \alpha_c^{(i)}$ is distributed over classes. At the class level, the certainty budget $N_c = \int N_c\, \mathbb{P}(\mathbf{z}|c;\phi)\, d\mathbf{z} = N_c \int \mathbb{P}(\mathbf{z}|c;\phi)\, d\mathbf{z}$ is distributed over samples. At the dataset level, the certainty budget $N = \sum_c \int N_c\, \mathbb{P}(\mathbf{z}|c;\phi)\, d\mathbf{z}$ is distributed over classes and samples. Regions of latent space with many training examples are assigned high density $\mathbb{P}(\mathbf{z}|c;\phi)$, forcing low density elsewhere to fulfill the integration constraint. Consequently, density estimation using normalizing flows enables PostNet to learn out-of-distribution uncertainty by observing only in-distribution data.

Overview. In Fig. 3.2 we provide an overview of Posterior Network. We have three example inputs, $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$, and $\boldsymbol{x}^{(3)}$, which are mapped onto their respective latent space coordinates $\mathbf{z}^{(i)}$ by the encoding neural network f_θ. The normalizing flow component learns flexible (normalized) density functions $\mathbb{P}(\mathbf{z}|c;\phi)$, for which we evaluate their densities at the positions of the latent vectors $\mathbf{z}^{(i)}$. These densities are used to parameterize a Dirichlet distribution for each data point, as seen on the right hand side. Higher densities correspond to higher confidence in the Dirichlet distributions – we can observe that the out-of-distribution sample $\boldsymbol{x}^{(3)}$ is mapped to a point with (almost) no density, and hence its predicted Dirichlet distribution has very high epistemic uncertainty. On the other hand, $\boldsymbol{x}^{(2)}$ is an ambiguous example that could depict either the digit 0 or 6. This is reflected in its corresponding Dirichlet distribution, which has high aleatoric uncertainty (as the sample is ambiguous), but low epistemic uncertainty (since it is from the distribution of hand-drawn digits). The unambiguous sample $\boldsymbol{x}^{(1)}$ has low overall uncertainty.

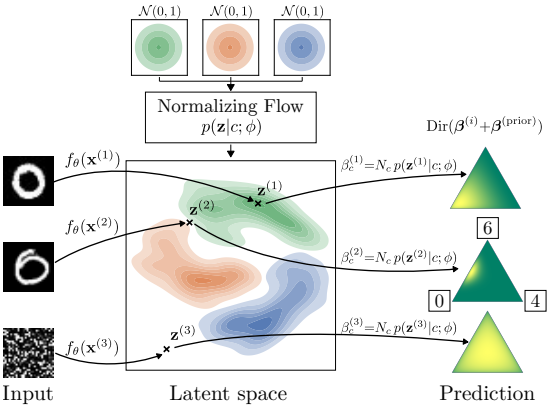


Figure 3.2: Overview of Posterior Network.

Lastly, since both the encoder network f_θ and the normalizing flow parameterized by ϕ are fully differentiable, we can learn their parameters jointly in an end-to-end fashion. We do this via a novel loss defined in Section 3.3 which emerges from Bayesian learning principles [378] and is related to UCE [36].

3.2.2 Density estimation for OOD detection

Normalized densities, as used by PostNet, are well suited to discriminate between ID data (with high likelihoood) and OOD data (with low likelihood). While it is also possible to learn a normalizing flow model on the input domain directly (e.g., [339, 224, 166]), this is very computationally demanding and might not be necessary for a discriminative model. Futhermore, density estimation is prone to the curse of dimensionality in high dimensional spaces [308, 77]. For example, unsupervised deep generative models like [220] or [422] have been shown to be unable to distinguish between ID and OOD samples in some situations when working on all features directly [307, 167].

To circumvent these issues, PostNet leverages two techniques. First, it uses the full class label information. Thus, PostNet assigns a density per class and regularizes the epistemic uncertainty with the training class counts N_c. Second, PostNet performs density estimation on a low dimensional latent space describing relevant features for the classification task (Fig. 3.1; Fig. 3.2). Hence, PostNet does not suffer from the curse of dimensionality and still enjoys the benefits of a properly normalized density function. Using the inductive bias of a discriminative task and low dimensional latent representations improved OOD detection in [224] as well.

3.3 Uncertainty-Aware Loss Computation

A crucial design choice for neural network learning is the loss function. PostNet estimates both aleatoric and epistemic uncertainty by learning a distribution $q^{(i)}$ for data point i which is close to the true posterior of the categorical distribution $\boldsymbol{p}^{(i)}$ given the training data $\boldsymbol{x}^{(i)}$: $q(\boldsymbol{p}^{(i)}) \simeq \mathbb{P}(\boldsymbol{p}^{(i)}|\boldsymbol{x}^{(i)})$. One way to approximate the posterior is the following Bayesian loss function [38, 378, 462], which has the nice property of being optimal when $q^{(i)}$ is equal to the true posterior distribution:

$$q^* = \underset{q^{(i)} \in \mathcal{P}}{\arg\min}\, \mathbb{E}_{\psi^{(i)} \sim q(\psi^{(i)})}[l(\psi^{(i)}, \boldsymbol{x}^{(i)})] - H(q^{(i)}), \tag{3.6}$$

where l is a generic loss over $\psi^{(i)}$ satisfying $0 < \int \exp(-l(\psi, x))d\psi < \infty$, $\mathcal{P}$ is the family of distributions we consider and $H(q^{(i)})$ denotes the entropy of $q^{(i)}$.

Applied in our case, Posterior Network learns a distribution q from the family of the Dirichlet distributions $\mathrm{Dir}(\boldsymbol{\alpha}^{(i)}) = \mathcal{P}$ over the parameters $\boldsymbol{p}^{(i)} = \psi^{(i)}$. Instantiating the loss l with the cross-entropy loss (CE) we obtain the following optimization objective

$$\min_{\theta,\phi} \mathcal{L} = \min_{\theta,\phi} \frac{1}{N} \sum_i^N \underbrace{\mathbb{E}_{q(\mathbf{p}^{(i)})}[\mathrm{CE}(\mathbf{p}^{(i)}, \boldsymbol{y}^{(i)})]}_{(1)} - \underbrace{H(q^{(i)})}_{(2)} \tag{3.7}$$

where $\mathbf{y}^{(i)}$ corresponds to the one-hot encoded ground-truth class of data point i. Optimizing this loss approximates the true posterior distribution for the the categorical distribution $\boldsymbol{p}^{(i)}$. The first term (1) corresponds the Uncertain Cross Entropy loss (UCE) introduced by [36], which is known to increase confidence for observed data. The second term (2) is an entropy regularizer, which emerges naturally and favors smooth distributions $q^{(i)}$. Here we are optimizing jointly over the neural network parameters θ and ϕ. We also experimented with sequential training i.e. optimizing over the normalizing flow component only with a pre-trained model (see Table 3.2 and Appendix A.4). Once trained, PostNet can predict uncertainty-aware Dirichlet distributions for unseen data points.

Observe that Eq. (3.7) is equivalent to the ELBO loss used in variational inference when using a uniform Dirichlet prior (i.e. $(1) = -\mathbb{E}_{q(\mathbf{p}^{(i)})}[\log \mathbb{P}(\boldsymbol{y}^{(i)}|\mathbf{p}^{(i)})]$ and $(2) = \mathrm{KL}(q^{(i)}||\mathbb{P}(\mathbf{p}^{(i)}))$ where $\mathbb{P}(\mathbf{p}^{(i)}) = \mathrm{Dir}(\mathbf{1})$). The more general Eq. (3.6), however, is not necessarily equal to an ELBO loss.

Another interesting special case is to consider the family of Dirac distributions as $\mathcal{P}$ instead of the family of Dirichlet distributions. In this case we find back the traditional cross-entropy loss, which performs a simple point estimate for the distribution $\boldsymbol{p}^{(i)}$. CE is therefore not suited to learn a distribution with non-zero variance, as explained in [36].

Other approaches, such as dropout, approximate the expectation in UCE by sampling from $q^{(i)}$. Our approach has the advantage of using closed-form expressions both for UCE [36] and the entropy term, thus being efficient and exact. The weight of the entropy regularization is a hyperparameter; experiments have shown PostNet to be fairly insensitive to it, so in our experiments we set it to 10^{-5}.

3.4 Experimental Evaluation

In this section we compare our model to previous methods on a rich set of experiments. The code and further supplementary material is available online (`www.daml.in.tum.de/postnet`).

Baselines. We have special focus on comparing with other models parametrizing Dirichlet distributions. We use Prior Networks (PN) trained with KL divergence (**KL-PN**) [270] and Reverse KL divergence (**RKL-PN**) [273]. These methods assume the knowledge of in- and out-of-distribution samples. For fair evaluation, the actual OOD test data cannot be used; instead, we used uniform noise on the valid domain as OOD training data. Additionally we trained RKL-PN with FashionMNIST as OOD data for MNIST (**RKL-PN w/ F.**). We also compare to Distribution Distillation (**Distill.**) [277], which learns Dirichlet distributions with maximum likelihood by using soft-labels from an ensemble of networks. As further baselines, we compare to dropout models (**Dropout Net**) [141] and ensemble methods (**Ensemble Net**) [246], which are state of the art in many tasks involving uncertainty estimation [335]. Empirical estimates of the mean and variance of $q^{(i)}$ are computed based on the neuron drop probability p_{drop}, and m individually trained networks for ensemble.

All models share the same core architecture using 3 dense layers for tabular data, and 3 conv. + 3 dense layers for image data. Similarly to [270, 273], we also used the VGG16 architecture [389] on CIFAR10. We performed a grid search on p_{drop}, m, learning rate and hidden dimensions, and report results for the best configurations. Results are obtained from 5 trained models with different initializations. Moreoever, for all experiments, we split the data into train, validation and test set (60%, 20%, 20%) and train/evaluate all models on 5 different splits. Besides the mean we also report the standard error of the mean. Further details are given in the appendix.

Datasets. We evaluate on the following real-world datasets: **Segment** [111], **Sensorless Drive** [111], **MNIST** [249] and **CIFAR10** [238]. The former two datasets (Segment and Sensorless Drive) are tabular datasets with dimensionality 18 and 49 and with 7 and 11 classes, respectively. We rescale all inputs between $[0, 1]$ by using the min and max value of each dimension from the training set. Additionally, we compare all models on 2D synthetic data composed of three Gaussians each. Datasets are presented in more detail in the appendix.

Metrics. We follow the method proposed in [335] and evaluate the coherence of confidence, uncertainty calibration and OOD detection. Note that our goal is not to improve accuracy; still we report the numbers in the experiments.

Confidence calibration: We aim to answer '*Are more confident (i.e. less uncertain) predictions more likely to be correct?*'. We use the area under the precision-recall curve (AUC-PR) to measure confidence calibration. For aleatoric confidence calibration (**Alea. Conf.**) we use $\max_c \bar{\boldsymbol{p}}_c^{(i)}$ as the scores with labels 1 for correct and 0 for incorrect predictions. For epistemic confidence calibration (**Epist. Conf.**), we distinguish Dirichlet-based models, and dropout and ensemble models. For the former we use $\max_c \boldsymbol{\alpha}_c^{(i)}$ as scores, and for the latter we use the (inverse) empirical variance $\tilde{\boldsymbol{p}}_c^{(i)}$ of the predicted class, estimated from 10 samples.

Uncertainty calibration: We used Brier score (**Brier**), which is computed as $\frac{1}{N}\sum_i^N \|\bar{\boldsymbol{p}}^{(i)} - \boldsymbol{y}^{(i)}\|_2$, where $\boldsymbol{y}^{(i)}$ is the one-hot encoded ground-truth class of data point i. For Brier score, lower is better.

OOD detection: Our main focus lies on the models' ability to detect out-of-distribution samples. We used AUC-PR to measure performance. For aleatoric OOD detection (**Alea. OOD**), the scores are $\max_c \bar{\boldsymbol{p}}_c^{(i)}$ with labels 1 for ID data and 0 for OOD data. Fo epistemic OOD detection (**Epist. OOD**), the scores for Dirichlet-based models are given by $\alpha_0^{(i)} = \sum_c \alpha_c^{(i)}$, while we use the (inverse) empirical variance $\tilde{\boldsymbol{p}}^{(i)}$ for ensemble and dropout models. To provide a comprehensive overview of OOD detection results we use different types of OOD data as described in the following:

- *Unseen datasets.* We use data from other datasets as OOD data for the image-based models. We use data from FashionMNIST [450] and K-MNIST [83] as OOD data for models trained on MNIST, and data from SVHN [313] as OOD for CIFAR10.

	Acc.	Alea. Conf.	Epist. Conf.	Brier	OOD Alea.	OOD Epist.
Drop Out	89.32±0.2	98.21±0.1	95.24±0.2	28.86±0.4	35.41±0.4	40.61±0.7
Ensemble	99.37±0.0	99.99±0.0	*99.98±0.0	2.47±0.1	50.01±0.0	50.62±0.1
Distill.	93.66±1.5	98.29±0.5	98.15±0.5	44.94±1.4	32.1±0.6	31.17±0.2
KL-PN	94.77±0.9	99.52±0.1	99.47±0.1	21.47±1.9	35.48±0.8	33.2±0.6
RKL-PN	99.42±0.0	99.96±0.0	99.89±0.0	9.07±0.1	45.89±1.6	38.14±0.8
PostN Rad.	98.02±0.1	99.89±0.0	99.47±0.0	5.51±0.2	72.89±0.8	***88.73±0.5**
PostN IAF	***99.52±0.0**	***100.0±0.0**	**99.92±0.0**	***1.43±0.1**	***82.96±0.8**	88.65±0.4

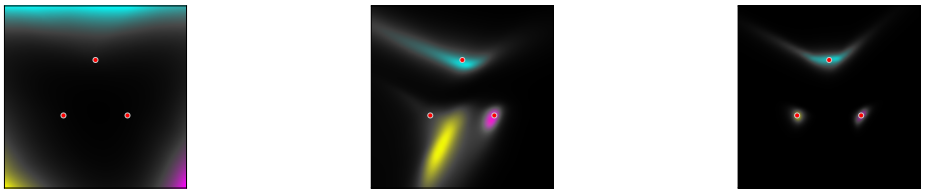
Table 3.1: Results on Sensorless Drive dataset. Bold numbers indicate best score among Dirichlet parametrized models and starred numbers indicate best scores among all models.

- *Left-out classes.* For the tabular datasets (Segment and Sensorless Drive) there are no other datasets that are from the same domain. To simulate OOD data we remove one or more classes from the training data and instead consider them as OOD data. We removed one class (class sky) from the Segment dataset and two classes from Sensorless Drive (class 10 and 11).

- *Out-of-domain.* In this novel evaluation we consider an extreme case of OOD data for which the data comes from different value ranges (**OODom**). E.g., for images we feed unscaled versions in the range $[0, 255]$ instead of scaled versions in $[0, 1]$. We argue that models should easily be able to detect data that is extremely far from the data distribution. However, as it turns out, this is surprisingly difficult for many baseline models.

- *Dataset shifts.* Finally, for CIFAR10, we use 15 different image corruptions at 5 different severity levels [183]. This setting evaluates the models' ability to detect low-quality data (Fig. 3.4b,c).

Results. Results for the Sensorles Drive dataset are shown in Table 3.1. Tables for other datasets are in the appendix. Even without requiring expensive sampling, PostNet performs on par for accuracy and confidence scores with other models, brings a significant improvement for calibration within the Dirichlet-based models, and outperforms all other models by a large margin (more than +30% abs. improvement) for OOD detection. Radial flow and IAF variants both achieve strong performance for all datasets (see Appendix A.4). We use the smaller model (i.e. Radial flow) for comparison in the following. In our experiments, note that using one Radial flow per class represents a small overhead of only 80 parameters per class, which is negligible compared to the encoder architectures (e.g. VGG16 has 138M parameters).



(a) PostNet: No-Flow **(b)** PostNet: No-Bayes-Loss **(c)** PostNet: Complete

Figure 3.3: Uncertainty visualization for a 2D 3-Gaussians dataset. Red dots indicate the Gaussians means. Darker regions indicate high epistemic uncertainty for a class prediction. Ablated models fail even a simple dataset while PostNet shows high certainty around gaussians means only.

	Acc.	Alea. Conf.	Epist. Conf.	Brier	OOD Alea.	OOD Epist.
PostN: No-Flow	55.38±0.7	85.46±0.3	82.58±0.6	64.4±0.6	29.59±0.1	31.15±0.4
PostN: No-Bayes-Loss	96.6±0.2	99.74±0.0	98.68±0.1	8.85±0.4	62.39±1.5	82.63±1.4
PostN: Seq-No-Bn	15.09±1.0	39.88±7.2	39.86±7.2	89.88±1.3	57.19±2.5	56.74±2.4
PostN: Seq-Bn	98.42±0.1	99.92±0.0	98.76±0.1	5.41±0.1	52.35±0.7	71.75±1.9

Table 3.2: Ablation study results on Sensorless Drive dataset. Gray cells indicate significant drops in scores compared to complete PostNet Rad. in Table 3.1.

We performed an ablation study on each component of PostNet to evaluate their individual contributions. We were especially interested in comparing stability and uncertainty estimates. Thus, we removed independently the normalizing flow component (No-Flow) and the novel Bayesian loss (No-Bayes-Loss) replaced by the classic cross-entropy loss. Furthermore, we used pre-trained models and subsequently only trained the normalizing flow component, with or without a batch normalization layer (Seq-Bn and Seq-No-Bn). We report results in Table 3.2. No-Flow has a significant drop in OOD detection scores similarly to Prior Networks; not surprising since they mainly differ by their loss. This underlines the importance of using normalized density estimation to differentiate ID and OOD data. The lower performance of No-Bayes-Loss compared to the original model indicates the benefit of using our Bayesian loss. Seq-Bn obtains good performance for some of the metrics, which as a by-product, allows to estimate uncertainty on pre-trained models. Though, we noticed better performance for joint training in general. As shown by Seq-No-Bn scores, the batch normalization layer brings stability. It intuitively facilitates predicted latent positions to lie on non-zero density regions. Similar conclusions can be drawn on the toy dataset (see Fig. 3.3) and the Segment dataset (see Appendix A.4). We further compare various density types and latent dimensions in appendix. We noticed that a too high latent dimension leads to a performance decrease. We also observed that flow-based density estimation generally achieves better scores.

	OOD K. Alea.	OOD K. Epist.	OOD F. Alea.	OOD F. Epist.	OODom K. Alea.	OODom K. Epist.	OODom F. Alea.	OODom F. Epist.
RKL-PN	60.76±2.9	53.76±3.4	78.45±3.1	72.18±3.6	9.35±0.1	8.94±0.0	9.53±0.1	8.96±0.0
RKL-PN w/ F.	81.34±4.5	78.07±4.8	**100.0±0.0**	**100.0±0.0**	9.24±0.1	9.08±0.1	88.96±4.4	87.49±5.0
PostN	**95.75±0.2**	**94.59±0.3**	97.78±0.2	97.24±0.3	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**

Table 3.3: Results on MNIST for OOD detection against KMNIST (K.) and FashionMNIST (F.). We trained Rev. KL divergence PriorNets with uniform noise (RKL-PN) and Fashion MNIST (RKL-PN w/ F.) as OOD. PostNet requires no OOD data. Larger numbers are better.

Results of the comparison between RKL-PN, RKL-PN w/ F and PostNet for OOD detection on MNIST are shown in Table 3.3. Not surprisingly, the usage of FashionMNIST as OOD data for training helped RKL-PN to detect other FashionMNIST data. Except for FashionMNIST OOD, PostNet still outperforms RKL-PN w/ F. in OOD detection for other datasets. We noticed that tabular datasets, defined on an unbounded input domain, are more difficult for baselines. One explanation is that due to the min/max normalization it can happen that test samples lie outside the interval $[0, 1]$ observed during training. For images, the input domain is compact, which allows to define a valid distribution for OOD data (e.g. uniform) which makes OODom data challenging (see OOD vs OODom in Table 3.3).

	Acc.	Alea. Conf.	Epist. Conf.	Brier	OOD Alea.	OOD Epist.	OODom Alea.	OODom Epist.
Drop Out C.	71.73±0.2	92.18±0.1	84.38±0.3	49.76±0.2	**72.94±0.3**	41.68±0.5	28.3±1.8	47.1±3.3
KL-PN C.	48.84±0.5	78.01±0.6	77.99±0.7	83.11±0.6	59.32±1.1	58.03±0.8	17.79±0.0	20.25±0.2
RKL-PN C.	62.91±0.3	85.62±0.2	81.73±0.2	58.12±0.4	67.07±0.4	56.64±0.8	17.83±0.0	17.76±0.0
PostN C.	**76.46±0.3**	**94.75±0.1**	**94.34±0.1**	**37.39±0.4**	72.83±0.6	**72.82±0.7**	**100.0±0.0**	**100.0±0.0**
Drop Out V.	82.84±0.1	97.15±0.0	96.6±0.0	27.15±0.2	51.39±0.1	53.64±0.1	51.38±0.1	53.66±0.1
KL-PN V.	27.46±1.7	50.61±4.0	52.49±4.2	87.28±1.0	43.96±1.9	43.23±2.3	18.14±0.1	19.12±0.4
RKL-PN V.	64.76±0.3	86.11±0.4	85.59±0.3	54.73±0.4	53.61±1.1	49.37±0.8	29.07±2.1	24.84±1.3
PostN V.	**84.85±0.0**	**97.76±0.0**	**97.25±0.0**	**22.84±0.0**	**80.21±0.2**	**77.71±0.3**	**91.35±0.5**	**99.25±0.1**

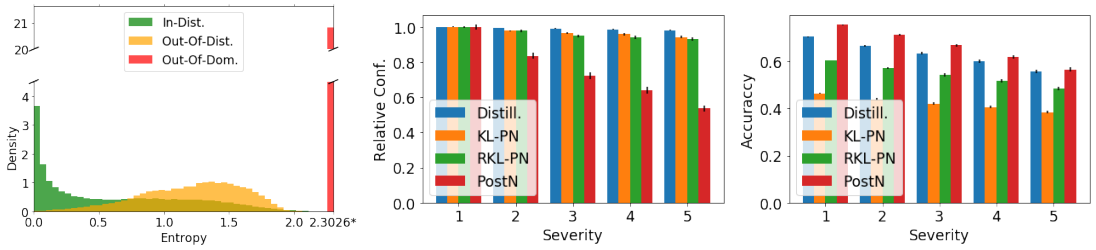
Table 3.4: Results on CIFAR10 with simple convolutional architectures (C.) and VGG16 (V.). Bold numbers indicate best score among one architecture type.

Uncertainty estimation should be good regardless of the model accuracy. It is even more important for less accurate models since they actually *do not know* (i.e. they do more mistakes). Thus, we compared the models that use a single network for training (using a convolutional architecture and VGG16) in Table 3.4. Without the knowledge of true OOD data (SVHN) during training, Prior Networks struggle to achieve good performance. In contrast, PostNet outputs high quality uncertainty estimates regardless of the architecture used for the encoder. We report additional results for PostNet using other encoder architectures (convolutional architecture, AlexNet [239], VGG [389] and ResNet [181]) in Table 3.5. Deep generative models as Glow [220] using density estimation on input space are unable to distinguish between CIFAR10 and SVHN [307]. In contrast, PostNet clearly distinguishes between in-distribution data (CIFAR10) with low entropy, out-of-distribution (OOD SVHN) with high entropy, and close to the maximum possible entropy for out-of-domain data (OODom SVHN) (see Fig. 3.4a). Similar conclusions hold for MNIST and FashionMNIST (see Appendix A.4). Furthermore, results for the image

perturbations on CIFAR10 introduced by [183] are presented in Fig. 3.4. We define the average change in confidence as the ratio between the average confidence $\frac{1}{N}\sum_i^N \alpha_0^{(i)}$ at severity 1 vs other severity levels. As larger shifts correspond to larger differences in the underlying distributions, we expect uncertainty-aware models to become less certain for more severe perturbations. Posterior Network exhibits, as desired, the largest decrease in confidence with stronger corruptions (see Fig. 3.4b) while maintaining a high accuracy (see Fig. 3.4c).

	Acc.	Alea. Conf.	Epist. Conf.	Brier	OOD Alea.	OOD Epist.	OODom Alea.	OODom Epist.
PostNet: Conv.	78.58±0.1	95.45±0.0	93.36±0.0	33.84±0.2	72.21±0.1	57.72±0.7	100.0±0.0	100.0±0.0
PostNet: Alexnet	80.81±0.2	96.33±0.1	95.35±0.1	29.99±0.3	73.4±0.7	67.05±0.6	97.64±0.4	99.64±0.1
PostNet: VGG	84.85±0.0	97.76±0.0	97.25±0.0	22.84±0.0	80.21±0.2	77.71±0.3	91.35±0.5	99.25±0.1
PostNet: Resnet	87.86±0.2	98.35±0.0	97.13±0.0	19.33±0.3	79.92±0.4	72.25±0.6	99.94±0.0	99.94±0.0

Table 3.5: Results of PostNet with different encoder architectures. It shows good uncertainty estimation regardless of the architecture complexity.



(a) ID/OOD/OODom entropy **(b)** Confidence under data shifts **(c)** Accuracy under data shifts

Figure 3.4: (a) shows entropy of the aleatoric distributions predicted by PostNet on CIFAR10 (ID) and SVHN (OOD, OODom). The value 2.3026* denotes the highest achievable entropy for 10 classes. PostNet can easily distinguish between the three data types. (b) and (c) present averaged confidence and accuracy under 15 dataset shifts introduced by [183] on CIFAR10 with conv. architecture. On more severe perturbations (i.e. data further away from data distribution), PostNet assigns higher epistemic uncertainty as desired. Baselines keeps same confidence even for less accurate predictions.

3.5 Conclusion

We propose Posterior Network, a model for uncertainty estimation in classification without requiring out-of-distribution samples for training or costly sampling for uncertainty estimation. PostNet is composed of three main components: an encoder which outputs a position in a latent space, a normalizing flow which performs a density estimation in this latent space, and a Bayesian loss for uncertainty-aware training. In our extensive experimental evaluation, PostNet achieves state-of-the-art performance with a strong improvement for detection of in- and out-of-distribution samples.

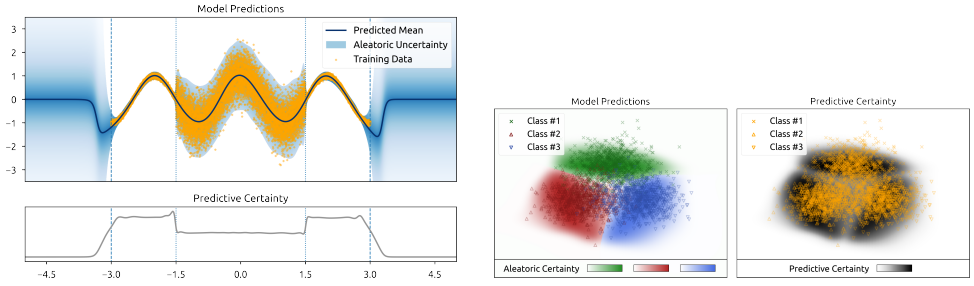
4 Uncertainty Estimation for Regression

Ignorance more frequently begets confidence than does knowledge.

Charles Darwin

4.1 Introduction

In Chapter 3, we have viewed how we can efficiently predict uncertainty estimates for classification tasks on independent data. In this chapter, we now study how to efficiently predict uncertainty estimates for a more diverse set of tasks. More Specifically, beyond classification tasks, we focus on uncertainty estimation for regressions and count prediction on independent data. In these tasks, traditional neural networks are not readily applicable in safety-critical domains as they show overconfident prediction, in particular on data that is different from training data [175, 246]. To mitigate this problem, we have seen the important family of evidential models for uncertainty estimation directly predicts the parameters of a *conjugate prior distribution* on the predicted *target probability distribution*, thus accounting for the different levels of uncertainty. These models are efficient as they only require *a single forward pass* for target and uncertainty prediction. Similarly to PostNet presented in Chapter 3, most of those models focus on classification and thus predict parameters of a Dirichlet distribution [36, 270, 273, 310, 375, 386, 398, 471, 149]. However, only two works [10, 276] have focused on regression by learning parameters of a Normal Inverse-Gamma (NIG) distribution as conjugate prior. Hence, all these models are limited to a *single* task (e.g. either classification or regression). Some approaches



(a) Toy Regression Task **(b)** Toy Classification Task

Figure 4.1: Visualization of the aleatoric and predictive uncertainty estimates of NatPN on two toy regressions and classification tasks. NatPN correctly assigns higher uncertainty to regions far from the training data.

even require out-of-distribution (OOD) data at training time [270, 273] which is an unrealistic assumption in many real-world applications where anomalies are a priori diverse, rare or unknown.

Contributions. We propose Natural Posterior Network (NatPN) as a new approach parametrizing conjugate prior distributions for versatile uncertainty estimation. NatPN is motivated from both the theoretical and practical perspective. **(1)** NatPN can estimate predictive uncertainty for *any* task described by the general group of exponential family distributions contrary to existing approaches from this family of models. Notably, this encompasses very common tasks such as classification, regression and count prediction which can be described with Categorical, Normal and Poisson distributions, respectively. **(2)** In theory, NatPN is based on a *new unified exponential family framework* which performs an input-dependent Bayesian update. For every input, it predicts the parameters of the posterior over the target exponential family distribution. We show that this Bayesian update is *guaranteed* to predict high uncertainty far from training data. **(3)** In practice, NatPN requires *no OOD data for training*, only adds *a single normalizing flow* density to the last predictor layer and provides fast uncertainty estimation in a *single forward pass*. Our extensive experiments showcase the high performances of NatPN for various criteria (accuracy, calibration, OOD and shift detection) and tasks (classification, regression and count prediction). We illustrate the accurate aleatoric and predictive uncertainty predictions of NatPN on two toy examples for classification and regression in Fig. 4.1. *None of the conjugate prior related works* have similar theoretical and practical properties.

4.2 Related Work

In this section, we describe other work related to uncertainty estimation for supervised learning. We refer to [150] for a detailed survey on uncertainty estimation in deep learning.

Sampling-based methods. A first family of models estimates uncertainty by aggregating statistics (e.g. mean and variance) from different samples of an implicit predictive distribution. Examples are ensemble [217, 246, 390, 434, 439] and dropout [141] models which provide high-quality uncertainty estimates [335] at the cost of an expensive sampling phase at inference time. Moreover, ensembles usually require training multiple models. Further, Bayesian neural networks (BNN) [42, 359, 268] model the uncertainty on the weights and also require multiple samples to estimate the uncertainty on the final prediction. While recent BNNs have shown reasonably good performance [114, 327, 128], modelling the distribution on the weights suffers from pathological behavior thus limiting these approaches in practice [135, 170, 206]. In particular, [206] uses an enormous computation budget by parallelizing the computation over 512 TPUv3 devices and running tens of thousands of training epochs to achieve a more exact Bayesian inference which is not suitable for practical applications. In contrast, NatPN predicts uncertainty in *a*

single forward pass with a *closed-form posterior distribution* over the target variable. NatPN *does not* model uncertainty on the weights.

Sampling-free methods. A second family of models is capable of estimating uncertainty in a single forward pass. The family of models parametrizing conjugate prior distributions is the main focus of this chapter [419, 235, 310, 375, 386, 398, 149]. Beyond this family of models, we differentiate between four other families of sampling-free models for uncertainty estimation. A first family aims at learning deep Gaussian processes with random features projections or learned inducing points [247, 421, 420, 36]. A second family aims at learning deep energy-based models [120, 168]. Another family of models aims at propagating uncertainty across layers [425, 343, 384, 149, 185]. They model uncertainty at the weight and/or activation levels and are generally constrained to specific transformations. In contrast, NatPN only models the uncertainty on the predicted target variable and does not enforce any constraint on the encoder architecture. Further, some of the models propagating uncertainty already used the exponential family framework [425, 354]. However, while they parametrize exponential family distributions, NatPN parametrizes the *conjugate prior of the target exponential family distributions* which accounts for the epistemic uncertainty. Finally, while the family of calibration models aims at calibrating predictions [242, 299, 469, 396, 353], NatPN aims at accurately modelling both aleatoric and epistemic uncertainty on in- and out-of-distribution data.

4.3 Natural Posterior Network

At the very core of NatPN stands the Bayesian update rule: $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D}) \propto \mathbb{P}(\mathcal{D} \mid \boldsymbol{\theta}) \times \mathbb{Q}(\boldsymbol{\theta})$ where $\mathbb{P}(\mathcal{D} \mid \boldsymbol{\theta})$ is the target distribution of the target data $\mathcal{D}$ given its parameter $\boldsymbol{\theta}$, and $\mathbb{Q}(\boldsymbol{\theta})$ and $\mathbb{Q}(\boldsymbol{\theta} \mid \mathcal{D})$ are the prior and posterior distributions, respectively, over the target distribution parameters. The target distribution $\mathbb{P}(\mathcal{D} \mid \boldsymbol{\theta})$ could be any likelihood describing the observed target labels. The Bayesian update has three main advantages: **(1)** it introduces a prior belief which represents the safe default prediction if no data is observed, **(2)** it updates the prior prediction based on observed target labels, and **(3)** it assigns a confidence for the new target prediction given the aggregated evidence count of observed target labels. While NatPN is capable to perform a Bayesian update for every possible input given the observed training data, we first recall the Bayesian background for a single exponential family distribution.

Likelihood $\mathbb{P}$	Conjugate Prior $\mathbb{Q}$	Parametrization Mapping m	Bayesian Loss (Eq. (4.5))
$y \sim \text{Cat}(\boldsymbol{p})$	$\boldsymbol{p} \sim \text{Dir}(\boldsymbol{\alpha})$	$\boldsymbol{\chi} = \boldsymbol{\alpha}/n$ $n = \sum_c \alpha_c$	**(i)** $= \psi(\alpha_{y^*}^{(i)}) - \psi(\alpha_0^{(i)})$ **(ii)** $= \log B(\boldsymbol{\alpha}^{(i)}) + (\alpha_0^{(i)} - C)\psi(\alpha_0^{(i)}) - \sum_c (\alpha_c^{(i)} - 1)\psi(\alpha_c^{(i)})$
$y \sim \mathcal{N}(\mu, \sigma)$	$\mu, \sigma \sim \mathcal{N}\Gamma^{-1}(\mu_0, \lambda, \alpha, \beta)$	$\boldsymbol{\chi} = \begin{pmatrix} \mu_0 \\ \mu_0^2 + \frac{2\beta}{n} \end{pmatrix}$ $n = \lambda = 2\alpha$	**(i)** $= \frac{1}{2}\left(-\frac{\alpha}{\beta}(y-\mu_0)^2 - \frac{1}{\lambda} + \psi(\alpha) - \log\beta - \log 2\pi\right)$ **(ii)** $= \frac{1}{2} + \log\left((2\pi)^{\frac{1}{2}}\beta^{\frac{3}{2}}\Gamma(\alpha)\right) - \frac{1}{2}\log\lambda + \alpha - (\alpha + \frac{3}{2})\psi(\alpha)$
$y \sim \text{Poi}(\lambda)$	$\lambda \sim \Gamma(\alpha, \beta)$	$\chi = \alpha/n$ $n = \beta$	**(i)** $= (\psi(\alpha) - \log\beta)y - \frac{\alpha}{\beta} - \sum_{k=1}^{y} \log k$ **(ii)** $= \alpha + \log\Gamma(\alpha) - \log\beta + (1-\alpha)\psi(\alpha)$

Table 4.1: Examples of Exponential Family Distributions where $\psi(x)$ and $B(x)$ denote Digamma and Beta function, respectively.

4.3.1 Exponential Family Distribution

Distributions from the exponential family are very widely used and have favorable analytical properties. Indeed, **(1)** they cover a wide range of target variables like discrete, continuous, counts or spherical coordinates, and **(2)** they benefit from intuitive and generic formulae for their parameters, density functions and statistics which can often be evaluated in closed-form. Important examples of exponential family distributions are Normal, Categorical and Poisson distributions (see Table 4.1). Formally, an exponential family distribution on a target variable $y \in \mathbb{R}$ with *natural parameters* $\boldsymbol{\theta} \in \mathbb{R}^L$ can be denoted as

$$\mathbb{P}(y \,|\, \boldsymbol{\theta}) = h(y) \exp\left(\boldsymbol{\theta}^T \boldsymbol{u}(y) - A(\boldsymbol{\theta})\right) \tag{4.1}$$

where $h : \mathbb{R} \to \mathbb{R}$ is the *carrier or base measure*, $A : \mathbb{R}^L \to \mathbb{R}$ the *log-normalizer* and $\boldsymbol{u} : \mathbb{R} \to \mathbb{R}^L$ the *sufficient statistics* [37, 319]. The entropy of an exponential family distribution can always be written as $\mathbb{H}[\mathbb{P}] = A(\boldsymbol{\theta}) - \boldsymbol{\theta}^T \nabla_{\boldsymbol{\theta}} A(\boldsymbol{\theta}) - \mathbb{E}[\log h(y)]$ [319]. An exponential family distribution always admits a conjugate prior, which often also is a member of the exponential family:

$$\mathbb{Q}(\boldsymbol{\theta} \,|\, \boldsymbol{\chi}, n) = \eta(\boldsymbol{\chi}, n) \exp\left(n\, \boldsymbol{\theta}^T \boldsymbol{\chi} - n A(\boldsymbol{\theta})\right) \tag{4.2}$$

where $\eta(\boldsymbol{\chi}, n)$ is a normalization coefficient, $\boldsymbol{\chi} \in \mathbb{R}^L$ are *prior parameters* and $n \in \mathbb{R}^+$ is the *evidence*. Given a set of N target observations $\{y^{(i)}\}_i^N$, it is easy to compute a closed-form Bayesian update $\mathbb{Q}(\boldsymbol{\theta} \,|\, \boldsymbol{\chi}^{\text{post}}, n^{\text{post}}) \propto \mathbb{P}(\{y^{(i)}\}_i^N \,|\, \boldsymbol{\theta}) \times \mathbb{Q}(\boldsymbol{\theta} \,|\, \chi^{\text{prior}}, n^{\text{prior}})$:

$$\mathbb{Q}(\boldsymbol{\theta} \,|\, \boldsymbol{\chi}^{\text{post}}, n^{\text{post}}) \propto \exp\left(n^{\text{post}} \boldsymbol{\theta}^T \boldsymbol{\chi}^{\text{post}} - n^{\text{post}} A(\boldsymbol{\theta})\right) \tag{4.3}$$

where $\boldsymbol{\chi}^{\text{post}} = \frac{n^{\text{prior}} \boldsymbol{\chi}^{\text{prior}} + \sum_j^N \boldsymbol{u}(y^{(j)})}{n^{\text{prior}} + N}$ and $n^{\text{post}} = n^{\text{prior}} + N$. We see that $\boldsymbol{\chi}^{\text{prior}}$ (resp. $\boldsymbol{\chi}^{\text{post}}$) can be viewed as the average sufficient statistics of n^{prior} (resp. n^{post}) fictitious samples [37]. Further, the average sufficient statistic of fictitious samples is equal to the expected sufficient statistic of the conjugate distribution, i.e. $\boldsymbol{\chi} = \mathbb{E}_{\mathbb{Q}(\boldsymbol{\chi}, n)}[\boldsymbol{\theta}]$ [55, 106]. Thus, the parameter $\boldsymbol{\chi}^{\text{post}}$ carries the inherent aleatoric uncertainty on the target distribution with natural parameters $\boldsymbol{\theta}$, while the evidence n^{post} aligns well with the epistemic uncertainty (i.e. a low evidence means few prior target observations). We stress that the natural conjugate prior parametrization $\boldsymbol{\chi}, n$ is often different from the "well-known" parametrization $\boldsymbol{\kappa}$ used by standard coding libraries. By definition, a bijective mapping $m(\boldsymbol{\kappa}) = (\boldsymbol{\chi}, n)$ from the natural parametrization to the commonly used parametrization always exists (see examples in Table 4.1). Finally, exponential family distributions always admit a closed-form posterior predictive distribution [152].

4.3.2 Input-Dependent Bayesian Update for Exponential Family Distributions

We propose to leverage the power of exponential family distributions for the more complex task when the prediction $y^{(i)}$ depends on the input $\boldsymbol{x}^{(i)}$. Hence, NatPN extends the Bayesian treatment of a single exponential family distribution prediction by predicting

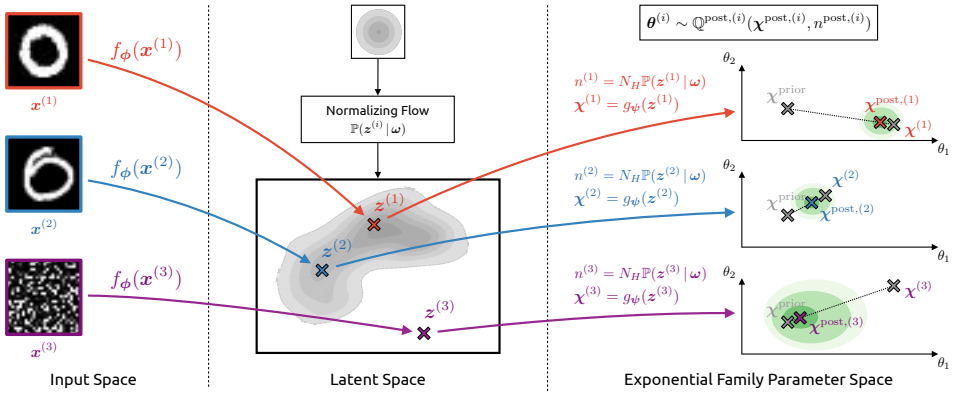


Figure 4.2: Overview of Natural Posterior Network. Inputs $\boldsymbol{x}^{(i)}$ are first mapped to a low-dimensional latent representation $\boldsymbol{z}^{(i)}$ by the encoder $f_{\boldsymbol{\phi}}$. From $\boldsymbol{z}^{(i)}$, the decoder $g_{\boldsymbol{\psi}}$ derives the parameter update $\boldsymbol{\chi}^{(i)}$ while a normalizing flow $\mathbb{P}_{\boldsymbol{\omega}}$ yields the evidence update $n^{(i)}$. Posterior parameters are obtained from a weighted combination of prior and update parameters according to $n^{\text{post},(i)}$.

an individual posterior update per input. We distinguish between the chosen prior parameters $\boldsymbol{\chi}^{\text{prior}}$, n^{prior} shared among samples, and the additional predicted parameters $\boldsymbol{\chi}^{(i)}$, $n^{(i)}$ dependent on the input $\boldsymbol{x}^{(i)}$ leading to the updated posterior parameters:

$$\boldsymbol{\chi}^{\text{post},(i)} = \frac{n^{\text{prior}}\boldsymbol{\chi}^{\text{prior}} + n^{(i)}\boldsymbol{\chi}^{(i)}}{n^{\text{prior}} + n^{(i)}}, \qquad n^{\text{post},(i)} = n^{\text{prior}} + n^{(i)} \tag{4.4}$$

Equivalently, NatPN may be interpreted as predicting a set of $n^{(i)}$ pseudo observations $\{y^{(j)}\}_j^{(i)}$ such that their aggregated sufficient statistics satisfy $\sum_j^{n^{(i)}} y^{(j)} = n^{(i)}\boldsymbol{\chi}^{(i)}$, and perform the respective Bayesian update. This Bayesian update works for *any* choice of exponential family distributions as long as parameters are mapped to their standard form (see Table 4.1). According to the *principle of maximum entropy* [364], a practical choice for the prior is to enforce high entropy for the prior distribution which is usually considered less informative. It is typically achieved when the prior pseudo-count n^{prior} is small and the prior parameter $\boldsymbol{\chi}^{\text{prior}}$ shows a high aleatoric uncertainty.

Hence, NatPN proposes a generic way to perform the input-dependent Bayesian update $\boldsymbol{\chi}^{(i)}$, $n^{(i)}$ for *any* exponential family distribution in three steps (see Fig. 4.2): **(1)** An encoder $f_{\boldsymbol{\phi}}$ maps the input $\boldsymbol{x}^{(i)}$ onto a low-dimensional latent vector $\boldsymbol{z}^{(i)} = f_{\phi}(\boldsymbol{x}^{(i)}) \in \mathbb{R}^H$ representing useful features for the prediction task (see left Fig. 4.2). Note that the architecture of the encoder can be arbitrarily complex. Then, **(2)** the latent representation $\boldsymbol{z}^{(i)}$ is used in two different ways to predict the parameter update $\boldsymbol{\chi}^{(i)}$ and the evidence update $n^{(i)}$ (see center Fig. 4.2). On the one hand, a linear decoder $g_{\boldsymbol{\psi}}$ is trained to output the parameter update $\boldsymbol{\chi}^{(i)} = g_{\boldsymbol{\psi}}(\boldsymbol{z}^{(i)}) \in \mathbb{R}^L$ accounting for the aleatoric uncertainty. On the other hand, a single normalized density is trained to output the evidence update $n^{(i)} = N_H \mathbb{P}(\boldsymbol{z}^{(i)} \,|\, \boldsymbol{\omega})$ accounting for the epistemic uncertainty. The intuition is that increasing the evidence on training data during training forces the evidence everywhere else (incl. far from training data) to decrease thanks to the density normalization constraint.

The constant N_H is a certainty budget distributed by the normalized density $\mathbb{P}(\boldsymbol{z}^{(i)} \,|\, \boldsymbol{\omega})$ over the latent representations $\boldsymbol{z}^{(i)}$ i.e. $N_H = \int N_H \mathbb{P}(\boldsymbol{z}^{(i)} \,|\, \boldsymbol{\omega}) d\boldsymbol{z}^{(i)} = \int n^{(i)} d\boldsymbol{z}^{(i)}$. In practice, we observed that scaling the certainty budget w.r.t. the latent dimension H helped the density to cover larger volumes in higher dimension (see Appendix B.9). Finally, **(3)** NatPN computes the posterior parameters $\boldsymbol{\chi}^{\text{post},(i)}$ and $n^{\text{post},(i)}$ which can be viewed respectively as the mean and concentration of the posterior distribution (see right Fig. 4.2). Note that the posterior parameter $\boldsymbol{\chi}^{\text{post},(i)}$ is a simple weighted average of the prior parameter $\boldsymbol{\chi}^{\text{prior}}$ and the update parameter $\boldsymbol{\chi}^{(i)}$ as shown by Eq. (4.4).

NatPN extends PostNet [67] which also performs an input-dependent Bayesian update with density estimation. Yet, it has three crucial differences which lead to major practical improvements. First, the new exponential family framework is significantly more flexible and is not restricted to classification. Second, the Dirichlet $\boldsymbol{\alpha}$ parameter computation is different: NatPN computes the $\boldsymbol{\chi}$ parameters – which can be viewed as standard softmax output – and the n evidence separately (i.e. $\boldsymbol{\alpha} = n\boldsymbol{\chi}$) while PostNet computes one evidence pseudo-count per class. Third, NatPN is computationally more efficient. It requires a single density while PostNet requires C densities.

4.3.3 ID and OOD Uncertainty Estimates

NatPN intuitively leads to reasonable uncertainty estimation for the two limit cases of strong in-distribution (ID) and out-of-distribution (OOD) inputs (see red and purple samples in Fig. 4.2). For very likely *in-distribution* data (i.e. $\mathbb{P}(\boldsymbol{z}^{(i)} \,|\, \boldsymbol{\omega}) \to \infty$), the posterior parameter overrules the prior (i.e. $\boldsymbol{\chi}^{\text{post},(i)} \to \boldsymbol{\chi}^{(i)}$). Conversely, for very unlikely *out-of-distribution* data (i.e. $\mathbb{P}(\boldsymbol{z}^{(i)} \,|\, \boldsymbol{\omega}) \to 0$), the prior parameter takes over in the posterior update (i.e. $\boldsymbol{\chi}^{\text{post},(i)} \to \boldsymbol{\chi}^{\text{prior}}$). Hence, the choice of the prior parameter should reflect the default prediction when the model lacks knowledge. We formally show under mild assumptions on the encoder that NatPN predicts very low additional evidence ($n^{(i)} \approx 0$) for (almost) any input $\boldsymbol{x}^{(i)}$ far away from the training data (i.e. $||\boldsymbol{x}^{(i)}|| \to +\infty$), thus recovering prior predictions (i.e. $\boldsymbol{\chi}^{\text{post},(i)} \approx \boldsymbol{\chi}^{\text{prior}}$) (see proof in Appendix B.1).

Theorem 1. *Let a NatPN model be parametrized with a (deep) encoder f_ϕ with ReLU activations, a decoder g_ψ and the density $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$. Let $f_\phi(\boldsymbol{x}) = V^{(l)}\boldsymbol{x} + a^{(l)}$ be the piecewise affine representation of the ReLU network f_ϕ on the finite number of affine regions $Q^{(l)}$ [17]. Suppose that $V^{(l)}$ have independent rows and the density function $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$ has bounded derivatives, then for almost any $\boldsymbol{x}$ we have $\mathbb{P}(f_\phi(\delta \cdot \boldsymbol{x}) \,|\, \boldsymbol{\omega}) \underset{\delta \to \infty}{\to} 0$. i.e the evidence becomes small far from training data.*

This theorem only requires that the density avoids very unlikely pathological behavior with unbounded derivatives [108]. A slightly weaker conclusion holds using the notion of limit in density if the density function does not have bounded derivatives [318]. Finally, the independent rows condition is realistic for trained networks with no constant output [182]. It advantageously leads NatPN to consistent uncertainty estimation contrary to standard ReLU networks which are overconfident far from training data [182].

4.3.4 Bayesian NatPN Ensemble

Interestingly, it is natural to extend the Bayesian treatment of a single NatPN to an ensemble of NatPN models (NatPE). An ensemble of m NatPN models is intuitively equivalent to performing m successive Bayesian updates using each NatPN member separately. More formally, given an input $\boldsymbol{x}^{(i)}$ and an ensemble of m jointly trained NatPN models, the Bayesian update for the posterior distribution becomes $\boldsymbol{\chi}^{\text{post},(i)} = \frac{n^{\text{prior}}\boldsymbol{\chi}^{\text{prior}} + \sum_k^m n_k^{(i)} \boldsymbol{\chi}_k^{(i)}}{n^{\text{prior}} + \sum_k^m n_k^{(i)}}$ and $n^{\text{post},(i)} = n^{\text{prior}} + \sum_k^m n_k^{(i)}$. Note that the standard Bayesian averaging which is used in many ensembling methods [76, 246, 434, 439] is different from this Bayesian combination. While Bayesian averaging assume that only one model is correct, the Bayesian combination of NatPN allows *more* or *none* of the models to be "expert" for some input [298]. For example, an input $\boldsymbol{x}^{(i)}$ unfamiliar to every model m (i.e. $n_m^{(i)} \approx 0$) would recover the prior default prediction $\boldsymbol{\chi}^{\text{prior}}, n^{\text{prior}}$. Existing models already had similar properties for Bayesian combination of classifiers [217, 390].

4.3.5 Optimization

The choice of the optimization procedure is of primary importance in order to obtain both high-quality target predictions and uncertainty estimates regardless of the task.

Bayesian Loss. We follow [67] and aim at minimizing the Bayesian formulation:

$$\mathcal{L}^{(i)} = \underbrace{-\mathbb{E}_{\boldsymbol{\theta}^{(i)} \sim \mathbb{Q}^{\text{post},(i)}}[\log \mathbb{P}(y^{(i)} \mid \boldsymbol{\theta}^{(i)})]}_{\text{(i)}} \underbrace{- \mathbb{H}[\mathbb{Q}^{\text{post},(i)}]}_{\text{(ii)}} \tag{4.5}$$

where $\mathbb{H}[\mathbb{Q}^{\text{post},(i)}]$ denotes the entropy of the predicted posterior distribution $\mathbb{Q}^{\text{post},(i)}$. Similarly to the ELBO loss, this loss is guaranteed to be optimal when the predicted posterior distribution is close to the true posterior distribution $\mathbb{Q}^*(\boldsymbol{\theta} \mid \boldsymbol{x}^{(i)})$ i.e. $\mathbb{Q}^{\text{post},(i)} \approx \mathbb{Q}^*(\boldsymbol{\theta} \mid \boldsymbol{x}^{(i)})$ [38, 378, 461]. However, this loss is generally *not* equal to the ELBO loss especially for real valued targets i.e. $y \in \mathbb{R}$ (see Appendix B.2). The term **(i)** is the expected likelihood under the predicted posterior distribution. It can be viewed as the Uncertain Cross Entropy (UCE) loss [36] which is known to reduce uncertainty on observed data. The term **(ii)** is an entropy regularizer acting as a prior which favors uninformative distributions $\mathbb{Q}^{\text{post},(i)}$ with high entropy. In our case, we assume the likelihood $\mathbb{P}(y^{(i)} \mid \boldsymbol{\theta}^{(i)})$ and the posterior $\mathbb{Q}^{\text{post},(i)}$ to be members of the exponential family. We take advantage of the convenient computations for such distributions and derive a more explicit formula for the Bayesian formulation (4.5) (see derivation in the appendix):

$$\mathcal{L}_\lambda^{(i)} \propto \mathbb{E}[\boldsymbol{\theta}]^T \boldsymbol{u}(y^{(i)}) - \mathbb{E}[A(\boldsymbol{\theta})] - \lambda\, \mathbb{H}[\mathbb{Q}^{\text{post},(i)}] \tag{4.6}$$

where λ is an additional regularization weight tuned with a grid search. Note that the term $\mathbb{E}[\boldsymbol{\theta}]^T \boldsymbol{u}(y^{(i)})$ favors a good alignment of the expected sufficient statistic $\mathbb{E}[\boldsymbol{\theta}] = \boldsymbol{\chi}$ with the observed sufficient statistic $\boldsymbol{u}(y^{(i)})$. In practice, all terms can be computed

efficiently in closed form for most exponential family distributions (see examples in Table 4.1). In particular, simplifications are possible when the conjugate prior distribution is also in an exponential family which is often the case. Ultimately Eq. (4.6) applies to *any* exponential family distribution unlike [67].

Optimization Scheme. NatPN is fully differentiable using the closed-form Bayesian loss. Thus, we train the encoder $f_{\boldsymbol{\phi}}$, the parameter decoder $g_{\boldsymbol{\psi}}$ and the normalizing flow $\mathbb{P}(\boldsymbol{z}^{(i)} \,|\, \boldsymbol{\omega})$ w.r.t. parameters $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\omega}$ jointly. Further, we observed that "warm-up training" [19] and "fine-tuning" [211] of the density helped to improve uncertainty estimation for more complex flows and datasets. Thus, we train the normalizing flow density to maximize the likelihood of the latent representations before and after the joint optimization while keeping all other parameters fixed.

4.3.6 Model Limitations

Task-Specific OOD. Previous works show that density estimation is unsuitable for acting on the raw image input [77, 307, 308] or on a non-carefully transformed space [248]. To circumvent this issue, NatPN does not perform OOD detection directly on the input but rather fits a normalizing flow on a learned space. In particular, the latent space is **(1)** low-dimensional, **(2)** task-specific and **(3)** encodes meaningful semantic features. Similarly, [67, 224, 302, 443] already improved OOD detection of density-based methods by leveraging a task-induced bias or low-dimensional statistics. In the case of NatPN, the low-dimensional latent space has to contain relevant features to linearly predict the sufficient statistics required for the task. For example, NatPN aims at a linearly separable latent space for classification. The downside is that NatPN is capable of detecting OOD samples only with respect to the considered task and requires labeled examples during training. As an example, NatPN likely fails to detect a change of image color if the task aims at classifying object shapes and the latent space has no notion of color. Hence, we underline that NatPN comes with a task-dependent OOD definition, which is a reasonable choice in practice.

Model-Task Mismatch. Second, we emphasize that the uncertainty estimation quality of NatPN for (close to) ID data depends on the convergence of the model, the encoder architecture (e.g. MLP, Conv., DenseDepth [118]) and the target distribution (e.g. Poisson, Normal distributions) choice which should match the task needs. However, we show *empirically* that NatPN provides high quality uncertainty estimates in practice on a wide range of tasks. Further, we show *theoretically* that NatPN leads to uncertain prediction far away from training data for *any* exponential family target distributions. In comparison, [286] showed akin guarantees for classification only.

4.4 Experiments

In this section, we compare NatPN to existing methods on extensive experiments including three different tasks: classification, regression and count prediction. For each task

type, we evaluate the prediction quality based on target error and uncertainty metrics. These various set-ups aim to highlight the versatility of NatPN. In particular, NatPN is the only model that adapts to all tasks and achieves high performances for all metrics without requiring multiple forward passes.

Table 4.2: Classification results on Sensorless Drive with Categorical target distribution. Best scores among all single-pass models are in bold. Best scores among all models are starred.

	Accuracy	Brier	9/10 Alea.	9/10 Epist.	OODom Alea.	OODom Epist.
Dropout	98.62 ± 0.11	3.79 ± 0.29	30.20 ± 0.85	32.57 ± 1.45	27.03 ± 0.51	95.30 ± 1.66
Ensemble	98.83 ± 0.17	3.00 ± 0.54	30.79 ± 0.74	32.61 ± 1.06	27.16 ± 0.59	99.97 ± 0.01
NatPE	*99.66 ± 0.03	*0.68 ± 0.05	77.05 ± 1.93	83.73 ± 1.89	99.99 ± 0.00	*100.00 ± 0.00
R-PriorNet	98.85 ± 0.25	2.01 ± 0.47	40.13 ± 2.99	30.07 ± 0.81	***100.00 ± 0.00**	23.59 ± 0.00
EnD^2	93.95 ± 2.35	28.09 ± 6.40	26.35 ± 0.60	24.85 ± 0.43	84.43 ± 15.21	23.58 ± 0.00
PostNet	**99.64 ± 0.02**	**0.75 ± 0.08**	80.60 ± 1.68	***92.57 ± 1.41**	***100.00 ± 0.00**	***100.00 ± 0.00**
NatPN	99.61 ± 0.05	1.04 ± 0.29	***81.43 ± 1.89**	79.54 ± 2.62	99.98 ± 0.00	***100.00 ± 0.00**

Table 4.3: Classification results on CIFAR-10 with Categorical target distribution. Best scores among all single-pass models are in bold. Best scores among all models are starred. Gray numbers indicate that R-PriorNet has seen samples from the SVHN dataset during training.

	Accuracy	Brier	SVHN Alea.	SVHN Epist.	CelebA Alea.	CelebA Epist.	OODom Alea.	OODom Epist.
Dropout	88.15 ± 0.20	19.59 ± 0.41	80.63 ± 1.59	73.09 ± 1.51	71.84 ± 4.28	71.04 ± 3.92	18.42 ± 1.11	49.69 ± 9.10
Ensemble	*89.95 ± 0.11	17.33 ± 0.17	85.26 ± 0.84	82.51 ± 0.63	76.20 ± 0.87	74.23 ± 0.78	25.30 ± 4.02	89.21 ± 7.55
NatPE	89.21 ± 0.09	17.41 ± 0.12	85.66 ± 0.34	*83.16 ± 0.67	*78.95 ± 1.15	*82.06 ± 1.30	87.27 ± 1.79	*98.88 ± 0.26
R-PriorNet	**88.94 ± 0.23**	***15.99 ± 0.32**	99.87 ± 0.02	99.94 ± 0.01	67.74 ± 4.86	59.55 ± 7.90	42.21 ± 8.77	38.25 ± 9.82
EnD^2	84.03 ± 0.25	40.84 ± 0.36	***86.47 ± 0.66**	**81.84 ± 0.92**	75.54 ± 1.79	75.94 ± 1.82	42.19 ± 8.77	15.79 ± 0.27
PostNet	87.95 ± 0.20	20.19 ± 0.40	82.35 ± 0.68	79.24 ± 1.49	72.96 ± 2.33	75.84 ± 1.61	85.89 ± 4.10	92.30 ± 2.18
NatPN	87.90 ± 0.16	19.99 ± 0.46	82.29 ± 1.11	77.83 ± 1.22	**76.01 ± 1.18**	**76.87 ± 3.38**	***93.67 ± 3.03**	**94.90 ± 3.09**

Table 4.4: Results on the Bike Sharing Dataset with Normal $\mathcal{N}$ and Poison Poi target distributions. Best scores among all single-pass models are in bold. Best scores among all models are starred.

	RMSE	Calibration	Winter Epist.	Spring Epist.	Autumn Epist.	OODom Epist.
Dropout-$\mathcal{N}$	70.20 ± 1.30	6.05 ± 0.77	15.26 ± 0.51	13.66 ± 0.16	15.11 ± 0.46	99.99 ± 0.01
Ensemble-$\mathcal{N}$	*48.02 ± 2.78	5.88 ± 1.00	42.46 ± 2.29	21.28 ± 0.38	21.97 ± 0.58	*100.00 ± 0.00
EvReg-$\mathcal{N}$	**49.58 ± 1.51**	3.77 ± 0.81	17.19 ± 0.76	15.54 ± 0.65	14.75 ± 0.29	34.99 ± 17.02
NatPN-$\mathcal{N}$	49.85 ± 1.38	***1.95 ± 0.34**	***55.04 ± 6.81**	***23.25 ± 1.20**	***27.78 ± 2.47**	***100.00 ± 0.00**
Dropout-Poi	66.57 ± 4.61	55.00 ± 0.22	16.02 ± 0.48	13.48 ± 0.38	18.09 ± 0.82	***100.00 ± 0.00**
Ensemble-Poi	***48.22 ± 2.06**	55.31 ± 0.21	83.88 ± 1.22	34.21 ± 1.81	41.29 ± 3.23	***100.00 ± 0.00**
NatPN-Poi	51.79 ± 0.78	***31.04 ± 1.81**	***85.15 ± 3.61**	***37.03 ± 2.35**	***42.73 ± 4.38**	***100.00 ± 0.00**

4.4.1 Setup

In our experiments, we change the encoder architecture of NatPN to match the dataset needs. We perform a grid search over normalizing flows types (i.e. radial flows [358] and MAF [154, 339]) and latent dimensions. We show further experiments on architecture, latent dimension, normalizing flow choices and certainty budget choice in the appendix. Furthermore, we use approximations of the log-Gamma $\log\Gamma(x)$ and the

Table 4.5: Regression results on models trained on different UCI datasets with Normal target distribution. The upper half displays models trained on Kin8nm, the lower half shows models trained on Concrete Compressive Strength.

	RMSE	Calibration	Energy Alea.	Energy Epist.	Concrete Alea.	Concrete Epist.	Kin8nm Alea.	Kin8nm Epist.
Dropout	0.09 ± 0.00	3.13 ± 0.43	90.18 ± 6.00	99.94 ± 0.06	*100.00 ± 0.00	*100.00 ± 0.00	**in-distribution**	
Ensemble	*0.07 ± 0.00	2.69 ± 0.49	*100.00 ± 0.00	*100.00 ± 0.00	*100.00 ± 0.00	*100.00 ± 0.00		
NatPE	0.08 ± 0.00	5.49 ± 0.30	*100.00 ± 0.00	*100.00 ± 0.00	*100.00 ± 0.00	*100.00 ± 0.00		
EvReg	0.09 ± 0.00	3.74 ± 0.53	88.06 ± 11.94	88.06 ± 11.94	***100.00 ± 0.00**	86.84 ± 13.16	**in-distribution**	
NatPN	**0.08 ± 0.00**	***2.04 ± 0.45**	***100.00 ± 0.00**	***100.00 ± 0.00**	***100.00 ± 0.00**	***100.00 ± 0.00**		
Dropout	5.67 ± 0.07	*3.03 ± 0.40	9.33 ± 0.36	93.53 ± 2.41	**in-distribution**		1.09 ± 0.13	64.30 ± 7.14
Ensemble	5.69 ± 0.20	3.81 ± 0.67	54.19 ± 18.93	*100.00 ± 0.00			72.57 ± 19.32	*100.00 ± 0.00
NatPE	*4.78 ± 0.20	5.58 ± 1.27	*100.00 ± 0.00	*100.00 ± 0.00			*100.00 ± 0.00	*100.00 ± 0.00
EvReg	6.04 ± 0.18	7.36 ± 1.04	8.93 ± 0.02	51.39 ± 18.56	**in-distribution**		0.93 ± 0.00	34.44 ± 20.95
NatPN	**5.83 ± 0.23**	**5.41 ± 1.33**	***100.00 ± 0.00**	***100.00 ± 0.00**			***100.00 ± 0.00**	***100.00 ± 0.00**

Table 4.6: Regression results on NYU Depth v2 with Normal target distribution. RMSE is in cm. OOD scores on LSUN are reported on the held-out classes 'classrooms' (left) and 'churches' (right).

	RMSE	Calibration	LSUN Alea.	LSUN Epist.	KITTI Alea.	KITTI Epist.	OODom Alea.	OODom Epist.
Dropout	46.95	4.03	*95.29 / 97.74	83.89 / 83.22	98.07	84.90	74.40	*100.00
EvReg	***28.88**	***1.05**	58.70 / 56.71	70.19 / 64.02	56.60	62.67	75.43	56.39
NatPN	29.72	1.14	**94.13** / ***98.67**	***89.08** / ***90.56**	***98.93**	***93.15**	***100.00**	***100.00**

Digamma $\psi(x)$ functions for large input values to avoid unstable floating computations (see Appendix B.2). As prior parameters, we set $\boldsymbol{\chi}^{\text{prior}} = \mathbf{1}_C/C, n^{\text{prior}} = C$ for classification, $\boldsymbol{\chi}^{\text{prior}} = (0, 100)^T, n^{\text{prior}} = 1$ for regression and $\chi^{\text{prior}} = 1, n^{\text{prior}} = 1$ for count prediction enforcing high entropy for prior distributions.

Baselines. We focus on recent models parametrizing prior distributions over the target distribution. For classification, we compare NatPN to Reverse KL divergence Prior Networks (R-PriorNet) [273], Ensemble Distribution Distillation (EnD^2) [277] and Posterior Networks (PostNet) [67]. Note that Prior Networks require OOD training data — we use an auxiliary dataset when available and Gaussian noise otherwise. For regression, we compare to Evidential Regression (EvReg) [10]. Beyond baselines parametrizing conjugate prior distributions, we also compare to dropout (Dropout) [141] and ensemble (Ensemble) [246] models for all tasks. These sampling baselines require multiple forward passes for uncertainty estimation. Further details are given in the appendix.

Datasets. We split all datasets into train, validation and test sets. For classification, we use one tabular dataset (Sensorless Drive [111]) and three image datasets (MNIST [249], FMNIST [450] and CIFAR-10 [238]). For count prediction, we use the Bike Sharing dataset [126] to predict the number of bike rentals within an hour. For regression, we also use the Bike Sharing dataset where the target is viewed as continuous, real-world UCI datasets used in [10, 185] and the image NYU Depth v2 dataset [311] where the goal is to predict the image depth per pixel. All inputs are rescaled with zero mean and unit variance. We also scale the output target for regression. Further details are given in the appendix.

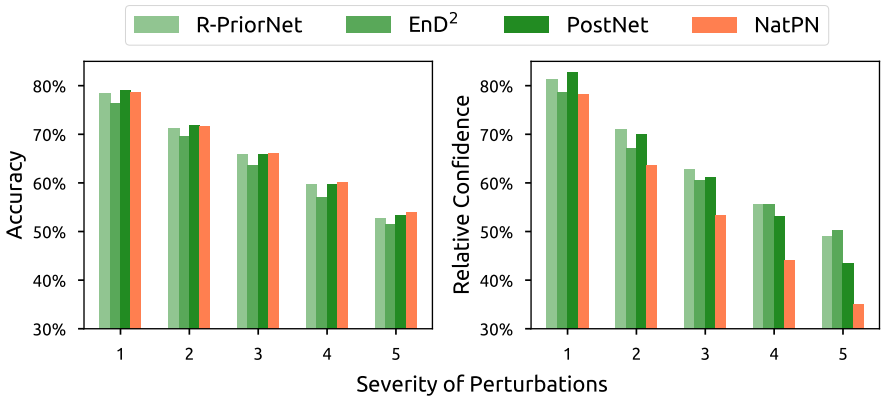


Figure 4.3: Averaged accuracy and confidence under 15 dataset shifts on CIFAR-10 [183]. On more severe perturbations (i.e. data further away from data distribution), NatPN maintains a competitive accuracy while assigning higher epistemic uncertainty as desired. Baselines provide a slower relative confidence decay.

Metrics. Beyond the target prediction error, we evaluate model uncertainty estimation using calibration and OOD detection scores. Furthermore, we report the inference speed. Further results including histograms with uncertainty estimates or latent space visualization are presented in appendix.

Target error: We use the accuracy (Accuracy) for classification and the Root Mean Squared Error (RMSE) for regression and count prediction.

Calibration: For classification, we use the Brier score (Brier) [161]. For regression and count prediction, we use the quantile calibration score (Calibration) [242].

OOD detection: We evaluate how the uncertainty scores enable to detect OOD data using the area under the precision-recall curve (AUC-PR) the area under the receiver operating characteristic curve (AUC-ROC) in the appendix. We use two different uncertainty measures: the negative entropy of the predicted target distribution accounting for the aleatoric uncertainty (Alea. OOD) and the predicted evidence or variance of the predicted mean (Epist. OOD). Similarly to [335, 67], we use four different types of clear OOD samples: Unseen datasets (KMNIST [83], Fashion-MNIST [450], SVHN [313], LSUN [457], CelebA [262], KITTI [151]), left-out data (classes 9/10 for Sensorless Drive, winter/spring/autumn seasons for Bike Sharing), out-of-domain data not normalized in $[0, 1]$ (OODom) and dataset shifts (corrupted CIFAR-10 [183]). Further details are given in the appendix.

4.4.2 Results

Classification. We show results for the tabular dataset Sensorless Drive with unbounded input domain in Table 4.2, and the image datasets MNIST, FMNIST and CIFAR-10 with bounded input domain in Table 4.3 and appendix. Overall, for classification NatPN performs on par with best single-pass baselines (i.e. 12/30 top-1 scores, 25/30 top-2 scores) and NatPE performs the best among multiple-pass models (i.e. 28/30 top-1 scores). A single NatPN achieves accuracy and calibration performance on par with the most calibrated baselines, namely PostNet and R-PriorNet which requires one normalizing flow

per class or training OOD data. Further, NatPE consistently improves accuracy and calibration performance of a single NatPN which underlines the benefit of aggregating multiple models predictions for accuracy and calibration [246]. Without requiring OOD data during training, both NatPN and NatPE achieve excellent OOD detection scores w.r.t. all OOD types. This strongly suggests that NatPN does *not* suffer from the flaws in [77, 307, 308]. In particular, NatPN and NatPE achieve almost perfect OODom scores contrary to all other baselines except PostNet. This observation aligns well with the theoretical guarantee of NatPN far from training data (see Theorem 1) which also applies to each NatPE member. The similar performance of NatPN and PostNet for classification is intuitively explained by their akin design: both models perform density estimation on a low-dimensional latent space. Similarly to [67], we compute the average confidence avg-conf $= \frac{1}{N}\sum_i^N n^{(i)} = \frac{1}{N}\sum_i^N \alpha_0^{(i)}$ and then compare the average confidence change. The average confidence change is computed by taking the ratio of the average confidence of N corrupted data at severity s and the average confidence of N clean data (i.e. the corrupted data at severity 0) i.e. $\frac{\text{avg-conf}_s}{\text{avg-conf}_0}$ for $s \in [\![1, 5]\!]$. NatPN maintains a competitive accuracy (Fig. 4.3, left) while assigning higher epistemic uncertainty as desired (Fig. 4.3, right). Baselines provide a slower relative confidence decay.

Regression & Count Prediction. We show the results for the Bike Sharing, the tabular UCI datasets and the image NYU Depth v2 datasets in Tables 4.4 to 4.6. For the large NYU dataset, we compare against all baselines which require only a single model to be trained. Overall, NatPN outperforms other single-pass models for 23/26 scores for regression, thus significantly improving calibration and OOD detection scores. Further, NatPN shows a strong improvement for calibration and OOD detection for count prediction with Poisson distributions among all models. Interestingly, all the models are less calibrated on the Bike Sharing dataset using a target Poisson distribution rather than a target Normal distribution. This suggests a mismatch of the Poisson distribution for this particular task. The almost perfect OODom scores of NatPN validate again Theorem 1 which also holds for regression.

Inference Speed. We show the average inference time per batch for all models on CIFAR-10 for classification and the NYU Depth v2 dataset for regression in Table 4.7. NatPN shows a significant improvement over Dropout and Ensemble which are both approximately five times slower since they require five forward passes for prediction. Notably, the NatPN speedup does not deteriorate target error and uncertainty scores. NatPN is slightly slower than R-PriorNet, EnD^2 and EvReg as they do not evaluate an additional normalizing flow. However, NatPN – which uses a single normalizing flow – is faster than PostNet – which

Table 4.7: Batched Inference Time (in ms), NVIDIA GTX 1080 Ti

	CIFAR-10 (batch size 4,096)	NYU Depth v2 (batch size 4)
Dropout	407.91 ± 5.65	650.96 ± 0.22
Ensemble	361.61 ± 5.41	649.78 ± 0.18
R-PriorNet	61.83 ± 2.57	–
EnD^2	61.83 ± 2.57	–
PostNet	88.56 ± 0.06	–
EvReg	–	129.88 ± 0.75
NatPN	75.64 ± 0.04	137.13 ± 0.18
NatPE	370.17 ± 0.09	676.74 ± 0.38

scales linearly w.r.t. the number of classes since it evaluates one normalizing flow per class. Lastly, NatPN is the only single-pass model that can be used for *both* tasks.

4.5 Conclusion

We introduce Natural Posterior Network which belongs to the family of models parametrizing conjugate prior distributions. NatPN is capable of efficient uncertainty estimation for *any* task where the target distribution is in the exponential family (incl. classification, regression and count prediction). NatPN relies on the Bayes formula and the general form of exponential family distributions to perform a closed-form input-dependent posterior update over the target distribution. Further, an ensemble of NatPNs has a principled Bayesian combination interpretation. Theoretically, NatPN guarantees high uncertainty far from training data. Experimentally, NatPN achieves fast, versatile and high-quality uncertainty predictions with strong performance in calibration and OOD detection.

5 Practicality of Uncertainty Estimation

I can live with doubt and uncertainty and not knowing.
I think it is much more interesting to live not knowing than to have answers that might be wrong.

Richard P. Feynman

5.1 Introduction

While we have focused in the previous sections on proposing new Bayesian models for efficient uncertainty estimation on independent data, we now turn our attention on the practical considerations when using Deterministic Uncertainty Methods (DUMs) [346] like evidential models and GP-based models discussed in Section 2.2. Contrary to uncertainty methods such as Ensembles [246], MC Dropout [141] or other Bayesian neural networks on weights [42], which require multiple forward passes to make predictions, DUMs only require a single forward pass, thus making them significantly more computationally efficient. Generally, DUMs are composed of three components with high potential impact on their performances: the *training* procedure which is supposed to optimize the model toward high predictive and uncertainty performances, the core *architecture* which is supposed to define informative embeddings used to make predictions, and the *prior* which is supposed to define the default uncertain predictions.

Contributions. In this chapter, we investigate the role of the three training, architecture, and prior components on the quality of DUMs uncertainty estimates by evaluating calibration performances, OOD detection, and OOD generalization. Our main contributions are:

- **Training**: We show that *decoupling the learning rates* of the core architecture and uncertainty heads of DUMs, *jointly training* the core architecture and the uncertainty head of DUMs, and *pretraining* with *more data* and *higher data quality* improve uncertainty performances.
- **Architecture**: We demonstrate that the expressiveness of the core architecture defined by the *architecture type*, *architecture size*, and *dimension of the latent space* is crucial for *both* predictive and uncertainty performances. Further, we show that applying additional regularization constraints to avoid *feature collapse* does not find better trade-off between OOD detection and generalization, even sometimes degrading performances.

- **Prior**: In contrast to Bayesian neural networks on weights where the choice of prior is critical [438, 137, 323, 214], we empirically show that the choice of prior defined in the training loss or in the uncertainty head of DUMs has a relatively small effect on the final uncertainty performances.

5.2 Deterministic Uncertainty Methods

We consider a classification task where the goal is to predict the label $y^{(i)} \in \{1, \dots, C\}$ based on the input feature $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$. In this case, a DUM generally performs predictions in two main steps: **(1)** A core *architecture* f_ϕ maps the input features $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$ to a latent representation $\boldsymbol{z}^{(i)} \in \mathbb{R}^H$, i.e. $f_\phi(\boldsymbol{x}^{(i)}) = \boldsymbol{z}^{(i)}$. In practice, important design choices are the latent dimension H and the architecture f_ϕ which should be adapted to the task (see Section 5.4). Further, multiple works proposed to apply additional bi-Lipschitz or reconstruction constraints to enrich the informativeness of the latent representation $\boldsymbol{z}^{(i)}$ (see Section 5.4). **(2)** An *uncertainty head* g_ψ maps the latent representation $\boldsymbol{z}^{(i)}$ to a predicted label $\hat{y}^{(i)}$ and an associated (aleatoric, epistemic, or predictive) uncertainty estimate $u^{(i)}$, i.e. $g_\psi(\boldsymbol{z}^{(i)}) = (\hat{y}^{(i)}, u^{(i)})$. In practice, important design choices are the type of uncertainty head which are generally instantiated with a Gaussian Process (GP) [247, 421, 420, 36, 71], a density estimator [67, 68, 71, 398, 36, 303, 344, 443, 448], or an evidential model [67, 68, 71, 398, 36, 271], and the choice of the prior used by the uncertainty head (see Section 5.5). Beyond core architecture and uncertainty head, another important choice is the training procedure which can either couple or decouple the parameters of the core architecture and uncertainty head (see Section 5.3).

In this work we focus on two recent DUMs which cover different types of uncertainty heads: Natural Posterior Network (NatPN) [68] which learns an evidential distribution based on density estimation on the latent space, and Deterministic Uncertainty Estimator (DUE) [421] which learns a deep Gaussian Process by parametrizing learnable inducing points in the latent space (see Appendix C.0.1 for further method details). While NatPN is capable to differentiate aleatoric, epistemic, and predictive uncertainty, DUE only outputs the predictive uncertainty. For all the experiments, we use the same default setup: we first *pretrain* the encoder with the cross-entropy loss until convergence, then load the pretrained encoder and jointly *train* the encoder and uncertainty head (see Appendices C.1 to C.3 for further component details). We report the predictive performance via accuracy, and the uncertainty performances with Brier Score and OOD detection results after averaging over 5 seeds (see Appendix C.0.3 for further metric details). We perform our experiments on MNIST [249], CIFAR10 and CIFAR100 [238], and Camelyon [233]. OOD results reported in Tables 5.1 to 5.4 averages the uncertainty estimation from five OOD datasets: SVHN, STL10, CelebA, Camelyon and SVHN OODom [313, 86, 262]. Our code and additional material is available online[1].

Related work. Previous works survey OOD detection methods [453], OOD generazilation methods [385], or a wide range of uncertainty estimation methods [150, 348, 419, 3] by presenting key methods and challenges. These surveys do not focus on determinis-

[1] `https://www.cs.cit.tum.de/daml/training-architecture-prior-dum/`

tic methods and do not make empirical analysis. Other works propose great empirical studies to compare uncertainty estimation methods under shifts [335], or analyze the role of the prior in Bayesian neural networks on weights [438, 137, 323, 214]. These works do not focus on DUMs. Closer to our work, Postels et al. [346] compares methods in the DUMs family and demonstrate calibration limitations. In contrast, we evaluate the role of *components* in DUMs and show that carefully specifying *training*, *architecture*, or *prior* can improve uncertainty metrics like calibration and OOD detection but also ID and OOD predictive performances.

5.3 Training for DUMs

In this section, we study the importance of the training procedure in the performance of DUMs. To this end, we look at *decoupling the learning rates* of the core encoder architecture and the uncertainty head, different *training schemes*, and different *pretraining schemes*.

Decoupling learning rates. We decouple the learning rates of the core architecture and the uncertainty head. We show the validation results for CIFAR100 as ID and SVHN as OOD with the core architecture ResNet18 in Fig. 5.1. *Observation:* We observe that, when using different learning rates for the core architecture and the uncertainty head, NatPN improves Brier Score and OOD epistemic results and DUE significantly improves both predictive and uncertainty results. Hence, this shows that decoupling learning rates can improve results of DUMs, thus suggesting that the core architecture and the uncertainty head have training dynamics which requires different considerations.

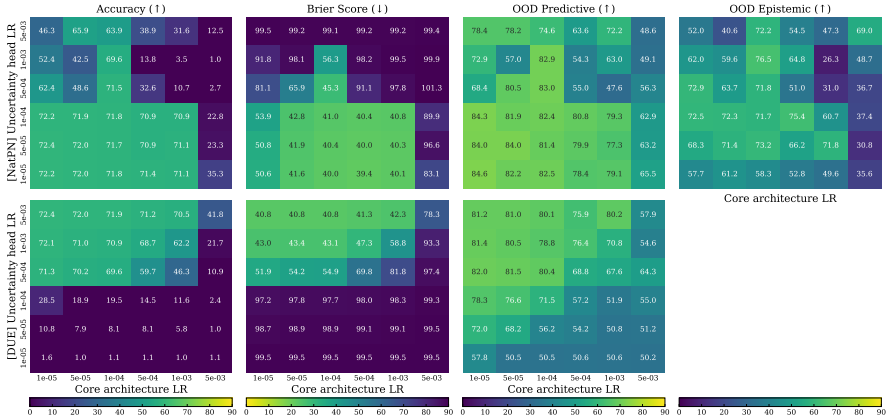


Figure 5.1: Results of DUMS on CIFAR100 with ResNet18 when **decoupling learning rates** of the core architecture and the uncertainty head. Decoupling learning rates improve DUMs performance.

Training schemes. We compare two settings: the *joint* training in which we jointly train the weights of the core architecture and uncertainty head, and the *sequential* training in which we only train the uncertainty head by keeping the weights of the pretrained core architecture fixed. For each of the setting, we apply two additional techniques to

stabilize the training: adding a *batch normalization* to the last layer of the encoder to enforce latent representations to locate in a normalized region [204, 68], and *resetting the last layer* to retrain its weights to improve robustness to spurious correlation [225]. We show the results for CIFAR100 as ID and five difference OOD datasets with the ResNet18 as core architecture in Table 5.1 and additional results in the appendix Table C.3. *Observation:* We observe that, compared to its sequential counterpart, joint training consistently improves DUMs performance for most metrics, thus suggesting that joint training should be preferred in practice for DUMs. Furthermore, while the GP-based method DUE does not benefit from stabilization techniques, we observe that they can significantly increase performance of the density-based method NatPN. This behavior is intuitively explained by the practical difficulty to accurately fit densities in high dimensional latent space. This can be significantly improved by using more powerful density estimator (see Table C.1 in appendix).

Pretraining schemes. We compare multiple training schemes which differ in terms of *amount* and *quality* of data used for pretraining. Hence, we do not pretrain the core architecture or pretrain it with 10% of CIFAR100, 100% of CIFAR100 without and with Gaussian noise, or ImageNet. We show the results for CIFAR100 as ID and five different OOD datasets with ResNet50 as core architecture in Table 5.2 and additional results in the appendix Table C.4. *Observation:* We observe that, while too few data for pretraining does not improve final performance of DUMs, the overall performance significantly increase when the encoder is pretrained with high quantity and high quality of data. Similarly to Kirichenko et al. [224], this suggests that the embedding quality is important to improve uncertainty quantification. Here, we show additionally that embeddings pretrained with many high quality data are crucial to facilitate the prediction of the uncertainty head.

Table 5.1: Results of DUMs on CIFAR100 with ResNet18 under different **training schemes** using *joint/sequential* training with no additional layer, an additional batch norm layer, or resetting the last layer. Gray cells indicate the best between *joint/sequential* while bold numbers indicate the best overall. OOD results are averaged over OOD datasets. We observe that joint training works best and stabilization techniques can improve performances.

Method	Train Schema	Accuracy (↑)	Brier Score (↓)	OOD Pred. (↑)	OOD Epis. (↑)
NatPN	joint	71.12 ± 0.18	41.06 ± 0.18	75.17 ± 1.60	63.94 ± 2.80
	joint + bn	71.60 ± 0.14	41.11 ± 0.12	74.22 ± 0.94	66.17 ± 2.55
	joint + reset	71.61 ± 0.18	**40.76 ± 0.18**	**75.35 ± 0.71**	**69.02 ± 1.49**
	sequential	**72.00 ± 0.19**	42.20 ± 0.09	75.09 ± 0.86	53.49 ± 2.56
	sequential + bn	71.98 ± 0.18	42.39 ± 0.11	75.01 ± 0.86	52.34 ± 2.81
	sequential + reset	71.79 ± 0.17	40.95 ± 0.14	74.63 ± 0.85	61.90 ± 2.14
DUE	joint	**72.33 ± 0.11**	**40.80 ± 0.11**	74.74 ± 0.89	-
	joint + bn	72.30 ± 0.09	40.85 ± 0.12	74.63 ± 0.95	-
	joint + reset	71.94 ± 0.12	41.43 ± 0.12	74.89 ± 0.76	-
	sequential	72.07 ± 0.10	41.66 ± 0.10	74.82 ± 0.90	-
	sequential + bn	72.04 ± 0.13	41.73 ± 0.11	74.88 ± 0.95	-
	sequential + reset	71.56 ± 0.14	42.30 ± 0.11	**75.08 ± 1.01**	-

Table 5.2: Results of DUMs with ResNet50 under different **pretraining schemes** using no pretraining, pretraining on 10% of CIFAR100, 100% of CIFAR100 without Gaussian noise and with Gaussian noise, or ImageNet. OOD results are averaged over OOD datasets. Bold numbers indicate best results among all settings. We observe that high quantity and high quality of data can improve performances.

Method	Pretrain Schema	Accuracy (↑)	Brier Score (↓)	OOD Pred. (↑)	OOD Epis. (↑)
NatPN	None	78.45 ± 1.94	30.16 ± 2.57	79.85 ± 3.30	85.81 ± 2.05
	C100 (10%)	67.25 ± 0.71	44.69 ± 0.94	68.10 ± 1.90	78.50 ± 3.27
	C100 (100%) + $\mathcal{N}(0.5)$	72.50 ± 1.07	39.36 ± 1.43	68.66 ± 3.76	73.54 ± 1.84
	C100 (100%) + $\mathcal{N}(0.1)$	75.25 ± 0.61	35.99 ± 0.88	69.78 ± 4.65	75.81 ± 2.85
	C100 (100%)	76.31 ± 0.45	34.32 ± 0.51	78.95 ± 3.19	76.91 ± 2.64
	ImageNet	**84.22 ± 0.12**	**23.67 ± 0.27**	**84.95 ± 1.48**	**89.08 ± 0.70**
DUE	None	72.41 ± 0.24	47.35 ± 0.25	80.04 ± 1.28	-
	C100 (10%)	63.86 ± 0.58	50.94 ± 0.53	72.44 ± 1.32	-
	C100 (100%)	76.38 ± 0.35	36.89 ± 0.50	81.71 ± 1.87	-
	C100 (100%) + $\mathcal{N}(0.5)$	72.10 ± 1.00	42.48 ± 1.18	74.89 ± 1.97	-
	C100 (100%) + $\mathcal{N}(0.1)$	75.31 ± 0.91	38.31 ± 1.22	79.43 ± 1.93	-
	ImageNet	**82.42 ± 0.14**	**28.09 ± 0.19**	**90.24 ± 0.51**	-

5.4 Architecture for DUMs

In this section, we study the impact of the architecture component in DUMs. To this end, we look at different *latent dimensions*, different *architectural types and size*, and applying different *regularization constraints* to avoid *feature collapse* [421].

Latent dimension. We vary the dimension of the output space of the core architecture. We show the results for each pair of ID dataset and its distribution shifted OOD dataset (MNIST/CMNIST, CIFAR/CIFAR-C, CamelyonID/CamelyonOOD) with the core architecture ResNet18 for MNIST/CIFAR, and WideResNet-28-10 for Camelyon in Fig. 5.2 and additional uncertainty estimation results in the appendix Fig. C.5. *Observation:* We observe that increasing the latent dimensions leads to improvement for DUMs on ID and OOD datasets with particularly significant improvement for NatPN (see Fig. 5.2). This suggests that higher latent dimensions are more expressive by encoding more information. However, we observe that a too high latent dimension can degrade OOD detection performance by causing numerical instabilities in the training (see Fig. C.5), suggesting a trade-off between OOD generalization and OOD detection.

Architecture type and size. We compare the influence of the type and size of the core architecture on the performance of DUMs. We consider residual, convolutional, and transformer architectures like ResNet18, ResNet50, EfficientNetV2, and Swin [181, 405, 260]. We show the results for DUMs trained on CIFAR100 as ID with the different core architectures in Table 5.3 and additional results at appendix Table C.5. *Observation:* We observe that models with more parameters achieve better results. In particular, ResNet50 achieves significantly better results than ResNet18. Further, more recent core architectures like EfficientNetV2 and Swin are better calibrated and more expressive leading to a better overall performance. This can be explained by the fact that they are more expressive and provide more informative embeddings for the uncertainty head to operate on. This aligns with Minderer et al. [288] which states that the architecture type is important for the calibration properties.

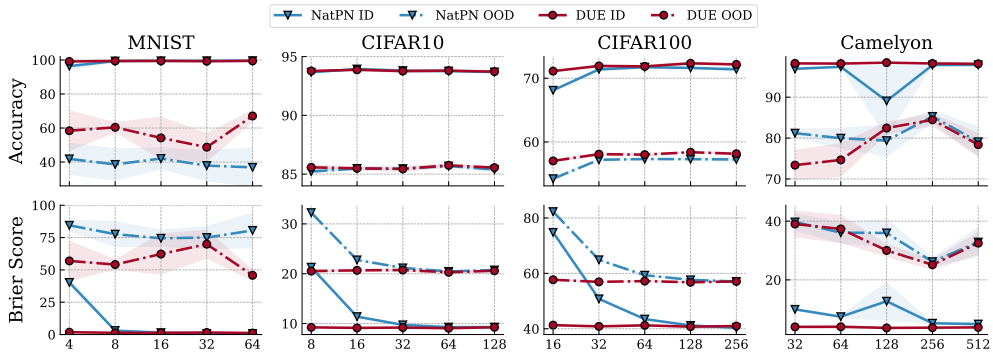


Figure 5.2: Results of DUMs when varying the **latent dimension size**. We observe that increasing the latent dimension consistently leads to similar or better predictive performance.

Table 5.3: Results of DUMs for different **architecture types.** including residual, convolutional, and transformer architectures on CIFAR100. OOD results are averaged over OOD datasets. Bold numbers indicate best results among all settings. Larger and more recent architectures are better calibrated with similar or better uncertainty estimation.

Method	Architecture	#Parameters	Accuracy (↑)	Brier Score (↓)	OOD Pred. (↑)	OOD Epis. (↑)
NatPN	ResNet18	11.6M	80.31 ± 0.09	33.69 ± 0.15	87.49 ± 1.90	81.79 ± 1.38
	ResNet50	25.5M	84.22 ± 0.12	23.67 ± 0.27	84.95 ± 1.48	89.08 ± 0.70
	EffNet_V2_S	21.4M	**88.43 ± 0.10**	**17.08 ± 0.10**	**87.79 ± 0.77**	89.47 ± 0.59
	Swin_T	28.2M	87.99 ± 0.09	18.48 ± 0.06	85.91 ± 1.17	**90.23 ± 0.81**
DUE	ResNet18	11.6M	78.85 ± 0.19	36.57 ± 0.15	88.04 ± 0.67	-
	ResNet50	25.5M	82.42 ± 0.14	28.09 ± 0.19	**90.24 ± 0.51**	-
	EffNet_V2_S	21.4M	86.92 ± 0.08	**21.07 ± 0.06**	89.43 ± 0.67	-
	Swin_T	28.2M	**86.93 ± 0.05**	23.23 ± 0.05	89.90 ± 0.36	-

Regularization constraints. *Feature collapse* is a phenomenon where a model may discard important parts of the input information during its training phase, which may degrade OOD detection performance [421]. Two techniques to avoid feature collapses are *bi-Lipschitz* constraints via combining residual connections and lipschitz constraints [247], and *reconstruction* constraints via adding an additional reconstruction term in the loss [344]. We show the results for DUMs trained on the datasets MNIST and CIFAR100 with ResNet18 in Figs. 5.3 and 5.4 and additional results for other datasets (Toy dataset, CIFAR10, Camelyon) at Appendix C.2. *Observation:* We observe that the reconstruction technique is not capable to avoid feature collapse. Indeed, we show that, even with reconstruction constraints, some (non-discriminative) features can completely collapse (see Figs. C.3b and C.4b for toy examples). Hence, while this method can lead to small OOD improvements on simple tasks (see e.g. MNIST in Fig. 5.4), this benefit does not generalize to more complex tasks (see e.g. CIFAR100 in Fig. 5.4). In contrast, we observe that bilipschitz constraints indeed mitigate the collapse of features (see Figs. C.3a and C.4a for toy examples), leading to similar or higher OOD detection performance (see Fig. 5.3). The mitigation of feature collapse can be mostly assigned to the residual connection constraints. However, bilipschitz constraints can improve OOD detection results

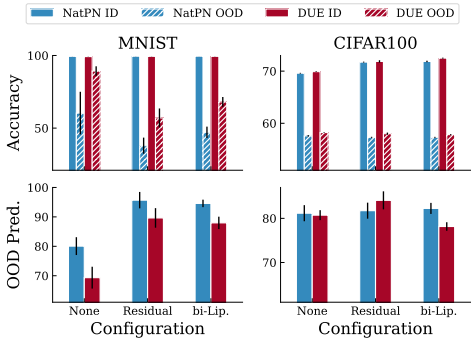


Figure 5.3: Results OOD generalization and detection of DUMs with none, residual and bi-lipschitz **architecture constraints** on MNIST/CMNIST and CIFAR/CIFAR-C. Bi-lipschitz can improve OOD detection by mitigating feature collapse (see Fig. C.3a) at the expense of degrading OOD generalization.

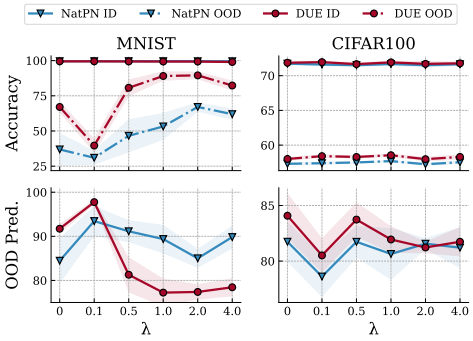


Figure 5.4: Results OOD generalization and detection of DUMs with reconstruction **architecture constraints** on MNIST/CMNIST and CIFAR/CIFAR-C. Increasing the reconstruction strength λ improves the OOD generalization on simple MNIST/CMNIST dataset but fails for complex datasets. Reconstruction fails to improve OOD detection since it does not avoid feature collapse (see Fig. C.3b).

on simple tasks (e.g. MNIST, CIFAR10), it degrades OOD generalization performance (see Fig. 5.3) and does not significantly improve OOD detection on more complex tasks (see e.g. Camelyon, CIFAR100 in Fig. C.1). Intuitively, maintaining features which are not discriminative to the task might introduce spurious correlations, thus degrading performances. E.g. enforces the architecture to encode the color feature in the latent space decreases the performance of the OOD CMNIST datasets after training on the ID MNIST dataset.

5.5 Prior for DUMs

In this section, we study the effect that the prior component has in DUMs. More specifically, we investigate the relationships between aleatoric uncertainty and the prior specified for DUMs. In particular, this is motivated by Kapoor et al. [214] which shows that using priors that forces model to be confident on the training data points can improve its performance by explicitly accounting for aleatoric uncertainty. To this end, we look at *entropy regularization* defining a training prior in the loss, *prior evidence* and *kernel function* defining a functional prior in the uncertainty head.

Prior. We compare different prior specifications including *entropy regularization* defining a training prior in the loss, *prior evidence* and *kernel function* defining a functional prior in the uncertainty head. Entropy regularization is the entropy term $H(Q)$ in the

Bayesian loss used to train NatPN which encourages a (uniform) prior distributions with high entropy [68]. We control the strength of the regularization factor λ. Further, NatPN also explicitly defines a prior via the parameters χ^{prior} and n^{prior}. While χ^{prior} defines the default categorical prediction via a uniform categorical distribution, the evidence parameter n^{prior} defines the prior number of pseudo-observations and can be varied. Finally, we vary the prior of DUE by using different kernel functions in the learned GP including Matern kernel, RQ kernel, and RBF [356]. *Observation:* Contrary to other Bayesian neural networks [214], we observe that predictive and uncertainty performances of DUMs are not very sensitive to the prior specification (see Figs. 5.5 and C.9 and Table 5.4), thus suggesting a higher robustness to prior mispecification. Nonetheless, a too strong entropy regularization toward an uniform prior degrades more performance of DUMs trained on dataset with low label noise than on high label noise. This suggests that a too high discrepancy between the model prior and the dataset aleatoric uncertainties can impact performance.

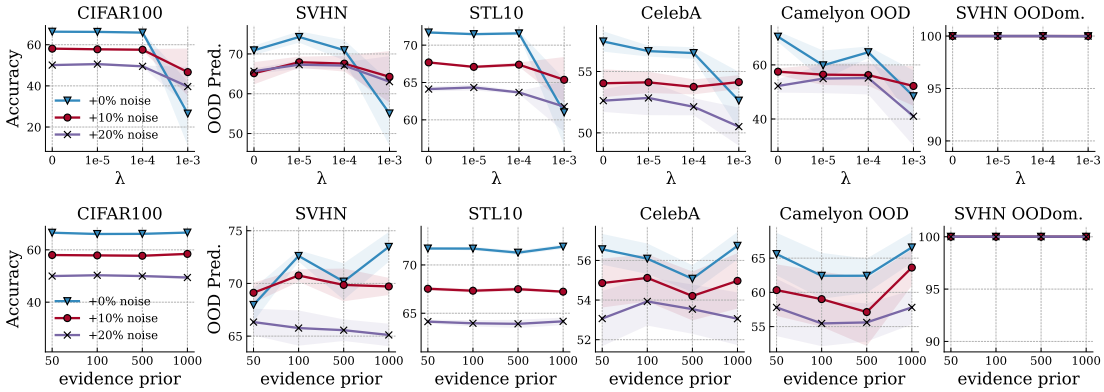


Figure 5.5: Results of enforcing different **prior** in NatPN on CIFAR100 by changing the (top) *entropy regularization* λ and the (bottom) *evidence prior* n^{prior}. Different priors do not lead consistent results improvements.

Table 5.4: Results of enforcing different **prior** in DUE on CIFAR100 and Camelyon by changing the kernel function. OOD results are averaged over OOD datasets. Different priors lead to similar performance.

	CIFAR100			Camelyon		
Kernel	**Accuracy (↑)**	**Brier Score (↓)**	**OOD Pred. (↑)**	**Accuracy (↑)**	**Brier Score (↓)**	**OOD Pred. (↑)**
Matern52	71.80 ± 0.18	41.37 ± 0.24	75.90 ± 1.18	79.81 ± 2.72	32.46 ± 3.22	58.86 ± 6.20
Matern32	71.80 ± 0.21	41.62 ± 0.22	**76.15 ± 1.18**	80.23 ± 2.71	32.77 ± 3.20	58.50 ± 5.83
Matern12	71.70 ± 0.18	43.10 ± 0.22	75.70 ± 1.19	79.30 ± 2.96	32.67 ± 3.28	**59.13 ± 6.39**
RQ	71.83 ± 0.19	**41.16 ± 0.25**	75.93 ± 1.21	80.31 ± 2.55	32.22 ± 3.14	58.69 ± 6.09
RBF	**71.85 ± 0.19**	41.17 ± 0.24	76.14 ± 1.19	**80.45 ± 2.49**	**32.13 ± 3.11**	58.86 ± 5.91

5.6 Conclusion

We investigate important design choice in DUMs. We show that *training* of DUMs can be improved by decoupling the the optimization of the core architecture and the uncertainty

head. We show that expressive core *architecture* can improve DUMs performances. In contrast, additional constraints to avoid feature collapse do not consistently lead to better performance, potentially degrading the OOD generalization and detection trade-off. Finally, we show that the choice of *prior* for DUMs does not lead to important performance improvements.

6 Robustness of Uncertainty Estimation

Uncertainty is the only certainty there is, and knowing how to live with insecurity is the only security.

John Allen Paulos

6.1 Introduction

In Chapter 5, we studied practical considerations when using efficient uncertainty estimation methods in non-adversarial contexts. In this chapter, we now focus on the robustness of uncertainty estimates in adversarial contexts.

Indeed, although neural networks achieve high predictive accuracy in many tasks, they are known to have two substantial weaknesses: First, neural networks are not robust against adversarial perturbations, i.e., semantically meaningless input changes that lead to wrong predictions [403, 163]. Second, standard neural networks are unable to identify samples that are different from the ones they were trained on and tend to make overconfident predictions at test time [246]. These weaknesses make them impracticable in sensitive domains like financial, autonomous driving or medical areas which require trust in predictions.

To increase trust in neural networks, models that provide predictions along with the corresponding uncertainty have been proposed. In previous chapters we have seen PostNet (see Chapter 3) and NatPN (see Chapter 4) which, when applied to classification tasks, belong to the important family of Dirichlet-based uncertainty (DBU) models [271, 277, 374, 273, 67, 471, 310, 386, 375]. Contrary to other approaches such as Bayesian Neural Networks [42, 327, 268], drop out [141] or ensembles [246], DBU models provide efficient uncertainty estimates at test time in a single forward pass by directly predicting the parameters of a Dirichlet distribution over categorical probability distributions. DBU models have the advantage that they provide both, aleatoric uncertainty estimates resulting from irreducible uncertainty (e.g. class overlap or noise) and epistemic uncertainty estimates resulting from the lack of knowledge about unseen data (e.g. an unknown object is presented to the model). Both uncertainty types can be quantified from Dirichlet distributions using different uncertainty measures such as differential entropy, mutual information, or pseudo-counts. These uncertainty measures show outstanding performance in, e.g., the detection of OOD samples and thus are superior to softmax based confidence [271, 273, 67].

Neural networks from the families outlined above are expected to *know what they don't know*, i.e. they are supposed to notice when they are unsure about a prediction.

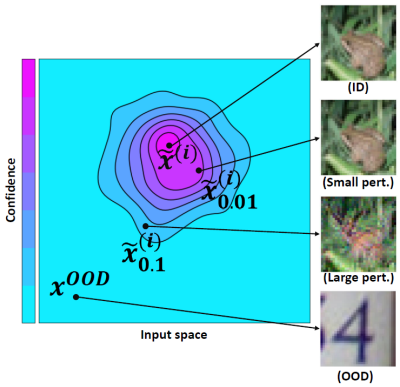


Figure 6.1: Visualization of the desired uncertainty estimates.

This raises questions with regards to adversarial examples: should uncertainty estimation methods *detect* these corrupted samples by indicating a high uncertainty on them and abstain from making a prediction? Or should uncertainty estimation be *robust* to adversarial examples and assign the correct label even under perturbations? We argue that being robust to adversarial perturbations is the best option (see Fig. 6.1) for two reasons. First, in image classification a human is usually not able to observe any difference between an adversarial example and an unperturbed image. Second, the size of the perturbation corresponding to a good adversarial example is typically small w.r.t. the L_p-norm and thus assumed to be semantically meaningless. Importantly, robustness should not only be required for the class predictions, but also for the uncertainty estimates. This means that DBU models should be able to distinguish robustly between ID and OOD data even if those are perturbed.

In this chapter, we focus on DBU models and analyze their robustness capacity w.r.t. class predictions as well as uncertainty predictions. In doing so, we go beyond simple softmax output confidence by investigating advanced uncertainty measures such as differential entropy. Specifically, we study the following questions:

1. *Is low uncertainty a reliable indicator of correct predictions?*
2. *Can we use uncertainty estimates to detect label attacks on the class prediction?*
3. *Are uncertainty estimates such as differential entropy a robust feature for OOD detection?*

In addressing these questions we place particular focus on adversarial perturbations of the input to evaluate the *worst case* performance of the models on increasing complex data sets and attacks. We evaluate robustness of DBU models w.r.t. to these three questions by comparing their performance on unperturbed and perturbed inputs. Perturbed inputs are obtained by computing *label attacks* and *uncertainty attacks*, which are a new type of attacks we propose. While label attacks aim at changing the class prediction, uncertainty attacks aim at changing the uncertainty estimate such that ID data is marked as OOD data and vice versa. In total, we performed more than $138,800$ attack settings

to explore the robustness landscape of DBU models. Those settings cover different data sets, attack types, attack losses, attack radii, DBU model types and initialisation seeds. Finally, we propose and evaluate median smoothing and adversarial training based on label attacks and uncertainty attacks to make DBU models more robust. Our median smoothing approach provides certificates on epistemic uncertainty measures such as differential entropy and allows to certify uncertainty estimation. The code and further supplementary material is available online (`www.daml.in.tum.de/dbu-robustness`).

6.2 Related work

The existence of adversarial examples is a problematic property of neural networks [403, 163]. Previous works have study this phenomena by proposing adversarial attacks [63, 53, 477], defenses [82, 171] and verification techniques [446, 391, 89, 47, 234]. This includes the study of different settings such as i.i.d. inputs, sequential inputs and graphs [472, 46, 75, 371].

In the context of uncertainty estimation, robustness of the class prediction has been studied in previous works for Bayesian Neural Networks [42, 327, 268], drop out [141] or ensembles [246] focusing on data set shifts [335] or adversarial attacks [60, 61, 441]. Despite their efficient and high quality uncertainty estimates, the robustness of DBU models has not been investigated in detail yet — indeed only for one single DBU model, [273] has briefly performed attacks aiming to change the label. In contrast, our work focuses on a large variety of DBU models and analyzes two robustness properties: robustness of the class prediction w.r.t. adversarial perturbations and robustness of uncertainty estimation w.r.t. our newly proposed attacks against uncertainty measures.

This so called *uncertainty attack* directly targets uncertainty estimation and are different from traditional *label attacks*, which target the class prediction [269, 97]. They allow us to jointly evaluate robustness of the class prediction and robustness of uncertainty estimation. This goes beyond previous attack defenses that were either focused on evaluating *robustness w.r.t. class predictions* [63, 435] or detecting attacks against the class prediction [62].

Different models have been proposed to account for uncertainty while being robust. [392] and [253] have tried to improve label attack detection based on uncertainty using drop-out or density estimation. In addition to improving label attack detection for large unseen perturbations, [400] aimed at improving robustness w.r.t. class label predictions on small input perturbations. They used adversarial training and soft labels for adversarial samples further from the original input. [349] suggested a similar adversarial training procedure, that softens labels depending on the input robustness. These previous works consider the aleatoric uncertainty that is contained in the predicted categorical probabilities, but in contrast to DBU models they are not capable of taking epistemic uncertainty into account.

Recently, four studies tried to obtain certificates on aleatoric uncertainty estimates. [404] and [244] compute confidence intervals and certificates on softmax predictions. [39] uses interval bound propagation to compute bounds on softmax predictions within

the L_∞-ball around an OOD sample using ReLU networks. [286] focuses on obtaining certifiably low confidence for OOD data. These four studies estimate confidence based on softmax predictions, which accounts for aleatoric uncertainty only. In this chapter, we provide certificates which apply for all uncertainty measures. In particular, we use our certificates on epistemic uncertainty measures such as differential entropy which are well suited for OOD detection.

6.3 Dirichlet-based uncertainty models

Standard (softmax) neural networks predict the parameters of a categorical distribution $\boldsymbol{p}^{(i)} = [p_1^{(i)}, \dots, p_C^{(i)}]$ for a given input $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$, where C is the number of classes. Given the parameters of a categorical distribution, the *aleatoric uncertainty* can be evaluated. The aleatoric uncertainty is the uncertainty on the class label prediction $y^{(i)} \in \{1, \dots, C\}$. For example if we predict the outcome of an unbiased coin flip, the model is expected to have high aleatoric uncertainty and predict $p(\text{head}) = 0.5$.

	$\alpha^{(i)}$-parametrization	Loss	OOD training data	Ensemble training	Density estimation
Posterior Network	$f_\theta(\boldsymbol{x}^{(i)}) = \mathbf{1} + \boldsymbol{\alpha}^{(i)}$	Bayesian loss	No	No	Yes
PriorNet	$f_\theta(\boldsymbol{x}^{(i)}) = \boldsymbol{\alpha}^{(i)}$	Reverse KL	Yes	No	No
DDNet	$f_\theta(\boldsymbol{x}^{(i)}) = \boldsymbol{\alpha}^{(i)}$	Dir. Likelihood	No	Yes	No
EvNet	$f_\theta(\boldsymbol{x}^{(i)}) = \mathbf{1} + \boldsymbol{\alpha}^{(i)}$	Expected MSE	No	No	No

Table 6.1: Summary of DBU models. Further details on the loss functions are provided in the appendix.

In contrast to standard (softmax) neural networks, DBU models predict the parameters of a Dirichlet distribution – the natural prior of categorical distributions – given input $\boldsymbol{x}^{(i)}$ (i.e. $q^{(i)} = \text{Dir}(\boldsymbol{\alpha}^{(i)})$ where $f_\theta(\boldsymbol{x}^{(i)}) = \boldsymbol{\alpha}^{(i)} \in \mathbb{R}_+^C$). Hence, the *epistemic distribution* $q^{(i)}$ expresses the *epistemic* uncertainty on $\boldsymbol{x}^{(i)}$, i.e. the uncertainty on the categorical distribution prediction $\boldsymbol{p}^{(i)}$. From the epistemic distribution, follows an estimate of the *aleatoric distribution* of the class label prediction $\text{Cat}(\bar{\boldsymbol{p}}^{(i)})$ where $\mathbb{E}_{q^{(i)}}[\boldsymbol{p}^{(i)}] = \bar{\boldsymbol{p}}^{(i)}$. An advantage of DBU models is that one pass through the neural network is sufficient to compute epistemic distribution, aleatoric distribution, and predict the class label:

$$q^{(i)} = \text{Dir}(\boldsymbol{\alpha}^{(i)}), \;\; \bar{p}_c^{(i)} = \frac{\alpha_c^{(i)}}{\alpha_0^{(i)}}, \;\; y^{(i)} = \arg\max_c \, [\bar{p}_c^{(i)}] \tag{6.1}$$

where $\alpha_0^{(i)} = \sum_{c=1}^C \alpha_c^{(i)}$. This parametrization allows to compute classic uncertainty measures in closed-form such as the total pseudo-count $m_{\alpha_0}^{(i)} = \sum_c \alpha_c^{(i)}$, the differential entropy of the Dirichlet distribution $m_{\text{diffE}}^{(i)} = h(\text{Dir}(\boldsymbol{\alpha}^{(i)}))$ or the mutual information $m_{\text{MI}}^{(i)} = I(y^{(i)}, \boldsymbol{p}^{(i)})$ (Appendix D.1.1, [271]). Hence, these measure can efficiently be used to assign high uncertainty to unknown data, which makes DBU models specifically suited for detection of OOD samples.

Several recently proposed models for uncertainty estimations belong to the family of DBU models, such as PriorNet, EvNet, DDNet and Posterior Network. These models

differ in terms of their parametrization of the Dirichlet distribution, the training, and density estimation. An overview of theses differences is provided in Table 6.1. In our study we evaluate all recent versions of these models.

Contrary to the other models, Prior Networks **(PriorNet)** [271, 273] require OOD data for training to "teach" the neural network the difference between ID and OOD data. PriorNet is trained with a loss function consisting of two KL-divergence terms. The fist term is designed to learn Dirichlet parameters for ID data, while the second one is used to learn a flat Dirichlet distribution for OOD data:

$$L_{\text{PriorNet}} = \frac{1}{N}\left[\sum_{\boldsymbol{x}^{(i)}\in\text{ID data}} [\text{KL}[\text{Dir}(\alpha^{\text{ID}})||q^{(i)}]] + \sum_{\boldsymbol{x}^{(i)}\in OODdata} [\text{KL}[\text{Dir}(\alpha^{\text{OOD}})||q^{(i)}]]\right] \tag{6.2}$$

where α^{ID} and α^{OOD} are hyper-parameters. Usually α^{ID} is set to $1e^1$ for the correct class and 1 for all other classes, while α^{OOD} is set to $\mathbf{1}$ for all classes. There a two variants of PriorNet. The first one is trained based on reverse KL-divergence [273], while the second one is trained with KL-divergence [271]. In our experiments, we include the most recent reverse version of PriorNet, as it shows superior performance [277].

Evidential Networks **(EvNet)** [374] are trained with a loss that computes the sum of squares between the on-hot encoded true label $\boldsymbol{y}*^{(i)}$ and the predicted categorical $\boldsymbol{p}^{(i)}$ under the Dirichlet distribution:

$$L_{\text{EvNet}} = \frac{1}{N}\sum_i \mathbb{E}_{\boldsymbol{p}^{(i)}\sim\text{Dir}(\boldsymbol{\alpha}^{(i)})}||\boldsymbol{y}*^{(i)} - \boldsymbol{p}^{(i)}||^2 \tag{6.3}$$

Ensemble Distribution Distillation **(DDNet)** [277] is trained in two steps. First, an ensemble of M classic neural networks needs to be trained. Then, the soft-labels $\{\boldsymbol{p}_m^{(i)}\}_{m=1}^M$ provided by the ensemble of networks are distilled into a Dirichlet-based network by fitting them with the maximum likelihood under the Dirichlet distribution:

$$L_{\text{DDNet}} = -\frac{1}{N}\sum_i \sum_{m=1}^{M} [\ln q^{(i)}(\pi^{im})] \tag{6.4}$$

where π^{im} denotes the soft-label of mth neural network.

Posterior Network **(PostNet)** [67] performs density estimation for ID data with normalizing flows and uses a Bayesian loss formulation:

$$L_{\text{PosteriorNetwork}} = \frac{1}{N}\sum_i \mathbb{E}_{q(p^{(i)})}[\text{CE}(p^{(i)}, y^{(i)})] - H(q^{(i)}) \tag{6.5}$$

where CE denotes the cross-entropy. All loss functions can be computed in closed-form. For more details please have a look at the original paper on PriorNet [271], Posterior Network [67], DDNet [277] and EvNet [374]. Note that EvNet and Posterior Network model the Dirichlet parameters as $f_\theta(\boldsymbol{x}^{(i)}) = 1 + \boldsymbol{\alpha}^{(i)}$ while PriorNet, RevPriorNet and DDNet compute them as $f_\theta(\boldsymbol{x}^{(i)}) = \boldsymbol{\alpha}^{(i)}$.

6.4 Robustness of Dirichlet-based uncertainty models

We analyze robustness of DBU models on tasks in connection with uncertainty estimation w.r.t. the following four aspects: *accuracy*, *confidence calibration*, *label attack detection* and *OOD detection*. Uncertainty is quantified by differential entropy, mutual information or pseudo counts. A formal definition of all uncertainty estimation measures is provided in the appendix (see Appendix D.1.1).

Robustness of Dirichlet-based uncertainty models is evaluated based on *label attacks* and a newly proposed type of attacks called *uncertainty attacks*. While label attacks aim at changing the predicted class, uncertainty attacks aim at changing the uncertainty assigned to a prediction. All previous works are based on label attacks and focus on robustness w.r.t. the class prediction. Thus, we are the first to propose attacks targeting uncertainty estimates such as differential entropy and analyze desirable robustness properties of DBU models beyond the class prediction. Label attacks and uncertainty attacks both compute a perturbed input $\tilde{\boldsymbol{x}}^{(i)}$ close to the original input $\boldsymbol{x}^{(i)}$ i.e. $||\boldsymbol{x}^{(i)} - \tilde{\boldsymbol{x}}^{(i)}||_2 < r$ where r is the attack radius. This perturbed input is obtained by optimizing a loss function $l(\boldsymbol{x})$ using Fast Gradient Sign Method (FGSM) or Projected Gradient Descent (PGD). Furthermore, we include a black box attack setting (Noise) which generates 10 noise samples from a Gaussian distribution, which is centered at the original input. From these 10 perturbed samples we choose the one with the greatest effect on the loss function and use it as attack. To complement attacks, we compute certificates on uncertainty estimates using median smoothing [455].

The following questions we address by our experiments have a common assessment metric and can be treated as binary classification problems: distinguishing between correctly and wrongly classified samples, discriminating between non-attacked input and attacked inputs or differentiating between ID data and OOD data. To quantify the performance of the models on these binary classification problems, we compute the area under the precision recall curve (AUC-PR).

Experiments are performed on two image data sets (MNIST [249] and CIFAR10 [238]), which contain bounded inputs and two tabular data sets (Segment [111] and Sensorless drive [111]), consisting of unbounded inputs. Note that unbounded inputs are challenging since it is impossible to describe the infinitely large OOD distribution. As PriorNet requires OOD training data, we use two further image data sets (FashionMNIST [450] and CIFAR100 [238]) for training on MNIST and CIFAR10, respectively. All other models are trained without OOD data. To obtain OOD data for the tabular data sets, we remove classes from the ID data set (class window for the Segment data set and class 9 for Sensorless drive) and use them as the OOD data. Further details on the experimental setup are provided in the appendix (see Appendix D.1.2).

6.4.1 Uncertainty estimation under label attacks

Label attacks aim at changing the predicted class. To obtain a perturbed input with a different label, we maximize the cross-entropy loss $\tilde{\boldsymbol{x}}^{(i)} \approx \arg\max_{\boldsymbol{x}} l(\boldsymbol{x}) = \mathrm{CE}(\boldsymbol{p}^{(i)}, \boldsymbol{y}^{(i)})$ under the radius constraint. For the sake of completeness we additionally analyze label

attacks w.r.t. to their performance of changing the class prediction and the accuracy of the neural network under label attacks constraint by different radii (see Appendix, Table D.1). As expected and partially shown by previous works, none of the DBU models is robust against label attacks. However, we note that PriorNet is slightly more robust than the other DBU models. This might be explained by the use of OOD data during training, which can be seen as some kind of robust training. From now on, we switch to the core focus of this chapter and analyze robustness properties of uncertainty estimation.

	CIFAR10						Sensorless					
Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	0.0	0.1	0.2	0.5	1.0	2.0
PostNet	**98.7**	88.6	56.2	7.8	1.2	0.4	99.7	8.3	3.9	3.6	**7.0**	**9.8**
PriorNet	92.9	77.7	60.5	**37.6**	**24.9**	**11.3**	99.8	10.5	3.2	0.7	0.2	0.2
DDNet	97.6	**91.8**	**78.3**	18.1	0.8	0.0	99.7	11.9	1.6	0.4	0.2	0.1
EvNet	97.9	85.9	57.2	10.2	4.0	2.4	**99.9**	**22.9**	**13.0**	**6.0**	3.7	3.2

Table 6.2: Distinguishing between correctly predicted and wrongly predicted labels based on the differential entropy under PGD label attacks (metric: AUC-PR).

Is low uncertainty a reliable indicator of correct predictions?

Expected behavior: Predictions with low uncertainty are more likely to be correct than high uncertainty predictions. *Assessment metric:* We distinguish between correctly classified samples (label 0) and wrongly classified ones (label 1) based on the differential entropy scores produced by the DBU models [271]. Correctly classified samples are expected to have low differential entropy, reflecting the model's confidence, and analogously wrongly predicted samples are expected to have higher differential entropy. *Observed behavior:* Note that the positive and negative class are not balanced, thus, the use of AUC-PR scores [369] are important to enable meaningful measures. While uncertainty estimates are indeed an indicator of correctly classified samples on unperturbed data, none of the models maintains its high performance on perturbed data computed by PGD, FGSM or Noise label attacks (see. Tables 6.2, D.7 and D.8). Thus, using uncertainty estimates as indicator for correctly labeled inputs is not robust to adversarial perturbations. This result is notable, since the used attacks do not target uncertainty.

	CIFAR10					Sensorless				
Att. Rad.	0.1	0.2	0.5	1.0	2.0	0.1	0.2	0.5	1.0	2.0
PostNet	**63.4**	**66.9**	42.1	32.9	31.6	47.7	42.3	36.9	**48.5**	**85.0**
PriorNet	53.3	56.0	55.6	**49.2**	42.2	38.8	33.6	31.4	33.1	40.9
DDNet	55.8	60.5	**57.3**	38.7	32.3	**53.5**	42.2	35.0	32.8	32.6
EvNet	48.4	46.9	46.3	46.3	**44.5**	48.2	**42.6**	**38.2**	36.0	37.2

Table 6.3: Label Attack-Detection by normally trained DBU models based on differential entropy under PGD label attacks (AUC-PR).

Can uncertainty estimates be used to detect label attacks against the class prediction?

Expected behavior: Adversarial examples are not from the natural data distribution. Therefore, DBU models are expected to detect them as OOD data by assigning them a higher uncertainty. We expect that perturbations computed based on a bigger attack radius r are easier to detect as their distance from the data distribution is larger. *Assessment metric:* The goal of attack-detection is to distinguish between unperturbed samples (label 0) and perturbed samples (label 1). Uncertainty on samples is quantified by the differential uncertainty [271]. Unperturbed samples are expected to have low differential entropy, because they are from the same distribution as the training data, while perturbed samples are expected to have a high differential entropy. *Observed behavior:* Table D.1 shows that the accuracy of all models decreases significantly under PGD label attacks, but none of the models is able to provide an equivalently increasing attack detection rate (see Table 6.3). Even larger perturbations are hard to detect for DBU models.

Similar results are obtained when we use mutual information or the precision α_0 to quantify uncertainty (see appendix Tables D.5 and D.6). Although PGD label attacks do not explicitly consider uncertainty, they seem to generate adversarial examples with similar uncertainty as the original input. Such high-certainty adversarial examples are illustrated in Fig. 6.2, where certainty is visualized based on the precision α_0, which is supposed to be high for ID data and low for OOD data. While the original input (perturbation size 0.0) is correctly classified as frog and ID data, there exist adversarial examples that are classified as deer or bird. The certainty (α_0-score) on the prediction of these adversarial examples has a similar or even higher value than on the prediction of the original input. Using the differential entropy to distinguish between ID and OOD data results in the same ID/OOD assignment since the differential entropy of the three right-most adversarial examples is similar or even smaller than on the unperturbed input.

Under the less powerful FGSM and Noise attacks (see Appendix), DBU models achieve mostly higher attack detection rates than under PGD attacks. This suggests that uncertainty estimation is able to detect weak attacks, which is consistent with the observations in [272] but fails under stronger PGD attacks.

On tabular data sets, PostNet shows a better label attack detection rate for large perturbations. This observation might be explained by the fact that the density estimation of the ID samples has been shown to work better for tabular data sets [67]. Overall, none of the DBU models provides a reliable indicator for adversarial inputs that target the class prediction.

6.4.2 Attacking uncertainty estimation

DBU models are designed to provide sophisticated uncertainty estimates (beyond softmax scores) alongside predictions and use them to detect OOD samples. In this section, we propose and analyze a new attack type that targets these uncertainty estimates. DBU models enable us to compute uncertainty measures i.e. differential entropy, mutual

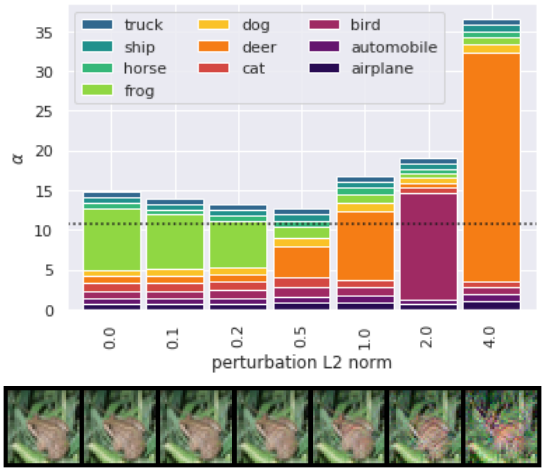


Figure 6.2: Input and predicted Dirichlet-parameters under label attacks (dotted line: threshold to distinguish ID and OOD data).

information and precision α_0 in closed from (see [271] for a derivation). Uncertainty attacks use this closed form solution as loss function for PGD, FGSM or Noise attacks. Since differential entropy is the most widely used metric for ID-OOD-differentiation, we present results based on the differential entropy loss function $\tilde{\boldsymbol{x}}^{(i)} \approx \arg\max_{\boldsymbol{x}} l(\boldsymbol{x}) = \text{Diff-E}(\text{Dir}(\alpha^{(i)}))$:

$$\begin{aligned}\text{Diff-E}(\text{Dir}(\alpha^{(i)})) = &\sum_{c}^{K} \ln\Gamma(\alpha_c^{(i)}) - \ln\Gamma(\alpha_0^{(i)}) \\ &- \sum_{c}^{K} (\alpha_c^{(i)} - 1) \cdot (\Psi(\alpha_c^{(i)}) - \Psi(\alpha_0^{(i)}))\end{aligned} \tag{6.6}$$

where $\alpha_0^{(i)} = \sum_c \alpha_c^{(i)}$. Result based on further uncertainty measures, loss functions and more details on attacks are provided in the appendix.

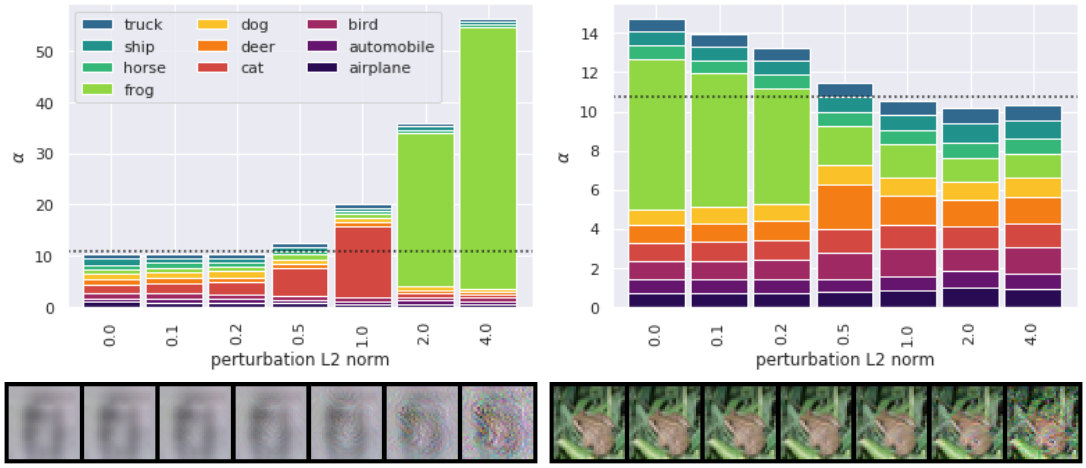
We analyze the performance of DBU models under uncertainty attacks w.r.t. two tasks. First, uncertainty attacks are computed on ID data aiming to indicate it as OOD data, while OOD data is left non-attacked. Second, we attack OOD data aiming to indicate it as ID data, while ID data is not attacked. Hence, uncertainty attacks target at posing ID data as OOD data and vice versa.

Are uncertainty estimates a robust feature for OOD detection?

Expected behavior: We expect DBU models to be able to distinguish between ID and OOD data by providing reliable uncertainty estimates, even under small perturbations. Thus, we expect uncertainty estimates of DBU models to be robust under attacks. *Assessment metric:* We distinguish between ID data (label 0) and OOD data (label 1) based on the differential entropy as uncertainty scoring function [271]. Differential

	ID-Attack (non-attacked OOD)						OOD-Attack (non-attacked ID)					
Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	0.0	0.1	0.2	0.5	1.0	2.0
CIFAR10 – SVHN												
PostNet	81.8	64.3	47.2	22.4	17.6	**16.9**	81.8	60.5	40.7	23.3	21.8	19.8
PriorNet	54.4	40.1	30.0	17.9	15.6	15.4	54.4	40.7	30.7	19.5	16.5	15.7
DDNet	**82.8**	**71.4**	**59.2**	**28.9**	16.0	15.4	**82.8**	**72.0**	**57.2**	20.8	15.6	15.4
EvNet	80.3	62.4	45.4	21.7	**17.9**	16.5	80.3	58.2	46.5	**34.6**	**28.0**	**23.9**
Sens. – Sens. class 10, 11												
PostNet	**74.5**	**39.8**	**36.1**	**36.0**	**45.9**	**46.0**	**74.5**	**43.3**	**42.0**	**32.1**	**35.1**	**82.6**
PriorNet	32.3	26.6	26.5	26.5	26.6	28.3	32.3	26.7	26.6	26.6	27.0	30.4
DDNet	31.7	26.8	26.6	26.5	26.6	27.1	31.7	27.1	26.7	26.7	26.8	26.9
EvNet	66.5	30.5	28.2	27.1	28.1	31.8	66.5	38.7	36.1	30.2	28.2	28.8

Table 6.4: OOD detection based on differential entropy under PGD uncertainty attacks against differential entropy computed on ID data and OOD data (metric: AUC-PR).



(a) OOD uncertainty attack **(b)** ID uncertainty attack

Figure 6.3: ID and OOD input with corresponding Dirichlet-parameters under uncertainty attacks (dotted line: threshold to distinguish ID and OOD).

entropy is expected to be small on ID samples and high on OOD samples. Experiments on further uncertainty measure and results on the AUROC metric are provided in the appendix. *Observed behavior:* OOD samples are perturbed as illustrated in Fig. 6.3. Part (a) of the figure illustrates an OOD-samples, that is correctly identified as OOD. Adding adversarial perturbations ≥ 0.5 changes the Dirichlet parameters such that the resulting images are identified as ID, based on precision or differential entropy as uncertainty measure. Perturbing an ID sample (part (b)) results in images that are marked as OOD samples. OOD detection performance of all DBU models rapidly decreases with the size of the perturbation, regardless of whether attacks are computed on ID or OOD data (see Table 6.4). This performance decrease is also observed with AUROC as metric, attacks

based on FGSM, Noise, when we use mutual information or precision α_0 to distinguish between ID samples and OOD samples (see appendix Tables D.14 to D.21). Thus, using uncertainty estimation to distinguish between ID and OOD data is not robust.

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
	PostNet	80.5 · **91.5** · 94.5	52.8 · **71.6** · 95.2	31.9 · **51.0** · 96.8	5.6 · **11.7** · 100.0	0.3 · **0.6** · 100.0	0.0 · **0.0** · 100.0
Smoothed	**PriorNet**	81.9 · **86.8** · 88.0	69.6 · **78.0** · 90.1	50.9 · **65.8** · 89.4	36.5 · **59.9** · 97.0	24.3 · **39.3** · 100.0	9.2 · **17.9** · 100.0
models	**DDNet**	65.9 · **81.2** · 83.0	55.8 · **70.5** · 87.2	37.8 · **56.8** · 88.1	10.1 · **21.9** · 94.3	0.9 · **1.6** · 99.6	0.0 · **0.0** · 100.0
	EvNet	76.3 · **90.2** · 91.7	54.7 · **74.3** · 95.7	31.6 · **51.5** · 94.5	5.8 · **11.9** · 86.9	1.9 · **7.0** · 100.0	1.1 · **4.0** · 100.0
Smoothed	**PostNet**	-	52.1 · **71.8** · 95.6	31.2 · **47.9** · 96.1	7.8 · **14.7** · 98.6	1.8 · **4.4** · 100.0	0.3 · **0.5** · 100.0
+ adv.	**PriorNet**	-	57.6 · **71.7** · 88.9	46.1 · **64.5** · 90.1	38.1 · **59.3** · 99.5	32.3 · **51.7** · 100.0	22.1 · **41.6** · 97.4
label	**DDNet**	-	58.6 · **78.4** · 92.2	49.4 · **66.0** · 90.5	12.0 · **21.4** · 98.1	0.8 · **1.0** · 96.6	0.0 · **0.0** · 100.0
attacks	**EvNet**	-	24.3 · **34.2** · 51.8	32.6 · **49.5** · 95.5	5.9 · **13.0** · 100.0	2.6 · **5.2** · 99.9	2.9 · **5.9** · 100.0
Smoothed	**PostNet**	-	52.8 · **74.2** · 94.6	33.0 · **49.4** · 87.5	7.7 · **14.2** · 99.0	0.6 · **1.2** · 100.0	0.7 · **1.1** · 100.0
+ adv.	**PriorNet**	-	50.6 · **68.1** · 88.6	44.4 · **66.1** · 96.0	35.1 · **57.4** · 98.4	18.4 · **32.2** · 100.0	15.2 · **29.3** · 100.0
uncert.	**DDNet**	-	68.8 · **84.4** · 93.2	45.1 · **60.8** · 86.8	12.3 · **22.0** · 91.0	0.8 · **1.7** · 87.0	0.0 · **0.0** · 100.0
attacks	**EvNet**	-	54.2 · **73.7** · 96.1	30.5 · **50.0** · 99.5	7.1 · **13.9** · 100.0	3.7 · **8.7** · 75.2	3.3 · **5.8** · 100.0

Table 6.5: Distinguishing between correctly and wrongly predicted labels based on differential entropy under PGD label attacks. Smoothed DBU models on CIFAR10. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.1	0.2	0.5	1.0	2.0
	PostNet	33.1 · **50.4** · 89.9	31.0 · **50.2** · 96.9	30.7 · **50.2** · 100.0	30.7 · **50.0** · 100.0	30.7 · **50.2** · 100.0
Smoothed	**PriorNet**	35.9 · **50.6** · 74.5	33.0 · **50.3** · 82.8	31.2 · **50.0** · 95.7	30.7 · **50.4** · 99.9	30.7 · **50.4** · 100.0
models	**DDNet**	36.3 · **50.3** · 76.4	32.8 · **49.9** · 84.6	30.8 · **50.1** · 98.0	30.7 · **50.2** · 100.0	30.7 · **50.2** · 100.0
	EvNet	32.9 · **50.4** · 89.8	31.4 · **50.1** · 94.0	30.8 · **50.0** · 98.0	30.7 · **50.3** · 100.0	30.7 · **49.6** · 100.0
Smoothed	**PostNet**	32.7 · **50.1** · 90.4	31.1 · **50.2** · 96.5	30.7 · **50.2** · 99.7	30.7 · **50.3** · 100.0	30.7 · **50.2** · 100.0
+ adv.	**PriorNet**	35.2 · **51.8** · 78.6	32.8 · **51.1** · 84.4	30.8 · **50.2** · 98.7	30.7 · **50.5** · 100.0	30.8 · **50.1** · 98.2
label	**DDNet**	35.5 · **50.6** · 79.2	33.4 · **50.3** · 84.1	30.8 · **50.1** · 99.2	30.7 · **50.0** · 100.0	30.7 · **50.5** · 100.0
attacks	**EvNet**	40.3 · **50.4** · 66.8	31.4 · **50.3** · 95.8	30.7 · **50.3** · 100.0	30.7 · **50.1** · 100.0	30.7 · **50.0** · 100.0
Smoothed	**PostNet**	33.3 · **50.6** · 88.7	32.5 · **50.1** · 87.9	30.7 · **49.9** · 99.8	30.7 · **50.1** · 100.0	30.7 · **50.0** · 100.0
+ adv.	**PriorNet**	34.5 · **51.0** · 80.1	31.4 · **50.6** · 92.8	30.9 · **50.0** · 97.7	30.7 · **50.1** · 100.0	30.7 · **50.0** · 100.0
uncert.	**DDNet**	37.4 · **50.8** · 74.5	33.4 · **50.2** · 83.0	30.9 · **50.1** · 96.8	30.8 · **49.9** · 98.1	30.7 · **49.9** · 100.0
attacks	**EvNet**	32.8 · **50.1** · 92.0	30.8 · **50.0** · 99.6	30.7 · **50.1** · 100.0	31.2 · **50.2** · 96.1	31.0 · **50.0** · 100.0

Table 6.6: Attack detection (PGD label attacks) based on differential entropy. Smoothed DBU models on CIFAR10. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

6.4.3 How to make DBU models more robust?

Our robustness analysis based on label attacks and uncertainty attacks shows that predictions, uncertainty estimation and the differentiation between ID and OOD data are not

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
				ID-Attack			
Smoothed models	**PostNet**	72.1 · **82.7** · 88.0	35.0 · **56.6** · 97.4	31.9 · **65.6** · 99.8	30.7 · **50.6** · 100.0	30.7 · **46.9** · 100.0	30.7 · **51.6** · 100.0
	PriorNet	50.2 · **53.1** · 55.9	33.5 · **43.3** · 65.3	31.3 · **39.7** · 69.1	31.3 · **48.3** · 98.2	30.7 · **44.4** · 99.9	30.7 · **45.4** · 100.0
	DDNet	72.0 · **75.8** · 79.8	35.6 · **46.2** · 69.8	32.9 · **50.3** · 87.1	31.1 · **58.7** · 98.6	30.7 · **59.3** · 100.0	30.7 · **44.5** · 100.0
	EvNet	79.5 · **87.1** · 92.8	34.1 · **58.6** · 95.1	32.5 · **61.2** · 96.9	31.7 · **60.6** · 98.7	30.7 · **62.4** · 100.0	30.7 · **57.3** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	35.0 · **58.5** · 97.7	31.2 · **46.6** · 97.4	30.8 · **57.7** · 99.7	30.7 · **49.8** · 100.0	30.7 · **50.9** · 100.0
	PriorNet	-	31.5 · **36.7** · 57.2	33.1 · **51.8** · 84.8	30.7 · **57.7** · 98.7	30.7 · **40.0** · 99.9	30.9 · **53.6** · 96.7
	DDNet	-	36.2 · **50.0** · 78.6	32.1 · **41.3** · 70.2	30.8 · **56.4** · 100.0	30.7 · **49.4** · 100.0	30.7 · **54.8** · 100.0
	EvNet	-	46.8 · **61.0** · 79.7	32.3 · **58.9** · 99.1	30.7 · **45.0** · 100.0	30.7 · **63.3** · 100.0	30.8 · **38.1** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	35.2 · **55.9** · 96.0	34.5 · **59.2** · 94.9	30.7 · **47.0** · 100.0	30.7 · **58.2** · 100.0	30.7 · **42.9** · 100.0
	PriorNet	-	31.8 · **38.9** · 64.1	31.0 · **41.8** · 87.9	30.7 · **42.9** · 99.2	30.7 · **48.6** · 100.0	30.7 · **46.6** · 100.0
	DDNet	-	39.7 · **52.1** · 75.7	36.4 · **56.8** · 83.8	31.0 · **51.5** · 97.4	31.0 · **56.8** · 97.8	30.7 · **49.1** · 100.0
	EvNet	-	34.8 · **64.9** · 99.6	30.8 · **48.9** · 99.8	30.7 · **66.8** · 100.0	30.9 · **41.5** · 93.8	31.1 · **55.1** · 100.0
				OOD-Attack			
Smoothed models	**PostNet**	72.0 · **82.7** · 88.0	35.1 · **56.8** · 97.3	32.0 · **65.8** · 99.8	30.7 · **50.7** · 100.0	30.7 · **46.5** · 100.0	30.7 · **51.7** · 100.0
	PriorNet	50.3 · **53.1** · 55.9	33.6 · **43.7** · 65.9	31.3 · **39.8** · 69.4	31.3 · **48.3** · 98.2	30.7 · **44.5** · 99.9	30.7 · **46.4** · 100.0
	DDNet	72.0 · **75.8** · 79.8	35.6 · **46.2** · 70.0	32.9 · **50.1** · 86.7	31.1 · **58.8** · 98.6	30.7 · **59.3** · 100.0	30.7 · **44.6** · 100.0
	EvNet	79.5 · **87.1** · 92.8	34.1 · **58.8** · 95.2	32.6 · **61.2** · 96.9	31.7 · **60.5** · 98.7	30.7 · **62.4** · 100.0	30.7 · **57.6** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	35.0 · **58.5** · 97.8	31.2 · **46.6** · 97.2	30.8 · **57.7** · 99.7	30.7 · **50.2** · 100.0	30.7 · **51.5** · 100.0
	PriorNet	-	31.6 · **37.3** · 59.3	33.2 · **52.7** · 85.8	30.7 · **57.8** · 98.7	30.7 · **40.1** · 99.9	30.9 · **53.8** · 96.8
	DDNet	-	36.4 · **50.2** · 78.9	32.1 · **41.5** · 70.4	30.9 · **56.2** · 100.0	30.7 · **49.3** · 100.0	30.7 · **55.1** · 100.0
	EvNet	-	47.2 · **61.1** · 80.0	32.4 · **59.1** · 99.1	30.7 · **45.0** · 100.0	30.7 · **63.2** · 100.0	30.8 · **38.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	35.3 · **56.4** · 96.1	34.5 · **59.0** · 94.9	30.7 · **46.8** · 100.0	30.7 · **57.8** · 100.0	30.7 · **43.2** · 100.0
	PriorNet	-	31.9 · **39.4** · 65.5	31.0 · **42.0** · 88.6	30.7 · **42.9** · 99.2	30.7 · **48.4** · 100.0	30.7 · **47.1** · 100.0
	DDNet	-	40.2 · **52.9** · 76.5	36.4 · **56.9** · 83.9	31.1 · **51.5** · 97.3	31.0 · **57.0** · 97.8	30.7 · **49.1** · 100.0
	EvNet	-	34.9 · **64.8** · 99.6	30.8 · **48.8** · 99.8	30.7 · **66.1** · 100.0	30.9 · **41.6** · 93.6	31.1 · **54.7** · 100.0

Table 6.7: OOD detection based on differential entropy under PGD uncertainty attacks against differential entropy on ID data and OOD data. Smoothed DBU models on CIFAR10. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

robust. Next, we explore approaches to improve robustness properties of DBU models w.r.t. these tasks based on randomized smoothing and adversarial training.

Randomized smoothing was originally proposed for certification of classifiers [89]. The core idea is to draw multiple samples $\boldsymbol{x}_s^{(i)} \sim \mathcal{N}(\boldsymbol{x}^{(i)}, \sigma)$ around the input data $\boldsymbol{x}^{(i)}$, to feed all these samples through the neural network, and to aggregate the resulting set of predictions (e.g. by taking their mean), to get a smoothed prediction. Besides allowing certification, as a side effect, the smoothed model is more robust. Our idea is to use randomized smoothing to improve robustness of DBU models, particularly w.r.t. uncertainty estimation. In contrast to discrete class predictions, however, certifying uncertainty estimates such as differential entropy scores requires a smoothing approach that is able to handle continuous values as in regression tasks. So far, only few works for randomized smoothing for regression models have been proposed [244, 455]. We choose median smoothing [455], because it is applicable to unbounded domains as required for

the uncertainty estimates covered in this chapter. In simple words: The set of uncertainty scores obtained from the $\boldsymbol{x}_s^{(i)} \sim \mathcal{N}(\boldsymbol{x}^{(i)}, \sigma)$ is aggregated by taking their median.

In the following experiments we focus on differential entropy as the uncertainty score. We denote the resulting smoothed differential entropy, i.e. the median output, as $m(\boldsymbol{x}^{(i)})$. Intuitively, we expect that the random sampling around a data point as well as the outlier-insensitivity of the median to improve the robustness of the uncertainty estimates w.r.t. adversarial examples.

To measure the performance and robustness of our smoothed DBU models, we apply median smoothing on the same tasks as in the previous sections, i.e., distinguishing between correctly and wrongly labeled inputs, attack detection, OOD detection and compute the corresponding AUC-PR score under label attacks and uncertainty attacks. The bold, middle part of the columns in Tables 6.5 to 6.7 show the AUC-PR scores on CIFAR10, which we call *empirical performance* of the smoothed models. To facilitate the comparison with the base model of Section 6.4, we highlight the AUC-PR scores in blue in cases where the smooth model is more robust. The highlighting clearly shows that randomized smoothing increases the robustness of the empirical performance on OOD detection. OOD detection under strong PGD attacks (attack radius ≥ 0.5) performs comparable to random guessing (i.e. AUC-PR scores around 50% whith 50% ID and 50% OOD data). This shows that DBU models are not reliably efficient w.r.t. this task. In attack detection and distinguishing between correctly and wrongly predicted labels the smoothed DBU model are mostly more robust than the base models for attack radii ≥ 0.5.

Certified performance. Using the median based on smoothing improves the empirical robustness, but it does not provide formal guarantees how low/high the performance might actually get under perturbed data (since any attack is only a heuristic). Here, we propose novel guarantees by exploiting the individual certificates we obtain via randomized smoothing. Note that the certification procedure [455] enables us to derive lower and upper bounds $\underline{m}(\boldsymbol{x}^{(i)}) \leq m(\boldsymbol{x}^{(i)}) \leq \overline{m}(\boldsymbol{x}^{(i)})$ which hold with high probability and indicate how much the median might change in the worst-case when $\boldsymbol{x}^{(i)}$ gets perturbed subject to a specific (attack) radius.

These bounds allow us to compute certificates that bound the performance of the smooth models, which we refer to as the *guaranteed lowest performance* and *guaranteed highest performance*. More precisely, for the guaranteed lowest performance of the model we take the pessimistic view that all ID data points realize their individual upper bounds $\overline{m}(\boldsymbol{x}^{(i)})$, i.e. have their highest possible uncertainty (worst case). On the other hand, we assume all OOD samples realize their lower bounds $\underline{m}(\boldsymbol{x}_s^{(i)})$. Using these values as the uncertainty scores for all data points we obtain the guaranteed lowest performance of the model. A guaranteed lowest performance of e.g. 35.0 means that even under the worst case conditions an attack is not able to decrease the performance below 35.0. Analogously, we can take the optimistic view to obtain the guaranteed highest performance of the smoothed models. Tables 6.5 to 6.7 show the guaranteed lowest/highest performance (non-bold, left/right of the empirical performance). Our results show that the difference

between guaranteed highest and guaranteed lowest performance increases with the attack radius, which might be explained by the underlying lower/upper bounds on the median being tighter for smaller perturbations.

Adversarial training. Randomized smoothing improves robustness of DBU models and allows us to compute performance guarantees. However, an open question is whether it is possible to increase robustness even further by combining it with adversarial training. To obtain adversarially trained models we augment the data set using perturbed samples that are computed by PGD attacks against the cross-entropy loss (label attacks) or the differential entropy (uncertainty attacks). These perturbed samples $\tilde{\boldsymbol{x}}^{(i)}$ are computed during each epoch of the training based on inputs $\boldsymbol{x}^{(i)}$ and added to the training data (with the label $y^{(i)}$ of the original input). Tables 6.5 to 6.7 illustrate the results. We choose the attack radius used during training and the σ used for smoothing to be equal. To facilitate comparison, we highlight the empirical performance of the adversarially trained models in blue if it is better than the performance of the base model. Our results show that the additional use of adversarial training has a minor effect on the robustness and does not result in a significant further increase of the robustness.

We conclude that median smoothing is a promising technique to increase robustness w.r.t. distinguishing between correctly labeled samples and wrongly labeled samples, attack detection and differentiation between in-distribution data and out-of-distribution data of all Dirichlet-based uncertainty models, while additional adversarial training has a minor positive effect on robustness.

6.5 Conclusion

This chapter analyzes robustness of uncertainty estimation by DBU models and answers multiple questions in this context. Our results show: (1) While uncertainty estimates are a good indicator to identify correctly classified samples on unperturbed data, performance decrease drastically on perturbed data-points. (2) None of the Dirichlet-based uncertainty models is able to detect PGD label attacks against the class prediction by uncertainty estimation, regardless of the used uncertainty measure. (3) Detecting OOD samples and distinguishing between ID-data and OOD-data is not robust. (4) Applying median smoothing to uncertainty estimates increases robustness of DBU models w.r.t. all analyzed tasks, while adversarial training based on label or uncertainty attacks resulted in minor improvements.

7 Retrospective

In this section, we provide a retrospective on the Chapters 3 to 6 since their publication by discussing potential improvements and the related works published a posteriori.

7.1 Uncertainty estimation for classification and regression

Potential improvements. The proposed methods PostNet (see Chapter 3) and NatPN (see Chapter 4) for uncertainty estimation for classification and regression are composed of several components (e.g. encoder/decoder, density estimator, prior, loss, optimizer) with potential improvements. While Chapter 5 studies the impact of training, architecture, and prior specification for deterministic uncertainty methods, we believe that further improvements can be achieved in these directions. First, more expressive density estimator like other recent normalizing flows [232] and diffusion models [218] could improve uncertainty estimation. Second, it would be interesting to explore if different choices of prior can improve performance as it have been shown to have a significant impact in other Bayesian neural networks [438, 5]. Finally, the design of Bayesian loss has shown to be an important choice for uncertainty estimation [32, 33].

Recent related works. Recently, the approaches presented in this thesis have been at the core of a survey on evidential deep learning [419] and implemented in Google uncertainty benchmark [305]. Similar to our approach, other works have also subsequently explored Bayesian neural networks which are not fully stochastic [377] and uncertainty estimation methods with density estimation [110, 345, 375]. Some other works explored efficient uncertainty estimation by proposing to train a single larger network [4], an ensemble of subnetworks [178], training energy-based models [120], or pruning neural networks [21]. Further, multiple methods proposed to use conformal predictions to provide uncertainty estimates for any trained model by using an additional calibration set [13, 340]. Finally, other recent works [288, 414] had a close look at the evaluation of uncertainty estimation for modern and large pretrained models. Beyond uncertainty models, [203] has also analyzed what are appropriate uncertainty measures for epistemic and aleatoric uncertainty in machine learning.

7.2 Practicality and Robustness of uncertainty estimation

Potential improvements. First, the study in Chapter 5 could be extended to account for different task settings involving small datasets, large datasets, or active learning scenarios. Beyond further analysis and improvements of the components of deterministic

uncertainty methods, we believe that it would be interesting to extend the practical study in Chapter 5 with theoretical results. In particular, it might be interesting to show that the trade-off between OOD generalization and OOD detection is not only due to empirically observed feature collapse but is also due to fundamental theoretical limitations. Second, the proposed methods and evaluations for the robustness of uncertainty estimation in Chapter 6 has two main directions of improvements. First, it would be interesting to extend the benchmark to other recent uncertainty methods and datasets. This would allow to give a more extensive view on the weaknesses of existing uncertainty methods. Second, no approaches have shown significant gain in uncertainty robustness. Indeed, adversarial training and smoothing approaches detailed in Chapter 6 have shown only small improvement.

Recent related works. Recently, [144] and [463] proposed attacks on uncertainty estimations which are very similar to our approach without proposing solutions for robust uncertainty estimation. Only [287] has proposed another method for certifiable uncertainty estimation. On a different direction, [105] proposed to use input uncertainty to design less perceptible adversarial attacks. Finally, [9] proposed to provide uncertainty estimates based on adversarial attacks. Despite the fast progress of uncertainty attacks, this field is relatively new and adversarial robustness for uncertainty estimation is still unsolved.

Part III

Uncertainty Estimation for Non-Independent Data

8 Uncertainty Estimation for Graph Data

Doubt is not a pleasant condition, but certainty is absurd.

Voltaire

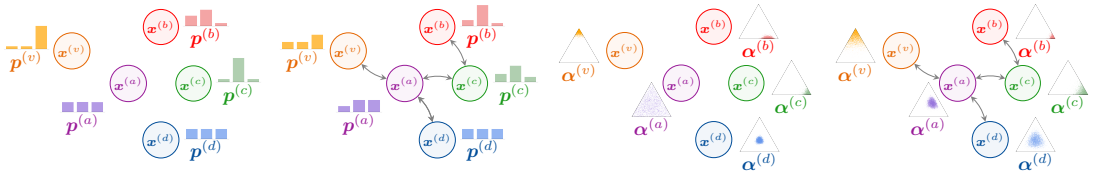
8.1 Introduction

After previously studying uncertainty estimation on independent data in Part II, we now focus on uncertainty estimation for non-independent data (e.g. graph data, sequential data). To this end, we start in this chapter by considering the task of node classification on graphs.

Traditionally, machine learning models assume i.i.d. inputs, thus performing predictions based on input features only. For uncertainty estimation on i.i.d. inputs, a large class of definitions, models and evaluation methods have been introduced [141, 275, 3, 335, 235]. Further, uncertainty estimation has been successfully applied to different tasks e.g. out-of-distribution (OOD) or shift detection [335], active learning [316, 251], continual learning [7] or reinforcement learning [84].

In contrast, uncertainty estimation on interdependent nodes is more complex than on i.i.d. inputs and under-explored [3]. A node in an attributed graph is characterized by two types of information: its features and its neighborhood. While the feature information indicates the node position in the feature space – similarly to i.i.d. inputs –, the neighborhood information indicates the additional node position in the network space. To leverage the neighborhood information, recent graph neural networks (GNNs) successfully proposed to enrich and correct the possibly noisy information of the features of a single node by aggregating them with the features of its neighborhood [222, 423, 229]. It naturally leads to the distinction between predictions *without network effects* based exclusively on their own node feature representation, and predictions *with network effects* based on neighborhood aggregation. The aggregation step commonly assumes *network homophily* which states that nodes with similar properties tend to connect to each other more densely, thus violating the i.i.d. assumption between node features given their neighborhood.

The core motivation of our work is to transfer some of the existing uncertainty estimation definitions, models and evaluations from i.i.d. inputs to interdependent node inputs by leveraging both the feature and the neighborhood information. In particular, we aim at an accurate quantification of the aleatoric and epistemic uncertainty without and with network effect under network homophily (see Fig. 8.1).



(a) AU w/o network **(b)** AU w/ network **(c)** EU w/o network **(d)** EU w/ network

Figure 8.1: Illustration of aleatoric uncertainty (AU) and epistemic uncertainty (EU) without and with network effects (i.e. i.i.d. inputs vs interdependent inputs). Nodes have the same features in all cases. Network effects are visualized through edges between nodes which change the predicted distributions. The aleatoric uncertainty is high if the categorical distribution $\hat{y}^{(v)} \sim \text{Cat}(\boldsymbol{p}^{(v)})$ is flat. The epistemic uncertainty is high if the Dirichlet distribution $\boldsymbol{p}^{(v)} \sim \text{Dir}(\boldsymbol{\alpha}^{(v)})$ is spread out. We refer the reader to Section 8.3.2 for formal definitions of those distributions.

Our contribution. In this chapter, we consider uncertainty estimation on semi-supervised node classification. First, we derive three desiderata which materialize reasonable uncertainty for non-independent inputs. These desiderata cover the traditional notions of aleatoric and epistemic uncertainty and distinguish between the uncertainty with and without network effects. Second, we propose Graph Posterior Network (GPN)[1] for uncertainty estimation for node classification and prove formally that it follows the desiderata requirements contrary to popular GNNs. Third, we build an extensive evaluation setup for uncertainty estimation which relies on the assessment of uncertainty estimation quality of OOD detection and robustness against shifts of the attributed graph properties. Both OOD data and attributed graph shifts distinguish between attribute and structure anomalies. The theoretical properties of GPN manifest in these experiments where it outperforms all other baselines on uncertainty evaluation.

8.2 Related Work

In this section, we cover the related work for predictive uncertainty estimation for i.i.d. inputs and for graphs. To this end, we review the commonly accepted *desiderata* defining the desired uncertainty estimation under different circumstances, the *methods* capable of consistent uncertainty quantification and the *evaluation* validating the quality of the uncertainty estimates in practice.

Uncertainty for i.i.d. inputs. The related work for uncertainty quantification on i.i.d. inputs is rich as for example shown in a recent survey [3]. *Desiderata:* Far from ID data, the predicted uncertainty is expected to be high [286, 68, 237, 142]. Close to ID data, the desired uncertainty is more complicated. Indeed, while some works expected models to be robust to small dataset shifts [335, 400], other works expected to detect near OOD classes based on uncertainty [443, 235, 64]. *Methods:* Many methods already exist for

[1] Project page including code at `https://www.daml.in.tum.de/graph-postnet`

uncertainty quantification for i.i.d. inputs like images or tabular data. A first family of models quantifies uncertainty by aggregating statistics (e.g. mean, variance or entropy) from sub-networks with different weights. Important examples are ensemble [246, 434, 439, 178], dropout [397] or Bayesian Neural Networks (BNN) [42, 103, 268, 128, 114]. Most of these approaches require multiple forward-passes for uncertainty quantification. Further, dropout and BNN may have other pitfalls regarding their limited applicability to more complex tasks [329, 191, 170, 135]. A second family quantifies uncertainty by using the logit information. Important examples are temperature scaling which rescale the logits after training [175, 255] and energy-based models which interpret the logits as energy scores [259, 169]. A third family of model quantifies uncertainty based on deep Gaussian Processes (GP). Important examples use GP at activation-level [300] or at (last) layer-level [247, 237, 420, 36]. Finally, a last family of models quantifies uncertainty by directly parameterizing a conjugate prior distribution over the target variable. Important examples explicitly parameterize prior distributions [374, 277, 270, 273, 10] or posterior distributions [67, 68]. Methods based on GP and conjugate prior usually have the advantage of deterministic and fast inference. *Evaluation:* Previous works have already proposed empirical evaluation of uncertainty estimation by looking at accuracy, calibration or OOD detection metrics under dataset shifts or adversarial perturbations for i.i.d. inputs [335, 235]. In contrast with all these approaches, this chapter studies uncertainty quantification for classification of *interdependent nodes*.

Uncertainty for graphs. Notably, the recent survey [3] points out that there is only a limited number of studies on uncertainty quantification on GNN and semi-supervised learning. Moreover, they recommend proposing new methods. *desiderata:* To the best of our knowledge, only [124] proposed explicit desiderata for node classification for non-attributed graphs. They expect disconnected nodes to recover prior predictions and nodes with higher beliefs to be more convincing. In this chapter, we clarify the desired uncertainty estimation for node classification on attributed graphs based on *motivated and explicit desiderata*. *Methods:* The largest family of models for uncertainty for graphs are dropout- or Bayesian-based methods. Important examples propose to drop or assign probabilities to edges [363, 73, 177, 94, 195]. Further works proposed to combine the uncertainty on the graph structure with uncertainty on the transformation weights similarly to BNN [121, 468, 337, 338]. Importantly, these models do not directly quantify uncertainty on the prediction. Similarly to the i.i.d. case, a second family of models focuses on deterministic uncertainty quantification. Important examples mostly use Graph Gaussian Processes, which do not easily scale to large graphs [315, 473, 261, 49]. Only [470] explicitly parameterized a Dirichlet conjugate prior. They combined it with multiple components (Graph-Based Kernel, dropout, Teacher Network, loss regularizations) which cannot easily distinguish between uncertainty without and with network effects. In contrast, GPN is a simple approach based on conjugate prior parametrization and disentangles uncertainty with and without network effects. *Evaluation:* The evaluation of most of those methods was not focused on the quality of the uncertainty estimates but on the target task metrics (e.g. accuracy for classification, distance to ground truth for regres-

sion). Some methods focus on robustness of the target task metrics against adversarial perturbations [174, 480, 478]. Other methods only relied on uncertainty quantification to build more robust models [475, 132]. For node classification, only few works evaluated uncertainty by using Left-Out classes or detection of missclassified samples [470], active learning [315] or visualization [49]. Note that proposed uncertainty evaluations on molecules at graph level [467, 367, 8, 188, 415] is an orthogonal problem. In this chapter, we propose a *sound and extensive evaluation* for uncertainty in node classification. It distinguishes between OOD nodes w.r.t. features and structure, and graph dataset shifts w.r.t. the percentage of perturbed node features and the percentage of perturbed edges.

8.3 Uncertainty Quantification for Node Classification

We consider the task of (semi-supervised) node classification on an attributed graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ with adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N \times N}$ and node attribute matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$. We aim at inferring the labels $y^{(v)} \in \{1, ..., C\}$ plus the the aleatoric uncertainty $u^{(v)}_{\text{alea}}$ and the epistemic uncertainty $u^{(v)}_{\text{epist}}$ of unlabeled nodes $v \in \mathcal{T}$ given a set of labelled nodes $u \in \mathcal{U}$ in the graph where $\mathcal{V} = \mathcal{T} \cup \mathcal{U}$ denotes the set of vertices.

8.3.1 Desiderata

Uncertainty estimation in the setting of interdependent inputs is not well-studied. It often leaves the expected behavior and interpretations for uncertainty estimation unclear. Thus, we need well-grounded desiderata to derive meaningful models. In this section, we aim at specifying the desired uncertainty predictions under various circumstances in homophilic attributed graphs. To this end, we propose three desiderata which are based on the two following distinctions. The first distinction differentiates between aleatoric and epistemic uncertainty which are commonly used concepts under the i.i.d. assumptions [141, 275]. The second distinction differentiates between uncertainty without and with network effects which are motivated by the concepts of attribute and structure anomalies used in the attributed graph setting [45]. These new desiderata cover all possible combinations encountered by these distinctions and extend the desiderata proposed by [124] for non-attributed graphs. We designed the desiderata to be informal and generic so that they are application independent, model-agnostic and do not require complex mathematical notations similarly to [124, 324]. In practice, formal definitions need to instantiate general concepts like aleatoric/epistemic uncertainty and with/without network effects noting that some definitions might be more convenient depending on the task. The first desideratum deals with (epistemic and aleatoric) uncertainty estimation without network effects (see Figs. 8.1a and 8.1c). :

Desideratum 8.3.1. *A node's prediction in the absence of network effects should only depend on its own features. A node with features more different from training features should be assigned higher uncertainty.*

Desideratum 8.3.1 states that if a node v has no neighbors, then the final prediction $\boldsymbol{p}^{(v)}$ should only depend on its own node features $\boldsymbol{x}^{(v)}$. Further, for anomalous features the

model should fall back to safe prior predictions, indicating high aleatoric and epistemic uncertainty. This aligns with [124] which expects to recover prior predictions for non-attributed nodes without network effect, and [286, 68] which expect to recover prior predictions far from training data for i.i.d. inputs. The second desideratum deals with epistemic uncertainty estimation with network effects (see Figs. 8.1c and 8.1d):

Desideratum 8.3.2. *All else being equal, if a node's prediction in the absence of network effects is more epistemically certain, then its neighbors' predictions in the presence of network effects should become more epistemically certain.*

Desideratum 8.3.2 states that a node v with confident feature predictions $\boldsymbol{x}^{(v)}$ is expected to be convincing and make its neighbors $u \in \mathcal{N}(v)$ more confident. Conversely, a node with anomalous features is expected to make its neighborhood less confident. This desideratum materializes the network homophily assumption at the epistemic level i.e. connected nodes have similar epistemic uncertainty estimates. For non-attributed graphs, [124] similarly expects a more confident node to have more influence on a direct neighbor. The third desideratum deals with aleatoric uncertainty estimation with network effects (see Figs. 8.1a and 8.1b):

Desideratum 8.3.3. *All else being equal, a node's prediction in the presence of network effects should have higher aleatoric uncertainty if its neighbors' predictions in the absence of network effects have high aleatoric uncertainty. Further, a node prediction in the presence network effects should have higher aleatoric uncertainty if its neighbors' predictions in the absence network effects are more conflicting.*

Desideratum 8.3.3 states that no clear classification decision should be made for a node v if no clear classification decisions can be made for its neighbors. Further, the classification decision becomes less certain if a neighbor has a conflicting classification decision. Note that this desideratum is more subtle than the direct application of network homophily at the aleatoric level. Indeed a node can have a high aleatoric uncertainty contrary to its neighbors which predict different classes with low aleatoric uncertainty. This aligns with the intuition that conflicting information from the neighborhood provides an irreducible uncertainty to the considered node.

8.3.2 Graph Posterior Network

The Bayesian update rule is a key component of GPN to model uncertainty on the predicted categorical distribution. For a single categorical distribution $y \sim \text{Cat}(\boldsymbol{p})$, the *standard* Bayesian update is straightforward. A natural choice for a prior distribution over the parameters $\boldsymbol{p}$ is its conjugate prior i.e. the Dirichlet distribution $\mathbb{P}(\boldsymbol{p}) = \text{Dir}(\boldsymbol{\alpha}^{\text{prior}})$ with $\alpha_c^{\text{prior}} \in \mathbb{R}_+^C$. Given the observations $y^{(1)}, ..., y^{(N)}$, the Bayesian update then consists in applying the Bayes' theorem

$$\mathbb{P}\left(\boldsymbol{p} \,|\, \{y^{(j)}\}_{j=1}^N\right) \propto \mathbb{P}\left(\{y^{(j)}\}_{j=1}^N \,|\, \boldsymbol{p}\right) \times \mathbb{P}(\boldsymbol{p}) \tag{8.1}$$

producing the posterior distribution $\mathbb{P}(\boldsymbol{p} \,|\, \{y^{(j)}\}_{j=1}^N) = \text{Dir}(\boldsymbol{\alpha}^{\text{post}})$ where $\boldsymbol{\alpha}^{\text{post}} = \boldsymbol{\alpha}^{\text{prior}} + \boldsymbol{\beta}$ are the parameters of the posterior and $\beta_c = \sum_j = \mathbb{1}_{y^{(j)}=c}$ are the class counts. This

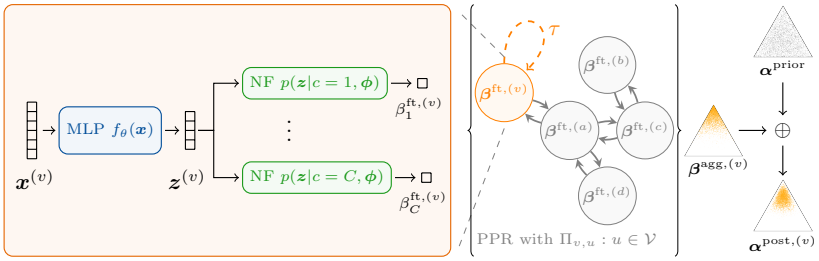


Figure 8.2: Overview of Graph Posterior Network: (1) node-level pseudo-counts computed by the feature encoder in the orange box, (2) PPR-based message passing visualized between the curly braces, and (3) input-dependent Bayesian update illustrated with the Dirichlet triangles on the right.

framework naturally disentangles the aleatoric and epistemic uncertainty by defining the Dirichlet mean $\bar{\boldsymbol{p}} = \frac{\boldsymbol{\alpha}}{\alpha_0}$ and the total evidence count $\alpha_0 = \sum_c \alpha_c$. Indeed, the aleatoric uncertainty is commonly measured by the entropy of the categorical distribution i.e. $u_{\text{alea}} = \mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}})$ [275, 67, 68] and the epistemic uncertainty can be measured by the total evidence count α_0 of observations i.e. $u_{\text{epist}} = -\alpha_0$ [67, 68]. Alternatively, the epistemic uncertainty can also be measured with the Dirichlet differential entropy [275]. Note that the reparameterization using $\bar{\boldsymbol{p}}$ and α_0 can apply to any class counts including the prior counts $\boldsymbol{\alpha}^{\text{prior}}$, the class counts $\boldsymbol{\beta}$ and the posterior counts $\boldsymbol{\alpha}^{\text{post}}$.

For classification, the predicted categorical distribution $\hat{y}^{(v)} \sim \text{Cat}(\boldsymbol{p}^{(v)})$ additionally depends on the specific input v. Hence, the *input-dependent* Bayesian rule [67, 68] extends the Bayesian treatment of a single categorical distribution to classification by predicting an individual posterior update for any possible input. Specifically, it first introduces a fixed Dirichlet prior over the categorical distribution $\boldsymbol{p}^{(v)} \sim \text{Dir}(\boldsymbol{\alpha}^{\text{prior}})$ where $\boldsymbol{\alpha}^{\text{prior}} \in \mathbb{R}_+^C$ is usually set to 1, and second predicts the input-dependent update $\boldsymbol{\beta}^{(v)}$ which forms the posterior distribution $\boldsymbol{p}^{(v)} \sim \text{Dir}(\boldsymbol{\alpha}^{\text{post},(v)})$ where the posterior parameters are equal to

$$\boldsymbol{\alpha}^{\text{post},(v)} = \boldsymbol{\alpha}^{\text{prior}} + \boldsymbol{\beta}^{(v)}. \tag{8.2}$$

The variable $\boldsymbol{\beta}^{(v)}$ can be interpreted as learned class pseudo-counts and its parametrization is crucial. For i.i.d. inputs, PostNet [67] models the pseudo-counts $\boldsymbol{\beta}^{(v)}$ in two main steps. **(1)** it maps the inputs features $\boldsymbol{x}^{(v)}$ onto a low-dimensional latent vector $\boldsymbol{z}^{(v)} = f_\theta(\boldsymbol{x}^{(v)}) \in \mathbb{R}^H$. **(2)**, it fits one conditional probability density $\mathbb{P}(\boldsymbol{z}^{(v)}|c;\boldsymbol{\phi})$ per class on this latent space with normalizing flows. The final pseudo count for class c is set proportional to its respective conditional density i.e. $\beta_c^{(v)} = N\mathbb{P}(\boldsymbol{z}^{(v)}|c;\boldsymbol{\phi})\mathbb{P}(c)$ where N is a total certainty budget and $\mathbb{P}(c) = \frac{1}{C}$ for balanced classes. Note that this implies $\alpha_0^{(v)} = N\mathbb{P}(\boldsymbol{z}^{(v)}|\boldsymbol{\phi})$. This architecture has the advantage of decreasing the evidence outside the known distribution when increasing the evidence inside the known distribution, thus leading to consistent uncertainty estimation far from training data.

Bayesian Update for Interdependent Inputs. We propose a simple yet efficient modification for parameterizing $\beta_c^{(v)}$ to extend the input-dependent Bayesian update for interdependent attributed nodes. The core idea is to first predict the feature class pseudo-counts

$\boldsymbol{\beta}^{\text{ft},(v)}$ based on independent node features only, and then diffuse them to form the aggregated class pseudo-counts $\boldsymbol{\beta}^{\text{agg},(v)}$ based on neighborhood features. Hence, the feature class pseudo-counts $\boldsymbol{\beta}^{\text{ft},(v)}$ intuitively act as uncertainty estimates without network effects while the aggregated class pseudo-counts $\boldsymbol{\beta}^{\text{agg},(v)}$ intuitively act as uncertainty estimates with network effects.

To this end, GPN performs three main steps (see Fig. 8.2). **(1)** A (feature) encoder maps the features of v onto a low-dimensional latent representation $\boldsymbol{z}$ i.e. $\boldsymbol{z}^{(v)} = f_\theta(\boldsymbol{x}^{(v)}) \in \mathbb{R}^H$. In practice, we use a simple MLP encoder in our experiments similarly to APPNP [229]. **(2)** One conditional probability density per class $\mathbb{P}(\boldsymbol{z}^{(v)} \,|\, c; \boldsymbol{\phi})$ is used to compute $\beta_c^{\text{ft},(v)}$ i.e $\beta_c^{\text{ft},(v)} \propto \mathbb{P}(\boldsymbol{z}^{(v)} \,|\, c; \boldsymbol{\phi})$. Note that the the total feature evidence $\alpha_0^{\text{ft},(v)} = \sum_c \beta_c^{\text{ft},(v)}$ and the parameter $\bar{\boldsymbol{p}}^{\text{ft},(v)} = \boldsymbol{\beta}^{\text{ft},(v)} / \alpha_0^{\text{ft},(v)}$ are only based on node features and can be seen as epistemic and aleatoric uncertainty measures *without network effects*. In practice, we used radial normalizing flows for density estimation similarly to [67] and scaled the certainty N budget w.r.t. the latent dimension H similarly to [68]. **(3)** A Personalized Page Rank (PPR) message passing scheme is used to diffuse the feature class pseudo-counts $\beta_c^{\text{ft},(v)}$ and form the aggregated class pseudo-counts $\beta_c^{\text{agg},(v)}$ i.e.

$$\beta_c^{\text{agg},(v)} = \sum_{u \in \mathcal{V}} \Pi_{v,u}^{ppr} \beta_c^{\text{ft},(u)} \tag{8.3}$$

where $\Pi_{v,u}^{ppr}$ are the dense PPR scores implicitly reflecting the importance of node u on v. We approximate the dense PPR scores using power iteration similarly to [229]. The aggregated pseudo-count $\beta_c^{\text{agg},(v)}$ is then used in the input-dependent Bayesian update (see Eq. (8.2)). Remark that the scores $\Pi_{v,u}^{ppr}$ define a valid conditional distribution over all nodes associated to the PPR random walk (i.e. $\sum_u \Pi_{v,u}^{ppr} = 1$). It can be viewed as a soft neighborhood for v accounting for all neighborhood hops through infinitely many message passing steps [229]. Hence, on one hand, the PPR scores define a probability distribution over nodes using the node edges only. On the other hand, the quantity $\mathbb{P}(\boldsymbol{z}^{(u)} \,|\, c; \boldsymbol{\phi})$ defines a probability distribution over nodes using the node features only. Therefore, we can equivalently rewrite this step using probabilistic notations $\mathbb{P}(v \,|\, u) = \Pi_{v,u}^{ppr}$ and $\mathbb{P}(u \,|\, c) = \mathbb{P}(\boldsymbol{z}^{(u)} \,|\, c; \boldsymbol{\phi})$:

$$\beta_c^{\text{agg},(v)} \propto \bar{\mathbb{P}}(v \,|\, c) = \sum_{u \in \mathcal{V}} \mathbb{P}(v \,|\, u) \mathbb{P}(u \,|\, c) \tag{8.4}$$

Interestingly, the quantity $\bar{\mathbb{P}}(v \,|\, c)$ defines a valid distribution which normalizes over all node features and accounts for the soft neighborhood (i.e. $\int ... \int \bar{\mathbb{P}}(v \,|\, c) d\boldsymbol{z}^{(u_1)} ... d\boldsymbol{z}^{(u_{|\mathcal{V}|})} = 1$). Hence, the message passing step is a simple but efficient method to transform the feature distributions of a single node into a joint distributions over the soft neighborhood features. Finally, the evidence $\alpha_0^{\text{agg},(v)} = \sum_c \beta_c^{\text{agg},(v)}$ and the parameter $\boldsymbol{p}^{\text{agg},(v)} = \boldsymbol{\beta}^{\text{agg},(v)} / \alpha_0^{\text{agg},(v)}$ are based on neighborhood features and can be seen as epistemic and aleatoric uncertainty measures *with network effects*. Remark that, the sequential processing of the features (i.e. steps (1)+(2)) and network information (i.e. step (3)) in GPN is a key element to differentiate between the uncertainty without and with network effects and is a building block to provably obey the desiderata.

GPN extends both APPNP [229] and PostNet [67] approaches. The key difference to APPNP is the density estimation modeling the epistemic uncertainty (i.e. steps (1)+(2)) and the input-dependent Bayesian update allowing to recover the prior prediction (i.e. Eq. (8.2)). The key difference to PostNet is the PPR diffusion which accounts for dependence between nodes (step (3)).

Optimization. We follow [67] and train GPN by minimizing the following Bayesian loss with two terms i.e.:

$$\mathcal{L}^{(v)} = -\,\mathbb{E}_{\boldsymbol{p}^{(v)} \sim \mathbb{Q}^{post,(v)}} \Big[\log \mathbb{P}(y^{(v)} \,|\, \boldsymbol{p}^{(v)}) \Big] - \lambda\, \mathbb{H}\, \mathbb{Q}^{post,(v)} \tag{8.5}$$

where λ is a regularization factor. It can be computed quickly in closed-form and provides theoretical guarantees for optimal solutions [67]. All parameters of GPN are trained jointly. Similarly to [68], we also observed that "warm-up" training for the normalizing flows is helpful.

8.3.3 Uncertainty Estimation Guarantees

In this section, we provide theoretical guarantees showing that GPN fulfills the three desiderata under mild assumptions given the specific definitions of concepts of aleatoric/epistemic uncertainty and with/without network effects presented in Section 8.3.2. Throughout this section, we consider a GPN model parameterized with a (feature) encoder f_ϕ with piecewise ReLU activations, a PPR diffusion, and a density estimator $\mathbb{P}(\boldsymbol{z}^{\text{ft},(v)} \,|\, \omega)$ with bounded derivatives. We present detailed proofs in appendix.

The first theorem shows that GPN follows des. 8.3.1 and guarantees that GPN achieves reasonable uncertainty estimation on extreme node features without network effects:

Theorem 2. *Lets consider a GPN model. Let $f_\phi(\boldsymbol{x}^{(v)}) = V^{(l)}\boldsymbol{x}^{(v)} + a^{(l)}$ be the piecewise affine representation of the ReLU network f_ϕ on the finite number of affine regions $Q^{(l)}$ [17]. Suppose that $V^{(l)}$ have independent rows, then for any node v and almost any $\boldsymbol{x}^{(v)}$ we have $\mathbb{P}(f_\phi(\delta \cdot \boldsymbol{x}^{(v)}) \,|\, c; \boldsymbol{\phi}) \underset{\delta \to \infty}{\to} 0$. Without network effects, it implies that $\beta_c^{ft,(v)} = \beta_c^{agg,(v)} \underset{\delta \to \infty}{\to} 0$.*

The proof relies on two main points: the equivalence of the GPN and PostNet architectures without network effects, and the uncertainty guarantees of PostNet far from training data similarly to [68]. It intuitively states that, without network effects, GPN predict small evidence (i.e. $\boldsymbol{\beta}^{\text{agg},(v)} \approx \mathbf{0}$) far from training features (i.e. $||\delta \cdot \boldsymbol{x}^{(v)}|| \to \infty$) and thus recover the prior prediction (i.e. $\boldsymbol{\alpha}^{\text{post},(v)} \approx \boldsymbol{\alpha}^{\text{prior}}$). Note that contrary to GPN, methods which do not account for node features (e.g. Label Propagation) or methods which only use ReLU activations [182] cannot validate des. 8.3.1. Further, methods which perform aggregation steps in early layers (e.g. GCN [222]) do not separate the processing of the feature and network information making unclear if they fulfill the des. 8.3.1 requirements.

The second theorem shows that GPN follows des. 8.3.2 and guarantees that a node v becomes more epistemically certain if its neighbors are more epistemically certain:

Theorem 3. *Lets consider a GPN model. Then, given a node v, the aggregated feature evidence $\alpha_0^{agg,(v)}$ is increasing if the feature evidence $\alpha_0^{ft,(u)}$ of one of its neighbors $u \in \mathcal{N}(v)$ is increasing.*

The proof directly relies on Eq. (8.3). Intuitively, this theorem states that the epistemic uncertainty $u_{\text{epist}}^{(v)} = -\alpha_0^{\text{agg, (v)}}$ of a node v with network effects decreases if the epistemic uncertainty of the neighboring nodes without network effects decreases. Note that contrary to GPN, methods which do not model the epistemic uncertainty explicitly (e.g. GCN [222], GAT [423] or APPNP [229]) are not guaranteed to fulfil des. 8.3.2.

The third theorem shows that GPN follows des. 8.3.3. It guarantees that a node v becomes more aleatorically uncertain if its neighbors are more aleatorically uncertain, or if a neighbor prediction disagrees more with the current node prediction:

Theorem 4. *Lets consider a GPN model. Lets denote $\bar{\boldsymbol{p}}^{agg,\ (v)} = \boldsymbol{\beta}^{agg,(v)} / \alpha_0^{agg,(v)}$ the diffused categorical prediction for node v where c^* is its winning class. Further, lets denote $\bar{\boldsymbol{p}}^{ft,\ (u)} = \boldsymbol{\beta}^{ft,(v)} / \alpha_0^{ft,(v)}$ the non-diffused categorical prediction for a node $u \in \mathcal{V}$. First, there exists normalized weights $\Pi'_{v,u}$ such that $\sum_{u \in \mathcal{V}} \Pi'_{v,u} \mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{ft,\ (u)}) \leq \mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{agg,\ (v)})$. Second, if for any node $u \in \mathcal{V}$ the probability of $\bar{\boldsymbol{p}}_{c^*}^{ft,\ (u)}$ decreases, then $\mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{agg,\ (v)})$ increases.*

The proof of the first part of the theorem is based on the entropy convexity. Intuitively, it states that the aleatoric uncertainty $u_{\text{alea}}^{(v)} = \mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{\text{agg, (v)}})$ of a node v with network effects is lower bounded by a weighted average of the aleatoric uncertainty without network effects of its soft neighborhood. The second part of the theorem intuitively states that if the prediction of a neighboring node u without neighbor effects disagrees more with the current class prediction c^* of the node v, then the aleatoric uncertainty $u_{\text{alea}}^{(v)} = \mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{\text{agg, (v)}})$ with network effects becomes higher. Note that contrary to GPN, methods which do not use edges (e.g. PostNet [67]) cannot validate des. 8.3.3 and des. 8.3.2.

8.3.4 Limitations & Impact

OOD data close to ID data. While GPN is guaranteed to provide consistent uncertainty estimates for nodes with extreme OOD features, it does not guarantee any specific uncertainty estimation behavior for OOD data close to ID data. Note that there exist two possible desired behaviors for OOD close to ID data: being robust to small dataset shifts [335, 400] or detect near OOD data [443, 235, 64]. The duality of these two views makes unclear what would be the desired behavior even for i.i.d. data.

Non-homophilic uncertainty. Our approach assumes that connected nodes are likely to have similar uncertainty estimates as defined in des. 8.3.2 and des. 8.3.3. Contrary to [476], we do not tackle the problem of heterophilic graphs where two neighboring nodes might reasonably have different uncertainty estimates.

Task-specific OOD. Density estimation is shown to be inappropriate for OOD detection when acting directly on raw images [308, 77, 307] or on arbitrarily transformed space [248]. One of the reasons is that normalizing flows learn pixel correlations in images. This phenomena does not happen for tabular data with more semantic features [224]. First note that, similarly to tabular data, semantic node features are less likely to suffer from the same flaws. Second, following previous works [67, 68, 224, 302, 443], GPN mitigates this issue by using density estimation on a latent space which is low-dimensional and task-specific. Nonetheless, we emphasize that GPN provides predictive uncertainty estimates which depends on the considered task i.e. OOD data w.r.t. features which are not useful for the specific task are likely not to be encoded in the latent space, and thus not to be detected.

8.4 Experiments

In this section, we provide an extensive evaluation set-up for uncertainty quantification for node classification. It compares **GPN** to 13 baselines on 8 datasets and consists in two task types. First, we evaluate the detection of OOD nodes with features perturbations and Left-Out classes. Second, we evaluate the robustness of accuracy, calibration and uncertainty metrics w.r.t. feature and edge shifts.

8.4.1 Setup

Ablation. In the experiments, GPN uses a MLP as feature encoder, radial normalizing flows [358] for the density estimation and a certainty budget N which scales with respect to the latent dimension [68]. We provide an ablation study covering aleatoric uncertainty through APPNP, feature-level estimates through PostNet, diffusing resulting pseudo-counts after training, and GPN with diffusion of $\log(\beta_c^{\text{ft},(v)})$ instead of $\beta_c^{\text{ft},(v)}$ (see Appendix E.5.1). The complete GPN model outperforms the ablated models for uncertainty estimation. Further, we provide a hyper-parameter study covering for example different number of flow layers, latent dimensions, PPR teleport probabilities (see Appendix E.5.2)).

Baselines. We used 13 baselines covering a wide variety of models for semi-supervised node classification and uncertainty estimation. We show the results of 5 baselines in the main paper and the full results in appendix. It contains two standard GNNs (i.e. Vanilla GCN **VGCN** [222, 379] and **APPNP** [229]), one robust GNN (i.e. **RGCN** [475]), one dropout-based method for GNN (i.e. **DropEdge** [363]), two Graph Gaussian Processes methods (i.e. **GGP** [315] and **Matern-GGP** [49]), the Graph-based Kernel Dirichlet GCN method (i.e. **GKDE-GCN** [470]) and two parameter-less methods (i.e. **GKDE** [470] and Label Propagation **LP** see Appendix E.4). Further, we also compared to direct adaptation of dropout (i.e. **VGCN-Dropout**[141]), ensemble (i.e. **VGCN-Ensemble** [246]), BNN (i.e. **VGCN-BNN** [42]) and energy-based models (i.e. **VGCN-Energy** [259]) to vanilla GCNs. All models are trained using the same number of layers

and similar number of hidden dimensions. We used early stopping and report the used hyperparameters in appendix. The results are averaged over 10 initialization seeds per split. Further model details are given in appendix.

Datasets. We used 8 datasets with different properties summarized in appendix. We show the results of 3 datasets in the main part of this chapter and the full results in appendix. It contains common citation network datasets (i.e. **CoraML** [282, 160, 156, 373], **CiteSeer** [160, 156, 373], **PubMed** [309], **CoauthorPhysics** [379] **CoauthorCS** [379]) and co-purchase datasets (i.e. **AmazonPhotos** [281, 379], **AmazonComputers** [281, 379]). The results are averaged over 10 initialization splits with a train/val/test split of 5%/15%/80% using stratified sampling. Further, we evaluate on the large **OGBN Arxiv** dataset with 169,343 nodes and 2,315,598 edges [196, 427]. Further dataset details are given in the appendix.

8.4.2 Results

OOD Detection. In this section, we evaluate uncertainty estimation for OOD detection. To this end, we use the Area Under Receiving Operator Characteristics Curve (AUC-ROC) with aleatoric scores $u_{\text{alea}}^{(v)}$ **(Alea)** and epistemic scores $u_{\text{epist}}^{(v)}$ **(Epist)** similarly to [67, 470, 270, 273, 277, 259]. For GPN, we differentiate between epistemic uncertainty scores without network effects **(w/o Net.)** and with network effects **(w/ Net.)**. Further, we report results with the Area Under the Precision-Recall Curve (AUC-PR) in appendix. The definition of OOD for nodes in the presence of feature and network information is more complex than for i.i.d. input features. Hence, we propose two types of OOD nodes: nodes with OOD feature perturbations and nodes from Left-Out classes. For feature perturbations, we compute the accuracy on the perturbed nodes **(OOD-Acc)** to evaluate if the model can correct anomalous features. For Left-Out classes, we compute the accuracy on the observed classes **(ID-Acc)**. We report the short results in Table 8.1. We set a threshold of 64 GiB and 12 hours per training run. We also exclude methods which do not use attributes for detection of OOD feature perturbations.

Feature perturbations: These perturbations aim at isolating the contribution of the node feature information on the model predictions. To this end, we randomly select a subset of the nodes. For each single node v, we perturb individually its features using a Bernoulli or a Normal distribution (i.e. $\boldsymbol{x}^{(v)} \sim \text{Ber}(0.5)$ and $\boldsymbol{x}^{(v)} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$) keeping all other node features fixed. We then compare the uncertainty prediction on the perturbed and unperturbed node. On one hand, Bernoulli noise corresponds to small perturbations in the domain of discrete bag-of-words features. On the other hand, Normal noise corresponds to extreme perturbations out of the domain of discrete bag-of-words features. In practice, we expect out-of-domain perturbations to be easily detected [67]. First, we remark that uncertainty estimates of GPN based on features achieves an absolute improvement of at least +15% and +29% for Bernoulli and Normal perturbations over all baselines using network effects. This shows that GPN disentangles well the uncertainty without and with network effects. Second, we remark that all uncertainty estimates with network effects achieve poor results. This is expected if models can recover the correct

prediction after aggregation steps. Specifically, we observe that GPN achieves an accuracy improvement between +16% and +64% for Normal perturbations on perturbed nodes compared to baselines. It stresses that GPN performs a consistent evidence aggregation from neighborhood to recover from anomalous features. Further, note that GPN is still capable to detect those perturbed nodes almost perfectly using feature uncertainty. These remarks aligns with des. 8.3.1.

Left-Out classes: Detection of Left-Out classes involves both feature and neighborhood information. In this case, we remove the Left-Out classes from the training set but keep them in the graph similarly to [470]. We observe that the uncertainty estimates with network effects of GPN achieves an absolute improvement between +12% and +16% compared to its uncertainty estimates without network effects. It highlights the benefit of incorporating network information for uncertainty predictions when OOD samples (i.e. samples from the Left-Out classes) are likely to be connected to each other. This remark aligns with des. 8.3.2. Further, GPN outperforms other baselines by +2% to +22% for LOC detection while maintaining a competitive accuracy on other classes.

Misclassified samples: In addition to the OOD scores, we also report the results for the detection of misclassified samples with aleatoric and epistemic uncertainty on several datasets and models in Appendix E.5.3 for the sake of completeness. GPN performs competitively with the baselines. Moreover, we observe that epistemic uncertainty is better for OOD detection and aleatoric uncertainty is better for misclassification detection as already observed e.g. in [470].

	Model	ID-ACC	OOD-AUC-ROC	OOD-ACC	OOD-AUC-ROC	OOD-ACC	OOD-AUC-ROC
		Leave-Out Classes		$\boldsymbol{x}^{(v)} \sim \text{Ber}(0.5)$		$\boldsymbol{x}^{(v)} \sim \mathcal{N}(0,1)$	
CoraML	Matern-GGP	87.03	83.13 / 82.98 / *n.a.*	*n.a.*	*n.a.*	*n.a.*	*n.a.*
	VGCN-Dropout	89.08	81.27 / 71.65 / *n.a.*	77.76	62.06 / 50.38 / *n.a.*	18.28	40.53 / 71.06 / *n.a.*
	VGCN-Energy	89.66	81.70 / 83.15 / *n.a.*	78.90	63.68 / 66.26 / *n.a.*	18.37	9.34 / 0.32 / *n.a.*
	VGCN-Ensemble	**89.87**	81.85 / 74.24 / *n.a.*	78.00	63.58 / 56.81 / *n.a.*	21.00	33.72 / 64.92 / *n.a.*
	GKDE-GCN	89.33	82.23 / 82.09 / *n.a.*	76.40	61.74 / 63.15 / *n.a.*	16.86	40.03 / 1.42 / *n.a.*
	GPN	88.51	83.25 / **86.28** / 80.95	**80.98**	57.99 / 55.23 / **89.47**	**81.53**	55.96 / 56.51 / **100.00**
Amazon Photos	Matern-GGP	88.65	87.26 / 86.75 / *n.a.*	*n.a.*	*n.a.*	*n.a.*	*n.a.*
	VGCN-Dropout	94.04	80.90 / 70.11 / *n.a.*	83.86	56.85 / 55.04 / *n.a.*	22.29	49.11 / 66.74 / *n.a.*
	VGCN-Energy	94.24	82.44 / 79.64 / *n.a.*	83.91	57.91 / 59.07 / *n.a.*	21.40	31.07 / 6.42 / *n.a.*
	VGCN-Ensemble	**94.28**	82.72 / 88.53 / *n.a.*	84.40	57.86 / 56.01 / *n.a.*	20.30	44.14 / 69.01 / *n.a.*
	GKDE-GCN	89.84	73.65 / 69.09 / *n.a.*	73.17	57.01 / 58.00 / *n.a.*	24.04	24.45 / 9.82 / *n.a.*
	GPN	94.01	82.72 / **91.98** / 76.57	**87.47**	56.25 / 60.52 / **75.24**	**88.29**	51.89 / 61.89 / **100.00**
OGBN Arxiv	Matern-GGP	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*
	VGCN-Dropout	75.47	65.35 / 64.24 / *n.a.*	65.30	48.11 / 50.64 / *n.a.*	49.90	60.10 / 62.87 / *n.a.*
	VGCN-Energy	75.61	64.91 / 64.50 / *n.a.*	65.70	46.16 / 48.54 / *n.a.*	51.30	53.83 / 48.53 / *n.a.*
	VGCN-Ensemble	**76.12**	65.93 / 70.77 / *n.a.*	**67.00**	45.99 / 47.41 / *n.a.*	49.00	59.94 / 66.44 / *n.a.*
	GKDE-GCN	73.89	68.84 / 72.44 / *n.a.*	65.20	50.98 / 51.31 / *n.a.*	45.40	53.94 / 55.28 / *n.a.*
	GPN	73.84	66.33 / **74.82** / 62.17	65.50	51.49 / 55.82 / **93.05**	**65.50**	51.43 / 55.85 / **95.54**

Table 8.1: LOC and Feature Perturbations: Accuracy is reported on ID nodes for LOC experiments and on OOD nodes for feature perturbation experiments. OOD-AUC-ROC scores are given as *[Alea w/ Net] / [Epist w/ Net] / [Epist w/o Net]*. *n.a.* means either model or metric not applicable and *n.f.* means not finished within our constraints.

Attributed Graph Shifts. In this section, we focus on evaluating the robustness of the accuracy, calibration and the evolution of the uncertainty estimation under node feature shifts and edges shifts. This aligns with [335] which aims at evaluating the reliability

of uncertainty estimates under dataset shifts for i.i.d. inputs. Specifically, we evaluates the evolution of the accuracy, the ECE [306] calibration score, the epistemic and the aleatoric uncertainty measures.

Feature shifts: We perturbed the features of a fraction of the nodes using unit Gaussian perturbations. We report the short results in Fig. 8.3 and the full results in appendix. On one hand, we observe that GPN is significantly more robust to feature perturbations than all baselines. Indeed, the accuracy of GPN decreases by less than 5% even when 80% of the nodes are perturbed while the accuracy of other baselines decreases by more than 50% when only 20% of the nodes are perturbed. Similarly, we observed that GPN remains calibrated even when a high fraction of nodes are perturbed contrary to baselines. Hence, GPN intuitively discards uncertain features from perturbed nodes and only accounts for certain features from other nodes for more accurate predictions. On the other hand, we observe that, as desired, the average epistemic uncertainty of GPN consistently decreases when more nodes are perturbed. This remark aligns with des. 8.3.2. In contrast, baselines dangerously become more certain while achieving a poorer accuracy similarly to ReLU networks [182]. Hence GPN predictions are significantly more reliable than baselines under feature shifts.

Edge shifts: For edge shifts, we perturbed a fraction of edges at random. We report the results in appendix. As desired, we observe that the aleatoric uncertainty increases for all models including GPN. This aligns with des. 8.3.3 and the expectations that conflicting neighborhood should lead to more aleatorically uncertain predictions. Furthermore, the average epistemic uncertainty of GPN remains constant which is reasonable since the average evidence of a node's neighborhood remains constant.

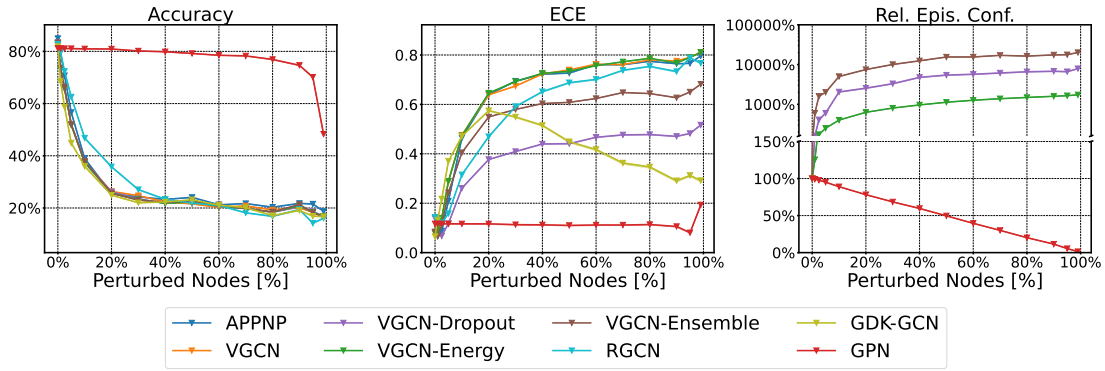


Figure 8.3: Accuracy, ECE, and average epistemic confidence under feature shifts for CoraML. We perturb features of different percentage of nodes using a Unit Gaussian noise.

Qualitative Evaluation. We show the abstracts of the CoraML papers achieving the highest and the lowest epistemic uncertainty without network effects in Table 8.2 and in the appendix. Interestingly, we observed that most uncertain papers corresponds to short and unconventional abstracts which can be seen as anomalous features. Furthermore, we also ranked the nodes w.r.t. to their epistemic uncertainty with network effects. In this

case, we observed that 78/100 nodes with the highest uncertainty do not belong to the largest connected component of the CoraML dataset. We propose additional uncertainty visualizations for GPN in Appendix E.5.6.

Inference & training time. We provide a comparison of inference and training times for most of the datasets and models under consideration in Appendix E.5.7. GPN needs a single pass for uncertainty estimation but requires the additional evaluation of one normalizing flow per class compared to APPNP. Hence, GPN brings a small computational overhead for uncertainty estimation at inference time. Furthermore, GPN is usually converging relatively fast during training and does not require pre-computing kernel values. In contrast, GKDE-GCN [470] requires the computation of the underlying Graph Kernel with a complexity of $\mathcal{O}\left(N^2\right)$ where N is the number of nodes in the graph. Finally, GPN is significantly more efficient than dropout or ensemble approaches as it does not require training or evaluating multiple models.

IlliGAL Report No. 95006 July 1995	Report of the 1996 Workshop on Reinforcement
Reihe FABEL-Report Status: extern Dokumentbezeichner: Org/Reports/nr-35 Erstellt am: 21.06.94 Korrigiert am: 28.05.95 ISSN 0942-413X	We tend to think of what we really know as what we can talk about, and disparage knowledge that we can't verbalize. [Dowling 1989, p. 252]
Keith Mathias and Darrell Whitley Technical Report CS-94-101 January 7, 1994	Multigrid Q-Learning Charles W. Anderson and Stewart G. Crawford-Hines Technical Report CS-94-121 October 11, 1994
Internal Report 97-01	A Learning Result for Abstract

Table 8.2: A selection of abstracts from CoraML which are assigned low feature evidences by GPN.

8.5 Conclusion

We introduce a well-grounded framework for uncertainty estimation on interdependent nodes. First, we propose explicit and motivated desiderata describing desired properties for aleatoric and epistemic uncertainty in the absence or in the presence of network effects. Second, we propose GPN, a GNN for uncertainty estimation which provably follows our desiderata. GPN performs a Bayesian update over the class predictions based on density estimation and diffusion. Third, we conduct extensive experiments to evaluate the uncertainty performances of a broad range of baselines for OOD detection and robustness against node feature or edge shifts. GPN outperforms all baselines in these experiments.

9 Uncertainty Estimation for Sequential Data

Uncertainty that comes from knowledge (knowing what you don't know) is different from uncertainty coming from ignorance.

Isaac Asimov

9.1 Introduction

In Chapter 8, we have introduced a Bayesian method to estimate uncertainty estimation on graph data. In this chapter, we now focus on uncertainty estimation on time events.

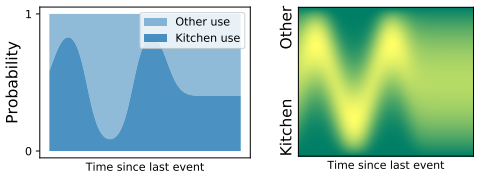
Discrete events, occurring irregularly over time, are a common data type generated naturally in our everyday interactions with the environment (see Fig. 9.2a for an illustration). Examples include messages in social networks, medical histories of patients in healthcare, and integrated information from multiple sensors in complex systems like cars. The problem we are solving in this chapter is: given a (past) sequence of asynchronous events, what will happen next? Answering this question enables us to predict, e.g., what action an internet user will likely perform or which part of a car might fail.

While many recurrent models for asynchronous sequences have been proposed in the past [312, 109], they are ill-suited for this task since they output a *single prediction* (e.g. the most likely next event) only. In an asynchronous setting, however, such a single prediction is not enough since the most likely event can change with the passage of time – even if no other events happen. Consider a car approaching another vehicle in front of it. Assuming nothing happens in the meantime, we can expect different events at *different times in the future*. When forecasting a short time, one expects the driver to start overtaking; after a longer time one would expect braking; in the long term, one would expect a collision. Thus, the expected behavior changes depending on the time we forecast, assuming no events occured in the meantime. Fig. 9.2a illustrates this schematically: having observed a square and a pentagon, it is likely to observe a square after a short time, while a circle after a longer time. Clearly, if some event occurs, e.g. braking/square, the event at the (then) observed time will be taken into account, updating the temporal prediction.

An ad-hoc solution to this problem would be to discretize time. However, if the events are near each other, a high sampling frequency is required, giving us very high computational cost. Besides, since there can be intervals without events, an artificial 'no event' class is required.

In this chapter, we solve these problems by directly predicting the entire evolution of the events over (continuous) time. Given a past asynchronous sequence as input, we can predict and evaluate for *any* future timepoint what the next event will likely be (under the assumption that no other event happens in between which would lead to an update of our model). Crucially, the likelihood of the events might change and one event can be more likely than others multiple times in the future. This periodicity exists in many event sequences. For instance, given that a person is currently at home, a smart home would predict a high probability that the kitchen will be used at lunch and/or dinner time (see Fig. 9.1a for an illustration). We require that our model captures such multimodality.

While Fig. 9.1a illustrates the evolution of the categorical distribution (corresponding to the probability of a specific event class to happen), an issue still arises outside of the observed data distribution. E.g. in some time intervals we can be *certain* that two classes are *equiprobable*, having observed many similar examples. However, if the model has not seen any examples at specific time intervals during training, we do not want to give a confident prediction. Thus, we incorporate *uncertainty* in a prediction directly in our model. In places where we expect events, the confidence will be higher, and outside of these areas the uncertainty in a prediction will grow as illustrated in Fig. 9.1b. Technically, instead of modeling the evolution of a categorical distribution, we model the *evolution of a distribution on the probability simplex*. Overall, our model enables us to operate with the *asynchronous discrete* event data from the past as input to perform *continuous-time* predictions to the future incorporating the predictions' uncertainty. This is in contrast to existing works as [109, 285].



(a) **(b)**

Figure 9.1: (a) An event can be expected multiple times in the future. (b) At some times we should be uncertain in the prediction. Yellow denotes higher probability density.

9.2 Model Description

We consider a sequence $[e_1, \ldots, e_n]$ of events $e_i = (c_i, t_i)$, where $c_i \in \{1, \ldots, C\}$ denotes the class of the ith event and $t_i \in \mathbb{R}$ is its time of occurrence. We assume the events arrive over time, i.e. $t_i > t_{i-1}$, and we introduce $\tau_i^* = t_i - t_{i-1}$ as the observed time gap between the ith and the $(i-1)$th event. The history preceding the ith event is denoted by $\mathcal{H}_i$. Let $S = \{\boldsymbol{p} \in [0,1]^C, \sum_c p_c = 1\}$ denote the set of probability vectors that form the $(C-1)$-dimensional simplex, and $P(\theta)$ be a family of probability distributions on this simplex parametrized by parameters θ. Every sample $\boldsymbol{p} \sim P(\theta)$ corresponds to a (categorical) class distribution.

Given e_{i-1} and $\mathcal{H}_{i-1}$, our goal is to model the evolution of the class probabilities, and their uncertainty, of the next event i over time. Technically, we model parameters $\theta(\tau)$, leading to a distribution P over the class probabilities $\boldsymbol{p}$ for all $\tau \geq 0$. Thus, we can

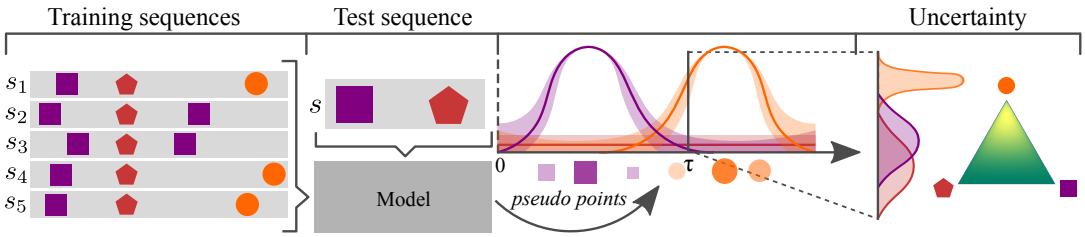


Figure 9.2: The model framework. (a) During training we use sequences s_i. (b) Given a new sequence of events s the model generates pseudo points that describe $\boldsymbol{\theta}(\tau)$, i.e. the temporal evolution of the distribution on the simplex. These pseudo points are based on the data that was observed in the training examples and weighted accordingly. We also have a measure of certainty in our prediction.

estimate the most likely class after a time gap τ by calculating $\text{argmax}_c \bar{\boldsymbol{p}}(\tau)_c$, where $\bar{\boldsymbol{p}}(\tau) := \mathbb{E}_{\boldsymbol{p}(\tau)\sim P(\theta(\tau))}[\boldsymbol{p}(\tau)]$ is the expected probability vector. Even more, since we do not consider a point estimate, we can get the amount of certainty in a prediction. For this, we estimate the probability of class c being more likely than the other classes, given by $q_c(\tau) := \mathbb{E}_{\boldsymbol{p}(\tau)\sim P(\theta(\tau))}[\mathbb{1}_{\boldsymbol{p}(\tau)_c \geq \max_{c' \neq c} \boldsymbol{p}(\tau)_{c'}}]$. This tells us how certain we are that one class is the most probable (i.e. 'how often' is c the argmax when sampling from P).

Two expressive and well-established choices for the family P are the Dirichlet distribution and the logistic-normal distribution (Appendix F.1). Based on a common modeling idea, we present two models that exploit the specificities of these distributions: the WGP-LN (Section 9.2.1) and the FD-Dir (Section 9.2.2). We also introduce a novel loss to train these models in Section 9.2.3.

Independent of the chosen model, we have to tackle two core challenges: (1) **Expressiveness.** Since the time dependence of $\theta(\tau)$ may be of different forms, we need to capture complex behavior. (2) **Locality.** For regions out of the observed data we want to have a higher uncertainty in our predictions. Specifically for $\tau \to \infty$, i.e. far into the future, the distribution should have a high uncertainty.

9.2.1 Logistic-Normal via a Weighted Gaussian Process (WGP-LN)

We start by describing our model for the case when P is the family of logistic-normal (LN) distributions. How to model a compact yet expressive evolution of the LN distribution? Our core idea is to exploit the fact that the LN distribution corresponds to a multivariate random variable whose *logits* follow a *normal distribution* – and a natural way to model the evolution of a normal distribution is a *Gaussian Process*. Given this insight, the core idea of our model is illustrated in Fig. 9.2: (1) we generate M pseudo points based on a hidden state of an RNN whose input is a sequence, (2) we fit a Gaussian Process to the pseudo points, thus capturing the temporal evolution, and (3) we use the learned GP for estimating the parameters $\boldsymbol{\mu}(\tau)$ and $\boldsymbol{\Sigma}(\tau)$ of the final LN distribution at any specific time τ. Thus, by generating a small number of points we characterize the full distribution.

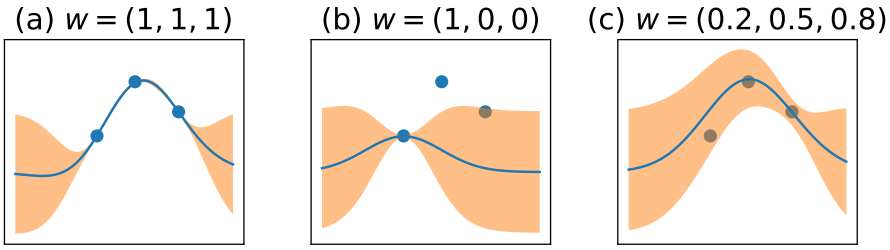


Figure 9.3: WGP on toy data with different weights. (a) All weights are 1 – classic GP. (b) Zero weights discard points. (c) Mixed weight assignment.

Classic GP. To keep the complexity low, we train one GP per class c. That is, our model generates M points $(\tau_j^{(c)}, y_j^{(c)})$ per class c, where $y_j^{(c)}$ represents logits. Note that the first coordinate of each pseudo point corresponds to time, leading to the temporal evolution when fitting the GP. Essentially we perform a non-parameteric regression from the time domain to the logit space. Indeed, using a classic GP along with the pseudo points, the parameters θ of the logistic-normal distribution, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, can be easily computed for any time τ in closed form:

$$\mu_c(\tau) = \boldsymbol{k}_c^T \boldsymbol{K}_c^{-1} \boldsymbol{y}_c, \;\; \sigma_c^2(\tau) = s_c - \boldsymbol{k}_c^T \boldsymbol{K}_c^{-1} \boldsymbol{k}_c \tag{9.1}$$

where $\boldsymbol{K}_c$ is the gram matrix w.r.t. the M pseudo points of class c based on a kernel k (e.g. $k(\tau_1, \tau_2) = \exp(-\gamma^2(\tau_1 - \tau_2)^2)$). Vector $\boldsymbol{k}_c$ contains at position j the value $k(\tau_j^{(c)}, \tau)$, and $\boldsymbol{y}_c$ the value $y_j^{(c)}$, and $s_c = k(\tau, \tau)$. At every time point τ the logits then follow a multivariate normal distribution with mean $\boldsymbol{\mu}(\tau)$ and covariance $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2(\tau))$.

Using a GP enables us to describe complex functions. Furthermore, since a GP models uncertainty in the prediction depending on the pseudo points, uncertainty is higher in areas far away from the pseudo points. Specifically, it holds for distant future; thus, matching the idea of locality. However, uncertainty is always low around the M pseudo points. Thus M should be carefully picked since there is a trade-off between having high certainty at (too) many time points and the ability to capture complex behavior. Thus, in the following we present an extended version solving this problem.

Weighted GP. We would like to pick M large enough to express rich multimodal functions and allow the model to discard unnecessary points. To do this we generate an additional weight vector $\boldsymbol{w}^{(c)} \in [0, 1]^M$ that assigns the weight $w_j^{(c)}$ to a point $\tau_j^{(c)}$. Giving a zero weight to a point should discard it, and giving 1 will return the same result as with a classic GP. To achieve this goal, we introduce a new kernel function:

$$k'(\tau_1, \tau_2) = f(w_1, w_2) k(\tau_1, \tau_2) \tag{9.2}$$

where k is the same as above. The function f weights the kernel k according to the weigths for τ_1 and τ_2. We require f to have

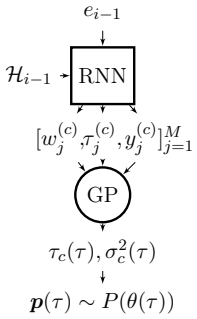


Figure 9.4: Model diagram

the following properties: (1) f should be a valid kernel over the weights, since then the function k' is a valid kernel as well; (2) the importance of pseudo points should not increase, giving $f(w_1, w_2) \leq \min(w_1, w_2)$; this fact implies that a point with zero weight will be discarded since $f(w_1, 0) = 0$ as desired. The function $f(w_1, w_2) = \min(w_1, w_2)$ is a simple choice that fulfills these properties. In Fig. 9.3 we show the effect of different weights when fitting of a GP (see Appendix F.2 for a more detailed discussion of the behavior of the min kernel). To predict μ and σ^2 for a new time τ, we can now simply apply Eq. (9.1) based on the new kernel k', where the weight for the *query* point τ is 1.

To summarize: From a hidden state $h_i = \text{RNN}(e_{i-1}, \mathcal{H}_{i-1})$ we use a a neural network to generate M weighted pseudo points $(w_j^{(c)}, \tau_j^{(c)}, x_j^{(c)})$ per class c. Fitting a Weighted GP to these points enables us to model the temporal evolution of $\mathcal{N}(\mu_c(\tau), \sigma_c^2(\tau))$ and, thus, accordingly of the logistic-Normal distribution. Fig. 9.4 shows an illustration of this model. Note that the cubic complexity of a GP, due to the matrix inversion, is not an issue since the number M is usually small (< 10), while still allowing to represent rich multimodal functions. Crucially, given the loss defined in Section 9.2.3, our model is fully differentiable, enabling us efficient training.

9.2.2 Dirichlet via a Function Decomposition (FD-Dir)

Next, we consider the Dirichlet distribution to model the uncertainty in the predictions. The goal is to model the evolution of the concentrations parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_C)^T$ of the Dirichlet over time. Since unlike to the logistic-normal, we cannot draw the connection to the GP, we propose to decompose the parameters of the Dirichlet distribution with expressive (local) functions in order to allow complex dependence on time. Since the concentration parameters $\alpha_c(\tau)$ need to be positive, we propose the following decomposition of $\alpha_c(\tau)$ in the log-space

$$\log \alpha_c(\tau) = \sum_{j=1}^{M} w_j^{(c)} \cdot \mathcal{N}(\tau | \tau_j^{(c)}, \sigma_j^{(c)}) + \nu \tag{9.3}$$

where the real-valued scalar ν is a constant prior on $\log \alpha_c(\tau)$ which takes over in regions where the Gaussians are close to 0.

The decomposition into a sum of Gaussians is beneficial for various reasons: **(i)** First note that the concentration parameter α_c can be viewed as the effective number of observations of class c. Accordingly the larger $\log \alpha$, the more certain becomes the prediction. Thus, the functions $\mathcal{N}(\tau | \tau_j^{(c)}, \sigma_j^{(c)})$ can describe time regions where we observed data and, thus, should be more certain; i.e. regions around the time $\tau_j^{(c)}$ where the 'width' is controlled by $\sigma_j^{(c)}$. **(ii)** Since most of the functions' mass is centered around their mean, the locality property is fulfilled. Put differently: In regions where we did not observed data (i.e. where the functions $\mathcal{N}(\tau | \tau_j^{(c)}, \sigma_j^{(c)})$ are close to 0), the value $\log \alpha_c(\tau)$ is close to the prior value ν. In the experiments, we use $\nu = 0$, thus $\alpha_c(\tau) = 1$ in the out of observed data regions; a common (uninformative) prior value for the Dirichlet parameters. Specifically for $\tau \to \infty$ the resulting predictions have a high uncertainty. **(iii)** Lastly, a linear combination of translated Gaussians is able to approximate a wide family

of functions [58]. And similar to the weighted GP, the coefficients $w_j^{(c)}$ allow discarding unnecessary basis functions.

The basis functions parameters $(w_j^{(c)}, \tau_j^{(c)}, \sigma_j^{(c)})$ are the output of the neural network, and can also be interpreted as weighted pseudo points that determine the regression of Dirichlet parameters $\theta(\tau)$, i.e. $\alpha_c(\tau)$, over time (Figs. 9.2 and 9.4). The concentration parameters $\alpha_c(\tau)$ themselves have also a natural interpretation: they can be viewed as the rate of events after time gap τ.

9.2.3 Model Training with the Distributional Uncertainty Loss

The core feature of our models is to perform predictions in the future with uncertainty. The classical cross-entropy loss, however, is not well suited to learn uncertainty on the categorical distribution since it is only based on a single (point estimate) of the class distribution. That is, the standard cross-entropy loss for the i^{th} event between the true categorical distribution $\boldsymbol{p}_i^*$ and the predicted (mean) categorical distribution $\overline{\boldsymbol{p}}_i$ is $\mathcal{L}_i^{\text{CE}} = \mathrm{H}[\boldsymbol{p}_i^*, \overline{\boldsymbol{p}}_i(\tau_i^*)] = -\sum_c p_{ic}^* \log \overline{p}_{ic}(\tau_i^*)$. Due to the point estimate $\overline{\boldsymbol{p}}_i(\tau) = \mathbb{E}_{\boldsymbol{p}_i \sim P_i(\theta(\tau))}[\boldsymbol{p}_i]$, the uncertainty on $\boldsymbol{p}_i$ is completely neglected.

Instead, we propose the uncertainty cross-entropy which takes into account uncertainty:

$$\mathcal{L}_i^{\text{UCE}} = \mathbb{E}_{\boldsymbol{p}_i \sim P_i(\theta(\tau_i^*))}[\mathrm{H}[\boldsymbol{p}_i^*, \boldsymbol{p}_i]] = -\int P_i(\theta(\tau_i^*)) \sum_c p_{ic}^* \log p_{ic} \tag{9.4}$$

Remark that the uncertainty cross-entropy does not use the compound distribution $\overline{\boldsymbol{p}}_i(\tau)$ but considers the expected cross-entropy. Based on Jensen's inequality, it holds: $0 \leq \mathcal{L}_i^{\text{CE}} \leq \mathcal{L}_i^{\text{UCE}}$. Consequently, a low value of the uncertainty cross-entropy guarantees a low value for the classic cross entropy loss, while additionally taking the variation in the class probabilities into account. A comparison between the classic cross entropy and the uncertainty cross-entropy on a simple classification task and anomaly detection in asynchronous event setting is presented in Appendix F.6.

In practice the true distribution $\boldsymbol{p}_i^*$ is often a one hot-encoded representation of the observed class c_i which simplifies the computations. During training, the models compute $P_i(\theta(\tau))$ and evaluate it at the true time of the next event τ_i^* given the past event e_{i-1} and the history $\mathcal{H}_{i-1}$. The final loss for a sequence of events is simply obtained by summing up the loss for each event $\mathcal{L} = \sum_i \mathbb{E}_{\boldsymbol{p}_i \sim P_i(\theta(\tau_i^*))}[\mathrm{H}[\boldsymbol{p}_i^*, \boldsymbol{p}_i]]$.

Fast computation. In order to have an efficient computation of the uncertainty cross-entropy, we propose closed-form expressions. *(1) Closed-form loss for Dirichlet.* Given that the observed class c_i is one hot-encoded by $\boldsymbol{p}_i^*$, the uncertain loss can be computed in closed form for the Dirichlet:

$$\mathcal{L}_i^{\text{UCE}} = \mathbb{E}_{\boldsymbol{p}_i(\tau_i^*) \sim \mathbf{Dir}(\boldsymbol{\alpha}(\tau_i^*))}[\log p_{c_i}(\tau_i^*)] = \Psi(\alpha_{c_i}(\tau_i^*)) - \Psi(\alpha_0(\tau_i^*)) \tag{9.5}$$

where Ψ denotes the digamma function and $\alpha_0(\tau_i^*) = \sum_c^C \alpha_c(\tau_i^*)$. *(2) Loss approximation for GP.* For WGP-LN, we approximate $\mathcal{L}_i^{\text{UCE}}$ based on second order series expansion

(Appendix F.3):

$$\mathcal{L}_i^{\text{UCE}} \approx \mu_{c_i}(\tau_i^*) - \log\Big(\sum_c^C \exp(\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*)/2)\Big) + \frac{\sum_c^C (\exp(\sigma_c^2(\tau_i^*)) - 1)\exp(2\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*))}{2\Big(\sum_c^C \exp(\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*)/2)\Big)^2} \tag{9.6}$$

Note that we can now fully backpropagate through our loss (and through the models as well), enabling to train our methods efficiently with automatic differentiation frameworks and, e.g., gradient descent.

Regularization. While the above loss much better incorporates uncertainty, it is still possible to generate pseudo points with high weight values outside of the observed data regime giving us predictions with high confidence. To eliminate this behaviour we introduce a regularization term r_c:

$$r_c = \alpha \underbrace{\int_0^T (\mu_c(\tau))^2 \, d\tau}_{\text{Pushes mean to 0}} + \beta \underbrace{\int_0^T (\nu - \sigma_c^2(\tau))^2 \, d\tau}_{\text{Pushes variance to } \nu} \tag{9.7}$$

For the WGP-LN, $\mu_c(\tau)$ and $\sigma_c(\tau)$ correspond to the mean and the variance of the class logits which are pushed to prior values of 0 and ν. For the FD-Dir, $\mu_c(\tau)$ and $\sigma_c(\tau)$ correspond to the mean and the variance of the class probabilities where the regularizer on the mean can actually be neglected because of the prior ν introduced in the function decomposition (Eq. (9.3)). In experiments, ν is set to 1 for WGP-LN and $\frac{C-1}{C^2(C+1)}$ for FD-Dir which is the variance of the classic Dirichlet prior with concentration parameters equal to 1. For both models, this regularizer forces high uncertainty on the interval $(0, T)$. In practice, the integrals can be estimated with Monte-Carlo sampling whereas α and β are hyperparameters which are tuned on a validation set.

In [270], to train models capable of uncertain predictions, another dataset or a generative models to access out of observed distribution samples is required. In contrast, our regularizer suggests a simple way to consider out of distribution data which does not require another model or dataset.

9.3 Point Process Framework

Our models FD-Dir and WGP-LN predict $P(\theta(\tau))$, enabling to evaluate, e.g., $\overline{\boldsymbol{p}}$ after a specific time gap τ. This corresponds to a conditional distribution $q(c|\tau) := \overline{p}_c(\tau)$ over the classes. In this section, we introduce a *point process* framework to generalize FD-Dir to also predict the time distribution $q(\tau)$. This enables us to predict, e.g., the most likely time the next event is expected or to evaluate the joint distribution $q(c|\tau) \cdot q(\tau)$. We call the model FD-Dir-PP.

We modify the model so that each class c is modelled using an inhomogeneous Poisson point process with positive locally integrable intensity function $\lambda_c(\tau)$. Instead of generating parameters $\theta(\tau) = (\alpha_1(\tau), ..., \alpha_C(\tau))$ by function decomposition, FD-Dir-PP

generates intensity parameters over time: $\log \lambda_c(\tau) = \sum_{j=1}^{M} w_j^{(c)} \mathcal{N}(\tau | \tau_j^{(c)}, \sigma_j^{(c)}) + \nu$. The main advantage of such general decomposition is its potential to describe complex multimodal intensity functions contrary to other models like RMTPP [109] (Appendix F.4). Since the concentration parameter $\alpha_c(\tau)$ and the intensity parameter $\lambda_c(\tau)$ both relate to the number of events of class c around time τ, it is natural to convert one to the other.

Given this C-multivariate point process, the probability of the next class given time and the probability of the next event time are $q(c|\tau) = \frac{\lambda_c(\tau)}{\lambda_0(\tau)}$ and $q(\tau) = \lambda_0(\tau) e^{-\int_0^\tau \lambda_0(s) ds}$ where $\lambda_0(\tau) = \sum_{c=1}^{C} \lambda_c(\tau)$. Since the classes are now modelled via a point proc., the log-likelihood of the event $e_i = (c_i, \tau_i^*)$ is:

$$\log q(c_i, \tau_i^*) = \log q(c_i|\tau_i^*) + \log q(\tau_i^*) = \underbrace{\log \frac{\lambda_{c_i}(\tau_i^*)}{\lambda_0(\tau_i^*)}}_{(i)} + \underbrace{\log \lambda_0(\tau_i^*)}_{(ii)} - \underbrace{\int_0^{\tau_i^*} \lambda_0(t) dt}_{(iii)} \tag{9.8}$$

The terms (ii) and (iii) act like a regularizer on the intensities by penalizing large cumulative intensity $\lambda_0(\tau)$ on the time interval $[t_{i-1}, t_i]$ where no events occurred. The term (i) is the standard cross-entropy loss at time τ_i. Or equivalently, by modeling the distribution $\mathbf{Dir}(\lambda_1(\tau), .., \lambda_C(\tau))$, we see that term (i) is equal to $\mathcal{L}_i^{\text{CE}}$ (see Section 9.2.3). Using this insight, we obtain our final FD-Dir-PP model: We achieve uncertainty on the class prediction by modeling $\lambda_c(\tau)$ as concentration parameters of a Dirichlet distribution and train the model with the loss of Eq. (9.8) replacing term (i) by $\mathcal{L}_i^{\text{UCE}}$. As it becomes apparent FD-Dir-PP differs from FD-Dir only in the regularization of the loss function, enabling it to be interpreted as a point process.

9.4 Related Work

Predictions based on discrete sequences of events regardless of time can be modelled by Markov Models [28] or RNNs, usually with its more advanced variants like LSTMs [189] and GRUs [80]. To exploit the time information some models [254, 312] additionally take time as an input but still output a single prediction for the entire future. In contrast, temporal point process framework defines the intensity function that describes the rate of events occuring over time.

RMTPP [109] uses an RNN to encode the event history into a vector that defines an exponential intensity function. Hence, it is able to capture complex past dependencies and model distributions resulting from simple point processes, such as Hawkes [179] or self-correcting [205], but not e.g. multimodal distributions. On the other hand, Neural Hawkes Process [285] uses continuous-time LSTM which allows specifying more complex intensity functions. Now the likelihood evaluation is not in closed-form anymore, but requires Monte Carlo integration. However, these approaches, unlike our models, do not provide any uncertainty in the predictions. In addition, WGP-LN and FD-Dir can be extended with a point process framework while having the expressive power to represent complex time evolutions.

Uncertainty in machine learning has shown a great interest [138, 124, 246]. For example, uncertainty can be imposed by introducing distributions over the weights

[42, 283, 359]. Simpler approaches introduce uncertainty directly on the class prediction by using Dirichlet distribution independent of time [270, 368]. In contrast, the FD-Dir model models complex temporal evolution of Dirichlet distribution via function decomposition which can be adapted to have a point process interpretation.

Other methods introduce uncertainty time series prediction by learning state space model with Gaussian processes [119, 418]. Alternatively, RNN architecture has been used to model the probability density function over time [456]. Compared to these models, the WGP-LN model uses both Gaussian processes and RNN to model uncertainty and time. Our models are based on pseudo points. Pseudo points in a GP have been used to reduce the computational complexity [394]. Our goal is not to speed up the computation, since we control the number of points that are generated, but to give them different importance. In [433] a weighted GP has been considered by rescaling points; in contrast, our model uses a custom kernel to discard (pseudo) points.

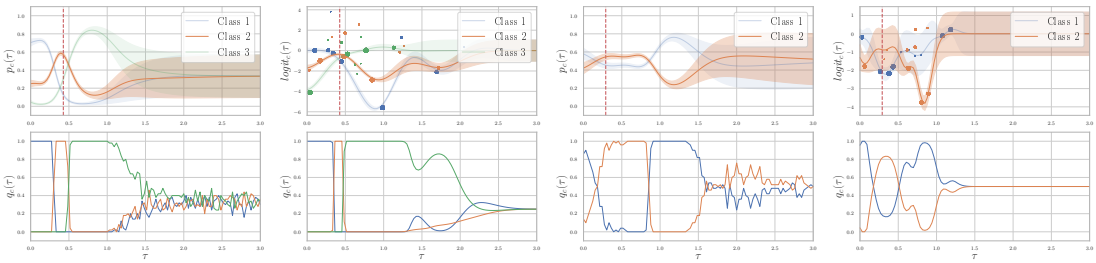
9.5 Experiments

We evaluate our models on large-scale synthetic and real world data. We compare to neural point process models: **RMTPP** [109] and **Neural hawkes process** [285]. Additionally, we use various RNN models with the knowledge of the time of the next event. We measure the accuracy of class prediction, accuracy of time prediction, and evaluate on an anomaly detection task to show prediction uncertainty.

We split the data into train, validation and test set (60%–20%–20%) and tune all models on a validation set using grid search over learning rate, hidden state dimension and L_2 regularization. After running models on all datasets 5 times we report mean and standard deviation of test set accuracy. Details on model selection can be found in Appendix F.8.1. The code and further supplementary material is available online.[1]

We use the following data (more details in Appendix F.7): (1) **Graph.** We generate data from a directed Erdős–Rényi graph where nodes represent the states and edges the weighted transitions between them. The time it takes to cross one edge is modelled with one normal distribution per edge. By randomly walking along this graph we created 10K asynchronous events with 10 unique classes. (2) **Stack Exchange.**[2] Sequences contain rewards as events that users get for participation on a question answering website. After preprocessing according to [109] we have 40 classes and over 480K events spread over 2 years of activity of around 6700 users. The goal is to predict the next reward a user will receive. (3) **Smart Home** [413].[3] We use a recorded sequence from a smart house with 14 classes and over 1000 events. Events correspond to the usage of different appliances. The next event will depend on the time of the day, history of usage and other appliances. (4) **Car Indicators.** We obtained a sequence of events from car's indicators that has around 4000 events with 12 unique classes. The sequence is highly asynchronous, with τ ranging from milliseconds to minutes.

[1] `https://www.daml.in.tum.de/uncertainty-event-prediction`
[2] `https://archive.org/details/stackexchange`
[3] `https://sites.google.com/site/tim0306/datasets`



FD-Dir on 3-G | WGP-LN on 3-G | FD-Dir on Multi-G | WGP-LN on Multi-G

Figure 9.5: Visualization of the prediction evolution. The red line indicates the true time of the next event for an example sequence. Here, both models predict the orange class, which is correct, and capture the variation of the class distributions over time. Generated points from WGP-LN are plotted with the size corresponding to the weight. For predictions in the far future, both models given high uncertainty.

Visualization. To analyze the behaviour of the models, we propose visualizations of the evolutions of the parameters predicted by FD-Dir and WGP-LN.

Set-up: We use two toy datasets where the probability of an event depends only on time. The first one (**3-G**) has three classes occuring at three distinct times. It represents the events in the Fig. F.6a. The second one (**Multi-G**) consists of two classes where one of them has two modes and corresponds to the Fig. 9.1a. We use these datasets to showcase the importance of time when predicting the next event. In Fig. 9.5, the four top plots show the evolution of the categorical distribution for the FD-Dir and the logits for the WGP-LN with 10 points each. The four bottom plots describe the certainty of the models on the probability prediction by plotting the probability $q_c(\tau)$ that the probability of class c is higher than others, as introduced in Section 9.2. Additionally, the evolution of the dirichlet distribution over the probability simplex is presented in Appendix F.5.

Results. Both models learn meaningful evolutions of the distribution on the simplex. For the 3-G data, we can distinguish four areas: the first three correspond to the three classes; after that the prediction is uncertain. The Multi-G data shows that both models are able to approximate multimodal evolutions.

Class prediction accuracy. The aim of this experiment is to assess whether our models can correctly predict the class of the next event, given the time at which it occurs. For this purpose, we compare our models against Hawkes and RMTPP and evalute prediction accuracy on the test set.

Results. We can see (Fig. 9.6) that our models consistently outperform the other methods on all datasets. Results of the other baselines can be found in Appendix F.8.2.

Time-Error evaluation. Next, we aim to assess the quality of the time intervals at which we have confidence in one class. Even though WGP-LN and the FD-Dir do not model a distribution on time, they still have intervals at which we are certain in a class

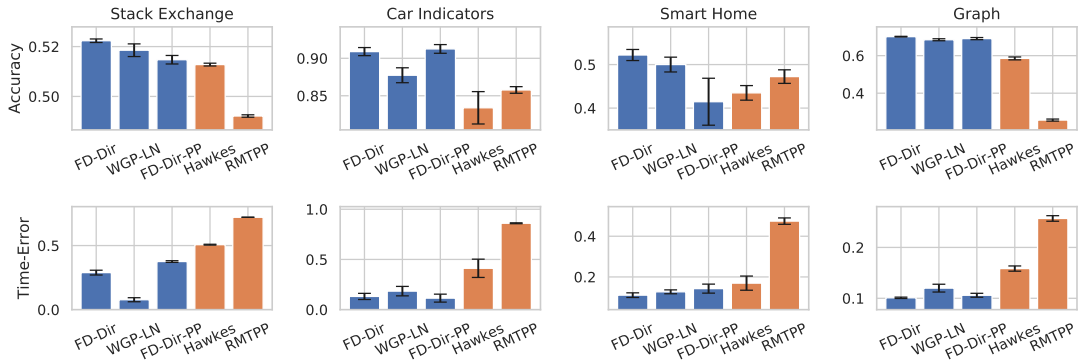


Figure 9.6: Class accuracy (top; higher is better) and Time-Error (bottom; lower is better).

prediction, making the conditional probability a good indicator of the time occurrence of the event.

Set-up. While models predicting a *single* time $\hat{\tau}_i$ for the next event often use the MSE score $\frac{1}{n}\sum_{i=1}^{n}(\hat{\tau}_i - \tau_i^*)^2$, in our case the MSE is not suitable since one event can occur at multiple time points. In the conventional least-squares approach, the mean of the true distribution is an optimal prediction; however, here it is almost always wrong. Therefore, we use another metric which is better suited for multimodal distributions. Assume that a model returns a score function $g_i^{(c)}(\tau)$ for each class regarding the next event i, where a large value means the class c is likely to occur at time τ. We define Time-Error $= \frac{1}{n}\sum_{i=1}^{n}\int \mathbb{1}_{g_i^{(c)}(\tau) \geq g_i^{(c)}(\tau_i^*)} d\tau$. The Time-Error computes the size of the time intervals where the predicted score is larger than the score of the observed time τ_i^*. Hence, a performant model would achieve a low Time-Error if its score function $g_i^{(c)}(\tau)$ is high at time τ^*. As the score function in our models, we use the corresponding class probability $\bar{p}_{ic}(\tau)$.

Results. We can see that our models clearly obtain the best results on all datasets. The point process version of FD-Dir does not improve the performance. Thus, taking also into account the class prediction performance, we recommend to use our other two models. In Appendix F.8.3 we compare FD-Dir-PP with other neural point process models on time prediction using the MSE score and achieve similar results.

Anomaly detection & Uncertainty. The goal of this experiment is twofold: (1) it assesses the ability of the models to detect anomalies in asynchronous sequences, (2) it evaluates the quality of the predicted uncertainty on the categorical distribution. For this, we use a similar set-up as [270].

Set-up: The experiments consist in introducing anomalies in datasets by changing the occurrence time of 10% of the events (at random after the time transformation described in appendix F.7). Hence, the anomalies form out-of-distribution data, whereas unchanged events represent in-distribution data. The performance of the anomaly detection is assessed using Area Under Receiver Operating Characteristic (AUROC) and Area Under Precision-Recall (AUPR). We use two approaches: (i) We consider the *categorical uncertainty* on $\bar{\boldsymbol{p}}(\tau)$, i.e., to detect anomalies we use the predicted probability

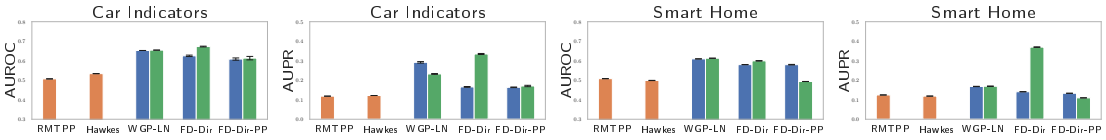


Figure 9.7: AUROC and APR comparison across dataset on anomaly detection. The orange and blue bars use categorical uncertainty score whereas the green bars use distributional uncertainty.

of the true event as the anomaly score. (ii) We use the *distribution uncertainty* at the observed occurrence time provided by our models. For WGP-LN, we can evaluate $q_c(\tau)$ directly (difference of two normal distributions). For FD-Dir, this probability does not have a closed-form solution so instead, we use the concentration parameters which are also indicators of out-of-distribution events. For all scores, i.e $\bar{\boldsymbol{p}}(\tau)_c$, $q_c(\tau)$ and $\alpha_c(\tau)$, a low value indicates a potential anomaly around time τ.

Results. As seen in Fig. 9.5, the FD-Dir and the WGP-LN have particularly good performance. We observe that the FD-Dir gives better results especially with distributional uncertainty. This might be due to the power of the concentration parameters that can be viewed as number of similar events around a given time.

9.6 Conclusion

We proposed two new methods to predict the evolution of the probability of the next event in asynchronous sequences, including the distributions' uncertainty. Both methods follow a common framework consisting in generating pseudo points able to describe rich multimodal time-dependent parameters for the distribution over the probability simplex. The complex evolution is captured via a Gaussian Process or a function decomposition, respectively; still enabling easy training. We also provided an extension and interpretation within a point process framework. In the experiments, WGP-LN and FD-Dir have clearly outperformed state-of-the-art models based on point processes; for event and time prediction as well as for anomaly detection.

10 Uncertainty Estimation for Reinforcement Learning

Knowledge progresses by integrating uncertainty into itself,
not by exorcising it

Edgar Morin

10.1 Introduction

In the previous chapters, we focused on application of uncertainty estimation in supervised tasks where target labels are provided by the task during training. In contrast, we consider in this chapter the application of uncertainty estimation to reinforcement learning tasks where the agent aims to maximize its total return without taking excessive risks.

We generally want to satisfy three important properties for a reliable deployment of an RL agent in real-world applications: **(i)** The agent should learn *fast* with as few episode failures as possible. **(ii)** The agent should *maintain high reward* when facing new environments similar to the training environment after deployment. **(iii)** The agent should *flag anomalous environment states* when it does not know what action to take in an unknown environment. These three practically desirable properties translate into three technical properties in reinforcement learning agents. Indeed, a reinforcement learning agent should achieve high *sample efficiency* at training time [430], high *generalization* performance on test environments similar to the training environment [159], and high *Out-Of-Distribution (OOD) detection* scores on environments unrelated to the training task [453, 321].

In this chapter, we argue that *aleatoric* and *epistemic* uncertainty are key concepts to achieve these desired practical and technical properties. The aleatoric uncertainty represents the irreducible and inherent stochasticity of the environment. Thus, an environment region with high aleatoric uncertainty is unlikely to be interesting to explore at training time because it could be uninformative (e.g. a sensor is very noisy) or dangerous (e.g. the environment has an unpredictable behavior). In contrast, the epistemic uncertainty represents the lack of information for accurate prediction. Thus, an environment region with high epistemic uncertainty is potentially promising to explore to build a better understanding of the environment (e.g., a state has unknown transition dynamics because it has never been explored).

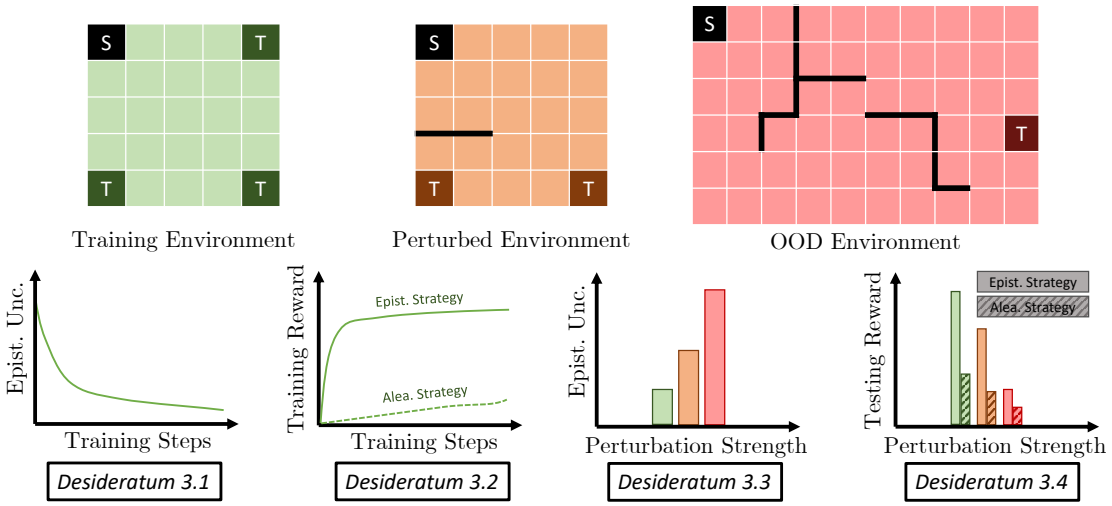


Figure 10.1: Overview of our proposed desiderata for uncertainty in RL (See Section 10.3).

The core motivation of this chapter is to disentangle the properties of aleatoric and epistemic uncertainty estimates in RL to build agents with reliable performance in real-world applications. This motivation is similar to supervised learning (SL) where previous works defined *desiderata*, *models*, and *evaluation* methods for aleatoric and epistemic uncertainty [140, 3, 335, 235]. Important examples of models using a single or multiple forward passes for uncertainty estimation in SL are MC dropout [141], ensemble [246, 439, 434], deep kernel learning [268, 421, 420, 36], and evidential networks [67, 270, 69, 10]. Further, empirical evaluation of uncertainty estimates in SL focuses *only* on testing time with Out-Of-Distribution (OOD) detection and generalization or detection of shifts [335, 278]. In contrast to SL, the RL setting is more complex because it involves the performance of uncertainty estimates at *both* training and testing time.

Our Contributions. In this chapter, we propose a framework for aleatoric and epistemic uncertainty estimation in RL: **(Desiderata)** We explicitly define four desiderata for uncertainty estimation in RL at both *training* and *testing time* (See Fig. 10.1). They cover the behavior of *aleatoric* and *epistemic* uncertainty estimates w.r.t. the sample efficiency in the training environment, and anomaly detection and generalization performance in different testing environments. **(Models)** We carefully combine a diverse set of uncertainty estimation methods in SL (i.e. MC dropout, ensemble, deep kernel learning, and evidential networks) with Deep Q-Networks (DQN) [291], a RL model that is not equipped with uncertainty estimates by default. These combinations require a *minimal* modification to the training procedure of the RL agent. We discuss *theoretical* evidence on the ability of these combinations to fulfill the uncertainty desiderata. **(Evaluation)** Finally, we also propose a *practical* methodology to evaluate uncertainty in RL based on OOD environments and domain shifts..

10.2 Problem Setup

Uncertainty in SL. The objective of SL is to accurately predict the output $y^{(i)}$ given an input $\boldsymbol{x}^{(i)}$ with index i. It differentiates between two types of uncertainty: the uncertainty on the label prediction $y^{(i)}$ described by the *aleatoric distribution* $\mathbb{P}(y^{(i)} \mid \boldsymbol{\theta}^{(i)})$ with parameters $\boldsymbol{\theta}^{(i)}$ estimated from the input $\boldsymbol{x}^{(i)}$, and the uncertainty on the predicted label distribution parameters $\boldsymbol{\theta}^{(i)}$ described by the *epistemic distribution* $\mathbb{Q}(\boldsymbol{\theta}^{(i)} \mid \boldsymbol{\chi}^{(i)})$ with parameters $\boldsymbol{\chi}^{(i)}$ estimated from the input $\boldsymbol{x}^{(i)}$. Intuitively, the variations of the aleatoric distribution will be high when the input $\boldsymbol{x}^{(i)}$ does not provide discriminative information to determine the label $y^{(i)}$. The variations of the epistemic distribution will be high when the input $\boldsymbol{x}^{(i)}$ does not provide enough information to determine the label distribution $\mathbb{Q}(\boldsymbol{\theta}^{(i)} \mid \boldsymbol{\chi}^{(i)})$ described by the parameters $\boldsymbol{\chi}^{(i)}$. Thus, the aleatoric uncertainty can be measured in practice by computing the entropy, i.e $u_{\text{alea}}(\boldsymbol{x}^{(i)}) = \mathbb{H}(\mathbb{P}(y^{(i)} \mid \boldsymbol{\theta}^{(i)}))$, or the variance, i.e. $u_{\text{alea}}(\boldsymbol{x}^{(i)}) = \text{Var}(\mathbb{P}(y^{(i)} \mid \boldsymbol{\theta}^{(i)}))$, of the aleatoric distribution, while the epistemic uncertainty can be measured by computing the entropy, i.e. $u_{\text{epist}}(\boldsymbol{x}^{(i)}) = \mathbb{H}(\mathbb{Q}(\boldsymbol{\theta}^{(i)} \mid \boldsymbol{\chi}^{(i)})$, of the epistemic distribution [270, 67, 69]. Further, SL also distinguishes between sampling-based which often require multiple forward passes for uncertainty estimation, and sampling-free methods which often require a single forward pass for uncertainty estimation. Sampling-based methods like MC dropout [141] and ensemble [246, 439, 434] estimate uncertainty by aggregating statistics (e.g. mean and variance) from different samples which *implicitly* describe the epistemic distribution $\mathbb{Q}(\boldsymbol{\theta}^{(i)} \mid \boldsymbol{\chi}^{(i)})$. Sampling-free methods like deep kernel learning models [268, 421, 420, 36] and evidential networks [67, 270, 69, 10] estimate uncertainty by *explicitly* parametrizing the epistemic distributions $\mathbb{Q}(\boldsymbol{\theta}^{(i)}) \mid \boldsymbol{\chi}^{(i)})$ with known distributions such as Normal and Normal Inverse-Gamma (NIG) distributions, thus enabling efficient and closed-form computation of the distribution statistics (e.g. mean, variance or entropy).

Uncertainty in RL. We consider the task of learning RL policies interacting with an environment with fully observed states at every time step t. The environment is described by a Markov Decision Process (MDP) $(S, A, R, T, \rho, \gamma)$, where S is the state space, A is the action space, $R(\boldsymbol{s}^{(t)}, a^{(t)})$ is the reward associated with the action $a^{(t)}$ and state $\boldsymbol{s}^{(t)}$, $T(\boldsymbol{s}^{(t+1)} | \boldsymbol{s}^{(t)}, a^{(t)})$ is the transition probability, $\rho(\boldsymbol{s}^{(0)})$ is the initial state distribution, and γ is the discount factor. Given the current state $\boldsymbol{s}^{(t)}$, our goal is to learn a policy predicting the action $a^{(t)}$ leading to the highest discounted return $Q^{\pi}(\boldsymbol{s}^{(t)}, a^{(t)}) = R(\boldsymbol{s}^{(t)}, a^{(t)}) + \gamma\, \mathbb{E}_{T,\pi}[Q^{\pi}(\boldsymbol{s}^{(t+1)}, a^{(t+1)})]$ (a.k.a. discounted expected reward) in addition to the aleatoric uncertainty $u_{\text{alea}}(\boldsymbol{s}^{(t)}, a^{(t)})$ and the epistemic uncertainty $u_{\text{epist}}(\boldsymbol{s}^{(t)}, a^{(t)})$ on the predicted return. Similar to SL, the aleatoric and epistemic distributions can be instantiated with $\mathbb{P}(y^{(t)} \mid \boldsymbol{\theta}^{(t)})$ and $\mathbb{Q}(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\chi}^{(t)})$ where the predicted value is the future return i.e. $y^{(t)} = Q^{\pi}(\boldsymbol{s}^{(t)}, a^{(t)})$. Intuitively, the variation of the aleatoric distribution will be high when the current state $\boldsymbol{s}^{(t)}$ and action $a^{(t)}$ only contains noisy information to determine the future return $y^{(t)}$, while the epistemic uncertainty will be high when the current state $\boldsymbol{s}^{(t)}$ and action $a^{(t)}$ does not provide enough information according to the model to determine the return distribution $\mathbb{Q}(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\chi}^{(t)})$ described by the parameters $\boldsymbol{\chi}^{(t)}$. We decide to estimate uncertainty of the return $y^{(t)} = Q^{\pi}(\boldsymbol{s}^{(t)}, a^{(t)})$

for two reasons. First, the return is estimated in many common Deep RL models (e.g. DQN [291], PPO [372], A2C [293]). Second, uncertainty of the return implicitly quantifies uncertainty of the optimal action defined by $a^{(t)} = \arg\max_a Q^\pi(\boldsymbol{s}^{(t)}, a^{(t)})$.

Finally, we consider three action selection strategies which is a crucial choice for exploration and generalization in RL:

- The *epsilon-greedy* strategy [292] which selects the action with the highest predicted reward with probability $1 - \epsilon$ and samples randomly otherwise.
- The *sampling-aleatoric* strategy which takes the action with the highest predicted reward based on one aleatoric distribution sample i.e. $a^{(t)} = \arg\max_a y^{(t)}$ where $y^{(t)} \sim \mathbb{P}(y^{(t)}|\boldsymbol{\theta}^{(t)})$.
- The *sampling-epistemic* strategy which takes the action with the highest predicted reward based on one epistemic distribution sample i.e. $a^{(t)} = \arg\max_a \mathbb{E}_{\mathbb{P}(y^{(t)}|\boldsymbol{\theta}^{(t)})}[y^{(t)}]$ where $\boldsymbol{\theta}^{(t)} \sim \mathbb{Q}(\boldsymbol{\theta}^{(t)}|\boldsymbol{\chi}^{(t)})$. This strategy is similar to Thompson sampling [411].

10.3 Desiderata for Uncertainty Quantification in RL

In this section, we *explicitly* define four intuitive and general desiderata that capture the desired behavior for uncertainty estimates in our RL setup. The desiderata cover *aleatoric* and *epistemic* uncertainty at both *training* and *testing* time. The distinction between aleatoric and epistemic uncertainty is commonly studied in SL [140, 270, 69]. In contrast, RL mostly focuses on accounting for aleatoric uncertainty with risk-sensitive policy or distributional RL to achieve higher returns [29, 30, 93]. We design the desiderata to be informal and generic for two main reasons. First, it allows them to intuitively indicate the expected behavior of the uncertainty estimation without complex mathematical notations [398, 124]. Second, it makes the desiderata application independent and model-agnostic.

The second distinction differentiates between training and testing time relevant to sample efficiency and generalization in RL. In contrast, SL mostly focuses on testing time performance.

Training Time. We describe the desired behavior of uncertainty estimates at training time. First, we describe the desired uncertainty behavior when observing more samples of the training environment.

Desideratum 10.3.1. *An agent interacting longer with one specific environment at training time should become more epistemically confident when predicting actions on this same environment.*

Intuitively, an agent observing more samples from the same environment distribution should accumulate more knowledge through time, thus being more epistemically certain. In practice, des. 10.3.1 expresses that the epistemic uncertainty estimates should reflect the accumulated knowledge, and thus the convergence, of the agent during training. We test des. 10.3.1 in the experiments by tracking the epistemic uncertainty at training time (see Section 10.5). Second, we describe the behavior of the total reward when selecting actions based on uncertainty estimates at training time.

Desideratum 10.3.2. *All else being equal, an agent selecting actions with the sampling-aleatoric strategy at training time should achieve lower sample efficiency than an agent selecting actions with the sampling-epistemic strategy.*

Intuitively, an agent exploring states with more (irreducible) aleatoric uncertainty would gain less knowledge about the environment dynamic than an agent exploring states with high epistemic uncertainty where the agent lacks knowledge. However, there is an important trade-off between over- or under-exploring epistemically uncertain actions which could lead to lower sample efficiency. The sampling-epistemic strategy, which corresponds to Thompson sampling [411], mitigates this exploration-exploitation problem by sampling action w.r.t. the epistemic distribution. Thompson sampling has already *empirically* demonstrated high sample efficiency in deep RL problems [141] and provably achieve low regret in many decision-making problems like multi-arm Bandits [6, 366]. In practice, des. 10.3.2 suggests that an agent should use the sampling-epistemic strategy for a better exploration-exploitation trade-off. We test des. 10.3.2 in the experiments by comparing the sample efficiency of the sampling-aleatoric and the sampling-epistemic strategies during training (see Section 10.5).

Testing Time. We describe the desired behavior of uncertainty estimates at testing time. First, we describe the desired uncertainty behavior when observing samples from an environment different from the training environment.

Desideratum 10.3.3. *At testing time, epistemic uncertainty should be greater when the agent interact with environments that are very different from the original training environments.*

The environment difference can be measured using different distance metrics depending on the task or application requirements [176]. Intuitively, an agent should be less confident when observing new states at test time that were not used to collect knowledge at training time. In practice, des. 10.3.3 suggests that an agent should be able to use epistemic uncertainty estimates to detect states that are abnormal compared to the states observed during training. We test des. 10.3.3 in the experiments by comparing the epistemic uncertainty of the training environment against the uncertainty in noisy environments at testing time (see Section 10.5). Noisy environments include environments with completely random states, and environments with different strengths of perturbation on the original states, actions, or transition dynamics. Depending on the application, different noise or perturbation types might be used to characterize the difference of the testing environment with the training environment. Second, we describe the behavior of the total reward when selecting at testing time actions based on uncertainty aleatoric or epistemic uncertainty estimates.

Desideratum 10.3.4. *All else being equal, an agent sampling actions from the epistemic uncertainty at training and testing time should generalize better at testing time than an agent sampling actions from the aleatoric uncertainty.*

Intuitively, an agent exploring more epistemically uncertain states at training time would collect more knowledge about the environment, thus generalizing to more states at testing time. Further, since the environment dynamic is not directly observed, an

agent should account for the epistemic certainty on the current state to take actions that generalize better at testing time. In particular, it has been shown that the Bayes-optimal Markovian policy at testing time is stochastic in general due to the partially observed MDP dynamic sometimes called epistemic POMDP [159]. In practice, des. 10.3.4 suggests that an agent should use the sampling-epistemic strategy for more robust generalization performance. We test des. 10.3.4 in the experiments (see Section 10.5) by comparing the reward obtained by the sampling-aleatoric and the sampling-epistemic strategies at testing time. The two latter desiderata 10.3.3 and 10.3.4 express an important trade-off between assigning high uncertainty and generalizing to new test environments. Since an agent cannot generalize to all new environments because of the No Free Lunch Theorem [445], an agent should assign higher uncertainty to environments where it does not generalize. We jointly test des. 10.3.3 and des. 10.3.4 in the experiments (see Section 10.5) by tracking the reward and the uncertainty estimates in test environments with different perturbation strengths.

10.4 Models for Uncertainty Quantification in RL

Model-free RL agents commonly rely on learning the expected return $Q^{\pi}(\boldsymbol{s}^{(t)}, a^{(t)})$ associated with taking action $a^{(t)}$ in state $\boldsymbol{s}^{(t)}$ and then following a policy π. It is defined by the Bellman equation: $Q^{\pi}(\boldsymbol{s}^{(t)}, a^{(t)}) = R(\boldsymbol{s}^{(t)}, a^{(t)}) + \gamma\, \mathbb{E}[Q^{\pi}(\boldsymbol{s}^{(t+1)}, a^{(t+1)})]$. Similarly, the optimal policy π^* satisfies the highest expected return $Q^*(\boldsymbol{s}^{(t)}, a^{(t)})$ defined by the optimal Bellman equation [31]:

$$Q^*(\boldsymbol{s}^{(t)}, a^{(t)}) = R(\boldsymbol{s}^{(t)}, a^{(t)}) + \gamma\, \mathbb{E}[\max_{a^{(t+1)}} Q^*(\boldsymbol{s}^{(t+1)}, a^{(t+1)})] \tag{10.1}$$

where the expectation is taken over the random transitions. However, the exact computation of the optimal Q-value is often intractable for large action or state spaces. Therefore, deep RL agents like DQN [291], PPO [372], and A2C [293] aim at approximating the optimal Q-value $Q^{\pi^*}(\boldsymbol{s}^{(t)}, a^{(t)})$ with a neural network $f_{\theta}(\boldsymbol{s}^{(t)}, a^{(t)})$ with parameter $\boldsymbol{\theta}$. In particular, DQN enforces Eq. (10.1) by minimizing the squared temporal difference (TD) error $\|R(\boldsymbol{s}^{(t)}, a^{(t)}) + \gamma \max_{a^{(t+1)}} f_{\boldsymbol{\theta}'}(\boldsymbol{s}^{(t+1)}, a^{(t+1)}) - f_{\boldsymbol{\theta}}(\boldsymbol{s}^{(t)}, a^{(t)})\|_2$, where $f_{\boldsymbol{\theta}}$ is the learned prediction network and $f_{\boldsymbol{\theta}'}$ is the frozen target network regularly updated with the prediction network parameters during training. The TD error minimization is similar to SL regression with a MSE loss between the prediction $\hat{y}^{(t)} = f_{\boldsymbol{\theta}}(\boldsymbol{s}^{(t)}, a^{(t)})$ and the target $y^{(t)} = R(\boldsymbol{s}^{(t)}, a^{(t)}) + \gamma f_{\boldsymbol{\theta}'}(\boldsymbol{s}^{(t+1)}, a^{(t+1)})$ with the key difference that the exploration strategy select the data samples that will be used during training.

Model-free Deep RL agents often show important limitations for uncertainty estimation because of their neural network architecture choice. For, instance, while DQN only outputs a single scalar representing the mean Q-value with no uncertainty estimates, PPO and A2C policies parameterized with standard ReLU networks would provably produce overconfident predictions for extreme input states [182]. In this chapter, we focus on equipping the widely used DQN RL agent with reliable uncertainty estimates. To this end, we combine DQN with four SL architectures for uncertainty estimation (e.g. MC dropout, ensemble, deep kernel learning, and evidential networks) covering a diverse

range of sampling-based and sampling-free methods. These four DQN combinations allow to instantiate both aleatoric and epistemic uncertainty with minimal modifications to the training procedure. We provide a summary of the uncertainty properties of these models in Table 10.1.

Table 10.1: Summary of the uncertainty properties of the models.

	DropOut	Ensemble	Deep Kernel Learning	Evidential Networks
Uncertainty concentration (Des. 10.3.1)	✗	✗	✗	✓
Alea. vs epist. sampling at training time (Des. 10.3.2)	✓	✓	✗	✓
OOD detection (Des. 10.3.3)	✗	✗	✓	✓
Alea. vs epist. sampling at testing time (Des. 10.3.2)	✓	✓	✗	✓

MC Dropout. DQN is combined with MC Dropout [141] in three steps: **(1)** it samples K independent sets of model parameters $\{\boldsymbol{\theta}_k\}_{k=1}^K$ by dropping activations with probability p, **(2)** it performs K forward passes $\mu_k, \sigma_k = f_{\boldsymbol{\theta}_k}(s^{(t)}, a^{(t)})$, and **(3)** it aggregates predictions to form the mean prediction $\mu(s^{(t)}, a^{(t)}) = \frac{1}{K}\sum_{k=1}^K \mu_k$, the aleatoric uncertainty estimate $u_{\text{alea}}(s^{(t)}, a^{(t)}) = \frac{1}{K}\sum_{k=1}^K \sigma_s$, and the epistemic uncertainty estimate $u_{\text{epist}}(s_t, a_t) = \frac{1}{K}\sum_{k=1}^K (\mu_k - \mu(s^{(t)}, a^{(t)}))^2$. In this case, the aleatoric distribution is Gaussian while the epistemic distribution is implicitly represented by the sampled parameters $\{\theta_k\}_{k=1}^K$. Further, the sampling-epistemic strategy is achieved by performing one single forward pass with a single set of sampled model parameters. This is similar to the Thompson sampling strategy used by Gal and Ghahramani [141]. During training, we train the neural network parameters θ by using a Gaussian negative log-likelihood loss. The combination of DQN and dropout has been shown to practically improve sample efficiency Gal and Ghahramani [141]. However, dropout has several limitations. First, the dropout uncertainty estimates *provably* do not concentrate with more observed data [331], thus potentially violating des. 10.3.1. Second, there is no guarantee that dropout produce meaningful uncertainty estimates for extreme input states with a finite number of samples K, thus potentially violating des. 10.3.3. Third, dropout might be computationally expensive for large K value since it would require many forward passes for uncertainty estimation.

Ensemble. DQN is combined with ensembles [246] in three steps: **(1)** it train K independent models with parameters $\{\theta_k\}_{k=1}^K$, **(2)** it performs K forward passes $\mu_k, \sigma_k = f_{\boldsymbol{\theta}_k}(s^{(t)}, a^{(t)})$, and **(3)** it aggregates predictions to form the mean prediction $\mu(\boldsymbol{s}^{(t)}, a^{(t)}) = \frac{1}{K}\sum_{k=1}^K \mu_k$, the aleatoric uncertainty estimate $u_{\text{alea}}(s^{(t)}, a^{(t)}) = \frac{1}{K}\sum_{k=1}^K \sigma_s$, and the epistemic uncertainty estimate $u_{\text{epist}}(s_t, a_t) = \frac{1}{K}\sum_{k=1}^K (\mu_k - \mu(s^{(t)}, a^{(t)}))^2$. In this case, the aleatoric and epistemic distributions have the same form as for MC Dropout. Further, the sampling-epistemic strategy is achieved by performing one single forward pass with one randomly selected network. This is similar to the Thompson sampling strategy used by Osband et al. [330]. We train the K independent neural network parameters θ_k with a Gaussian negative log-likelihood loss. However, an ensemble has several limitations. First, while the combination of DQN with bootstrapped ensemble and prior functions

has been *empirically* shown to improve learning for complex tasks with sparse rewards [330, 331], there is no explicit theoretical or empirical evidence that their uncertainty estimates concentrate with more observed data. Second, there is no guarantee that ensembles produce meaningful uncertainty estimates for extreme input states with a finite number of samples K, thus potentially violating des. 10.3.3. Third, an ensemble is computationally expensive for large K value since it would require many forward passes and many neural networks.

Deep Kernel Learning. DQN is combined with deep kernel learning [421] in three steps: **(1)** it predicts one latent representation of each input state i.e. $\boldsymbol{z}^{(t)} = f_{\boldsymbol{\theta}}(\boldsymbol{s}^{(t)})$, and **(2)** one Gaussian Process per action a defined from a fixed set of K learnable inducing points $\{\boldsymbol{\phi}_{a,k}\}_{k=1}^{K}$ and a predefined positive definite kernel $\kappa(\cdot,\cdot)$ predicts the mean $\mu(\boldsymbol{s}^{(t)}, a)$ and the variance $\sigma(\boldsymbol{s}^{(t)}, a)$ of a Gaussian distribution. We train the neural network parameters $\boldsymbol{\theta}$ and the inducing points $\{\boldsymbol{\phi}_{a,k}\}_{k=1}^{K}$ jointly with a variational ELBO loss similarly to [421]. In this case, the epistemic distribution is Gaussian, i.e. $u_{\text{epist}}(s_t, a_t) = \mathbb{H}(\mathcal{N}(\mu(\boldsymbol{s}^{(t)}, a^{(t)}), \sigma(\boldsymbol{s}^{(t)}, a^{(t)})))$. Indeed, we show *theoretically* that epistemic uncertainty increases far from training data (see Appendix G.1). Thus, the combination of DQN and deep kernel learning does not suffer from arbitrary uncertainty estimates for extreme input states contrary to ReLU networks [182]. However, one of the limitation of deep kernel learning is that it does not disentangle *aleatoric* and *epistemic* uncertainty. This is similar to deep kernel learning methods in SL [421, 420].

Evidential Networks. The combination of DQN and posterior networks [69, 67] which belong to the class of evidential networks consists of three steps: **(1)** an encoder $f_{\boldsymbol{\theta}}$ predicts one latent representation of each input state i.e. $\boldsymbol{z}^{(t)} = f_{\boldsymbol{\theta}}(\boldsymbol{s}^{(t)})$, **(2)** one normalizing flow density estimator $\mathbb{P}(. \mid \boldsymbol{\omega}_a)$ and one linear decoder $g_{\boldsymbol{\psi}_a}$ per action a predict a Normal Inverse-Gamma distribution $\mathbb{Q}(\boldsymbol{\chi}(\boldsymbol{s}^{(t)}, a), n(\boldsymbol{s}^{(t)}, a))$ with parameters $\boldsymbol{\chi}(\boldsymbol{s}^{(t)}, a) = g_{\boldsymbol{\psi}_a}(\boldsymbol{z}^{(t)}, a)$ and $n(\boldsymbol{s}^{(t)}, a) \propto \mathbb{P}(\boldsymbol{z}^{(i)} \mid \boldsymbol{\omega}_a)$, and **(3)** it computes the posterior parameters $\boldsymbol{\chi}^{\text{post}}(\boldsymbol{s}^{(t)}, a) = \frac{n^{\text{prior}}\boldsymbol{\chi}^{\text{prior}} + n(\boldsymbol{s}^{(t)}, a))\boldsymbol{\chi}(\boldsymbol{s}^{(t)}, a)}{n^{\text{prior}} + n(\boldsymbol{s}^{(t)}, a))}, n^{\text{post}}(\boldsymbol{s}^{(t)}, a)) = n^{\text{prior}} + n(\boldsymbol{s}^{(t)}, a)$ where the prior parameters are chosen to enforce high entropy for the prior distribution e.g. $\boldsymbol{\chi}^{\text{prior}} = (0, 100)^T, n^{\text{prior}} = 1$ [69]. In this case, the epistemic distribution is a Normal Inverse-Gamma distribution and the aleatoric distribution is a Normal distribution [69]. We train the neural network parameters $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ and the normalizing flow parameters $\boldsymbol{\omega}$ jointly with the MSE loss. The entropy of the conjugate prior distribution represents the epistemic uncertainty, i.e. $u_{\text{epist}}(s^{(t)}, a^{(t)}) = \mathbb{H}(\mathcal{N}\Gamma^{-1}(\boldsymbol{\chi}(\boldsymbol{s}^{(t)}, a^{(t)}), n(\boldsymbol{s}^{(t)}, a^{(t)})))$. The entropy of the likelihood distribution represents the aleatoric uncertainty, i.e. $u_{\text{alea}}(s^{(t)}, a^{(t)}) = \mathbb{H}(\mathcal{N}(\mu(\boldsymbol{s}^{(t)}, a^{(t)}), \sigma(\boldsymbol{s}^{(t)}, a^{(t)})))$. Indeed, it has been showed *theoretically* that epistemic uncertainty increases for input states far from input states observed during training (see Appendix G.1) [69]. Thus, the combination of DQN and posterior networks does not suffer from arbitrary uncertainty estimates for extreme input states contrary to ReLU networks [182].

10.5 Evaluation of Uncertainty Quantification in RL

In this section, we provide an extensive evaluation of uncertainty estimation for model-free RL. It compares four uncertainty estimation methods for model-free RL in three environments. First, we evaluate the uncertainty predictions at *training time* to assess the uncertainty concentration (des. 10.3.1) and the sample efficiency of the uncertainty-guided exploration-exploitation strategy (des. 10.3.2). Second, we evaluate the uncertainty estimates at *testing time* to assess the OOD detection performances (des. 10.3.3) and the generalization performances of the uncertainty-guided decision strategy (des. 10.3.4). In particular, we evaluate the trade-off between the generalization performance and the detection performance in new perturbed test environments.

Models. We consider the four uncertainty models MC Dropout **(DropOut)**, **Ensemble**, deep kernel learning **(DKL)** and the evidential model based on Posterior Networks **(PostNet)** combined with the DQN RL policy (see sec. 10.4). In this chapter, we focus on DQN [291] since it is a widely used model-free RL agent that does not provide any uncertainty estimates by default. All models use the same encoder architecture and DQN hyper-parameters. We performed a grid search over all hyper-parameters. We compute the mean and standard error of the mean over 5 seeds. Further details are given in Appendix G.2.

Environments. We used three training environments **CartPole** [25], **Acrobot** [402, 153] and **LunarLander** [52] from the Open AI gym environments [54]. Packer et al. [336] also used similar environments to assess generalization in RL. We focus on these environments since they turn out to be challenging settings for the uncertainty methods and the sampling strategies. Some methods and strategies are indeed already unable to achieve high performance for sample efficiency, generalization, and OOD detection. We provide further details on the environments in Appendix G.3.

OOD environments: The states, actions, and transition dynamics of the OOD environments should not be relevant to the original training environment task, thus being a reasonable failure mode. To this end, the input state is composed of Gaussian noise at every time step independently of the previous actions.

Perturbed environments: These environments are perturbed versions of the original training environment with different perturbation strengths. We separately perturb the *state* space, the *action* space, and the *transition* dynamics with different strengths of Gaussian or uniform noises. These perturbations follow the MDP structure of the environment as proposed by the formal framework for domain shifts presented by Haider et al. [176]. We did not consider perturbation on the initial state only, which would be a weaker version of the state perturbations, and perturbations on the reward function which would not affect the model at testing time. Further details are given in Appendix G.3.

Training Time. First, we compare the sample efficiency and the uncertainty predictions of the four uncertainty methods using the epsilon-greedy exploration-exploitation

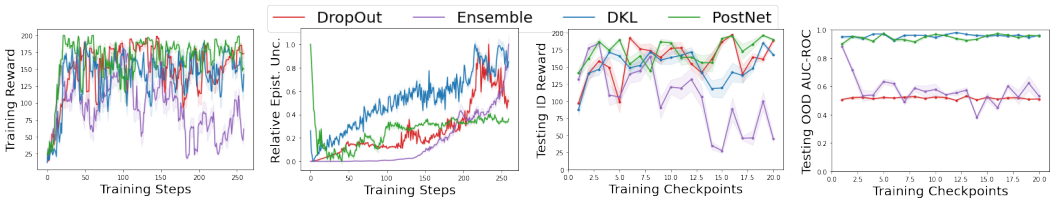


Figure 10.2: Comparison of the training performance (Figs. 10.2a and 10.2b) and testing performance (Figs. 10.2c and 10.2d) of the four uncertainty methods using epsilon-greedy strategies on CartPole. Ideally, an uncertainty aware-model in RL should achieve high reward with few training samples at training and testing time, a decreasing epistemic uncertainty at training time and high OOD detection scores at testing time.

strategy at training time. We normalize the epistemic uncertainty in $[0, 1]$ with min-max normalization to compute the relative epistemic uncertainty. It allows us to easily compare the trend of the epistemic uncertainty of all models. We show the key results for Cartpole in Fig. 10.2 and the detailed results for CartPole, Acrobot, and LunarLander in Fig. G.1 in Appendix G.5. We observe that all methods achieve similar sample efficiency. Ensemble with an epsilon-greedy strategy struggles to maintain high reward on CartPole. This can be explained by the under-exploration of the epsilon-greedy strategy as also observed by Osband et al. [331] and Gal and Ghahramani [141]. Further, we observe that only the epistemic uncertainty estimates of the combination of DQN and PostNet decreases during training. Thus, PostNet *empirically* validates des. 10.3.1. In contrast, the epistemic uncertainty estimates of other methods increase or do not converge. This corroborates with the findings of [331] which *theoretically* shows that the uncertainty estimates of dropout and ensemble might not converge even on simple tasks.

Second, we compare the sample and episode efficiency of the *sampling-epistemic* and the *sampling-aleatoric* strategies for each model during training. We show the results for Acrobot in Fig. 10.3, and additional results for CartPole and LunarLander in Fig. G.2 in Appendix G.5. We observe that all models achieve high rewards by using the sampling-epistemic strategy. In particular, we observed that Ensemble with epistemic sampling achieves more stable rewards than with epsilon-greedy which aligns with observations in Osband et al. [330]. Further, we observe that all models instantiating both aleatoric and epistemic uncertainty achieve significantly better sample efficiency with sampling-epistemic than sampling-aleatoric. Contrary to the sampling-epistemic strategy, the sampling-aleatoric strategy intuitively fails at visiting new under-explored states/actions, thus achieving low and unstable rewards. Hence, Drop-Out, Ensemble and PostNet *empirically* validate des. 10.3.2. Thus, disentangling aleatoric and epistemic uncertainty can speed learning in a training environment. Further, the sampling-epistemic strategy requires fewer finished episodes on CartPole and LunarLander. This represents a more

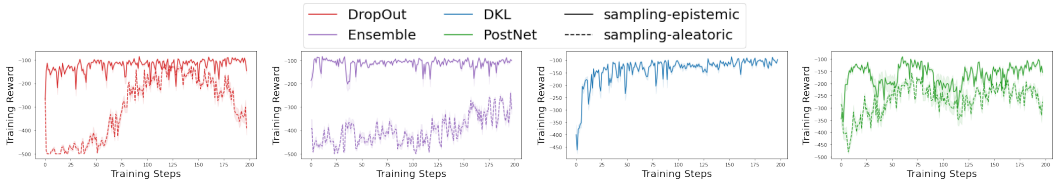


Figure 10.3: Comparison of the training performance on Acrobot. The four uncertainty methods use the sampling-aleatoric or the sampling-epistemic at training time. Ideally, an uncertainty aware-model should high reward with few samples.

reliable training for these two environments since each finished episode translates into a failure and a restart of the systems (see Appendix G.5).

Testing Time. In this section, we save 20 models during training to evaluate their performance at testing time. The testing performance can be viewed as the model performance after deployment. First, we evaluate the testing in-distribution (ID) reward in the training environment and the out-of-distribution (OOD) detection performance against the OOD environment composed of fully noisy states. All the methods used the same epsilon-greedy strategy at training time and the action lead to the highest predicted expected return at testing time. The OOD detection performance is measured by comparing the predicted epistemic uncertainty of the states/actions of 10 episodes with the area under the receiver operating characteristic curve (AUC-ROC). We show the results for CartPole in Fig. 10.2, and additional results for Acrobot and LunarLander in Fig. G.3 in Appendix G.5. We observe that DKL and PostNet achieve very high OOD detection scores compared to DropOut and Ensemble. These *empirical* results align with the *theoretical* results stating that DKL and PostNet should assign high uncertainty to states very different from states observed during training. Thus, DKL and PostNet validate des. 10.3.3. In particular, DKL and PostNet can reliably equip DQN with epistemic uncertainty estimates that can be used to flag anomalous OOD states. In contrast, DropOut and Ensemble achieve poor OOD detection scores. This aligns with Osband et al. [331] and Charpentier et al. [69] who show through multiple experiments that the uncertainty estimates assigned to OOD inputs by DropOut and Ensemble are not significantly smaller than the uncertainty estimates assigned to inputs close to training data.

Second, we compare the testing in-distribution (ID) reward and the out-of-distribution (OOD) detection performance when models use the *sampling-aleatoric* and *sampling-epistemic* strategies at *both* training and testing time. We show the results for the testing reward and the OOD detection scores on the LunarLander in Fig. 10.4, and additional results on the CartPole and the Acrobot environments in Fig. G.4 in Appendix G.5. We observe that the sampling-epistemic strategy achieves significantly better rewards than the sampling-aleatoric for almost any checkpointed models during training. Thus, all models *empirically* satisfy des. 10.3.3. These empirical results underline the need to disentangle both aleatoric and epistemic uncertainty for high reward performance at testing time.

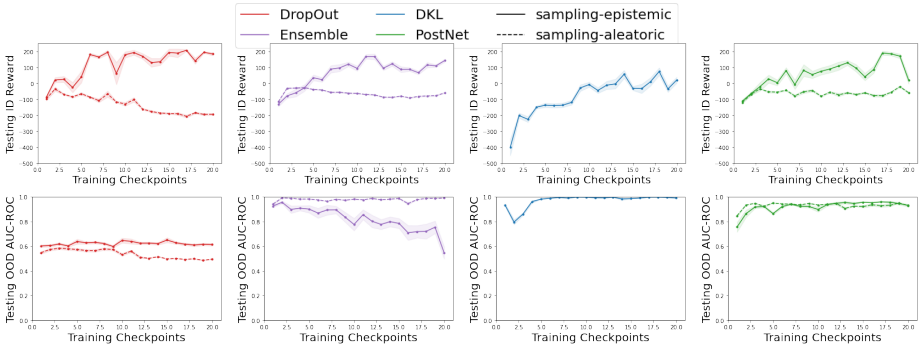


Figure 10.4: Comparison of the testing reward and OOD performance on LunarLander. The four uncertainty methods use the sampling-aleatoric or sampling-epistemic strategies at both training and testing time. Ideally, an uncertainty aware-model should achieve high testing reward and high OOD AUC-ROC detection score.

Third, we compare the *sampling-epistemic* and the *sampling-aleatoric* strategies for each model at testing time. All models use the same epsilon-greedy strategy at training time. We show the key results for the testing reward and the testing epistemic uncertainty on Cartpole with perturbed states, actions, and transition dynamics in Fig. 10.5, and detailed results with other perturbations and environments in Appendix G.5. We observe that stronger state and action perturbations deteriorate the reward performance of all models. This is expected because the input state or the output actions diverge from the training environment with stronger perturbations. Further, while the models were trained using the same epsilon-greedy strategy, we observe that the sampling-epistemic strategy generalizes significantly better to all types of perturbed environments than the sampling-aleatoric strategy. In particular, all models achieve high rewards with epistemic sampling on environments with perturbed transitions. Intuitively, sampling-aleatoric select actions with more inherent risk, while the sampling-epistemic select actions accounting for the knowledge accumulated by the agent in the training environment. The generalization capacity of the sampling-epistemic strategy aligns with that of Ghosh et al. [159] who recasts the problem of generalization in RL as solving an epistemic POMDP. Thus, differentiating between aleatoric and epistemic uncertainty can improve generalization. Finally, we observed that DKL and PostNet consistently assign higher epistemic uncertainty to environments with perturbed states, aligning with their theoretical guarantees on extreme input states. The most challenging perturbations are perturbed actions since none of the models provide guarantees for this perturbation type. Overall, DKL and PostNet reliably assign higher epistemic uncertainty to most of the perturbation types. Therefore, DKL and PostNet perform a good trade-off between generalization and detection of new perturbed environments.

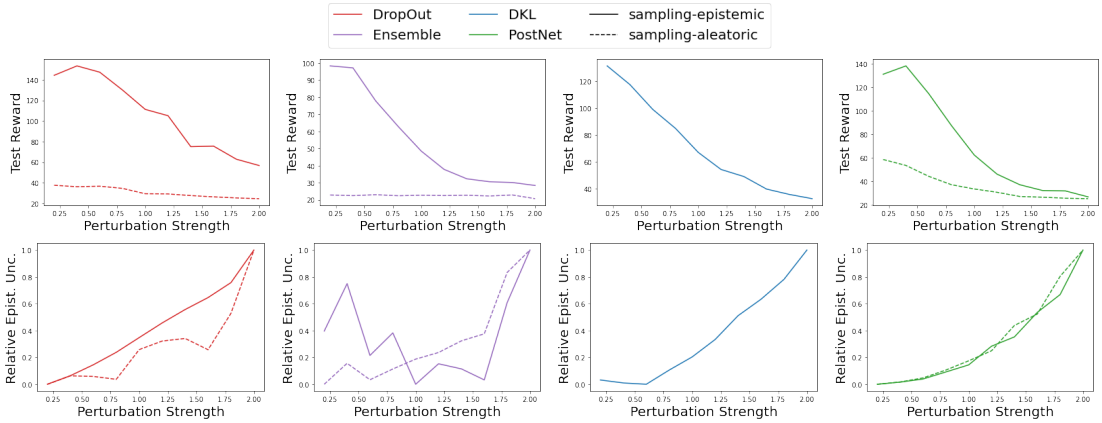


Figure 10.5: Comparison of the testing performance and the epistemic uncertainty predictions on CartPole with perturbed states. The four uncertainty methods use the epsilon-greedy strategy at training time and the sampling-aleatoric or sampling-epistemic strategy at testing time. Ideally, an uncertainty-aware model should maintain high reward while assigning higher epistemic uncertainty on more severe perturbations.

10.6 Related Work

In this section, we cover the related work for aleatoric and epistemic uncertainty estimation for RL. We refer the reader to the survey [3] for an exhaustive overview on uncertainty estimation.

Desiderata: The notion of *risk* is well studied in RL and closely connected to the notion of uncertainty. Risk-sensitive RL usually aims at reducing the number of failures at training time for safer RL [129, 78, 190, 146]. In particular, [85, 130, 123] discuss the trade-off between risk-sensitivity and sample efficiency at training time. Distributional RL methods expect that accounting for aleatoric uncertainty allows to achieve higher returns. Further, [159] aim at improving generalization at testing time within the Bayesian RL framework. In SL, the predicted uncertainty is expected to increase further from training data [286, 69, 237, 398]. None of these previous works give *explicit* desiderata for both *aleatoric* and *epistemic uncertainty* in RL at both *training* and *testing time*.

Models: The related work for uncertainty-aware model in RL is rich as shown in surveys on distributional and Bayesian RL [158, 30]. Distributional RL [30, 301, 29, 93] achieves higher returns by learning the distribution of return which generally captures the aleatoric uncertainty [320]. Bayesian RL methods includes sampling-based methods models such as on dropout [141, 213] and ensembles [330, 331, 266, 416, 159]. These methods are often combined with bootstrapping during training. In particular, [85] proposed to decompose aleatoric and epistemic uncertainty to the cost of multiple trained networks and [103] decompose aleatoric and epistemic uncertainty with latent variables for model-based RL. Bayesian RL also includes Gaussian processes [245, 122] and more specifically the deep kernel learning method [452] which requires storing uncertainty estimates in the experience replay buffer during training. Unlike RL, SL includes many uncertainty methods using deep kernel learning [421, 420] and evidential network [67, 69, 398, 270, 276, 10, 235]. In contrast, we look at *both* sampled-based and sampled-free uncertainty methods for *aleatoric* and *epistemic* uncertainty estimation with *minimal* modification to the training

procedure of the RL agent, thus ensuring easy adaptation of new uncertainty quantification techniques from SL to RL.

Evaluation: [332, 333] proposed to evaluate uncertainty in RL by focusing on joint predictive distributions instead of marginal distributions. Many works [430, 57, 172, 51] used sample efficiency as evaluation method. Further, previous works proposed generalization benchmarks for RL [226, 336, 87]. Finally, [96, 295] have recently proposed benchmarks for OOD detection relevant to RL. In contrast, we propose a simple evaluation method which *jointly* look at *multiple* tasks relevant to real-world applications of uncertainty in RL. It covers epistemic uncertainty tracking and sample efficiency at training time, and generalization and OOD detection at testing time. In particular, we evaluate the trade-off between OOD generalization [385] and OOD detection [453].

10.7 Limitations

Desiderata: Our desiderata, similar to others [398, 20, 304], are designed to be application and model agnostic. In practice, the desiderata should be instantiated with formal definitions and could be customized depending on the application. *Models:* To validate the key contributions, similar to [29, 93, 330], we restrict our experiments to DQNs. However, the four uncertainty methods essentially modify the encoder architecture, it is possible to adapt them to other model-free RL methods such as PPO [372] and A2C [293]. *Evaluation:* Our approach focuses on a simple and task-diverse evaluation methodology for uncertainty estimation. Contrary to Cobbe et al. [87], we do not focus on scaling RL methods to more complex tasks in this chapter.

Conclusion

We introduce a new framework to characterize aleatoric and epistemic uncertainty estimation in RL with four explicit desiderata, four RL models inspired from SL and a practical evaluation methodology. The desiderata characterize the behavior of uncertainty estimates at both training and testing time. The models combine DQN with sampling-based and sampling-free uncertainty methods in SL. We give theoretical and empirical evidence that these methods can fulfil the uncertainty desiderata. The evaluation method assesses the quality of uncertainty estimates on sample efficiency, generalization and OOD detection tasks.

11 Retrospective

In this section, we provide a retrospective on the Chapters 8 to 10 since their publications by discussing potential improvements and the related works published a posteriori.

11.1 Uncertainty for graph data.

Potential improvements. The proposed method GPN (see Chapter 8) has two main directions of improvements. First, GPN focuses on homophilic graphs. Recent works have proposed methods [43, 104] working on both homophilic and heterophilic graphs but do not provide uncertainty estimates. In particular, we proposed in [365] a simple method to improve leanring on heterophilic graphs by using edge directionality. Second, it would be interesting to extend our proposed benchmark for uncertainty estimation on more datasets including very large scale datasets. Recently, [173] has proposed to extend OOD detection benchmarks for graph datasets.

Recent related works. While the field of uncertainty estimation for graph data is still new, multiple recent works already proposed extension for uncertainty estimation at node level, edge level, and graph level. Indeed, [27] proposed to benchmark uncertainty estimation for node classification and finds that GPN and its combination with Natural Posterior Network achieves strong results on various datasets. Further, other recent works [409, 193, 428] had a deeper focused on calibration for GNNs at node level. They observed that GNNs are generally miscalibrated but can be partially recalibrated with calibration methods like temperature rescaling. Other works [474, 192] have extended our approach by proposing uncertainty on edges for calibration and OOD detection. Finally, other approaches [444, 395] focused on uncertainty estimation for graph-level tasks like molecular property prediction. In particular, [26] showed that OOD detection on graph classification is still a widely open research field.

11.2 Uncertainty for sequential data.

Potential improvements. The approaches proposed in Chapter 9 for uncertainty estimation has two main directions of improvement. First, while the uncertainty on the event type is represented via explicit and expressive categorical distributions, the uncertainty on the event time is represented via implicit temporal point processes distributions with contained intensity functions. To solve this issue, [380] and [381] extended our work by modelling expressive point processes which are intensity free and point processes where likelihood computation, sampling, and prediction can all be done efficiently in closed

form. Second, it would be interesting to extend the benchmark for uncertainty estimation for event sequences. Recently, [382] had a closer look at anomalous event detection in both simulated and real-world data and [383] provided an overview of application areas for temporal point processes.

Recent related works. Beyond sequential data with time events, other works have recently looked at uncertainty estimation for sequential data with text to account for the emergence of new powerful large language models. For example, [274] models uncertainty at token and sequence level with applications to translation datasets. [241] recently proposes to estimate uncertainty on semantic meaning in question answering tasks. Further, [180, 197] proposed uncertainty methods for text classification.

11.3 Uncertainty for reinforcement learning.

Potential improvements. The uncertainty framework proposed in Chapter 10 has several directions of improvements. First, similarly to generalization benchmarks for RL [226, 336, 87], it would be interesting to extend this uncertainty benchmark for RL to more complex environments. Second, our uncertainty framework is limited to model-free RL methods and could be extended to model-based RL methods.

Recent related works. Although the uncertainty estimation for RL is not new, this field is still an active domanin of research. Recently, [410] and [449] proposed a new uncertainty methods for model-based RL. Similar to our approach, another work [258] used aleatoric and epistemic uncertainty with density estimation in RL for player evaluation in games. Finally, another work [265] developed other method for distributional RL.

Part IV

Conclusion

12 Conclusion

The more I see the less I know for sure.

John Lennon

I know one thing, that I know nothing.

Socrates

12.1 Uncertainty estimation for independent and non-independent data

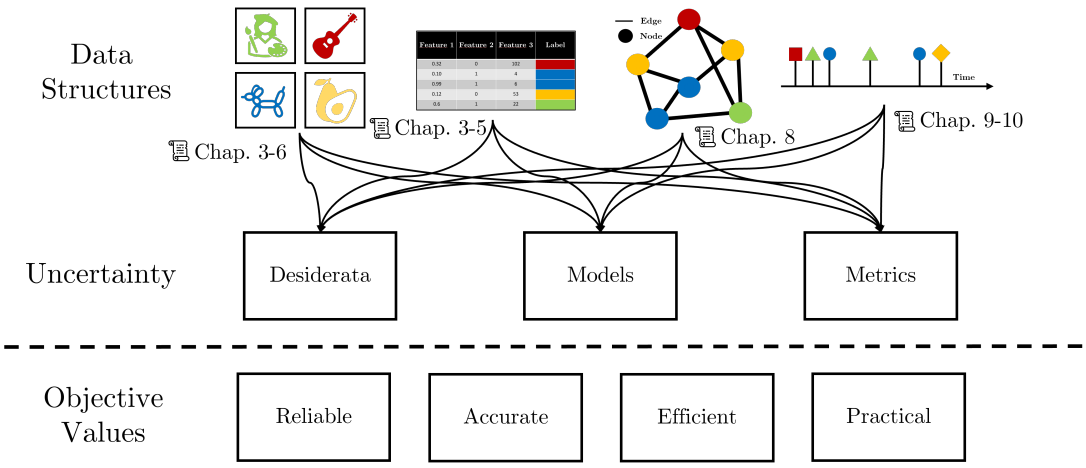


Figure 12.1: Overview of the contributions of this thesis.

In this thesis, we cover a wide range of data structures. While some of them satisfy the independence assumption (e.g. tabular data, image data), some of them satisfy the non-independence assumption (e.g. graph data, seqential data). Beyond label predictions, this thesis brings uncertainty estimation on these data structures for various task types by defining desiderata, designing models, and completing benchmarks for uncertainty estimation. In particular, we show how to leverage the crucial information described by the data sctructure to provide informative uncertainty estimates with important benefits

for critical objective values like reliability, accuracy, efficiency, and ease of use of machine learning models (see Fig. 12.1).

12.2 Reliable ML beyond uncertainty estimation

Accurate uncertainty estimation aims at improving trust in safety-critical domains subject to automation, in application of ML models in areas requiring fairness, or in a maintenance context where the underlying data distribution might slowly shift over time. In this regard, this thesis significantly improves the applicability of uncertainty estimation across a wide range of input (e.g. tabular, images, graph data, sequential data, etc) and output domains (e.g. classification, regression, count prediction, etc) while maintaining a fast inference time. This could be particularly beneficial in industrial applications with time pressure and potential critical consequences (e.g. finance, medicine, policy decision-making, etc).

Nonetheless, while methods described in this thesis achieve high-quality uncertainty estimation, there is always a risk that they do not fully capture the real-world complexity e.g. for OOD data close to ID data. Furthermore, we raise awareness about two other risks of excessive trust related to the *Dunning-Kruger effect* [240]: human excessive trust in Machine Learning model capacity, and human excessive trust in its own interpretation capacity. Therefore, we encourage practitioners to proactively confront the model design and its uncertainty estimates to desired behaviors in real-world use cases.

Further, beyond uncertainty estimation, multiple other aspects play a critical role in the reliability of ML models including robustness, interpretability, privacy, or AI alignment.

Robustness. While Chapter 6 focuses on the robustness of uncertainty estimates, robustness is a more general field which aims to guarantee that any types of model predictions do not change when imperceptible perturbations are applied to the input data. This covers empirical robustness and certifiable robustness against both natural and adversarial perturbations [417, 79, 89, 479]. Robustness is a very active field of research with many studies on robustness for independent data [387] but also for dependent data like graph data [174, 165] or sequential data [75].

Interpretability. Interpretability is important in ML to explain the reasons for model predictions. It ensures impartial decision-making and indicate if meaningful variables are used to infer the prediction output. It covers different types of explanations like text explanations, visual explanations, local explanations or explanations by example. Interpretability and explainability are also very active fields with many existing studies [18, 304].

Privacy. Privacy consists in preventing an attacker to infer facts about members of a population, about data in the training set, or about the model parameters [101, 91]. Privacy is important to avoid data leakage in both centralized and federated learning. An important defense against privacy attackers is differential privacy [116] which tried

to learn useful information from the whole training dataset without retaining specific information about individual samples.

AI alignment. AI alignment aim to align the intended goals of the AI designers, the specified goals to the AI system, and the emergent goals that the AI system actually advances [50, 139]. This field is still at its infancy. Recent works on AI alignment involve large foundational models (e.g. [56, 362, 408]) which might show misaligned behavior which can be (partially) realigned similarly to InstructGPT [334] or ChatGPT [1].

12.3 Open questions

Beyond the potential improvements mentioned in Chapters 7 and 11, we identified other open questions related to uncertainty estimation in ML which are not directly treated in this thesis.

How to estimate uncertainty for large foundational models? Recently, many large foundational models have been developed (e.g. [56, 362, 408]) with already high impact on real-world applications like art creation, text writing, coding, or scientific research. It is important to develop uncertainty methods adapted to these methods to avoid misusage of their (potentially wrong) predictions. Hence, multiple recent works already started to develop promising methods to efficiently estimate uncertainty for large models with a focus on languages models (e.g. [241, 210]). Nonetheless, the emergence of foundational models for different data types and applications motivates further research on uncertainty methods for large models especially when their defined goals might be misaligned with the intended goals.

How to continually learn in the presence of uncertainty? Intuitively, uncertainty is expected to help to continually learn on a series of tasks (e.g. [117, 216, 127]) or actively select useful data for training (e.g. [207, 143, 227, 228, 407]). Indeed, low uncertainty should indicate already learned inputs, while high uncertainty should indicate unknown data regions. However, the study of uncertainty estimation to achieve state-of-the-arts performance is still an important research direction.

Why is the model uncertain? Uncertainty estimates on the predictions might be sensitive to small input data perturbations (see Chapter 6) which makes unclear whether uncertainty predictions are always based on valid reasons. Hence, it is crucial to have reliable explanations on the causes of the uncertainty predictions. However, apart from [15] which proposed a first method to interpret uncertainty estimates and [280] which aim to construct uncertainty set which contains the correct explanation with high probability, there have been very few works on generating explanations on the causes of uncertainty estimates.

How to estimate uncertainty on inferred causal relationships? Causal inference aims at inferring causal relationships from observed data. However, the observed data might not be sufficient to have a clear conclusion on the causal relationship between two variables, thus making them non-identifiable [342]. Only very few works defined distributions on DAGs to provide uncertainty estimates [70]. Hence, the area of uncertainty estimation for causal inference is still underexplored.

Bibliography

[1] Chatgpt: Optimizing language models for dialogue, 2022.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[3] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Abbas Khosravi, U Rajendra Acharya, Vladimir Makarenkov, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 2021.

[4] Taiga Abe, E. Kelly Buchanan, Geoff Pleiss, Richard Zemel, and John Patrick Cunningham. Deep ensembles work, but are they necessary? In *Advances in Neural Information Processing Systems*, 2022.

[5] Ben Adlam, Jasper Roland Snoek, and Sam Smith. Cold posteriors and aleatoric uncertainty. 2020.

[6] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, 2012.

[7] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. In *Advances in Neural Information Processing Systems*, 2019.

[8] Hirotaka Akita, Kosuke Nakago, Tomoki Komatsu, Yohei Sugawara, Shin-ichi Maeda, Yukino Baba, and Hisashi Kashima. Bayesgrad: Explaining predictions of graph convolutional networks. In *Advances in Neural Information Processing Systems*, 2018.

[9] Ismail Alarab and Simant Prakoonwit. Adversarial attack for uncertainty estimation: Identifying critical regions in neural networks. *Neural Processing Letters*, 2021.

[10] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *NeurIPS*, 2020.

[11] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016.

[12] Janna Anderson and Lee Rainie. Concerns about human agency, evolution and survival, 2018.

[13] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2021.

[14] Antreas Antoniou, Massimiliano Patacchiola, Mateusz Ochal, and Amos Storkey. Defining benchmarks for continual few-shot learning, 2020.

[15] Javier Antoran, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a {clue}: A method for explaining uncertainty estimates. In *International Conference on Learning Representations*, 2021.

[16] Martín Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv*, 2019.

[17] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *ICLR*, 2018.

[18] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, 2019.

[19] Jordan Ash and Ryan P Adams. On warm-starting neural network training. *NeurIPS*, 2020.

[20] Rob Ashmore, Radu Calinescu, and Colin Paterson. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *ACM Comput. Surv.*, 2021.

[21] Morgane Ayle, Bertrand Charpentier, John Rachwan, Daniel Zügner, Simon Geisler, and Stephan Günnemann. On the robustness and anomaly detection of sparse neural networks. In *Sparsity in Neural Networks Workshop, SNN*, 2022.

[22] Juhan Bae, Guodong Zhang, and Roger Grosse. Eigenvalue corrected noisy natural gradient, 2018.

[23] Christoph Bartneck, Christoph Lütge, and Sean Wagner, Alan Welsh. *Privacy Issues of AI.* Springer International Publishing, 2021.

[24] Christoph Bartneck, Christoph Lütge, Alan Wagner, and Sean Welsh. *An Introduction to Ethics in Robotics and AI.* 2021.

[25] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on Systems, Man, & Cybernetics*, 1983.

[26] Gleb Bazhenov, Sergei Ivanov, Maxim Panov, Alexey Zaytsev, and Evgeny Burnaev. Towards ood detection in graph classification from uncertainty estimation perspective, 2022.

[27] Gleb Bazhenov, Denis Kuznedelev, Andrey Malinin, Artem Babenko, and Liudmila Prokhorenkova. Revisiting uncertainty estimation for node classification: New benchmark and insights, 2023. URL `https://openreview.net/forum?id=DB3BH3arU2Y`.

[28] Ron Begleiter, Ran El-Yaniv, and Golan Yona. On prediction using variable order markov models. *J. Artif. Int. Res.*, 2004.

[29] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2017.

[30] Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2022.

[31] Richard Bellman. Dynamic programming. *Science*, 1966.

[32] Viktor Bengs, Eyke Hüllermeier, and Willem Waegeman. Pitfalls of epistemic uncertainty quantification through loss minimisation. In *Advances in Neural Information Processing Systems*, 2022.

[33] Viktor Bengs, Eyke Hüllermeier, and Willem Waegeman. On second-order scoring rules for epistemic uncertainty quantification, 2023.

[34] Michael Betancourt. The fundamental incompatibility of scalable hamiltonian monte carlo and naive data subsampling. In *ICML*, 2015.

[35] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[36] *Marin Biloš, *Bertrand Charpentier, and Stephan Günnemann. Uncertainty on asynchronous time event prediction. *Neural Information Processing Systems, NeurIPS*, 2019.

[37] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[38] P. G. Bissiri, C. C. Holmes, and S. G. Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2016.

[39] Julian Bitterwolf, Alexander Meinke, and Matthias Hein. Provable worst case guarantees for the detection of out-of-distribution data. *Neural Information Processing Systems*, 2020.

[40] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 2017.

[41] Nicholas Bloom. Fluctuations in uncertainty. *Journal of Economic Perspectives*, 2014.

[42] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *ICML*, 2015.

[43] Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Lio, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[44] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

[45] Aleksandar Bojchevski and Stephan Günnemann. Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure. In *AAAI Conference on Artificial Intelligence*, 2018.

[46] Aleksandar Bojchevski and Stephan Günnemann. Certifiable robustness to graph perturbations. *Neural Information Processing Systems*, 2019.

[47] Aleksandar Bojchevski, Johannes Klicpera, and Stephan Günnemann. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. *International Conference on Machine Learning*, 2020.

[48] Thomas Bonald, Nathan de Lara, Quentin Lutz, and Bertrand Charpentier. Scikit-network: Graph analysis in python. *Journal of Machine Learning Research, JMLR*, 2020.

[49] Viacheslav Borovitskiy, Iskander Azangulov, Alexander Terenin, Peter Mostowsky, Marc Deisenroth, and Nicolas Durrande. Matérn Gaussian Processes on Graphs. In *International Conference on Artificial Intelligence and Statistics*, 2021.

[50] Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2014.

[51] Mathew Botvinick, Sam Ritter, Jane Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in Cognitive Sciences*, 2019.

[52] Tye Brady and Stephen Paschall. The challenge of safe lunar landing. In *2010 IEEE Aerospace Conference*, 2010.

[53] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *International Conference on Learning Representations*, 2018.

[54] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[55] L. D. Brown. *Fundamentals of Statistical Exponential Families: With Applications in Statistical Decision Theory.* Institute of Mathematical Statistics, 1986.

[56] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[57] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

[58] Craig Calcaterra and Axel Boldt. Approximating with gaussians. *arXiv preprint arXiv:0805.3795*, 2008.

[59] Tianshi Cao, Chin-Wei Huang, David Yu-Tung Hui, and Joseph Paul Cohen. A benchmark of medical out of distribution detection, 2020.

[60] Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of bayesian neural networks to gradient-based attacks. *Neural Information Processing Systems*, 2020.

[61] Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, Nicola Paoletti, Andrea Patane, and Matthew Wicker. Statistical guarantees for the robustness of bayesian neural networks. *Conference on Artificial Intelligence*, 2019.

[62] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *ACM Workshop on Artificial Intelligence and Security*, 2017.

[63] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *Symposium on Security and Privacy*, 2017.

[64] Nicholas Carlini and David A. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *Computing Research Repository*, 2017.

[65] David M Chan, Roshan Rao, Forrest Huang, and John F Canny. Gpu accelerated t-distributed stochastic neighbor embedding. *JDPC*, 2019.

[66] Bertrand Charpentier and Thomas Bonald. Tree sampling divergence: An information-theoretic metric for hierarchical graph clustering. In *International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

[67] Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. *Neural Information Processing Systems, NeurIPS*, 2020.

[68] *Bertrand Charpentier, *Oliver Borchert, Daniel Zügner, Simon Geisler, and Stephan Günnemann. Natural posterior network: Deep bayesian predictive uncertainty for exponential family distributions. In *International Conference on Learning Representations, ICLR*, 2021.

[69] *Bertrand Charpentier, *Oliver Borchert, Daniel Zügner, Simon Geisler, and Stephan Günnemann. Natural Posterior Network: Deep Bayesian Predictive Uncertainty for Exponential Family Distributions. In *International Conference on Learning Representations*, 2022.

[70] Bertrand Charpentier, Simon Kibler, and Stephan Günnemann. Differentiable dag sampling. In *International Conference on Learning Representations, ICLR*, 2022.

[71] Bertrand Charpentier, Ransalu Senanayake, Mykel Kochenderfer, and Stephan Günnemann. Disentangling epistemic and aleatoric uncertainty in reinforcement learning. In *ICML Workshop on Distribution-Free Uncertainty Quantification, ICML - DFUQ*, 2022.

[72] Bertrand Charpentier, Chenxiang Zhang, and Stephan Günnemann. Training, architecture, and prior for deterministic uncertainty methods. In *ICLR 2023 Workshop on Pitfalls of limited data and computation for Trustworthy ML*, 2023. URL `https://openreview.net/forum?id=iYA80086YH`.

[73] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

[74] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*. ACM, 2016.

[75] Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *AAAI Conference on Artificial Intelligence*, 2020.

[76] Hugh Chipman, Edward George, and Robert Mcculloch. Bayesian ensemble learning. *NeurIPS*, 2007.

[77] Hyunsun Choi, Eric Jang, and Alexander A Alemi. Generative ensembles for robust anomaly detection. *ICLR*, 2019.

[78] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 2018.

[79] Sanghyuk Chun, Seong Joon Oh, Sangdoo Yun, Dongyoon Han, Junsuk Choe, and Youngjoon Yoo. An empirical evaluation on robustness and uncertainty of regularization methods, 2020.

[80] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[81] Youngseog Chung, Ian Char, Han Guo, Jeff Schneider, and Willie Neiswanger. Uncertainty toolbox: an open-source library for assessing, visualizing, and improving uncertainty quantification. *arXiv preprint arXiv:2109.10254*, 2021.

[82] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. *International Conference on Machine Learning*, 2017.

[83] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature, 2018.

[84] William R. Clements, Bastien Van Delft, Benoît-Marie Robaglia, Reda Bahi Slaoui, and Sébastien Toth. Estimating risk and uncertainty in deep reinforcement learning, 2019.

[85] William R Clements, Bastien Van Delft, Benoît-Marie Robaglia, Reda Bahi Slaoui, and Sébastien Toth. Estimating risk and uncertainty in deep reinforcement learning. *arXiv preprint arXiv:1905.09638*, 2019.

[86] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

[87] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020.

[88] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, 2021.

[89] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. *International Conference on Machine Learning*, 2019.

[90] Charles Corbière, Nicolas THOME, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. In *NeurIPS*, 2019.

[91] E. De Cristofaro. A critical overview of privacy in machine learning. *IEEE Security; Privacy*, 2021.

[92] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[93] Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit quantile networks for distributional reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018.

[94] Michele Dallachiesa, Charu Aggarwal, and Themis Palpanas. Node classification in uncertain graphs. In *Proceedings of the 26th international conference on scientific and statistical database management*, 2014.

[95] Andreas C. Damianou and Neil D. Lawrence. Deep gaussian processes, 2012.

[96] Mohamad H Danesh and Alan Fern. Out-of-distribution dynamics detection: Rl-relevant benchmarks and results. *International Conference on Machine Learning workshop on Uncertainty in Deep Learning*, 2021.

[97] Raphaël Dang-Nhu, Gagandeep Singh, Pavol Bielik, and Martin Vechev. Adversarial attacks on probabilistic autoregressive forecasting models. *International Conference on Machine Learning*, 2020.

[98] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. *International Conference on Machine Learning*, 2006.

[99] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux - effortless bayesian deep learning. In *NeurIPS*, 2021.

[100] Erik Daxberger, Eric Nalisnick, James Urquhart Allingham, Javier Antorán, and José Miguel Hernández-Lobato. Bayesian deep learning via subnetwork inference. In *ICML*, 2021.

[101] Emiliano De Cristofaro. An overview of privacy in machine learning, 2020.

[102] Arthur P. Dempster. A generalization of bayesian inference. In *Classic Works of the Dempster-Shafer Theory of Belief Functions*, 1968.

[103] Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, 2018.

[104] Francesco Di Giovanni, James Rowbottom, Benjamin P. Chamberlain, Thomas Markovich, and Michael M. Bronstein. Graph neural networks as gradient flows: understanding graph convolutions via energy, 2022.

[105] Ousmane Amadou Dia, Theofanis Karaletsos, Caner Hazirbas, Cristian Canton Ferrer, Ilknur Kaynar Kabul, and Erik Meijer. Localized uncertainty attacks, 2021.

[106] Persi. Diaconis and Donald Ylvisaker. Conjugate priors for exponential families. *The Annals of Statistics*, 1979.

[107] Thomas G. Dietterich. Machine learning for sequential data: A review. 2002.

[108] James Dix. Existence of the limit at infinity for a function that is integrable on the half line. *Ros-Hulman Undergraduate Mathematics Journal*, 2013.

[109] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*, 2016.

[110] Xuefeng Du, Zhaoning Wang, Mu Cai, and Yixuan Li. Vos: Learning what you don't know by virtual outlier synthesis. *Proceedings of the International Conference on Learning Representations*, 2022.

[111] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[112] Simon Duane, A. D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 1987.

[113] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *NeurIPS*, 2019.

[114] Michael W. Dusenberry, Ghassen Jerfel, Yeming Wen, Yi-An Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. *ICML*, 2020.

[115] David Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge, 2014.

[116] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming*. Springer Berlin Heidelberg, 2006.

[117] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. In *International Conference on Learning Representations*, 2020.

[118] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *NeurIPS*, 2014.

[119] Stefanos Eleftheriadis, Tom Nicholson, Marc Deisenroth, and James Hensman. Identification of gaussian process state space models. In *NIPS*. 2017.

[120] Sven Elflein, Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. On out-of-distribution detection with energy-based models. In *ICML workshop on Uncertainty and Robustness in Deep Learning Workshop, UDL - ICML*, 2021.

[121] Pantelis Elinas, Edwin V Bonilla, and Louis Tiao. Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. *Advances in Neural Information Processing Systems*, 2020.

[122] Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with gaussian processes. In *International Conference on Machine Learning*. Association for Computing Machinery, 2005.

[123] Hannes Eriksson and Christos Dimitrakakis. Epistemic risk-sensitive reinforcement learning. *ArXiv*, 2020.

[124] Dhivya Eswaran, Stephan Günnemann, and Christos Faloutsos. The power of certainty: A dirichlet-multinomial model for belief propagation. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, 2017.

[125] William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 2019.

[126] Hadi Fanaee-T and João Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2014.

[127] S. Farquhar and Y. Gal. Differentially private continual learning. In *Privacy in Machine Learning and Artificial Intelligence workshop, ICML*, 2018.

[128] Sebastian Farquhar, Lewis Smith, and Yarin Gal. Liberty or depth: Deep bayesian neural nets do not need complex weight posterior approximations. *NeurIPS*, 2020.

[129] Yingjie Fei, Zhuoran Yang, Yudong Chen, Zhaoran Wang, and Qiaomin Xie. Risk-sensitive reinforcement learning: Near-optimal risk-sample tradeoff in regret. In *Advances in Neural Information Processing Systems*, 2020.

[130] Yingjie Fei, Zhuoran Yang, Yudong Chen, Zhaoran Wang, and Qiaomin Xie. Risk-sensitive reinforcement learning: Near-optimal risk-sample tradeoff in regret. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2020.

[131] Andrej Karpathy Fei-Fei Li and Justin Johnson. Tiny imagenet, 2017.

[132] Boyuan Feng, Yuke Wang, Zheng Wang, and Yufei Ding. Uncertainty-aware attention graph neural network for defending adversarial attacks. *arXiv preprint arXiv:2009.10235*, 2020.

[133] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[134] Chaz Firestone. Performance vs. competence in human–machine comparisons. *Proceedings of the National Academy of Sciences*, 2020.

[135] Andrew Y. K. Foong, David R. Burt, Yingzhen Li, and Richard E. Turner. On the expressiveness of approximate inference in bayesian neural networks. *NeurIPS*, 2020.

[136] Forbes. 3 ways artificial intelligence is transforming business operations. 2019.

[137] Vincent Fortuin, Adrià Garriga-Alonso, Sebastian W. Ober, Florian Wenzel, Gunnar Rätsch, Richard E. Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian neural network priors revisited. In *ICLR*, 2022.

[138] Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

[139] Iason Gabriel. Artificial intelligence, values, and alignment. *Minds and Machines*, 2020.

[140] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[141] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *ICML*, 2016.

[142] Yarin Gal and Lewis Smith. Sufficient conditions for idealised models to have no adversarial examples: a theoretical and empirical study with bayesian neural networks, 2018.

[143] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017.

[144] Ido Galil and Ran El-Yaniv. Disrupting deep uncertainty estimation without harming accuracy. In *Advances in Neural Information Processing Systems*, 2021.

[145] Nicholas Gao and Stephan Günnemann. Ab-initio potential energy surfaces by pairing GNNs with neural wave functions. In *International Conference on Learning Representations*, 2022.

[146] Javier García, Fern, and o Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 2015.

[147] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[148] Adrià Garriga-Alonso and Vincent Fortuin. Exact langevin dynamics with stochastic gradients. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2021. URL `https://openreview.net/forum?id=Rprd8aVUYkE`.

[149] Jochen Gast and Stefan Roth. Lightweight probabilistic deep networks. *CVPR*, 2018.

[150] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv: 2107.03342*, 2021.

[151] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013.

[152] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2004.

[153] Alborz Geramifard, Christoph Dann, Robert H. Klein, William Dabney, and Jonathan P. How. Rlpy: A value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research*, 2015.

[154] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. *ICML*, 2015.

[155] Daniel Gervais. Is Intellectual Property Law Ready for Artificial Intelligence? *SSRN*, 2022.

[156] Lise Getoor. Link-based classification. In *Advanced methods for knowledge discovery from complex data*. Springer, 2005.

[157] Johannes Getzner, Bertrand Charpentier, and Stephan Günnemann. Accuracy is not the only metric that matters: Estimating the energy consumption of deep learning models. In *ICLR 2023 Workshop on Tackling Climate Change with Machine Learning: Global Perspectives and Local Challenges*, 2023.

[158] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 2015.

[159] Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in RL is difficult: Epistemic POMDPs and implicit partial observability. In *Advances in Neural Information Processing Systems*, 2021.

[160] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, 1998.

[161] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 2007.

[162] Eduardo Dadalto Camara Gomes, Florence Alberge, Pierre Duhamel, and Pablo Piantanida. Igeood: An information geometry approach to out-of-distribution detection. In *ICLR*, 2022.

[163] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.

[164] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[165] Lukas Gosch, Simon Geisler, Daniel Sturm, Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Adversarial training for graph neural networks: Pitfalls, solutions, and new directions. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[166] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019.

[167] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020.

[168] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *ICLR*, 2020.

[169] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *ICLR*, 2020.

[170] Alex Graves. Practical variational inference for neural networks. *NeurIPS*, 2011.

[171] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *International Conference on Learning Representations*, 2015.

[172] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations*, 2017.

[173] Shurui Gui, Xiner Li, Limei Wang, and Shuiwang Ji. GOOD: A graph out-of-distribution benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[174] Stephan Günnemann. Graph neural networks: Adversarial robustness. In *Graph Neural Networks: Foundations, Frontiers, and Applications*, chapter 8. Springer, 2022.

[175] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. *ICML*, 2017.

[176] Tom Haider, Felippe Schmoeller Roza, Dirk Eilers, Karsten Roscher, and Stephan Günnemann. Domain shifts in reinforcement learning: Identifying disturbances in environments. In *Workshop on AISafety at the International Joint Conference on Artificial Intelligence*, 2021.

[177] Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayesian graph neural networks with adaptive connection sampling. In *International Conference on Machine Learning*, 2020.

[178] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2021.

[179] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 1971.

[180] Jianfeng He, Xuchao Zhang, Shuo Lei, Zhiqian Chen, Fanglan Chen, Abdulaziz Alhamadani, Bei Xiao, and Chang-Tien Lu. Towards more accurate uncertainty estimation in text classification. 2020.

[181] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[182] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *CVPR*, 2019.

[183] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR*, 2019.

[184] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[185] Jose Miguel Hernandez-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. *ICML*, 2015.

[186] Leon Hetzel, Simon Boehm, Niki Kilbertus, Stephan Günnemann, Mohammad Lotfollahi, and Fabian J Theis. Predicting cellular responses to novel drug perturbations at a single-cell resolution. In *Advances in Neural Information Processing Systems*, 2022.

[187] Jan Hilgevoord and Jos Uffink. The Uncertainty Principle. In *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2016 edition, 2016.

[188] Lior Hirschfeld, Kyle Swanson, Kevin Yang, Regina Barzilay, and Connor W. Coley. Uncertainty quantification using neural networks for molecular property prediction, 2020.

[189] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[190] Ronald A. Howard and James E. Matheson. Risk-sensitive markov decision processes. *Management Science*, 1972.

[191] Jiri Hron, Alex Matthews, and Zoubin Ghahramani. Variational bayesian dropout: pitfalls and fixes. In *International Conference on Machine Learning*, 2018.

[192] Hans Hao-Hsun Hsu, Yuesong Shen, and Daniel Cremers. A graph is more than its nodes: Towards structured uncertainty-aware learning on graphs, 2022.

[193] Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. What makes graph neural networks miscalibrated?, 2022.

[194] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In *NeurIPS Continual learning Workshop*, 2018.

[195] Jiafeng Hu, Reynold Cheng, Zhipeng Huang, Yixang Fang, and Siqiang Luo. On embedding uncertain graphs. In *Information and Knowledge Management*, 2017.

[196] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[197] Yibo Hu and Latifur Khan. Uncertainty-aware reliable text classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. ACM, 2021.

[198] Bing Huang and O. Anatole von Lilienfeld. Ab initio machine learning in chemical compound space. *Chemical Reviews*, 2021.

[199] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, 2018.

[200] Haiwen Huang, Zhihan Li, Lulu Wang, Sishuo Chen, Bin Dong, and Xinyu Zhou. Feature space singularity for out-of-distribution detection. In *Proceedings of the Workshop on Artificial Intelligence Safety 2021 (SafeAI 2021)*, 2021.

[201] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993*, 2020.

[202] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 2021.

[203] Eyke Hüllermeier. Quantifying aleatoric and epistemic uncertainty in machine learning: Are conditional entropy and mutual information appropriate measures?, 2022.

[204] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[205] Valerie Isham and Mark Westcott. A self-correcting point process. *Stochastic Processes and Their Applications*, 1979.

[206] Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Wilson. What are bayesian neural network posteriors really like? *ICML*, 2021.

[207] Moksh Jain, Emmanuel Bengio, Alex-Hernandez Garcia, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ekbote, Jie Fu, Tianyu Zhang, Micheal Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with gflownets, 2022.

[208] Kalvik Jakkala. Deep gaussian processes: A survey, 2021.

[209] Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 1946.

[210] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022.

[211] Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. Fine-tuning deep neural networks in continuous learning scenarios. In *ACCV Workshops*, 2016.

[212] Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

[213] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance, 2017.

[214] Sanyam Kapoor, Wesley J. Maddox, Pavel Izmailov, and Andrew Gordon Wilson. On uncertainty, tempering, and data augmentation in bayesian classification. *arXiv*, 2022.

[215] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *NeurIPS*, 2017.

[216] Mohammad Emtiyaz Khan and Siddharth Swaroop. Knowledge-adaptation priors. *Advances in Neural Information Processing Systems*, 2021.

[217] Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. *AISTATS*, 2012.

[218] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021.

[219] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[220] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*. 2018.

[221] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*. 2016.

[222] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations*, 2016.

[223] Konstantin Kirchheim, Marco Filax, and Frank Ortmeier. Pytorch-ood: A library for out-of-distribution detection based on pytorch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2022.

[224] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Why normalizing flows fail to detect out-of-distribution data. *NeurIPS*, 2020.

[225] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer retraining is sufficient for robustness to spurious correlations. *arXiv*, 2022.

[226] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. 2021.

[227] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[228] Andreas Kirsch, Sebastian Farquhar, and Yarin Gal. A simple baseline for batch active learning with stochastic acquisition functions. *CoRR*, 2021.

[229] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[230] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *arXiv preprint arXiv:1911.05485*, 2019.

[231] Frank H. Knight. *Risk, Uncertainty and Profit*. Houghton Mifflin Co, 1921.

[232] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *TPAMI*, 2020.

[233] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, 2021.

[234] Anna-Kathrin Kopetzki and Stephan Günnemann. Reachable sets of classifiers and regression models: (non-)robustness analysis and robust training. *Machine Learning Journal*, 2021.

[235] *Anna-Kathrin Kopetzki, *Bertrand Charpentier Daniel Zügner, Sandhya Giri, and Stephan Günnemann. Evaluating robustness of predictive uncertainty estimation: Are dirichlet-based models reliable? In *International Conference on Machine Learning, ICML*, 2020.

[236] Maya Krishnan. Against interpretability: a critical examination of the interpretability problem in machine learning. *Philosophy & Technology*, 2020.

[237] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks, 2020.

[238] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[239] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 2012.

[240] Justin Kruger and David Dunning. Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments. *JPSP*, 2000.

[241] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations*, 2023.

[242] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. *ICML*, 2018.

[243] Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with dirichlet calibration. In *Advances in Neural Information Processing Systems*, 2019.

[244] Aounon Kumar, Alexander Levine, Soheil Feizi, and Tom Goldstein. Certifying confidence via randomized smoothing. *Neural Information Processing Systems*, 2020.

[245] Malte Kuss and Carl Rasmussen. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

[246] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *NeurIPS*, 2017.

[247] Balaji Lakshminarayanan, Dustin Tran, Jeremiah Liu, Shreyas Padhy, Tania Bedrax-Weiss, and Zi Lin. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *NeurIPS*, 2020.

[248] Charline Le Lan and Laurent Dinh. Perfect density models cannot guarantee anomaly detection. *arXiv preprint arXiv:2012.03808*, 2020.

[249] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[250] Yann Lecun, Sumit Chopra, Raia Hadsell, Marc Aurelio Ranzato, and Fu Jie Huang. *A tutorial on energy-based learning*. MIT Press, 2006.

[251] Hae Beom Lee, Hayeon Lee, Donghyun Na, Saehoon Kim, Minseop Park, Eunho Yang, and Sung Ju Hwang. Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks, 2020.

[252] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *ICLR*, 2018.

[253] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Neural Information Processing Systems*, 2018.

[254] Yang Li, Nan Du, and Samy Bengio. Time-dependent representation for neural event sequence prediction. In *ICLR Workshop*, 2018.

[255] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[256] Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[257] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 2017.

[258] Guiliang Liu, Yudong Luo, Oliver Schulte, and Pascal Poupart. Uncertainty-aware reinforcement learning for risk-sensitive player evaluation in sports game. In *Advances in Neural Information Processing Systems*, 2022.

[259] Weitang Liu, Xiaoyun Wang, John D Owens, and Yixuan Li. Energy-based out-of-distribution detection. *NeurIPS*, 2020.

[260] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[261] Zhao-Yang Liu, Shao-Yuan Li, Songcan Chen, Yao Hu, and Sheng-Jun Huang. Uncertainty aware graph gaussian process for semi-supervised learning. In *AAAI Conference on Artificial Intelligence*, 2020.

[262] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *ICCV*, 2015.

[263] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, 2017.

[264] D. Lu and Q. Weng. A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 2007.

[265] Yudong Luo, Guiliang Liu, Haonan Duan, Oliver Schulte, and Pascal Poupart. Distributional reinforcement learning with monotonic splines. In *International Conference on Learning Representations*, 2022.

[266] Björn Lütjens, Michael Everett, and Jonathan P. How. Safe reinforcement learning with model uncertainty estimates. *CoRR*, 2018.

[267] Yi-An Ma, Tianqi Chen, and Emily B. Fox. A complete recipe for stochastic gradient mcmc. In *NeurIPS*, 2015.

[268] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *NeurIPS*, 2019.

[269] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018.

[270] Andrey Malinin and Mark Gales. Predictive Uncertainty Estimation via Prior Networks. In *NIPS*, 2018.

[271] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *Neural Information Processing Systems*, 2018.

[272] Andrey Malinin and Mark Gales. Prior networks for detection of adversarial attacks. *arXiv:1812.02575 [stat.ML]*, 2018.

[273] Andrey Malinin and Mark Gales. Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. *NeurIPS*, 2019.

[274] Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021.

[275] Andrey Malinin, Anton Ragni, Kate Knill, and Mark Gales. Incorporating uncertainty into deep learning for spoken language assessment. In *Annual Meeting of the Association for Computational Linguistics*, 2017.

[276] Andrey Malinin, Sergey Chervontsev, Ivan Provilkov, and Mark Gales. Regression prior networks. *arXiv preprint arXiv:2006.11590*, 2020.

[277] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. *ICLR*, 2020.

[278] Andrey Malinin, Neil Band, Ganshin, Alexander, German Chesnokov, Yarin Gal, Mark J. F. Gales, Alexey Noskov, Andrey Ploskonosov, Liudmila Prokhorenkova, Ivan Provilkov, Vatsal Raina, Vyas Raina, Roginskiy, Denis, Mariya Shmatova, Panos Tigas, and Boris Yangel. Shifts: A dataset of real distributional shift across multiple large-scale tasks. *arXiv preprint arXiv:2107.07455*, 2021.

[279] Charles Marx, Shengjia Zhao, Willie Neiswanger, and Stefano Ermon. Modular conformal calibration. In *ICML*, 2022.

[280] Charlie Marx, Youngsuk Park, Hilaf Hasson, Yuyang (Bernie) Wang, Stefano Ermon, and Jun Huan. But are you sure? an uncertainty-aware perspective on explainable ai. In *NeurIPS 2022 Workshop on Trustworthy and Socially Responsible Machine Learning (TSRML)*, 2022.

[281] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Research and Development in Information Retrieval*, 2015.

[282] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 2000.

[283] Patrick L. McDermott and Christopher K. Wikle. Bayesian Recurrent Neural Network Models for Forecasting and Quantifying Uncertainty in Spatial-Temporal Data. *Entropy 21(2): 184 (2019)*, 2019.

[284] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning, 2019.

[285] Hongyuan Mei and Jason M Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *NIPS*, 2017.

[286] Alexander Meinke and Matthias Hein. Towards neural networks that provably know when they don't know. *ICLR*, 2020.

[287] Alexander Meinke, Julian Bitterwolf, and Matthias Hein. Provably robust detection of out-of-distribution data (almost) for free, 2021.

[288] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021.

[289] Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark W. Schmidt, and Mohammad Emtiyaz Khan. SLANG: fast structured covariance approximations for Bayesian deep learning with natural gradient. In *NeurIPS*, 2018.

[290] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[291] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2013.

[292] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[293] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

[294] Anke Moerland. Artificial intelligence and intellectual property law. *GRUR International*, 2020.

[295] Aaqib Parvez Mohammed and Matias Valdenegro-Toro. Benchmark for out-of-distribution detection in deep reinforcement learning. *NeurIPS Workshop on Bayesian Deep Learning*, 2021.

[296] Sina Mohseni, Mandar Pitale, JBS Yadawa, and Zhangyang Wang. Self-supervised learning for generalizable out-of-distribution detection. *AAAI*, 2020.

[297] Christoph Molnar. Interpretable machine learning. 2020.

[298] K. Monteith, J. L. Carroll, K. Seppi, and T. Martinez. Turning bayesian model averaging into bayesian model combination. *IJCNN*, 2011.

[299] Jooyoung Moon, Jihyo Kim, Younghak Shin, and Sangheum Hwang. Confidence-aware learning for deep neural networks. *ICML*, 2020.

[300] Pablo Morales-Alvarez, Daniel Hernández-Lobato, Rafael Molina, and José Miguel Hernández-Lobato. Activation-level uncertainty in deep neural networks. In *International Conference on Learning Representations*, 2021.

[301] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. Nonparametric return distribution approximation for reinforcement learning. In *International Conference on Machine Learning*, 2010.

[302] Warren R. Morningstar, Cusuh Ham, Andrew G. Gallagher, Balaji Lakshminarayanan, Alexander A. Alemi, and Joshua V. Dillon. Density of states estimation for out-of-distribution detection. *arXiv preprint arXiv:2006.09273*, 2020.

[303] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip H. S. Torr, and Yarin Gal. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv*, 2021.

[304] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 2019.

[305] Zachary Nado, Neil Band, Mark Collier, Josip Djolonga, Michael Dusenberry, Sebastian Farquhar, Angelos Filos, Marton Havasi, Rodolphe Jenatton, Ghassen Jerfel, Jeremiah Liu, Zelda Mariet, Jeremy Nixon, Shreyas Padhy, Jie Ren, Tim Rudner, Yeming Wen, Florian Wenzel, Kevin Murphy, D. Sculley, Balaji Lakshminarayanan, Jasper Snoek, Yarin Gal, and Dustin Tran. Uncertainty Baselines: Benchmarks for uncertainty and robustness in deep learning. *arXiv preprint arXiv:2106.04015*, 2021.

[306] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI Conference on Artificial Intelligence*, 2015.

[307] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *ICLR*, 2019.

[308] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using typicality, 2020.

[309] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U M D EDU. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, 2012.

[310] Jay Nandy, Wynne Hsu, and Mong-Li Lee. Towards maximizing the representation gap between in-domain & out-of-distribution examples. *NeurIPS*, 2020.

[311] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. *ECCV*, 2012.

[312] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *NIPS*, 2016.

[313] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[314] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[315] Yin Cheng Ng, Nicolò Colombo, and Ricardo Silva. Bayesian semi-supervised learning with graph gaussian processes. *arXiv preprint arXiv:1809.04379*, 2018.

[316] Cuong Nguyen, Thanh-Toan Do, and Gustavo Carneiro. Uncertainty in model-agnostic meta-learning using variational inference, 2019.

[317] Thanh Thi Nguyen, Quoc Viet Hung Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen, Thien Huynh-The, Saeid Nahavandi, Thanh Tam Nguyen, Quoc-Viet Pham, and Cuong M. Nguyen. Deep learning for deepfakes creation and detection: A survey. *Computer Vision and Image Understanding*, 2022.

[318] Constantin P. Niculescu and Florin Popovici. A note on the behavior of integrable functions at infinity. *Journal of Mathematical Analysis and Applications*, 2011.

[319] Frank Nielsen and Richard Nock. Entropies and cross-entropies of exponential families. *ICIP*, 2010.

[320] Nikolay Nikolov, Johannes Kirschner, Felix Berkenkamp, and Andreas Krause. Information-directed exploration for deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

[321] Julia Nitsch, Masha Itkina, Ransalu Senanayake, Juan Nieto, Max Schmidt, Roland Siegwart, Mykel J. Kochenderfer, and Cesar Cadena. Out-of-distribution detection for automotive perception. In *24th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2021.

[322] Cedrique Rovile Njieutcheu Tassi. Bayesian convolutional neural network: Robustly quantify uncertainty for misclassifications detection. In *Pattern Recognition and Artificial Intelligence*. Springer International Publishing, 2020.

[323] Lorenzo Noci, Kevin Roth, Gregor Bachmann, Sebastian Nowozin, and Thomas Hofmann. Disentangling the roles of curation, data-augmentation and the prior in the cold posterior effect. In *NeurIPS*, 2021.

[324] Matan Orbach and Koby Crammer. Graph-based transduction with confidence. In *Machine Learning and Knowledge Discovery in Databases*, 2012.

[325] Toby Ord. The edges of our universe, 2021.

[326] O'Reilly. The state of data quality in 2020. 2020.

[327] Kazuki Osawa, Siddharth Swaroop, Mohammad Emtiyaz E. Khan, Anirudh Jain, Runa Eschenhagen, Richard E. Turner, and Rio Yokota. Practical deep learning with bayesian principles. *Neural Information Processing Systems*, 2019.

[328] Kazuki Osawa, Siddharth Swaroop, Mohammad Emtiyaz E Khan, Anirudh Jain, Runa Eschenhagen, Richard E Turner, and Rio Yokota. Practical deep learning with bayesian principles. In *Advances in Neural Information Processing Systems*. 2019.

[329] Ian Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NeurIPS Workshop on Bayesian Deep Learning*, 2016.

[330] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, 2016.

[331] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *International Conference on Neural Information Processing Systems*, 2018.

[332] Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado Van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020.

[333] Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Botao Hao, Morteza Ibrahimi, Dieterich Lawson, Xiuyuan Lu, Brendan O'Donoghue, and Benjamin Van Roy. The neural testbed: Evaluating predictive distributions, 2021.

[334] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[335] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *NeurIPS*, 2019.

[336] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning, 2018.

[337] Soumyasundar Pal, Florence Regol, and Mark Coates. Bayesian graph convolutional neural networks using node copying. *arXiv preprint arXiv:1911.04965*, 2019.

[338] Soumyasundar Pal, Florence Regol, and Mark Coates. Bayesian graph convolutional neural networks using non-parametric graph learning. *arXiv preprint arXiv:1910.12132*, 2019.

[339] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *NeurIPS*, 2017.

[340] Sangdon Park, Osbert Bastani, Nikolai Matni, and Insup Lee. Pac confidence sets for deep neural networks via calibrated prediction. In *International Conference on Learning Representations*, 2020.

[341] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. 2019.

[342] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009.

[343] Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. *arXiv preprint arXiv:1908.00598*, 2019.

[344] Janis Postels, Hermann Blum, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. Quantifying aleatoric and epistemic uncertainty using density estimation in latent space. *arXiv*, 2020.

[345] Janis Postels, Hermann Blum, Yannick Strümpler, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. The hidden uncertainty in a neural networks activations, 2020.

[346] Janis Postels, Mattia Segù, Tao Sun, Luca Daniel Sieber, Luc Van Gool, Fisher Yu, and Federico Tombari. On the practicality of deterministic epistemic uncertainty. In *ICML*, 2022.

[347] D. M. W. Powers. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2011.

[348] Apostolos F. Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 2023.

[349] Yao Qin, Xuezhi Wang, Alex Beutel, and Ed H. Chi. Improving uncertainty estimates through the relationship with adversarial robustness. *arXiv:2006.16375 [cs.LG]*, 2020.

[350] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[351] John Rachwan, Daniel Zügner, Bertrand Charpentier, Simon Geisler, Morgane Ayle, and Stephan Günnemann. Winning the lottery ahead of time: Efficient early network pruning. In *International Conference on Machine Learning, ICML*, 2022.

[352] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

[353] Amir Rahimi, Amirreza Shaban, Ching-An Cheng, Richard Hartley, and Byron Boots. Intra order-preserving functions for calibration of multi-class neural networks. *Advances in Neural Information Processing Systems*, 2020.

[354] Rajesh Ranganath, Linpeng Tang, Laurent Charlin, and David Blei. Deep exponential families. *AISTATS*, 2015.

[355] Sebastian Raschka. A short chronology of deep learning for tabular data, 2022.

[356] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[357] Grand View Research. Artificial intelligence market report, 2022-2030. 2020.

[358] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *ICML*, 2015.

[359] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. *ICLR*, 2018.

[360] Christian P. Robert. The metropolis-hastings algorithm, 2015.

[361] Otto Rudolf Rocktäschel. *Methoden zur Berechnung der Gammafunktion für Komplexes Argument*. PhD thesis, Dresden University of Technology, 1922.

[362] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[363] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[364] Roger Rosenkrantz. *Prior Probabilities (1968)*. Springer Netherlands, 1989.

[365] Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M. Bronstein. Edge directionality improves learning on heterophilic graphs. In *The Second Learning on Graphs Conference*, 2023.

[366] Daniel Russo and Benjamin Van Roy. An information-theoretic analysis of thompson sampling. *The Journal of Machine Learning Research*, 2016.

[367] Seongok Ryu, Yongchan Kwon, and Woo Youn Kim. Uncertainty quantification of molecular property prediction with bayesian neural networks. *arXiv preprint arXiv:1903.08375*, 2019.

[368] Peter Sadowski and Pierre Baldi. Neural network regression with beta, dirichlet, and dirichlet-multinomial outputs, 2019.

[369] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 2015.

[370] Robert E Schapire. Explaining adaboost. In *Empirical inference*. Springer, 2013.

[371] Jan Schuchardt, Aleksandar Bojchevski, Johannes Klicpera, and Stephan Günnemann. Collective robustness certificates: Exploiting interdependence in graph neural networks. *International Conference on Learning Representations*, 2021.

[372] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[373] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 2008.

[374] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. *Neural Information Processing Systems*, 2018.

[375] Murat Sensoy, Lance Kaplan, Federico Cerutti, and Maryam Saleki. Uncertainty-aware deep classifiers using generative models. *AAAI*, 2020.

[376] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[377] Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. Do bayesian neural networks need to be fully stochastic?, 2022.

[378] John Shawe-Taylor and Robert C. Williamson. A pac analysis of a bayesian estimator. *COLT*, 1997.

[379] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[380] Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes. *International Conference on Learning Representations (ICLR)*, 2020.

[381] Oleksandr Shchur, Nicholas Gao, Marin Biloš, and Stephan Günnemann. Fast and flexible temporal point processes with triangular maps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[382] Oleksandr Shchur, Ali Caner Turkmen, Tim Januschowski, Jan Gasthaus, and Stephan Günnemann. Detecting anomalous event sequences with temporal point processes. In *Advances in Neural Information Processing Systems*, 2021.

[383] Oleksandr Shchur, Ali Caner Turkmen, Tim Januschowski, and Stephan Günnemann. Neural temporal point processes: A review. In *IJCAI 2021*, 2021.

[384] Alexander Shekhovtsov and Boris Flach. Feed-forward propagation in probabilistic neural networks with categorical and max layers. *ICLR*, 2019.

[385] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey, 2021.

[386] Weishi Shi, Xujiang Zhao, Feng Chen, and Qi Yu. Multifaceted uncertainty estimation for label-efficient deep learning. *NeurIPS*, 2020.

[387] Samuel Henrique Silva and Peyman Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey, 2020.

[388] Daniele Silvestro and Tobias Andermann. Prior choice affects ability of bayesian neural networks to identify unknowns, 2020.

[389] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations,*, 2015.

[390] Edwin Simpson, Stephen Roberts, Ioannis Psorakis, and Arfon Smith. Dynamic bayesian combination of multiple imperfect classifiers. *arXiv preprint arXiv:1206.1831*, 2012.

[391] G. Singh, R. Ganvir, M. Püschel, and M. Vechev. Beyond the single neuron convex barrier for neural network certification. *Neural Information Processing Systems*, 2019.

[392] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. *Uncertainty in Artificial Intelligence*, 2018.

[393] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *NeurIPS*, 2005.

[394] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *NIPS*, 2006.

[395] Ava P Soleimany, Alexander Amini, Samuel Goldman, Daniela Rus, Sangeeta Bhatia, and Connor Coley. Evidential deep learning for guided molecular property prediction and discovery. *ACS Central Science*, 2021.

[396] Hao Song, Tom Diethe, Meelis Kull, and Peter Flach. Distribution calibration for regression. *arXiv preprint arXiv:1905.06023*, 2019.

[397] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 2014.

[398] *Maximilian Stadler, *Bertrand Charpentier, Simon Geisler, Daniel Zügner, and Stephan Günnemann. Graph posterior network: Bayesian predictive uncertainty for node classification. In *Advances in Neural Information Processing Systems*, 2021.

[399] Vincent Stimper, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Resampling Base Distributions of Normalizing Flows. In *AISTATS*, 2022.

[400] David Stutz, Matthias Hein, and Bernt Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. *International Conference on Machine Learning*, 2020.

[401] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. *ICML*, 2022.

[402] Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, 1995.

[403] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *International Conference on Learning Representations*, 2014.

[404] Natasa Tagasovska and David Lopez-Paz. Single-model uncertainties for deep learning. *Neural Information Processing Systems*, 2019.

[405] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. In *ICML*, 2021.

[406] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020.

[407] Ganesh Tata, Gautham Krishna Gudur, Gopinath Chennupati, and Mohammad Emtiyaz Khan. Can calibration improve sample prioritization? In *Has it Trained Yet? NeurIPS 2022 Workshop*, 2022.

[408] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science, 2022.

[409] Leonardo Teixeira, Brian Jalaian, and Bruno Ribeiro. Are graph neural networks miscalibrated? *CoRR*, 2019.

[410] Guy Tennenholtz and Shie Mannor. Uncertainty estimation using riemannian model dynamics for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

[411] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

[412] Michalis Titsias and Neil D. Lawrence. Bayesian gaussian process latent variable model. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. PMLR, 2010.

[413] G. Englebienne T.L.M. van Kasteren and Ben Kröse. Activity recognition in pervasive intelligent environments of the atlantis ambient and pervasive intelligence series, atlantis press. chapter 8. Springer, 2010.

[414] Dustin Tran, Jeremiah Liu, Michael W. Dusenberry, Du Phan, Mark Collier, Jie Ren, Kehang Han, Zi Wang, Zelda Mariet, Huiyi Hu, Neil Band, Tim G. J. Rudner, Karan Singhal, Zachary Nado, Joost van Amersfoort, Andreas Kirsch, Rodolphe Jenatton, Nithum Thain, Honglin Yuan, Kelly Buchanan, Kevin Murphy, D. Sculley, Yarin Gal, Zoubin Ghahramani, Jasper Snoek, and Balaji Lakshminarayanan. Plex: Towards reliability using pretrained large model extensions, 2022.

[415] Kevin Tran, Willie Neiswanger, Junwoong Yoon, Qingyang Zhang, Eric Xing, and Zachary W. Ulissi. Methods for comparing uncertainty quantifications for material property predictions, 2020.

[416] Alexander Tschantz, Beren Millidge, Anil K. Seth, and Christopher L. Buckley. Reinforcement learning through active inference, 2020.

[417] Lifu Tu, Garima Lalwani, Spandana Gella, and He He. An empirical study on robustness to spurious correlations using pre-trained language models. *Transactions of the Association for Computational Linguistics*, 2020.

[418] Ryan Turner, Marc Deisenroth, and Carl Rasmussen. State-space inference and learning with gaussian processes. In *AISTATS*, 2010.

[419] Dennis Ulmer. A survey on evidential deep learning for single-pass uncertainty estimation. *CoRR*, 2021.

[420] Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. *ICML*, 2020.

[421] Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409*, 2021.

[422] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*. 2016.

[423] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[424] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning*. PMLR, 2013.

[425] Hao Wang, Xingjian SHI, and Dit-Yan Yeung. Natural-parameter networks: A class of probabilistic neural networks. *NeurIPS*, 2016.

[426] Hongwei Wang and Jure Leskovec. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755*, 2020.

[427] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 2020.

[428] Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. Be confident! towards trustworthy graph neural networks via confidence calibration. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021.

[429] Yezhen Wang, Bo Li, Tong Che, Kaiyang Zhou, Ziwei Liu, and Dongsheng Li. Energy-based open-world uncertainty modeling for confidence calibration. In *ICCV*, 2021.

[430] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *International Conference on Learning Representations*, 2016.

[431] Marcin Waniek, Tomasz P Michalak, Michael J Wooldridge, and Talal Rahwan. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2018.

[432] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

[433] Junfeng Wen, Negar Hassanpour, and Russell Greiner. Weighted gaussian process for estimating treatment effect. 2018.

[434] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *ICLR*, 2020.

[435] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *International Conference on Learning Representations*, 2018.

[436] Jonathan Wenger, Hedvig Kjellström, and Rudolph Triebel. Non-parametric calibration for classification, 2019.

[437] Jonathan Wenger, Geoff Pleiss, Marvin Pförtner, Philipp Hennig, and John Patrick Cunningham. Posterior and computational uncertainty in gaussian processes. In *Advances in Neural Information Processing Systems*, 2022.

[438] Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub wikatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning*. JMLR.org, 2020.

[439] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *NeurIPS*, 2020.

[440] Edmund Whittaker and George Watson. *A Course of Modern Analysis*. Cambridge University Press, 1927.

[441] Matthew Wicker, Luca Laurenti, Andrea Patane, and Marta Kwiatkowska. Probabilistic safety for bayesian neural networks. *Uncertainty in Artificial Intelligence*, 2020.

[442] Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *NeurIPS*, 2016.

[443] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R. Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, Taylan Cemgil, S. M. Ali Eslami, and Olaf Ronneberger. Contrastive training for improved out-of-distribution detection. *arXiv preprint arXiv:2007.05566*, 2020.

[444] Tom Wollschläger, Nicholas Gao, Bertrand Charpentier, Mohamed Amine Ketata, and Stephan Günnemann. Uncertainty esstimation for molecules: Desiderata and methods. In *International Conference on Machine Learning (ICML)*, 2023.

[445] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1997.

[446] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. *International Conference on Machine Learning*, 2018.

[447] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, 2022.

[448] Mike Wu and Noah D. Goodman. A simple framework for uncertainty in contrastive learning. *arXiv*, 2020.

[449] Zifan Wu, Chao Yu, Chen Chen, Jianye HAO, and Hankz Hankui Zhuo. Plan to predict: Learning an uncertainty-foreseeing model for model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

[450] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[451] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 2020.

[452] Junyu Xuan, Jie Lu, Zheng Yan, and Guangquan Zhang. Bayesian deep reinforcement learning via deep kernel learning. *International Journal of Computational Intelligence Systems*, 2018.

[453] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[454] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, Dan Hendrycks, Yixuan Li, and Ziwei Liu. Openood: Benchmarking generalized out-of-distribution detection. 2022.

[455] Ping yeh Chiang, Michael J. Curry, Ahmed Abdelkader, Aounon Kumar, John Dickerson, and Tom Goldstein. Detection as regression: Certified object detection by median smoothing. *Neural Information Processing Systems*, 2020.

[456] Kyongmin Yeo, Igor Melnyk, Nam Nguyen, and Eun Kyung Lee. Learning temporal evolution of probability distribution with recurrent neural network, 2018.

[457] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[458] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, 2001.

[459] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *KDD*, 2002.

[460] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[461] Arnold Zellner. Optimal information processing and bayes's theorem. *The American Statistician*, 1988.

[462] Arnold Zellner. Optimal information processing and bayes's theorem. *The American Statistician*, 1988.

[463] Huimin Zeng, Zhenrui Yue, Yang Zhang, Ziyi Kou, Lanyu Shang, and Dong Wang. On attacking out-domain uncertainty estimation in deep neural networks, 2022.

[464] Haoxi Zhan and Xiaobing Pei. I-gcn: Robust graph convolutional network via influence mechanism. *arXiv preprint arXiv:2012.06110*, 2020.

[465] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *TPAMI*, 2018.

[466] Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. *ICML*, 2017.

[467] Yao Zhang and Others. Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning. *Chemical science*, 2019.

[468] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *AAAI Conference on Artificial Intelligence*, 2019.

[469] Shengjia Zhao, Tengyu Ma, and Stefano Ermon. Individual calibration with randomized forecasting. *ICML*, 2020.

[470] Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. Uncertainty aware semi-supervised learning on graph data. *Advances in Neural Information Processing Systems*, 2020.

[471] Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. Uncertainty aware semi-supervised learning on graph data. *NeurIPS*, 2020.

[472] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. *Conference on Computer Vision and Pattern Recognition*, 2016.

[473] Yin-Cong Zhi, Yin Cheng Ng, and Xiaowen Dong. Gaussian processes on graphs via spectral kernel learning. *arXiv preprint arXiv:2006.07361*, 2020.

[474] Yangze Zhou, Gitta Kutyniok, and Bruno Ribeiro. Ood link prediction generalization capabilities of message-passing gnns in larger test graphs, 2022.

[475] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[476] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*, 2020.

[477] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations*, 2018.

[478] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2019.

[479] Daniel Zügner and Stephan Günnemann. Certifiable robustness of graph convolutional networks under structure perturbations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining*. Association for Computing Machinery, 2020.

[480] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *SIGKDD*, 2018.

[481] Daniel Zügner, Bertrand Charpentier, Morgane Ayle, Sascha Geringer, and Stephan Günnemann. End-to-end learning of probabilistic hierarchies on graphs. In *International Conference on Learning Representations, ICLR*, 2022.

A Uncertainty Estimation for Classification

A.1 Dirichlet Distribution Computations

A.1.1 Dirichlet distribution

The Dirichlet distribution with concentration parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_C)$, where $\alpha_c > 0$, has the probability density function:

$$f(\boldsymbol{x}; \boldsymbol{\alpha}) = \frac{\prod_{c=1}^{C} \Gamma(\alpha_c)}{\Gamma\left(\sum_{c=1}^{C} \alpha_c\right)} \prod_{c=1}^{C} x_i^{\alpha_c - 1} \tag{A.1}$$

where Γ is a gamma function:

$$\Gamma(\alpha) = \int_0^\infty \alpha^{z-1} e^{-\alpha} dz$$

A.1.2 Closed-form formula for Bayesian loss.

The novel Bayesian loss described in Eq. (3.7) can be computed in closed form. For the sample $\boldsymbol{x}^{(i)}$, it is given by:

$$\mathcal{L}^{(i)} = \underbrace{\mathbb{E}_{q(\mathbf{p}^{(i)})}[\mathrm{CE}(\mathbf{p}^{(i)}, \boldsymbol{y}^{(i)})]}_{(1)} - \underbrace{H(q^{(i)})}_{(2)} \tag{A.2}$$

where the distribution q belongs to the family of the Dirichlet distributions $\mathrm{Dir}(\boldsymbol{\alpha}^{(i)})$. The term (1) is the UCE loss [36]. Given that the observed class one-hot encoded by $\boldsymbol{y}^{(i)}$ is denoted by $c*$, the term (1) is equal to:

$$\mathbb{E}_{q(\mathbf{p}^{(i)})}[\mathrm{CE}(\mathbf{p}^{(i)}, \boldsymbol{y}^{(i)})] = \Psi(\alpha_{c*}^{(i)}) - \Psi(\alpha_0^{(i)}) \tag{A.3}$$

where Ψ denotes the digamma function. The term (2) is the entropy of a Dirichlet distribution and is given by:

$$H(q^{(i)}) = \log B(\boldsymbol{\alpha}^{(i)}) + (\alpha_0^{(i)} - C)\Psi(\alpha_0^{(i)}) - \sum_c (\alpha_c^{(i)} - 1)\Psi(\alpha_c^{(i)}) \tag{A.4}$$

where B denotes the beta function.

A.1.3 Epistemic covariance for in-distribution samples in PostNet.

The epistemic distribution in PostNet is a Dirichlet distribution $\text{Dir}(\boldsymbol{\alpha}^{(i)})$ with the following concentration parameters $\alpha_c^{(i)} = \beta_c^{\text{prior}} + N \cdot \mathbb{P}(c|\mathbf{z}^{(i)};\phi) \cdot \mathbb{P}(\mathbf{z}^{(i)};\phi)$ and $\alpha_0^{(i)} = \sum_c \beta_c^{\text{prior}} + N \cdot \mathbb{P}(\mathbf{z}^{(i)};\phi)$. We can write the variance:

$$\text{Var}_{\boldsymbol{p}\sim\text{Dir}(\boldsymbol{\alpha}^{(i)})}(p_c) = \frac{\alpha_c(\alpha_0 - \alpha_c)}{\alpha_0^2(\alpha_0+1)};\ \text{Cov}_{\boldsymbol{p}\sim\text{Dir}(\boldsymbol{\alpha}^{(i)})}(p_c, p_{c'}) \ = \frac{-\alpha_c\alpha_{c'}}{\alpha_0^2(\alpha_0+1)} \tag{A.5}$$

For in-distribution data (i.e. $\mathbb{P}(\mathbf{z}^{(i)};\phi) \to \infty$), we have

$$\text{Var}_{\boldsymbol{p}\sim\text{Dir}(\boldsymbol{\alpha}^{(i)})}(p_c) = \mathcal{O}\left(\frac{\mathbb{P}(c|\mathbf{z}^{(i)};\phi)(1-\mathbb{P}(c|\mathbf{z}^{(i)};\phi))}{\mathbb{P}(\mathbf{z}^{(i)};\phi)N}\right)$$

$$\text{Cov}_{\boldsymbol{p}\sim\text{Dir}(\boldsymbol{\alpha}^{(i)})}(p_c, p_{c'}) = \mathcal{O}\left(\frac{-\mathbb{P}(c|\mathbf{z}^{(i)};\phi)\mathbb{P}(c'|\mathbf{z}^{(i)};\phi)}{\mathbb{P}(\mathbf{z}^{(i)};\phi)N}\right)$$

. Hence both terms converge to 0 when $\mathbb{P}(\mathbf{z}^{(i)};\phi) \to \infty$.

A.2 Model details

For vector datasets, all models share an architecture of 3 linear layers with Relu activation. A grid search in $[32, 64, 128]$ led to no significant changes in the performances. Therefore, we decided to use 64 hidden units for each layer. For image datasets, we used LeakyRelu activation and add on the top 3 convolutional layers with kernel size of 5, followed by a Max-pooling of size 2. Alternatively, we used the VGG16 architecture with batch normalization [389] adapted from PyTorch implementation [341]. All models are trained after a grid search for learning rate in $[1e^{-3}, 1e^{-5}]$. All models were optimized with Adam optimizer without further learning rate scheduling. We performed early stopping by checking loss improvement every 2 epochs and a patience equal to 10. We trained all models on GPUs (1TB SSD).

For the dropout models, we used $p_{\text{drop}} = .25$ after a grid search in $[.25, .5, .75]$ and sampled 10 times for uncertainty estimation. As an indicator, the original paper [141], uses a dropout probability of .5 for MNIST. It also states that 10 samples already lead to reasonable uncertainty estimates. For the ensemble models, we used $m = 10$ networks after a grid search in $[2, 5, 10, 20]$. A greater number of networks was also found to give no great improvements in the original paper [246]. To be fair with these models, we distilled the knowledge of 10 neural networks for Distribution Distillation. We also trained Prior Networks where target parameters $\beta_{\text{in}} = 1e^2$ as suggested in original papers [270, 273].

For PostNet, we used a 1D batch normalization after the encoder. Experiments on latent dimensions and density types are presented in following sections. If not explicitley mentioned otherwise, we used by default radial flow with a flow length of 6 and a latent dimension of 6. This leads to only 80 parameters. Comparison with IAF are done with two layers of size 256. In general, we found out that a latent dimension smaller or equal

to the number of classes is sufficient. It enables classes to be orthogonal in the latent space if necessary.

All metrics have been scaled by 100. We obtain numbers in $[0, 100]$ for all scores instead of $[0, 1]$.

A.3 Datasets details

For all datasets, we use 5 different random splits to train all models. We split the data in training, validation and test sets (60%, 20%, 20%). In particular, we did not restrict to classic MNIST and CIFAR10 splits in order do prevent overfitting to a specific split.

We use one toy dataset composed of three 2D isotropic Gaussians corresponding to three classes. The Gaussians means are $[0, 2.]$, $[-1.73205081, -1.]$ and $[1.73205081, -1.]$. The variance of the Gaussians is 0.2. A visualization of the true distributions for the three Gaussians is given in Fig. A.1. The final dataset is composed in total of 1500 samples.

We use the segment vector dataset [111], where the goal is to classify areas of images into 7 classes (window, foliage, grass, brickface, path, cement, sky). We remove the class 'sky' from training and instead use it as the OOD dataset for OOD detection experiments. Each input is composed of 18 attributes describing the image area. The dataset contains $2,310$ samples in total.

We further use the Sensorless Drive vector dataset [111], where the goal is to classify extracted motor current measurements into 11 different classes. We remove classes 10 and 11 from training and use them as the OOD dataset for OOD detection experiments. Each input is composed of 49 attributes describing motor behaviour. The dataset contains $58,509$ samples in total.

Additionally, we use the MNIST image dataset [249] where the goal is to classify pictures of hand-drawn digits into 10 classes (from digit 0 to digit 9). Each input is composed of a $1 \times 28 \times 28$ tensor. The dataset contains $70,000$ samples. For OOD detection experiments, we use KMNIST [83] and FashionMNIST [450] containing images of Japanese characters and images of clothes, respectively.

Finally, we use the CIFAR10 image dataset [238] where the goal is to classify a picture of objects into 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). Each input is a $3 \times 32 \times 32$ tensor. The dataset contains $60,000$ samples. For OOD detection experiments, we use street view house numbers (SVHN) [313] containing images of numbers. For the dataset shift experiments, we use the classic split of CIFAR10 to avoid data leakage with the corrupted images from the test set that is provided online.

A.4 Additional Experimental Results

In this section, we present additional results for uncertainty estimation on other datasets. Tables A.1, A.3 to A.5 and A.7 show the performance of all models on the Segment, Sensorless Drive, MNIST, and CIFAR10 datasets. In the same way as for the other datasets, PostNet is competitive for all metrics and show a significant improvement on

calibration among Dirichlet parametrized models and on OOD detection tasks among all models. We evaluated the performances of all models on MNIST using different uncertainty measures and observed very correlated results (see Table A.6). We compared different encoder architectures on CIFAR10 (see Table 3.5). Without further parameter tuning, PostNet adapted well to the convolutional architecture, AlexNet [239], VGG [389] and ResNet [181]. For easier comparison, we also trained models on the classic CIFAR10 split (79%, 5%, 16%) with VGG architecture. We noticed that a larger training set leads to better accuracy for all models.

We also show results of experiments with different latent dimensions (see Figs. A.3, A.4, A.9 and A.10) and density types (MoG, radial, IAF) (see Tables A.1, A.3 to A.5 and A.7) for all datasets. We remarked that PostNet works with various type of densities even if using mixture of Gaussians presented more instability in practice. We observed no clear winner between Radial flow and IAF. We observed a bit lower performances for MoG which could be explained by its lack of expressiveness. Furthermore, we observed that a too high latent dimension would affect the performance.

Beside tables and figures with detailed metrics, we report additional visualizations. We present the uncertainty visualization on the input space for a 2D toy dataset (see Fig. A.2). We show this visualization for all models parametrizing Dirichlet distributions. PostNet is the only model which is not overconfident for OOD data. In particular, it demonstrates the best fit of the true in-distribution data shown in Fig. A.1. Other models show overconfident prediction for OOD regions and fail even on this simple dataset.

Furthermore, we plotted histograms of entropy of ID, OOD, OODom data for MNIST and CIFAR10 (see Figs. 3.4 and A.6). For both datasets, PostNet can easily distinguish between the three data types.

Finally, We also included the evolution of the uncertainty while interpolating linearly between images of MNIST (see Figs. A.7 and A.8). It corresponds to a smooth walk in latent space. As shown in Fig. A.7, PostNet predicts correctly on clean images and outputs more balanced class predictions for mixed images. Additionally Fig. A.8 shows the evolution of the concentration parameters and consequently the epistemic uncertainty. We observe that the epistemic uncertainty (i.e. low α_c) is higher on mixed images which do not correspond to proper digits.

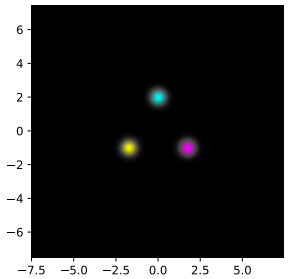


Figure A.1: Three Gaussians toy dataset.

	Acc.	Alea. Conf.	Epist. Conf.	Brier	OOD Alea.	OOD Epist.
Drop Out	95.25±0.1	99.75±0.0	99.43±0.0	11.89±0.2	41.48±0.5	43.11±0.6
Ensemble	*97.27±0.1	*99.88±0.0	*99.85±0.0	*7.64±0.2	54.76±1.6	58.13±1.7
Distill.	96.21±0.1	99.82±0.0	**99.8±0.0**	57.77±0.6	37.12±0.5	35.83±0.4
KL-PN	95.61±0.1	99.79±0.0	99.76±0.0	16.84±0.3	65.62±2.4	57.07±3.7
RKL-PN	96.36±0.2	99.71±0.0	99.58±0.0	11.97±0.1	75.46±2.4	51.02±0.6
PostN Rad. (2)	95.76±0.1	99.23±0.1	98.82±0.1	13.33±1.3	92.75±1.3	90.41±1.5
PostN Rad. (6)	96.52±0.2	99.82±0.0	99.43±0.0	8.69±0.3	98.27±0.2	98.09±0.3
PostN Rad. (10)	94.9±0.2	99.51±0.0	98.57±0.1	12.22±0.7	95.53±0.8	97.51±0.7
PostN IAF (2)	93.94±0.3	99.02±0.1	98.3±0.2	15.33±0.7	***98.3±0.3**	***99.33±0.1**
PostN IAF (6)	95.71±0.2	99.63±0.0	99.11±0.1	10.16±0.3	96.92±0.9	98.17±0.6
PostN IAF (10)	**96.92±0.1**	**99.83±0.0**	99.49±0.0	**8.45±0.4**	95.75±1.1	96.74±0.9
PostN MoG (2)	63.43±5.3	79.61±6.2	79.05±6.1	54.14±5.4	90.87±1.4	91.62±1.4
PostN MoG (6)	89.75±2.5	95.28±1.6	93.15±2.1	24.42±4.3	96.04±1.3	97.71±0.8
PostN MoG (10)	94.44±0.5	99.64±0.1	99.08±0.2	14.79±1.3	91.14±1.5	90.82±1.3

Table A.1: Results on Segment dataset with all models. It shows results with different density types. Number into parentheses indicates flow size (for radial flow and IAF) or number of components (for MoG). Bold numbers indicate best score among Dirichlet parametrized models and starred numbers indicate best scores among all models.

	Acc.	Alea. Conf.	Epist. Conf.	Brier	OOD Alea.	OOD Epist.
PostN: No-Flow	93.13±0.3	99.48±0.1	98.41±0.3	12.94±0.3	47.3±2.9	35.49±0.3
PostN: No-Bayes-Loss	93.94±0.8	98.53±0.3	96.08±1.1	16.15±1.9	94.71±1.0	95.92±0.8
PostN: Seq-No-Bn	18.94±1.1	20.42±1.7	20.42±1.7	91.29±0.0	58.91±0.8	58.43±0.8
PostN: Seq-Bn	93.89±0.1	99.38±0.1	98.93±0.0	14.64±0.3	98.02±0.4	99.93±0.0

Table A.2: Ablation study results on Segment dataset. Gray cells indicate significant drops in scores compare to the complete PostNet Rad. (6) model in Table A.1.

	Acc.	Alea. Conf.	Epist. Conf.	Brier	OOD Alea.	OOD Epist.
Drop Out	89.32±0.2	98.21±0.1	95.24±0.2	28.86±0.4	35.41±0.4	40.61±0.7
Ensemble	99.37±0.0	99.99±0.0	*99.98±0.0	2.47±0.1	50.01±0.0	50.62±0.1
Distill.	93.66±1.5	98.29±0.5	98.15±0.5	44.94±1.4	32.1±0.6	31.17±0.2
KL-PN	94.77±0.9	99.52±0.1	99.47±0.1	21.47±1.9	35.48±0.8	33.2±0.6
RKL-PN	99.42±0.0	99.96±0.0	99.89±0.0	9.07±0.1	45.89±1.6	38.14±0.8
PostN Rad. (2)	96.07±0.0	99.28±0.0	98.88±0.0	19.94±0.0	***98.22±0.0**	***98.03±0.0**
PostN Rad. (6)	98.02±0.1	99.89±0.0	99.47±0.0	5.51±0.2	72.89±0.8	88.73±0.5
PostN Rad. (10)	97.3±0.0	99.82±0.0	99.31±0.0	7.93±0.0	66.65±0.0	87.91±0.0
PostN IAF (2)	99.19±0.0	99.98±0.0	99.78±0.0	2.45±0.0	78.13±0.0	85.9±0.0
PostN IAF (6)	99.11±0.1	99.98±0.0	99.72±0.0	2.71±0.1	78.48±0.7	86.47±0.5
PostN IAF (10)	***99.52±0.0**	***100.0±0.0**	**99.92±0.0**	***1.43±0.1**	82.96±0.8	88.65±0.4
PostN MoG (2)	59.63±4.8	72.2±4.7	70.38±4.8	68.41±4.6	67.2±3.1	72.3±2.9
PostN MoG (6)	96.83±0.2	99.72±0.0	99.16±0.1	13.24±1.0	59.82±2.3	60.61±2.6
PostN MoG (10)	96.65±0.2	99.64±0.0	99.12±0.1	13.12±1.1	61.54±1.8	65.35±2.0

Table A.3: Results on Sensorless Drive dataset with all models. It shows results with different density types. Number into parentheses indicates flow size (for radial flow and IAF) or number of components (for MoG).

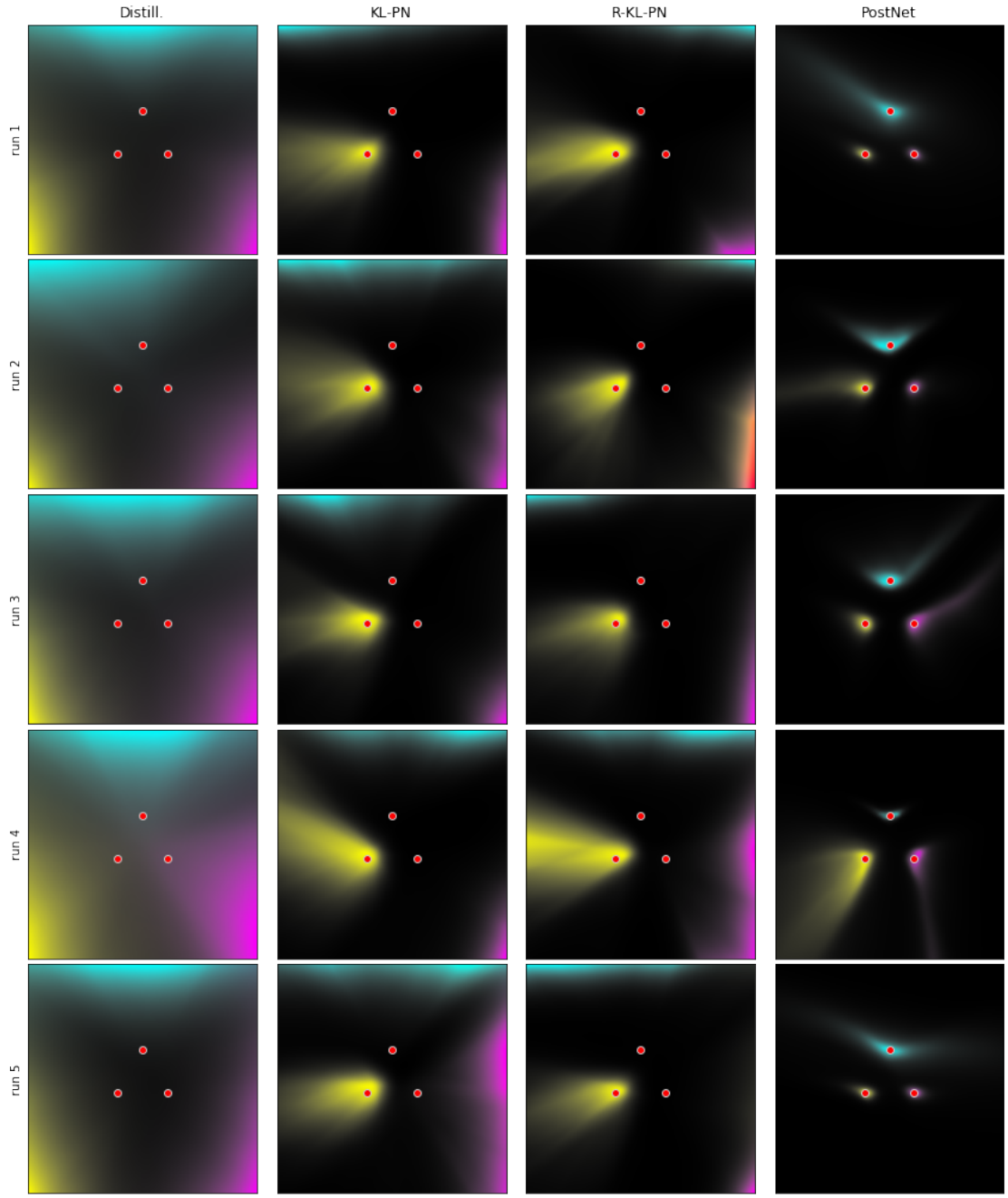


Figure A.2: Visualization of the concentration parameters predicted by Distribution Distillation, Prior Networks trained with KL and reverse KL divergence and Posterior Network on a 3-Gaussians toy dataset over 5 runs. Red dots indicate the mean of the 3 Gaussians. Colours indicate class labels predicted by the models, dark regions correspond to high epistemic uncertainty. PostNet consistently predicts low uncertainty around the training data and high uncertainty for OOD data.

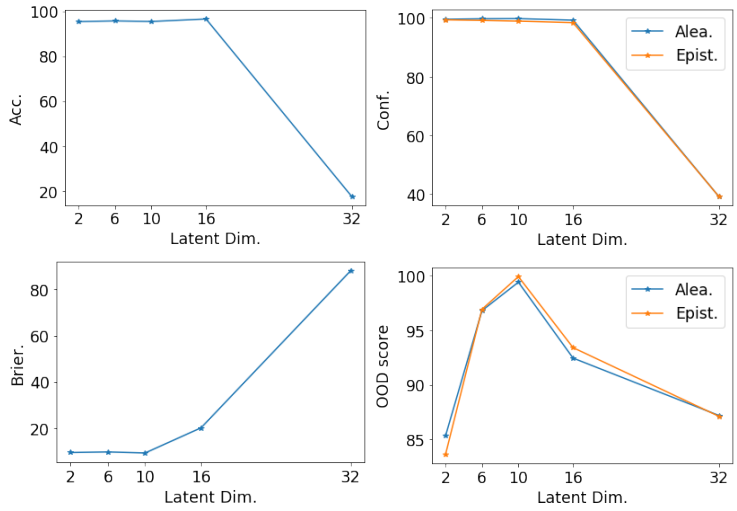


Figure A.3: Accuracy and uncertainty scores of PostNet with latent dimension in $[2, 6, 10, 32]$ on the Segment dataset. We observed that the performances remains high for small dimensions (i.e. 2, 6, 10) and drop for a too high dimension (i.e. 32).

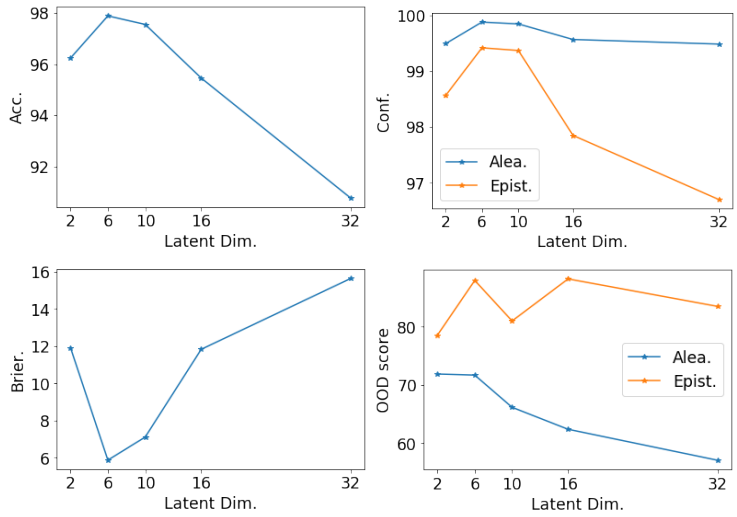
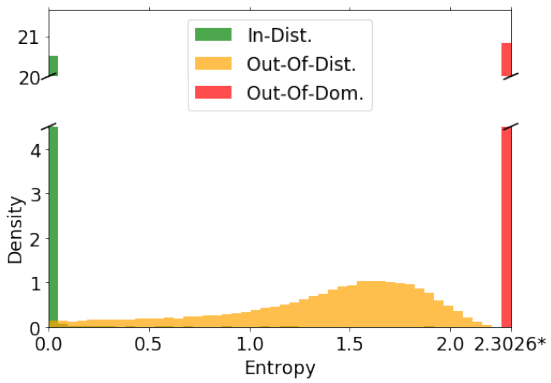


Figure A.4: Accuracy and uncertainty scores of PostNet with latent dimension in $[2, 6, 10, 32]$ on the Sensorless Drive dataset. OOD scores are computed against the left out sky class. We observed that the performances remains high for medium dimensions (i.e. 6, 10) and drop for a too high dimension (i.e. 32).

(a) ID/OOD data (PriorNetworks) **(b)** ID/OOD data (PostNet)

Figure A.5: This figure should be seen in perspective with Fig. 3.1. We plot FashionMNIST OODom data with black crosses to show where these data would land. OODom data were not used for training the models, A comparison of Fig. A.5(a) with Fig. 3.1(b) show that Prior Network assigns high certainty to OODom data. In contrast, a comparison of Fig. A.5(b) and Fig. 3.1(c) shows that Posterior Network assigns low uncertainty to OODom data as desired.



(a) MNIST

Figure A.6: Histograms of the entropy of the predicted categorical distributions for in-distribution (green), out-of-distribution (yellow) and out-of-domain (red) data. The value 2.3026* denotes the maximal entropy achievable for a categorical distribution with 10 classes. We use MNIST, FashionMNIST and the unscaled version of FashionMNIST as in-distribution, out-of-distribution and out-of-domain data. PostNet clearly distinguishes between the three types of data with low entropy for in-distribution data and high entropy for out-of-distribution, and close to the maximum possible entropy for out-of-domain data.

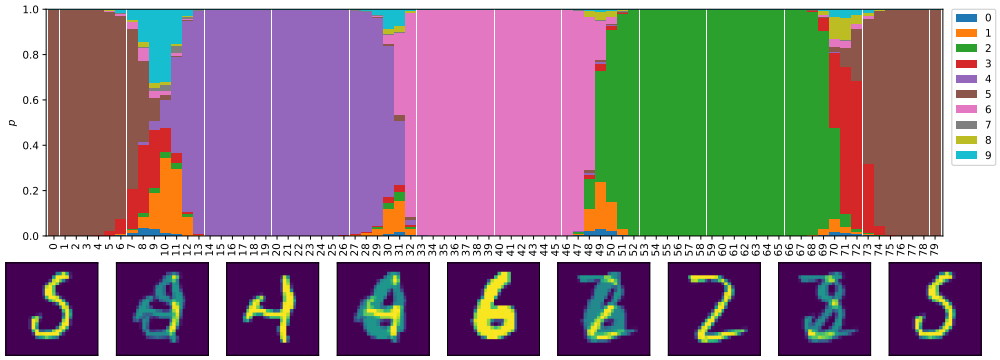


Figure A.7: Evolution of the probability predictions when interpolating linearly between four MNIST images. The interpolation goes from the clean digits 5, 4, 6 and 2 in a cyclic way with 20 interpolated images between each pair. As desired, We can observe correct predictions around clean images with higher (aleatoric) uncertainty for mixed images, and smooth transitions in between.

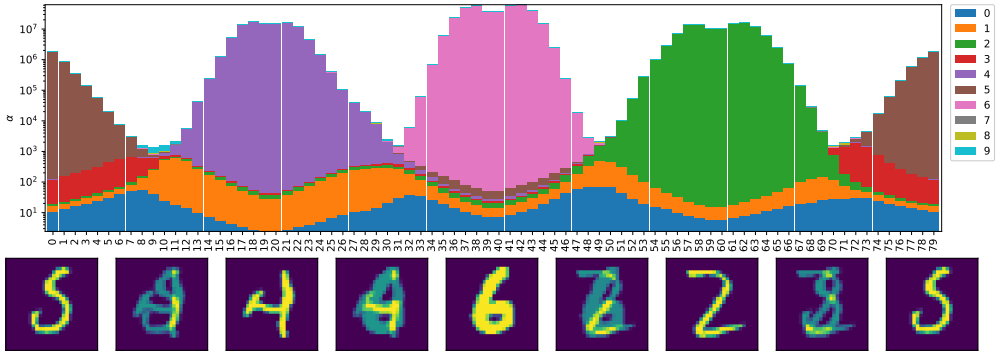


Figure A.8: Evolution of the concentration parameters predictions when interpolating linearly between four MNIST images. The interpolation goes from the clean digits 5, 4, 6 and 2 in a cyclic way with 20 interpolated images between each pair. As desired, we can observe correct and confident predictions around clean images with higher (epistemic) uncertainty for mixed images.

	Acc.	Alea. Conf.	Epist. Conf.	Brier
Drop Out	99.26±0.0	99.98±0.0	99.97±0.0	1.78±0.0
Ensemble	*99.35±0.0	*99.99±0.0	*99.98±0.0	1.67±0.0
Distill.	**99.34±0.0**	**99.98±0.0**	***99.98±0.0**	72.55±0.2
KL-PN	99.01±0.0	99.92±0.0	99.95±0.0	10.82±0.0
RKL-PN	99.21±0.0	99.67±0.0	99.57±0.0	9.76±0.0
RKL-PN w/ F.	99.2±0.0	99.75±0.0	99.68±0.0	9.9±0.0
PostN Rad. (2)	**99.34±0.0**	**99.98±0.0**	99.97±0.0	***1.25±0.0**
PostN Rad. (6)	99.28±0.0	99.97±0.0	99.96±0.0	1.36±0.0
PostN Rad. (10)	99.22±0.0	99.97±0.0	99.97±0.0	1.41±0.0
PostN IAF (2)	99.06±0.0	99.96±0.0	99.94±0.0	1.48±0.0
PostN IAF (6)	99.08±0.0	99.96±0.0	99.94±0.0	1.45±0.1
PostN IAF (10)	98.97±0.0	99.96±0.0	99.94±0.0	1.61±0.0
PostN MoG (2)	76.41±2.3	99.93±0.0	99.92±0.0	23.23±2.2
PostN MoG (6)	99.21±0.0	99.94±0.0	99.92±0.0	1.61±0.0
PostN MoG (10)	99.22±0.0	99.94±0.0	99.92±0.0	1.53±0.0

Table A.4: Accuracy, confidence and calibration results on MNIST dataset with all models. It shows results with different density types. Number into parentheses indicates flow size (for radial flow and IAF) or number of components (for MoG). Bold numbers indicate best score among Dirichlet parametrized models and starred numbers indicate best scores among all models.

	OOD K. Alea.	OOD K. Epist.	OOD F. Alea.	OOD F. Epist.	OODom K. Alea.	OODom K. Epist.	OODom F. Alea.	OODom F. Epist.
Drop Out	94.0±0.1	93.01±0.2	96.56±0.2	95.0±0.2	31.59±0.5	31.97±0.5	27.2±1.1	27.52±1.1
Ensemble	*97.12±0.0	*96.5±0.0	98.15±0.1	96.76±0.0	41.7±0.3	42.25±0.3	37.22±1.0	37.73±1.0
Distill.	**96.64±0.1**	85.17±1.0	98.83±0.0	94.09±0.4	11.49±0.3	10.66±0.2	13.82±0.5	12.03±0.3
KL-PN	92.97±1.2	93.39±1.0	98.44±0.1	98.16±0.0	9.54±0.1	9.78±0.1	9.57±0.1	10.06±0.1
RKL-PN	60.76±2.9	53.76±3.4	78.45±3.1	72.18±3.6	9.35±0.1	8.94±0.0	9.53±0.1	8.96±0.0
RKL-PN w/ F.	81.34±4.5	78.07±4.8	***100.0±0.0**	***100.0±0.0**	9.24±0.1	9.08±0.1	88.96±4.4	87.49±5.0
PostN Rad. (2)	95.49±0.3	93.12±0.7	96.2±0.3	94.6±0.4	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**
PostN Rad. (6)	95.75±0.2	**94.59±0.3**	97.78±0.2	97.24±0.3	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**
PostN Rad. (10)	95.46±0.4	94.19±0.4	97.33±0.2	96.75±0.3	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**
PostN IAF (2)	92.24±0.3	91.75±0.3	96.58±0.2	96.6±0.2	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**
PostN IAF (6)	90.74±0.6	90.63±0.6	93.66±0.5	93.17±0.6	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**
PostN IAF (10)	87.08±0.2	86.52±0.3	92.34±0.6	91.27±0.9	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**
PostN MoG (2)	74.27±2.0	73.34±1.9	76.99±2.0	76.74±1.9	***100.0±0.0**	***100.0±0.0**	99.99±0.0	99.99±0.0
PostN MoG (6)	84.67±1.5	81.46±1.9	88.98±1.7	87.07±2.1	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**
PostN MoG (10)	85.14±1.3	81.12±1.5	94.43±0.8	93.8±1.0	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**

Table A.5: OOD results on MNIST dataset with all models. It shows results with different density types. Number into parentheses indicates flow size (for radial flow and IAF) or number of components (for MoG).

	OOD K. α_0/var.	OOD K. MI.	OOD F. α_0/var.	OOD F. MI.	OODom K. α_0/var.	OODom K. MI.	OODom F. α_0/var.	OODom F. MI
Ensemble	*97.19±0.0	*97.44±0.0	97.53±0.1	97.69±0.1	42.36±0.3	42.38±0.3	37.85±1.1	37.86±1.1
RKL-PN	54.11±3.4	54.9±3.3	72.54±3.6	73.33±3.5	8.94±0.0	8.94±0.0	8.96±0.0	8.96±0.0
RKL-PN w/ F.	78.4±4.8	78.73±4.8	***100.0±0.0**	***100.0±0.0**	9.08±0.1	9.08±0.1	87.49±5.0	87.49±5.0
PostN	**96.04±0.2**	**96.05±0.2**	98.17±0.2	98.17±0.2	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**	***100.0±0.0**

Table A.6: OOD detection on MNIST with other uncertainty measures. Mutual Information [270] and α_0 (Dirichlet) / variance (Ensemble) results are highly correlated.

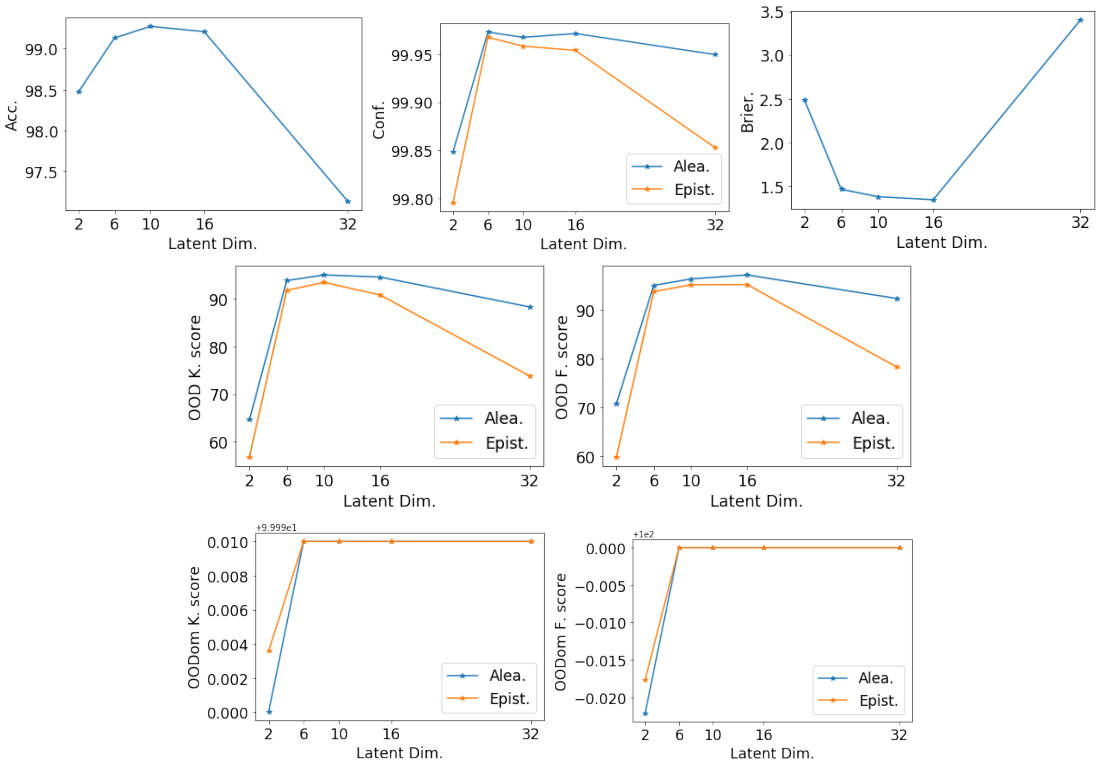


Figure A.9: Accuracy and uncertainty scores of PostNet with latent dimension in $[2, 6, 10, 32]$ on the MNIST dataset. OOD and OODom scores are computed against scaled and unscaled KMNIST and FashionMNIST datasets. We observed that the performances remains high for medium dimensions (i.e. 6, 10) and drop for a too high dimension (i.e. 32).

	Acc.	Alea. Conf.	Epist. Conf.	Brier	OOD Alea.	OOD Epist.	OODom Alea.	OODom Epist.
Drop Out	71.73±0.2	92.18±0.1	84.38±0.3	49.76±0.2	72.94±0.3	41.68±0.5	28.3±1.8	47.1±3.3
Ensemble	*81.24±0.1	*96.61±0.0	93.25±0.1	38.88±0.1	*77.82±0.2	55.17±0.3	63.18±1.1	89.97±0.9
Distill.	72.11±0.4	91.72±0.2	90.73±0.2	88.04±0.1	**75.63±0.6**	52.18±2.1	17.76±0.0	17.76±0.0
KL-PN	48.84±0.5	78.01±0.6	77.99±0.7	83.11±0.6	59.32±1.1	58.03±0.8	17.79±0.0	20.25±0.2
RKL-PN	62.91±0.3	85.62±0.2	81.73±0.2	58.12±0.4	67.07±0.4	56.64±0.8	17.83±0.0	17.76±0.0
PostN Rad. (2)	76.43±0.1	94.59±0.1	94.02±0.1	37.59±0.3	72.91±0.4	69.26±1.1	99.99±0.0	***100.0±0.0**
PostN Rad. (6)	76.46±0.3	94.75±0.1	***94.34±0.1**	***37.39±0.4**	72.83±0.6	***72.82±0.7**	***100.0±0.0**	***100.0±0.0**
PostN Rad. (10)	75.43±0.2	94.16±0.1	93.64±0.1	39.3±0.4	71.94±0.3	70.99±0.5	***100.0±0.0**	***100.0±0.0**
PostN IAF (2)	76.75±0.2	**94.78±0.1**	92.98±0.2	37.87±0.5	73.07±0.5	65.61±1.0	***100.0±0.0**	***100.0±0.0**
PostN IAF (6)	**76.79±0.1**	94.73±0.0	93.7±0.1	37.86±0.2	73.58±0.2	69.74±0.3	***100.0±0.0**	***100.0±0.0**
PostN IAF (10)	75.92±0.2	94.48±0.1	93.23±0.2	39.09±0.3	72.4±0.3	69.04±0.3	***100.0±0.0**	***100.0±0.0**
PostN MoG (2)	44.7±5.9	54.12±7.9	52.12±7.8	68.57±5.2	48.53±3.6	47.45±4.1	99.91±0.0	99.96±0.0
PostN MoG (6)	71.05±1.6	91.21±1.0	86.91±1.2	46.37±2.0	73.49±0.6	56.04±3.8	98.04±0.7	99.62±0.1
PostN MoG (10)	71.63±1.3	91.57±0.8	88.92±0.8	46.07±1.9	72.61±0.3	56.28±1.8	99.88±0.0	***100.0±0.0**

Table A.7: Results on CIFAR10 dataset with all models with convolutional architecture. It shows results with different density types. Number into parentheses indicates flow size (for radial flow and IAF) or number of components (for MoG).

	Acc.	Alea. Conf.	Epist. Conf.	Brier	OOD S. Alea.	OOD S. Epist.	OODom S. Alea.	OODom S. Epist.
Ensemble	*91.34±0.0	*99.1±0.0	98.77±0.0	17.69±0.1	*80.1±0.3	75.14±0.2	21.1±3.1	24.42±3.7
RKL-PN	60.05±0.7	85.63±0.8	82.11±1.3	70.84±0.9	50.97±3.9	55.37±4.3	56.16±1.4	51.33±2.4
RKL-PN w/ C100	88.18±0.1	95.44±0.3	94.15±0.3	79.99±2.0	56.67±2.1	73.37±2.3	57.06±1.7	50.31±1.4
PostNet	**90.05±0.1**	**98.87±0.0**	***98.82±0.0**	***15.44±0.1**	**76.04±0.4**	***75.57±0.4**	***87.65±0.3**	***92.13±0.5**

Table A.8: Results with VGG16 on CIFAR10 on classic split (79%, 5%, 16%). RKL-PN w/ C100 uses CIFAR100 as training OOD.

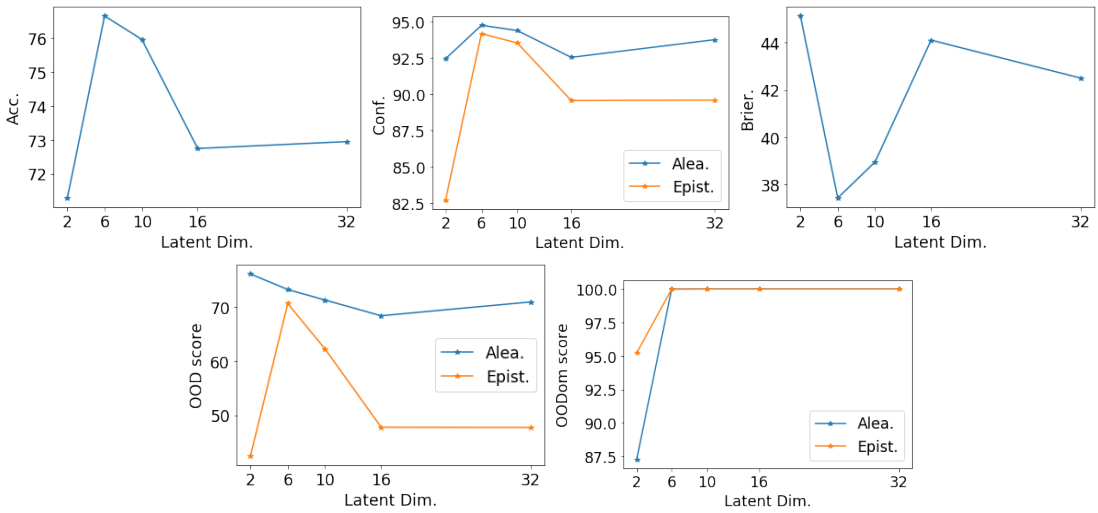


Figure A.10: Accuracy and uncertainty scores of PostNet with latent dimension in $[2, 6, 10, 32]$ on the CIFAR10 dataset. OOD and OODom scores are computed against scaled and unscaled SVHN dataset. We observed that the performances remains high for medium dimensions (i.e. 6, 10) and drop for a too high dimension (i.e. 32).

B Uncertainty Estimation for Regression

B.1 Theorem 1

We prove Theorem 1 based on Lemma 5 and Lemma 6. Lemma 5 states that the input space can be divided in a finite number of linear regions [17]. Lemma 6 states that a probability density with bounded derivatives has to converge to 0 at infinity [108]. We additionally recall Lemma 7 which provide a similar convergence guarantee without the bounded derivative constraint [318]. Finally, Lemma 8 particularly shows that the guarantee of theorem 1 can be obtained with Gaussian Mixtures which are commonly used for density estimation or radial flows which are used in the experiments.

Lemma 5. *[17] Let $\{Q_l\}_l^R$ be the set of linear regions associated to the piecewise ReLU network $f_\phi(\boldsymbol{x})$. For any $\boldsymbol{x} \in \mathbb{R}^D$, there exists $\delta^* \in \mathbb{R}^+$ and $l^* \in 1, ..., R$ such that $\delta \boldsymbol{x} \in Q_{l^*}$ for all $\delta > \delta^*$.*

Lemma 6. *[108] Let $p \in L^1(0, \infty)$ with bounded first derivative p', then $p(\delta) \underset{\delta \to \infty}{\to} 0$. This convergence is stronger than in Lemma 7 as the limit is not in density but with standard limit notation.*

Lemma 7. *[318] Let $p \in L^1(0, \infty)$, then $p(\delta) \underset{\delta \to \infty}{\to} 0$ in density. This means that the sets where $p(t)$ is far from its 0 limit (i.e. $\{t \geq 0 : |p(t)| \geq \epsilon\}$ with $\epsilon > 0$) has zero density.*

Lemma 8. *Let $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$ be parametrized with a Gaussian Mixture Model (GMM) or a radial flow, then $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega}) \underset{||\boldsymbol{z}|| \to \infty}{\to} 0$.*

Proof. We prove now Lemma 8 for GMM and radial flow. The proof is straightforward for the GMM parametrization since every Gaussian component of the mixture has 0 limit when $||\boldsymbol{z}|| \to \infty$.

Let denote now $p_1(z) = \mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$ be parametrized with a radial flow transformation $g(\boldsymbol{z})$ and a base unit Gaussian distribution p_0 i.e.:

$$p_1(\boldsymbol{z}) = p_0(g(\boldsymbol{z})) \times |\det \frac{\partial g(\boldsymbol{z})}{\partial \boldsymbol{z}}|$$

Further, we can express the transformation $g(\boldsymbol{z})$ and its determinant $\det \frac{\partial g(z)}{\partial \boldsymbol{z}}$ as follows:

$$g(\boldsymbol{z}) = \boldsymbol{z} + \beta h(\alpha, r)(\boldsymbol{z} - \boldsymbol{z}_0)$$
$$\det \frac{\partial g(z)}{\partial \boldsymbol{z}} = \frac{1 + \beta h(\alpha, r) + \beta h'(\alpha, r) r}{(1 + \beta h(\alpha, r))^{H-1}}$$

where $h(\alpha, r) = \frac{1}{\alpha+r}$ and $r = ||\boldsymbol{z} - \boldsymbol{z}_0||$. On one hand, we have $||g(\boldsymbol{z})|| \to +\infty$ when $||\boldsymbol{z}|| \to \infty$ since $||\beta h(\alpha, r)(\boldsymbol{z} - \boldsymbol{z}_0)|| < \beta$. Thus, the base Gaussian density $p_0(g(\boldsymbol{z})) \to 0$ when $||\boldsymbol{z}|| \to \infty$. On the other hand, we have $|\det \frac{\partial g(z)}{\partial \boldsymbol{z}}| \to 1$ since $\beta h(\alpha, r) \to 0$ and $\beta h'(\alpha, r)r \to 0$ when $||\boldsymbol{z}|| \to \infty$. Therefore, the transformed density $p_0(g(\boldsymbol{z})) \times |\det \frac{\partial g(\boldsymbol{z})}{\partial \boldsymbol{z}}| \to 0$ when $||\boldsymbol{z}|| \to \infty$ which ends the proof. Note that this proof can be extended to stacked radial flows by induction. □

Theorem. *Let a NatPN model parametrized with a (deep) encoder f_ϕ with piecewise ReLU activations, a decoder g_ψ and the density $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$. Let $f_\phi(\boldsymbol{x}) = V^{(l)}\boldsymbol{x} + a^{(l)}$ be the piecewise affine representation of the ReLU network f_ϕ on the finite number of affine regions $Q^{(l)}$ [17]. Suppose that $V^{(l)}$ have independent rows and the density function $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$ has bounded derivatives, then for almost any $\boldsymbol{x}$ we have $\mathbb{P}(f_\phi(\delta \cdot \boldsymbol{x}) \,|\, \boldsymbol{\omega}) \underset{\delta \to \infty}{\to} 0$. i.e the evidence becomes small far from training data.*

Proof. We prove now Theorem 1. Let $\boldsymbol{x} \in \mathbb{R}^D$ be a non-zero input and f_ϕ be a ReLU network. Lemma 5 implies that there exists $\delta^* \in \mathbb{R}^+$ and $l \in \{1, ..., R\}$ such that $\delta \cdot \boldsymbol{x} \in Q^{(l)}$ for all $\delta > \delta^*$. Thus, $\boldsymbol{z}_\delta = f_\phi(\delta \cdot \boldsymbol{x}) = \delta \cdot (V^{(l)}\boldsymbol{x}) + a^{(l)}$ for all $\delta > \delta^*$. Note that for $\delta \in [\delta^*, +\infty]$, $\boldsymbol{z}_\delta$ follows an affine half line $S_{\boldsymbol{x}} = \{\boldsymbol{z} \,|\, \boldsymbol{z} = \delta \cdot (V^{(l)}\boldsymbol{x}) + a^{(l)}, \delta > \delta^*\}$ in the latent space. Further, note that $V^{(l)}\boldsymbol{x} \neq 0$ and $||\boldsymbol{z}_\delta|| \underset{\delta \to \infty}{\to} +\infty$ since $\boldsymbol{x} \neq 0$ and $V^{(l)}$ has independent rows.

We now define the function $p(\delta) = \mathbb{P}(\boldsymbol{z}_\delta \,|\, \boldsymbol{\omega})$ which is the density function $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$ restricted on the affine half line $S_{\boldsymbol{x}}$. Since $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$ is a normalized probability density, then the function $\delta \mapsto p(\delta - \delta^*)$ is integrable on $[0, +\infty]$. Indeed we have:

$$\begin{aligned}\int_0^{+\infty} p(\delta - \delta^*)d\delta &= \int_{\delta^*}^{+\infty} p(\delta)d\delta \\ &= \int_{\delta^*}^{+\infty} \mathbb{P}(\delta \cdot (V^{(l)}\boldsymbol{x}) + a^{(l)} \,|\, \boldsymbol{\omega})d\delta \\ &= \int^{S_{\boldsymbol{x}}} \mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})d\boldsymbol{z} < +\infty\end{aligned}$$

Further since the function $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$ has bounded derivatives, we can apply Lemma 6 to the function $\delta \mapsto p(\delta - \delta^*)$ to get the expected result i.e.

$$\mathbb{P}(f_\phi(\delta \cdot \boldsymbol{x}) \,|\, \boldsymbol{\omega}) = p(\delta) = p((\delta + \delta^*) - \delta^*) \underset{\delta \to \infty}{\to} 0$$

which ends the proof.

Alternatively a slightly weaker conclusion also holds if the density function does not have bounded derivatives using Lemma 7 (instead of Lemma 6) with the notion of limit in density. The stronger conclusion is valid if we parametrize $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$ with a Gaussian Mixture Model or a radial flow density according to Lemma 8 since $||\boldsymbol{z}_\delta|| \underset{\delta \to \infty}{\to} +\infty$. □

Further, we provide additional comments on the assumption that a trained network converges to linear transformation with exactly two or more dependent rows in Theorem 1. Under this realistic condition [182], the null space is reduced to 0 according

to the rank-nullity theorem meaning that there should be no dead input feature/pixel. If this condition does not hold, this would mean that this specific input feature/pixel is not informative for the prediction task. Thus it could be desired in practice that it does not affect the uncertainty on the prediction. This latter aspect is discussed in the "Task-Specific OOD" paragraph in Section 4.3.6.

B.2 Bayesian Loss

NatPN minimizes the following Bayesian formulation:

$$\mathcal{L}^{(i)} = \underbrace{-\mathbb{E}_{\boldsymbol{\theta}^{(i)}\sim\mathbb{Q}^{\text{post},(i)}}[\log \mathbb{P}(y^{(i)} \mid \boldsymbol{\theta}^{(i)})]}_{\text{(i)}} \underbrace{- \mathbb{H}[\mathbb{Q}^{\text{post},(i)}]}_{\text{(ii)}} \tag{B.1}$$

where $\mathbb{H}[\mathbb{Q}^{\text{post},(i)}]$ denotes the entropy of the predicted posterior distribution $\mathbb{Q}^{\text{post},(i)}$. This loss is generally not equal to the ELBO loss. While the term **(i)** can be viewed as an ELBO loss without KL regularization, the term **(ii)** is not necessarily equal to the prior KL regularization term in the ELBO loss since a proper uniform prior might not exist (e.g. the target y is a real number). Indeed, if the target y is a real number, there exists no uniform prior on $\boldsymbol{\theta}$ and the Bayesian loss and ELBO loss are different i.e. $KL(\mathbb{Q}\,||\,\mathbb{Q}^{\text{prior}}) = \int \mathbb{Q}(\theta)\log(\frac{\mathbb{Q}(\theta)}{\mathbb{Q}^{\text{prior}}(\theta)})d\theta \neq \int \mathbb{Q}(\theta)\log\mathbb{Q}(\theta)d\theta = \mathbb{H}(\mathbb{Q})$. Nonetheless, when a uniform prior $\mathbb{Q}^{\text{unif}}$ exists (e.g. the target y is a class), the loss optimization can be seen as an amortized variational optimization of an ELBO loss [465] i.e. $\mathcal{L}^{(i)} = -\mathbb{E}_{\mathbb{Q}^{\text{post},(i)}}[\log \mathbb{P}(y^{(i)} \mid \boldsymbol{\theta}^{(i)})] + \text{KL}[\mathbb{Q}^{\text{post},(i)} \,\|\, \mathbb{Q}^{\text{unif}}]$ where the predicted distribution $\mathbb{Q}^{\text{post},(i)}$ is the variational distribution — which approximates the true posterior distribution. Indeed, the KL regularization term is equal to the entropy regularization term i.e. $KL(\mathbb{Q}\,||\,\mathbb{Q}^{\text{unif}}) = \int \mathbb{Q}(\theta)\log(\frac{\mathbb{Q}(\theta)}{\mathbb{Q}^{\text{unif}}(\theta)}d\theta) \propto \int \mathbb{Q}(\theta)\log Q(\theta)d\theta) = \mathbb{H}(\mathbb{Q})$. Hence, the loss name "Bayesian loss" [67] is motivated by its difference with the ELBO loss and its Bayesian property at optimum.

B.3 Formulae for Exponential Family Distributions

B.3.1 General Case

Target Distribution. An exponential family distribution on a target variable $y \in \mathbb{R}$ with natural parameters $\boldsymbol{\theta} \in \mathbb{R}^L$ can be denoted as

$$\mathbb{P}(y \mid \boldsymbol{\theta}) = h(y)\exp\left(\boldsymbol{\theta}^T\boldsymbol{u}(y) - A(\boldsymbol{\theta})\right) \tag{B.2}$$

where $h : \mathbb{R} \to \mathbb{R}$ is the carrier measure, $A : \mathbb{R}^L \to \mathbb{R}$ the log-normalizer and $\boldsymbol{u} : \mathbb{R} \to \mathbb{R}^L$ the sufficient statistics.

Conjugate Prior Distribution. An exponential family distribution $\mathbb{P}$ always admits a conjugate prior:

$$\mathbb{Q}(\boldsymbol{\theta} \mid \boldsymbol{\chi}, n) = \eta(\boldsymbol{\chi}, n)\exp\left(n\,\boldsymbol{\theta}^T\boldsymbol{\chi} - nA(\boldsymbol{\theta})\right) \tag{B.3}$$

where $\eta : \mathbb{R}^L \times \mathbb{R} \to \mathbb{R}$ is a normalization factor and A the log-normalizer of the distribution $\mathbb{P}$ as in Eq. (B.2)).

Posterior Predictive Distribution. The posterior predictive distribution is given as $\int \mathbb{P}(y^{(i)}|\boldsymbol{\theta})\, \mathbb{Q}(\boldsymbol{\theta}|\boldsymbol{\chi}^{\text{post},(i)}, n^{\text{post},(i)})d\boldsymbol{\theta}$ where the parameter $\boldsymbol{\theta}$ is marginalized out [140]. This distribution can always be computed in closed form for exponential family distributions:

$$\mathbb{P}(y \,|\, \boldsymbol{\chi}, n) = h(y)\frac{\eta(\boldsymbol{\chi}, n)}{\eta\left(\frac{n\boldsymbol{\chi}+\boldsymbol{u}(y)}{n+1}, n+1\right)} \tag{B.4}$$

where h is the carrier measure defined in Eq. (B.2) and η is the normalization factor defined in Eq. (B.3). In particular, the posterior predictive distributions for Categorical, Normal and Poisson target distributions are Categorical, Student and Negative Binomial distributions, respectively.

Likelihood. The log-likelihood of an exponential family distribution can be written as follows:

$$\log \mathbb{P}(y^{(i)} \,|\, \boldsymbol{\theta}) = \log h(y^{(i)}) + \boldsymbol{\theta}^T \boldsymbol{u}(y^{(i)}) - A(\boldsymbol{\theta}) \tag{B.5}$$

Expected Log-Likelihood. Given the log-likelihood of an exponential family distribution, its expectation under the conjugate prior distribution $\mathbb{Q}(\boldsymbol{\theta}|\boldsymbol{\chi}, n)$ can be written as

$$\mathbb{E}_{\boldsymbol{\theta}\sim\mathbb{Q}(\boldsymbol{\chi},n)}[\log \mathbb{P}(y^{(i)} \,|\, \boldsymbol{\theta})] = \log h(y^{(i)}) + \mathbb{E}_{\boldsymbol{\theta}\sim\mathbb{Q}(\boldsymbol{\chi},n)}[\boldsymbol{\theta}]^T \boldsymbol{u}(y^{(i)}) - \mathbb{E}_{\boldsymbol{\theta}\sim\mathbb{Q}(\boldsymbol{\chi},n)}[A(\boldsymbol{\theta})] \tag{B.6}$$

where $\mathbb{E}_{\mathbb{Q}(\boldsymbol{\theta}|\boldsymbol{\chi},n)}[\boldsymbol{\theta}] = \boldsymbol{\chi}$ [55, 106].

Entropy. The entropy of a random variable $y \sim \mathbb{P}(y|\boldsymbol{\theta})$ for an exponential family distribution $\mathbb{P}$ can be written as follows [319]:

$$\mathbb{H}[\mathbb{P}(y|\boldsymbol{\theta})] = A(\boldsymbol{\theta}) - \boldsymbol{\theta}^T \nabla_{\boldsymbol{\theta}} A(\boldsymbol{\theta}) - \mathbb{E}_{y\sim\mathbb{P}(\boldsymbol{\theta})}[\log h(y)] \tag{B.7}$$

B.3.2 Categorical & Dirichlet Distributions

The Dirichlet distribution $\boldsymbol{p} \sim \text{Dir}(\boldsymbol{\alpha})$ is the conjugate prior of the categorical distributions $\boldsymbol{y} \sim \text{Cat}(\boldsymbol{p})$.

Target Distribution. The density and the entropy of the categorical distribution are:

$$\text{Cat}(y \,|\, \boldsymbol{p}) = \sum_{i=1}^{K} \mathbb{I}[y_i = 1]\, p_i \tag{B.8}$$

$$\mathbb{H}[\text{Cat}(\boldsymbol{p})] = \sum_{c=1}^{C} \log p_c \tag{B.9}$$

Conjugate Prior Distribution. The density and the entropy of the Dirichlet distribution are:

$$\mathrm{Dir}(\boldsymbol{p} \,|\, \boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{c=1}^{C} \alpha_c\right)}{\prod_{c=1}^{K} \Gamma(\alpha_c)} \prod_{c=1}^{C} p_c^{\alpha_c - 1} \tag{B.10}$$

$$\mathbb{H}[\mathrm{Dir}(\boldsymbol{\alpha})] = \log B(\boldsymbol{\alpha}) + (\alpha_0 - C)\psi(\alpha_0) - \sum_c (\alpha_c - 1)\psi(\alpha_c) \tag{B.11}$$

where $\psi(\alpha)$ and $B(\boldsymbol{\alpha})$ denote Digamma and Beta functions, respectively, and $\alpha_0 = \sum_c \alpha_c$.

Expected Log-Likelihood. The expected likelihood of the categorical distribution $\mathrm{Cat}(\boldsymbol{p})$ under the Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$ is

$$\mathbb{E}_{\boldsymbol{p} \sim \mathrm{Dir}(\boldsymbol{\alpha})}[\log \mathrm{Cat}(y \,|\, \boldsymbol{p})] = \psi(\alpha_y) - \psi(\alpha_0) \tag{B.12}$$

where $\psi(\alpha)$ denotes Digamma function.

B.3.3 Normal & Normal-Inverse-Gamma Distributions

The Normal-Inverse-Gamma (NIG) distribution $\mu, \sigma \sim \mathcal{N}\Gamma^{-1}(\mu_0, \lambda, \alpha, \beta)$ is the conjugate prior of the normal distribution $y \sim \mathcal{N}(\mu, \sigma)$. Note that as both parameters λ and α can be viewed as pseudo-counts. However, the natural prior parametrization enforces a single pseudo-count n corresponding to $\lambda = 2\alpha$.

Target Distribution. The density and the entropy of the Normal distribution are:

$$\mathcal{N}(y \,|\, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{B.13}$$

$$\mathbb{H}[\mathcal{N}(\mu, \sigma)] = \frac{1}{2}\log(2\pi\sigma^2) \tag{B.14}$$

Conjugate Prior Distribution. The density and the entropy of the NIG distribution are:

$$\mathcal{N}\Gamma^{-1}(\mu, \sigma \,|\, \mu_0, \lambda, \alpha, \beta) = \frac{\beta^\alpha \sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left(-\frac{2\beta + \lambda(\mu - \mu_0)^2}{2\sigma^2}\right) \tag{B.15}$$

$$\mathbb{H}[\mathcal{N}\Gamma^{-1}(\mu_0, \lambda, \alpha, \beta)] = \frac{1}{2} + \log\left((2\pi)^{\frac{1}{2}}\beta^{\frac{3}{2}}\Gamma(\alpha)\right) - \frac{1}{2}\log\lambda + \alpha - \left(\alpha + \frac{3}{2}\right)\psi(\alpha) \tag{B.16}$$

where $\Gamma(\alpha)$ denotes the Gamma function.

Expected Log-Likelihood. The expected likelihood of the Normal distribution $\mathcal{N}(\mu, \sigma)$ under the NIG distribution $\mathcal{NI\Gamma}^{-1}(\mu_0, \lambda, \alpha, \beta)$ is:

$$\mathbb{E}_{(\mu,\sigma)\sim\mathcal{NI\Gamma}^{-1}(\mu,\lambda,\alpha,\beta)}[\log \mathcal{N}(y \,|\, \mu, \sigma)] \tag{B.17}$$

$$= \mathbb{E}\left[-\frac{(y-\mu)^2}{2\sigma^2} - \log(\sigma\sqrt{2\pi})\right] \tag{B.18}$$

$$= \frac{1}{2}\left(-\mathbb{E}\left[\frac{(y-\mu_0)^2}{2\sigma^2}\right] - \mathbb{E}\left[\log \sigma^2\right] - \log 2\pi\right) \tag{B.19}$$

$$= \frac{1}{2}\left(-y^2\,\mathbb{E}\left[\frac{1}{\sigma^2}\right] + 2y\,\mathbb{E}\left[\frac{\mu}{\sigma^2}\right] - \mathbb{E}\left[\frac{\mu^2}{\sigma^2}\right] + \mathbb{E}\left[\log\frac{1}{\sigma^2}\right] - \log 2\pi\right) \tag{B.20}$$

$$= \frac{1}{2}\left(-\frac{\alpha}{\beta}(y-\mu_0)^2 - \frac{1}{\lambda} + \psi(\alpha) - \log\beta - \log 2\pi\right) \tag{B.21}$$

where $\psi(\alpha)$ denotes the Digamma function. We used here the moments of the NIG distribution $\mathbb{E}\left[\frac{\mu}{\sigma^2}\right] = \frac{\alpha\mu_0}{\beta}$, $\mathbb{E}\left[\frac{1}{\sigma^2}\right] = \frac{\alpha}{\beta}$, $\mathbb{E}\left[\frac{\mu^2}{\sigma^2}\right] = \frac{\alpha\mu_0^2}{\beta} + \frac{1}{\lambda}$, and the moment of the inverse Gamma distribution $\mathbb{E}\left[\log\frac{1}{\sigma^2}\right] = \psi(\alpha) - \log\beta$.

B.3.4 Poisson & Gamma Distributions

The Gamma distribution $\lambda \sim \Gamma(\alpha, \beta)$ is the conjugate prior of the Poisson distributions $y \sim \mathrm{Poi}(\lambda)$.

Target Distribution. The density and the entropy of the Poisson distribution are:

$$\mathrm{Poi}(y \,|\, \lambda) = \frac{\lambda^y \exp(-\lambda)}{y!} \tag{B.22}$$

$$\mathbb{H}[\mathrm{Poi}(\lambda)] = \lambda(1 - \log(\lambda))) + \exp(-\lambda)\sum_{k=0}^{\infty}\frac{\lambda^k \log(k!)}{k!} \tag{B.23}$$

Conjugate Prior Distribution. The density and the entropy of the Gamma distribution are:

$$\Gamma(\lambda \,|\, \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}\exp(-\beta\lambda) \tag{B.24}$$

$$\mathbb{H}[\Gamma(\alpha, \beta)] = \alpha + \log\Gamma(\alpha) - \log\beta + (1-\alpha)\psi(\alpha) \tag{B.25}$$

where $\Gamma(\alpha)$ denotes the Gamma function.

Expected Log-Likelihood. The expected likelihood of the Poisson distribution $\mathrm{Poi}(\lambda)$ under the Gamma distribution $\Gamma(\alpha, \beta)$ is

$$\mathbb{E}_{\lambda\sim\Gamma(\alpha,\beta)}[\log \mathrm{Poi}(y \,|\, \lambda)] = \mathbb{E}[\log\lambda]y - \mathbb{E}[\lambda] - \sum_{k=1}^{y}\log k \tag{B.26}$$

$$= (\psi(\alpha) - \log\beta)y - \frac{\alpha}{\beta} - \sum_{k=1}^{y}\log k \tag{B.27}$$

where $\psi(\alpha)$ denotes Digamma function. We used here the moments the Gamma distributions $\mathbb{E}[\log \lambda] = \psi(\alpha) - \log \beta$ and $\mathbb{E}[\lambda] = \frac{\alpha}{\beta}$. Note that $\sum_{k=1}^{y} \log k$ is constant w.r.t. parameters α, β.

B.4 Approximation of Entropies

The computation of a distribution's entropy often requires subtracting huge numbers from each other. While these numbers tend to be very close together, this introduces numerical challenges. For large parameter values, we therefore approximate the entropy by substituting numerically unstable terms and simplifying the resulting formula. For this procedure, we make use of the following equivalences (taken from Rocktäschel [361] and Whittaker and Watson [440], respectively):

$$\log \Gamma(x) \approx \frac{1}{2} \log 2\pi - x + \left(x - \frac{1}{2}\right) \log x \tag{B.28}$$

$$\psi(x) = \log x - \frac{1}{2x} + \mathcal{O}\left(\frac{1}{x^2}\right) \tag{B.29}$$

We note that Eq. (B.29) especially implies $\psi(x) \approx \log x$ and $x\,\psi(x) \approx x \log x - \frac{1}{2}$ for large x.

B.4.1 Dirichlet Distribution

We consider a Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$ of order K with $\alpha_0 = \sum_{i=1}^{K} \alpha_i$. For $\alpha_0 \geq 10^4$, we use the following approximation:

$$\mathbb{H}\left[\mathrm{Dir}(\boldsymbol{\alpha})\right] \approx \frac{K-1}{2}(1 + \log 2\pi) + \frac{1}{2} \sum_{i=1}^{K} \log \alpha_i - \left(K - \frac{1}{2}\right) \log \sum_{i=1}^{K} \alpha_i \tag{B.30}$$

B.4.2 Normal-Inverse-Gamma Distribution

We consider a Normal-Inverse-Gamma distribution $\mathcal{N}\Gamma^{-1}(\mu, \lambda, \alpha, \beta)$. For $\alpha \geq 10^4$, we use the following approximation:

$$\mathbb{H}\left[\mathcal{N}\Gamma^{-1}(\mu, \lambda, \alpha, \beta)\right] \approx 1 + \log 2\pi - 2 \log \alpha + \frac{3}{2} \log \beta - \frac{1}{2} \log \lambda \tag{B.31}$$

B.4.3 Gamma Distribution

We consider a Gamma distribution $\Gamma(\alpha, \beta)$. For $\alpha \geq 10^4$, we use the following approximation:

$$\mathbb{H}\left[\Gamma(\alpha, \beta)\right] \approx \frac{1}{2} + \frac{1}{2} \log 2\pi + \frac{1}{2} \log \alpha - \log \beta \tag{B.32}$$

B.5 Formulae for Uncertainty Estimates

Aleatoric Uncertainty. The entropy of the target distribution $\mathbb{P}(y|\boldsymbol{\theta})$ was used to estimate the aleatoric uncertainty i.e. $\mathbb{H}[\mathbb{P}(y|\boldsymbol{\theta})]$.

Epistemic Uncertainty. The evidence parameter $n^{\text{post},(i)}$ was used to estimate the epistemic uncertainty. Due to its interpretation as a pseudo-count of observed labels, the posterior evidence parameter is indeed a natural indicator for the epistemic uncertainty.

Predictive Uncertainty. The entropy of the posterior distribution $\mathbb{Q}(\boldsymbol{\theta}|\boldsymbol{\chi}^{\text{post},(i)}, n^{\text{post},(i)})$ was used to estimate the predictive uncertainty.

B.6 Dataset Details

We use a train/validation/test split in all experiments. For datasets with a dedicated test split, we split the rest of the data into training and validation sets of size 80%/20%. For all other datasets, we used 70%/15%/15% for the train/validation/test sets. All inputs are rescaled with zero mean and unit variance. Similarly, we also scale the output target for regression. We provide the datasets at the project page [1].

Sensorless Drive [111] This is a tabular dataset where the goal is to classify extracted motor current measurements into 11 different classes. We remove the last two classes (9 and 10) from training and use them as the OOD dataset for OOD detection experiments. Each input is composed of 48 attributes describing motor behavior. The dataset contains $58,509$ samples in total.

MNIST [249] & Fashion-MNIST [450] These are image dataset where the goal is to classify pictures of hand-drawn digits into 10 classes (from digit 0 to digit 9) or classify pictures of clothers. Each input is composed of a $1 \times 28 \times 28$ tensor. The dataset contains $70,000$ samples. For OOD detection experiments againt MNIST data, we use KMNIST [83] and Fashion-MNIST [450] containing images of Japanese characters and images of clothes, respectively. For OOD detection experiments againt Fasgion-MNIST data, we use KMNIST [83] and MNIST [249] containing images of Japanese characters and images of digits, respectively. It uses the MIT License (MIT).

CIFAR-10 [238] This is an image dataset where the goal is to classify a picture of objects into 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). Each input is a $3 \times 32 \times 32$ tensor. The dataset contains $60,000$ samples. For OOD detection experiments, we use street view house numbers (SVHN) [313] containing images of numbers and CelebA [262] containing images of celebrity faces. We do not use CIFAR100 [238] or TinyImageNet [131] as OOD as they also contain images of vehicles and animals similar to CIFAR10. This rightly questions to what extent are these datasets really OOD for CIFAR10. Furthermore, we generate the corrupted CIFAR-10 dataset [183] with 15 corruption types per image, each with 5 different severities. It uses the MIT License (MIT).

[1] `https://www.daml.in.tum.de/natpn`

Bike Sharing [126] This is a tabular dataset where the goal is to predict the total number of rentals within an hour. Each input is composed of 15 attributes. We removed features related to the year period (i.e. record index, date, season, months) which would make OOD detection trivial, leading to 11 attributes. The dataset contains $17,389$ samples in total. For OOD detection, we removed the attribute season from the input data and only trained on the summer season. The samples related to winter, spring and autumn were used as OOD datasets.

Concrete [111] This is a tabular dataset where the goal is to predict the compressive strength of high-performance concrete. Each input is composed of 8 attributes. The dataset contains $1,030$ samples in total. For OOD detection, we use the Energy and Kin8nm datasets which have the same input size.

Kin8nm [111] This is a tabular dataset where the goal is to predict the forward kinematics of an 8-link robot arm. Each input is composed of 8 attributes. The dataset contains $8,192$ samples in total. For OOD detection, we the Concrete and Energy datasets which have the same input size.

NYU Depth v2 [311] This is an image dataset where the goal is to predict the depth of room images at each pixel position. All inputs are of shape 3x640x480 tensors while we rescale outputs to be 320x240 tensors at both training and test time. This setting is slightly different from Kendall and Gal [215] and Nathan Silberman and Fergus [311]. Indeed, [215] up-scales the model output to 640x480 at training and test time while [311] up-scales the model output to 640x480 at test time only. The dataset contains $50,000$ samples in total available on the DenseDepth GitHub [2]. For OOD detection, we use the KITTI [151] dataset containing images of driving cars and two out of the 20 categories from the LSUN [457] dataset.

B.7 Model Details

We train all models using 5 seeds except for the large NYU dataset where we use a single randomly selected seed. All models are optimized with the Adam optimizer without further learning rate scheduling. We perform early stopping by checking loss improvement every epoch and a patience p selected per dataset (Sensorless Drive: $p = 15$, MNIST: $p = 15$, CIFAR10: $p = 20$, Bike Sharing: $p = 50$, Concrete: $p = 50$, Kin8nm: $p = 30$, NYU Depth v2: $p = 2$). We train all models on a single GPU (NVIDIA GTX 1080 Ti or NVIDIA GTX 2080 Ti, 11 GB memory). All models are trained after a grid search for the learning rate in $[1e^{-2}, 5e^{-4}]$. The backbone architecture is shared across models and selected per dataset to match the task needs (Sensorless Drive: 3 lin. layers with 64 hidden dim, MNIST: 6 conv. layers with 32/32/32/64/64/64 filters + 3 lin. layers with hidden dim 1024/128/64, CIFAR10: 8 conv. layers with 32/32/32/64/64/128/128/128 filters + 3 lin. layers with hidden dim 1024/128/64, Bike Sharing: 3 lin. layers with 16/16/16 hidden dim, Concrete: 2 lin. layers with 16/16 hidden dim, Kin8nm: 2 lin. layers with 16/16 hidden dim, NYU Depth v2: DenseDepth + 4 upsampling layers with convolutions and skip connections). For the NYU Depth v2 dataset, we use a

[2] `https://github.com/ialhashim/DenseDepth`

pretrained DenseNet for initialization of the backbone architecture which was fine-tuned during training. The remaining layers are trained from scratch. All architectures use LeakyReLU activations. For further details, we provide the code at the project page [3].

Baselines. For the dropout models, we use the best drop out rate p_{drop} per dataset after a grid search in $\{0.1, 0.25, 0.4\}$ and sample 5 times for uncertainty estimation. Similarly, we use $m = 5$ for the ensemble baseline and the distribution distillation. Note that Ovadia et al. [335] found that a relative small ensemble size (e.g. $m = 5$) may indeed be sufficient in practice. We also train Prior Networks where we set $\beta_{\text{in}} = 1e^2$ as suggested in the original papers [270, 273]. Prior Networks use Fashion-MNIST and SVHN as training OOD datasets for MNIST and CIFAR-10, respectively. As there is no available OOD dataset for the Sensorless Drive dataset, we use Gaussian noise as training OOD data.

Natural Posterior Network. We perform a grid search for the entropy regularizer λ in the range $[1e^{-5}, 0]$, for the latent dimension H in $\{4, 8, 16, 32\}$, for the certainty budget N_H in $\{e^{\frac{1}{2}H}, e^H, e^{\log(\sqrt{4\pi})H}\}$, and for normalizing flow type between radial flows [358] with 8, 16 layers and Masked Autoregressive flows [339, 154] with 4, 8, 16 layers. Further results on latent dimensions, density types, number of normalizing flow layers and certainty budget are presented in Appendix B.9.6. We use "warm-up" training for the normalizing flows for all datasets except for the simple Concrete and Kin8nm datasets, and the NYU Depth v2 dataset which starts from a pretrained encoder. We use "fine-tuning" for the normalizing flows for all datasets except for the simple Concrete and Kin8nm datasets. As prior parameters, we set $\boldsymbol{\chi}^{\text{prior}} = \mathbf{1}_C/C, n^{\text{prior}} = C$ for classification, $\boldsymbol{\chi}^{\text{prior}} = (0, 100)^T, n^{\text{prior}} = 1$ for regression and $\chi^{\text{prior}} = 1, n^{\text{prior}} = 1$ for count prediction. Note that the mean of these prior distributions correspond to an equiprobable Categorical distribution $\text{Cat}(\mathbf{1}_C/C)$, a Normal distribution with large variance $\mathcal{N}(0, 10)$ and a Poisson distribution with a unitary mean Poi(1). Those prior target distributions represent the safe default prediction when no evidence is predicted.

B.8 Experiment Details

Target Error Metric. For classification, we use the standard accuracy $\frac{1}{N}\sum_i \mathbb{I}[\boldsymbol{y}^{*,(i)} = \boldsymbol{y}^{(i)}]$ where $\boldsymbol{y}^{*,(i)}$ is the one-hot true label and $\boldsymbol{y}^{(i)}$ is the one-hot predicted label. For regression, we use the standard Root Mean Square Error $\sqrt{\frac{1}{N}\sum_i^N (y^{*,(i)} - y^{(i)})^2}$.

Calibration Metric. For classification, we use the Brier score which is computed as $\frac{1}{C}\sum_i^N ||\boldsymbol{p}^{(i)} - \boldsymbol{y}^{(i)}||_2$ where $\boldsymbol{p}^{(i)}$ is the predicted softmax probability and $\boldsymbol{y}^{(i)}$ is the one-hot encoded ground-truth label. For regression and count prediction, we use the absolute difference between the percentile p and the percentage of target lying in the confidence interval $I_p = [0, \frac{p}{2}] \cup [1 - \frac{p}{2}, 1]$ under the predicted target distribution. Formally, we compute $p_{\text{pred}} = \frac{1}{N}\sum_i \mathbb{I}[F_{\boldsymbol{\theta}^{(i)}}(y^{*,(i)})) \in I_p]$ where $F_{\boldsymbol{\theta}^{(i)}}(y^{*,(i)}) = \mathbb{P}(y \leq y^{*,(i)} \,|\, \boldsymbol{\theta}^{(i)})$ is the cumulative function of the predicted target distribution evaluated at the true target. For

[3] `https://www.daml.in.tum.de/natpn`

example, the percentile $p = 0.1$ would be compared to $p_{\text{pred}} = \frac{1}{N}\sum_i \mathbb{I}[F_{\boldsymbol{\theta}^{(i)}}(y^{*,(i)})) \in [0, 0.05] \cup [0.95, 1]]$ which should be close to 0.10 for calibrated predictions. We compute a single calibration score by summing the square difference for $p \in \{0.1, \ldots, 0.9\}$ i.e. $\sqrt{\sum_p (p - p_{\text{pred}})^2}$ [242].

OOD Metric. The OOD detection task can be evaluated as a binary classification. Hence, we assign class 1 to ID data and class 0 to OOD data task and use the aleatoric and epistemic uncertainty estimates as scores for OOD data. It enables to compute final scores using the area under the precision-recall curve (AUC-PR) and the area under the receiver operating characteristic curve (AUC-ROC). Both metrics have been scaled by 100. We obtain numbers in $[0, 100]$ for all scores instead of $[0, 1]$. Results for AUC-ROC are reported in Appendix B.9.7. For the aleatoric uncertainty, we use the negative entropy of the predicted target distribution. For the epistemic uncertainty, we use the predicted evidence for models parametrizing conjugate-prior, or the variance of the predicted winning probability class for classification and the variance of the mean for regression and class count prediction for ensemble or dropout.

Inference Time Metric. We measure inference time of models in ms and used NVIDIA GTX 1080 Ti GPUs. We evaluate the inference for one classification dataset (CIFAR-10) and one regression dataset (NYU Depth v2). For evaluation, we use a randomly initialized model and simply push random data through the model with batch size of 4,096 CIFAR10 and batch size of 4 for NYU Depth v2. The final numbers are averaged over 100 batches excluding the first batch due to GPU initialization. Compared models shared the same backbone architecture.

B.9 Additional Experiments

B.9.1 MNIST, Fashion MNIST, CIFAR10 and Bike Sharing results

Table B.1: Results on MNIST (classification with Categorical target distribution). Best scores among all single-pass models are in bold. Best scores among all models are starred. Gray numbers indicate that R-PriorNet has seen samples from the FMNIST dataset during training.

	Accuracy	Brier	K. Alea.	K. Epist.	F. Alea.	F. Epist.	OODom Alea.	OODom Epist.
Dropout	99.45 ± 0.01	1.07 ± 0.05	98.27 ± 0.05	97.82 ± 0.08	*99.40 ± 0.03	98.01 ± 0.14	43.86 ± 1.62	74.09 ± 0.92
Ensemble	99.46 ± 0.02	1.02 ± 0.02	98.39 ± 0.07	98.43 ± 0.05	99.33 ± 0.06	98.73 ± 0.08	40.98 ± 1.80	66.54 ± 0.58
NatPE	*99.55 ± 0.01	*0.84 ± 0.03	96.39 ± 0.73	*99.61 ± 0.02	97.49 ± 0.85	*99.70 ± 0.04	*100.00 ± 0.00	*100.00 ± 0.00
StandardNet	98.91 ± 0.06	1.81 ± 0.14	95.81 ± 0.44	–	96.29 ± 1.04	–	47.53 ± 3.44	–
SNGP	99.34 ± 0.03	2.62 ± 0.04	98.85 ± 0.11	–	98.04 ± 0.34	–	***100.00 ± 0.00**	–
R-PriorNet	99.35 ± 0.04	**0.97 ± 0.03**	***99.33 ± 0.18**	99.28 ± 0.25	100.00 ± 0.00	100.00 ± 0.00	97.48 ± 0.66	31.03 ± 0.13
EnD^2	99.24 ± 0.05	6.19 ± 0.13	98.36 ± 0.15	98.76 ± 0.13	**99.25 ± 0.16**	99.35 ± 0.14	48.09 ± 1.38	31.60 ± 0.39
PostNet	99.36 ± 0.02	1.33 ± 0.04	98.88 ± 0.05	98.79 ± 0.07	98.89 ± 0.23	98.85 ± 0.23	***100.00 ± 0.00**	***100.00 ± 0.00**
NatPN	**99.47 ± 0.02**	1.09 ± 0.03	99.20 ± 0.20	**99.39 ± 0.08**	99.16 ± 0.28	**99.54 ± 0.09**	99.99 ± 0.01	***100.00 ± 0.00**

Table B.2: Results on FMNIST (classification with Categorical target distribution). Best scores among all single-pass models are in bold. Best scores among all models are starred. Gray numbers indicate that R-PriorNet has seen samples from the KMNIST dataset during training.

	Accuracy	Brier	M. Alea.	M. Epist.	K. Alea.	K. Epist.	OODom Alea.	OODom Epist.
Dropout	92.44 ± 0.17	13.89 ± 0.31	60.75 ± 1.41	75.85 ± 1.73	76.57 ± 1.30	92.48 ± 0.46	39.97 ± 0.69	90.90 ± 1.74
Ensemble	92.64 ± 0.10	13.63 ± 0.25	77.14 ± 1.49	90.78 ± 0.75	86.20 ± 0.76	95.16 ± 0.35	37.30 ± 0.83	82.93 ± 0.96
NatPE	*92.89 ± 0.06	14.44 ± 0.06	82.56 ± 0.33	96.38 ± 0.29	92.12 ± 0.17	*98.79 ± 0.09	*100.00 ± 0.00	*100.00 ± 0.00
StandardNet	90.28 ± 0.24	17.12 ± 0.53	71.81 ± 2.43	–	82.28 ± 0.97	–	32.82 ± 0.73	–
SNGP	91.38 ± 0.08	16.73 ± 0.46	89.40 ± 1.66	–	95.31 ± 0.42	–	**100.00 ± 0.00**	–
R-PriorNet	91.53 ± 0.10	***12.21 ± 0.20**	***98.83 ± 0.49**	***99.54 ± 0.18**	99.96 ± 0.02	99.99 ± 0.00	72.23 ± 6.32	48.84 ± 6.09
EnD^2	**91.84 ± 0.03**	29.23 ± 0.79	79.32 ± 1.39	91.61 ± 1.04	91.99 ± 0.06	98.36 ± 0.20	43.70 ± 3.37	36.73 ± 3.74
PostNet	91.04 ± 0.10	16.11 ± 0.30	90.56 ± 1.25	92.10 ± 1.77	***96.65 ± 0.33**	97.06 ± 0.42	***100.00 ± 0.00**	***100.00 ± 0.00**
NatPN	91.65 ± 0.14	14.88 ± 0.30	81.12 ± 2.77	96.51 ± 0.81	93.03 ± 1.00	**98.38 ± 0.23**	99.99 ± 0.01	***100.00 ± 0.00**

Table B.3: Classification results on CIFAR-10 with Categorical target distribution. Best scores among all single-pass models are in bold. Best scores among all models are starred. Gray numbers indicate that R-PriorNet has seen samples from the SVHN dataset during training.

	Accuracy	Brier	SVHN Alea.	SVHN Epist.	CelebA Alea.	CelebA Epist.	OODom Alea.	OODom Epist.
Dropout	88.15 ± 0.20	19.59 ± 0.41	80.63 ± 1.59	73.09 ± 1.51	71.84 ± 4.28	71.04 ± 3.92	18.42 ± 1.11	49.69 ± 9.10
Ensemble	*89.95 ± 0.11	17.33 ± 0.17	85.26 ± 0.84	82.51 ± 0.63	76.20 ± 0.87	74.23 ± 0.78	25.30 ± 4.02	89.21 ± 7.55
NatPE	89.21 ± 0.09	17.41 ± 0.12	85.66 ± 0.34	*83.16 ± 0.67	*78.95 ± 1.15	*82.06 ± 1.30	87.27 ± 1.79	*98.88 ± 0.26
SNGP	84.06 ± 1.68	30.49 ± 2.99	79.95 ± 1.82	–	67.82 ± 3.67	–	***96.00 ± 1.67**	–
R-PriorNet	**88.94 ± 0.23**	***15.99 ± 0.32**	99.87 ± 0.02	99.94 ± 0.01	67.74 ± 4.86	59.55 ± 7.90	42.21 ± 8.77	38.25 ± 9.82
EnD^2	84.03 ± 0.25	40.84 ± 0.36	***86.47 ± 0.66**	**81.84 ± 0.92**	75.54 ± 1.79	75.94 ± 1.82	42.19 ± 8.77	15.79 ± 0.27
PostNet	87.95 ± 0.20	20.19 ± 0.40	82.35 ± 0.68	79.24 ± 1.49	72.96 ± 2.33	75.84 ± 1.61	85.89 ± 4.10	92.30 ± 2.18
NatPN	87.90 ± 0.16	19.99 ± 0.46	82.29 ± 1.11	77.83 ± 1.22	**76.01 ± 1.18**	**76.87 ± 3.38**	93.67 ± 3.03	**94.90 ± 3.09**

B.9.2 Uncertainty Visualization on Toy Datasets

We visualize the aleatoric and the epistemic uncertainty for two toy datasets with three classes with the same number of training examples for the three classes (see Fig. B.1) and different number of training examples for the three classes (see Fig. B.3). The predictions are more aleatorically certain close to training samples. The preidctions are more epistemically uncertain close to fewer training examples and very epistemically uncertain for region far from all training data.

Figure B.1: Visualization of the aleatoric and epistemic uncertainty on a 2D toy dataset with 3 classes with 900 training samples for each class.

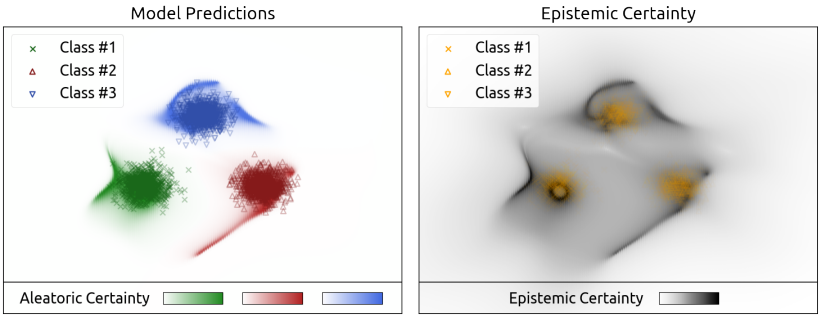
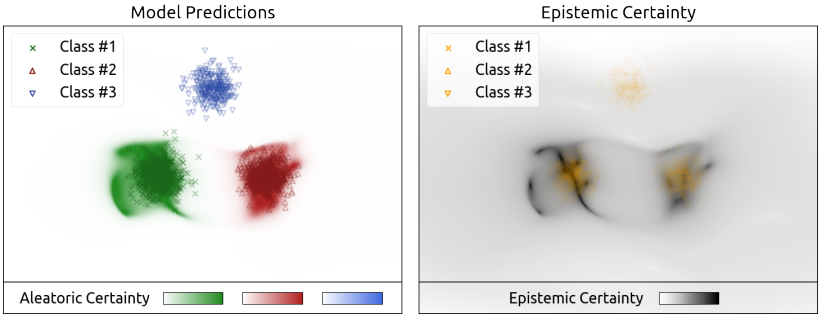


Table B.4: Results on the Bike Sharing Dataset with Normal $\mathcal{N}$ and Poison Poi target distributions. Best scores among all single-pass models are in bold. Best scores among all models are starred.

	RMSE	Calibration	Winter Epist.	Spring Epist.	Autumn Epist.	OODom Epist.
Dropout-$\mathcal{N}$	70.20 ± 1.30	6.05 ± 0.77	15.26 ± 0.51	13.66 ± 0.16	15.11 ± 0.46	99.99 ± 0.01
Ensemble-$\mathcal{N}$	*48.02 ± 2.78	5.88 ± 1.00	42.46 ± 2.29	21.28 ± 0.38	21.97 ± 0.58	*100.00 ± 0.00
StandardNet-$\mathcal{N}$	58.49 ± 4.37	2.32 ± 0.88	–	–	–	–
EvReg-$\mathcal{N}$	**49.58 ± 1.51**	3.77 ± 0.81	17.19 ± 0.76	15.54 ± 0.65	14.75 ± 0.29	34.99 ± 17.02
NatPN-$\mathcal{N}$	49.85 ± 1.38	***1.95 ± 0.34**	***55.04 ± 6.81**	***23.25 ± 1.20**	***27.78 ± 2.47**	***100.00 ± 0.00**

Figure B.3: Visualization of the aleatoric and epistemic uncertainty on a 2D toy dataset with 3 classes with 900 training samples for class 1 (green), 600 training samples for class 2 (red) and 300 training samples for class 2 (blue).



B.9.3 Latent Space Visualizations

We propose additional visualizations of the latent space for MNIST with t-SNE [65] with different perplexities (see Fig. B.5). For all perplexities, we clearly observe ten green clusters corresponding to the ten classes for MNIST. The KNMIST (OOD) samples in red can easily be separated from the MNIST (ID) samples in green. As desired, NatPN assigns higher log-probabilities used in evidence computation to ID samples from MNIST.

B.9.4 Histogram of Uncertainty Estimates

We visualize the histogram distribution of the entropy of the posterior distribution accounting for predictive uncertainty for ID (MNIST/NYU) and OOD (KNMIST and Fashion-MNIST/LSUN classroom and LSUN church + KITTI) (see Fig. B.6). We clearly observe lower predictive entropy for ID data than for OOD data for both MNIST and NYU datasets. On one hand, the entropy clearly differentiates between ID data (MNIST) and any other OOD datasets (KMNIST, Fashion MNIST, OODom) for classification. We intuitively explain this clear distinction since the samples from the OOD datasets are irrelevant for the digit classification task. On the other hand, the entropy is still a good indicator of ID (NYU) and OOD datasets (LSUN classroom and LSUN church + KITTI) for regression although the distinction between ID and OOD datasets is less strong compared to MNIST. We intuitively explain this behavior since the task of depth estimation is still relevant to LSUN classroom and LSUN church + KITTI.

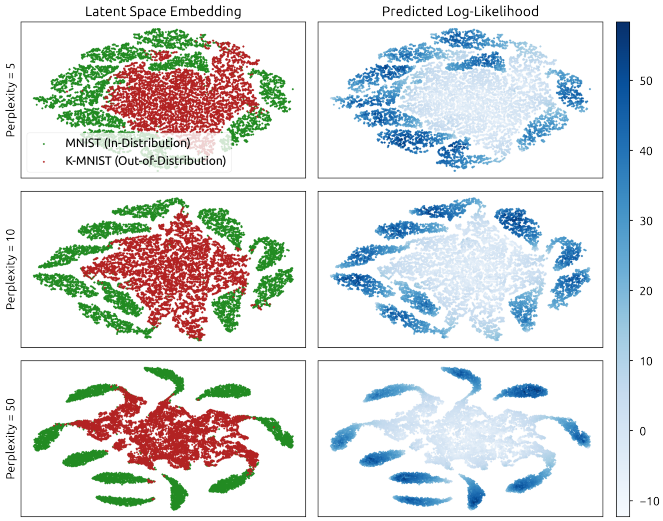
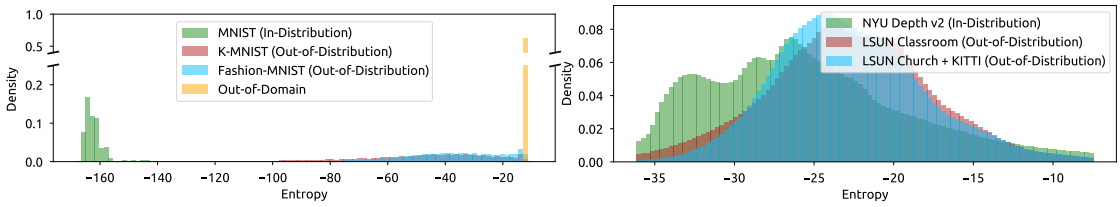


Figure B.5: t-SNE visualization of the latent space of NatPN on MNIST (ID) vs KMNIST (OOD). On the left, The ID data (MNIST in green) can easily be distinguished from the OOD data (KMNIST in red). On the right, NatPN correctly assigns higher likelihood to ID data.

Figure B.6: Histogram of the entropy of the posterior distribution accounting for the predictive uncertainty of NatPN on MNIST (ID) vs KMNIST, Fashion-MNIST, Out-Of-Domain (OOD) and NYU (ID) vs LSUN classroom and LSUN church + KITTI (OOD). In both cases, low entropy is a good indicator of in-distribution data.



B.9.5 Uncertainty Visualization on NYU Depth v2 Dataset

We visualize the prediction and the predictive uncertainty per pixel for the NYU Depth v2 dataset (see Fig. B.8). We observe accurate target predictions compared to the ground truth depth of the images. Further NatPN assigns higher uncertainty for pixels close to object edges, which is reasonable since the depth abruptly change at these locations.

B.9.6 Hyper-Parameter Study

As a ablation study, we also report the results of the grid search on the latent dimension, normalizing flow types and number of normalizing flow layers for MNIST, CIFAR-10 and Bike Sharing datasets in Tables B.5 to B.8. While most models converge to fairly good

Figure B.8: Visualization of the predicted depth and predictive uncertainty estimates of NatPN per pixel on the NYU Depth v2 dataset. NatPN predicts accurate depth uncertainty and reasonably assigns higher uncertainty to object edges.

uncertainty estimates, we notice that 16 layers of simple radial flows on latent spaces of 16 dimensions were achieving very good results in practice.

Changing the flow type or the number of normalizing flow layers does not lead to strong variations of the results except for Bike sharing with Poisson target distributions. In this case, more complex MAF normalizing flows improve NatPN performance.

The latent dimension appears to be a more important choice for the model convergence. As an example, a higher latent dimension of 16 or 32 leads to significantly better performances than a latent dimension of 4 on MNIST, CIFAR10 and Bike Sharing datasets. We hypothesize that too low latent dimensions are less able to encode the necessary information for the prediction task, leading to worse target errors.

Further, we compare three different variants of the certainty budget N_H used in the evidence computation $n^{(i)} = N_H \mathbb{P}(\boldsymbol{z}^{(i)} \,|\, \boldsymbol{\omega})$: a constant unit budget (i.e. $N_H = 1$ (I)) corresponding to a fixed budget regardless of the number of training data and the latent dimension, a budget equals to the number of training data (i.e. $N_H = N$ (II)) similarly to Charpentier et al. [67], or a budget which scales exponentially with the number of latent dimensions (e.g. N_H equal to $e^{\frac{1}{2}H}$ (III), e^{H} (IV) or $e^{\log(\sqrt{4\pi})H}$ (V)). We observe that scaling the budget w.r.t. the latent space dimension (H-budget) is more stable in practice than constant budget (1-budget) and budget related to the number of training data (N-budget) (see. Tables B.5 to B.8). In aprticular, the H-budgets achieve more better results on higher latent dimensions and performance on par with the other certainty

budget scheme otherwise. The intuition is that due to the curse of dimensionality, the expected value of a probability density function $\mathbb{E}_{\boldsymbol{z}}[\mathbb{P}(\boldsymbol{z})]$ tends to decrease exponentially fast. For example, we have $\mathbb{E}_{\boldsymbol{z}}[\mathbb{P}(\boldsymbol{z})] = \frac{1}{(\sqrt{4\pi})^H}$ when $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ in a H-dimensional space. Increasing the certainty budget N_H exponentially w.r.t. to the dimension H avoids numerical issues by allocating close to 0 evidence to latent representations. In our experiments, we use a grid search in different exponential scaling N_H equal to $e^{\frac{1}{2}H}$ (III), e^H (IV), $e^{\log(\sqrt{4\pi})H}$ (V).

B.9.7 OOD Detection with AUC-ROC Scores

In addition to the AUC-APR scores, we report the OOD detection results for MNIST, CIFAR10 and Bike Sharing datasets in Tables B.9 to B.11 with AUC-ROC scores. Similarly as with AUC-APR scores, NatPN and NatPE achieve very competitive performances compare to the baselines. It particular, they outperform all baselines to detect challenging OODom data.

Table B.5: MNIST comparison (<latent dim> – <certainty budget> – <radial layers>/<MAF layers>). Bold and starred number indicate best score among all models.

	Accuracy	Brier	K. Alea.	K. Epist.	F. Alea.	F. Epist.	OODom Alea.	OODom Epist.
H = **4**, N_H = **I**, **Flow = 8/0**	61.92 ± 3.79	64.33 ± 3.98	88.70 ± 1.65	87.66 ± 2.47	90.07 ± 3.43	89.52 ± 2.25	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **4**, N_H = **I**, **Flow = 16/0**	77.46 ± 6.26	59.11 ± 1.96	91.38 ± 2.50	91.48 ± 0.85	92.46 ± 1.72	93.03 ± 1.60	99.91 ± 0.04	***100.00 ± 0.00**
H = **4**, N_H = **I**, **Flow = 0/4**	75.04 ± 10.10	59.46 ± 9.92	92.36 ± 3.34	92.64 ± 4.06	93.28 ± 2.66	91.90 ± 4.71	99.82 ± 0.05	***100.00 ± 0.00**
H = **4**, N_H = **I**, **Flow = 0/8**	80.97 ± 4.29	58.35 ± 1.75	91.87 ± 2.55	92.65 ± 2.52	93.58 ± 2.93	96.50 ± 1.32	99.85 ± 0.04	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 8/0**	99.33 ± 0.05	3.43 ± 0.08	92.45 ± 0.33	67.84 ± 2.95	96.55 ± 1.23	71.57 ± 4.00	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 16/0**	99.36 ± 0.03	2.72 ± 0.08	94.87 ± 0.49	86.13 ± 1.47	95.03 ± 0.44	88.62 ± 1.80	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 0/4**	98.86 ± 0.05	4.26 ± 0.90	98.93 ± 0.16	98.04 ± 0.38	99.16 ± 0.23	98.52 ± 0.52	99.71 ± 0.02	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 0/8**	98.77 ± 0.15	5.44 ± 1.57	98.43 ± 0.26	97.87 ± 0.40	98.85 ± 0.26	98.35 ± 0.44	99.76 ± 0.03	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 8/0**	78.14 ± 5.55	48.46 ± 4.78	88.79 ± 1.05	87.28 ± 1.96	90.56 ± 1.10	89.69 ± 1.88	***100.00 ± 0.00**	99.99 ± 0.01
H = **4**, N_H = **III**, **Flow = 16/0**	79.56 ± 5.27	41.44 ± 4.86	93.92 ± 1.25	93.99 ± 0.39	95.61 ± 0.88	95.64 ± 0.58	99.92 ± 0.07	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 0/4**	76.08 ± 7.67	58.56 ± 11.60	93.72 ± 3.51	94.58 ± 2.66	94.34 ± 3.39	96.60 ± 1.86	99.86 ± 0.05	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 0/8**	79.91 ± 10.84	50.78 ± 10.99	94.91 ± 2.23	96.44 ± 1.49	95.97 ± 1.37	98.35 ± 0.54	99.78 ± 0.07	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 8/0**	85.31 ± 3.89	34.01 ± 5.20	89.65 ± 1.24	86.81 ± 1.36	92.78 ± 0.40	86.93 ± 2.47	***100.00 ± 0.00**	99.99 ± 0.00
H = **4**, N_H = **IV**, **Flow = 16/0**	89.36 ± 0.15	26.48 ± 2.64	94.56 ± 0.65	93.48 ± 0.88	96.03 ± 1.01	94.74 ± 1.26	99.98 ± 0.02	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 0/4**	91.64 ± 6.70	22.61 ± 13.67	96.63 ± 1.92	93.07 ± 1.49	98.43 ± 0.52	94.55 ± 2.01	99.81 ± 0.06	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 0/8**	98.14 ± 0.46	12.73 ± 5.11	98.68 ± 0.25	97.61 ± 0.64	98.93 ± 0.20	97.73 ± 0.63	99.78 ± 0.05	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 8/0**	93.59 ± 2.32	20.40 ± 3.68	92.06 ± 1.39	87.15 ± 2.25	93.55 ± 1.03	87.11 ± 2.97	***100.00 ± 0.00**	99.99 ± 0.00
H = **4**, N_H = **V**, **Flow = 16/0**	97.19 ± 1.97	14.66 ± 3.13	94.67 ± 1.45	92.82 ± 1.04	95.80 ± 1.17	95.27 ± 0.70	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 0/4**	96.69 ± 1.56	16.51 ± 6.21	97.78 ± 0.63	94.58 ± 1.06	96.95 ± 1.76	92.94 ± 1.74	99.80 ± 0.03	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 0/8**	98.34 ± 0.19	6.72 ± 1.54	98.85 ± 0.15	98.02 ± 0.49	99.08 ± 0.08	98.41 ± 0.34	99.69 ± 0.04	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 8/0**	99.45 ± 0.04	1.49 ± 0.11	97.75 ± 0.32	89.16 ± 0.34	98.13 ± 0.27	89.16 ± 0.41	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 16/0**	99.48 ± 0.01	1.41 ± 0.07	***99.32 ± 0.11**	99.34 ± 0.05	***99.52 ± 0.07**	***99.59 ± 0.05**	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 0/4**	99.18 ± 0.05	1.83 ± 0.08	98.78 ± 0.11	97.92 ± 0.15	99.39 ± 0.07	98.32 ± 0.51	99.93 ± 0.02	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 0/8**	99.13 ± 0.06	1.97 ± 0.09	98.97 ± 0.04	98.33 ± 0.07	***99.52 ± 0.04**	99.28 ± 0.09	99.88 ± 0.02	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 8/0**	99.42 ± 0.02	1.42 ± 0.10	96.62 ± 0.37	82.95 ± 1.46	97.34 ± 0.59	82.85 ± 1.37	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 16/0**	99.39 ± 0.02	1.52 ± 0.23	98.19 ± 0.38	95.92 ± 1.30	98.48 ± 0.37	96.20 ± 1.30	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 0/4**	99.24 ± 0.04	1.74 ± 0.08	98.65 ± 0.12	97.49 ± 0.20	99.29 ± 0.09	98.62 ± 0.20	99.94 ± 0.02	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 0/8**	99.26 ± 0.04	1.73 ± 0.09	98.89 ± 0.06	98.09 ± 0.12	99.33 ± 0.11	98.73 ± 0.27	99.95 ± 0.01	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 8/0**	99.47 ± 0.02	1.90 ± 0.50	95.08 ± 1.14	83.27 ± 0.43	96.03 ± 1.33	82.95 ± 0.30	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 16/0**	99.47 ± 0.02	***1.09 ± 0.03**	99.20 ± 0.20	***99.39 ± 0.08**	99.16 ± 0.28	99.54 ± 0.09	99.99 ± 0.01	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 0/4**	99.25 ± 0.03	1.65 ± 0.07	98.72 ± 0.07	97.95 ± 0.15	99.44 ± 0.04	98.96 ± 0.17	99.95 ± 0.01	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 0/8**	99.26 ± 0.05	1.78 ± 0.07	98.89 ± 0.08	98.24 ± 0.20	99.23 ± 0.28	98.70 ± 0.37	99.95 ± 0.01	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 8/0**	99.33 ± 0.04	1.58 ± 0.03	97.08 ± 0.35	77.40 ± 1.79	98.15 ± 0.61	77.46 ± 1.82	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 16/0**	99.37 ± 0.03	1.47 ± 0.11	97.52 ± 0.50	93.53 ± 0.89	98.04 ± 0.55	94.28 ± 0.60	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 0/4**	99.23 ± 0.06	1.74 ± 0.10	98.64 ± 0.06	97.00 ± 0.38	99.31 ± 0.07	98.24 ± 0.35	99.93 ± 0.02	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 0/8**	99.17 ± 0.03	1.83 ± 0.04	98.54 ± 0.10	97.43 ± 0.23	99.15 ± 0.15	98.28 ± 0.34	99.93 ± 0.01	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 8/0**	99.36 ± 0.03	1.46 ± 0.02	97.60 ± 0.63	72.45 ± 3.67	98.25 ± 0.83	72.69 ± 3.58	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 16/0**	99.38 ± 0.02	1.58 ± 0.12	96.17 ± 0.61	90.44 ± 1.80	96.80 ± 0.33	91.12 ± 1.87	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 0/4**	99.15 ± 0.04	1.89 ± 0.08	98.57 ± 0.13	96.06 ± 0.34	99.08 ± 0.11	97.79 ± 0.42	99.91 ± 0.01	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 0/8**	99.23 ± 0.04	1.98 ± 0.08	98.53 ± 0.04	97.03 ± 0.19	98.94 ± 0.13	97.83 ± 0.20	99.90 ± 0.01	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 8/0**	***99.49 ± 0.03**	2.28 ± 0.89	92.36 ± 1.75	76.18 ± 2.10	93.90 ± 1.59	76.38 ± 2.11	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 16/0**	99.47 ± 0.01	1.47 ± 0.26	96.20 ± 0.25	95.95 ± 1.21	95.96 ± 0.77	95.97 ± 1.14	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 0/4**	99.29 ± 0.02	1.42 ± 0.04	99.02 ± 0.05	98.14 ± 0.16	99.46 ± 0.02	98.57 ± 0.11	99.95 ± 0.01	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 0/8**	99.27 ± 0.04	1.45 ± 0.06	98.95 ± 0.08	98.52 ± 0.08	99.35 ± 0.06	98.80 ± 0.04	99.99 ± 0.00	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 8/0**	99.44 ± 0.05	1.29 ± 0.18	96.03 ± 1.88	73.89 ± 2.45	96.83 ± 1.13	73.98 ± 2.41	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 16/0**	99.46 ± 0.05	1.94 ± 0.79	96.57 ± 1.17	94.13 ± 1.47	97.82 ± 0.84	94.69 ± 1.39	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 0/4**	99.26 ± 0.01	1.48 ± 0.02	98.90 ± 0.05	97.98 ± 0.06	99.29 ± 0.05	98.44 ± 0.09	99.94 ± 0.01	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 0/8**	99.30 ± 0.06	1.39 ± 0.09	98.89 ± 0.07	98.14 ± 0.14	99.45 ± 0.06	98.64 ± 0.13	99.98 ± 0.01	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 8/0**	***99.49 ± 0.01**	1.20 ± 0.10	97.91 ± 0.79	73.77 ± 3.06	98.73 ± 0.57	73.79 ± 2.97	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 16/0**	99.44 ± 0.02	1.90 ± 0.69	96.16 ± 1.53	90.95 ± 1.20	97.42 ± 1.12	91.46 ± 1.24	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 0/4**	99.31 ± 0.02	1.38 ± 0.03	98.89 ± 0.10	97.73 ± 0.06	99.29 ± 0.11	98.41 ± 0.08	99.97 ± 0.01	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 0/8**	99.28 ± 0.04	1.33 ± 0.05	98.80 ± 0.09	98.16 ± 0.05	99.27 ± 0.08	98.52 ± 0.13	99.98 ± 0.01	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 8/0**	99.36 ± 0.03	1.47 ± 0.05	97.97 ± 0.39	71.87 ± 1.31	99.00 ± 0.10	72.28 ± 1.30	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 16/0**	99.38 ± 0.02	2.41 ± 0.66	95.45 ± 1.73	83.06 ± 1.62	96.91 ± 1.24	83.48 ± 1.65	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 0/4**	99.22 ± 0.03	1.50 ± 0.09	98.77 ± 0.08	97.49 ± 0.11	99.27 ± 0.05	98.33 ± 0.09	99.97 ± 0.01	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 0/8**	99.25 ± 0.04	1.34 ± 0.07	98.62 ± 0.09	97.70 ± 0.15	99.24 ± 0.06	98.09 ± 0.34	99.99 ± 0.00	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 8/0**	99.35 ± 0.02	1.43 ± 0.08	98.43 ± 0.29	64.15 ± 3.79	99.22 ± 0.16	64.37 ± 3.84	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 16/0**	99.33 ± 0.03	1.76 ± 0.33	97.41 ± 0.93	84.15 ± 1.84	98.26 ± 0.60	85.01 ± 2.01	***100.00 ± 0.00**	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 0/4**	99.18 ± 0.05	1.58 ± 0.08	98.61 ± 0.09	96.71 ± 0.24	99.11 ± 0.07	97.34 ± 0.17	99.94 ± 0.01	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 0/8**	99.19 ± 0.03	1.58 ± 0.06	98.58 ± 0.10	96.98 ± 0.15	99.19 ± 0.08	96.99 ± 0.28	99.99 ± 0.00	***100.00 ± 0.00**

Table B.6: CIFAR10 comparison (<latent dim> – <certainty budget> – <radial layers>/<MAF layers>). Bold and starred number indicate best score among all models.

	Accuracy	Brier	SVHN Alea.	SVHN Epist.	CelebA Alea.	CelebA Epist.	OODom Alea.	OODom Epist.
H = **4**, N_H = **I**, **Flow = 8/0**	42.73 ± 6.40	81.56 ± 2.72	67.03 ± 3.15	49.84 ± 6.09	56.86 ± 2.13	39.01 ± 7.65	95.33 ± 4.48	97.68 ± 2.32
H = **4**, N_H = **I**, **Flow = 16/0**	46.39 ± 6.64	77.73 ± 3.57	66.43 ± 2.39	46.63 ± 5.92	68.22 ± 3.82	41.98 ± 6.52	98.93 ± 0.85	99.61 ± 0.39
H = **4**, N_H = **I**, **Flow = 0/4**	53.79 ± 2.51	80.83 ± 4.54	59.77 ± 5.82	35.96 ± 2.59	64.95 ± 1.92	48.27 ± 5.85	97.76 ± 1.90	99.86 ± 0.07
H = **4**, N_H = **I**, **Flow = 0/8**	51.11 ± 3.67	79.13 ± 2.10	58.22 ± 4.19	33.22 ± 8.03	63.97 ± 3.12	42.75 ± 9.47	84.62 ± 10.88	88.24 ± 11.76
H = **4**, N_H = **II**, **Flow = 8/0**	88.02 ± 0.12	22.61 ± 0.18	75.40 ± 2.82	40.98 ± 3.98	62.73 ± 6.72	35.68 ± 3.66	82.17 ± 3.49	60.66 ± 7.27
H = **4**, N_H = **II**, **Flow = 16/0**	87.80 ± 0.09	22.43 ± 0.33	78.00 ± 1.68	60.02 ± 2.96	63.55 ± 2.08	46.15 ± 4.20	88.55 ± 3.72	88.82 ± 2.85
H = **4**, N_H = **II**, **Flow = 0/4**	87.10 ± 0.10	22.42 ± 0.16	83.35 ± 0.72	67.89 ± 4.11	75.76 ± 1.33	63.04 ± 4.04	89.20 ± 3.60	91.43 ± 3.81
H = **4**, N_H = **II**, **Flow = 0/8**	86.28 ± 0.74	24.15 ± 1.52	83.69 ± 0.48	67.20 ± 3.22	75.30 ± 1.66	63.56 ± 5.25	80.92 ± 8.99	83.04 ± 7.45
H = **4**, N_H = **III**, **Flow = 8/0**	68.54 ± 4.06	64.35 ± 3.30	71.61 ± 2.06	57.05 ± 5.90	70.14 ± 2.66	51.87 ± 3.56	93.39 ± 6.00	95.61 ± 3.82
H = **4**, N_H = **III**, **Flow = 16/0**	69.41 ± 4.00	59.98 ± 4.79	74.52 ± 1.11	52.59 ± 6.96	73.48 ± 2.43	51.59 ± 7.09	93.43 ± 3.42	93.45 ± 4.86
H = **4**, N_H = **III**, **Flow = 0/4**	64.80 ± 4.59	61.82 ± 5.62	67.99 ± 3.37	37.00 ± 6.95	66.88 ± 5.61	38.00 ± 10.01	99.70 ± 0.02	99.97 ± 0.02
H = **4**, N_H = **III**, **Flow = 0/8**	61.49 ± 5.78	66.53 ± 6.29	57.58 ± 5.84	34.37 ± 3.79	62.22 ± 6.68	39.86 ± 9.87	94.73 ± 3.13	98.71 ± 1.02
H = **4**, N_H = **IV**, **Flow = 8/0**	84.45 ± 0.44	38.86 ± 1.13	73.65 ± 1.66	53.64 ± 1.71	71.68 ± 2.13	52.30 ± 1.49	69.65 ± 5.61	66.36 ± 7.76
H = **4**, N_H = **IV**, **Flow = 16/0**	83.16 ± 0.83	35.51 ± 1.55	80.65 ± 2.67	59.70 ± 3.43	76.69 ± 3.23	50.09 ± 2.76	80.24 ± 7.59	83.05 ± 5.77
H = **4**, N_H = **IV**, **Flow = 0/4**	72.13 ± 2.90	48.67 ± 3.63	72.17 ± 1.95	27.43 ± 3.60	71.77 ± 2.24	27.09 ± 2.25	99.22 ± 0.53	99.85 ± 0.09
H = **4**, N_H = **IV**, **Flow = 0/8**	72.34 ± 1.84	51.55 ± 3.08	68.02 ± 3.39	28.48 ± 2.16	69.02 ± 2.91	32.72 ± 4.89	99.13 ± 0.44	98.91 ± 0.68
H = **4**, N_H = **V**, **Flow = 8/0**	86.32 ± 0.64	32.24 ± 1.38	73.15 ± 4.29	50.79 ± 3.87	66.37 ± 6.81	47.11 ± 7.35	73.58 ± 7.20	63.29 ± 5.99
H = **4**, N_H = **V**, **Flow = 16/0**	86.18 ± 0.39	27.92 ± 0.56	79.67 ± 2.46	56.96 ± 4.80	74.68 ± 2.69	45.29 ± 5.32	77.59 ± 10.23	80.75 ± 8.14
H = **4**, N_H = **V**, **Flow = 0/4**	80.39 ± 0.94	37.33 ± 2.23	77.81 ± 2.52	31.59 ± 3.34	74.14 ± 1.57	31.93 ± 3.75	93.63 ± 5.76	93.85 ± 5.61
H = **4**, N_H = **V**, **Flow = 0/8**	77.84 ± 0.71	42.01 ± 1.03	75.27 ± 2.23	40.87 ± 6.42	73.46 ± 1.81	33.18 ± 5.09	98.59 ± 0.72	99.08 ± 0.61
H = **16**, N_H = **I**, **Flow = 8/0**	82.83 ± 0.84	30.89 ± 1.71	81.35 ± 0.60	65.51 ± 2.65	75.48 ± 1.65	61.89 ± 3.89	99.73 ± 0.24	99.82 ± 0.18
H = **16**, N_H = **I**, **Flow = 16/0**	84.83 ± 0.60	26.24 ± 1.06	82.39 ± 1.11	75.54 ± 2.52	75.30 ± 0.72	69.12 ± 2.15	99.90 ± 0.06	99.96 ± 0.04
H = **16**, N_H = **I**, **Flow = 0/4**	83.71 ± 1.37	27.35 ± 2.34	81.38 ± 0.94	56.29 ± 4.77	75.88 ± 0.51	51.73 ± 4.32	99.17 ± 0.65	99.92 ± 0.07
H = **16**, N_H = **I**, **Flow = 0/8**	84.13 ± 0.53	27.02 ± 1.02	80.24 ± 1.88	67.03 ± 2.59	74.59 ± 0.68	58.67 ± 8.57	99.93 ± 0.01	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 8/0**	86.82 ± 0.46	22.45 ± 0.88	83.73 ± 1.26	56.60 ± 3.15	76.98 ± 1.67	63.55 ± 2.26	85.41 ± 11.05	87.59 ± 7.40
H = **16**, N_H = **II**, **Flow = 16/0**	86.40 ± 0.46	23.06 ± 0.82	83.60 ± 0.96	74.04 ± 2.78	76.87 ± 0.92	74.00 ± 1.92	82.09 ± 10.05	92.84 ± 3.70
H = **16**, N_H = **II**, **Flow = 0/4**	88.02 ± 0.12	19.83 ± 0.21	81.19 ± 0.77	72.46 ± 1.88	74.19 ± 1.54	65.32 ± 2.02	99.14 ± 0.59	99.90 ± 0.07
H = **16**, N_H = **II**, **Flow = 0/8**	87.73 ± 0.11	19.97 ± 0.33	83.29 ± 0.70	71.93 ± 1.90	74.14 ± 2.13	63.20 ± 3.16	99.94 ± 0.01	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 8/0**	86.54 ± 0.55	23.44 ± 1.10	80.97 ± 1.57	59.46 ± 1.33	73.31 ± 2.36	58.10 ± 3.38	93.95 ± 2.43	91.07 ± 3.26
H = **16**, N_H = **III**, **Flow = 16/0**	86.89 ± 0.20	22.07 ± 0.41	83.11 ± 0.43	72.37 ± 1.29	75.52 ± 0.83	72.66 ± 3.46	96.20 ± 2.51	98.21 ± 1.06
H = **16**, N_H = **III**, **Flow = 0/4**	88.16 ± 0.16	19.54 ± 0.22	82.50 ± 1.29	62.54 ± 1.59	74.62 ± 0.98	57.76 ± 2.44	99.89 ± 0.01	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 0/8**	87.67 ± 0.17	20.23 ± 0.38	84.03 ± 0.93	70.91 ± 1.97	73.43 ± 1.33	61.20 ± 1.55	99.90 ± 0.03	99.99 ± 0.00
H = **16**, N_H = **IV**, **Flow = 8/0**	87.97 ± 0.07	19.98 ± 0.17	***84.42 ± 0.82**	62.10 ± 0.56	72.25 ± 1.85	65.17 ± 5.34	90.96 ± 4.50	81.63 ± 9.18
H = **16**, N_H = **IV**, **Flow = 16/0**	87.90 ± 0.16	19.99 ± 0.46	82.29 ± 1.11	***77.83 ± 1.22**	76.01 ± 1.18	***76.87 ± 3.38**	93.67 ± 3.03	94.90 ± 3.09
H = **16**, N_H = **IV**, **Flow = 0/4**	87.92 ± 0.22	19.93 ± 0.38	82.26 ± 1.22	67.03 ± 2.75	73.28 ± 1.25	63.06 ± 2.78	99.91 ± 0.02	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 0/8**	87.90 ± 0.19	19.70 ± 0.28	81.72 ± 0.44	64.76 ± 4.18	74.90 ± 0.94	66.46 ± 3.04	99.82 ± 0.12	99.99 ± 0.01
H = **16**, N_H = **V**, **Flow = 8/0**	88.17 ± 0.17	19.78 ± 0.29	83.67 ± 0.66	56.34 ± 0.85	74.32 ± 2.32	62.47 ± 1.08	95.06 ± 1.51	82.09 ± 6.85
H = **16**, N_H = **V**, **Flow = 16/0**	88.17 ± 0.16	19.81 ± 0.34	81.76 ± 1.19	68.45 ± 1.60	72.98 ± 1.93	71.08 ± 4.11	83.74 ± 6.25	86.85 ± 3.37
H = **16**, N_H = **V**, **Flow = 0/4**	88.24 ± 0.12	19.30 ± 0.25	83.57 ± 0.67	66.44 ± 3.12	76.16 ± 1.94	63.04 ± 2.90	99.10 ± 0.84	99.97 ± 0.02
H = **16**, N_H = **V**, **Flow = 0/8**	88.10 ± 0.20	19.32 ± 0.40	81.22 ± 1.29	71.88 ± 2.05	75.42 ± 0.96	66.06 ± 2.02	99.74 ± 0.18	99.99 ± 0.00
H = **32**, N_H = **I**, **Flow = 8/0**	10.05 ± 0.28	95.02 ± 0.03	27.35 ± 1.29	23.46 ± 1.45	35.30 ± 1.88	34.31 ± 1.47	97.06 ± 0.64	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 16/0**	83.94 ± 0.61	27.83 ± 1.22	81.71 ± 0.62	74.15 ± 2.21	75.62 ± 1.48	75.68 ± 4.15	99.94 ± 0.02	99.97 ± 0.03
H = **32**, N_H = **I**, **Flow = 0/4**	85.30 ± 0.86	24.67 ± 1.56	82.70 ± 1.54	66.68 ± 2.56	76.15 ± 1.88	62.01 ± 0.96	99.93 ± 0.02	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 0/8**	85.58 ± 0.32	24.04 ± 0.73	82.77 ± 1.10	57.03 ± 2.71	75.48 ± 1.25	56.44 ± 1.77	99.84 ± 0.11	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 8/0**	86.11 ± 0.41	23.61 ± 0.82	83.38 ± 0.71	52.99 ± 2.55	76.14 ± 1.43	60.70 ± 1.29	95.80 ± 2.55	95.44 ± 2.79
H = **32**, N_H = **II**, **Flow = 16/0**	86.28 ± 0.31	23.11 ± 0.54	83.00 ± 0.53	63.42 ± 3.35	73.87 ± 1.19	72.90 ± 3.19	97.87 ± 2.04	99.92 ± 0.05
H = **32**, N_H = **II**, **Flow = 0/4**	87.98 ± 0.12	19.62 ± 0.22	82.72 ± 0.77	58.53 ± 4.37	74.45 ± 1.96	57.56 ± 4.88	99.93 ± 0.04	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 0/8**	87.41 ± 0.45	20.73 ± 0.78	81.91 ± 1.56	63.56 ± 2.61	75.30 ± 0.85	58.86 ± 3.76	99.97 ± 0.01	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 8/0**	86.65 ± 0.08	22.53 ± 0.18	82.38 ± 0.45	51.04 ± 1.13	74.93 ± 0.97	62.11 ± 1.92	95.20 ± 2.24	92.17 ± 5.43
H = **32**, N_H = **III**, **Flow = 16/0**	85.71 ± 0.47	24.29 ± 0.88	82.65 ± 0.59	66.63 ± 3.54	76.55 ± 1.22	73.27 ± 1.09	98.27 ± 1.24	98.75 ± 1.14
H = **32**, N_H = **III**, **Flow = 0/4**	88.05 ± 0.17	19.54 ± 0.34	81.85 ± 0.94	63.19 ± 2.75	76.21 ± 0.84	61.44 ± 4.30	99.75 ± 0.21	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 0/8**	88.23 ± 0.37	19.48 ± 0.62	84.08 ± 1.42	64.08 ± 3.71	***78.26 ± 1.27**	64.08 ± 1.55	***99.98 ± 0.01**	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 8/0**	88.02 ± 0.19	19.78 ± 0.32	83.24 ± 0.65	50.51 ± 4.65	77.52 ± 1.20	60.08 ± 2.16	95.24 ± 2.10	91.41 ± 3.61
H = **32**, N_H = **IV**, **Flow = 16/0**	88.29 ± 0.14	19.36 ± 0.31	82.62 ± 1.72	59.43 ± 5.41	73.52 ± 1.33	59.20 ± 5.22	97.56 ± 1.00	99.57 ± 0.39
H = **32**, N_H = **IV**, **Flow = 0/4**	87.83 ± 0.14	19.92 ± 0.24	83.23 ± 0.71	57.29 ± 5.16	74.99 ± 1.71	65.64 ± 3.16	99.93 ± 0.04	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 0/8**	88.24 ± 0.28	***19.25 ± 0.44**	82.82 ± 1.17	64.69 ± 2.96	73.58 ± 2.12	60.43 ± 3.12	99.27 ± 0.43	99.99 ± 0.01
H = **32**, N_H = **V**, **Flow = 8/0**	88.25 ± 0.11	19.43 ± 0.25	83.02 ± 0.60	48.85 ± 3.13	73.55 ± 2.21	54.24 ± 3.53	94.02 ± 2.56	83.94 ± 5.81
H = **32**, N_H = **V**, **Flow = 16/0**	88.30 ± 0.17	19.41 ± 0.34	83.35 ± 1.37	64.63 ± 3.58	75.89 ± 2.13	72.53 ± 3.19	95.68 ± 0.75	95.89 ± 2.84
H = **32**, N_H = **V**, **Flow = 0/4**	***88.41 ± 0.15**	19.34 ± 0.26	84.03 ± 0.94	59.12 ± 3.95	74.31 ± 1.96	63.33 ± 1.89	99.73 ± 0.13	99.99 ± 0.00
H = **32**, N_H = **V**, **Flow = 0/8**	88.26 ± 0.10	19.28 ± 0.10	83.68 ± 0.69	60.66 ± 2.92	73.08 ± 1.57	61.10 ± 5.28	99.28 ± 0.28	***100.00 ± 0.00**

Table B.7: Bike Sharing (Normal $\mathcal{N}$) comparison (<latent dim> – <certainty budget> – <radial layers>/<MAF layers>). Bold and starred number indicate best score among all models.

	RMSE	Calibration	Winter Epist.	Spring Epist.	Autumn Epist.	OODom Epist.
H = **4**, N_H = **I**, **Flow = 8/0**	60.12 ± 4.17	21.15 ± 2.06	21.11 ± 2.59	14.97 ± 0.48	17.61 ± 1.02	***100.00 ± 0.00**
H = **4**, N_H = **I**, **Flow = 16/0**	53.68 ± 3.45	21.92 ± 1.65	26.80 ± 2.61	17.54 ± 0.87	17.96 ± 1.20	***100.00 ± 0.00**
H = **4**, N_H = **I**, **Flow = 0/4**	83.51 ± 8.56	23.94 ± 1.94	30.25 ± 6.17	17.49 ± 1.50	18.79 ± 1.30	***100.00 ± 0.00**
H = **4**, N_H = **I**, **Flow = 0/8**	75.16 ± 8.97	25.93 ± 1.01	29.86 ± 4.07	17.71 ± 0.78	17.98 ± 0.41	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 8/0**	53.90 ± 3.72	22.76 ± 0.93	24.59 ± 2.43	16.69 ± 0.76	18.30 ± 1.10	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 16/0**	56.08 ± 4.96	26.30 ± 1.32	26.93 ± 4.34	17.11 ± 1.06	20.41 ± 1.84	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 0/4**	65.03 ± 8.07	23.10 ± 1.68	27.99 ± 3.46	17.04 ± 1.06	22.55 ± 2.60	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 0/8**	74.25 ± 7.89	26.51 ± 1.09	31.66 ± 3.21	18.31 ± 0.90	20.35 ± 2.26	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 8/0**	51.68 ± 1.25	19.04 ± 1.75	25.06 ± 2.21	17.05 ± 0.47	17.44 ± 1.50	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 16/0**	55.23 ± 4.45	19.04 ± 1.13	27.77 ± 2.88	16.69 ± 0.60	18.74 ± 0.19	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 0/4**	79.25 ± 11.47	25.54 ± 0.94	26.66 ± 2.88	16.73 ± 0.92	19.28 ± 0.68	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 0/8**	92.99 ± 3.68	26.48 ± 0.41	28.84 ± 1.87	17.26 ± 0.43	22.35 ± 2.26	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 8/0**	51.71 ± 1.50	18.36 ± 2.24	23.72 ± 2.95	16.19 ± 0.75	16.77 ± 0.38	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 16/0**	60.81 ± 5.56	21.01 ± 1.35	22.02 ± 1.85	15.43 ± 0.48	17.31 ± 0.61	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 0/4**	57.74 ± 4.29	22.78 ± 1.02	26.75 ± 1.99	17.57 ± 0.36	18.85 ± 1.22	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 0/8**	78.64 ± 8.29	23.29 ± 1.91	34.19 ± 2.72	19.14 ± 0.86	20.46 ± 0.70	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 8/0**	53.74 ± 2.95	23.27 ± 1.51	30.83 ± 4.89	17.34 ± 0.25	20.26 ± 1.24	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 16/0**	56.01 ± 2.32	23.35 ± 0.94	27.03 ± 2.64	17.52 ± 1.04	17.07 ± 0.81	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 0/4**	83.87 ± 8.67	24.20 ± 2.03	27.64 ± 3.59	16.48 ± 0.78	21.24 ± 2.11	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 0/8**	90.87 ± 5.11	22.54 ± 0.99	30.61 ± 2.20	17.46 ± 0.72	19.84 ± 0.93	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 8/0**	58.52 ± 5.08	9.86 ± 3.10	22.85 ± 3.51	15.57 ± 1.36	19.51 ± 3.12	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 16/0**	58.33 ± 4.80	10.29 ± 4.02	38.20 ± 3.14	19.10 ± 0.91	21.55 ± 1.90	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 0/4**	52.02 ± 1.93	7.98 ± 2.43	48.51 ± 5.78	22.11 ± 1.76	31.46 ± 3.21	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 0/8**	59.43 ± 5.49	9.20 ± 3.32	55.93 ± 8.21	25.67 ± 2.12	32.76 ± 5.61	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 8/0**	58.02 ± 2.35	4.25 ± 1.32	23.71 ± 1.27	16.69 ± 0.69	17.95 ± 0.53	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 16/0**	58.92 ± 6.56	4.69 ± 1.23	33.81 ± 5.95	18.33 ± 1.66	21.59 ± 2.26	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 0/4**	51.87 ± 2.45	8.01 ± 1.89	48.39 ± 10.55	23.12 ± 3.23	25.23 ± 2.61	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 0/8**	58.00 ± 5.68	4.57 ± 1.15	51.38 ± 8.78	24.60 ± 2.22	26.31 ± 3.59	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 8/0**	60.73 ± 4.01	4.45 ± 0.89	30.21 ± 2.21	17.74 ± 0.74	20.59 ± 2.19	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 16/0**	59.87 ± 5.68	5.70 ± 1.47	36.65 ± 6.92	18.95 ± 1.61	21.12 ± 3.18	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 0/4**	58.80 ± 4.49	7.34 ± 2.08	56.72 ± 3.51	25.77 ± 1.72	27.21 ± 3.06	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 0/8**	54.08 ± 2.95	8.56 ± 2.70	55.37 ± 8.22	25.82 ± 2.32	31.20 ± 4.73	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 8/0**	52.49 ± 1.77	5.54 ± 1.22	33.26 ± 4.84	16.93 ± 1.23	22.32 ± 1.87	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 16/0**	53.29 ± 3.17	4.15 ± 1.67	30.48 ± 4.05	17.72 ± 1.15	20.31 ± 1.66	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 0/4**	56.11 ± 3.47	5.48 ± 1.63	54.10 ± 10.31	23.80 ± 3.44	25.87 ± 2.60	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 0/8**	56.03 ± 2.51	3.74 ± 1.59	55.41 ± 7.83	23.78 ± 2.41	30.82 ± 2.93	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 8/0**	55.11 ± 3.53	6.72 ± 1.17	39.26 ± 7.71	19.03 ± 1.41	23.03 ± 3.74	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 16/0**	57.19 ± 4.36	6.13 ± 1.77	30.47 ± 3.14	17.24 ± 1.30	22.40 ± 3.35	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 0/4**	54.33 ± 2.57	4.31 ± 1.03	40.28 ± 7.97	18.99 ± 1.47	24.85 ± 3.95	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 0/8**	55.97 ± 4.00	2.09 ± 0.55	57.11 ± 9.02	26.22 ± 3.44	25.91 ± 4.23	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 8/0**	55.82 ± 1.49	3.19 ± 0.78	33.14 ± 4.82	18.14 ± 1.42	19.36 ± 0.80	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 16/0**	60.20 ± 4.97	3.11 ± 0.78	40.89 ± 5.92	19.51 ± 1.50	19.08 ± 1.31	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 0/4**	59.49 ± 2.94	2.90 ± 0.54	40.57 ± 8.66	19.14 ± 2.25	26.81 ± 2.90	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 0/8**	61.86 ± 4.98	1.94 ± 0.33	***72.07 ± 8.34**	***28.46 ± 1.65**	32.84 ± 3.41	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 8/0**	57.46 ± 3.24	5.08 ± 1.96	30.09 ± 2.78	17.55 ± 0.67	18.25 ± 0.78	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 16/0**	55.34 ± 2.78	2.15 ± 0.41	35.14 ± 2.49	19.24 ± 1.00	20.53 ± 2.12	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 0/4**	58.50 ± 7.43	2.23 ± 0.29	57.94 ± 5.29	24.43 ± 0.60	29.14 ± 3.00	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 0/8**	54.68 ± 4.01	2.26 ± 0.60	45.24 ± 5.14	19.82 ± 1.48	29.27 ± 2.28	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 8/0**	53.35 ± 3.39	2.88 ± 0.81	32.52 ± 1.58	18.28 ± 1.29	20.08 ± 1.05	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 16/0**	54.91 ± 2.39	4.51 ± 0.92	35.21 ± 4.26	18.68 ± 1.00	19.81 ± 1.46	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 0/4**	58.84 ± 1.61	***1.49 ± 0.20**	50.90 ± 11.50	21.83 ± 2.72	33.23 ± 5.33	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 0/8**	***51.12 ± 1.93**	2.17 ± 0.35	55.22 ± 5.21	24.55 ± 2.43	28.35 ± 2.18	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 8/0**	52.58 ± 2.37	6.09 ± 1.30	28.88 ± 5.31	16.77 ± 1.24	20.28 ± 2.63	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 16/0**	58.58 ± 5.19	5.05 ± 1.43	28.78 ± 2.57	16.47 ± 0.83	21.19 ± 1.64	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 0/4**	51.88 ± 1.57	2.72 ± 0.74	55.32 ± 7.42	24.11 ± 2.10	***35.59 ± 3.99**	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 0/8**	53.84 ± 2.82	1.85 ± 0.33	49.44 ± 3.65	21.06 ± 2.11	30.14 ± 2.12	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 8/0**	53.52 ± 1.64	8.24 ± 0.64	30.51 ± 2.22	18.50 ± 0.67	20.98 ± 2.00	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 16/0**	58.21 ± 9.11	6.91 ± 1.16	35.75 ± 3.59	18.07 ± 0.91	19.09 ± 1.29	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 0/4**	56.22 ± 4.42	1.98 ± 0.45	40.97 ± 5.06	20.21 ± 1.98	23.40 ± 1.69	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 0/8**	54.06 ± 3.18	2.09 ± 0.32	35.75 ± 7.51	18.86 ± 2.13	22.27 ± 1.83	***100.00 ± 0.00**

Table B.8: Bike Sharing (Poisson Poi) comparison (<latent dim> – <certainty budget> – <radial layers>/<MAF layers>). Bold and starred number indicate best score among all models.

	RMSE	Winter Epist.	Spring Epist.	Autumn Epist.	OODom Epist.
H = **4**, N_H = **I**, **Flow = 8/0**	937.56 ± 238.40	33.50 ± 4.83	18.04 ± 0.58	20.44 ± 1.53	***100.00 ± 0.00**
H = **4**, N_H = **I**, **Flow = 16/0**	777.22 ± 328.71	22.53 ± 2.16	17.00 ± 0.74	18.30 ± 1.25	***100.00 ± 0.00**
H = **4**, N_H = **I**, **Flow = 0/4**	33780.69 ± 19607.73	45.22 ± 7.96	21.35 ± 2.09	37.89 ± 4.13	***100.00 ± 0.00**
H = **4**, N_H = **I**, **Flow = 0/8**	22686.67 ± 5815.18	53.51 ± 5.83	21.80 ± 1.14	39.25 ± 2.59	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 8/0**	60383.07 ± 21389.07	15.20 ± 0.57	13.50 ± 0.22	15.28 ± 0.77	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 16/0**	32982.63 ± 17819.36	33.09 ± 8.79	17.55 ± 2.25	19.17 ± 1.19	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 0/4**	21467.31 ± 17019.59	44.60 ± 8.61	20.27 ± 1.97	33.25 ± 8.69	***100.00 ± 0.00**
H = **4**, N_H = **II**, **Flow = 0/8**	11231.30 ± 4784.62	42.64 ± 7.58	21.06 ± 2.63	23.93 ± 2.05	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 8/0**	2048.78 ± 538.12	22.84 ± 1.90	16.59 ± 0.73	17.21 ± 0.50	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 16/0**	5181.92 ± 3581.91	25.42 ± 2.91	15.24 ± 0.53	17.00 ± 1.34	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 0/4**	35092.52 ± 10813.54	47.42 ± 5.98	22.49 ± 1.88	28.50 ± 1.52	***100.00 ± 0.00**
H = **4**, N_H = **III**, **Flow = 0/8**	86946.86 ± 42792.69	53.87 ± 7.18	22.87 ± 2.17	33.78 ± 5.12	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 8/0**	10255.94 ± 6207.90	19.89 ± 1.82	14.93 ± 0.30	16.38 ± 0.57	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 16/0**	6665.70 ± 3160.53	29.95 ± 6.24	17.66 ± 1.06	17.33 ± 0.46	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 0/4**	119600.14 ± 85229.53	35.15 ± 5.20	19.58 ± 2.37	27.15 ± 2.87	***100.00 ± 0.00**
H = **4**, N_H = **IV**, **Flow = 0/8**	132950.39 ± 98199.93	56.85 ± 8.13	24.17 ± 1.99	40.23 ± 5.69	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 8/0**	131051.70 ± 124947.53	23.89 ± 2.55	15.68 ± 0.71	17.94 ± 1.34	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 16/0**	16481.96 ± 7339.53	26.55 ± 1.59	16.40 ± 0.25	26.36 ± 5.15	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 0/4**	28238.41 ± 10202.38	39.68 ± 9.73	19.04 ± 2.13	29.09 ± 3.57	***100.00 ± 0.00**
H = **4**, N_H = **V**, **Flow = 0/8**	27167.10 ± 9698.30	46.59 ± 6.57	23.03 ± 1.22	27.77 ± 5.03	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 8/0**	633.14 ± 237.64	35.20 ± 5.92	18.16 ± 0.91	22.16 ± 2.43	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 16/0**	408.28 ± 246.12	45.32 ± 4.53	19.96 ± 1.66	34.91 ± 6.35	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 0/4**	276.23 ± 151.72	64.86 ± 8.92	26.40 ± 3.85	37.05 ± 5.43	***100.00 ± 0.00**
H = **16**, N_H = **I**, **Flow = 0/8**	262.91 ± 126.12	80.30 ± 4.85	32.12 ± 2.86	38.83 ± 3.14	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 8/0**	1325.94 ± 79.27	30.06 ± 3.95	18.63 ± 1.33	21.43 ± 1.82	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 16/0**	1042.48 ± 413.69	45.96 ± 5.13	20.63 ± 1.63	24.25 ± 1.57	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 0/4**	129.87 ± 79.26	71.03 ± 5.39	34.02 ± 4.65	36.16 ± 5.75	***100.00 ± 0.00**
H = **16**, N_H = **II**, **Flow = 0/8**	182.97 ± 129.05	81.19 ± 6.60	36.17 ± 5.02	***43.97 ± 7.12**	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 8/0**	1233.19 ± 76.60	34.22 ± 4.42	19.09 ± 1.47	25.34 ± 1.63	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 16/0**	881.87 ± 367.31	38.58 ± 3.13	21.66 ± 1.47	22.85 ± 1.57	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 0/4**	89.12 ± 37.11	84.06 ± 3.99	37.00 ± 3.23	36.77 ± 5.20	***100.00 ± 0.00**
H = **16**, N_H = **III**, **Flow = 0/8**	93.06 ± 28.11	76.50 ± 6.64	30.94 ± 5.14	37.96 ± 5.69	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 8/0**	1893.36 ± 970.62	40.06 ± 6.68	20.01 ± 1.96	25.71 ± 4.91	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 16/0**	2212.13 ± 1084.13	41.60 ± 4.41	17.93 ± 0.92	29.77 ± 3.60	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 0/4**	56.60 ± 2.25	72.66 ± 6.46	32.09 ± 4.74	34.56 ± 5.52	***100.00 ± 0.00**
H = **16**, N_H = **IV**, **Flow = 0/8**	52.01 ± 2.10	79.58 ± 5.81	33.47 ± 2.84	36.19 ± 4.91	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 8/0**	4434.32 ± 3059.38	31.63 ± 4.31	17.82 ± 1.02	20.77 ± 2.04	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 16/0**	4115.67 ± 1891.56	47.58 ± 4.69	22.82 ± 2.46	26.51 ± 5.27	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 0/4**	50.47 ± 1.54	83.71 ± 5.23	***37.46 ± 5.13**	42.63 ± 4.37	***100.00 ± 0.00**
H = **16**, N_H = **V**, **Flow = 0/8**	51.79 ± 0.78	***85.15 ± 3.61**	37.03 ± 2.35	42.73 ± 4.38	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 8/0**	351.49 ± 157.14	38.59 ± 5.39	21.90 ± 2.62	25.23 ± 2.66	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 16/0**	167.67 ± 116.18	45.10 ± 6.51	21.90 ± 2.76	24.84 ± 3.40	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 0/4**	50.10 ± 1.55	73.09 ± 9.70	27.10 ± 2.42	40.78 ± 8.19	***100.00 ± 0.00**
H = **32**, N_H = **I**, **Flow = 0/8**	51.97 ± 2.57	58.80 ± 10.68	23.64 ± 2.46	30.77 ± 5.51	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 8/0**	580.40 ± 250.82	38.80 ± 5.56	20.62 ± 1.89	25.80 ± 1.92	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 16/0**	49.96 ± 1.60	46.52 ± 6.52	22.78 ± 1.69	27.16 ± 5.09	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 0/4**	***48.85 ± 0.92**	60.12 ± 10.37	22.57 ± 2.44	37.51 ± 6.70	***100.00 ± 0.00**
H = **32**, N_H = **II**, **Flow = 0/8**	50.12 ± 2.29	69.33 ± 4.57	30.96 ± 2.55	35.11 ± 6.33	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 8/0**	462.85 ± 169.59	43.86 ± 7.00	20.62 ± 2.41	30.58 ± 4.59	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 16/0**	569.28 ± 219.42	54.12 ± 8.10	22.49 ± 1.98	31.49 ± 4.90	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 0/4**	49.83 ± 1.25	67.93 ± 9.50	27.61 ± 3.98	31.87 ± 6.17	***100.00 ± 0.00**
H = **32**, N_H = **III**, **Flow = 0/8**	51.26 ± 1.53	70.68 ± 5.97	32.89 ± 3.07	28.56 ± 5.79	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 8/0**	50.79 ± 1.07	42.61 ± 8.46	18.84 ± 2.13	26.45 ± 3.68	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 16/0**	49.91 ± 1.54	45.15 ± 7.53	23.00 ± 2.26	27.90 ± 3.31	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 0/4**	49.82 ± 1.59	59.64 ± 7.63	26.64 ± 4.29	34.17 ± 7.65	***100.00 ± 0.00**
H = **32**, N_H = **IV**, **Flow = 0/8**	51.66 ± 1.79	65.31 ± 9.06	28.34 ± 3.83	38.84 ± 6.46	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 8/0**	52.95 ± 1.36	39.37 ± 7.16	17.74 ± 1.36	28.73 ± 5.78	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 16/0**	146.99 ± 94.70	56.50 ± 3.61	24.44 ± 2.00	34.28 ± 4.16	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 0/4**	51.42 ± 1.68	74.23 ± 7.35	27.64 ± 1.85	35.90 ± 6.06	***100.00 ± 0.00**
H = **32**, N_H = **V**, **Flow = 0/8**	49.31 ± 1.81	76.36 ± 10.30	31.86 ± 4.07	41.32 ± 4.75	***100.00 ± 0.00**

Table B.9: MNIST - OOD detection with AUC-ROC scores. Bold numbers indicate best score among single-pass models. Starred numbers indicate best scores among all models. Gray numbers indicate that R-PriorNet has seen samples from the Fashion-MNIST dataset during training.

	K. Alea.	K. Epist.	F. Alea.	F. Epist.	OODom Alea.	OODom Epist.
Dropout	98.12 ± 0.05	97.16 ± 0.11	*99.26 ± 0.03	96.87 ± 0.25	15.19 ± 1.70	88.16 ± 0.59
Ensemble	98.17 ± 0.07	98.03 ± 0.05	99.15 ± 0.07	98.04 ± 0.11	11.83 ± 1.81	81.53 ± 0.38
NatPE	98.25 ± 0.27	99.48 ± 0.03	98.79 ± 0.33	*99.61 ± 0.07	*100.00 ± 0.00	*100.00 ± 0.00
R-PriorNet	***99.44 ± 0.09**	***99.59 ± 0.08**	100.00 ± 0.00	100.00 ± 0.00	99.44 ± 0.16	1.82 ± 0.67
EnD^2	98.21 ± 0.16	98.65 ± 0.15	99.06 ± 0.20	99.21 ± 0.17	56.31 ± 2.78	4.60 ± 1.89
PostNet	98.73 ± 0.05	98.62 ± 0.06	98.65 ± 0.35	98.57 ± 0.34	***100.00 ± 0.00**	***100.00 ± 0.00**
NatPN	99.11 ± 0.17	99.25 ± 0.09	**99.13 ± 0.24**	**99.45 ± 0.11**	99.98 ± 0.01	***100.00 ± 0.00**

Table B.10: CIFAR-10 - OOD detection with AUC-ROC scores. Bold numbers indicate best score among single-pass models. Starred numbers indicate best scores among all models. Gray numbers indicate that R-PriorNet has seen samples from the SVHN dataset during training.

	SVHN Alea.	SVHN Epist.	CelebA Alea.	CelebA Epist.	OODom Alea.	OODom Epist.
Dropout	84.67 ± 1.42	75.79 ± 0.86	75.95 ± 3.61	75.00 ± 3.21	21.75 ± 6.36	78.81 ± 6.91
Ensemble	88.08 ± 0.85	85.70 ± 0.70	78.80 ± 0.82	77.63 ± 0.61	40.53 ± 9.95	96.71 ± 2.31
NatPE	88.73 ± 0.26	*86.73 ± 0.82	*80.46 ± 0.82	*85.75 ± 1.09	92.45 ± 1.37	*99.56 ± 0.11
R-PriorNet	99.94 ± 0.01	99.98 ± 0.00	74.69 ± 2.39	70.63 ± 6.14	64.45 ± 10.72	59.61 ± 13.23
EnD^2	***89.56 ± 0.67**	**84.36 ± 0.84**	77.94 ± 1.62	78.14 ± 1.66	53.05 ± 6.07	4.42 ± 2.57
PostNet	85.52 ± 0.58	84.25 ± 0.90	75.68 ± 2.05	77.96 ± 2.05	93.00 ± 2.46	97.22 ± 0.86
NatPN	85.24 ± 0.98	81.74 ± 1.05	**77.98 ± 1.22**	**81.62 ± 3.15**	***96.94 ± 1.53**	**97.41 ± 1.63**

Table B.11: Bike Sharing - OOD detection with AUC-ROC scores. Bold numbers indicate best score among single-pass models. Starred numbers indicate best scores among all models. Normal and Poisson Regression are treated separately.

	Winter Epist.	Spring Epist.	Autumn Epist.	OODom Epist.
Dropout-$\mathcal{N}$	53.98 ± 1.60	51.24 ± 0.96	53.89 ± 1.16	*100.00 ± 0.00
Ensemble-$\mathcal{N}$	81.53 ± 1.11	67.07 ± 0.55	67.78 ± 1.31	*100.00 ± 0.00
EvReg-$\mathcal{N}$	55.26 ± 2.14	53.76 ± 1.35	52.39 ± 1.31	47.68 ± 17.67
NatPN-$\mathcal{N}$	***87.67 ± 3.13**	***68.68 ± 2.58**	***71.70 ± 3.23**	***100.00 ± 0.00**
Dropout-Poi	55.30 ± 0.58	50.75 ± 0.56	59.05 ± 1.15	***100.00 ± 0.00**
Ensemble-Poi	95.31 ± 0.41	75.62 ± 0.85	78.93 ± 1.35	***100.00 ± 0.00**
NatPN-Poi	***96.67 ± 1.02**	***78.45 ± 2.58**	***82.42 ± 1.73**	***100.00 ± 0.00**

C Practicality of Uncertainty Estimation

C.0.1 Deterministic Uncertainty Methods

NatPN. The deep Bayesian uncertainty model NatPN [68] can be decomposed into these steps: **(1)** a core architecture predicts one latent representation of the input $\boldsymbol{x}^{(i)}$ i.e. $\boldsymbol{z}^{(i)} = f_{\boldsymbol{\phi}}(\boldsymbol{x}^{(i)}) \in \mathbb{R}^H$, **(2)** While a density estimator $\mathbb{P}(.|\boldsymbol{w})$ predicts the evidence parameter update $n^{(i)} = N_H \mathbb{P}(z^{(i)}|\boldsymbol{w})$ where the N_H is a scaling factor named certainty budget, a single linear decoder $g_{\boldsymbol{\psi}}$ outputs the parameter update $\boldsymbol{\chi}^{(i)} = g_{\boldsymbol{\psi}}(\boldsymbol{z}^{(i)}) \in \mathbb{R}^L$, which can be viewed as a softmax output prediction. **(3)** We perform an input-dependent Bayesian update which can be expressed in a closed-form as:

$$\mathbb{Q}(\boldsymbol{\theta}^{(i)}|\boldsymbol{\chi}^{\text{post},(i)}, n^{\text{post},(i)}) = \exp(n^{\text{post},(i)}\boldsymbol{\theta}^{(i)T}\boldsymbol{\chi}^{\text{post},(i)} - n^{\text{post},(i)}A(\boldsymbol{\theta}^{(i)}))$$

$$\text{where} \qquad \boldsymbol{\chi}^{\text{post},(i)} = \frac{n^{\text{prior}}\boldsymbol{\chi}^{\text{prior}} + n^{(i)}\boldsymbol{\chi}^{(i)}}{n^{\text{prior}} + n^{(i)}}, \quad n^{\text{post},(i)} = n^{\text{prior}} + n^{(i)}$$

where $\boldsymbol{\chi}^{\text{prior}}, n^{\text{prior}}$ are fixed prior parameters, and $\boldsymbol{\chi}^{\text{post},(i)}, n^{\text{post},(i)}$ are the input-dependent posterior parameters. For the classification, the variable $\boldsymbol{\theta}^{(i)}$ represents the normalized categorical vector $\boldsymbol{p}^{(i)}$. The predictive uncertainty is computed via the entropy of the predictive categorical distribution, and the epistemic uncertainty is computed via the evidence parameter $n^{post,(i)}$. We train all the components of neural network parameters $\{\boldsymbol{\phi}, \boldsymbol{w}, \boldsymbol{\psi}\}$ jointly with the Bayesian loss [68]:

$$\mathcal{L}^{(i)} \propto \mathbb{E}[\boldsymbol{\theta}^{(i)}]^T u(y^{(i)}) - \mathbb{E}[A(\boldsymbol{\theta}^{(i)})] - \lambda \mathbb{H}[\mathbb{Q}^{post,(i)}] \tag{C.1}$$

where λ is the regularization factor of the entropy term representing. We refer to [68] for a more detailed description of the method.

DUE. The deep kernel learning method DUE [421] can be decomposed into these steps: **(1)** a core architecture predicts one latent representation of the input $\boldsymbol{x}^{(i)}$ i.e. $\boldsymbol{z}^{(i)} = f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \in \mathbb{R}^H$, **(2)** a Gaussian Process defined from a fixed set of K learnable inducing points $\{\boldsymbol{\phi}_k\}_{k=1}^K$ and a predefined positive definite kernel $\kappa(\cdot, \cdot)$ predicts the mean $\mu(\boldsymbol{x}^{(i)})$ and the variance $\sigma(\boldsymbol{x}^{(i)})$ of a Gaussian distribution, and **(3)** we apply softmax to the mean output $\mu(\boldsymbol{x}^{(i)})$ for the classification prediction, i.e. $\boldsymbol{p}^{(i)} = \text{softmax}(\mu(\boldsymbol{x}^{(i)})$. We train the neural network parameters $\boldsymbol{\theta}$ and the inducing points $\{\boldsymbol{\phi}_k\}_{k=1}^K$ jointly with a variational ELBO loss. For classification, the predictive uncertainty is computed as the entropy of the predictive categorical distribution. We refer to [421] for a more detailed description of the method.

C.0.2 Dataset details

We split all the training datasets into train, validation and test sets. For all the datasets, the test set is fixed while the training/validation sets are split in 80/20% respectively. The random split of training/validation sets change depending on the seeds to ensure more diversity.

MNIST [249]. Image classification dataset. Similarly as in [16], we create the CMNIST dataset for domain generalization experiments by expanding the input's size to 3 x 28 x 28 and zeroing one of the three channels. For OOD detection we use the test set of MNIST as ID dataset and compare to: KMNIST [83], CIFAR10, CMNIST, and KMNIST OODom, where we scale the input by 255. The batch size used is 512.

CIFAR [238]. Image classification dataset. We apply two data augmentations methods to the training data:the random horizontal flip and random cropping with padding equal to 4. For domain generalization we use the corrupted version CIFAR-C [183] and report the average metric of 15 corruptions for the level of corruption severity of 1. For OOD detection we use the test set of CIFAR10 as ID dataset and compare to: SVHN [313], STL10 [86], CelebA [262], Camelyon (Test OOD), and SVHN OODom. Since the Camelyon (Test OOD) dataset is large (85k), we extract only 10k subset of images as the OOD datasets. The batch size used is 128.

Camelyon [233]. Image classification dataset. We apply two data augmentations emthods to the training data: random horizontal flip and random rotation of 15 degrees. For domain generalization the dataset already provide the distribution shifted validation and test splits. For OOD detection we use the ID validation set of Camelyon as ID dataset (the ID test set is not available) and compare to: SVHN, STL10, CelebA, Camelyon (Test OOD), and SVHN OODom. The batch size used is 32.

Each OOD dataset is rescaled to the same size as the ID dataset and normalized with zero mean and unit variance based on the statistics of ID dataset (for the Camelyon dataset we don't apply any normalization as in [233]).

C.0.3 Metric details

Accuracy. The standard accuracy $\frac{1}{N}\sum_i \mathbb{1}[y^{*,(i)} = y^{(i)}]$ is used, where $y^{*,(i)}$ is the true label and $y^{(i)}$ is the predicted label.

Calibration. The Brier score $\frac{1}{C}\sum_i^N ||\boldsymbol{p}^{(i)} - \boldsymbol{y}^{*,(i)}||$ is used, where $\boldsymbol{p}^{(i)}$ is the predicted softmax probability and $\boldsymbol{y}^{*,(i)}$ is the one-hot encoded true label. Lower calibration indicates a better calibrated model. Note that in constrast with the Expected Calibration Error (ECE), the Brier score is a strictly proper scoring rule which makes it a particularly good evaluation metric for calibration [161].

OOD Generalization. We apply accuracy and calibration to the distribution shifted OOD dataset and compare the results with the ID dataset to estimate the model's ability for generalization.

OOD Detection. We treat this task as a binary classification, where we assign class 1 to ID data and class 0 to OOD data using the aleatoric, epistemic, and predictive uncertainty estimates as scores for OOD data. This allows to compute the final scores

using the area under the receiver operating characteristic curve (AUC-ROC) to measure the model's ability to detect OOD data.

C.0.4 Model details

Core architecture. We use the same feature extractor for both the DUMs architecture. The list of core architectures used across the experiments are: *ResNet18 / ResNet50 / EfficientNet / Swin* [181, 405, 260] from the torchvision repository [1] and *Wide-ResNet-28-10* [460] from the original implementation of DUE. Except for the experiment on architecture type and size where ResNet18 has output channels for the residual blocks with size $[64, 128, 256, 512]$, ResNet18 has output channels for the residual blocks with size $[32, 64, 128, 256]$ which causes small differences in final accuracy.

Uncertainty head. For DUE we use the original implementation [2] with by default we use the RBF kernel function. For NatPN we use the original implementation [3] but change the uncertainty head with a more expressive density estimator. As seen in Table C.1, we found that a more expressive normalizing flow with resampled base [113, 399] improves significantly the results over a simpler radial normalizing flow [358] across all the metrics. For all the experiments (except toys where we use radial flow) we use NSF-R with 16 layers.

Table C.1: Normalizing flow expressivity comparison. Using more expressive normalizing flow significantely improves all the results for NatPN.

	CIFAR100				Camelyon			
Head	Accuracy (↑)	Brier Score (↓)	OOD Pred. (↑)	OOD Epis. (↑)	Accuracy (↑)	Brier Score (↓)	OOD Pred. (↑)	OOD Epis. (↑)
Radial	71.09 ± 0.21	52.27 ± 0.28	72.84 ± 1.82	50.95 ± 2.16	83.14 ± 0.93	24.55 ± 1.91	60.27 ± 5.29	69.16 ± 7.94
NSF-R	**71.61 ± 0.07**	**43.44 ± 0.11**	**73.54 ± 1.69**	**72.85 ± 1.25**	**89.84 ± 7.93**	**12.52 ± 6.17**	**64.14 ± 10.42**	**81.33 ± 8.78**

C.1 Training for DUMs Details

By default, we first start by only pretraining the core encoder architecture using the cross-entropy loss, before attaching the DUM uncertainty head to the pretrained encoder and continue with the joint training phase. We do not pretrain the encoder for MNIST. we provide further details in Table C.2. Following the original method in [68], we train the NatPN uncertainty head before (*warmup*) and after (*finetune*) the joint training. In the warmup phase, we use the lambda scheduler increasing linearly from zero to LR head value in Table C.2. In the finetune phase, we use a multistep scheduler that scales the learning rate by 0.2 at 70% and 90% of the training starting from the LR head value in Table C.2. We warmup for 0/5/0 and finetune for 60/200/5 epochs for the datasets MNIST/CIFAR/Camelyon respectively.

Decoupling learning rate. In this experiment we use different values for the learning rates of the core architecture and of the uncertainty head. After the decoupling learning

[1] `https://pytorch.org/vision/stable/models.html`
[2] `https://github.com/y0ast/DUE`
[3] `https://github.com/borchero/natural-posterior-network`

Table C.2: Default training hyperparameters. For CIFAR10, CIFAR100 and Camelyon we first pretrain a core encoder architecture for the join training phase. In MNIST we directly joint train given its lower computational cost.

Dataset	Phase	Encoder	Epochs	Optimizer Enc. / Head	LR Enc. / Head	LR scheduler Enc. / Head	Weight decay Enc. / Head	Latent Dimension
MNIST	Pretrain	-	-	-	-	-	-	-
	Joint train (DUE)	ResNet18	20	AdamW / AdamW	1e-3 / 1e-4	cosine η_{min}=5e-4 / -	1e-6 / 1e-6	16
	Joint train (NatPN)	ResNet18	20	AdamW / AdamW	1e-3 / 5e-3	cosine η_{min}=5e-4 / -	1e-6 / 1e-6	16
CIFAR10 & CIFAR100	Pretrain	ResNet18	200	SGD	1e-1	cosine η_{min}=5e-4	5e-4	-
	Joint train (DUE)	ResNet18	20	AdamW / AdamW	1e-4 / 1e-4	cosine η_{min}=1e-5 / -	1e-6 / 1e-6	64
	Joint train (NatPN)	ResNet18	20	AdamW / AdamW	1e-5 / 5e-3	cosine η_{min}=1e-5 / -	1e-6 / 1e-6	64
Camelyon	Pretrain	WideResNet28-10	5	AdamW	1e-3	cosine η_{min}=1e-5	1e-8	-
	Joint train (DUE)	WideResNet28-10	1	AdamW / AdamW	1e-5 / 5e-3	cosine η_{min}=1e-6 / -	1e-6 / 1e-6	128
	Joint train (NatPN)	WideResNet28-10	1	AdamW / AdamW	5e-6 / 1e-5	cosine η_{min}=1e-6 / -	1e-6 / 1e-6	128

rate experiment, we choose the best combination of learning rates through model selection via the validation results and apply it to other experiments. E.g., for the joint training schema and pretraining schema experiments, NatPN uses a learning rate of 1e-4/1e-4 for encoder/head respectively, while for DUE it is 1e-5/5e-3.

Training schemes. In this experiment, we compare the *joint* training in which we jointly train the weights of the core architecture and uncertainty head, and the *sequential* training in which we only train the uncertainty head by keeping the weights of the pretrained core architecture fixed. For each of the setting, we apply two additional techniques to stabilize the training: adding a *batch normalization* to the last layer of the encoder to enforce latent representations to locate in a normalized region [204, 68], and *resetting the last layer* to retrain its weights to improve robustness to spurious correlation [225].

Pretraining schemes. In this experiment, we do not pretrain the core encoder architecture or pretrain it with 10% of CIFAR100, 100% of CIFAR100, and ImageNet. For the schemes *None* and *C100 (10%)* which use no or few pretraining data, we increase the joint training phase to 200 epochs with for the core architecture to ensure proper convergence.

C.2 Architecture for DUMs Details

For all the architecture experiments we used the default training settings in Appendix C.1 and Table C.2.

Bi-Lipschitz training details. In the *None* configuration, we removed the residual connection from the architecture for both the pretraining of the encoder and the joint training phase. In the *Residual* configuration, we did not modify anything since both ResNet and WideResNet already use the residual connection. While for the *bi-Lipschitz* configuration, we added the spectral normalization during both the pretraining and also joint training phase. Following the original method presented in [421], we used the same implementation and applied spectral normalization to the linear, convolution, and batch normalization layers. During the model selection, using the validation set results, we find that the best *Lipschitz constant c* for DUE is 4, and for NatPN is 5. For both the models the power iteration parameter is set to 1. For the toy dataset we use an encoder with 4 linear layers of 128 dimension each.

Table C.3: Train schema OOD detection. Uncertainty estimation results broken down for each OOD dataset. We observe that NatPN performs significantly better when using *joint*, while DUE is insensitive to the schema used. The ID dataset used for training is CIFAR100.

Model	OOD Data	Train Schema	OOD Alea. (↑)	OOD Epis. (↑)	OOD Pred. (↑)
NatPN	SVHN	joint	80.64 ± 1.22	65.00 ± 3.18	80.64 ± 1.22
		joint + bn	**82.07 ± 0.62**	69.98 ± 4.40	**82.07 ± 0.62**
		joint + reset	80.77 ± 1.63	**74.67 ± 2.37**	80.77 ± 1.63
		sequential	80.76 ± 0.95	43.99 ± 4.16	80.76 ± 0.95
		sequential + bn	80.64 ± 0.83	44.46 ± 5.63	80.64 ± 0.83
		sequential + reset	78.92 ± 1.06	59.54 ± 4.25	78.92 ± 1.06
	STL10	joint	76.63 ± 0.20	58.79 ± 0.24	76.63 ± 0.20
		joint + bn	76.53 ± 0.22	63.51 ± 0.38	76.53 ± 0.22
		joint + reset	76.92 ± 0.36	**64.44 ± 0.52**	76.92 ± 0.36
		sequential	77.57 ± 0.20	42.83 ± 0.62	77.57 ± 0.20
		sequential + bn	**77.64 ± 0.22**	44.26 ± 0.59	**77.64 ± 0.22**
		sequential + reset	77.35 ± 0.15	57.56 ± 0.38	77.35 ± 0.15
	CelebA	joint	51.42 ± 1.29	28.90 ± 1.59	51.42 ± 1.29
		joint + bn	51.45 ± 1.15	27.67 ± 2.45	51.45 ± 1.15
		joint + reset	52.01 ± 0.46	**32.54 ± 0.32**	52.01 ± 0.46
		sequential	52.58 ± 1.08	24.41 ± 1.94	52.58 ± 1.08
		sequential + bn	52.44 ± 1.11	23.65 ± 1.85	52.44 ± 1.11
		sequential + reset	**53.12 ± 0.92**	26.82 ± 1.41	**53.12 ± 0.92**
	Camelyon	joint	**67.17 ± 5.30**	67.03 ± 9.00	**67.17 ± 5.30**
		joint + bn	61.06 ± 2.70	69.69 ± 5.54	61.06 ± 2.70
		joint + reset	67.07 ± 1.12	**73.45 ± 4.22**	67.07 ± 1.12
		sequential	64.54 ± 2.08	56.23 ± 6.07	64.54 ± 2.08
		sequential + bn	64.34 ± 2.12	49.31 ± 5.99	64.34 ± 2.12
		sequential + reset	63.76 ± 2.13	65.59 ± 4.65	63.76 ± 2.13
	SVHN OODom.	joint	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		joint + bn	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		joint + reset	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		sequential	99.99 ± 0.00	100.00 ± 0.00	99.99 ± 0.00
		sequential + bn	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		sequential + reset	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
DUE	SVHN	joint	-	-	80.75 ± 0.79
		joint + bn	-	-	80.47 ± 1.09
		joint + reset	-	-	80.92 ± 0.70
		sequential	-	-	81.04 ± 0.82
		sequential + bn	-	-	**81.20 ± 0.86**
		sequential + reset	-	-	80.66 ± 1.07
	STL10	joint	-	-	77.20 ± 0.19
		joint + bn	-	-	77.24 ± 0.22
		joint + reset	-	-	77.42 ± 0.25
		sequential	-	-	77.50 ± 0.09
		sequential + bn	-	-	**77.52 ± 0.13**
		sequential + reset	-	-	77.48 ± 0.17
	CelebA	joint	-	-	47.99 ± 1.25
		joint + bn	-	-	47.90 ± 1.13
		joint + reset	-	-	49.10 ± 0.86
		sequential	-	-	48.07 ± 0.99
		sequential + bn	-	-	48.43 ± 1.06
		sequential + reset	-	-	**49.39 ± 1.29**
	Camelyon	joint	-	-	67.76 ± 2.20
		joint + bn	-	-	67.54 ± 2.33
		joint + reset	-	-	67.03 ± 2.00
		sequential	-	-	67.49 ± 2.58
		sequential + bn	-	-	67.23 ± 2.72
		sequential + reset	-	-	**67.85 ± 2.54**
	SVHN OODom.	joint	-	-	100.00 ± 0.00
		joint + bn	-	-	100.00 ± 0.00
		joint + reset	-	-	100.00 ± 0.00
		sequential	-	-	100.00 ± 0.00
		sequential + bn	-	-	100.00 ± 0.00
		sequential + reset	-	-	100.00 ± 0.00

Reconstruction training details. The decoder reconstructs the input extracted from the last residual block of the encoder, before the pooling layer. During the pre-

Table C.4: Pretrain schema OOD detection. Uncertainty estimation results broken down for each OOD dataset. We see how ImageNet pretrained encoder consistently performs better than other settings for 4/5 OOD datasets. The ID dataset used in the joint training phase is CIFAR100.

Model	OOD Data	Pretrain Schema	OOD Alea. (↑)	OOD Epis. (↑)	OOD Pred. (↑)
NatPN	SVHN	None	79.95 ± 1.18	77.87 ± 2.54	79.95 ± 1.18
		C100 (10%)	71.81 ± 1.67	75.16 ± 2.20	71.81 ± 1.67
		C100 (100%) + $\mathcal{N}(0.5)$	80.76 ± 1.03	61.15 ± 6.04	80.76 ± 1.03
		C100 (100%) + $\mathcal{N}(0.1)$	79.86 ± 1.81	60.50 ± 7.54	79.86 ± 1.81
		C100 (100%)	81.76 ± 1.38	65.72 ± 4.91	81.76 ± 1.38
		ImageNet	**89.34 ± 0.66**	**92.02 ± 0.49**	**89.34 ± 0.66**
	STL10	None	80.34 ± 0.77	78.05 ± 2.37	80.34 ± 0.77
		C100 (10%)	74.64 ± 0.72	62.43 ± 2.92	74.64 ± 0.72
		C100 (100%) + $\mathcal{N}(0.5)$	79.92 ± 0.32	56.81 ± 3.69	79.92 ± 0.32
		C100 (100%) + $\mathcal{N}(0.1)$	80.63 ± 0.94	57.72 ± 3.93	80.63 ± 0.94
		C100 (100%)	80.70 ± 1.33	66.07 ± 2.60	80.70 ± 1.33
		ImageNet	**85.46 ± 0.50**	**90.76 ± 0.31**	**85.46 ± 0.50**
	CelebA	None	73.59 ± 3.78	**76.19 ± 3.32**	73.59 ± 3.78
		C100 (10%)	73.26 ± 1.79	63.31 ± 3.66	73.26 ± 1.79
		C100 (100%) + $\mathcal{N}(0.5)$	66.88 ± 2.72	49.00 ± 3.23	66.88 ± 2.72
		C100 (100%) + $\mathcal{N}(0.1)$	73.15 ± 0.92	46.91 ± 6.50	73.15 ± 0.92
		C100 (100%)	**73.61 ± 2.16**	58.11 ± 2.85	**73.61 ± 2.16**
		ImageNet	59.30 ± 3.77	64.80 ± 2.15	59.30 ± 3.77
	Camelyon	None	65.37 ± 10.77	96.95 ± 2.01	65.37 ± 10.77
		C100 (10%)	20.84 ± 5.27	91.58 ± 7.56	20.84 ± 5.27
		C100 (100%) + $\mathcal{N}(0.5)$	40.19 ± 5.07	76.33 ± 5.84	40.19 ± 5.07
		C100 (100%) + $\mathcal{N}(0.1)$	45.39 ± 10.58	83.79 ± 5.27	45.39 ± 10.58
		C100 (100%)	58.69 ± 11.10	94.63 ± 2.85	58.69 ± 11.10
		ImageNet	**90.64 ± 2.47**	**97.82 ± 0.56**	**90.64 ± 2.47**
	SVHN OODom.	None	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		C100 (10%)	99.96 ± 0.04	100.00 ± 0.00	99.96 ± 0.04
		C100 (100%) + $\mathcal{N}(0.5)$	99.94 ± 0.06	100.00 ± 0.00	99.94 ± 0.06
		C100 (100%) + $\mathcal{N}(0.1)$	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		C100 (100%)	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		ImageNet	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
DUE	SVHN	None	-	-	82.66 ± 0.77
		C100 (10%)	-	-	77.24 ± 0.53
		C100 (100%) + $\mathcal{N}(0.5)$	-	-	80.61 ± 1.01
		C100 (100%) + $\mathcal{N}(0.1)$	-	-	77.94 ± 1.18
		C100 (100%)	-	-	78.76 ± 1.74
		ImageNet	-	-	**92.18 ± 0.11**
	STL10	None	-	-	78.91 ± 1.06
		C100 (10%)	-	-	75.98 ± 0.87
		C100 (100%) + $\mathcal{N}(0.5)$	-	-	79.49 ± 0.33
		C100 (100%) + $\mathcal{N}(0.1)$	-	-	80.21 ± 0.83
		C100 (100%)	-	-	79.63 ± 1.34
		ImageNet	-	-	**89.56 ± 0.53**
	CelebA	None	-	-	65.64 ± 1.77
		C100 (10%)	-	-	**74.62 ± 1.56**
		C100 (100%) + $\mathcal{N}(0.5)$	-	-	66.58 ± 3.20
		C100 (100%) + $\mathcal{N}(0.1)$	-	-	75.15 ± 1.84
		C100 (100%)	-	-	74.60 ± 1.11
		ImageNet	-	-	72.19 ± 1.42
	Camelyon	None	-	-	73.00 ± 2.81
		C100 (10%)	-	-	34.84 ± 3.53
		C100 (100%) + $\mathcal{N}(0.5)$	-	-	47.78 ± 5.30
		C100 (100%) + $\mathcal{N}(0.1)$	-	-	63.88 ± 5.79
		C100 (100%)	-	-	75.58 ± 5.17
		ImageNet	-	-	**97.28 ± 0.47**
	SVHN OODom.	None	-	-	100.00 ± 0.00
		C100 (10%)	-	-	99.51 ± 0.12
		C100 (100%) + $\mathcal{N}(0.5)$	-	-	99.99 ± 0.00
		C100 (100%) + $\mathcal{N}(0.1)$	-	-	99.99 ± 0.00
		C100 (100%)	-	-	99.99 ± 0.00
		ImageNet	-	-	**100.00 ± 0.00**

training phase, both the encoder and decoder are trained with the cross-entropy loss plus a MSE reconstruction term. During the joint training phase, we load the pretrained

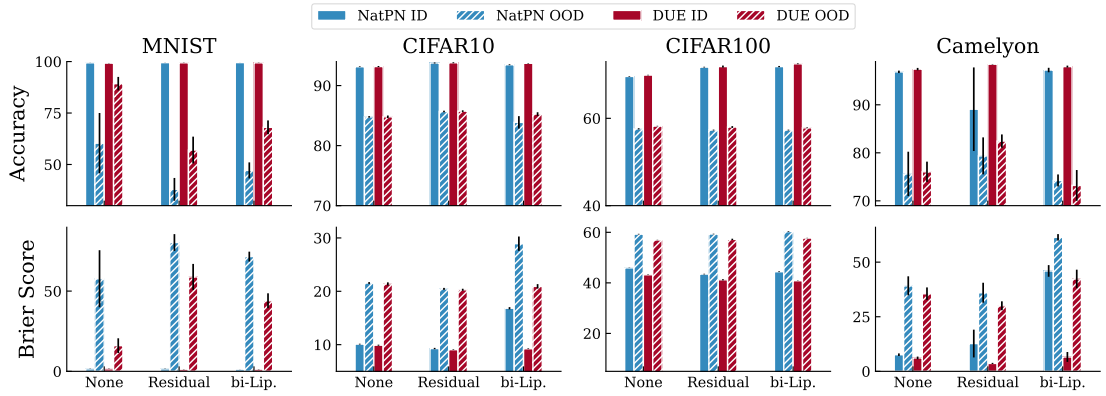


Figure C.1: Results OOD generalization and OOD detection results of DUMs with none, residual and bi-lipschitz **architecture constraints**. Bi-lipschitz and more specifically can improve OOD detection by mitigating feature collapse (see Fig. C.3a) at the expense of degrading OOD generalization.

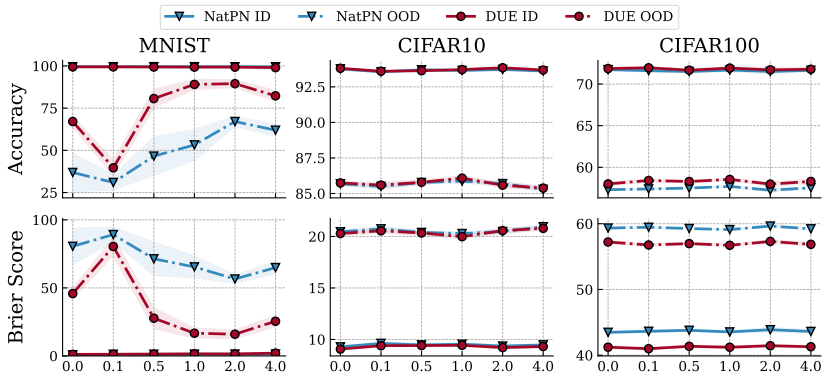


Figure C.2: Results OOD generalization and OOD detection results of DUMs with reconstruction **architecture constraints**. Increasing the strength of the reconstruction factor λ improves the OOD generalization only on the simpler MNIST/CMNIST datasets but fails for more complex datasets.

encoder and decoder, and joint train with the DUMs' respective loss plus the MSE reconstruction term. For the toy dataset we use an encoder with 4 linear layers of 128 dimension each.

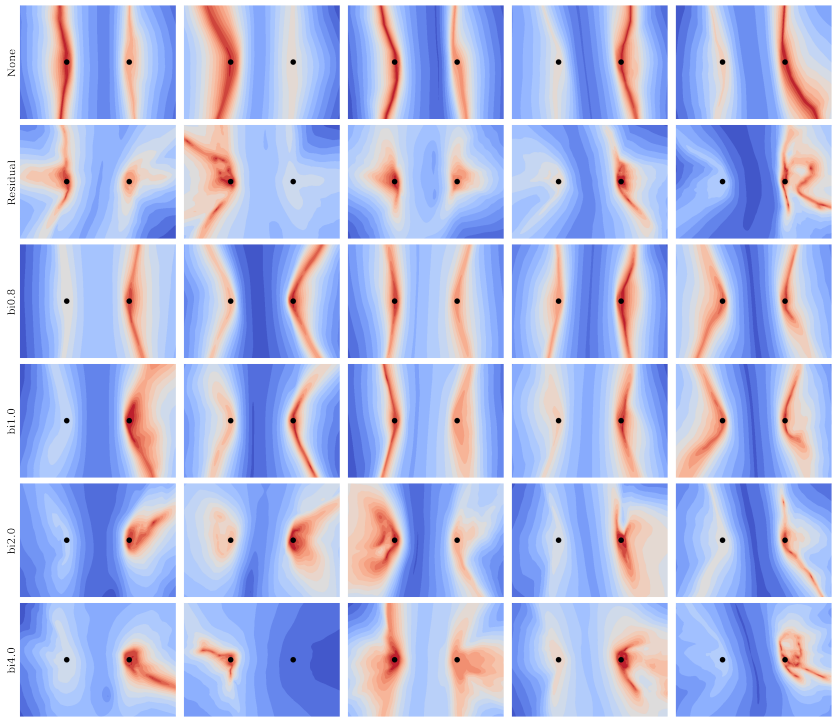
C.3 Prior for DUMs Details

For all the prior experiments we use the default training settings in Appendix C.1 and Table C.2. In the following experiments, we vary the *entropy regularization* λ and the *evidence prior* n^{prior} for NatPN, and the choice of kernel for DUE.

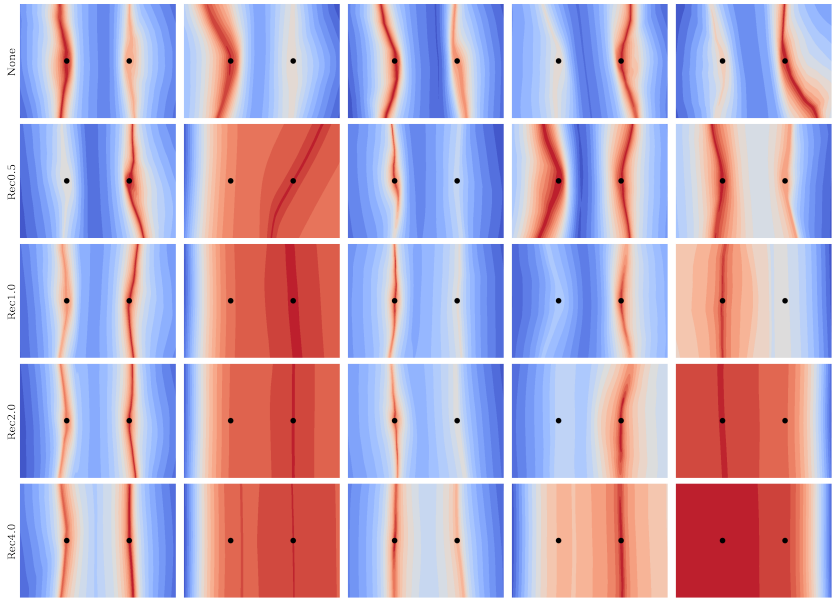
Table C.5: Encoder architecture OOD detection. For each training dataset we show the uncertainty estimation results on the corresponding OOD dataset. We observe that new architectures (EfficientNet, Swin) have consistently better results. Interestingly, the transformer based model Swin, is not able to detect out-of-domain data, which should be easy in principle.

Model	OOD Data	Architecture	OOD Alea. (↑)	OOD Epis. (↑)	OOD Pred. (↑)
NatPN	SVHN	ResNet18	**89.90 ± 1.22**	78.14 ± 3.13	**89.90 ± 1.22**
		ResNet50	89.34 ± 0.66	92.02 ± 0.49	89.34 ± 0.66
		EfficientNet_V2_S	88.65 ± 0.68	92.52 ± 0.60	88.65 ± 0.68
		Swin_T	87.73 ± 1.36	**94.17 ± 0.74**	87.73 ± 1.36
	STL10	ResNet18	**90.99 ± 0.37**	83.90 ± 0.59	**90.99 ± 0.37**
		ResNet50	85.46 ± 0.50	90.76 ± 0.31	85.46 ± 0.50
		EfficientNet_V2_S	88.68 ± 0.62	91.11 ± 0.47	88.68 ± 0.62
		Swin_T	85.44 ± 0.56	**92.16 ± 0.40**	85.44 ± 0.56
	CelebA	ResNet18	66.46 ± 3.48	50.61 ± 2.14	66.46 ± 3.48
		ResNet50	59.30 ± 3.77	64.80 ± 2.15	59.30 ± 3.77
		EffNet_V2_S	**67.04 ± 1.74**	66.29 ± 1.45	**67.04 ± 1.74**
		Swin_T	63.60 ± 2.16	**72.80 ± 1.00**	63.60 ± 2.16
	Camelyon	ResNet18	90.09 ± 4.43	96.28 ± 1.06	90.09 ± 4.43
		ResNet50	90.64 ± 2.47	97.82 ± 0.56	90.64 ± 2.47
		EfficientNet_V2_S	94.56 ± 0.79	97.42 ± 0.44	94.56 ± 0.79
		Swin_T	**95.43 ± 1.47**	**98.71 ± 0.50**	**95.43 ± 1.47**
	SVHN OODom.	ResNet18	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
		ResNet50	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
		EfficientNet_V2_S	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
		Swin_T	97.37 ± 0.29	93.31 ± 1.42	97.37 ± 0.29
DUE	SVHN	ResNet18	-	-	88.77 ± 0.28
		ResNet50	-	-	92.18 ± 0.11
		EfficientNet_V2_S	-	-	90.95 ± 0.53
		Swin_T	-	-	**93.62 ± 0.39**
	STL10	ResNet18	-	-	**90.66 ± 0.48**
		ResNet50	-	-	89.56 ± 0.53
		EfficientNet_V2_S	-	-	89.08 ± 0.39
		Swin_T	-	-	89.73 ± 0.36
	CelebA	ResNet18	-	-	64.75 ± 1.44
		ResNet50	-	-	72.19 ± 1.42
		EffNet_V2_S	-	-	**72.28 ± 1.76**
		Swin_T	-	-	69.37 ± 0.67
	Camelyon	ResNet18	-	-	96.01 ± 1.13
		ResNet50	-	-	97.28 ± 0.47
		EfficientNet_V2_S	-	-	94.82 ± 0.65
		Swin_T	-	-	**99.33 ± 0.14**
	SVHN OODom.	ResNet18	-	-	**100.00 ± 0.00**
		ResNet50	-	-	**100.00 ± 0.00**
		EfficientNet_V2_S	-	-	**100.00 ± 0.00**
		Swin_T	-	-	97.47 ± 0.22

Before starting the training, we inject the artificial aleatoric noise by reassigning the target y with a randomly chosen class. Two datasets with different degree of noise are used, where 10% and 20% of all the labels in the training dataset are reassigned.

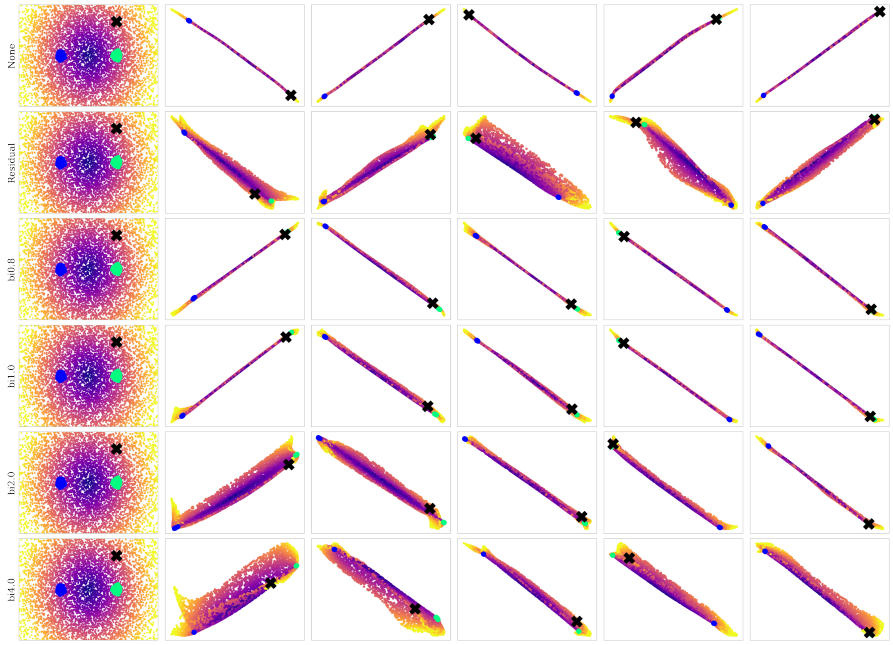


(a) Bi-Lipschitz

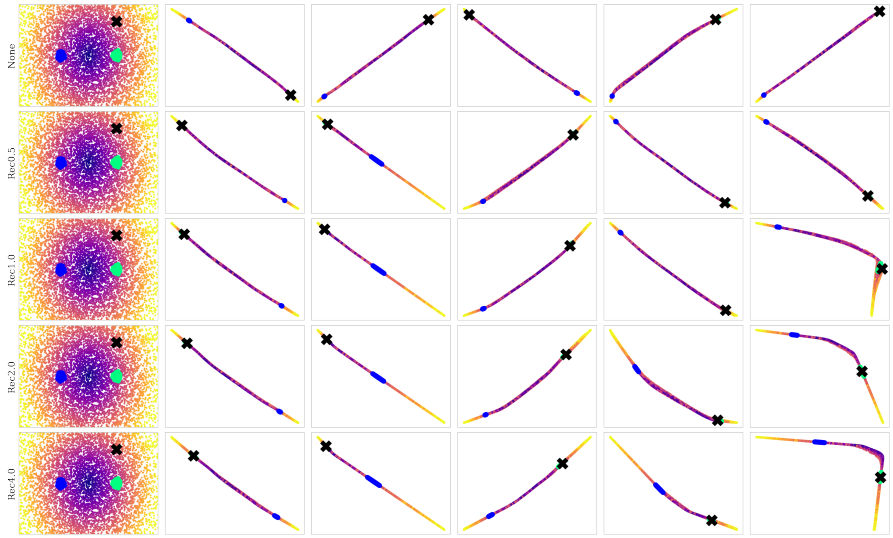


(b) Reconstruction

Figure C.3: Regularization constraint toy dataset uncertainty boundaries with NatPN. The two black dots represent the center of two different class of Gaussian data, sharing the same y-axis to trigger the *feature collapse* phenomenon. The color represents the likelihood produced by the uncertainty head. Each row is a different setting, e.g. *bi1.0* is the bi-Lipschitz constraint with the Lipschitz constant $c = 1$ and *rec1.0* is the reconstruction term with $\lambda = 1$. Each column is a different seed initialization. (top) **Bi-Lipschitz** experiment shows that the core encoder architecture constrained with a larger *Lipschitz constant* in the last two rows behaves similar to the encoder constrained with only the residual connection (second row) showing that relaxing the spectral normalization constraint falls back to the residual connection, preventing the feature collapse. (bottom) **Reconstruction** experiment shows that it does not help to prevent feature collapse by itself. The core encoder architecture is not constrained with bi-Lipschitz.

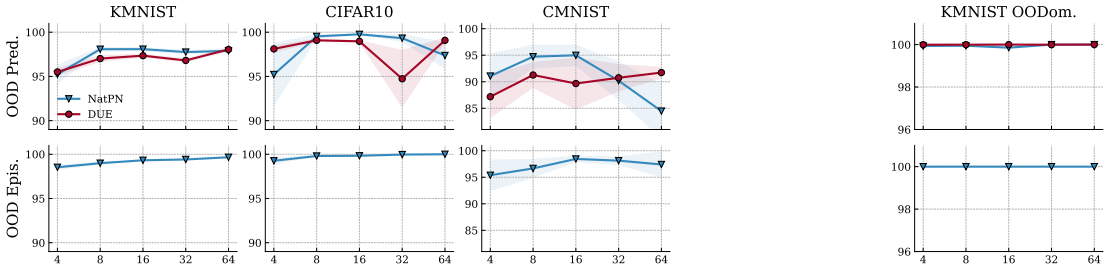


(a) Bi-Lipschitz

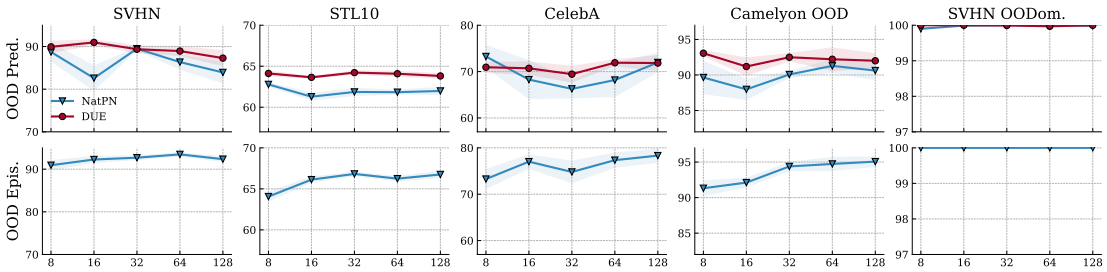


(b) Reconstruction

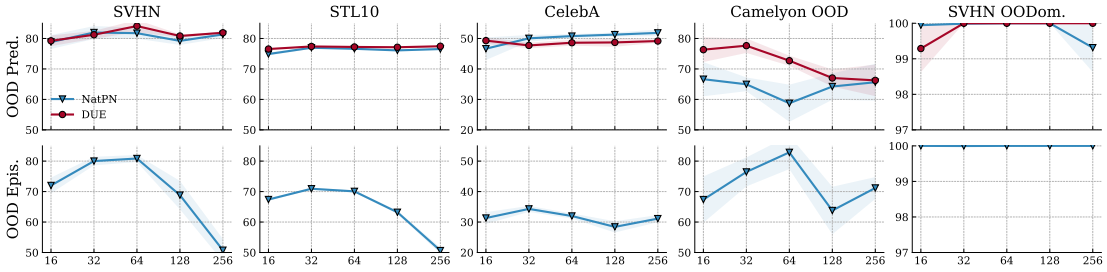
Figure C.4: Regularization constraint toy dataset feature collapse with NatPN. Similarly to [421], we run the toy experiment where the first column represent two Gaussian class of data, sharing the same y-axis center to trigger the *feature collapse* phenomenon, and a grid of unrelated point to simulate the space distorsion (colors are based on the Gaussian's generating distribution). Each row is a different setting, e.g. *bi1.0* is the bi-Lipschitz constraint with the Lipschitz constant $c = 1$ and *rec1.0* is the reconstruction term with $\lambda = 1$. Each column is a different seed initialization. Results are the same as C.3a. Larger Lipschitz constant c reverts back to the residual connection, and reconstruction regularization collapses the 2D dimension into one single dimension.



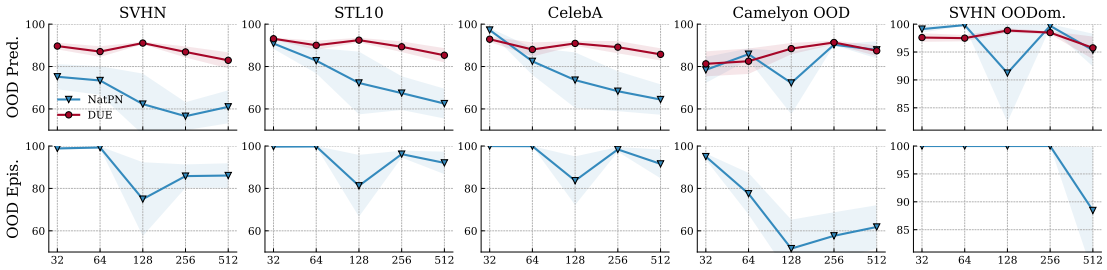
(a) MNIST



(b) CIFAR10

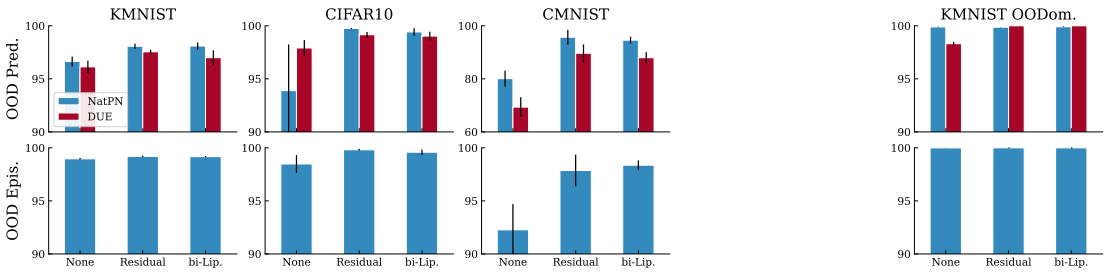


(c) CIFAR100

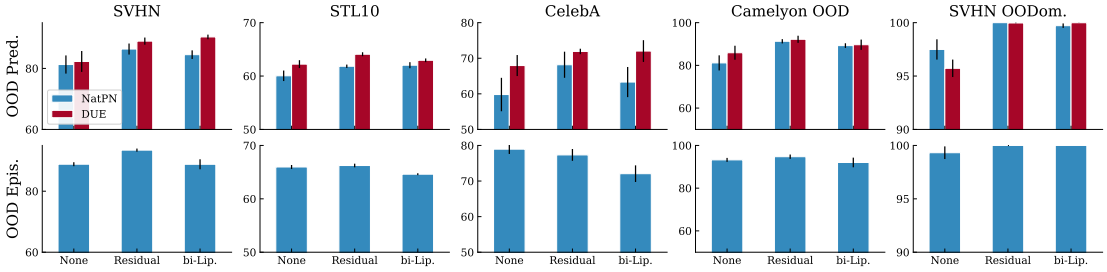


(d) Camelyon ID

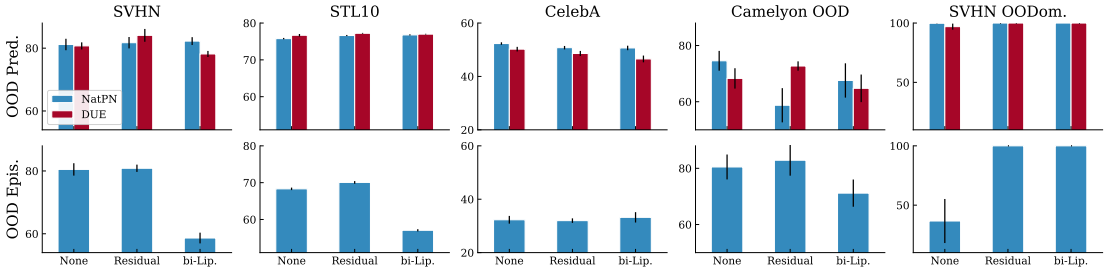
Figure C.5: Latent dimension OOD detection. For each training dataset we show the uncertainty estimation results on the corresponding OOD dataset. NatPN encounters numerical instabilities with high latent dimension on Camelyon dataset, while DUE is less sensitive to the variation.



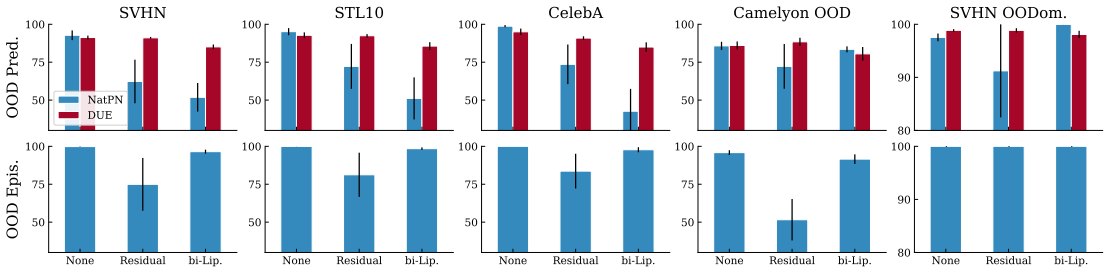
(a) MNIST



(b) CIFAR10

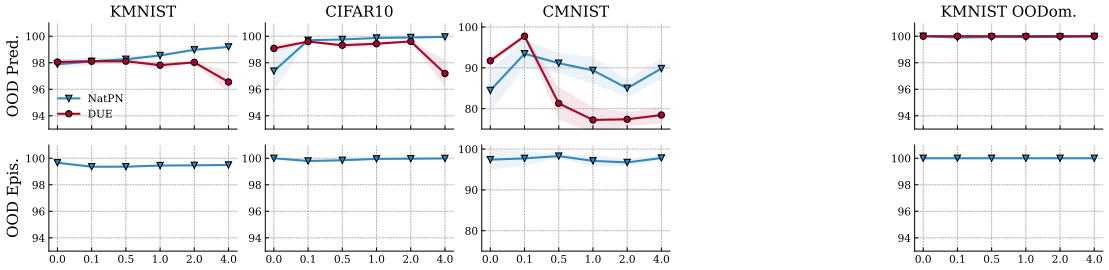


(c) CIFAR100

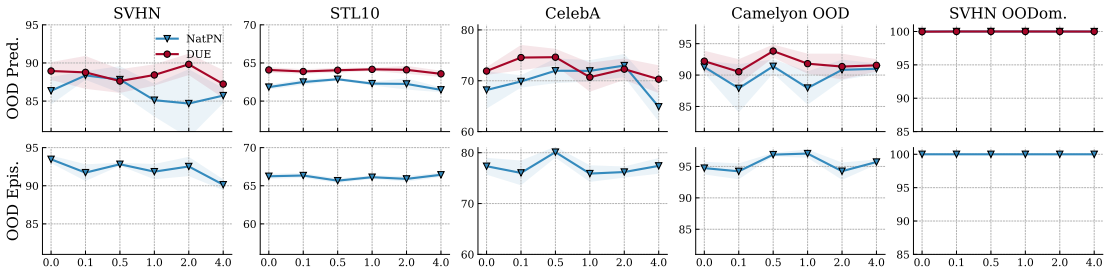


(d) Camelyon ID

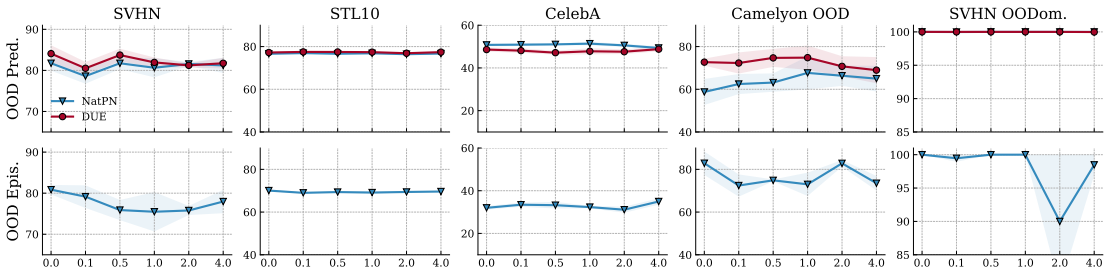
Figure C.6: Bi-lipschitz OOD detection. For each training dataset we show the uncertainty estimation results on the corresponding OOD dataset. Bi-Lipschitz improvements are not consistent across different OOD datasets.



(a) MNIST



(b) CIFAR10



(c) CIFAR100

Figure C.7: Reconstruction regularization OOD detection. Increasing the weight coefficient of the reconstruction loss term improved the OOD detection of NatPN in MNIST. However, we did not observe improvements on more complex datasets such as CIFAR.

Figure C.8: Reconstruction regularization CIFAR samples. (left) The original input (right) the reconstructed input after NatPN's joint training phase. The reconstruction discards detailed information compared to the original input.

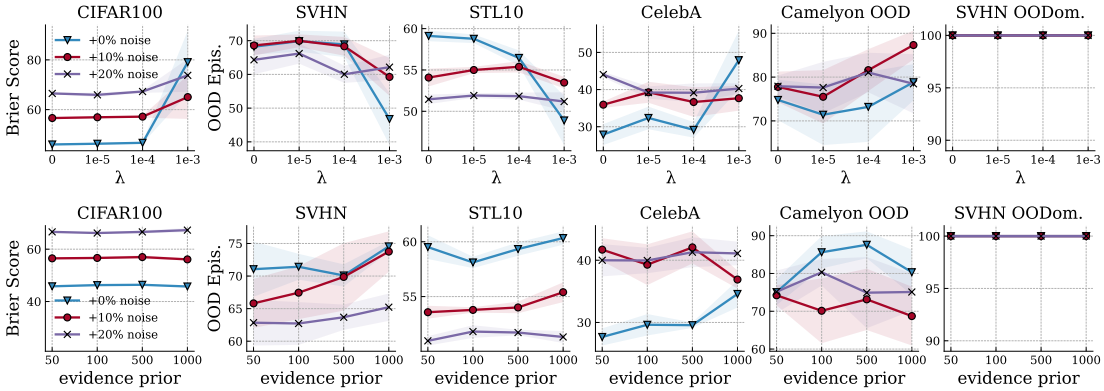


Figure C.9: Results of enforcing different **prior** in NatPN on CIFAR100 by changing the (top) *entropy regularization* λ and the (bottom) *evidence prior* n^{prior}. Different priors do not lead consistent results improvements.

D Robustness of Uncertainty Estimation

D.1 Appendix

D.1.1 Closed-form computation of uncertainty measures & Uncertainty attacks

Dirichlet-based uncertainty models allow to compute several uncertainty measures in closed form (see [271] for a derivation). As proposed by [271], we use precision m_{α_0}, differential entropy m_{diffE} and mutual information m_{MI} to estimate uncertainty on predictions.

The differential entropy m_{diffE} of a DBU model reaches its maximum value for equally probable categorical distributions and thus, a on flat Dirichlet distribution. It is a measure for distributional uncertainty and expected to be low on ID data, but high on OOD data.

$$\begin{aligned} m_{\text{diffE}} = &\sum_{c}^{K} \ln \Gamma(\alpha_c) - \ln \Gamma(\alpha_0) \\ &- \sum_{c}^{K} (\alpha_c - 1) \cdot (\Psi(\alpha_c) - \Psi(\alpha_0)) \end{aligned} \tag{D.1}$$

where α are the parameters of the Dirichlet-distribution, Γ is the Gamma function and Ψ is the Digamma function.

The mutual information m_{MI} is the difference between the total uncertainty (entropy of the expected distribution) and the expected uncertainty on the data (expected entropy of the distribution). This uncertainty is expected to be low on ID data and high on OOD data.

$$m_{\text{MI}} = -\sum_{c=1}^{K} \frac{\alpha_c}{\alpha_0} \left(\ln \frac{\alpha_c}{\alpha_0} - \Psi(\alpha_c + 1) + \Psi(\alpha_0 + 1) \right) \tag{D.2}$$

Furthermore, we use the precision α_0 to measure uncertainty, which is expected to be high on ID data and low on OOD data.

$$m_{\alpha_0} = \alpha_0 = \sum_{c=1}^{K} \alpha_c \tag{D.3}$$

As these uncertainty measures are computed in closed form and it is possible to obtain their gradients, we use them (i.e. m_{diffE}, m_{MI}, m_{α_0}) are target function of our uncertainty attacks. Changing the attacked target function allows us to use a wide range of gradient-based attacks such as FGSM attacks, PGD attacks, but also more sophisticated attacks such as Carlini-Wagner attacks.

D.1.2 Details of the Experimental setup

Models. We trained all models with a similar based architecture. We used namely 3 linear layers for vector data sets, 3 convolutional layers with size of $5 + 3$ linear layers for MNIST and the VGG16 [389] architecture with batch normalization for CIFAR10. All the implementation are performed using Pytorch [341]. We optimized all models using Adam optimizer. We performed early stopping by checking for loss improvement every 2 epochs and a patience of 10. The models were trained on GPUs (1 TB SSD).

We performed a grid-search for hyper-parameters for all models. The learning rate grid search was done in $[1e^{-5}, 1e^{-3}]$. For Posterior Network, we used Radial Flows with a depth of 6 and a latent space equal to 6. Further, we performed a grid search for the regularizing factor in $[1e^{-7}, 1e^{-4}]$. For PriorNet, we performed a grid search for the OOD loss weight in $[1, 10]$. For DDNet, we distilled the knowledge of 5 neural networks after a grid search in $[2, 5, 10, 20]$ neural networks. Note that it already implied a significant overhead at training compare to other models.

Metrics. For all experiments, we focused on using AUC-PR scores since it is well suited to imbalance tasks [369] while bringing theoretically similar information than AUC-ROC scores [98]. We scaled all scores from $[0, 1]$ to $[0, 100]$. All results are average over 5 training runs using the best hyper-parameters found after the grid search.

Data sets. For vector data sets, we use 5 different random splits to train all models. We split the data in training, validation and test sets (60%, 20%, 20%).

We use the segment vector data set [111], where the goal is to classify areas of images into 7 classes (window, foliage, grass, brickface, path, cement, sky). We remove class window from ID training data to provide OOD training data to PriorNet. Further, We remove the class 'sky' from training and instead use it as the OOD data set for OOD detection experiments. Each input is composed of 18 attributes describing the image area. The data set contains $2,310$ samples in total.

We further use the Sensorless Drive vector data set [111], where the goal is to classify extracted motor current measurements into 11 different classes. We remove class 9 from ID training data to provide OOD training data to PriorNet. We remove classes 10 and 11 from training and use them as the OOD dataset for OOD detection experiments. Each input is composed of 49 attributes describing motor behaviour. The data set contains $58,509$ samples in total.

Additionally, we use the MNIST image data set [249] where the goal is to classify pictures of hand-drawn digits into 10 classes (from digit 0 to digit 9). Each input is composed of a $1 \times 28 \times 28$ tensor. The data set contains $70,000$ samples. For OOD detection experiments, we use FashionMNIST [450] and KMNIST [83] containing images of Japanese characters and images of clothes, respectively. FashionMNIST was used as training OOD for PriorNet while KMNIST is used as OOD at test time.

Finally, we use the CIFAR10 image data set [238] where the goal is to classify a picture of objects into 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). Each input is a $3 \times 32 \times 32$ tensor. The data set contains $60,000$ samples.

For OOD detection experiments, we use street view house numbers (SVHN) [313] and CIFAR100 [238] containing images of numbers and objects respectively. CIFAR100 was used as training OOD for PriorNet while SVHN is used as OOD at test time.

Perturbations. For all label and uncertainty attacks, we used Fast Gradient Sign Methods and Project Gradient Descent. We tried 6 different attack radii 0.0, 0.1, 0.2, 0.5, 1.0, 2.0, and 4.0. These radii operate on the input space after data normalization. We bound perturbations by L_∞-norm or by L_2-norm, with

$$L_\infty(x) = \max_{i=1,\dots,D} |x_i| \quad \text{and} \quad L_2(x) = (\sum_{i=1}^{D} x_i^2)^{0.5}. \tag{D.4}$$

For L_∞-norm it is obvious how to relate perturbation size ε with perturbed input images, because all inputs are standardized such that the values of their features are between 0 and 1. A perturbation of size $\varepsilon = 0$ corresponds to the original input, while a perturbation of size $\varepsilon = 1$ corresponds to the whole input space and allows to change all features to any value.

For L_2-norm the relation between perturbation size ε and perturbed input images is less obvious. To justify our choice for ε w.r.t. this norm, we relate perturbations size ε_2 corresponding to L_2-norm with perturbations size ε_∞ corresponding to L_∞-norm. First, we compute ε_2, such that the L_2-norm is the smallest super-set of the L_∞-norm. Let us consider a perturbation of ε_∞. The largest L_2-norm would be obtained if each feature is perturbed by ε_∞. Thus, perturbation ε_2, such that L_2 encloses L_∞ is $\varepsilon_2 = (\sum_{i=1}^{D} \varepsilon_\infty^2)^{0.5} = \sqrt{D}\varepsilon_\infty$. For the MNIST-data set, with $D = 28 \times 28$ input features L_2-norm with $\varepsilon_2 = 28$ encloses L_∞-norm with $\varepsilon_\infty = 1$.

Alternatively, ε_2 can be computes such that the volume spanned by L_2-norm is equivalent to the one spanned by L_∞-norm. Using that the volume spanned by L_∞-norm is ε_∞^D and the volume spanned by L_2-norm is $\frac{\pi^{0.5D}\varepsilon_2^D}{\Gamma(0.5D+1)}$ (where Γ is the Gamma-function), we obtain volume equivalence if $\varepsilon_2 = \Gamma(0.5D+1)^{\frac{1}{D}}\sqrt{\pi}\varepsilon_\infty$. For the MNIST-data set, with $D = 28 \times 28$ input features L_2-norm with $\varepsilon_2 \approx 21.39$ is volume equivalent to L_∞-norm with $\varepsilon_\infty = 1$.

D.1.3 Additional Experiments

Tables D.1 and D.2 illustrate that no DBU model maintains high accuracy under gradient-based label attacks. Accuracy under PGD attacks decreases more than under FGSM attacks, since PGD is stronger. Interestingly Noise attacks achieve also good performances with increasing Noise standard deviation. Note that the attack is not constraint to be with a given radius for Noise attacks.

Table D.1: Accuracy under PGD label attacks.

Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST							CIFAR10						
PostNet	**99.4**	**99.2**	**98.8**	96.8	89.6	53.8	13.0	89.5	73.5	51.7	13.2	2.2	0.8	0.3
PriorNet	99.3	99.1	**98.8**	97.4	**93.9**	**75.3**	4.8	88.2	**77.8**	**68.4**	**54.0**	**37.9**	**17.5**	**5.1**
DDNet	**99.4**	99.1	**98.8**	**97.5**	91.6	48.8	0.2	86.1	73.9	59.1	20.5	1.5	0.0	0.0
EvNet	99.2	98.9	98.4	96.8	92.4	73.1	**40.9**	**89.8**	71.7	48.8	11.5	2.7	1.5	0.4
	Sensorless							Segment						
PostNet	98.3	13.1	6.4	4.0	**7.0**	**9.8**	**11.3**	98.9	82.8	**50.1**	**19.2**	**8.8**	**5.1**	**8.6**
PriorNet	**99.3**	16.5	5.6	1.2	0.4	0.2	1.6	**99.5**	90.7	47.6	7.8	0.2	0.0	0.4
DDNet	**99.3**	12.4	2.4	0.6	0.3	0.1	0.1	99.2	**90.8**	45.7	6.9	0.0	0.0	0.0
EvNet	99.0	**35.3**	**22.3**	**11.2**	**7.0**	5.2	4.0	99.3	91.8	54.0	10.3	0.8	0.5	0.6

Table D.2: Accuracy under FGSM label attacks.

Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST							CIFAR10						
PostNet	**99.4**	**99.2**	**98.9**	97.7	95.2	**90.1**	**79.2**	89.5	72.3	54.9	31.2	21.0	16.8	15.6
PriorNet	99.3	99.1	**98.9**	97.7	**95.8**	93.2	76.7	88.2	**77.3**	**70.1**	**59.4**	**52.3**	**48.5**	**46.8**
DDNet	**99.4**	**99.2**	**98.9**	**97.8**	94.7	79.2	25.2	86.1	73.0	60.2	32.5	14.6	7.1	6.0
EvNet	99.2	98.9	98.6	97.6	**95.8**	**90.1**	74.4	**89.8**	71.4	54.5	29.6	18.1	14.4	13.4
	Sensorless							Segment						
PostNet	98.3	19.6	10.9	10.9	11.9	12.4	12.5	98.9	79.6	**57.3**	**31.5**	**18.4**	**20.6**	**19.9**
PriorNet	**99.3**	24.7	11.8	8.6	8.5	8.1	8.3	**99.5**	85.5	40.5	8.9	0.4	0.3	0.2
DDNet	**99.3**	18.0	8.2	6.5	5.4	6.7	7.8	99.2	86.4	36.2	11.9	0.9	0.0	0.0
EvNet	99.0	**42.0**	**28.0**	**17.5**	**13.7**	**13.6**	**14.9**	99.3	**90.6**	55.2	14.2	2.4	0.5	0.1

Table D.3: Accuracy under Noise label attacks.

Noise Std	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST							CIFAR10						
PostNet	**99.4**	**98.6**	91.8	**14.9**	**1.3**	**0.1**	0.0	**91.7**	21.5	10.1	0.1	1.2	0.0	1.9
PriorNet	99.3	98.5	**95.7**	14.4	0.0	0.0	0.0	87.7	**28.1**	**11.2**	9.7	5.0	**8.5**	**9.0**
DDNet	**99.4**	**98.6**	92.4	13.3	0.7	0.0	0.0	81.7	23.0	**11.2**	**11.2**	**11.0**	7.8	6.7
EvNet	99.3	96.9	81.6	11.7	0.5	0.0	0.0	89.5	20.7	11.1	5.2	0.5	2.3	3.9
	Sensorless							Segment						
PostNet	98.1	0.1	**3.7**	**11.7**	**11.7**	**11.7**	**11.7**	98.5	39.4	3.9	**1.8**	**12.1**	**20.3**	**22.1**
PriorNet	**99.3**	0.2	0.0	0.0	0.0	0.3	2.4	**99.4**	47.9	8.8	0.0	0.0	0.0	0.0
DDNet	99.0	**0.4**	0.1	0.0	0.0	0.0	0.0	99.1	50.0	**10.3**	0.0	0.0	0.3	0.0
EvNet	98.6	0.2	0.0	0.1	1.4	4.6	8.8	99.1	**50.3**	**10.3**	1.2	0.3	0.0	1.5

D.1.3.1 Uncertainty estimation under label attacks

Is low uncertainty a reliable indicator of correct predictions?

On non-perturbed data uncertainty estimates are an indicator of correctly classified samples, but if the input data is perturbed none of the DBU models maintains its high performance. Thus, uncertainty estimates are not a robust indicator of correctly labeled inputs.

Table D.4: Distinguishing between correctly and wrongly predicted labels based on the differential entropy under PGD label attacks (AUC-PR).

	MNIST							Segment						
Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
PostNet	99.9	99.9	99.8	98.7	89.5	43.5	9.0	99.9	77.6	31.6	**11.1**	**5.3**	**4.4**	8.7
PriorNet	99.9	99.8	99.6	97.7	90.5	**69.1**	6.4	**100.0**	**96.8**	44.5	4.5	0.4	0.0	**15.2**
DDNet	**100.0**	**100.0**	**99.9**	**99.7**	**97.6**	50.2	0.1	**100.0**	**96.8**	**54.0**	4.3	0.0	0.0	0.0
EvNet	99.6	99.3	98.7	96.1	88.8	63.1	**31.7**	**100.0**	95.9	44.3	5.9	0.8	0.6	0.7

Table D.5: Distinguishing between correctly and wrongly predicted labels based on the precision α_0 under PGD label attacks (AUC-PR).

Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST							CIFAR10						
PostNet	**100.0**	99.9	99.7	98.2	87.9	39.1	6.9	**98.7**	88.6	56.2	7.8	1.2	0.4	0.3
PriorNet	99.9	99.8	99.6	97.7	90.4	**69.1**	6.6	92.9	77.7	60.5	**37.6**	**24.9**	**11.3**	**3.0**
DDNet	**100.0**	**100.0**	**100.0**	**99.8**	**98.2**	51.1	0.1	97.6	**91.8**	**78.3**	18.1	0.8	0.0	0.0
EvNet	99.6	99.2	98.6	95.7	88.6	63.6	**32.6**	97.9	85.9	57.2	10.2	4.0	2.4	0.3
	Sensorless							Segment						
PostNet	99.6	7.0	3.3	3.1	**6.9**	**9.8**	**11.3**	99.9	74.2	31.6	**11.1**	**5.0**	**4.2**	**8.6**
PriorNet	99.8	10.5	3.2	0.6	0.2	0.2	1.8	**100.0**	96.9	**45.2**	4.4	0.4	0.0	1.2
DDNet	99.8	8.7	1.3	0.3	0.2	0.1	0.2	**100.0**	**97.1**	45.0	4.1	0.0	0.0	0.0
EvNet	**99.9**	**23.2**	**13.2**	**6.0**	3.7	2.7	2.1	**100.0**	95.7	44.5	5.9	0.8	0.6	0.7

Table D.6: Distinguishing between correctly and wrongly predicted labels based on the mutual information under PGD label attacks (AUC-PR).

Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST							CIFAR10						
PostNet	99.7	99.7	99.6	99.2	92.4	40.0	6.9	**97.3**	84.5	56.2	12.2	2.4	0.7	0.3
PriorNet	99.9	99.8	99.6	97.7	90.3	**68.9**	6.4	82.7	65.6	51.4	**35.5**	**24.4**	**11.0**	**2.9**
DDNet	**100.0**	**99.9**	**99.9**	**99.7**	**97.4**	50.2	0.1	96.9	**90.8**	**77.2**	18.8	0.8	0.0	0.0
EvNet	97.8	97.0	95.7	92.6	86.1	62.3	**28.9**	91.3	72.4	47.9	11.4	1.6	0.9	1.6
	Sensorless							Segment						
PostNet	99.3	7.0	3.3	3.3	**7.0**	**9.8**	11.3	99.9	73.2	31.5	**11.1**	**5.0**	**4.3**	**8.7**
PriorNet	**99.8**	10.5	3.2	0.6	0.2	0.1	**11.8**	**100.0**	**96.6**	**45.2**	4.5	0.4	0.0	1.1
DDNet	99.6	8.6	1.3	0.3	0.2	0.1	0.1	**100.0**	96.5	42.4	4.1	0.0	0.0	0.0
EvNet	99.1	**22.0**	**12.6**	**5.9**	3.7	2.7	2.2	**100.0**	90.5	41.0	5.9	0.8	0.6	0.7

Tables 6.2 and D.4 to D.6 illustrate that neither differential entropy nor precision, nor mutual information are a reliable indicator of correct predictions under PGD attacks.

Table D.7: Distinguishing between correctly and wrongly predicted labels based on the differential entropy under FGSM label attacks (AUC-PR).

Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST							CIFAR10						
PostNet	99.9	99.9	99.8	99.4	97.8	**92.1**	**83.2**	**98.5**	88.7	68.9	31.0	18.6	15.5	16.7
PriorNet	99.9	99.9	99.7	98.3	94.1	88.5	78.6	90.1	73.6	61.6	**46.1**	**38.5**	**35.6**	**37.3**
DDNet	**100.0**	**100.0**	**99.9**	**99.8**	**98.7**	86.4	23.0	97.3	**90.6**	**78.7**	39.4	13.7	6.0	5.1
EvNet	99.6	99.4	99.1	97.8	95.8	90.4	76.8	98.0	86.2	67.4	32.7	19.9	18.2	19.7
	Sensorless							Segment						
PostNet	99.7	11.7	7.3	9.3	11.8	12.5	12.5	99.9	73.6	40.6	**23.7**	**17.2**	**19.8**	**20.2**
PriorNet	99.8	21.4	10.4	8.5	9.0	9.2	10.3	**100.0**	93.7	37.7	5.8	1.1	0.9	0.8
DDNet	99.7	18.5	5.4	4.3	4.2	5.7	7.9	**100.0**	**94.1**	42.9	7.2	1.0	0.0	0.0
EvNet	**99.9**	**44.8**	**29.2**	**18.2**	**15.1**	**14.9**	**15.5**	**100.0**	93.7	**48.7**	8.7	2.4	1.6	0.5

Table D.8: Distinguishing between correctly and wrongly predicted labels based on the differential entropy under Noise label attacks (AUC-PR).

Noise Std	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST							CIFAR10						
PostNet	99.9	99.8	99.6	**74.2**	**7.4**	**0.2**	0.0	**98.**7	**76.**3	24.3	0.4	4.9	0.0	1.7
PriorNet	99.9	99.9	**99.8**	73.4	0.0	0.0	0.0	85.0	27.8	15.9	**20.**4	7.0	**7.**7	**8.3**
DDNet	**100.0**	**99.9**	99.4	51.1	0.6	0.1	0.0	96.1	61.0	**39.**8	14.2	**11.**3	6.9	6.9
EvNet	99.5	98.4	88.5	20.2	0.9	0.0	0.0	97.5	66.1	21.4	7.7	2.3	3.0	3.8
	Sensorless							Segment						
PostNet	99.7	0.3	**3.2**	**13.3**	**12.0**	**11.7**	**11.7**	99.9	53.9	4.8	1.8	**11.2**	**21.7**	**21.6**
PriorNet	**100.0**	0.3	0.0	0.0	0.0	7.8	11.5	**100.0**	**84.5**	15.6	0.0	0.0	0.0	0.0
DDNet	99.7	**0.9**	0.6	0.0	0.0	0.0	0.0	**100.0**	82.7	**23.9**	0.0	0.0	0.6	0.0
EvNet	99.8	0.3	0.0	0.1	1.7	5.5	10.0	**100.0**	78.3	19.0	**3.5**	0.5	0.0	1.7

DBU-models achieve significantly better results when they are attacked by FGSM-attacks (Table D.7), but as FGSM attacks provide much weaker adversarial examples than PGD attacks, this cannot be seen as real advantage.

Can we use uncertainty estimates to detect attacks against the class prediction?

PGD attacks do not explicitly consider uncertainty during the computation of adversarial examples, but they seem to provide perturbed inputs with similar uncertainty as the original input.

Table D.9: Attack-Detection based on differential entropy under PGD label attacks (AUC-PR).

	MNIST						Segment					
Att. Rad.	0.1	0.2	0.5	1.0	2.0	4.0	0.1	0.2	0.5	1.0	2.0	4.0
PostNet	57.7	66.3	83.4	90.5	79.0	50.1	**95.6**	73.5	**47.0**	**42.3**	**53.4**	**82.7**
PriorNet	**67.7**	**83.2**	**97.1**	**96.7**	92.1	82.9	86.7	83.3	38.0	31.3	30.8	31.5
DDNet	53.4	57.1	68.5	83.9	**96.0**	**86.3**	76.1	**83.5**	45.4	32.4	30.8	30.8
EvNet	54.8	59.0	68.5	75.9	72.6	59.8	94.9	80.9	41.5	32.5	31.1	31.1

Table D.10: Attack-Detection based on precision α_0 under PGD label attacks (AUC-PR).

Att. Rad.	0.1	0.2	0.5	1.0	2.0	4.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST						CIFAR10					
PostNet	63.3	75.7	92.6	95.1	75.3	39.5	**63.4**	**66.9**	42.1	32.9	31.6	31.2
PriorNet	**67.6**	**83.2**	**97.1**	**96.9**	**92.7**	**84.7**	53.3	56.0	55.6	**49.2**	42.2	35.4
DDNet	52.7	55.7	64.7	78.4	91.9	80.9	55.8	60.5	**57.3**	38.7	32.3	31.4
EvNet	49.1	48.0	45.1	42.7	41.8	39.2	48.4	46.9	46.3	46.3	**44.5**	**42.5**
	Sensorless						Segment					
PostNet	39.8	35.8	35.4	**52.0**	**88.2**	**99.0**	**94.6**	70.3	**46.3**	**42.6**	**54.9**	**84.0**
PriorNet	40.9	35.1	32.0	31.1	30.7	30.7	82.7	82.6	39.4	31.6	30.8	30.8
DDNet	**47.7**	**40.3**	35.3	32.8	31.3	30.8	80.0	**86.0**	43.3	33.6	31.0	30.8
EvNet	45.4	39.7	**36.1**	34.8	34.7	36.0	90.9	72.4	40.4	32.4	31.1	31.1

Table D.11: Attack-Detection based on mutual information under PGD label attacks (AUC-PR).

Att. Rad.	0.1	0.2	0.5	1.0	2.0	4.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST						CIFAR10					
PostNet	42.2	37.5	36.7	54.5	70.5	70.3	52.2	52.1	50.0	**65.9**	**76.3**	**80.7**
PriorNet	**67.7**	**83.3**	**97.1**	**96.9**	92.6	**84.5**	54.0	56.9	56.3	49.7	42.4	35.5
DDNet	53.1	56.3	66.5	81.0	**94.0**	82.9	**56.0**	**60.8**	**57.4**	38.2	32.1	31.3
EvNet	49.1	48.0	45.2	42.9	41.9	39.3	48.7	47.3	46.3	46.0	44.1	42.2
	Sensorless						Segment					
PostNet	**75.3**	**76.6**	**66.5**	**57.7**	**85.6**	**98.7**	**94.8**	73.5	**55.9**	**47.9**	**58.0**	**84.0**
PriorNet	40.7	35.0	32.0	31.0	30.7	30.7	83.5	82.7	39.2	31.6	30.8	30.8
DDNet	48.0	40.0	35.2	32.6	31.2	30.8	82.4	**88.1**	43.4	33.4	30.9	30.8
EvNet	45.5	39.7	36.1	34.8	34.7	36.0	91.7	72.9	40.5	32.4	31.1	31.1

FGSM and Noise attacks are easier to detect, but also weaker thand PGD attacks. This suggests that DBU models are capable of detecting weak attacks by using uncertainty estimation.

Table D.12: Attack-Detection based on differential entropy under FGSM label attacks (AUC-PR).

Att. Rad.	0.1	0.2	0.5	1.0	2.0	4.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST						CIFAR10					
PostNet	55.9	61.8	74.8	84.0	88.9	89.9	**62.1**	**67.2**	65.7	63.1	65.4	73.8
PriorNet	**67.4**	**82.4**	**96.9**	**98.3**	**98.9**	**99.6**	58.4	63.1	68.5	**70.1**	68.5	62.5
DDNet	53.6	57.3	68.3	82.6	95.6	98.7	57.2	62.9	**69.1**	68.7	**69.7**	**76.5**
EvNet	54.1	57.4	63.8	67.6	68.6	69.9	57.8	61.7	63.3	62.9	65.7	72.5
	Sensorless						Segment					
PostNet	**98.4**	**99.8**	**99.9**	**99.9**	**99.9**	**99.9**	**96.9**	**93.9**	**99.5**	**99.9**	**100.0**	**100.0**
PriorNet	48.7	38.6	32.7	32.9	38.6	44.3	89.0	80.8	46.7	37.2	33.7	32.4
DDNet	61.5	47.8	37.1	33.1	32.4	33.2	79.6	86.2	60.2	47.5	36.6	31.6
EvNet	67.3	65.5	72.3	73.4	75.3	79.1	95.7	87.2	59.3	51.7	51.1	53.5

Table D.13: Attack-Detection based on differential entropy under Noise label attacks (AUC-PR).

Noise Std.	0.1	0.2	0.5	1.0	2.0	4.0	0.1	0.2	0.5	1.0	2.0	4.0
	MNIST						CIFAR10					
PostNet	51.3	65.3	93.8	95.1	95.2	95.2	**80.8**	**84.5**	**97.6**	**99.5**	99.3	98.2
PriorNet	32.5	36.8	88.9	99.6	99.7	92.7	34.7	32.3	34.3	60.3	95.5	**100.0**
DDNet	**60.7**	**87.6**	**99.8**	**100.0**	**99.9**	**99.8**	59.1	62.6	81.5	98.6	**99.8**	98.7
EvNet	51.2	55.7	66.9	70.3	68.0	67.1	75.7	78.6	88.2	97.8	96.4	95.6
	Sensorless						Segment					
PostNet	**99.8**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**95.6**	**99.4**	**100.0**	**100.0**	**100.0**	**100.0**
PriorNet	42.0	33.8	31.5	34.7	43.7	47.0	56.7	56.7	39.8	33.7	31.9	33.7
DDNet	53.4	43.5	34.3	31.6	32.5	36.1	57.0	58.9	43.1	33.7	31.5	31.3
EvNet	67.1	78.8	88.3	95.4	96.9	97.8	60.8	63.5	61.2	64.8	73.7	85.2

D.1.3.2 Attacking uncertainty estimation

Are uncertainty estimates a robust feature for OOD detection?

Using uncertainty estimation to distinguish between ID and OOD data is not robust as shown in the following tables.

Table D.14: OOD detection based on differential entropy under PGD uncertainty attacks against differential entropy on ID data and OOD data (AUC-PR).

	ID-Attack (non-attacked OOD)							OOD-Attack (non-attacked ID)						
Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
MNIST – KMNIST														
PostNet	94.5	94.1	93.9	91.1	77.1	44.0	31.9	94.5	93.1	91.4	82.1	62.2	50.7	48.8
PriorNet	**99.6**	**99.4**	**99.1**	**97.8**	**93.8**	**77.6**	**32.0**	**99.6**	**99.4**	**99.1**	98.0	94.6	85.5	**73.9**
DDNet	99.3	99.1	98.9	**97.8**	93.5	63.3	30.7	99.3	99.1	99.0	**98.3**	**96.7**	**91.3**	73.8
EvNet	69.0	67.1	65.6	61.8	57.4	50.9	43.6	69.0	55.8	48.0	39.4	36.2	34.9	34.4
Seg. – Seg. class sky														
PostNet	**99.0**	**80.7**	**53.5**	**38.0**	**34.0**	**41.6**	**49.5**	**99.0**	**88.4**	69.2	45.1	**36.4**	**42.6**	**75.4**
PriorNet	34.8	31.4	30.9	30.8	30.8	30.8	30.8	34.8	31.8	31.0	30.8	30.8	30.8	32.1
DDNet	31.5	30.9	30.8	30.8	30.8	30.8	30.8	31.5	31.0	30.8	30.8	30.8	30.8	30.8
EvNet	92.5	67.2	43.2	31.6	30.9	30.9	31.2	92.5	86.1	**82.7**	**48.9**	32.7	30.9	30.9

Table D.15: OOD detection under PGD uncertainty attacks against differential entropy on ID data and OOD data (AUC-ROC).

	ID-Attack (non-attacked OOD)							OOD-Attack (non-attacked ID)						
Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
MNIST – KMNIST														
PostNet	91.6	91.3	91.9	91.5	80.2	38.8	9.2	91.6	90.4	89.0	81.6	62.6	45.0	43.1
PriorNet	**99.8**	**99.7**	**99.5**	**99.0**	**97.1**	**81.1**	8.7	**99.8**	**99.7**	**99.6**	**99.1**	**97.7**	**93.0**	**84.9**
DDNet	99.2	98.9	98.6	97.3	92.1	58.2	1.2	99.2	99.0	98.8	97.9	95.8	89.1	69.3
EvNet	81.2	79.6	78.2	74.6	69.5	58.7	**43.0**	81.2	67.2	54.8	35.4	25.5	20.7	18.5
CIFAR10 – SVHN														
PostNet	87.0	71.9	56.3	**30.2**	**20.2**	**15.0**	**9.7**	87.0	71.0	54.3	33.5	30.3	26.2	19.4
PriorNet	62.4	48.2	35.9	13.8	3.6	0.9	0.3	62.4	48.0	35.6	14.8	6.6	3.4	1.6
DDNet	87.0	**76.0**	**63.6**	29.3	6.1	1.1	0.4	87.0	**78.1**	**66.1**	26.2	5.1	0.7	0.1
EvNet	**88.0**	69.1	51.7	24.6	15.5	9.5	4.2	**88.0**	72.0	60.7	**47.9**	**42.1**	**33.3**	**24.0**
Sens. – Sens. class 10, 11														
PostNet	**85.3**	**49.1**	**38.1**	**7.8**	**8.2**	8.2	8.2	**85.3**	**57.2**	**54.0**	**27.3**	**31.5**	**86.7**	**99.5**
PriorNet	28.1	0.8	0.3	0.4	1.6	**8.4**	**26.8**	28.1	2.5	0.7	0.2	2.3	18.9	41.0
DDNet	21.0	3.0	0.9	0.4	0.6	2.1	7.3	21.0	4.4	2.1	1.9	2.2	2.2	4.1
EvNet	74.2	21.4	12.2	4.3	1.4	0.6	0.3	74.2	45.3	38.5	19.6	9.6	12.1	26.0
Seg. – Seg. class sky														
PostNet	**99.2**	**84.7**	**55.5**	**23.0**	**9.7**	**4.4**	**4.7**	**99.2**	**92.1**	**77.1**	41.5	**24.9**	**41.0**	**80.8**
PriorNet	17.1	4.4	1.3	0.0	0.0	0.0	0.1	17.1	5.9	1.5	0.1	0.0	0.1	5.8
DDNet	4.1	1.1	0.0	0.0	0.0	0.0	0.0	4.1	1.8	0.4	0.0	0.0	0.0	0.0
EvNet	91.2	54.5	23.3	3.9	0.9	0.4	0.2	91.2	82.9	76.4	**42.2**	9.7	0.8	0.6

Table D.16: OOD detection (AU-PR) under PGD uncertainty attacks against precision α_0 on ID data and OOD data.

	ID-Attack (non-attacked OOD)							OOD-Attack (non-attacked ID)						
Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
MNIST – KMNIST														
PostNet	98.4	97.4	96.0	88.8	70.9	39.3	31.3	98.4	97.2	95.2	82.8	52.6	34.3	32.1
PriorNet	**99.6**	**99.5**	**99.2**	**98.0**	**94.1**	**76.0**	31.1	**99.6**	**99.5**	**99.2**	**98.2**	**95.3**	**87.5**	**75.6**
DDNet	97.2	96.7	96.1	93.8	86.4	53.2	31.0	97.2	96.7	96.2	94.5	91.1	82.9	64.6
EvNet	39.8	39.2	38.8	37.9	37.1	36.3	**35.4**	39.8	34.5	32.5	31.2	31.0	30.9	31.0
CIFAR10 – SVHN														
PostNet	**82.4**	63.8	46.1	22.3	17.4	16.7	16.4	**82.4**	61.8	41.5	21.8	**19.8**	**17.5**	**15.8**
PriorNet	37.9	25.0	19.2	15.8	15.4	15.4	15.4	37.9	25.9	19.4	15.6	15.4	15.4	15.4
DDNet	81.1	**70.1**	**58.4**	**30.0**	16.7	15.5	15.4	81.1	**71.2**	**59.9**	**27.8**	16.5	15.5	15.4
EvNet	34.7	27.4	25.4	22.0	**19.7**	**18.1**	**17.1**	34.7	19.4	18.1	17.1	16.8	16.2	15.7
Sens. – Sens. class 10, 11														
PostNet	**77.4**	**39.6**	**35.9**	**31.7**	**44.4**	**44.4**	**44.4**	**77.4**	40.3	**38.6**	29.5	**34.0**	**79.4**	**97.4**
PriorNet	35.9	27.0	26.8	26.8	26.8	27.5	36.2	35.9	27.7	27.0	26.7	26.6	26.5	26.5
DDNet	55.6	34.4	31.7	30.4	29.5	30.2	33.4	55.6	**40.9**	34.1	28.0	26.9	26.6	26.5
EvNet	66.3	33.3	29.7	27.0	27.1	29.2	33.9	66.3	39.3	37.1	**31.3**	28.3	28.4	29.7
Seg. – Seg. class sky														
PostNet	**98.4**	74.8	51.0	**37.2**	**32.8**	**43.5**	**49.9**	**98.4**	84.7	66.1	42.4	34.8	**40.9**	**71.2**
PriorNet	32.1	30.9	30.8	30.8	30.8	30.8	30.8	32.1	31.0	30.8	30.8	30.8	30.8	30.8
DDNet	31.0	30.8	30.8	30.8	30.8	30.8	30.8	31.0	30.8	30.8	30.8	30.8	30.8	30.8
EvNet	98.3	**83.0**	**60.5**	34.0	31.0	30.8	30.8	98.3	**94.4**	**88.8**	**65.6**	**37.0**	31.4	30.9

Table D.17: OOD detection (AUC-ROC) under PGD uncertainty attacks against precision α_0 on ID data and OOD data.

	ID-Attack (non-attacked OOD)							OOD-Attack (non-attacked ID)						
Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
MNIST – KMNIST														
PostNet	98.4	97.6	96.4	90.9	74.0	28.9	6.3	98.4	97.6	96.3	89.0	61.3	19.6	9.7
PriorNet	**99.8**	**99.7**	**99.6**	**99.1**	**97.2**	**79.4**	4.4	**99.8**	**99.7**	**99.6**	**99.2**	**98.0**	**93.9**	**85.8**
DDNet	96.5	95.9	95.1	92.0	82.6	44.3	3.5	96.5	95.9	95.2	92.9	88.6	78.7	59.4
EvNet	35.9	34.1	32.8	30.1	27.4	24.6	**21.4**	35.9	18.7	10.4	3.7	2.0	1.7	2.0
CIFAR10 – SVHN														
PostNet	**87.4**	71.2	54.8	29.2	19.0	14.0	9.4	**87.4**	71.4	54.1	30.1	**25.8**	**17.5**	**5.8**
PriorNet	45.6	31.1	20.4	6.3	1.4	0.3	0.1	45.6	32.2	21.7	5.4	1.0	0.3	0.1
DDNet	84.9	**73.8**	**61.8**	30.2	9.3	3.0	0.8	84.9	**76.6**	**66.2**	**34.6**	10.4	2.3	0.3
EvNet	61.2	49.4	45.2	**37.6**	**30.5**	**23.4**	**17.0**	61.2	29.4	23.0	16.8	14.2	10.2	5.5
Sens. – Sens. class 10, 11														
PostNet	**87.2**	**48.8**	**37.3**	4.1	0.7	0.7	0.7	**87.2**	**50.0**	**45.4**	16.5	**27.6**	**81.9**	**98.0**
PriorNet	37.3	3.5	2.4	2.2	2.9	6.3	**19.2**	37.3	8.0	3.6	1.4	0.6	0.1	0.0
DDNet	55.2	23.7	17.7	**14.1**	**12.5**	**12.7**	15.7	55.2	37.1	27.7	9.4	2.5	0.6	0.1
EvNet	75.5	30.8	18.2	5.8	1.6	0.6	0.2	75.5	47.8	41.9	**24.1**	10.2	10.2	15.6
Seg. – Seg. class sky														
PostNet	**98.6**	77.7	**50.8**	**20.3**	**8.2**	**1.3**	**0.5**	**98.6**	88.9	73.4	36.2	19.4	**36.7**	**75.2**
PriorNet	8.5	1.3	0.2	0.0	0.0	0.0	0.1	8.5	2.0	0.4	0.0	0.0	0.0	0.0
DDNet	2.2	0.3	0.0	0.0	0.0	0.0	0.0	2.2	0.5	0.1	0.0	0.0	0.0	0.0
EvNet	97.7	**78.4**	47.7	9.9	1.2	0.2	0.1	97.7	**93.5**	**86.9**	**62.2**	**21.5**	3.7	1.0

Table D.18: OOD detection (AU-PR) under PGD uncertainty attacks against distributional uncertainty on ID data and OOD data.

	ID-Attack (non-attacked OOD)							OOD-Attack (non-attacked ID)						
Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
MNIST – KMNIST														
PostNet	80.5	76.2	73.4	69.1	66.6	65.4	**60.2**	80.5	72.1	63.9	43.9	33.0	30.9	30.8
PriorNet	**99.6**	**99.4**	**99.2**	**98.0**	**94.1**	**76.3**	31.2	**99.6**	**99.4**	**99.2**	**98.2**	**95.2**	**87.2**	**75.2**
DDNet	98.4	98.1	97.7	95.8	89.5	56.2	30.9	98.4	98.1	97.8	96.5	93.8	86.3	67.7
EvNet	40.1	39.5	39.1	38.2	37.3	36.5	35.6	40.1	34.6	32.6	31.3	31.0	31.0	31.1
CIFAR10 – SVHN														
PostNet	64.2	44.7	37.5	**31.1**	**28.5**	**25.0**	**19.3**	64.2	31.0	19.5	16.3	16.4	**16.5**	**16.3**
PriorNet	40.8	27.4	20.4	15.9	15.4	15.4	15.4	40.8	28.3	21.1	15.9	15.4	15.4	15.4
DDNet	**82.0**	**71.0**	**59.1**	29.9	16.6	15.5	15.4	**82.0**	**72.2**	**60.3**	**26.3**	16.2	15.4	15.4
EvNet	36.4	28.7	26.5	22.8	20.2	18.4	17.2	36.4	19.8	18.3	17.2	**16.9**	16.2	15.7
Sens. – Sens. class 10, 11														
PostNet	**79.1**	**40.3**	**35.9**	**33.0**	**45.5**	**45.5**	45.5	**79.1**	**47.3**	**43.7**	**36.5**	**37.9**	**74.6**	**96.5**
PriorNet	35.5	26.8	26.7	26.9	29.6	43.7	**68.7**	35.5	27.5	26.9	26.7	26.6	26.5	26.5
DDNet	52.9	31.7	29.8	29.1	28.4	30.1	37.6	52.9	38.4	31.5	27.5	26.8	26.6	26.5
EvNet	66.3	33.3	29.6	27.0	27.2	29.3	35.2	66.3	39.3	37.1	31.3	28.3	28.4	29.7
Seg. – Seg. class sky														
PostNet	98.0	76.3	53.1	**37.4**	**32.9**	**44.6**	**50.2**	98.0	83.5	64.8	41.8	35.4	**43.1**	**71.3**
PriorNet	32.3	30.9	30.8	30.8	30.8	32.5	45.0	32.3	31.0	30.8	30.8	30.8	30.8	30.8
DDNet	30.9	30.8	30.8	30.8	30.8	30.8	30.8	30.9	30.8	30.8	30.8	30.8	30.8	30.8
EvNet	**98.1**	**82.1**	**59.1**	33.8	31.0	30.8	30.8	**98.1**	**93.8**	**88.2**	**64.5**	**36.4**	31.3	31.0

Table D.19: OOD detection (AUC-ROC) under PGD uncertainty attacks against distributional uncertainty on ID data and OOD data.

	ID-Attack (non-attacked OOD)							OOD-Attack (non-attacked ID)						
Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
MNIST – KMNIST														
PostNet	90.1	88.0	86.2	82.2	79.0	77.1	**66.1**	90.1	84.5	77.2	46.4	12.9	2.7	2.4
PriorNet	**99.8**	**99.7**	**99.6**	**99.1**	**97.2**	**79.7**	4.7	**99.8**	**99.7**	**99.6**	**99.2**	**97.9**	**93.7**	**85.6**
DDNet	98.1	97.7	97.2	94.8	87.0	48.7	3.0	98.1	97.8	97.3	95.8	92.3	83.3	63.3
EvNet	36.8	35.0	33.7	30.9	28.2	25.3	22.1	36.8	19.3	10.7	3.9	2.1	1.8	2.2
CIFAR10 – SVHN														
PostNet	82.9	67.7	59.2	**51.3**	**47.7**	**40.1**	**24.2**	82.9	51.9	26.2	8.9	9.5	**11.1**	**9.9**
PriorNet	48.0	33.6	22.5	7.1	1.6	0.3	0.1	48.0	34.8	24.0	6.7	1.6	0.6	0.2
DDNet	**85.9**	**74.9**	**62.7**	30.1	8.3	2.3	0.6	**85.9**	**77.6**	**66.9**	**32.1**	8.0	1.5	0.2
EvNet	63.3	51.4	47.1	39.3	32.1	24.9	17.9	63.3	31.1	24.4	17.7	**15.0**	10.7	5.7
Sens. – Sens. class 10, 11														
PostNet	**87.1**	**50.9**	**37.8**	5.5	4.5	4.5	4.5	**87.1**	**55.3**	**51.1**	**34.4**	**38.9**	**79.7**	**97.9**
PriorNet	36.5	2.9	1.8	1.8	5.2	**21.5**	**52.8**	36.5	7.3	3.0	1.3	0.5	0.1	0.0
DDNet	52.3	18.7	13.1	**10.3**	**9.3**	10.8	18.4	52.3	33.1	22.0	6.7	2.2	0.6	0.1
EvNet	75.5	30.7	18.1	5.8	1.6	0.6	0.8	75.5	47.7	41.8	23.8	10.3	10.2	15.8
Seg. – Seg. class sky														
PostNet	**98.6**	**78.3**	**51.9**	**20.5**	**8.3**	**2.1**	1.7	**98.6**	88.8	73.1	35.9	**21.4**	**39.9**	**75.9**
PriorNet	9.4	1.6	0.3	0.0	0.0	1.8	**15.4**	9.4	2.4	0.4	0.0	0.0	0.0	0.0
DDNet	1.3	0.2	0.0	0.0	0.0	0.0	0.0	1.3	0.2	0.0	0.0	0.0	0.0	0.0
EvNet	97.4	77.1	45.9	9.4	1.3	0.2	0.1	97.4	**92.9**	**86.1**	**60.9**	20.4	3.0	1.2

Table D.20: OOD detection (AU-PR) under FGSM uncertainty attacks against differential entropy on ID data and OOD data.

	ID-Attack (non-attacked OOD)							OOD-Attack (non-attacked ID)						
Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
MNIST – KMNIST														
PostNet	94.5	94.2	94.1	93.5	89.9	81.2	**71.6**	94.5	93.3	92.0	87.6	81.1	75.7	75.7
PriorNet	**99.6**	**99.4**	**99.2**	**98.1**	**95.6**	**90.0**	65.3	**99.6**	**99.4**	**99.2**	**98.6**	97.5	**95.9**	**94.4**
DDNet	99.3	99.1	98.9	98.0	95.4	80.9	48.2	99.3	99.2	99.0	98.5	**97.6**	95.5	92.0
EvNet	69.0	67.4	66.2	64.0	61.9	59.8	56.70	9.0	60.1	56.5	53.4	52.7	52.9	53.5
CIFAR10 – SVHN														
PostNet	81.8	66.2	61.6	**64.2**	**65.7**	61.3	48.4	81.8	63.1	51.9	43.4	46.6	**61.7**	**77.0**
PriorNet	54.4	40.6	33.8	27.0	25.5	27.2	35.5	54.4	42.3	36.8	30.6	28.3	29.5	32.1
DDNet	**82.8**	**71.9**	**64.6**	53.8	50.2	47.8	41.0	**82.8**	**71.5**	**60.5**	39.1	31.4	41.2	66.6
EvNet	80.3	67.8	64.0	61.9	61.6	57.4	**49.6**	80.3	59.2	51.5	**46.7**	**49.0**	56.3	64.6
Sens. – Sens. class 10, 11														
PostNet	**74.5**	40.6	37.2	31.4	38.1	44.9	45.9	**74.5**	**99.6**	**99.8**	**99.9**	**99.9**	**99.9**	**99.9**
PriorNet	32.3	35.7	**57.6**	**83.1**	**88.8**	79.7	70.0	32.3	28.3	28.1	27.6	28.0	32.7	38.5
DDNet	31.7	31.3	44.4	70.3	87.9	**92.5**	**91.9**	31.7	28.8	29.3	29.1	27.7	27.9	28.01
EvNet	66.5	**45.7**	46.8	42.3	42.0	41.4	41.8	66.5	54.7	66.5	76.2	71.1	75.3	75.8
Seg. – Seg. class sky														
PostNet	**99.0**	**80.8**	**66.4**	43.6	37.0	35.5	43.0	**99.0**	**94.8**	**92.0**	**98.5**	**99.7**	**100.0**	**100.0**
PriorNet	34.8	31.2	31.4	46.3	**74.0**	**88.8**	**94.5**	34.8	31.6	31.0	31.2	30.9	30.8	30.8
DDNet	31.5	30.8	30.8	30.9	37.9	56.2	84.3	31.5	30.9	30.8	30.8	30.8	30.8	30.8
EvNet	92.5	64.9	54.6	**66.6**	69.5	69.6	64.6	92.5	85.9	83.0	66.3	66.1	61.1	56.8

Table D.21: OOD detection (AU-PR) under Noise uncertainty attacks against differential entropy on ID data and OOD data.

	ID-Attack (non-attacked OOD)							OOD-Attack (non-attacked ID)						
Noise Std	0.0	0.1	0.2	0.5	1.0	2.0	4.0	0.0	0.1	0.2	0.5	1.0	2.0	4.0
MNIST – KMNIST														
PostNet	93.0	94.2	82.3	34.4	31.6	31.0	30.9	92.2	91.8	91.5	92.3	92.7	93.2	93.5
PriorNet	**99.7**	**99.6**	**96.7**	**40.0**	**40.6**	**45.7**	**55.6**	**99.5**	97.3	96.5	99.4	**100.0**	99.5	72.4
DDNet	99.1	97.5	81.2	31.3	31.0	30.9	31.2	99.0	**98.8**	**99.2**	**99.8**	99.9	**99.8**	**99.1**
EvNet	65.5	60.5	51.4	35.3	34.5	35.5	35.0	62.5	47.2	40.9	35.1	34.6	33.5	34.9
CIFAR10 – SVHN														
PostNet	88.5	41.4	39.8	31.0	30.7	31.6	33.9	88.5	**86.6**	**81.9**	**93.**0	**98.5**	98.6	97.3
PriorNet	73.3	88.3	**95.3**	**92.4**	**70.4**	30.9	30.8	73.3	31.6	30.9	31.7	51.8	94.3	**100.0**
DDNet	87.3	69.3	78.4	55.2	31.6	30.7	31.4	87.3	55.8	57.9	73.9	97.3	**99.5**	97.2
EvNet	**92.4**	**56.8**	53.8	33.4	30.9	**32.9**	**36.6**	**92.4**	73.7	73.5	77.7	93.7	92.5	92.1
Sens. – Sens. class 10, 11														
PostNet	**85.3**	**30.8**	**39.4**	50.0	50.0	50.0	50.0	**85.3**	**98.9**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
PriorNet	32.3	**30.8**	34.9	**83.7**	**77.7**	49.8	**80.3**	32.3	30.7	30.7	32.5	40.1	49.9	47.6
DDNet	31.1	30.7	30.7	32.4	58.8	**88.1**	74.3	31.1	30.7	30.7	30.7	30.8	31.6	39.1
EvNet	80.3	**30.8**	31.2	37.9	46.3	50.0	50.0	80.3	34.6	38.4	53.9	69.3	78.8	81.5
Seg. – Seg. class sky														
PostNet	**99.9**	**41.8**	30.8	**34.5**	**49.1**	50.0	50.0	**99.9**	**97.4**	**96.6**	**99.5**	**100.0**	**100.0**	**100.0**
PriorNet	31.0	30.8	30.8	30.8	32.7	**69.0**	78.3	31.0	30.8	30.8	30.8	30.9	31.1	32.4
DDNet	30.8	30.8	30.8	30.8	30.8	58.2	**91.3**	30.8	30.8	30.8	30.8	30.8	30.8	31.9
EvNet	99.1	38.1	**32.2**	30.8	30.8	32.2	37.5	99.1	95.6	87.6	58.0	44.9	46.6	53.8

D.1.4 How to make DBU models more robust

To improve robustness of DBU models we perform median smoothing and adversarial training. Smoothing computes the smooth median, worst case and best case performance of DBU models for three tasks: distinguishing between correct and wrong predictions, attack detection, distinguishing between ID data and OOD data under label attacks and under uncertainty attacks.

Table D.22: Distinguishing between correctly and wrongly labeled inputs based on differential entropy under PGD label attacks. Smoothed DBU models on CIFAR10. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	80.5 · **91.5** · 94.5	52.8 · **71.6** · 95.2	31.9 · **51.0** · 96.8	5.6 · **11.7** · 100.0	0.3 · **0.6** · 100.0	0.0 · **0.0** · 100.0
	PriorNet	81.9 · **86.8** · 88.0	69.6 · **78.0** · 90.1	50.9 · **65.8** · 89.4	36.5 · **59.9** · 97.0	24.3 · **39.3** · 100.0	9.2 · **17.9** · 100.0
	DDNet	65.9 · **81.2** · 83.0	55.8 · **70.5** · 87.2	37.8 · **56.8** · 88.1	10.1 · **21.9** · 94.3	0.9 · **1.6** · 99.6	0.0 · **0.0** · 100.0
	EvNet	76.3 · **90.2** · 91.7	54.7 · **74.3** · 95.7	31.6 · **51.5** · 94.5	5.8 · **11.9** · 86.9	1.9 · **7.0** · 100.0	1.1 · **4.0** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	52.1 · **71.8** · 95.6	31.2 · **47.9** · 96.1	7.8 · **14.7** · 98.6	1.8 · **4.4** · 100.0	0.3 · **0.5** · 100.0
	PriorNet	-	57.6 · **71.7** · 88.9	46.1 · **64.5** · 90.1	38.1 · **59.3** · 99.5	32.3 · **51.7** · 100.0	22.1 · **41.6** · 97.4
	DDNet	-	58.6 · **78.4** · 92.2	49.4 · **66.0** · 90.5	12.0 · **21.4** · 98.1	0.8 · **1.0** · 96.6	0.0 · **0.0** · 100.0
	EvNet	-	24.3 · **34.2** · 51.8	32.6 · **49.5** · 95.5	5.9 · **13.0** · 100.0	2.6 · **5.2** · 99.9	2.9 · **5.9** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	52.8 · **74.2** · 94.6	33.0 · **49.4** · 87.5	7.7 · **14.2** · 99.0	0.6 · **1.2** · 100.0	0.7 · **1.1** · 100.0
	PriorNet	-	50.6 · **68.1** · 88.6	44.4 · **66.1** · 96.0	35.1 · **57.4** · 98.4	18.4 · **32.2** · 100.0	15.2 · **29.3** · 100.0
	DDNet	-	68.8 · **84.4** · 93.2	45.1 · **60.8** · 86.8	12.3 · **22.0** · 91.0	0.8 · **1.7** · 87.0	0.0 · **0.0** · 100.0
	EvNet	-	54.2 · **73.7** · 96.1	30.5 · **50.0** · 99.5	7.1 · **13.9** · 100.0	3.7 · **8.7** · 75.2	3.3 · **5.8** · 100.0

Table D.23: Distinguishing between correctly and wrongly labeled inputs based on differential entropy under PGD label attacks. Smoothed DBU models on MNIST. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	97.2 · **99.4** · 100.0	95.9 · **99.1** · 99.9	94.7 · **98.9** · 99.9	89.3 · **96.8** · 99.9	75.5 · **90.2** · 100.0	35.5 · **56.7** · 100.0
	PriorNet	96.8 · **99.2** · 99.3	95.5 · **99.1** · 99.7	94.6 · **98.8** · 99.7	90.2 · **97.2** · 99.9	81.1 · **93.4** · 99.9	53.9 · **75.2** · 100.0
	DDNet	97.6 · **99.4** · 99.5	96.8 · **99.2** · 99.4	95.5 · **98.8** · 99.4	90.4 · **97.2** · 99.8	77.0 · **91.3** · 100.0	29.2 · **48.6** · 100.0
	EvNet	97.3 · **99.4** · 99.4	95.4 · **98.8** · 99.6	93.9 · **98.7** · 99.9	89.0 · **96.5** · 100.0	78.9 · **92.9** · 100.0	52.2 · **73.2** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	94.4 · **98.6** · 99.5	90.6 · **97.9** · 99.9	83.4 · **93.1** · 99.9	72.1 · **91.2** · 100.0	41.8 · **65.0** · 100.0
	PriorNet	-	94.4 · **98.5** · 99.5	93.6 · **98.8** · 99.8	89.1 · **96.6** · 99.8	81.5 · **94.5** · 100.0	71.6 · **88.4** · 100.0
	DDNet	-	94.9 · **98.3** · 98.7	94.6 · **97.9** · 98.9	88.2 · **97.4** · 99.8	72.1 · **89.3** · 100.0	28.1 · **49.3** · 100.0
	EvNet	-	88.8 · **95.3** · 97.9	91.5 · **97.1** · 99.4	85.2 · **94.9** · 100.0	78.1 · **91.4** · 100.0	54.3 · **75.3** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	92.8 · **98.3** · 99.8	92.5 · **98.3** · 99.9	86.2 · **94.8** · 99.8	71.0 · **89.5** · 100.0	34.6 · **54.2** · 100.0
	PriorNet	-	95.1 · **98.6** · 99.6	94.1 · **98.0** · 99.4	87.7 · **97.2** · 99.9	80.2 · **93.4** · 100.0	68.5 · **87.8** · 100.0
	DDNet	-	96.0 · **98.4** · 98.8	95.0 · **97.6** · 98.7	87.6 · **95.3** · 99.7	73.9 · **90.2** · 100.0	32.8 · **54.4** · 100.0
	EvNet	-	93.3 · **98.6** · 99.5	89.8 · **97.2** · 99.2	86.2 · **95.4** · 100.0	82.1 · **93.7** · 100.0	52.4 · **73.3** · 100.0

Table D.24: Distinguishing between correctly and wrongly labeled inputs based on differential entropy under PGD label attacks. Smoothed DBU models on Sensorless. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	93.5 · **98.4** · 100.0	6.7 · **12.4** · 100.0	2.9 · **5.3** · 100.0	4.1 · **4.1** · 49.1	6.4 · **6.4** · 6.4	10.6 · **10.6** · 10.6
	PriorNet	97.1 · **99.3** · 100.0	8.6 · **17.6** · 100.0	3.3 · **7.7** · 100.0	0.7 · **1.5** · 100.0	0.4 · **0.7** · 100.0	0.1 · **0.2** · 100.0
	DDNet	95.9 · **98.9** · 99.7	7.0 · **14.0** · 100.0	0.8 · **1.3** · 100.0	0.2 · **0.4** · 100.0	0.2 · **0.2** · 100.0	0.2 · **0.4** · 100.0
	EvNet	94.0 · **99.0** · 99.9	18.1 · **34.2** · 100.0	9.6 · **17.1** · 100.0	4.1 · **6.8** · 100.0	2.7 · **4.9** · 100.0	2.4 · **4.3** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	7.9 · **14.9** · 100.0	2.9 · **6.3** · 100.0	6.6 · **6.6** · 6.6	7.2 · **7.2** · 7.2	9.6 · **9.6** · 9.6
	PriorNet	-	18.1 · **32.1** · 100.0	8.7 · **16.7** · 100.0	0.1 · **0.2** · 100.0	0.0 · **0.0** · 100.0	0.8 · **1.0** · 100.0
	DDNet	-	6.9 · **13.4** · 100.0	4.3 · **9.0** · 100.0	0.2 · **0.3** · 100.0	0.2 · **0.4** · 100.0	0.2 · **0.8** · 100.0
	EvNet	-	19.7 · **35.7** · 100.0	9.4 · **16.2** · 100.0	1.6 · **3.0** · 100.0	2.5 · **5.6** · 100.0	1.0 · **1.8** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	7.9 · **14.4** · 100.0	4.8 · **9.3** · 100.0	6.6 · **6.6** · 6.6	6.7 · **6.7** · 6.7	10.6 · **10.6** · 10.6
	PriorNet	-	19.1 · **32.7** · 100.0	6.9 · **13.7** · 100.0	0.7 · **1.7** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	DDNet	-	5.4 · **10.2** · 100.0	0.7 · **1.8** · 100.0	0.5 · **0.9** · 100.0	0.3 · **1.2** · 100.0	0.2 · **0.6** · 100.0
	EvNet	-	22.3 · **38.4** · 100.0	11.7 · **22.4** · 100.0	7.1 · **13.1** · 100.0	1.8 · **3.4** · 100.0	0.6 · **1.0** · 100.0

Table D.25: Distinguishing between correctly and wrongly labeled inputs based on differential entropy under PGD label attacks. Smoothed DBU models on Segment. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model)..

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	94.0 · **99.1** · 99.8	63.5 · **84.7** · 100.0	33.2 · **56.1** · 100.0	10.2 · **16.9** · 100.0	5.2 · **10.3** · 100.0	0.3 · **0.3** · 0.3
	PriorNet	97.0 · **99.8** · 99.9	75.6 · **90.8** · 100.0	31.1 · **50.8** · 100.0	2.6 · **4.7** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	DDNet	96.2 · **99.5** · 99.7	75.7 · **89.8** · 99.9	28.5 · **51.6** · 100.0	3.7 · **8.2** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	EvNet	95.8 · **99.6** · 99.9	80.2 · **93.7** · 100.0	35.2 · **57.2** · 100.0	6.8 · **12.0** · 100.0	1.2 · **2.1** · 100.0	1.1 · **2.0** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	66.0 · **85.5** · 100.0	22.5 · **41.0** · 100.0	9.0 · **16.3** · 100.0	5.2 · **9.7** · 100.0	0.6 · **0.6** · 0.6
	PriorNet	-	79.0 · **92.4** · 100.0	45.2 · **68.8** · 100.0	9.2 · **13.9** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	DDNet	-	76.2 · **91.0** · 99.6	27.2 · **45.3** · 100.0	2.3 · **4.3** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	EvNet	-	82.7 · **95.2** · 100.0	34.0 · **53.8** · 100.0	10.9 · **23.2** · 100.0	0.5 · **4.2** · 100.0	2.1 · **5.1** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	71.5 · **87.6** · 100.0	33.5 · **54.5** · 100.0	12.8 · **25.6** · 100.0	6.5 · **10.3** · 87.2	0.0 · **0.0** · 100.0
	PriorNet	-	82.1 · **96.5** · 100.0	44.1 · **65.4** · 100.0	9.0 · **15.7** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	DDNet	-	77.4 · **91.4** · 99.9	29.4 · **50.3** · 100.0	4.0 · **6.5** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	EvNet	-	76.2 · **90.7** · 100.0	35.7 · **55.4** · 100.0	4.2 · **6.4** · 100.0	0.8 · **1.4** · 100.0	0.0 · **0.0** · 100.0

Table D.26: Distinguishing between correctly and wrongly labeled inputs based on differential entropy under FGSM label attacks. Smoothed DBU models on CIFAR10. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	80.5 · **91.4** · 94.4	52.3 · **73.2** · 95.4	35.8 · **57.2** · 97.5	17.0 · **29.0** · 100.0	10.2 · **18.7** · 100.0	8.1 · **14.7** · 100.0
	PriorNet	81.9 · **87.7** · 88.8	69.6 · **78.4** · 90.3	53.3 · **70.5** · 91.7	42.1 · **62.6** · 97.2	37.5 · **55.7** · 100.0	36.0 · **59.5** · 100.0
	DDNet	65.9 · **84.1** · 85.6	55.3 · **69.6** · 87.0	38.6 · **55.8** · 87.2	16.3 · **28.5** · 94.6	6.4 · **12.0** · 99.9	3.6 · **7.2** · 100.0
	EvNet	76.3 · **90.4** · 91.7	54.1 · **74.5** · 95.5	35.5 · **54.7** · 95.1	14.6 · **29.3** · 95.6	8.6 · **16.1** · 100.0	7.2 · **13.0** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	52.3 · **71.6** · 95.1	34.7 · **54.8** · 96.6	18.9 · **32.1** · 99.4	10.9 · **19.2** · 100.0	8.5 · **16.2** · 100.0
	PriorNet	-	58.1 · **69.6** · 87.6	47.1 · **65.7** · 90.3	40.2 · **59.5** · 99.3	36.2 · **59.5** · 100.0	25.1 · **42.1** · 97.7
	DDNet	-	57.1 · **75.2** · 91.0	49.3 · **65.3** · 90.5	18.4 · **33.6** · 98.5	7.6 · **13.5** · 99.9	3.3 · **9.6** · 100.0
	EvNet	-	24.1 · **36.5** · 54.2	37.1 · **56.7** · 96.7	16.2 · **29.9** · 100.0	11.4 · **21.8** · 100.0	13.0 · **26.1** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	52.0 · **71.8** · 94.5	35.8 · **54.6** · 89.9	18.4 · **33.6** · 99.8	10.2 · **19.1** · 100.0	12.2 · **23.0** · 100.0
	PriorNet	-	50.6 · **67.3** · 88.5	46.2 · **64.3** · 95.1	39.9 · **60.8** · 98.5	27.7 · **46.2** · 100.0	28.5 · **48.6** · 100.0
	DDNet	-	67.7 · **82.2** · 92.4	45.7 · **64.7** · 88.8	20.5 · **34.8** · 93.6	6.1 · **13.1** · 91.8	4.1 · **8.4** · 100.0
	EvNet	-	53.9 · **73.6** · 96.3	34.2 · **55.3** · 99.7	16.1 · **31.2** · 100.0	6.1 · **13.5** · 86.1	18.1 · **34.0** · 100.0

Table D.27: Distinguishing between correctly and wrongly labeled inputs based on differential entropy under FGSM label attacks. Smoothed DBU models on MNIST. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	97.2 · **99.3** · 99.9	96.1 · **99.2** · 99.9	95.2 · **98.9** · 99.9	91.7 · **98.0** · 99.9	86.1 · **95.9** · 100.0	75.7 · **91.1** · 100.0
	PriorNet	96.8 · **99.2** · 99.3	95.5 · **99.0** · 99.6	94.7 · **98.7** · 99.6	91.3 · **97.6** · 99.9	85.5 · **95.6** · 100.0	78.7 · **92.4** · 100.0
	DDNet	97.6 · **99.3** · 99.4	96.8 · **99.2** · 99.5	95.6 · **98.7** · 99.4	91.7 · **97.7** · 99.9	83.4 · **95.2** · 100.0	58.3 · **79.6** · 100.0
	EvNet	97.3 · **99.3** · 99.4	95.5 · **99.0** · 99.6	94.3 · **98.9** · 99.9	92.0 · **97.7** · 100.0	87.4 · **96.3** · 100.0	78.8 · **92.4** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	95.1 · **98.9** · 99.8	91.2 · **97.2** · 99.6	87.6 · **96.3** · 99.9	81.0 · **93.3** · 100.0	69.9 · **87.2** · 100.0
	PriorNet	-	94.4 · **98.7** · 99.7	93.6 · **98.2** · 99.3	89.4 · **96.3** · 99.8	84.5 · **95.1** · 100.0	81.7 · **92.5** · 100.0
	DDNet	-	95.5 · **98.6** · 99.0	94.6 · **98.7** · 99.4	89.7 · **97.1** · 99.8	80.0 · **93.6** · 100.0	54.4 · **74.5** · 100.0
	EvNet	-	88.9 · **94.8** · 98.1	91.5 · **98.4** · 99.8	89.2 · **97.0** · 100.0	83.6 · **94.7** · 100.0	72.3 · **88.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	92.8 · **98.5** · 99.9	92.8 · **98.7** · 99.9	89.0 · **96.3** · 99.8	80.8 · **93.4** · 100.0	71.6 · **86.9** · 100.0
	PriorNet	-	95.1 · **98.1** · 98.9	94.3 · **97.7** · 99.1	88.5 · **96.8** · 99.9	83.4 · **94.5** · 100.0	78.9 · **92.2** · 100.0
	DDNet	-	96.0 · **98.7** · 99.0	95.5 · **98.6** · 99.3	89.5 · **95.6** · 99.7	79.6 · **93.1** · 100.0	55.9 · **77.1** · 100.0
	EvNet	-	93.3 · **98.9** · 99.4	90.1 · **97.9** · 99.4	87.9 · **96.3** · 100.0	84.1 · **94.2** · 100.0	69.2 · **86.9** · 100.0

Table D.28: Distinguishing between correctly and wrongly labeled inputs based on differential entropy under FGSM label attacks. Smoothed DBU models on Sensorless. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	94.5 · **98.1** · 100.0	10.3 · **19.6** · 100.0	5.1 · **11.0** · 100.0	6.4 · **6.4** · 6.4	10.4 · **10.4** · 10.4	11.4 · **11.4** · 11.4
	PriorNet	97.1 · **99.5** · 100.0	13.6 · **27.3** · 100.0	6.3 · **12.4** · 100.0	2.6 · **6.8** · 100.0	3.1 · **7.3** · 100.0	3.2 · **6.7** · 100.0
	DDNet	95.9 · **99.4** · 99.8	8.6 · **14.9** · 100.0	1.6 · **3.8** · 100.0	2.6 · **4.5** · 100.0	3.4 · **6.9** · 100.0	3.3 · **6.4** · 100.0
	EvNet	94.0 · **98.5** · 99.7	26.0 · **43.2** · 100.0	15.8 · **30.8** · 100.0	11.7 · **20.2** · 100.0	8.1 · **15.0** · 100.0	7.6 · **12.7** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	13.1 · **24.3** · 100.0	5.7 · **11.9** · 100.0	9.4 · **9.4** · 9.4	11.2 · **11.2** · 11.2	11.8 · **11.8** · 11.8
	PriorNet	-	22.4 · **38.2** · 100.0	11.8 · **22.1** · 100.0	0.2 · **0.6** · 100.0	0.0 · **0.0** · 100.0	0.1 · **0.1** · 100.0
	DDNet	-	7.3 · **13.2** · 100.0	8.5 · **17.2** · 100.0	3.6 · **7.9** · 100.0	3.8 · **7.6** · 100.0	0.8 · **1.2** · 100.0
	EvNet	-	25.5 · **42.0** · 100.0	15.6 · **30.2** · 100.0	10.4 · **19.5** · 100.0	8.6 · **16.4** · 100.0	7.8 · **14.7** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	10.6 · **20.3** · 100.0	5.2 · **9.9** · 100.0	10.9 · **10.9** · 10.9	11.6 · **11.6** · 11.6	11.7 · **11.7** · 11.7
	PriorNet	-	25.7 · **45.0** · 100.0	12.0 · **20.5** · 100.0	1.1 · **3.7** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	DDNet	-	7.9 · **16.4** · 100.0	1.2 · **3.8** · 100.0	3.4 · **6.3** · 100.0	3.9 · **7.9** · 100.0	3.3 · **8.0** · 100.0
	EvNet	-	27.9 · **49.2** · 100.0	18.4 · **32.9** · 100.0	16.4 · **29.3** · 100.0	5.9 · **10.8** · 100.0	8.5 · **16.1** · 100.0

Table D.29: Distinguishing between correctly and wrongly labeled inputs based on differential entropy under FGSM label attacks. Smoothed DBU models on Segment. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	94.0 · **99.2** · 99.8	55.2 · **78.3** · 100.0	40.1 · **61.4** · 100.0	17.9 · **31.7** · 100.0	6.8 · **12.7** · 100.0	17.6 · **17.9** · 18.0
	PriorNet	97.0 · **99.8** · 99.9	69.2 · **89.7** · 100.0	29.7 · **45.5** · 100.0	1.7 · **4.1** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	DDNet	96.2 · **99.5** · 99.6	70.6 · **86.3** · 99.8	22.3 · **38.8** · 100.0	6.3 · **13.3** · 100.0	1.1 · **3.0** · 100.0	0.0 · **0.0** · 100.0
	EvNet	95.8 · **99.1** · 99.8	78.4 · **92.5** · 100.0	40.7 · **62.1** · 100.0	9.8 · **17.6** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	66.0 · **83.5** · 100.0	28.8 · **44.9** · 100.0	12.3 · **24.3** · 100.0	9.3 · **17.3** · 100.0	24.8 · **24.8** · 24.8
	PriorNet	-	75.1 · **91.5** · 99.9	34.0 · **60.3** · 100.0	11.1 · **24.6** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	DDNet	-	65.4 · **82.8** · 99.5	23.1 · **35.3** · 100.0	4.8 · **10.4** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	EvNet	-	83.4 · **95.3** · 100.0	42.1 · **63.3** · 100.0	15.0 · **33.6** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	67.8 · **86.5** · 100.0	34.0 · **52.5** · 100.0	16.2 · **32.8** · 100.0	14.4 · **25.2** · 92.2	7.3 · **7.3** · 7.3
	PriorNet	-	77.3 · **91.2** · 99.9	39.3 · **62.7** · 100.0	9.0 · **17.8** · 100.0	0.0 · **0.0** · 100.0	0.0 · **0.0** · 100.0
	DDNet	-	68.8 · **88.3** · 99.9	20.4 · **35.2** · 100.0	7.5 · **12.6** · 100.0	0.3 · **0.9** · 100.0	0.0 · **0.0** · 100.0
	EvNet	-	74.0 · **92.9** · 100.0	44.1 · **61.8** · 100.0	5.3 · **13.0** · 100.0	3.9 · **8.2** · 100.0	0.5 · **4.2** · 100.0

Table D.30: Attack detection (PGD label attacks) based on differential entropy. Smoothed DBU models on CIFAR10. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	33.1 · **50.4** · 89.9	31.0 · **50.2** · 96.9	30.7 · **50.2** · 100.0	30.7 · **50.0** · 100.0	30.7 · **50.2** · 100.0
	PriorNet	35.9 · **50.6** · 74.5	33.0 · **50.3** · 82.8	31.2 · **50.0** · 95.7	30.7 · **50.4** · 99.9	30.7 · **50.4** · 100.0
	DDNet	36.3 · **50.3** · 76.4	32.8 · **49.9** · 84.6	30.8 · **50.1** · 98.0	30.7 · **50.2** · 100.0	30.7 · **50.2** · 100.0
	EvNet	32.9 · **50.4** · 89.8	31.4 · **50.1** · 94.0	30.8 · **50.0** · 98.0	30.7 · **50.3** · 100.0	30.7 · **49.6** · 100.0
Smoothed + adv. label attacks	**PostNet**	32.7 · **50.1** · 90.4	31.1 · **50.2** · 96.5	30.7 · **50.2** · 99.7	30.7 · **50.3** · 100.0	30.7 · **50.2** · 100.0
	PriorNet	35.2 · **51.8** · 78.6	32.8 · **51.1** · 84.4	30.8 · **50.2** · 98.7	30.7 · **50.5** · 100.0	30.8 · **50.1** · 98.2
	DDNet	35.5 · **50.6** · 79.2	33.4 · **50.3** · 84.1	30.8 · **50.1** · 99.2	30.7 · **50.0** · 100.0	30.7 · **50.5** · 100.0
	EvNet	40.3 · **50.4** · 66.8	31.4 · **50.3** · 95.8	30.7 · **50.3** · 100.0	30.7 · **50.1** · 100.0	30.7 · **50.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	33.3 · **50.6** · 88.7	32.5 · **50.1** · 87.9	30.7 · **49.9** · 99.8	30.7 · **50.1** · 100.0	30.7 · **50.0** · 100.0
	PriorNet	34.5 · **51.0** · 80.1	31.4 · **50.6** · 92.8	30.9 · **50.0** · 97.7	30.7 · **50.1** · 100.0	30.7 · **50.0** · 100.0
	DDNet	37.4 · **50.8** · 74.5	33.4 · **50.2** · 83.0	30.9 · **50.1** · 96.8	30.8 · **49.9** · 98.1	30.7 · **49.9** · 100.0
	EvNet	32.8 · **50.1** · 92.0	30.8 · **50.0** · 99.6	30.7 · **50.1** · 100.0	31.2 · **50.2** · 96.1	31.0 · **50.0** · 100.0

Table D.31: Attack detection (PGD label attacks) based on differential entropy. Smoothed DBU models on MNIST. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	30.9 · **52.5** · 95.6	31.5 · **51.5** · 90.9	31.1 · **49.9** · 97.1	30.7 · **47.6** · 100.0	30.7 · **45.0** · 100.0
	PriorNet	38.2 · **57.8** · 80.9	36.0 · **57.2** · 84.3	31.6 · **63.4** · 98.4	30.8 · **61.0** · 99.3	30.7 · **66.8** · 100.0
	DDNet	44.6 · **51.9** · 60.7	39.3 · **52.7** · 72.2	31.6 · **50.9** · 95.2	30.7 · **47.3** · 100.0	30.7 · **45.9** · 100.0
	EvNet	36.5 · **51.8** · 76.1	31.5 · **51.1** · 93.2	30.7 · **51.1** · 99.9	30.7 · **48.7** · 100.0	30.7 · **43.8** · 100.0
Smoothed + adv. label attacks	**PostNet**	33.6 · **52.8** · 82.3	31.4 · **51.2** · 91.6	30.9 · **49.4** · 99.1	30.7 · **49.3** · 100.0	30.7 · **56.0** · 100.0
	PriorNet	37.3 · **60.5** · 84.3	34.3 · **59.9** · 87.9	32.1 · **61.0** · 97.0	30.7 · **69.3** · 100.0	30.7 · **68.0** · 100.0
	DDNet	44.8 · **52.2** · 61.0	40.2 · **52.6** · 70.0	32.5 · **52.4** · 94.6	30.7 · **50.3** · 100.0	30.7 · **54.6** · 100.0
	EvNet	35.8 · **51.2** · 76.7	32.9 · **51.0** · 88.5	30.7 · **49.5** · 100.0	30.7 · **48.5** · 100.0	30.7 · **47.7** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	31.2 · **52.7** · 92.8	31.3 · **51.7** · 92.4	31.3 · **47.3** · 96.8	30.7 · **48.9** · 100.0	30.7 · **46.3** · 100.0
	PriorNet	38.3 · **58.2** · 81.5	36.9 · **55.5** · 79.9	31.3 · **63.5** · 98.9	30.7 · **68.6** · 100.0	30.7 · **74.6** · 100.0
	DDNet	44.9 · **52.2** · 60.7	39.6 · **53.3** · 72.1	31.8 · **51.7** · 95.4	30.7 · **46.1** · 100.0	30.7 · **46.0** · 100.0
	EvNet	38.8 · **51.9** · 70.9	34.5 · **52.3** · 82.9	30.8 · **49.9** · 99.6	30.7 · **47.7** · 100.0	30.8 · **49.4** · 100.0

Table D.32: Attack detection (PGD label attacks) based on differential entropy. Smoothed DBU models on Sensorless. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	30.7 · **61.9** · 100.0	30.7 · **60.1** · 100.0	46.5 · **50.0** · 75.5	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	30.7 · **50.1** · 100.0	30.7 · **46.5** · 100.0	30.7 · **42.3** · 100.0	30.7 · **66.7** · 100.0	30.9 · **79.2** · 100.0
	DDNet	30.7 · **57.5** · 100.0	30.7 · **49.9** · 100.0	30.7 · **45.5** · 100.0	30.7 · **50.0** · 100.0	30.7 · **59.3** · 100.0
	EvNet	30.7 · **62.0** · 100.0	30.7 · **59.6** · 100.0	30.7 · **55.8** · 100.0	30.7 · **48.3** · 100.0	31.8 · **50.0** · 100.0
Smoothed + adv. label attacks	**PostNet**	30.7 · **58.8** · 100.0	30.7 · **58.2** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	30.7 · **60.2** · 100.0	30.7 · **54.6** · 100.0	30.7 · **45.0** · 100.0	30.7 · **38.0** · 100.0	33.9 · **49.9** · 100.0
	DDNet	30.7 · **55.4** · 100.0	30.7 · **53.7** · 100.0	30.7 · **44.6** · 100.0	30.7 · **38.8** · 100.0	30.7 · **51.9** · 100.0
	EvNet	30.7 · **62.1** · 100.0	30.7 · **54.3** · 100.0	30.7 · **59.9** · 100.0	30.7 · **62.1** · 100.0	30.7 · **50.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	30.7 · **63.0** · 100.0	30.7 · **54.0** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	30.7 · **58.0** · 100.0	30.7 · **55.6** · 100.0	30.7 · **44.2** · 100.0	30.7 · **53.5** · 100.0	30.7 · **78.5** · 100.0
	DDNet	30.7 · **55.1** · 100.0	30.7 · **48.2** · 100.0	30.7 · **50.1** · 100.0	30.7 · **52.6** · 100.0	30.7 · **57.0** · 100.0
	EvNet	30.7 · **63.5** · 100.0	30.7 · **54.3** · 100.0	30.7 · **54.2** · 100.0	30.7 · **45.0** · 100.0	30.7 · **50.0** · 100.0

Table D.33: Attack detection (PGD label attacks) based on differential entropy. Smoothed DBU models on Segment. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	30.8 · **73.5** · 100.0	30.8 · **59.9** · 100.0	30.8 · **60.3** · 100.0	30.8 · **50.2** · 100.0	49.5 · **50.0** · 50.0
	PriorNet	30.9 · **77.1** · 99.9	30.8 · **78.1** · 100.0	30.8 · **39.5** · 100.0	30.8 · **35.2** · 100.0	30.8 · **41.4** · 100.0
	DDNet	31.4 · **69.6** · 99.5	30.8 · **71.2** · 100.0	30.8 · **54.3** · 100.0	30.8 · **35.5** · 100.0	30.8 · **35.7** · 100.0
	EvNet	30.8 · **86.2** · 100.0	30.8 · **80.3** · 100.0	30.8 · **54.0** · 100.0	30.8 · **43.3** · 100.0	30.8 · **40.5** · 100.0
Smoothed + adv. label attacks	**PostNet**	30.8 · **75.6** · 100.0	30.8 · **69.7** · 100.0	30.8 · **66.5** · 100.0	30.8 · **50.0** · 100.0	50.0 · **50.0** · 50.0
	PriorNet	31.0 · **74.4** · 99.2	30.8 · **74.0** · 100.0	30.8 · **59.8** · 100.0	30.8 · **56.0** · 100.0	30.8 · **38.8** · 100.0
	DDNet	31.6 · **68.9** · 99.0	30.8 · **72.9** · 100.0	30.8 · **47.5** · 100.0	30.8 · **32.2** · 100.0	30.8 · **31.8** · 100.0
	EvNet	30.8 · **83.4** · 100.0	30.8 · **87.0** · 100.0	30.8 · **61.9** · 100.0	30.8 · **39.2** · 100.0	30.8 · **41.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	30.8 · **73.9** · 100.0	30.8 · **64.5** · 100.0	30.8 · **68.3** · 100.0	33.0 · **50.0** · 100.0	50.0 · **50.0** · 50.0
	PriorNet	31.0 · **73.7** · 99.6	30.8 · **73.1** · 100.0	30.8 · **57.8** · 100.0	30.8 · **44.8** · 100.0	30.8 · **49.1** · 100.0
	DDNet	31.0 · **70.7** · 99.7	30.8 · **70.6** · 100.0	30.8 · **48.6** · 100.0	30.8 · **31.6** · 100.0	30.8 · **30.9** · 100.0
	EvNet	30.8 · **85.8** · 100.0	30.8 · **86.7** · 100.0	30.8 · **54.4** · 100.0	30.8 · **45.1** · 100.0	30.8 · **34.8** · 100.0

Table D.34: Attack detection (FGSM label attacks) based on differential entropy. Smoothed DBU models on CIFAR10. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	33.1 · **50.3** · 89.9	31.0 · **50.2** · 96.9	30.7 · **50.1** · 100.0	30.7 · **49.5** · 100.0	30.7 · **50.2** · 100.0
	PriorNet	36.0 · **50.8** · 74.6	33.0 · **50.4** · 82.8	31.2 · **50.2** · 95.6	30.7 · **50.7** · 99.9	30.7 · **51.4** · 100.0
	DDNet	36.4 · **50.4** · 76.4	32.8 · **49.9** · 84.6	30.8 · **50.1** · 97.9	30.7 · **50.2** · 100.0	30.7 · **49.9** · 100.0
	EvNet	32.9 · **50.3** · 89.7	31.4 · **50.2** · 94.0	30.8 · **50.1** · 98.0	30.7 · **49.7** · 100.0	30.7 · **49.7** · 100.0
Smoothed + adv. label attacks	**PostNet**	32.7 · **50.1** · 90.3	31.1 · **50.3** · 96.4	30.7 · **50.1** · 99.7	30.7 · **49.8** · 100.0	30.7 · **50.5** · 100.0
	PriorNet	35.4 · **52.3** · 78.9	32.9 · **51.3** · 84.5	30.7 · **50.3** · 98.7	30.7 · **50.7** · 100.0	30.8 · **50.2** · 98.2
	DDNet	35.5 · **50.6** · 79.3	33.4 · **50.3** · 84.2	30.8 · **50.1** · 99.2	30.7 · **49.9** · 100.0	30.7 · **50.1** · 100.0
	EvNet	40.3 · **50.4** · 66.8	31.4 · **50.3** · 95.9	30.7 · **50.2** · 100.0	30.7 · **50.1** · 100.0	30.7 · **49.6** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	33.3 · **50.7** · 88.7	32.5 · **50.1** · 87.8	30.7 · **50.1** · 99.8	30.7 · **50.5** · 100.0	30.7 · **50.2** · 100.0
	PriorNet	34.6 · **51.2** · 80.3	31.4 · **50.7** · 92.8	30.9 · **50.2** · 97.7	30.7 · **50.0** · 100.0	30.7 · **50.1** · 100.0
	DDNet	37.4 · **51.0** · 74.7	33.4 · **50.2** · 83.0	30.9 · **50.1** · 96.9	30.8 · **50.1** · 98.1	30.7 · **49.9** · 100.0
	EvNet	32.8 · **50.1** · 92.0	30.8 · **50.2** · 99.6	30.7 · **50.4** · 100.0	31.2 · **50.2** · 96.0	31.0 · **50.0** · 100.0

Table D.35: Attack detection (FGSM label attacks) based on differential entropy. Smoothed DBU models on MNIST. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	30.9 · **52.3** · 95.6	31.5 · **51.2** · 90.8	31.1 · **49.8** · 97.0	30.7 · **48.3** · 100.0	30.7 · **46.5** · 100.0
	PriorNet	38.1 · **57.7** · 80.8	35.8 · **56.6** · 84.0	31.5 · **61.7** · 98.3	30.8 · **58.9** · 99.2	30.7 · **62.3** · 100.0
	DDNet	44.7 · **52.0** · 60.9	39.4 · **52.9** · 72.5	31.6 · **50.8** · 95.2	30.7 · **47.5** · 100.0	30.7 · **46.8** · 100.0
	EvNet	36.5 · **51.7** · 76.0	31.5 · **51.1** · 93.2	30.7 · **50.9** · 99.9	30.7 · **48.9** · 100.0	30.7 · **46.2** · 100.0
Smoothed + adv. label attacks	**PostNet**	33.5 · **52.6** · 82.2	31.4 · **51.0** · 91.5	30.9 · **49.8** · 99.0	30.7 · **50.1** · 100.0	30.7 · **54.4** · 100.0
	PriorNet	37.3 · **60.6** · 84.3	34.2 · **59.5** · 87.8	32.1 · **60.0** · 96.9	30.7 · **66.3** · 100.0	30.7 · **63.3** · 100.0
	DDNet	44.9 · **52.3** · 61.0	40.3 · **52.8** · 70.2	32.5 · **52.4** · 94.6	30.7 · **50.0** · 100.0	30.7 · **57.1** · 100.0
	EvNet	35.8 · **51.5** · 76.7	32.9 · **50.9** · 88.5	30.7 · **50.0** · 100.0	30.7 · **48.9** · 100.0	30.7 · **48.6** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	31.2 · **52.6** · 92.9	31.3 · **51.5** · 92.3	31.3 · **48.0** · 96.9	30.7 · **49.4** · 100.0	30.7 · **48.1** · 100.0
	PriorNet	38.3 · **58.3** · 81.4	36.8 · **55.2** · 79.8	31.3 · **62.5** · 98.9	30.7 · **64.5** · 100.0	30.7 · **68.7** · 100.0
	DDNet	45.0 · **52.3** · 60.9	39.7 · **53.5** · 72.4	31.8 · **51.7** · 95.4	30.7 · **46.6** · 100.0	30.7 · **44.4** · 100.0
	EvNet	38.8 · **51.8** · 70.8	34.5 · **52.0** · 82.7	30.8 · **50.0** · 99.6	30.7 · **49.3** · 100.0	30.8 · **50.3** · 100.0

Table D.36: Attack detection (FGSM label attacks) based on differential entropy. Smoothed DBU models on Sensorless. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	30.7 · **82.0** · 100.0	30.7 · **88.6** · 100.0	50.0 · **50.0** · 50.1	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	30.7 · **51.7** · 100.0	30.7 · **48.2** · 100.0	30.7 · **48.6** · 100.0	30.7 · **68.6** · 100.0	31.4 · **63.7** · 100.0
	DDNet	30.7 · **67.1** · 100.0	30.7 · **58.2** · 100.0	30.7 · **51.7** · 100.0	30.7 · **69.8** · 100.0	30.7 · **73.7** · 100.0
	EvNet	30.7 · **77.9** · 100.0	30.7 · **85.3** · 100.0	30.7 · **90.5** · 100.0	30.8 · **84.3** · 100.0	34.0 · **50.0** · 100.0
Smoothed + adv. label attacks	**PostNet**	30.7 · **76.7** · 100.0	30.7 · **78.7** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	30.7 · **63.9** · 100.0	30.7 · **58.4** · 100.0	30.7 · **45.2** · 100.0	30.7 · **43.3** · 100.0	32.9 · **35.5** · 100.0
	DDNet	30.7 · **58.5** · 100.0	30.7 · **75.8** · 100.0	30.7 · **72.6** · 100.0	30.7 · **35.6** · 100.0	30.7 · **71.5** · 100.0
	EvNet	30.7 · **80.4** · 100.0	30.7 · **71.5** · 100.0	30.7 · **75.3** · 100.0	30.7 · **78.5** · 100.0	30.7 · **50.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	30.7 · **77.4** · 100.0	30.7 · **68.0** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	30.7 · **63.8** · 100.0	30.7 · **64.1** · 100.0	30.7 · **46.9** · 100.0	30.7 · **48.9** · 100.0	30.7 · **78.0** · 100.0
	DDNet	30.7 · **56.5** · 100.0	30.7 · **54.6** · 100.0	30.7 · **59.4** · 100.0	30.7 · **71.8** · 100.0	30.7 · **76.0** · 100.0
	EvNet	30.7 · **71.5** · 100.0	30.7 · **75.7** · 100.0	30.7 · **90.5** · 100.0	30.7 · **54.7** · 100.0	30.9 · **50.2** · 100.0

Table D.37: Attack detection (FGSM label attacks) based on differential entropy. Smoothed DBU models on Segment. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model)..

	Att. Rad.	0.1	0.2	0.5	1.0	2.0
Smoothed models	**PostNet**	30.8 · **76.9** · 100.0	30.8 · **62.5** · 100.0	30.8 · **59.2** · 100.0	30.8 · **48.7** · 100.0	49.7 · **50.0** · 50.0
	PriorNet	30.9 · **81.3** · 99.9	30.8 · **85.0** · 100.0	30.8 · **48.7** · 100.0	30.8 · **37.1** · 100.0	30.8 · **43.7** · 100.0
	DDNet	31.7 · **73.8** · 99.7	30.8 · **80.5** · 100.0	30.8 · **80.4** · 100.0	30.8 · **72.7** · 100.0	30.8 · **70.6** · 100.0
	EvNet	30.8 · **89.1** · 100.0	30.8 · **89.5** · 100.0	30.8 · **75.3** · 100.0	30.8 · **73.1** · 100.0	30.8 · **83.1** · 100.0
Smoothed + adv. label attacks	**PostNet**	30.8 · **81.0** · 100.0	30.8 · **75.6** · 100.0	30.8 · **56.3** · 100.0	30.8 · **50.0** · 100.0	50.0 · **50.0** · 50.0
	PriorNet	31.1 · **77.9** · 99.4	30.8 · **76.1** · 100.0	30.8 · **62.4** · 100.0	30.8 · **65.5** · 100.0	30.8 · **53.3** · 100.0
	DDNet	31.9 · **72.5** · 99.3	30.8 · **82.0** · 100.0	30.8 · **65.7** · 100.0	30.8 · **53.0** · 100.0	30.8 · **61.6** · 100.0
	EvNet	30.8 · **86.4** · 100.0	30.8 · **94.1** · 100.0	30.8 · **78.6** · 100.0	30.8 · **77.7** · 100.0	30.8 · **85.5** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	30.8 · **76.8** · 100.0	30.8 · **64.6** · 100.0	30.8 · **82.9** · 100.0	32.2 · **50.0** · 100.0	50.0 · **50.0** · 50.0
	PriorNet	31.1 · **77.6** · 99.7	30.8 · **76.7** · 100.0	30.8 · **69.0** · 100.0	30.8 · **53.1** · 100.0	30.8 · **61.4** · 100.0
	DDNet	31.1 · **74.3** · 99.8	30.8 · **77.1** · 100.0	30.8 · **76.0** · 100.0	30.8 · **57.0** · 100.0	30.8 · **43.5** · 100.0
	EvNet	30.8 · **88.8** · 100.0	30.8 · **92.6** · 100.0	30.8 · **70.2** · 100.0	30.8 · **62.0** · 100.0	30.8 · **96.2** · 100.0

Table D.38: OOD detection based on differential entropy under PGD uncertainty attacks against differential entorpy on ID data and OOD data. Smoothed DBU models on CIFAR10. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
				ID-Attack			
Smoothed models	**PostNet**	72.1 · **82.7** · 88.0	35.0 · **56.6** · 97.4	31.9 · **65.6** · 99.8	30.7 · **50.6** · 100.0	30.7 · **46.9** · 100.0	30.7 · **51.6** · 100.0
	PriorNet	50.2 · **53.1** · 55.9	33.5 · **43.3** · 65.3	31.3 · **39.7** · 69.1	31.3 · **48.3** · 98.2	30.7 · **44.4** · 99.9	30.7 · **45.4** · 100.0
	DDNet	72.0 · **75.8** · 79.8	35.6 · **46.2** · 69.8	32.9 · **50.3** · 87.1	31.1 · **58.7** · 98.6	30.7 · **59.3** · 100.0	30.7 · **44.5** · 100.0
	EvNet	79.5 · **87.1** · 92.8	34.1 · **58.6** · 95.1	32.5 · **61.2** · 96.9	31.7 · **60.6** · 98.7	30.7 · **62.4** · 100.0	30.7 · **57.3** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	35.0 · **58.5** · 97.7	31.2 · **46.6** · 97.4	30.8 · **57.7** · 99.7	30.7 · **49.8** · 100.0	30.7 · **50.9** · 100.0
	PriorNet	-	31.5 · **36.7** · 57.2	33.1 · **51.8** · 84.8	30.7 · **57.7** · 98.7	30.7 · **40.0** · 99.9	30.9 · **53.6** · 96.7
	DDNet	-	36.2 · **50.0** · 78.6	32.1 · **41.3** · 70.2	30.8 · **56.4** · 100.0	30.7 · **49.4** · 100.0	30.7 · **54.8** · 100.0
	EvNet	-	46.8 · **61.0** · 79.7	32.3 · **58.9** · 99.1	30.7 · **45.0** · 100.0	30.7 · **63.3** · 100.0	30.8 · **38.1** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	35.2 · **55.9** · 96.0	34.5 · **59.2** · 94.9	30.7 · **47.0** · 100.0	30.7 · **58.2** · 100.0	30.7 · **42.9** · 100.0
	PriorNet	-	31.8 · **38.9** · 64.1	31.0 · **41.8** · 87.9	30.7 · **42.9** · 99.2	30.7 · **48.6** · 100.0	30.7 · **46.6** · 100.0
	DDNet	-	39.7 · **52.1** · 75.7	36.4 · **56.8** · 83.8	31.0 · **51.5** · 97.4	31.0 · **56.8** · 97.8	30.7 · **49.1** · 100.0
	EvNet	-	34.8 · **64.9** · 99.6	30.8 · **48.9** · 99.8	30.7 · **66.8** · 100.0	30.9 · **41.5** · 93.8	31.1 · **55.1** · 100.0
				OOD-Attack			
Smoothed models	**PostNet**	72.0 · **82.7** · 88.0	35.1 · **56.8** · 97.3	32.0 · **65.8** · 99.8	30.7 · **50.7** · 100.0	30.7 · **46.5** · 100.0	30.7 · **51.7** · 100.0
	PriorNet	50.3 · **53.1** · 55.9	33.6 · **43.7** · 65.9	31.3 · **39.8** · 69.4	31.3 · **48.3** · 98.2	30.7 · **44.5** · 99.9	30.7 · **46.4** · 100.0
	DDNet	72.0 · **75.8** · 79.8	35.6 · **46.2** · 70.0	32.9 · **50.1** · 86.7	31.1 · **58.8** · 98.6	30.7 · **59.3** · 100.0	30.7 · **44.6** · 100.0
	EvNet	79.5 · **87.1** · 92.8	34.1 · **58.8** · 95.2	32.6 · **61.2** · 96.9	31.7 · **60.5** · 98.7	30.7 · **62.4** · 100.0	30.7 · **57.6** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	35.0 · **58.5** · 97.8	31.2 · **46.6** · 97.2	30.8 · **57.7** · 99.7	30.7 · **50.2** · 100.0	30.7 · **51.5** · 100.0
	PriorNet	-	31.6 · **37.3** · 59.3	33.2 · **52.7** · 85.8	30.7 · **57.8** · 98.7	30.7 · **40.1** · 99.9	30.9 · **53.8** · 96.8
	DDNet	-	36.4 · **50.2** · 78.9	32.1 · **41.5** · 70.4	30.9 · **56.2** · 100.0	30.7 · **49.3** · 100.0	30.7 · **55.1** · 100.0
	EvNet	-	47.2 · **61.1** · 80.0	32.4 · **59.1** · 99.1	30.7 · **45.0** · 100.0	30.7 · **63.2** · 100.0	30.8 · **38.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	35.3 · **56.4** · 96.1	34.5 · **59.0** · 94.9	30.7 · **46.8** · 100.0	30.7 · **57.8** · 100.0	30.7 · **43.2** · 100.0
	PriorNet	-	31.9 · **39.4** · 65.5	31.0 · **42.0** · 88.6	30.7 · **42.9** · 99.2	30.7 · **48.4** · 100.0	30.7 · **47.1** · 100.0
	DDNet	-	40.2 · **52.9** · 76.5	36.4 · **56.9** · 83.9	31.1 · **51.5** · 97.3	31.0 · **57.0** · 97.8	30.7 · **49.1** · 100.0
	EvNet	-	34.9 · **64.8** · 99.6	30.8 · **48.8** · 99.8	30.7 · **66.1** · 100.0	30.9 · **41.6** · 93.6	31.1 · **54.7** · 100.0

Table D.39: OOD detection based on differential entropy under PGD uncertainty attacks against differential entropy on ID data and OOD data. Smoothed DBU models on MNIST. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
				ID-Attack			
Smoothed models	**PostNet**	59.9 · **91.1** · 98.6	61.2 · **97.7** · 99.6	64.8 · **94.7** · 99.7	31.6 · **64.9** · 99.7	30.7 · **63.2** · 100.0	30.7 · **70.5** · 100.0
	PriorNet	99.8 · **99.8** · 99.8	99.4 · **99.8** · 99.9	98.3 · **99.6** · 99.9	48.5 · **91.9** · 99.9	31.1 · **74.6** · 99.8	30.7 · **67.3** · 100.0
	DDNet	98.5 · **98.6** · 98.7	95.0 · **97.6** · 98.9	74.7 · **92.0** · 98.2	31.4 · **52.0** · 98.5	30.7 · **52.0** · 100.0	30.7 · **41.1** · 100.0
	EvNet	85.7 · **87.5** · 89.2	68.9 · **90.4** · 97.7	42.5 · **90.2** · 99.6	30.7 · **69.8** · 100.0	30.7 · **50.3** · 100.0	30.7 · **45.6** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	84.3 · **96.2** · 99.3	50.4 · **89.2** · 99.5	30.9 · **46.2** · 99.4	30.7 · **46.9** · 100.0	30.7 · **62.2** · 100.0
	PriorNet	-	99.7 · **99.9** · 100.0	98.7 · **99.8** · 100.0	83.3 · **99.1** · 100.0	30.7 · **82.6** · 100.0	30.7 · **64.8** · 100.0
	DDNet	-	93.6 · **96.9** · 98.5	71.2 · **89.1** · 96.9	32.3 · **50.3** · 99.0	30.7 · **50.7** · 100.0	30.7 · **55.7** · 100.0
	EvNet	-	58.2 · **84.4** · 94.3	40.9 · **87.4** · 99.2	30.7 · **59.4** · 100.0	30.7 · **40.3** · 100.0	30.7 · **53.2** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	58.9 · **96.1** · 99.3	59.7 · **96.1** · 99.9	31.2 · **48.2** · 95.7	30.7 · **42.0** · 100.0	30.7 · **56.9** · 100.0
	PriorNet	-	99.9 · **100.0** · 100.0	96.5 · **99.2** · 99.9	49.2 · **96.9** · 100.0	31.3 · **88.1** · 100.0	30.7 · **77.8** · 100.0
	DDNet	-	95.0 · **97.5** · 98.8	80.6 · **94.1** · 98.7	31.7 · **55.6** · 98.6	30.7 · **52.0** · 100.0	30.7 · **47.6** · 100.0
	EvNet	-	66.5 · **91.3** · 98.1	48.1 · **84.1** · 97.6	30.8 · **49.7** · 99.9	30.7 · **37.9** · 100.0	30.8 · **63.5** · 100.0
				OOD-Attack			
Smoothed models	**PostNet**	59.0 · **91.2** · 97.7	57.8 · **97.2** · 99.6	61.4 · **93.8** · 99.6	31.5 · **58.9** · 99.5	30.7 · **51.5** · 100.0	30.7 · **53.5** · 100.0
	PriorNet	99.7 · **99.8** · 99.8	99.4 · **99.8** · 99.9	98.4 · **99.7** · 100.0	60.7 · **96.8** · 100.0	33.0 · **88.9** · 100.0	30.7 · **87.7** · 100.0
	DDNet	98.4 · **98.5** · 98.7	94.2 · **97.2** · 98.7	72.1 · **90.5** · 97.8	31.6 · **52.3** · 98.1	30.7 · **51.7** · 100.0	30.7 · **37.7** · 100.0
	EvNet	83.9 · **85.7** · 88.0	63.5 · **88.6** · 97.9	40.1 · **87.7** · 99.6	30.8 · **68.9** · 100.0	30.7 · **43.3** · 100.0	30.7 · **36.8** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	84.7 · **96.1** · 99.4	49.7 · **89.1** · 99.5	30.9 · **45.6** · 99.3	30.7 · **45.8** · 100.0	30.7 · **69.1** · 100.0
	PriorNet	-	99.7 · **99.9** · 100.0	98.7 · **99.8** · 100.0	86.8 · **99.5** · 100.0	30.9 · **93.2** · 100.0	30.7 · **81.4** · 100.0
	DDNet	-	93.9 · **97.0** · 98.6	72.0 · **89.4** · 97.0	33.0 · **52.4** · 98.8	30.7 · **51.5** · 100.0	30.7 · **60.1** · 100.0
	EvNet	-	59.5 · **85.3** · 94.6	40.7 · **86.9** · 99.2	30.7 · **57.4** · 100.0	30.7 · **39.2** · 100.0	30.7 · **49.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	55.7 · **96.1** · 99.3	58.4 · **95.7** · 99.8	31.1 · **44.2** · 93.1	30.7 · **41.2** · 100.0	30.7 · **48.8** · 100.0
	PriorNet	-	99.9 · **100.0** · 100.0	97.0 · **99.3** · 99.9	61.0 · **98.4** · 100.0	33.2 · **94.4** · 100.0	30.7 · **90.2** · 100.0
	DDNet	-	95.3 · **97.6** · 98.9	82.2 · **94.5** · 98.7	32.1 · **56.6** · 98.5	30.7 · **48.6** · 100.0	30.7 · **42.9** · 100.0
	EvNet	-	65.2 · **90.4** · 98.0	46.8 · **83.4** · 97.3	30.8 · **48.8** · 99.9	30.7 · **36.3** · 100.0	30.8 · **60.1** · 100.0

Table D.40: OOD detection based on differential entropy under PGD uncertainty attacks against differential entropy on ID data and OOD data. Smoothed DBU models on Sensorless. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
				ID-Attack			
Smoothed models	**PostNet**	49.3 · **90.4** · 99.8	30.7 · **49.2** · 100.0	30.7 · **36.0** · 100.0	49.2 · **50.0** · 74.9	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	31.2 · **39.0** · 66.9	30.7 · **35.5** · 100.0	30.7 · **38.9** · 100.0	30.7 · **46.2** · 100.0	30.7 · **62.7** · 100.0	30.7 · **51.3** · 100.0
	DDNet	31.0 · **31.5** · 32.7	30.7 · **30.8** · 100.0	30.7 · **31.8** · 100.0	30.7 · **53.6** · 100.0	30.7 · **43.9** · 100.0	30.7 · **40.5** · 100.0
	EvNet	33.6 · **55.2** · 91.3	30.7 · **44.2** · 100.0	30.7 · **43.8** · 100.0	30.7 · **39.3** · 100.0	30.8 · **51.6** · 100.0	32.4 · **50.0** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	30.7 · **62.4** · 100.0	30.7 · **39.2** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	-	30.7 · **30.9** · 100.0	30.7 · **32.4** · 100.0	30.7 · **31.0** · 100.0	30.8 · **30.7** · 100.0	38.2 · **48.9** · 100.0
	DDNet	-	30.7 · **32.2** · 100.0	30.7 · **30.9** · 100.0	30.7 · **37.1** · 100.0	30.7 · **42.1** · 100.0	30.7 · **37.7** · 100.0
	EvNet	-	30.7 · **48.9** · 100.0	30.7 · **34.0** · 100.0	30.7 · **35.6** · 100.0	30.7 · **33.6** · 100.0	30.7 · **50.0** · 100.0
Smoothed + adv. w. uncert. attacks	**PostNet**	-	30.7 · **46.0** · 100.0	30.7 · **46.6** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	-	30.7 · **35.8** · 100.0	30.7 · **32.1** · 100.0	30.7 · **81.6** · 100.0	30.8 · **41.7** · 100.0	30.7 · **61.9** · 100.0
	DDNet	-	30.7 · **32.8** · 100.0	30.7 · **31.0** · 100.0	30.7 · **31.8** · 100.0	30.7 · **43.7** · 100.0	30.7 · **34.7** · 100.0
	EvNet	-	30.7 · **31.0** · 100.0	30.7 · **49.6** · 100.0	30.7 · **47.7** · 100.0	30.7 · **42.6** · 100.0	30.7 · **50.0** · 100.0
				OOD-Attack			
Smoothed models	**PostNet**	49.3 · **90.4** · 99.8	30.8 · **76.4** · 100.0	30.7 · **61.3** · 100.0	47.7 · **50.0** · 75.1	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	31.2 · **39.0** · 66.9	30.7 · **33.9** · 100.0	30.7 · **34.3** · 100.0	30.7 · **37.0** · 100.0	30.7 · **74.0** · 100.0	30.9 · **78.1** · 100.0
	DDNet	31.0 · **31.5** · 32.7	30.7 · **30.7** · 100.0	30.7 · **31.8** · 100.0	30.7 · **47.7** · 100.0	30.7 · **43.8** · 100.0	30.7 · **52.5** · 100.0
	EvNet	33.6 · **55.2** · 91.2	30.7 · **54.7** · 100.0	30.7 · **54.0** · 100.0	30.7 · **51.0** · 100.0	30.7 · **45.2** · 100.0	31.7 · **50.0** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	30.7 · **82.2** · 100.0	30.7 · **61.4** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	-	30.7 · **31.2** · 100.0	30.7 · **31.4** · 99.9	30.7 · **30.8** · 100.0	30.8 · **30.7** · 100.0	33.8 · **34.0** · 100.0
	DDNet	-	30.7 · **32.2** · 100.0	30.7 · **30.8** · 100.0	30.7 · **33.6** · 100.0	30.7 · **46.9** · 100.0	30.7 · **40.3** · 100.0
	EvNet	-	30.8 · **75.3** · 100.0	30.7 · **31.6** · 100.0	30.7 · **42.1** · 100.0	30.7 · **38.7** · 100.0	30.7 · **50.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	30.7 · **73.7** · 100.0	30.7 · **61.6** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
	PriorNet	-	30.7 · **35.9** · 100.0	30.7 · **30.7** · 100.0	30.7 · **39.4** · 100.0	30.7 · **36.6** · 100.0	30.7 · **97.6** · 100.0
	DDNet	-	30.7 · **32.1** · 100.0	30.7 · **30.8** · 100.0	30.7 · **32.2** · 100.0	30.7 · **50.7** · 100.0	30.7 · **39.8** · 100.0
	EvNet	-	30.7 · **31.3** · 100.0	30.8 · **39.7** · 100.0	30.7 · **52.2** · 100.0	30.7 · **42.3** · 100.0	30.7 · **50.0** · 100.0

Table D.41: OOD detection based on differential entropy under PGD uncertainty attacks against differential entropy on ID data and OOD data. Smoothed DBU models on Segment. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
				ID-Attack			
Smoothed models	**PostNet**	99.6 · **99.9** · 99.9	33.0 · **83.0** · 100.0	30.8 · **43.8** · 100.0	30.8 · **31.7** · 100.0	30.8 · **40.8** · 100.0	41.4 · **50.0** · 50.2
	PriorNet	30.8 · **31.0** · 31.4	30.8 · **30.8** · 42.6	30.8 · **30.8** · 95.5	30.8 · **33.1** · 100.0	30.8 · **76.4** · 100.0	30.8 · **78.7** · 100.0
	DDNet	30.8 · **30.8** · 30.8	30.8 · **30.8** · 32.1	30.8 · **30.8** · 69.4	30.8 · **30.8** · 100.0	30.8 · **31.0** · 100.0	30.8 · **33.4** · 100.0
	EvNet	94.9 · **97.2** · 98.3	31.1 · **75.8** · 99.9	30.8 · **74.2** · 100.0	30.8 · **62.9** · 100.0	30.8 · **58.1** · 100.0	30.8 · **43.4** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	31.0 · **70.9** · 100.0	30.8 · **47.1** · 100.0	30.8 · **85.0** · 100.0	30.8 · **50.0** · 100.0	50.0 · **50.0** · 50.0
	PriorNet	-	30.8 · **30.8** · 46.0	30.8 · **30.8** · 32.7	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0	30.9 · **30.8** · 100.0
	DDNet	-	30.8 · **30.8** · 30.8	30.8 · **30.8** · 79.5	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0	30.8 · **57.3** · 100.0
	EvNet	-	36.3 · **94.3** · 100.0	30.8 · **32.2** · 100.0	30.8 · **50.2** · 100.0	30.8 · **93.9** · 100.0	30.8 · **56.3** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	30.8 · **49.5** · 100.0	30.8 · **34.5** · 100.0	30.8 · **96.1** · 100.0	41.2 · **50.0** · 82.7	50.0 · **50.0** · 50.0
	PriorNet	-	30.8 · **31.2** · 62.6	30.8 · **30.8** · 32.9	30.8 · **30.8** · 88.9	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0
	DDNet	-	30.8 · **30.8** · 31.2	30.8 · **30.8** · 68.9	30.8 · **30.8** · 100.0	30.8 · **30.9** · 100.0	30.8 · **38.6** · 100.0
	EvNet	-	30.9 · **83.5** · 100.0	30.8 · **84.0** · 100.0	30.8 · **98.6** · 100.0	30.8 · **92.8** · 100.0	30.8 · **45.6** · 100.0
				OOD-Attack			
Smoothed models	**PostNet**	99.6 · **99.9** · 99.9	31.3 · **95.2** · 100.0	30.8 · **48.7** · 100.0	30.8 · **34.0** · 100.0	30.8 · **41.0** · 100.0	41.8 · **50.0** · 50.2
	PriorNet	30.8 · **31.0** · 31.4	30.8 · **30.8** · 44.7	30.8 · **30.8** · 86.3	30.8 · **30.9** · 100.0	30.8 · **35.7** · 100.0	30.8 · **57.4** · 100.0
	DDNet	30.8 · **30.8** · 30.8	30.8 · **30.8** · 31.9	30.8 · **30.8** · 58.3	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0
	EvNet	94.9 · **97.2** · 98.3	31.4 · **92.5** · 100.0	30.8 · **94.2** · 100.0	30.8 · **80.4** · 100.0	30.8 · **70.2** · 100.0	30.8 · **48.2** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	30.8 · **88.7** · 100.0	30.8 · **70.9** · 100.0	30.8 · **97.2** · 100.0	30.8 · **50.0** · 100.0	50.0 · **50.0** · 50.0
	PriorNet	-	30.8 · **30.9** · 47.2	30.8 · **30.8** · 32.5	30.8 · **30.8** · 96.2	30.8 · **30.8** · 100.0	30.9 · **30.8** · 100.0
	DDNet	-	30.8 · **30.8** · 30.8	30.8 · **30.8** · 73.5	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0	30.8 · **34.3** · 100.0
	EvNet	-	35.9 · **95.9** · 100.0	30.8 · **36.6** · 100.0	30.8 · **45.8** · 100.0	30.8 · **75.2** · 100.0	30.8 · **93.8** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	30.8 · **64.6** · 100.0	30.8 · **31.9** · 100.0	30.8 · **99.1** · 100.0	37.2 · **50.0** · 100.0	49.8 · **50.0** · 50.0
	PriorNet	-	30.8 · **31.3** · 60.6	30.8 · **30.8** · 34.8	30.8 · **30.8** · 73.8	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0
	DDNet	-	30.8 · **30.8** · 31.7	30.8 · **30.8** · 64.6	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0
	EvNet	-	31.1 · **90.7** · 100.0	30.8 · **96.6** · 100.0	30.8 · **98.9** · 100.0	30.8 · **97.5** · 100.0	30.8 · **34.2** · 100.0

Table D.42: OOD detection based on differential entropy under FGSM uncertainty attacks against differential entropy on ID data and OOD data. Smoothed DBU models on CIFAR10. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
				ID-Attack			
Smoothed models	**PostNet**	72.2 · **82.7** · 88.0	35.0 · **56.5** · 97.5	31.9 · **65.5** · 99.8	30.7 · **50.6** · 100.0	30.7 · **46.9** · 100.0	30.7 · **51.4** · 100.0
	PriorNet	50.3 · **53.1** · 55.9	33.5 · **43.2** · 65.0	31.3 · **39.7** · 69.1	31.3 · **48.3** · 98.2	30.7 · **44.2** · 99.9	30.7 · **44.9** · 100.0
	DDNet	72.0 · **75.8** · 79.8	35.5 · **46.2** · 69.7	32.9 · **50.3** · 87.0	31.1 · **58.6** · 98.6	30.7 · **59.4** · 100.0	30.7 · **44.5** · 100.0
	EvNet	79.5 · **87.1** · 92.8	34.1 · **58.6** · 95.2	32.5 · **61.1** · 96.9	31.7 · **60.6** · 98.8	30.7 · **62.6** · 100.0	30.7 · **57.3** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	35.0 · **58.5** · 97.7	31.2 · **46.6** · 97.4	30.8 · **57.7** · 99.7	30.7 · **50.1** · 100.0	30.7 · **50.6** · 100.0
	PriorNet	-	31.5 · **36.6** · 56.7	33.1 · **51.7** · 84.4	30.7 · **57.5** · 98.7	30.7 · **40.1** · 99.9	30.9 · **53.5** · 96.7
	DDNet	-	36.2 · **50.0** · 78.5	32.1 · **41.3** · 70.1	30.9 · **56.3** · 100.0	30.7 · **49.5** · 100.0	30.7 · **54.9** · 100.0
	EvNet	-	46.8 · **60.9** · 79.6	32.3 · **58.9** · 99.1	30.7 · **45.1** · 100.0	30.7 · **63.1** · 100.0	30.8 · **38.1** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	35.2 · **56.0** · 95.9	34.5 · **59.0** · 94.8	30.7 · **47.0** · 100.0	30.7 · **57.2** · 100.0	30.7 · **42.7** · 100.0
	PriorNet	-	31.8 · **38.8** · 64.0	31.0 · **41.7** · 87.4	30.7 · **42.9** · 99.3	30.7 · **48.5** · 100.0	30.7 · **46.8** · 100.0
	DDNet	-	39.6 · **52.0** · 75.6	36.4 · **56.8** · 83.8	31.0 · **51.4** · 97.3	31.0 · **56.9** · 97.7	30.7 · **49.2** · 100.0
	EvNet	-	34.8 · **64.9** · 99.7	30.8 · **48.9** · 99.8	30.7 · **66.4** · 100.0	30.9 · **41.6** · 93.6	31.1 · **55.7** · 100.0
				OOD-Attack			
Smoothed models	**PostNet**	72.1 · **82.7** · 88.0	35.1 · **56.8** · 97.3	31.9 · **65.8** · 99.8	30.7 · **50.8** · 100.0	30.7 · **46.5** · 100.0	30.7 · **51.5** · 100.0
	PriorNet	50.3 · **53.1** · 55.9	33.6 · **43.7** · 65.9	31.3 · **39.8** · 69.4	31.3 · **48.3** · 98.2	30.7 · **44.4** · 99.9	30.7 · **45.9** · 100.0
	DDNet	72.0 · **75.8** · 79.8	35.6 · **46.1** · 70.0	32.9 · **50.1** · 86.7	31.1 · **58.7** · 98.6	30.7 · **59.3** · 100.0	30.7 · **44.6** · 100.0
	EvNet	79.5 · **87.1** · 92.8	34.1 · **58.8** · 95.2	32.6 · **61.3** · 96.9	31.7 · **60.5** · 98.8	30.7 · **62.2** · 100.0	30.7 · **57.7** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	35.0 · **58.4** · 97.9	31.2 · **46.6** · 97.3	30.8 · **57.7** · 99.7	30.7 · **50.1** · 100.0	30.7 · **51.4** · 100.0
	PriorNet	-	31.6 · **37.3** · 59.2	33.2 · **52.6** · 85.8	30.7 · **57.8** · 98.7	30.7 · **39.8** · 99.9	30.9 · **53.7** · 96.8
	DDNet	-	36.4 · **50.2** · 78.8	32.1 · **41.5** · 70.5	30.8 · **56.2** · 100.0	30.7 · **49.2** · 100.0	30.7 · **55.0** · 100.0
	EvNet	-	47.2 · **61.0** · 79.9	32.4 · **59.1** · 99.1	30.7 · **45.1** · 100.0	30.7 · **63.1** · 100.0	30.8 · **38.0** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	35.3 · **56.3** · 96.1	34.5 · **59.1** · 94.9	30.7 · **46.9** · 100.0	30.7 · **57.8** · 100.0	30.7 · **43.1** · 100.0
	PriorNet	-	31.9 · **39.4** · 65.4	31.0 · **42.0** · 88.7	30.7 · **42.9** · 99.2	30.7 · **48.3** · 100.0	30.7 · **47.2** · 100.0
	DDNet	-	40.1 · **52.8** · 76.5	36.5 · **56.9** · 83.9	31.1 · **51.5** · 97.3	31.0 · **57.0** · 97.8	30.7 · **48.7** · 100.0
	EvNet	-	34.9 · **65.0** · 99.6	30.8 · **48.8** · 99.8	30.7 · **66.6** · 100.0	30.9 · **41.1** · 93.4	31.1 · **55.3** · 100.0

Table D.43: OOD detection based on differential entropy under FGSM uncertainty attacks against differential entropy on ID data and OOD data. Smoothed DBU models on MNIST. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
				ID-Attack			
Smoothed models	**PostNet**	59.9 · **91.3** · 98.6	61.1 · **97.7** · 99.7	65.1 · **94.8** · 99.7	31.6 · **64.8** · 99.7	30.7 · **62.4** · 100.0	30.7 · **68.6** · 100.0
	PriorNet	99.8 · **99.8** · 99.8	99.4 · **99.8** · 99.9	98.4 · **99.7** · 99.9	49.8 · **92.7** · 99.9	31.3 · **76.6** · 99.8	30.7 · **71.8** · 100.0
	DDNet	98.5 · **98.6** · 98.7	95.0 · **97.6** · 98.9	74.4 · **91.9** · 98.2	31.4 · **52.0** · 98.5	30.7 · **51.8** · 100.0	30.7 · **40.2** · 100.0
	EvNet	85.7 · **87.5** · 89.2	69.0 · **90.4** · 97.7	42.5 · **90.2** · 99.6	30.7 · **70.1** · 100.0	30.7 · **50.0** · 100.0	30.7 · **43.9** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	84.4 · **96.3** · 99.4	50.6 · **89.3** · 99.5	30.9 · **46.3** · 99.4	30.7 · **46.3** · 100.0	30.7 · **63.3** · 100.0
	PriorNet	-	99.7 · **99.9** · 100.0	98.7 · **99.8** · 100.0	84.1 · **99.2** · 100.0	30.7 · **84.6** · 100.0	30.7 · **68.1** · 100.0
	DDNet	-	93.6 · **96.9** · 98.5	71.0 · **89.0** · 96.9	32.3 · **50.4** · 99.0	30.7 · **51.1** · 100.0	30.7 · **54.1** · 100.0
	EvNet	-	58.2 · **84.5** · 94.3	40.9 · **87.2** · 99.2	30.7 · **59.3** · 100.0	30.7 · **39.7** · 100.0	30.7 · **52.7** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	58.6 · **96.1** · 99.3	59.9 · **96.2** · 99.9	31.2 · **47.6** · 95.5	30.7 · **41.8** · 100.0	30.7 · **55.4** · 100.0
	PriorNet	-	99.9 · **100.0** · 100.0	96.6 · **99.2** · 99.9	50.3 · **97.1** · 100.0	31.7 · **89.7** · 100.0	30.7 · **81.8** · 100.0
	DDNet	-	95.0 · **97.5** · 98.8	80.5 · **94.0** · 98.6	31.7 · **55.6** · 98.6	30.7 · **52.0** · 100.0	30.7 · **49.5** · 100.0
	EvNet	-	66.5 · **91.4** · 98.1	48.5 · **84.5** · 97.6	30.8 · **49.3** · 99.9	30.7 · **37.3** · 100.0	30.8 · **62.0** · 100.0
				OOD-Attack			
Smoothed models	**PostNet**	59.2 · **91.3** · 97.7	57.9 · **97.2** · 99.6	61.4 · **93.8** · 99.6	31.5 · **59.1** · 99.5	30.7 · **52.4** · 100.0	30.7 · **53.9** · 100.0
	PriorNet	99.7 · **99.8** · 99.8	99.4 · **99.8** · 99.9	98.3 · **99.7** · 100.0	60.4 · **96.6** · 100.0	32.8 · **88.2** · 99.9	30.7 · **86.1** · 100.0
	DDNet	98.4 · **98.5** · 98.7	94.3 · **97.2** · 98.7	72.2 · **90.6** · 97.8	31.6 · **52.2** · 98.1	30.7 · **51.8** · 100.0	30.7 · **38.5** · 100.0
	EvNet	83.9 · **85.7** · 88.0	63.6 · **88.6** · 97.9	40.1 · **87.6** · 99.6	30.8 · **69.2** · 100.0	30.7 · **43.5** · 100.0	30.7 · **37.4** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	84.4 · **96.2** · 99.4	49.7 · **89.1** · 99.5	30.9 · **45.6** · 99.3	30.7 · **46.2** · 100.0	30.7 · **68.1** · 100.0
	PriorNet	-	99.7 · **99.9** · 100.0	98.7 · **99.8** · 100.0	86.3 · **99.4** · 100.0	30.9 · **91.9** · 100.0	30.7 · **77.5** · 100.0
	DDNet	-	93.9 · **97.0** · 98.6	72.1 · **89.5** · 97.0	33.0 · **52.3** · 98.8	30.7 · **51.5** · 100.0	30.7 · **60.4** · 100.0
	EvNet	-	59.4 · **85.6** · 94.6	40.7 · **86.7** · 99.2	30.7 · **57.3** · 100.0	30.7 · **39.4** · 100.0	30.7 · **49.0** · 100.0
Smoothed + adv. w. uncert. attacks	**PostNet**	-	55.8 · **96.1** · 99.3	58.4 · **95.7** · 99.8	31.1 · **44.6** · 93.3	30.7 · **41.4** · 100.0	30.7 · **50.1** · 100.0
	PriorNet	-	99.9 · **100.0** · 100.0	96.9 · **99.3** · 99.9	60.3 · **98.2** · 100.0	33.0 · **93.5** · 100.0	30.7 · **87.8** · 100.0
	DDNet	-	95.3 · **97.6** · 98.9	82.3 · **94.5** · 98.7	32.1 · **56.3** · 98.5	30.7 · **48.9** · 100.0	30.7 · **43.4** · 100.0
	EvNet	-	65.3 · **90.3** · 97.9	46.9 · **83.1** · 97.3	30.8 · **48.8** · 99.9	30.7 · **36.6** · 100.0	30.8 · **60.7** · 100.0

Table D.44: OOD detection based on differential entropy under FGSM uncertainty attacks against differential entropy on ID data and OOD data. Smoothed DBU models on Sensorless. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model)..

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
					ID-Attack		
	PostNet	49.3 · **90.4** · 99.8	30.7 · **50.3** · 100.0	30.7 · **36.6** · 100.0	49.1 · **50.0** · 74.9	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
Smoothed	**PriorNet**	31.2 · **39.0** · 66.9	30.7 · **40.1** · 100.0	30.7 · **48.2** · 100.0	30.7 · **54.2** · 100.0	30.7 · **46.3** · 100.0	30.7 · **47.6** · 100.0
models	**DDNet**	31.0 · **31.5** · 32.7	30.7 · **31.2** · 100.0	30.7 · **35.3** · 100.0	30.7 · **55.7** · 100.0	30.7 · **42.4** · 100.0	30.7 · **40.4** · 100.0
	EvNet	33.6 · **55.1** · 91.3	30.7 · **39.1** · 100.0	30.7 · **37.1** · 100.0	30.7 · **35.4** · 100.0	30.8 · **52.1** · 100.0	32.5 · **50.0** · 100.0
Smoothed	**PostNet**	-	30.7 · **60.8** · 100.0	30.7 · **40.7** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
+ adv.	**PriorNet**	-	30.7 · **31.3** · 100.0	30.7 · **32.9** · 100.0	30.7 · **40.1** · 100.0	30.8 · **31.1** · 100.0	38.1 · **91.0** · 100.0
label	**DDNet**	-	30.7 · **34.3** · 100.0	30.7 · **33.9** · 100.0	30.7 · **38.2** · 100.0	30.7 · **63.6** · 100.0	30.7 · **41.8** · 100.0
attacks	**EvNet**	-	30.8 · **41.0** · 100.0	30.7 · **34.2** · 100.0	30.7 · **38.0** · 100.0	30.7 · **39.0** · 100.0	30.7 · **50.0** · 100.0
Smoothed	**PostNet**	-	30.7 · **46.1** · 100.0	30.7 · **46.8** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
+ adv.	**PriorNet**	-	30.7 · **36.5** · 100.0	30.7 · **34.4** · 100.0	30.7 · **77.8** · 100.0	30.8 · **53.0** · 100.0	30.7 · **39.2** · 100.0
uncert.	**DDNet**	-	30.7 · **36.0** · 100.0	30.7 · **37.7** · 100.0	30.7 · **41.0** · 100.0	30.7 · **42.3** · 100.0	30.7 · **39.0** · 100.0
attacks	**EvNet**	-	30.7 · **31.3** · 100.0	30.7 · **43.3** · 100.0	30.7 · **36.3** · 100.0	30.7 · **43.2** · 100.0	30.7 · **50.0** · 100.0
					OOD-Attack		
	PostNet	49.3 · **90.4** · 99.8	30.8 · **75.3** · 100.0	30.7 · **68.5** · 100.0	46.1 · **50.0** · 74.8	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
Smoothed	**PriorNet**	31.2 · **38.9** · 67.0	30.7 · **34.1** · 100.0	30.7 · **35.7** · 100.0	30.7 · **35.0** · 100.0	30.7 · **77.6** · 100.0	30.8 · **95.3** · 100.0
models	**DDNet**	31.0 · **31.5** · 32.7	30.7 · **30.8** · 100.0	30.7 · **33.1** · 100.0	30.7 · **65.7** · 100.0	30.7 · **71.8** · 100.0	30.7 · **71.5** · 100.0
	EvNet	33.6 · **55.2** · 91.4	30.7 · **64.7** · 100.0	30.7 · **69.6** · 100.0	30.7 · **78.9** · 100.0	30.7 · **67.2** · 100.0	32.9 · **50.0** · 100.0
Smoothed	**PostNet**	-	30.7 · **86.0** · 100.0	30.7 · **86.6** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
+ adv.	**PriorNet**	-	30.7 · **31.0** · 99.9	30.7 · **31.2** · 98.9	30.7 · **30.7** · 100.0	30.8 · **30.7** · 100.0	36.1 · **35.3** · 100.0
label	**DDNet**	-	30.7 · **37.2** · 100.0	30.7 · **31.1** · 100.0	30.7 · **37.1** · 100.0	30.7 · **50.5** · 100.0	30.7 · **84.6** · 100.0
attacks	**EvNet**	-	30.8 · **82.5** · 100.0	30.7 · **51.7** · 100.0	30.7 · **91.5** · 100.0	30.7 · **70.0** · 100.0	30.9 · **50.0** · 100.0
Smoothed	**PostNet**	-	30.7 · **78.5** · 100.0	30.7 · **67.1** · 100.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0	50.0 · **50.0** · 50.0
+ adv.	**PriorNet**	-	30.7 · **35.8** · 100.0	30.7 · **30.7** · 100.0	30.7 · **39.0** · 100.0	30.7 · **58.5** · 100.0	30.7 · **100.0** · 100.0
uncert.	**DDNet**	-	30.7 · **40.8** · 100.0	30.7 · **33.1** · 100.0	30.7 · **30.8** · 100.0	30.7 · **34.3** · 100.0	30.7 · **35.2** · 100.0
attacks	**EvNet**	-	30.7 · **32.7** · 100.0	30.8 · **50.2** · 100.0	30.7 · **99.6** · 100.0	30.7 · **58.7** · 100.0	30.7 · **50.0** · 100.0

Table D.45: OOD detection based on differential entropy under FGSM uncertainty attacks against differential entropy on ID data and OOD data. Smoothed DBU models on Segment. Column format: guaranteed lowest performance · empirical performance · guaranteed highest performance (blue: normally/adversarially trained smooth classifier is more robust than the base model).

	Att. Rad.	0.0	0.1	0.2	0.5	1.0	2.0
				ID-Attack			
Smoothed models	**PostNet**	99.6 · **99.9** · 99.9	33.1 · **78.8** · 100.0	30.8 · **46.2** · 100.0	30.8 · **34.2** · 100.0	30.8 · **41.4** · 100.0	41.5 · **50.0** · 50.2
	PriorNet	30.9 · **31.0** · 31.4	30.8 · **30.8** · 39.3	30.8 · **30.8** · 94.7	30.8 · **41.2** · 100.0	30.8 · **92.7** · 100.0	30.8 · **79.9** · 100.0
	DDNet	30.8 · **30.8** · 30.8	30.8 · **30.8** · 31.8	30.8 · **30.8** · 66.8	30.8 · **30.8** · 100.0	30.8 · **32.6** · 100.0	30.8 · **38.2** · 100.0
	EvNet	94.9 · **97.2** · 98.2	31.0 · **73.1** · 100.0	30.8 · **72.3** · 100.0	30.8 · **57.1** · 100.0	30.8 · **63.3** · 100.0	30.8 · **49.6** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	31.0 · **62.9** · 100.0	30.8 · **47.1** · 100.0	30.8 · **90.0** · 100.0	30.8 · **50.0** · 100.0	50.0 · **50.0** · 50.0
	PriorNet	-	30.8 · **30.8** · 43.5	30.8 · **30.8** · 32.5	30.8 · **30.9** · 100.0	30.8 · **30.9** · 100.0	30.8 · **30.8** · 100.0
	DDNet	-	30.8 · **30.8** · 30.8	30.8 · **30.8** · 76.1	30.8 · **30.9** · 100.0	30.8 · **34.8** · 100.0	30.8 · **53.0** · 100.0
	EvNet	-	35.5 · **93.5** · 100.0	30.8 · **31.8** · 100.0	30.8 · **48.7** · 100.0	30.8 · **93.8** · 100.0	30.8 · **63.7** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	30.8 · **47.5** · 100.0	30.8 · **37.5** · 100.0	30.8 · **92.9** · 100.0	41.1 · **50.0** · 97.3	50.0 · **50.0** · 50.0
	PriorNet	-	30.8 · **31.1** · 60.8	30.8 · **30.8** · 32.3	30.8 · **30.8** · 90.3	30.8 · **30.8** · 100.0	30.8 · **36.3** · 100.0
	DDNet	-	30.8 · **30.8** · 31.0	30.8 · **30.8** · 66.8	30.8 · **30.8** · 100.0	30.8 · **31.2** · 100.0	30.8 · **57.2** · 100.0
	EvNet	-	30.9 · **80.3** · 100.0	30.8 · **78.1** · 100.0	30.8 · **99.4** · 100.0	30.8 · **97.7** · 100.0	30.8 · **41.5** · 100.0
				OOD-Attack			
Smoothed models	**PostNet**	99.6 · **99.9** · 99.9	31.2 · **94.3** · 100.0	30.8 · **44.8** · 100.0	30.8 · **36.8** · 100.0	30.8 · **39.9** · 100.0	44.3 · **50.0** · 50.0
	PriorNet	30.9 · **31.0** · 31.4	30.8 · **30.8** · 42.0	30.8 · **30.8** · 80.4	30.8 · **30.8** · 100.0	30.8 · **37.5** · 100.0	30.8 · **94.9** · 100.0
	DDNet	30.8 · **30.8** · 30.8	30.8 · **30.8** · 31.5	30.8 · **30.8** · 48.0	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0
	EvNet	94.9 · **97.2** · 98.3	31.3 · **92.1** · 100.0	30.8 · **90.8** · 100.0	30.8 · **89.6** · 100.0	30.8 · **89.8** · 100.0	30.8 · **87.3** · 100.0
Smoothed + adv. label attacks	**PostNet**	-	30.8 · **85.3** · 100.0	30.8 · **85.9** · 100.0	30.8 · **78.8** · 100.0	30.9 · **50.0** · 100.0	50.0 · **50.0** · 50.0
	PriorNet	-	30.8 · **30.8** · 45.0	30.8 · **30.8** · 32.1	30.8 · **30.8** · 90.3	30.8 · **30.8** · 100.0	31.0 · **30.8** · 100.0
	DDNet	-	30.8 · **30.8** · 30.8	30.8 · **30.8** · 64.9	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0	30.8 · **79.4** · 100.0
	EvNet	-	35.4 · **95.0** · 100.0	30.8 · **35.2** · 100.0	30.8 · **51.9** · 100.0	30.8 · **80.0** · 100.0	30.8 · **99.9** · 100.0
Smoothed + adv. uncert. attacks	**PostNet**	-	30.8 · **63.4** · 100.0	30.8 · **31.7** · 100.0	30.8 · **98.4** · 100.0	33.2 · **50.0** · 100.0	50.0 · **50.0** · 50.0
	PriorNet	-	30.8 · **31.1** · 58.0	30.8 · **30.8** · 34.1	30.8 · **30.8** · 66.8	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0
	DDNet	-	30.8 · **30.8** · 31.2	30.8 · **30.8** · 61.5	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0	30.8 · **30.8** · 100.0
	EvNet	-	31.0 · **89.0** · 100.0	30.8 · **96.2** · 100.0	30.8 · **99.6** · 100.0	30.8 · **99.6** · 100.0	30.8 · **69.7** · 100.0

D.1.5 Visualization of differential entropy distributions on ID data and OOD data

The following Figures visualize the differential entropy distribution for ID data and OOD data for all models with standard training. We used label attacks and uncertainty attacks for CIFAR10 and MNIST. Thus, they show how well the DBU models separate on clean and perturbed ID data and OOD data.

Figs. D.1 and D.2 visualizes the differential entropy distribution of ID data and OOD data under label attacks. On CIFAR10, PriorNet and DDNet can barely distinguish between clean ID and OOD data. We observe a better ID/OOD distinction for PostNet and EvNet for clean data. However, we do not observe for any model an increase of the uncertainty estimates on label attacked data. Even worse, Posterior Network, PriorNet and DDNet seem to assign higher confidence on class label attacks. On MNIST, models show a slightly better behavior. They are capable to assign a higher uncertainty to label attacks up to some attack radius.

Figs. D.3 to D.6 visualizes the differential entropy distribution of ID data and OOD data under uncertainty attacks. For both CIFAR10 and MNIST data sets, we observed that uncertainty estimations of all models can be manipulated. That is, OOD uncertainty attacks can shift the OOD uncertainty distribution to more certain predictions, and ID uncertainty attacks can shift the ID uncertainty distribution to less certain predictions.

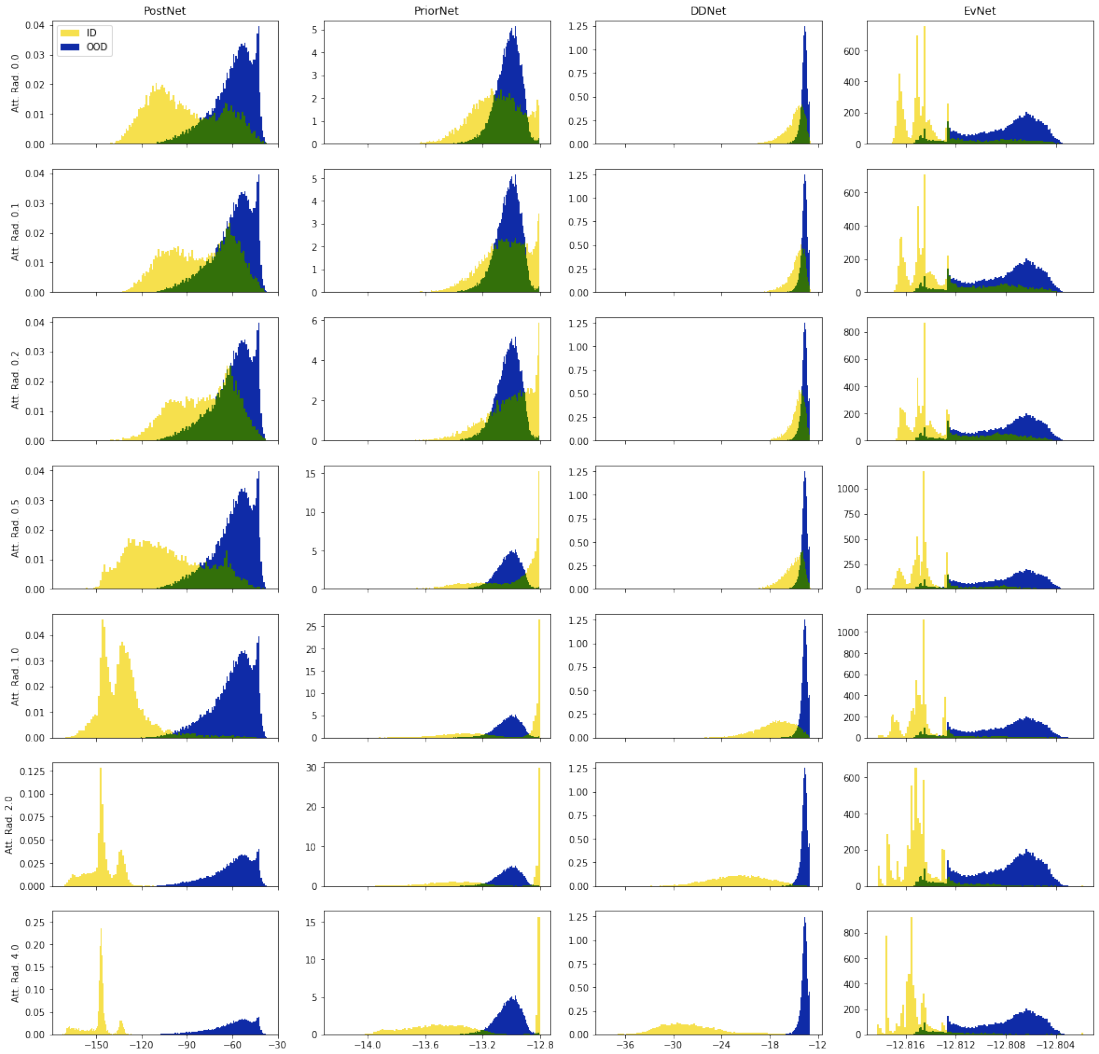


Figure D.1: Visualization of the differential entropy distribution of ID data (CIFAR10) and OOD data (SVHN) under label attack. The first row corresponds to no attack. The other rows correspond do increasingly stronger attack strength.

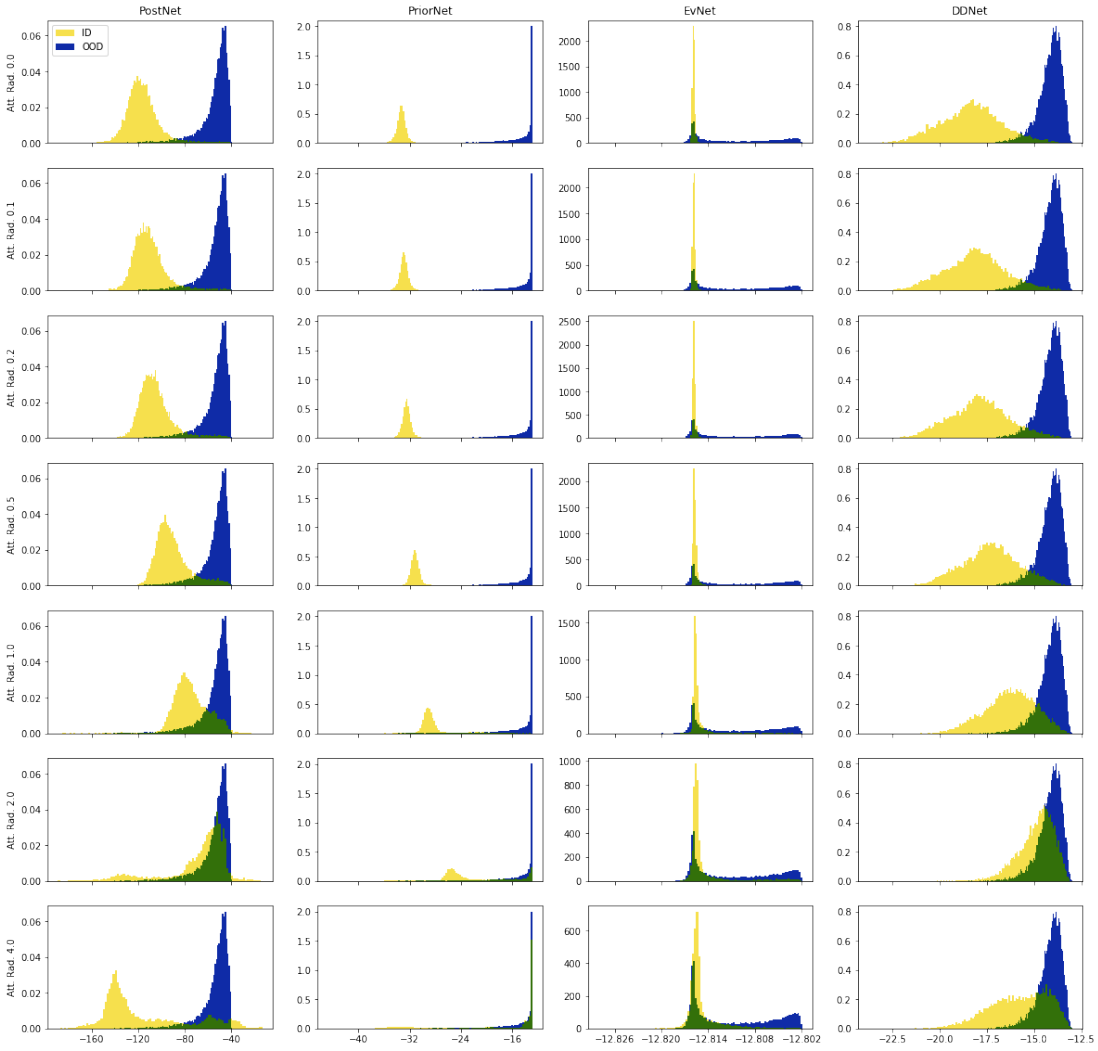


Figure D.2: Visualization of the differential entropy distribution of ID data (MNIST) and OOD data (KMNIST) under label attack. The first row corresponds to no attack. The other rows correspond do increasingly stronger attack strength.

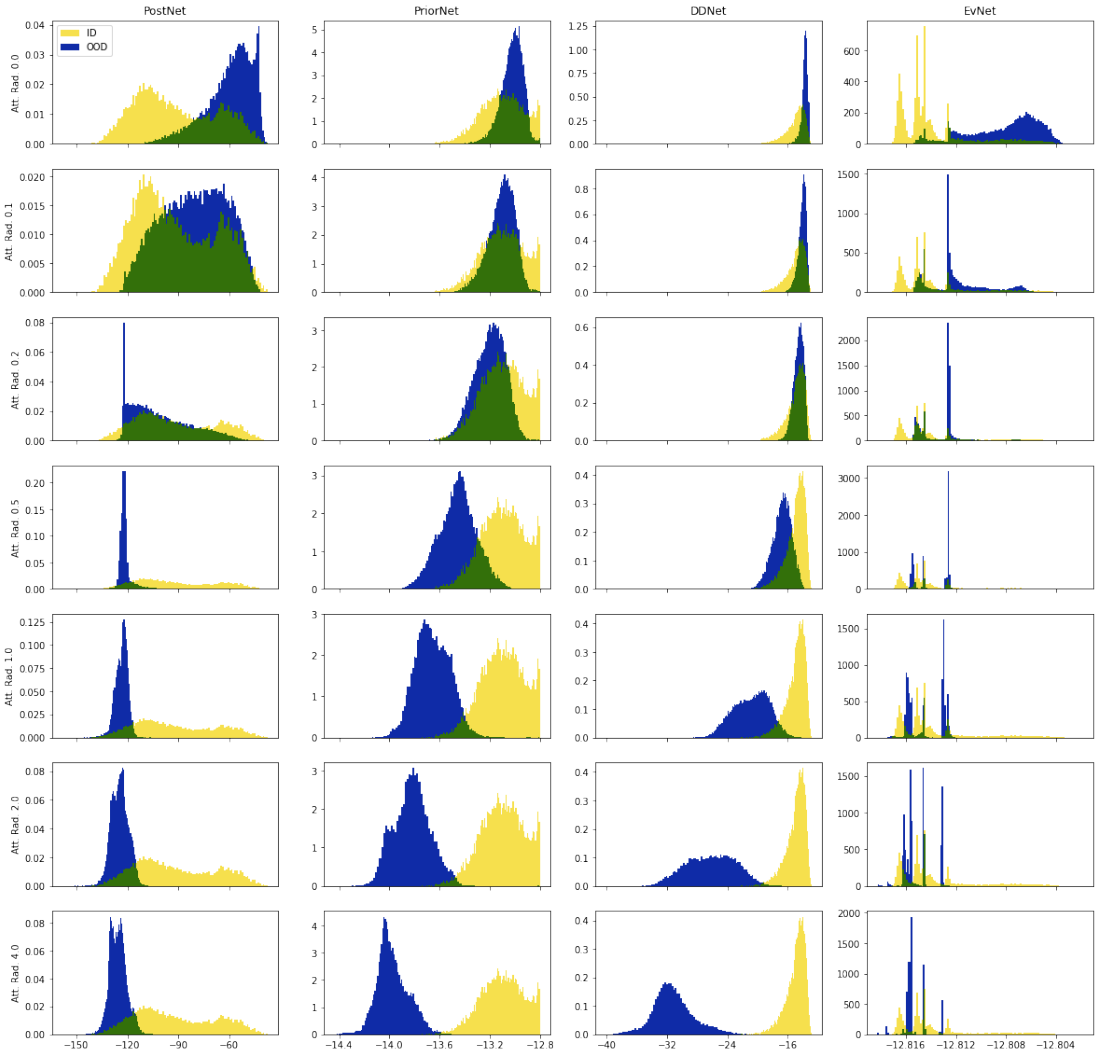


Figure D.3: Visualization of the differential entropy distribution of ID data (CIFAR10) and OOD data (SVHN) under OOD uncertainty attack. The first row corresponds to no attack. The other rows correspond do increasingly stronger attack strength.

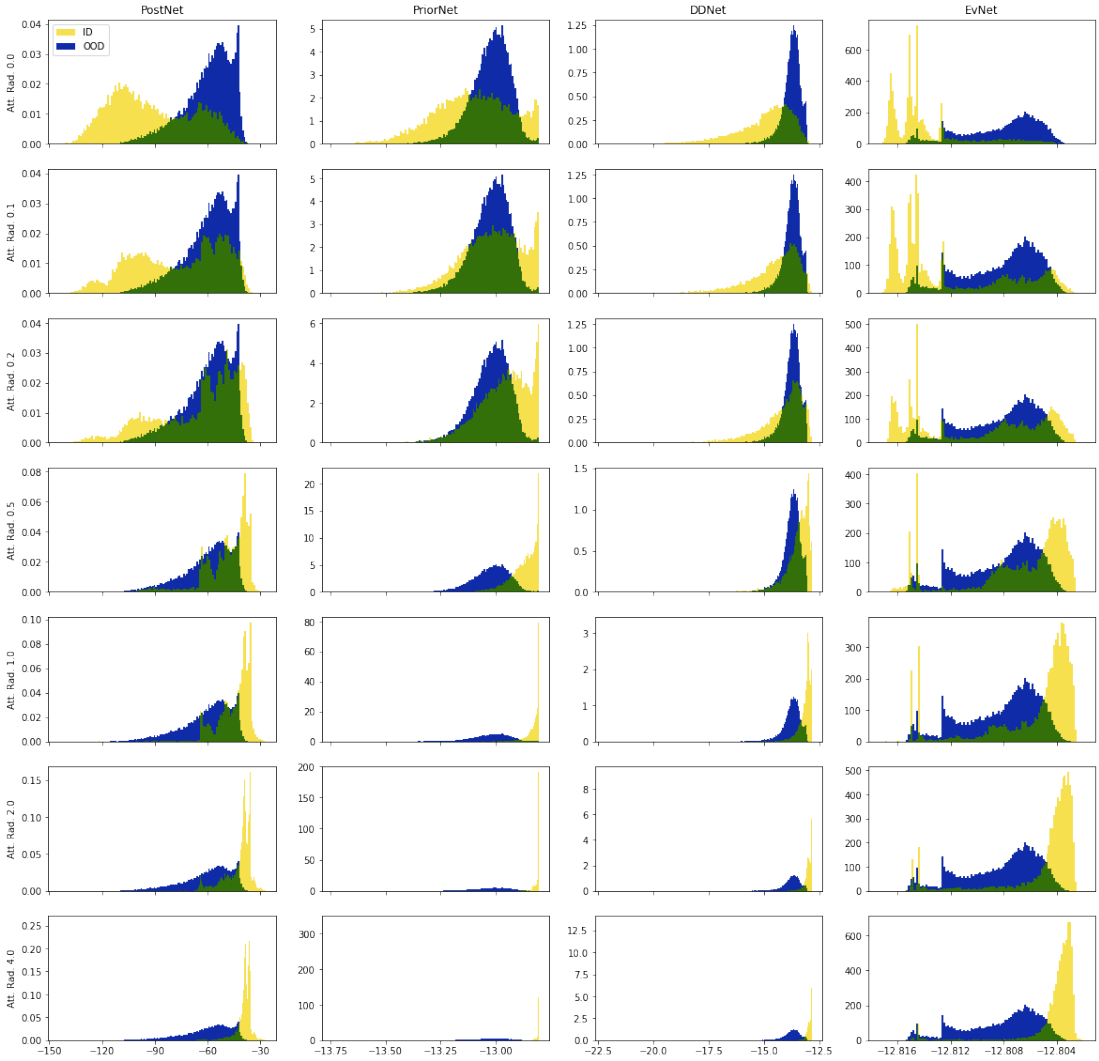


Figure D.4: Visualization of the differential entropy distribution of ID data (CIFAR10) and OOD data (SVHN) under ID uncertainty attack. The first row corresponds to no attack. The other rows correspond do increasingly stronger attack strength.

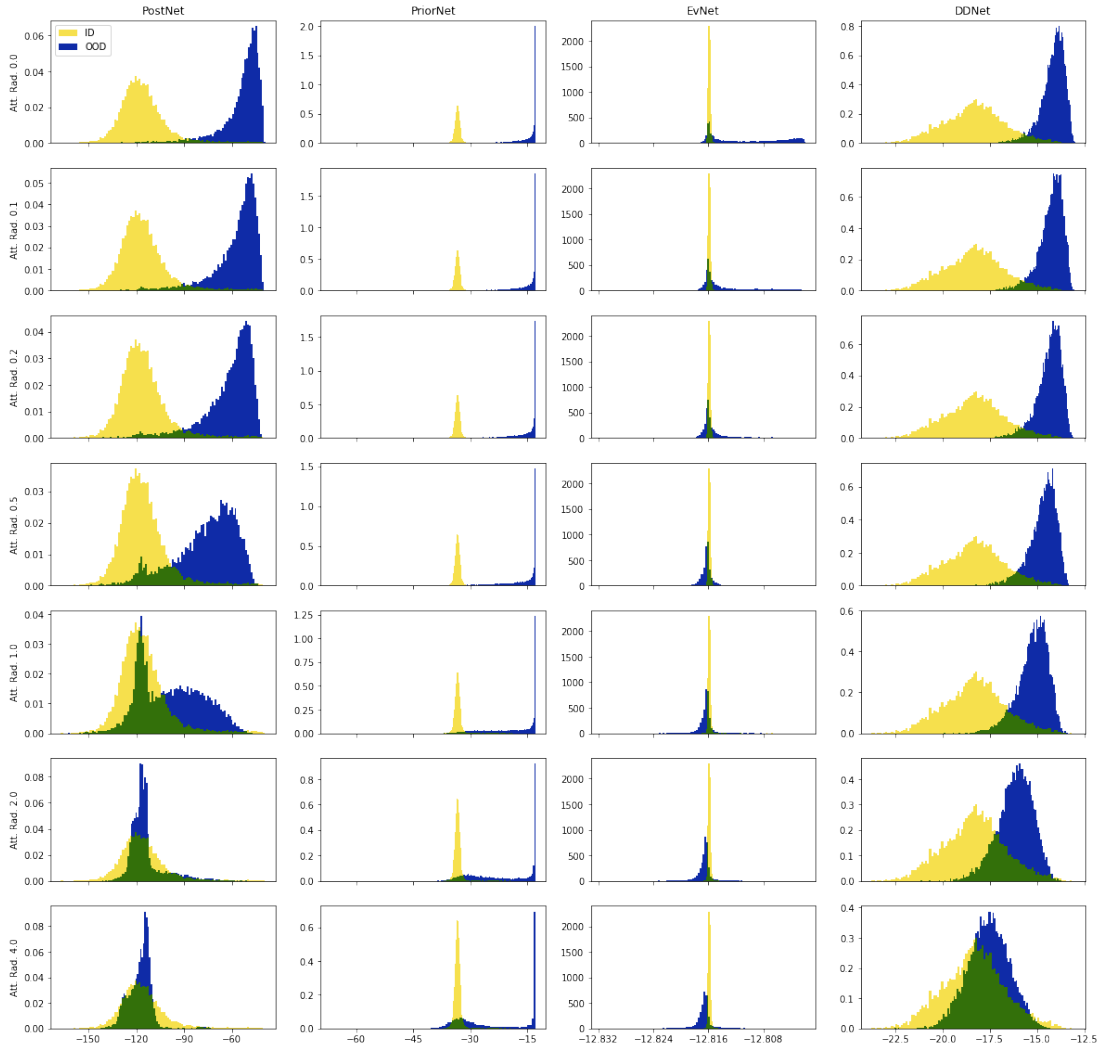


Figure D.5: Visualization of the differential entropy distribution of ID data (MNIST) and OOD data (KMNIST) under OOD uncertainty attack. The first row corresponds to no attack. The other rows correspond do increasingly stronger attack strength.

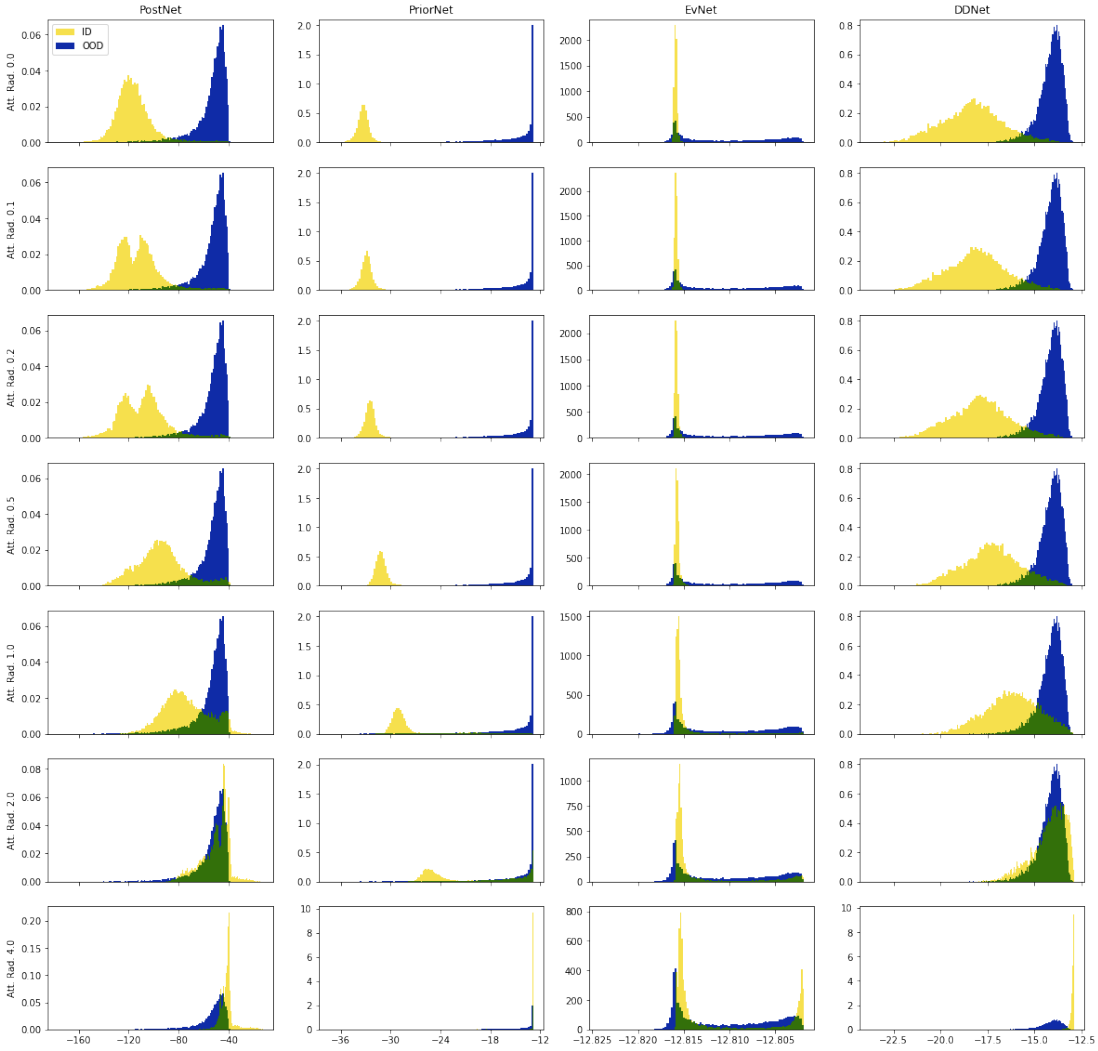


Figure D.6: Visualization of the differential entropy distribution of ID data (MNIST) and OOD data (KMNIST) under ID uncertainty attack. The first row corresponds to no attack. The other rows correspond do increasingly stronger attack strength.

E Uncertainty Estimation for Graph Data

E.1 Proofs

Lemma 9. *[68] Let a model be parameterized with an encoder f_ϕ with piecewise ReLU activations, a decoder g_ψ and the density estimator $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$. Let $f_\phi(\boldsymbol{x}) = V^{(l)}\boldsymbol{x} + a^{(l)}$ be the piecewise affine representation of the ReLU network f_ϕ on the finite number of affine regions $Q^{(l)}$ [17]. Suppose that $V^{(l)}$ have independent rows and the density function $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$ has bounded derivatives, then for almost any $\boldsymbol{x}$ we have $\mathbb{P}(f_\phi(\delta \cdot \boldsymbol{x}) \,|\, \boldsymbol{\omega}) \underset{\delta \to \infty}{\to} 0$. i.e the evidence becomes small far from training data.*

Theorem 10. *Lets consider a GPN model. Let $f_\phi(\boldsymbol{x}^{(v)}) = V^{(l)}\boldsymbol{x}^{(v)} + a^{(l)}$ be the piecewise affine representation of the ReLU network f_ϕ on the finite number of affine regions $Q^{(l)}$ [17]. Suppose that $V^{(l)}$ have independent rows, then for any node v and almost any $\boldsymbol{x}^{(v)}$ we have $\mathbb{P}(f_\phi(\delta \cdot \boldsymbol{x}^{(v)}) \,|\, c; \boldsymbol{\phi}) \underset{\delta \to \infty}{\to} 0$. Without network effects, it implies that $\beta_c^{ft,(v)} = \beta_c^{agg,(v)} \underset{\delta \to \infty}{\to} 0$.*

Proof. First, remark that each normalizing flow density in GPN fulfills the conditions of Lemma 9. This means that $\mathbb{P}(f_\phi(\delta \cdot \boldsymbol{x}^{(v)}) \,|\, c; \boldsymbol{\phi}) \underset{\delta \to \infty}{\to} 0$ which implies $\beta_c^{\text{ft},(v)} \underset{\delta \to \infty}{\to} 0$. Further note that in the absence of network effects, the PPR diffusion has no effect on the pseudo-counts i.e. $\beta_c^{\text{ft},(v)} = \beta_c^{\text{agg},(v)}$. □

Theorem 11. *Lets consider a GPN model. Then, given a node v, the aggregated feature evidence $\alpha_0^{agg,(v)}$ is increasing if the feature evidence $\alpha_0^{ft,(u)}$ of one of its neighbor $u \in \mathcal{N}(v)$ is increasing.*

Proof. We recall first the definition of the aggregated and feature total evidence pseudo-count and the PPR diffusion step of GPN:

$$\alpha_0^{\text{ft},(v)} = \sum_c \beta_c^{\text{ft},(v)}$$

$$\alpha_0^{\text{agg},(v)} = \sum_c \beta_c^{\text{agg},(v)}$$

$$\beta_c^{\text{agg},(v)} = \sum_{u \in \mathcal{V}} \Pi_{v,u}^{ppr} \beta_c^{\text{ft},(u)}$$

We combine these three equations to show a closed-form relation between the aggregated and the feature evidence pseudo-count:

$$\begin{aligned}\alpha_0^{\text{agg},(v)} &= \sum_c \beta_c^{\text{agg},(v)} \\ &= \sum_c \sum_{u \in \mathcal{V}} \Pi_{v,u}^{ppr} \beta_c^{\text{ft},(u)} \\ &= \sum_{u \in \mathcal{V}} \Pi_{v,u}^{ppr} \sum_c \beta_c^{\text{ft},(u)} \\ &= \sum_{u \in \mathcal{V}} \Pi_{v,u}^{ppr} \alpha_0^{\text{ft},(u)}\end{aligned}$$

This shows that the aggregated evidence is the diffused feature evidence with PPR. Hence, the aggregated evidence $\alpha_0^{\text{agg},(v)}$ is a strictly increasing function w.r.t. to the feature evidence of each individual neighbor $\alpha_0^{\text{ft},(u)}$ when $u \in \mathcal{N}(v)$. □

Theorem 12. *Lets consider a GPN model. Lets denote* $\bar{\boldsymbol{p}}^{agg,\,(v)} = \boldsymbol{\beta}^{agg,(v)} / \alpha_0^{agg,(v)}$ *the diffused categorical prediction for node* v *where* c^* *is its winning class. Further, lets denote* $\bar{\boldsymbol{p}}^{ft,\,(u)} = \boldsymbol{\beta}^{ft,(v)} / \alpha_0^{ft,(v)}$ *the non-diffused categorical prediction for a node* $u \in \mathcal{V}$*. First, there exists normalized weights* $\Pi'_{v,u}$ *such that* $\sum_{u \in \mathcal{V}} \Pi'_{v,u} \mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{ft,\,(u)}) \leq \mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{agg,\,(v)})$*. Second, if for any node* $u \in \mathcal{V}$ *the probability of* $\bar{\boldsymbol{p}}_{c^*}^{ft,\,(u)}$ *decreases, then* $\mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{agg,\,(v)})$ *increases.*

Proof. We first show the first part of the theorem. To this end, we use the relation between the aggregated and the feature evidence to derive a closed form relation between $\bar{\boldsymbol{p}}^{\text{agg},\,(\text{v})}$ and $\bar{\boldsymbol{p}}^{\text{ft},\,(\text{u})}$.

$$\begin{aligned}\bar{\boldsymbol{p}}^{\text{agg},\,(\text{v})} &= \frac{\boldsymbol{\beta}^{\text{agg},(v)}}{\alpha_0^{\text{agg},(v)}} \\ &= \frac{\sum_{u \in \mathcal{V}} \Pi_{v,u}^{ppr} \boldsymbol{\beta}^{\text{ft},(u)}}{\sum_{u' \in \mathcal{V}} \Pi_{v,u'}^{ppr} \alpha_0^{\text{ft},(u')}} \\ &= \frac{\sum_{u \in \mathcal{V}} \Pi_{v,u}^{ppr} \alpha_0^{\text{ft},(u)} \bar{\boldsymbol{p}}^{\text{ft},\,(\text{u})}}{\sum_{u' \in \mathcal{V}} \Pi_{v,u'}^{ppr} \alpha_0^{\text{ft},(u')}} \\ &= \sum_{u \in \mathcal{V}} \Pi'_{v,u} \bar{\boldsymbol{p}}^{\text{ft},\,(\text{u})}\end{aligned}$$

where $\Pi'_{v,u} = \frac{\Pi_{v,u}^{ppr} \alpha_0^{\text{ft},(u)}}{\sum_{u' \in \mathcal{V}} \Pi_{v,u'}^{ppr} \alpha_0^{\text{ft},(u')}}$. Hence, the probability vector $\bar{\boldsymbol{p}}^{\text{agg},\,(\text{v})}$ is a convex combination of the probability vectors $\bar{\boldsymbol{p}}^{\text{agg},\,(\text{u})}$ of other nodes. Further, using the concavity of the entropy function, we obtain the results:

$$\sum_{u \in \mathcal{V}} \Pi'_{v,u} \mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{\text{ft},\,(\text{u})}) \leq \mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{\text{agg},\,(\text{v})})$$

Second, we show the second part of the theorem. To this end, we suppose that, for a neighboring node $u \in \mathcal{N}(v)$, the probability of the winning class c^* decreases and the probability of another class $c^{'}$ increases i.e. $\bar{p}^{\text{ft, (u)}}_{c^*} - \epsilon$ and $\bar{p}^{\text{ft, (u)}}_{c^{'}} + \epsilon$. We define the following univariate function:

$$f(\epsilon) = \mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{\text{agg, (v)}}_{\epsilon})$$

where $\bar{\boldsymbol{p}}^{\text{agg, (v)}}_{\epsilon}$ is the new aggregated probability vector of the node v after the epsilon change of the probability vector for node u. Note that $\bar{p}^{\text{agg, (v)}}_{\epsilon,c^*} = \sum_{w \in \mathcal{V}} \Pi^{ppr}_{v,w} \bar{p}^{\text{ft, (w)}}_{c^*} - \Pi^{ppr}_{v,u}\epsilon$ and $\bar{p}^{\text{agg, (v)}}_{\epsilon,c^{'}} = \sum_{w \in \mathcal{V}} \Pi^{ppr}_{v,w} \bar{p}^{\text{ft, (w)}}_{c^{'}} + \Pi^{ppr}_{v,u}\epsilon$. We compute the derivative of $f(\epsilon)$:

$$\begin{aligned}
\frac{\partial f(\epsilon)}{\partial \epsilon} &= \frac{\partial(\bar{p}^{\text{agg, (v)}}_{\epsilon,c^*} \log \bar{p}^{\text{agg, (v)}}_{\epsilon,c^*} + \bar{p}^{\text{agg, (v)}}_{\epsilon,c^{'}} \log \bar{p}^{\text{agg, (v)}}_{\epsilon,c^{'}})}{\partial \epsilon} \\
&= \log \frac{\sum_{w \in \mathcal{V}} \Pi^{ppr}_{v,w} \bar{p}^{\text{ft, (w)}}_{c^*} - \Pi^{ppr}_{v,u}\epsilon}{\sum_{w \in \mathcal{V}} \Pi^{ppr}_{v,w} \bar{p}^{\text{ft, (w)}}_{c^{'}} + \Pi^{ppr}_{v,u}\epsilon}
\end{aligned}$$

Hence, we note that as long as the class c^* is winning (i.e $\bar{p}^{\text{agg, (v)}}_{\epsilon,c^*} = \sum_{w \in \mathcal{V}} \Pi^{ppr}_{v,w} \bar{p}^{\text{ft, (w)}}_{c^*} - \Pi^{ppr}_{v,u}\epsilon \geq \sum_{w \in \mathcal{V}} \Pi^{ppr}_{v,w} \bar{p}^{\text{ft, (w)}}_{c^{'}} + \Pi^{ppr}_{v,u}\epsilon = \bar{p}^{\text{agg, (v)}}_{\epsilon,c^{'}}$), the function $f(\epsilon)$ is increasing. It means that the entropy $\mathbb{H}\,\text{Cat}(\bar{\boldsymbol{p}}^{\text{agg, (v)}})$ increases when a neighboring node disagree more with the winning class c^*. In particular, note that the conclusion holds if the epsilon decrease of the winning class is compensated by the probability increase of K different classes. It would correspond to composing K decreasing functions $f_k(\epsilon_k)$ where $k \in \{0, ..., K\}$ and $\sum_k \epsilon_k = \epsilon$. □

E.2 Dataset Details

We consider the citation network datasets *CoraML* [282, 160, 156, 373, 44], *CiteSeer* [160, 156, 373], *PubMed* [309], *CoauthorPhysics* and *CoauthorCS* (based on the *Microsoft Academic Graph* from the *KDD Cup 2016* challenge) [379] as well as two co-purchase datasets, *AmazonPhotos* and *AmazonComputers* [281, 379]. Details are presented in the Table E.1. For all those datasets, we consider a train/val/test split of 5/15/80 using stratified sampling. For *CoraML*, this corresponds to the default split of 20 training samples per class with the difference of representing larger classes with more and smaller ones with less. We also note that this is significantly closer to the default split than approaches like [426, 201] introducing a 60/20/20 split. For all those datasets, we average the results over individual predictions from 10 random splits together with 10 random model initializations per split, i.e. 100 runs for each dataset and model. Those dataset are part of *PyTorch-Geometric* and under a MIT license. Besides those datasets, we also report results for the large dataset *OGBN Arxiv* dataset [196] which is based on the *Microsoft Academic Graph* [427] with the public split based on publication years. Since this makes random splits unnecessary, we report results as averages over 10 runs. This dataset is also available under a MIT license.

	CoraML	CiteSeer	PubMed	Amazon Computers	Amazon Photos	Coauthor CS	Coauthor Physics	OGBN - Arxiv
vertices	2,995	3,327	19,717	13,752	7,650	18,333	34,493	169,343
edges	16,316	9,104	88,648	491,722	238,162	163,788	495,924	2,315,598
homophily	78.86%	73.55%	80.24%	77.22%	82.72%	80.81%	93.15%	65.42%
feature dimension	2,879	3,703	500	767	745	6,805	8,415	128
max words	176	54	122	767	745	666	335	*N/A*
mean words	50.47	31.61	50.11	267.24	258.81	59.57	32.97	*N/A*
median words	49	32	50	204	193	45	27	*N/A*
classes	7	6	3	10	8	15	5	40
left-out classes	3	2	1	5	3	4	2	15
fraction left-out	44.91%	33.18%	39.94%	29.52%	40.46%	40.72%	18.18%	39.11%

Table E.1: Dataset details summarizing the graph size, the homophily ratio (fraction of intra-class edges), the number of classes, and statistics on the Bag-Of-Word features when available. In particular, *OGBN-Arxiv* uses averaged skip-gram embeddings for the nodes' features and thus does not require bag-of-word-features. Further, we provide the number of left-out classes and the fraction of left-out nodes for the LOC experiments.

E.3 Metrics

ACC - Accuracy is simply the fraction of predictions $\hat{y}^{(u)}$ that correspond to the ground-truth targets $y^{(u)}$ out of a set of all predictions of size N, i.e. $acc = \frac{1}{N} \sum_{u \in \mathcal{V}} 1_{y^{(u)} = \hat{y}^{(u)}}$.

Brier - The Brier Score is computed as $1/C \sum_{v}^{N} ||\boldsymbol{p}^{(v)} - \boldsymbol{y}^{(v)}||_2$ where $\boldsymbol{y}^{(v)}$ represent the one-hot encoded ground-truth label for a node v while $\boldsymbol{p}^{(v)}$ is the predicted probability score.

ECE - The expected calibration error on the other hand requires binning the predicted probability scores into M equally spaced bins with $conf(B_m)$ being the average probability score in the bin m. With $acc(B_m)$ being the average accuracy of predictions

in bin m, we obtain the final metric as

$$ECE = \sum_{m}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)| \tag{E.1}$$

OOD - Furthermore, we use **AUC-ROC** and **AUC-PR** scores for OOD-detection experiments. This problem is considered as a binary classification problem with the positive targets being the OOD-nodes and the negative targets being ID-data points. We use the same aleatoric and epistemic uncertainty measures used in [67]. For aleatoric uncertainty measures, we use $u_{\text{alea}}^{(v)} = -\max_c \bar{\mathbf{p}}_c^{(u)}$. For epistemic uncertainty, we use $u_{\text{epist}}^{(v)} = -\alpha_0^{(v)}$ for Dirichlet-based methods and $u_{\text{epist}}^{(v)} = \frac{1}{\sum_c \text{Var}\, \bar{\mathbf{p}}_c^{(u)}}$ for other methods. For Dirichlet-based methods and GPs, the corresponding quantities are predicted directly. For ensemble and dropout baselines, these quantities are computed based on the empirical mean and empirical variance.

Note that the *vacuity* uncertainty measure proposed in [470] and motivated from work on *subjective logic* is just the inverse transformation of α_0 given by $u_{\text{vacuity}} = \frac{C}{\alpha_0}$. Hence, the AUC-ROC and AUC-PR scores which evaluate the ranking of the examples lead to the *exact* same final scores using u_{vacuity} or u_{epist}.

E.4 Model Details

We follow [379, 230, 470] and baselines from the OGBN leaderboard[1] for the choice of the architectures. By default, we use a hidden dimension of $h = 64$ and $l = 2$ layers for parametric models on all datasets except for *OGBN Arxiv*. In this case, we use early stopping with a patience of 50 and a maximum of $100,000$ epochs. For *OGBN Arxiv*, we use a hidden dimension of $h = 256$ with $l = 3$ layers and use batch-norm. In this case, we use early stopping with a patience of 200, a maximum number of $1,000$ epochs and no weight-decay for all models. For Gaussian Processes, we implement our experiment pipelines and models in *PyTorch* [341] and rely on *PyTorch-Geometric* [133]. For all models, we use the Adam optimizer [219] with its default parameters and a learning rate of 0.01. For further details, we provide the code in the supplementary material.

GPN - We use a similar backbone architecture as for APPNP [229].[2] We report all the used hyper-parameters in Table E.2. Similarly to [68], we use a certainty budget N which scales exponentially w.r.t. the latent dimension (i.e. $N_H = \sqrt{4\pi}^H$) and 5 warm-up epochs maximizing the log likelihood of the normalizing flows. Furthermore, we use $K = 10$ power-iterations steps to approximate PPR scores. We do not use weight decay for the Normalizing Flows. Those parameters have been obtained after conducting an ablation and a hyper-parameter study on the *CoraML* and *OGBN Arxiv* datasets (see Appendices E.5.1 and E.5.2). Finally, we recall for completeness the closed-form of the Bayesian loss introduced in [67] when $\mathbb{Q}^{post,(v)} = \text{Dir}(\boldsymbol{\alpha}^{(v)})$ and $\mathbb{P}(y^{(v)} \,|\, \boldsymbol{p}^{(v)}) =$

[1] `https://ogb.stanford.edu/docs/leader_nodeprop/#ogbn-arxiv`
[2] Code available at `https://www.daml.in.tum.de/graph-postnet`

$\mathrm{Cat}(y^{(v)} \mid \boldsymbol{p}^{(v)})$:

$$\mathcal{L}^{(v)} = -\,\mathbb{E}_{\boldsymbol{p}^{(v)} \sim \mathbb{Q}^{post,(v)}} \left[\log \mathbb{P}(y^{(v)} \mid \boldsymbol{p}^{(v)}) \right] - \lambda\, \mathbb{H}\, \mathbb{Q}^{post,(v)} \tag{E.2}$$

The expected likelihood term is equal to:

$$\mathbb{E}_{\boldsymbol{p} \sim \mathrm{Dir}(\boldsymbol{\alpha})} [\log \mathrm{Cat}(y \mid \boldsymbol{p})] = \psi(\alpha_y) - \psi(\alpha_0) \tag{E.3}$$

The entropy term is equal to:

$$\mathbb{H}[\mathrm{Dir}(\boldsymbol{\alpha})] = \log B(\boldsymbol{\alpha}) + (\alpha_0 - C)\psi(\alpha_0) - \sum_c (\alpha_c - 1)\psi(\alpha_c) \tag{E.4}$$

	H	L	n_{layers}	n_{radial}	ACT	p_{drop}	τ	budget	λ	weight decay
(1)	64	10	2	10	ReLU	0.5	0.2	$N_H \cdot C$	$1.0e-05$	0.0005
(2)	64	16	2	10	ReLU	0.5	0.1	N_H	$1.0e-3$	0.001
(3)	256	16	3	10	ReLU + BN	0.25	0.2	N_H	$1.0e-3$	0.0

Table E.2: GPN hyperparameters used in our experiments. (1) is used for *Amazon Photos* and *Amazon Computers* datasets, (3) is used for *OGBN-Arxiv* dataset and (2) is used for all other datasets. For all those settings, we furthermore use $K = 10$ power-iterations steps, 5 warmup epochs for the Normalizing Flows, and no weight decay for the normalizing flows.

GKDE - We adopt the Graph Kernel Dirichlet Estimate from [470] as a standalone and parameterless baseline. With $d_{v,u}$ being the shortest path between nodes v and u and the Gaussian transformation $g(d_{v,u}) = {}^{1}/_{\sigma\sqrt{2\pi}} \exp\left({}^{-d_{v,u}^2}/_{2\sigma^2}\right)$, a Dirichlet estimate is obtained in the following way

$$\boldsymbol{\alpha}^{(v)} = 1 + \boldsymbol{e}^{(v)}$$

$$\text{with} \quad \boldsymbol{e}^{(v)} = \sum_{u \in \mathcal{T}} \boldsymbol{h}(y^{(u)}, d_{v,u}) \quad \text{and} \quad h_c(y^{(u)}, d_{v,u}) = \begin{cases} 0 & y^{(u)} \neq c \\ g(d_{v,u}) & y^{(u)} = c \end{cases}$$

Similar to [464], we use $\sigma = 1$. We would also like to point out that the computation of this kernel requires extracting the shorted distance of each node to each labeled node $u \in \mathcal{T}$. Larger datasets like *OGBN-Arxiv* come with larger sets of labeled data with the size $|\mathcal{T}|$ having a same magnitude as the number of nodes in the graph, i.e. $|\mathcal{V}|$. This approach therefore scales quadratically with the number of nodes in the graph and therefore does not generalize well to larger datasets.

LP - Following the idea of the GKDE baseline, we propose similar Dirichlet estimates by relying on Label Propagation which achieve strong results in Left-Out classes experiments. GKDE extracts Dirichlet evidence scores by relying on node distances. We propose taking the density of labeled nodes in neighborhoods instead. To this end, we

define one initial conditional density per class $\rho_0(u \mid c)$ and diffuse them with Personalized Page Rank i.e.

$$\rho_0(u \mid c) = \begin{cases} 0 & u \in \mathcal{U} \\ \frac{1}{|\mathcal{L}_c|} \cdot \delta_{y^{(u)},c} & u \in \mathcal{L} \end{cases} \rightarrow \rho(v \mid c) = \sum_u \Pi^{ppr}_{v,u} \cdot \rho_0(u \mid c)$$

where $\mathcal{L}_c$ is the set of labeled nodes for class c. The diffused density $\rho(v \mid c)$ is still a valid density i.e. $\sum_c \sum_u \rho(u \mid c) = \sum_c \sum_u \rho_0(u \mid c) = 1$. Finally, we use this diffused conditional densities to obtain Dirichlet evidence scores in a similar fashion to the GKDE kernel [470], i.e. $\alpha_c^{(v)} = 1.0 + \rho(v \mid c)$. The diffusion is performed with power-iteration similar to APPNP [229]. We use a teleport probability of $\tau = 0.1$ and $K = 10$ power iteration steps.

Gaussian Processes - We use the official implementations for **MaternGGP** [49] and the re-implementation[3] for **GGP** [315]. The re-implementation transfers the official implementation to Tensorflow 2.0 [2] which we wrapped in our Pytorch pipeline. Since those approaches do not scale well to large real-world datasets, we restrict to a single random initialization. GGP only finished the experiments on CoraML and CiteSeer. Similarly, MaternGGP did not finished the experiments on OGBN Arxiv. For recall, we set the memory an time limits to 64 GiB and 12 hours per run. For comparison, note that all GNN-based models require significantly less memory and finished *all* runs in a couple of hours.

GKDE-SGCN - We use the hyper-parameters suggested in the original paper [470]. We set the regularization factor λ to 0.001. This factor determines the weight of the Graph-Kernel-Dirichlet-Estimate which is key for OOD detection in graphs. Note that we did not use different factors for OOD experiments and classification experiments contrary to [470] since it leads to the leakage of task information. Indeed, the clean accuracy is significantly higher for $\lambda = 0.001$ compared to $\lambda = 0.1$. For *OGBN Arxiv*, we did not use teacher training as it harmed the performance. In this case, we used a dropout probability of $p = 0.5$ and $\lambda = 0.0001$ after a small grid-search with the overall architecture following the initial remarks above.

APPNP - We follow [230] and use an architecture that is comparable to other GNN approaches. We use ReLU activations, dropout with $p = 0.5$, no dropout on the adjacency matrix, a teleport probability of $\tau = 0.1$ and $K = 10$ power iteration steps. We also use a weight decay of $\lambda = 0.0001$.

VGCN - We use ReLU activations, dropout with $p = 0.8$, and a weight decay of $\lambda = 0.0001$. For the larger dataset *OGBN Arxiv*, we use a dropout of $p = 0.5$. **DropEdge** is similar to the Vanilla GCN model with an additional dropout on the edges with a dropout probability of $p = 0.5$ on both features and edges. For evaluation of dropout models, i.e. **DropEdge** and **VGCN-Dropout**, we use $S = 10$ Monte-Carlo samples having shown a reasonable estimate with more samples not leading to a visible improvement. For ensembles, i.e. **VGCN-Ensemble**, we use an ensemble of models of 10 different random initializations. For **VGCN-Energy**, we follow [259] and use a temperature of $T = 1.0$.

[3] https://github.com/FelixOpolka/GGP-TF2

VGCN-BNN - We follow *Bayes by Backprop* [42] and adopt a Bayesian GNN with uncertain weights. We use a hidden dimension of $h = 32$. This is equivalent to $h = 64$ for other models as each weight is represented by one mean parameter μ and one variance parameter Σ. We use 10 bayesian samples in our experiments. We follow the grid search suggested in the original paper [42]. We finally adopt $\pi = 0.75$, $\sigma_1 = 1.0$, and $\sigma_2 = 1.0e-6$. Since this models assumes uncertain weights, we do not apply any weight decay during training. Note that we do not report results for the larger dataset *OGBN Arxiv* for this baseline.

RGCN - We follow [475] for the hyper-parameter selection. We use a hidden size of $h = 32$. Again this is equivalent to $h = 64$ since RGCN models a mean parameter μ and a variance parameter Σ per layer. We further use dropout $p = 0.5$ on the features, $\gamma = 1$, $\beta_{KL} = 5.04 - 4$ and $\beta_{reg} = 5.0e-4$. As the latter is already a weight regularization in the loss, we do not apply weight decay.

E.5 Additional Experiments

E.5.1 Additional Experiments - Ablation Study

In this section, we evaluate the contribution of each component of GPN. To this end, we use **PostNet** which first trains the feature encoder and the normalizing flows without diffusion and **PostNet-Diff** which diffuses the ablation-counts only at test time. Further, we also compare to **APPNP** [229] which does not model the epistemic uncertainty with density estimation and **GPN-LOG** which diffuse the parameters $\log(\beta_c^{\text{ft},(v)})$ instead of $\beta_c^{\text{ft},(v)}$. We observed that training with diffusion is beneficial for all metrics. Further, we noted that diffusing $\log(\beta_c^{\text{ft},(v)})$ improves accuracy and calibration to the cost of a lower Left-Out classes detection scores. Similarly, APPNP also showed better accuracy when diffusing logits instead of softmax outputs in the original paper [229]. Finally, APPNP achieves significantly worse results for all OOD detection tasks showing the benefit of modelling the epistemic uncertainty.

Model	ACC	ECE	Ber-FT-AUC-ROC	$\mathcal{N}$-FT-AUC-ROC	LOC-ID-ACC	LOC-EPIS-AUC-ROC
APPNP	82.18 ± 0.08	8.46 ± 0.09	60.51 ± 0.06	16.32 ± 0.31	88.71 ± 0.03	84.75 ± 0.06
PostNet	52.24 ± 0.90	8.22 ± 1.39	83.09 ± 4.41	100.00 ± 0.00	69.37 ± 0.50	70.14 ± 0.93
PostNet-Diff	77.10 ± 0.58	30.35 ± 0.86	83.09 ± 4.41	100.00 ± 0.00	86.18 ± 0.65	78.45 ± 0.80
GPN	81.57 ± 0.18	12.77 ± 1.11	84.69 ± 4.60	100.00 ± 0.00	89.17 ± 0.39	87.34 ± 0.76
GPN-LOG-β	82.90 ± 0.26	8.77 ± 0.33	91.24 ± 2.67	100.00 ± 0.00	91.17 ± 0.59	70.88 ± 1.74

Table E.3: Accuracy, Calibration, and OOD-detection results for the ablation study on CoraML for the validation split. All models use epistemic uncertainty measures except APPNP which uses an aleatoric measure.

E.5.2 Additional Experiments - Hyper-parameter study

Besides the previously mentioned ablation study, we also performed a study on the influences of hyperparameters. We show findings for the *CoraML* dataset averaging runs

Latent Dim	ACC	ECE	Ber-FT-AUC-ROC	$\mathcal{N}$-FT-AUC-ROC	LOC-ID-ACC	LOC-Epis-AUC-ROC
6	79.18 ± 0.16	10.50 ± 0.56	81.75 ± 4.74	100.00 ± 0.00	90.17 ± 0.41	88.65 ± 0.58
10	79.14 ± 0.44	10.96 ± 0.23	77.92 ± 4.44	100.00 ± 0.00	89.39 ± 0.58	87.39 ± 0.43
16	81.57 ± 0.18	12.77 ± 1.11	84.69 ± 4.60	100.00 ± 0.00	89.17 ± 0.39	87.34 ± 0.76

Table E.4: Accuracy, Calibration, and OOD-detection results of GPN on the CoraML dataset with latent dimensions in $[6, 10, 16]$.

Radial Layers	ACC	ECE	Ber-FT-AUC-ROC	$\mathcal{N}$-FT-AUC-ROC	LOC-ID-ACC	LOC-Epis-AUC-ROC
6	80.04 ± 0.85	11.26 ± 0.48	84.26 ± 2.38	100.00 ± 0.00	90.74 ± 0.32	86.62 ± 0.25
10	81.57 ± 0.18	12.77 ± 1.11	84.69 ± 4.60	100.00 ± 0.00	89.17 ± 0.39	87.34 ± 0.76
16	75.88 ± 0.95	8.77 ± 0.32	94.06 ± 1.37	100.00 ± 0.00	89.39 ± 0.20	84.14 ± 0.12

Table E.5: Accuracy, Calibration, and OOD-detection results of GPN on the CoraML dataset with the number of radial layers in $[6, 10, 16]$.

with 3 different random dataset splits and 2 different random initializations. We analyzed the influence of the latent dimension, the number of radial layers, the teleport probability, the certainty budget, weight decay, and entropy regularization.

E.5.3 Additional Experiments - Misclassification

Work like [470] found that measures of aleatoric uncertainty are well suited for detecting inputs which are not classified correctly. Epistemic measures of uncertainty are found to be better suited for detecting OOD samples. Orthogonal work like [175, 335] uses calibration to determine how reliable predictions are. We adopt this point of view while reporting results for the task of OOD detection using epistemic measures. We also present aleatoric measures as a reference because simple baselines without any intrinsic uncertainty estimates solely can quantify uncertainty through simple aleatoric uncertainty measures. To facilitate an easy comparison to work like [470], we also present results for misclassification experiments in Table E.8 and in Table E.9. As in [464], we can observe that aleatoric uncertainty mostly is better for detecting misclassified samples for all models than epistemic uncertainty. GPN achieves a similar performance in this task while also showing that for some datasets measures not accounting for network effects can detect misclassified samples better than measures which account for network effects.

τ_{teleport}	ACC	ECE	Ber-FT-AUC-ROC	$\mathcal{N}$-FT-AUC-ROC	LOC-ID-ACC	LOC-Epis-AUC-ROC
0.05	80.39 ± 0.33	11.98 ± 0.53	85.31 ± 2.79	100.00 ± 0.00	89.32 ± 0.41	86.63 ± 0.65
0.1	81.57 ± 0.18	12.77 ± 1.11	84.69 ± 4.60	100.00 ± 0.00	89.17 ± 0.39	87.34 ± 0.76
0.2	80.00 ± 0.30	8.27 ± 0.59	86.65 ± 1.46	100.00 ± 0.00	88.89 ± 0.31	87.30 ± 0.30

Table E.6: Accuracy, Calibration, and OOD-detection results of GPN on the CoraML dataset with a teleport probability τ in $[0.05, 0.1, 0.2]$.

Budget	ACC	ECE	Ber-FT-AUC-ROC	$\mathcal{N}$-FT-AUC-ROC	LOC-ID-ACC	LOC-Epis-AUC-ROC
N_H	81.57 ± 0.18	12.77 ± 1.11	84.69 ± 4.60	100.00 ± 0.00	89.17 ± 0.39	87.34 ± 0.76
$N_H \cdot C$	79.76 ± 0.93	11.24 ± 0.89	82.82 ± 5.68	100.00 ± 0.00	88.68 ± 0.75	86.39 ± 1.27

Table E.7: Accuracy, Calibration, and OOD-detection results of GPN on the CoraML dataset with a budget that scales exponentially w.r.t. the latent dimension and a budget that scales exponentially w.r.t. the latent dimension and the number of classes.

E.5.4 Additional Experiments - OOD Detection

In this section, we provide additional results for the OOD detection experiments for the 8 datasets and the 13 baselines. We show the results for feature perturbations experiments using AUC-ROC and AUC-APR scores in Table E.10 and Table E.11. We show results for clean accuracy and calibration on the unperturbed graphs, and Left-Out classes using AUC-ROC and AUC-APR scores in Tables E.12 and E.13. Similarly to Table 8.1, we observe that GPN achieves the best results for the detection of feature perturbations with uncertainty without network effects by a significant margin while still maintaining a high accuracy. Further, GPN also performs favourably on Left-Out classes experiments using the uncertainty measures with network effects. All these observations show that GPN disentangles well between uncertainty without and with network effects while being robust against feature perturbations.

E.5.5 Additional Experiments - Attributed Graph Shifts

In this section, we provide additional results for the attributed graph shifts experiments. We show the results of the feature shifts and the results of the edge shifts in Figs. E.2 to E.9. The feature shifts include Bernoulli and Normal shifts (i.e. $\boldsymbol{x}^{(v)} \sim \text{Ber}(0.5)$ and $\boldsymbol{x}^{(v)} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$) with up to 99% of the nodes being perturbed. The edge shifts include randomly moved edges and DICE attacks [431] where we perturb up to 99% of the edges in the graph. The results are consistent with the observations made in Section 8.4. For feature shifts, we observe that GPN is more robust to feature perturbations than competitors for accuracy and calibration similarly to results Fig. 8.3. GPN is particularly robust against unit Gaussians perturbations. Further, as desired, GPN is more epistemically uncertain when features of a larger fraction of nodes are perturbed. For edge shifts, all models including GPN become more aleatorically uncertain. This aligns with des. 8.3.3. Furthermore, the average epistemic uncertainty of GPN remains constant which is reasonable since the average evidence of a node's neighborhood remains constant. We observed that the exponential activation in the last layer of GKDE-GCN leads to numerical instabilities under perturbations.

E.5.6 Additional Experiments - Qualitative Evaluation

In this section, we provide additional qualitative evaluations. Therefore, we present the abstracts of the most epistemically uncertain papers and the most epistemically certain

Dataset	Model	Alea w/ Net	Epis w/ Net	Epis w/o Net	Alea w/ Net	Epis w/ Net	Epis w/o Net
		misclassification AUC-ROC			misclassification AUC-PR		
CoraML	APPNP	**83.64 ± 0.08**	*n.a.*	*n.a.*	48.39 ± 0.19	*n.a.*	*n.a.*
	VGCN	81.02 ± 0.07	*n.a.*	*n.a.*	48.30 ± 0.23	*n.a.*	*n.a.*
	VGCN-Dropout	81.42 ± 0.06	65.52 ± 0.28	*n.a.*	49.34 ± 0.17	26.11 ± 0.21	*n.a.*
	VGCN-Energy	81.02 ± 0.07	*n.a.*	*n.a.*	48.30 ± 0.23	*n.a.*	*n.a.*
	VGCN-Ensemble	81.12 ± 0.17	72.62 ± 0.19	*n.a.*	49.16 ± 0.59	31.88 ± 0.29	*n.a.*
	VGCN-BNN	80.64 ± 0.10	65.40 ± 0.48	*n.a.*	47.49 ± 0.26	26.70 ± 0.33	*n.a.*
	GKDE-GCN	80.80 ± 0.14	76.83 ± 0.17	*n.a.*	**49.61 ± 0.47**	45.87 ± 0.48	*n.a.*
	GPN	81.19 ± 0.13	78.10 ± 0.26	77.46 ± 0.24	49.51 ± 0.26	44.42 ± 0.32	43.31 ± 0.41
CiteSeer	APPNP	73.55 ± 0.08	*n.a.*	*n.a.*	51.70 ± 0.15	*n.a.*	*n.a.*
	VGCN	74.64 ± 0.09	*n.a.*	*n.a.*	54.32 ± 0.18	*n.a.*	*n.a.*
	VGCN-Dropout	74.81 ± 0.11	64.09 ± 0.16	*n.a.*	55.12 ± 0.23	39.41 ± 0.19	*n.a.*
	VGCN-Energy	74.64 ± 0.09	*n.a.*	*n.a.*	54.32 ± 0.18	*n.a.*	*n.a.*
	VGCN-Ensemble	74.42 ± 0.26	68.15 ± 0.23	*n.a.*	53.28 ± 0.33	43.30 ± 0.30	*n.a.*
	VGCN-BNN	73.28 ± 0.11	61.68 ± 0.29	*n.a.*	54.62 ± 0.21	37.99 ± 0.24	*n.a.*
	GKDE-GCN	75.45 ± 0.11	73.83 ± 0.12	*n.a.*	54.78 ± 0.19	53.57 ± 0.20	*n.a.*
	GPN	**75.89 ± 0.15**	74.16 ± 0.17	72.50 ± 0.12	**60.78 ± 0.32**	59.32 ± 0.40	52.10 ± 0.18
PubMed	APPNP	80.98 ± 0.02	*n.a.*	*n.a.*	37.79 ± 0.08	*n.a.*	*n.a.*
	VGCN	81.16 ± 0.02	*n.a.*	*n.a.*	38.24 ± 0.08	*n.a.*	*n.a.*
	VGCN-Dropout	80.46 ± 0.04	72.69 ± 0.09	*n.a.*	38.63 ± 0.11	25.90 ± 0.09	*n.a.*
	VGCN-Energy	81.16 ± 0.02	*n.a.*	*n.a.*	38.24 ± 0.08	*n.a.*	*n.a.*
	VGCN-Ensemble	**81.31 ± 0.06**	79.30 ± 0.08	*n.a.*	38.06 ± 0.33	31.73 ± 0.19	*n.a.*
	VGCN-BNN	79.96 ± 0.07	72.63 ± 0.45	*n.a.*	39.31 ± 0.08	27.59 ± 0.39	*n.a.*
	GKDE-GCN	80.95 ± 0.09	73.99 ± 0.27	*n.a.*	39.64 ± 0.10	33.19 ± 0.14	*n.a.*
	GPN	80.46 ± 0.13	75.38 ± 0.25	80.48 ± 0.13	40.74 ± 0.19	35.11 ± 0.11	**51.12 ± 0.52**

Table E.8: Misclassification Scores on the clean graphs given as AUC-ROC and AUC-PR scores. AUC-ROC and AUC-APR scores are given as *[Alea w/ Net] / [Epist w/ Net] / [Epist w/o Net]*. *n.a.* means either model or metric not applicable. Bold numbers indicate the best model across all the uncertainty metrics for each dataset.

papers in CoraML in Table E.14 and Table E.15. Most epistemically certain papers shows significantly more reasonable abstracts compared to most epistemically uncertain papers.

Additionally, we provide visualizations of the latent space of GPN on the clean CoraML graph in Fig. E.10, and on the CoraML graph where 10% of the nodes are perturbed with the unit Gaussian perturbation in Fig. E.10, and on Left-Out classes experiments in Fig. E.12. We used T-SNE projections for 2D visualizations. We observed that the latent representations correlate with the class assignment. Further, GPN is capable to separate nodes with perturbed features in the latent space. The nodes with perturbed features are assigned high uncertainty without network effects but low uncertainty with network effects. This stresses the capacity of GPN to recover from feature perturbations.

E.5.7 Additional Experiments - Inference & Training Time

We provide a comparison of inference times for most of the datasets and models under consideration in Table E.16 and a comparison of training times in Table E.17. GPN needs a single pass for uncertainty estimation but requires the additional evaluation of one normalizing flow per class compared to APPNP. Hence, GPN brings a small

computational overhead for uncertainty estimation during inference but is significantly faster than ensemble or dropout approaches. Furthermore, GPN is usually converging relatively fast during training and does not require a pre-computing kernel values. In contrast, GKDE-GCN requires the computation of the underlying Graph Kernel with a complexity of $\mathcal{O}(N^2)$ where N is the number of nodes in the graph (see Appendix E.4). Finally, GPN is significantly more efficient than dropout or ensemble approaches as it does not require training or evaluation of multiple models.

E.6 Desiderata Diagram

We provide a larger version of Fig. 8.1 to visualize the distinction between aleatoric and epistemic uncertainty and the distinction between uncertainty without and with network effects in Fig. E.13. These two distinctions are used in the desiderata in Section 8.3.1.

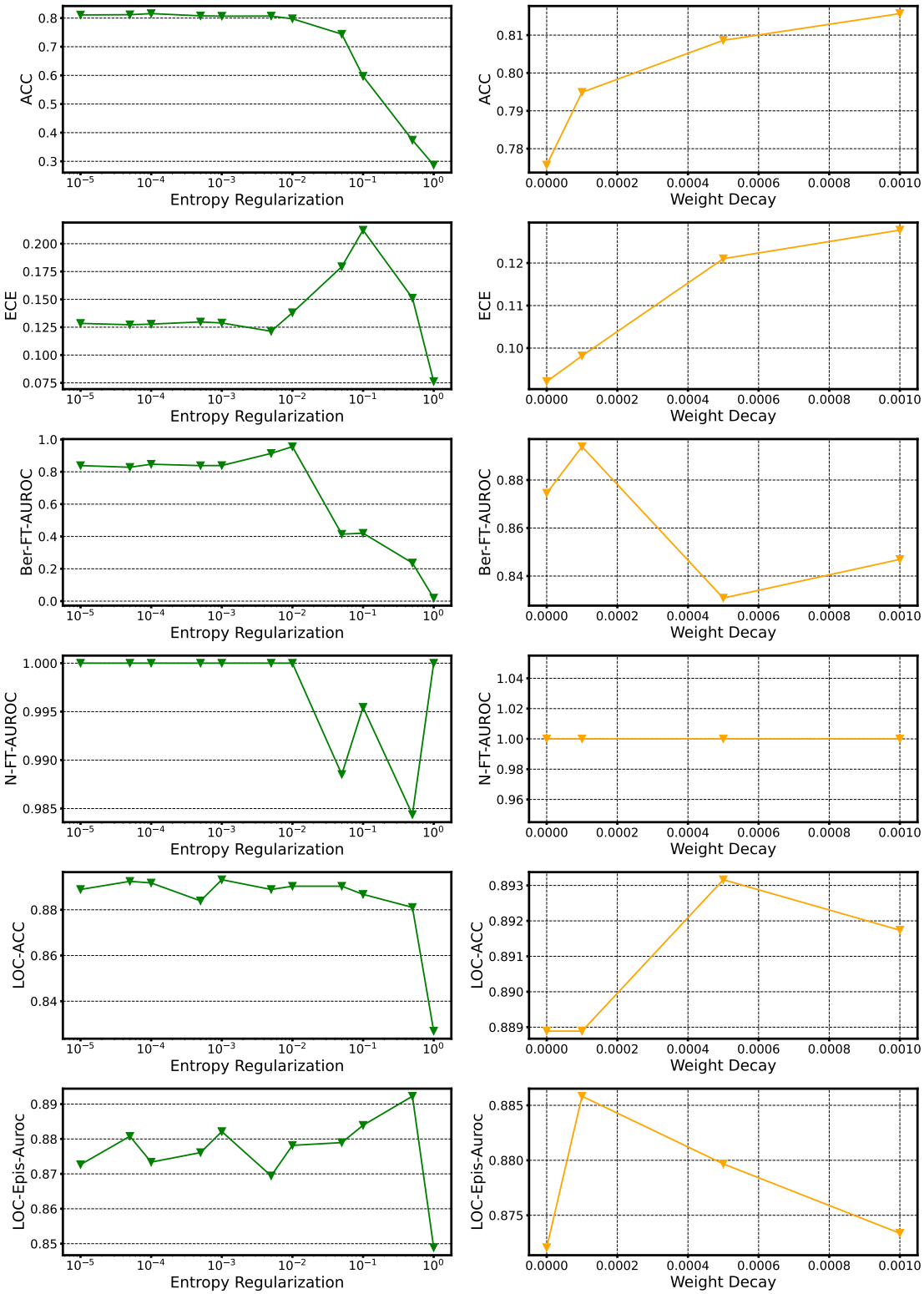


Figure E.1: Accuracy, Calibration, and OOD-detection results of GPN on CoraML for the entropy regularization factor and the weight decay.

Dataset	Model	Alea w/ Net	Epis w/ Net	Epis w/o Net	Alea w/ Net	Epis w/ Net	Epis w/o Net
		misclassification AUC-ROC			misclassification AUC-PR		
Amazon Computers	APPNP	79.75 ± 0.03	*n.a.*	*n.a.*	45.10 ± 0.11	*n.a.*	*n.a.*
	VGCN	82.08 ± 0.03	*n.a.*	*n.a.*	45.52 ± 0.12	*n.a.*	*n.a.*
	VGCN-Dropout	$\mathbf{82.70 \pm 0.04}$	72.02 ± 0.11	*n.a.*	47.19 ± 0.17	31.45 ± 0.14	*n.a.*
	VGCN-Energy	82.08 ± 0.03	*n.a.*	*n.a.*	45.52 ± 0.12	*n.a.*	*n.a.*
	VGCN-Ensemble	82.05 ± 0.06	67.62 ± 0.44	*n.a.*	45.40 ± 0.24	26.33 ± 0.37	*n.a.*
	VGCN-BNN	82.15 ± 0.17	48.65 ± 0.82	*n.a.*	$\mathbf{69.61 \pm 0.42}$	30.87 ± 0.43	*n.a.*
	GKDE-GCN	79.66 ± 0.19	73.66 ± 0.15	*n.a.*	63.26 ± 0.57	56.93 ± 0.54	*n.a.*
	GPN	82.20 ± 0.10	77.58 ± 0.16	70.06 ± 0.19	47.93 ± 0.42	41.80 ± 0.44	28.70 ± 0.51
Amazon Photos	APPNP	85.74 ± 0.06	*n.a.*	*n.a.*	37.00 ± 0.20	*n.a.*	*n.a.*
	VGCN	87.94 ± 0.05	*n.a.*	*n.a.*	48.35 ± 0.19	*n.a.*	*n.a.*
	VGCN-Dropout	$\mathbf{89.52 \pm 0.05}$	78.46 ± 0.10	*n.a.*	50.27 ± 0.19	23.08 ± 0.12	*n.a.*
	VGCN-Energy	87.94 ± 0.05	*n.a.*	*n.a.*	48.35 ± 0.19	*n.a.*	*n.a.*
	VGCN-Ensemble	88.08 ± 0.16	76.05 ± 0.54	*n.a.*	49.40 ± 0.67	24.13 ± 0.67	*n.a.*
	VGCN-BNN	84.17 ± 0.19	51.84 ± 0.66	*n.a.*	51.05 ± 0.64	18.69 ± 0.55	*n.a.*
	GKDE-GCN	84.11 ± 0.29	75.07 ± 0.50	*n.a.*	$\mathbf{54.35 \pm 0.58}$	45.43 ± 0.68	*n.a.*
	GPN	87.21 ± 0.10	83.38 ± 0.30	79.93 ± 0.23	46.32 ± 0.39	37.07 ± 0.60	29.90 ± 0.71
Coauthor CS	APPNP	89.92 ± 0.03	*n.a.*	*n.a.*	37.98 ± 0.12	*n.a.*	*n.a.*
	VGCN	89.46 ± 0.03	*n.a.*	*n.a.*	38.86 ± 0.10	*n.a.*	*n.a.*
	VGCN-Dropout	88.46 ± 0.04	79.03 ± 0.08	*n.a.*	38.06 ± 0.12	17.98 ± 0.09	*n.a.*
	VGCN-Energy	89.46 ± 0.03	*n.a.*	*n.a.*	38.86 ± 0.10	*n.a.*	*n.a.*
	VGCN-Ensemble	89.51 ± 0.08	86.61 ± 0.09	*n.a.*	38.74 ± 0.23	30.60 ± 0.38	*n.a.*
	VGCN-BNN	89.01 ± 0.06	78.40 ± 0.23	*n.a.*	38.17 ± 0.17	19.06 ± 0.20	*n.a.*
	GKDE-GCN	89.24 ± 0.05	80.98 ± 0.13	*n.a.*	39.30 ± 0.27	30.52 ± 0.25	*n.a.*
	GPN	85.72 ± 0.15	81.56 ± 0.29	$\mathbf{94.41 \pm 0.11}$	46.12 ± 0.32	38.98 ± 0.28	$\mathbf{77.26 \pm 0.45}$
Coauthor Physics	APPNP	93.27 ± 0.02	*n.a.*	*n.a.*	38.14 ± 0.09	*n.a.*	*n.a.*
	VGCN	92.86 ± 0.02	*n.a.*	*n.a.*	37.19 ± 0.10	*n.a.*	*n.a.*
	VGCN-Dropout	92.28 ± 0.03	89.85 ± 0.04	*n.a.*	35.47 ± 0.11	23.70 ± 0.08	*n.a.*
	VGCN-Energy	92.86 ± 0.02	*n.a.*	*n.a.*	37.19 ± 0.10	*n.a.*	*n.a.*
	VGCN-Ensemble	92.95 ± 0.07	91.92 ± 0.07	*n.a.*	37.96 ± 0.28	28.44 ± 0.21	*n.a.*
	VGCN-BNN	92.44 ± 0.09	89.03 ± 0.27	*n.a.*	36.79 ± 0.21	25.11 ± 0.49	*n.a.*
	GKDE-GCN	92.77 ± 0.02	86.12 ± 0.06	*n.a.*	37.08 ± 0.11	25.13 ± 0.10	*n.a.*
	GPN	91.14 ± 0.04	89.63 ± 0.07	$\mathbf{93.89 \pm 0.05}$	41.43 ± 0.13	35.64 ± 0.16	$\mathbf{59.03 \pm 0.28}$
OGBN Arxiv	APPNP	77.55 ± 0.05	*n.a.*	*n.a.*	54.57 ± 0.14	*n.a.*	*n.a.*
	VGCN	77.89 ± 0.05	*n.a.*	*n.a.*	54.87 ± 0.12	*n.a.*	*n.a.*
	VGCN-Dropout	78.11 ± 0.05	71.74 ± 0.14	*n.a.*	55.40 ± 0.13	43.43 ± 0.18	*n.a.*
	VGCN-Energy	77.89 ± 0.05	*n.a.*	*n.a.*	54.87 ± 0.12	*n.a.*	*n.a.*
	VGCN-Ensemble	**78.14**	71.48	*n.a.*	53.95	42.87	*n.a.*
	GKDE-GCN	77.47 ± 0.33	77.55 ± 0.33	*n.a.*	$\mathbf{61.62 \pm 1.00}$	62.33 ± 1.00	*n.a.*
	GPN	75.44 ± 0.19	72.71 ± 0.28	61.45 ± 0.49	55.64 ± 0.37	52.99 ± 0.49	39.37 ± 0.42

Table E.9: Misclassification Scores on the clean graphs given as AUC-ROC and AUC-PR scores. AUC-ROC and AUC-APR scores are given as *[Alea w/ Net] / [Epist w/ Net] / [Epist w/o Net]*. *n.a.* means either model or metric not applicable. Bold numbers indicate the best model across all the uncertainty metrics for each dataset.

		$\boldsymbol{x}^{(v)} \sim \text{Ber}(0.5)$				$\boldsymbol{x}^{(v)} \sim \mathcal{N}(0,1)$			
	Model	**OOD-ACC**	**OOD-AUC-ROC**			**OOD-ACC**	**OOD-AUC-ROC**		
			Alea w/ Net	*Epist w/ Net*	*Epist w/o Net*		*Alea w/ Net*	*Epist w/ Net*	*Epist w/o Net*
CoraML	APPNP	80.85±0.09	64.41±0.02	*n.a.*	*n.a.*	17.99±0.36	7.98±0.13	*n.a.*	*n.a.*
	VGCN	78.90±0.09	63.68±0.03	*n.a.*	*n.a.*	18.37±0.31	9.34±0.13	*n.a.*	*n.a.*
	RGCN	79.78±0.16	70.30±0.05	*n.a.*	*n.a.*	33.37±0.35	32.13±0.28	*n.a.*	*n.a.*
	VGCN-Dropout	77.76±0.15	62.06±0.06	50.38±0.12	*n.a.*	18.28±0.35	40.53±0.25	71.06±0.29	*n.a.*
	DropEdge	77.40±0.14	63.10±0.04	52.84±0.10	*n.a.*	16.60±0.26	23.10±0.29	46.82±0.41	*n.a.*
	VGCN-Energy	78.90±0.09	63.68±0.03	66.26±0.04	*n.a.*	18.37±0.31	9.34±0.13	0.32±0.03	*n.a.*
	VGCN-Ensemble	78.00±0.00	63.58±0.00	56.81±0.03	*n.a.*	21.00±0.00	33.72±0.02	64.92±0.08	*n.a.*
	VGCN-BNN	77.01±0.16	64.74±0.07	62.45±0.37	*n.a.*	18.79±0.31	34.85±0.50	67.43±0.71	*n.a.*
	GKDE-GCN	76.40±0.33	61.74±0.05	63.15±0.10	*n.a.*	16.86±0.35	40.03±0.46	1.42±0.15	*n.a.*
	GPN	**80.98**±0.22	57.99±0.12	55.23±0.16	**89.47**±0.86	**81.53**±0.23	55.96±0.08	56.51±0.21	**100.00**±0.00
CiteSeer	APPNP	**73.14**±0.12	65.43±0.02	*n.a.*	*n.a.*	20.13±0.22	4.78±0.11	*n.a.*	*n.a.*
	VGCN	71.30±0.13	65.27±0.03	*n.a.*	*n.a.*	17.55±0.36	5.48±0.11	*n.a.*	*n.a.*
	RGCN	72.29±0.09	71.99±0.04	*n.a.*	*n.a.*	28.15±0.40	23.28±0.41	*n.a.*	*n.a.*
	VGCN-Dropout	69.80±0.19	63.47±0.09	51.82±0.12	*n.a.*	19.60±0.28	31.79±0.27	72.62±0.34	*n.a.*
	DropEdge	72.00±0.23	65.00±0.05	54.71±0.16	*n.a.*	18.00±0.47	17.80±0.25	44.78±0.52	*n.a.*
	VGCN-Energy	71.30±0.13	65.27±0.03	68.16±0.06	*n.a.*	17.55±0.36	5.48±0.11	0.03±0.01	*n.a.*
	VGCN-Ensemble	72.00±0.00	65.20±0.00	51.81±0.01	*n.a.*	18.00±0.00	21.22±0.01	52.80±0.02	*n.a.*
	VGCN-BNN	70.38±0.15	65.52±0.09	49.33±0.70	*n.a.*	16.27±0.33	23.24±0.86	60.07±1.69	*n.a.*
	GKDE-GCN	72.75±0.18	66.70±0.09	67.29±0.06	*n.a.*	18.79±0.33	35.46±0.71	0.21±0.04	*n.a.*
	GPN	65.00±0.43	59.47±0.16	55.95±0.18	**80.06**±1.18	**66.70**±0.23	51.65±0.16	65.58±0.26	**100.00**±0.00
PubMed	APPNP	82.80±0.10	62.22±0.02	*n.a.*	*n.a.*	40.38±0.22	5.41±0.22	*n.a.*	*n.a.*
	VGCN	82.49±0.10	62.16±0.06	*n.a.*	*n.a.*	37.80±0.40	6.54±0.23	*n.a.*	*n.a.*
	RGCN	**83.75**±0.12	64.87±0.05	*n.a.*	*n.a.*	47.82±0.36	29.60±0.34	*n.a.*	*n.a.*
	VGCN-Dropout	82.26±0.06	60.39±0.10	51.80±0.14	*n.a.*	37.79±0.45	23.86±0.35	38.16±0.53	*n.a.*
	DropEdge	82.70±0.12	62.21±0.10	55.48±0.18	*n.a.*	36.36±0.47	13.32±0.35	21.68±0.53	*n.a.*
	VGCN-Energy	82.49±0.10	62.16±0.06	65.38±0.07	*n.a.*	37.80±0.40	6.54±0.23	2.97±0.10	*n.a.*
	VGCN-Ensemble	82.00±0.00	62.42±0.00	60.34±0.10	*n.a.*	39.10±0.10	11.74±0.03	18.79±0.04	*n.a.*
	VGCN-BNN	82.30±0.14	62.36±0.15	59.35±0.99	*n.a.*	37.56±0.54	12.74±0.34	27.56±0.64	*n.a.*
	GKDE-GCN	82.54±0.11	60.62±0.11	63.00±0.17	*n.a.*	37.77±0.48	24.07±0.43	3.43±0.13	*n.a.*
	GPN	81.54±0.39	57.05±0.08	58.87±0.14	**84.07**±0.55	**81.73**±0.34	53.43±0.04	60.94±0.13	**100.00**±0.00
Amazon Computers	APPNP	75.00±0.09	67.83±0.02	*n.a.*	*n.a.*	18.25±0.46	5.94±0.11	*n.a.*	*n.a.*
	VGCN	81.54±0.08	58.03±0.02	*n.a.*	*n.a.*	20.38±0.29	24.56±0.33	*n.a.*	*n.a.*
	RGCN	61.39±0.39	57.92±0.04	*n.a.*	*n.a.*	39.60±0.45	33.60±0.35	*n.a.*	*n.a.*
	VGCN-Dropout	**81.79**±0.10	57.15±0.04	55.52±0.08	*n.a.*	21.52±0.36	40.32±0.29	66.21±0.33	*n.a.*
	DropEdge	81.20±0.08	57.88±0.02	55.51±0.07	*n.a.*	21.75±0.36	33.10±0.26	57.47±0.37	*n.a.*
	VGCN-Energy	81.54±0.08	58.03±0.02	58.66±0.03	*n.a.*	20.38±0.29	24.56±0.33	4.82±0.09	*n.a.*
	VGCN-Ensemble	81.00±0.00	58.22±0.01	53.24±0.12	*n.a.*	23.60±0.16	28.97±0.05	66.42±0.19	*n.a.*
	VGCN-BNN	61.25±0.17	56.01±0.11	51.16±0.83	*n.a.*	28.54±0.47	21.72±0.19	56.41±0.60	*n.a.*
	GKDE-GCN	59.83±0.73	56.38±0.12	55.91±0.05	*n.a.*	28.46±0.36	26.48±0.45	16.10±0.53	*n.a.*
	GPN	79.70±0.46	61.21±0.11	61.07±0.11	**86.15**±0.28	**79.87**±0.46	60.42±0.12	61.56±0.12	**100.00**±0.00
Amazon Photos	APPNP	**88.12**±0.10	65.02±0.03	*n.a.*	*n.a.*	19.37±0.45	8.42±0.29	*n.a.*	*n.a.*
	VGCN	83.91±0.08	57.91±0.02	*n.a.*	*n.a.*	21.40±0.49	31.07±0.34	*n.a.*	*n.a.*
	RGCN	79.50±0.72	57.22±0.04	*n.a.*	*n.a.*	42.38±0.40	32.02±0.31	*n.a.*	*n.a.*
	VGCN-Dropout	83.86±0.18	56.85±0.04	55.04±0.08	*n.a.*	22.29±0.55	49.11±0.31	66.74±0.35	*n.a.*
	DropEdge	85.69±0.15	57.32±0.04	55.31±0.07	*n.a.*	22.90±0.43	39.14±0.20	56.18±0.21	*n.a.*
	VGCN-Energy	83.91±0.08	57.91±0.02	59.07±0.02	*n.a.*	21.40±0.49	31.07±0.34	6.42±0.07	*n.a.*
	VGCN-Ensemble	84.40±0.16	57.86±0.01	56.01±0.19	*n.a.*	20.30±0.21	44.14±0.05	69.01±0.14	*n.a.*
	VGCN-BNN	82.00±0.19	56.78±0.09	49.21±0.58	*n.a.*	25.84±0.46	23.16±0.37	59.31±0.73	*n.a.*
	GKDE-GCN	73.17±0.94	57.01±0.10	58.00±0.05	*n.a.*	24.04±0.42	24.45±0.62	9.82±0.36	*n.a.*
	GPN	87.47±0.20	56.25±0.16	60.52±0.18	**75.24**±0.63	**88.29**±0.20	51.89±0.09	61.89±0.18	**100.00**±0.00
Coauthor CS	APPNP	89.28±0.07	72.01±0.02	*n.a.*	*n.a.*	12.58±0.33	23.09±0.31	*n.a.*	*n.a.*
	VGCN	89.33±0.05	67.65±0.02	*n.a.*	*n.a.*	13.29±0.22	30.13±0.32	*n.a.*	*n.a.*
	RGCN	**90.50**±0.06	71.13±0.02	*n.a.*	*n.a.*	41.67±0.35	52.81±0.21	*n.a.*	*n.a.*
	VGCN-Dropout	88.96±0.09	65.91±0.05	60.56±0.07	*n.a.*	13.31±0.32	67.56±0.26	85.81±0.22	*n.a.*
	DropEdge	89.44±0.07	67.94±0.01	63.68±0.06	*n.a.*	11.65±0.28	49.77±0.33	70.31±0.49	*n.a.*
	VGCN-Energy	89.33±0.05	67.65±0.02	70.14±0.02	*n.a.*	13.29±0.22	30.13±0.32	0.89±0.06	*n.a.*
	VGCN-Ensemble	89.00±0.00	67.64±0.01	64.41±0.08	*n.a.*	11.00±0.00	60.89±0.11	85.09±0.32	*n.a.*
	VGCN-BNN	88.16±0.10	67.09±0.12	59.69±0.35	*n.a.*	12.48±0.25	67.00±0.54	87.27±0.47	*n.a.*
	GKDE-GCN	88.14±0.14	67.69±0.07	70.08±0.14	*n.a.*	9.71±0.29	32.67±0.51	0.23±0.03	*n.a.*
	GPN	83.99±0.31	57.66±0.10	62.08±0.18	**97.84**±0.23	**83.96**±0.31	57.04±0.09	62.39±0.18	**100.00**±0.00
Coauthor Physics	APPNP	96.16±0.08	67.63±0.02	*n.a.*	*n.a.*	28.71±0.40	24.97±0.20	*n.a.*	*n.a.*
	VGCN	96.00±0.00	60.30±0.02	*n.a.*	*n.a.*	33.26±0.67	40.19±0.42	*n.a.*	*n.a.*
	RGCN	94.69±0.07	65.84±0.03	*n.a.*	*n.a.*	58.56±0.36	52.91±0.32	*n.a.*	*n.a.*
	VGCN-Dropout	95.90±0.05	58.97±0.03	57.64±0.05	*n.a.*	32.52±0.60	55.07±0.48	61.85±0.50	*n.a.*
	DropEdge	95.90±0.03	60.40±0.04	59.09±0.05	*n.a.*	30.53±0.58	43.30±0.43	51.07±0.44	*n.a.*
	VGCN-Energy	96.00±0.00	60.30±0.02	61.59±0.02	*n.a.*	33.26±0.67	40.19±0.42	11.45±0.22	*n.a.*
	VGCN-Ensemble	96.00±0.00	60.29±0.00	59.05±0.01	*n.a.*	31.70±0.21	52.08±0.10	68.48±0.10	*n.a.*
	VGCN-BNN	95.65±0.07	60.99±0.19	56.95±0.35	*n.a.*	32.95±0.60	62.53±0.75	71.96±0.73	*n.a.*
	GKDE-GCN	**96.61**±0.05	60.46±0.01	60.99±0.07	*n.a.*	28.84±0.31	29.12±0.33	2.46±0.10	*n.a.*
	GPN	92.70±0.11	59.92±0.08	58.62±0.18	**99.15**±0.05	**92.70**±0.11	58.66±0.08	59.00±0.18	**100.00**±0.00
OGBN Arxiv	APPNP	63.50±0.95	62.51±0.51	*n.a.*	*n.a.*	51.10±1.12	59.92±0.64	*n.a.*	*n.a.*
	VGCN	65.70±0.47	46.16±0.13	*n.a.*	*n.a.*	51.30±0.75	53.83±0.59	*n.a.*	*n.a.*
	VGCN-Dropout	65.30±0.70	48.11±0.23	50.64±0.20	*n.a.*	49.90±0.77	60.10±0.68	62.87±0.29	*n.a.*
	VGCN-Energy	65.70±0.47	46.16±0.13	48.54±0.20	*n.a.*	51.30±0.75	53.83±0.59	48.53±0.48	*n.a.*
	VGCN-Ensemble	**67.00**	45.99	47.41	*n.a.*	49.00	59.94	66.44	*n.a.*
	GKDE-GCN	65.20±0.49	50.98±0.23	51.31±0.22	*n.a.*	45.40±0.62	53.94±1.41	55.28±1.69	*n.a.*
	GPN	65.50±0.70	51.49±0.37	55.82±0.30	**93.05**±3.44	**65.50**±0.70	51.43±0.32	55.85±0.30	**95.54**±0.89

Table E.10: Accuracy and OOD detection scores on Bernoulli and unit Gaussian feature perturbations using AUC-ROC. OOD-AUC-ROC scores are given as *[Alea w/ Net] / [Epist w/ Net] / [Epist w/o Net]*. *n.a.* means either model or metric not applicable. Bold numbers indicate best results for Accuracy and OOD detection.

		$\boldsymbol{x}^{(v)} \sim \text{Ber}(0.5)$				$\boldsymbol{x}^{(v)} \sim \mathcal{N}(0,1)$			
	Model	**OOD-ACC**	**OOD-AUC-PR**			**OOD-ACC**	**OOD-AUC-PR**		
			Alea w/ Net	*Epist w/ Net*	*Epist w/o Net*		*Alea w/ Net*	*Epist w/ Net*	*Epist w/o Net*
CoraML	APPNP	80.85±0.09	11.86±0.06	*n.a.*	*n.a.*	17.99±0.36	2.55±0.00	*n.a.*	*n.a.*
	VGCN	78.90±0.09	11.92±0.07	*n.a.*	*n.a.*	18.37±0.31	2.57±0.00	*n.a.*	*n.a.*
	VGCN-Dropout	77.76±0.15	12.51±0.21	4.58±0.01	*n.a.*	18.28±0.35	3.91±0.02	62.19±0.39	*n.a.*
	DropEdge	77.40±0.14	10.67±0.09	4.90±0.01	*n.a.*	16.60±0.26	2.93±0.01	42.96±0.39	*n.a.*
	VGCN-Energy	78.90±0.09	11.92±0.07	10.43±0.08	*n.a.*	18.37±0.31	2.57±0.00	2.50±0.00	*n.a.*
	VGCN-Ensemble	78.00±0.00	11.78±0.00	5.38±0.00	*n.a.*	21.00±0.00	3.43±0.00	60.47±0.09	*n.a.*
	VGCN-BNN	77.01±0.16	10.94±0.09	7.75±0.15	*n.a.*	18.79±0.31	3.49±0.03	62.75±0.65	*n.a.*
	RGCN	79.78±0.16	16.43±0.14	*n.a.*	*n.a.*	33.37±0.35	4.82±0.14	*n.a.*	*n.a.*
	GKDE-GCN	76.40±0.33	9.55±0.10	9.79±0.09	*n.a.*	16.86±0.35	34.20±0.53	2.64±0.06	*n.a.*
	GPN	**80.98**±0.22	6.96±0.07	5.63±0.03	**25.80**±1.43	**81.53**±0.23	6.63±0.07	6.21±0.08	**100.00**±0.00
CiteSeer	APPNP	**73.14**±0.12	7.49±0.01	*n.a.*	*n.a.*	20.13±0.22	2.27±0.00	*n.a.*	*n.a.*
	VGCN	71.30±0.13	8.92±0.03	*n.a.*	*n.a.*	17.55±0.36	2.28±0.00	*n.a.*	*n.a.*
	RGCN	72.29±0.09	14.49±0.14	*n.a.*	*n.a.*	28.15±0.40	3.26±0.07	*n.a.*	*n.a.*
	VGCN-Dropout	69.80±0.19	8.07±0.04	4.26±0.01	*n.a.*	19.60±0.28	2.97±0.01	64.29±0.37	*n.a.*
	DropEdge	72.00±0.23	9.34±0.10	4.60±0.02	*n.a.*	18.00±0.47	2.48±0.01	42.23±0.50	*n.a.*
	VGCN-Energy	71.30±0.13	8.92±0.03	9.09±0.05	*n.a.*	17.55±0.36	2.28±0.00	2.26±0.00	*n.a.*
	VGCN-Ensemble	72.00±0.00	8.76±0.00	4.34±0.00	*n.a.*	18.00±0.00	2.58±0.00	49.06±0.01	*n.a.*
	VGCN-BNN	70.38±0.15	8.86±0.07	5.14±0.16	*n.a.*	16.27±0.33	2.68±0.02	57.70±1.51	*n.a.*
	GKDE-GCN	72.75±0.18	8.57±0.06	9.82±0.11	*n.a.*	18.79±0.33	33.75±0.72	2.27±0.01	*n.a.*
	GPN	65.00±0.43	6.27±0.09	5.37±0.06	**14.25**±1.03	**66.70**±0.23	4.86±0.03	29.98±0.62	**100.00**±0.00
PubMed	APPNP	82.80±0.10	1.08±0.00	*n.a.*	*n.a.*	40.38±0.22	0.40±0.00	*n.a.*	*n.a.*
	VGCN	82.49±0.10	1.29±0.01	*n.a.*	*n.a.*	37.80±0.40	0.40±0.00	*n.a.*	*n.a.*
	RGCN	**83.75**±0.12	1.42±0.03	*n.a.*	*n.a.*	47.82±0.36	0.64±0.02	*n.a.*	*n.a.*
	VGCN-Dropout	82.26±0.06	1.41±0.03	0.77±0.00	*n.a.*	37.79±0.45	0.48±0.00	25.20±0.50	*n.a.*
	DropEdge	82.70±0.12	1.25±0.01	0.93±0.01	*n.a.*	36.36±0.47	0.42±0.00	15.48±0.51	*n.a.*
	VGCN-Energy	82.49±0.10	1.29±0.01	1.50±0.01	*n.a.*	37.80±0.40	0.40±0.00	0.58±0.04	*n.a.*
	VGCN-Ensemble	82.00±0.00	1.46±0.01	1.05±0.00	*n.a.*	39.10±0.10	0.42±0.00	13.71±0.03	*n.a.*
	VGCN-BNN	82.30±0.14	1.28±0.01	1.77±0.16	*n.a.*	37.56±0.54	0.42±0.00	15.60±0.60	*n.a.*
	GKDE-GCN	82.54±0.11	1.31±0.03	1.31±0.01	*n.a.*	37.77±0.48	16.95±0.49	0.92±0.08	*n.a.*
	GPN	81.54±0.39	0.85±0.00	1.01±0.01	**3.31**±0.24	**81.73**±0.34	0.78±0.00	1.27±0.01	**99.98**±0.00
Amazon Computers	APPNP	75.00±0.09	2.18±0.01	*n.a.*	*n.a.*	18.25±0.46	0.56±0.00	*n.a.*	*n.a.*
	VGCN	81.54±0.08	1.43±0.00	*n.a.*	*n.a.*	20.38±0.29	0.67±0.00	*n.a.*	*n.a.*
	RGCN	61.39±0.39	1.40±0.00	*n.a.*	*n.a.*	39.60±0.45	0.86±0.02	*n.a.*	*n.a.*
	VGCN-Dropout	**81.79**±0.10	1.38±0.00	1.24±0.00	*n.a.*	21.52±0.36	0.88±0.01	20.49±0.27	*n.a.*
	DropEdge	81.20±0.08	1.46±0.00	1.23±0.00	*n.a.*	21.75±0.36	0.77±0.00	16.09±0.22	*n.a.*
	VGCN-Energy	81.54±0.08	1.43±0.00	1.44±0.00	*n.a.*	20.38±0.29	0.67±0.00	0.56±0.00	*n.a.*
	VGCN-Ensemble	81.00±0.00	1.44±0.00	1.13±0.00	*n.a.*	23.60±0.16	0.71±0.00	51.48±0.16	*n.a.*
	VGCN-BNN	61.25±0.17	1.35±0.03	1.29±0.05	*n.a.*	28.54±0.47	0.64±0.00	21.82±0.68	*n.a.*
	GKDE-GCN	59.83±0.73	1.35±0.01	1.32±0.00	*n.a.*	28.46±0.36	2.96±0.14	1.86±0.16	*n.a.*
	GPN	79.70±0.46	1.55±0.01	1.55±0.01	**4.26**±0.08	**79.87**±0.46	2.56±0.01	2.77±0.03	**100.00**±0.00
Amazon Photos	APPNP	**88.12**±0.10	**5.44**±0.05	*n.a.*	*n.a.*	19.37±0.45	1.02±0.00	*n.a.*	*n.a.*
	VGCN	83.91±0.08	3.61±0.07	*n.a.*	*n.a.*	21.40±0.49	1.39±0.01	*n.a.*	*n.a.*
	RGCN	79.50±0.72	3.33±0.04	*n.a.*	*n.a.*	42.38±0.40	1.85±0.05	*n.a.*	*n.a.*
	VGCN-Dropout	83.86±0.18	3.12±0.04	2.19±0.01	*n.a.*	22.29±0.55	2.09±0.02	21.31±0.24	*n.a.*
	DropEdge	85.69±0.15	3.56±0.05	2.25±0.01	*n.a.*	22.90±0.43	1.63±0.01	16.54±0.20	*n.a.*
	VGCN-Energy	83.91±0.08	3.61±0.07	5.32±0.07	*n.a.*	21.40±0.49	1.39±0.01	1.01±0.00	*n.a.*
	VGCN-Ensemble	84.40±0.16	3.27±0.02	2.27±0.02	*n.a.*	20.30±0.21	1.87±0.00	59.99±0.19	*n.a.*
	VGCN-BNN	82.00±0.19	3.70±0.09	2.07±0.04	*n.a.*	25.84±0.46	1.18±0.01	28.71±0.92	*n.a.*
	GKDE-GCN	73.17±0.94	2.72±0.04	3.13±0.04	*n.a.*	24.04±0.42	4.88±0.19	1.17±0.03	*n.a.*
	GPN	87.47±0.20	2.38±0.01	2.81±0.02	4.66±0.18	**88.29**±0.20	2.10±0.01	3.32±0.04	**100.00**±0.00
Coauthor CS	APPNP	89.28±0.07	2.32±0.01	*n.a.*	*n.a.*	12.58±0.33	0.50±0.00	*n.a.*	*n.a.*
	VGCN	89.33±0.05	1.75±0.01	*n.a.*	*n.a.*	13.29±0.22	0.57±0.00	*n.a.*	*n.a.*
	RGCN	**90.50**±0.06	1.89±0.00	*n.a.*	*n.a.*	41.67±0.35	1.16±0.02	*n.a.*	*n.a.*
	VGCN-Dropout	88.96±0.09	1.62±0.02	1.05±0.00	*n.a.*	13.31±0.32	1.68±0.02	71.46±0.30	*n.a.*
	DropEdge	89.44±0.07	1.97±0.03	1.18±0.00	*n.a.*	11.65±0.28	0.90±0.01	55.73±0.45	*n.a.*
	VGCN-Energy	89.33±0.05	1.75±0.01	2.38±0.01	*n.a.*	13.29±0.22	0.57±0.00	0.42±0.00	*n.a.*
	VGCN-Ensemble	89.00±0.00	1.76±0.00	1.19±0.00	*n.a.*	11.00±0.00	1.39±0.01	73.57±0.28	*n.a.*
	VGCN-BNN	88.16±0.10	1.91±0.02	1.01±0.01	*n.a.*	12.48±0.25	1.53±0.03	73.43±0.70	*n.a.*
	GKDE-GCN	88.14±0.14	2.00±0.02	2.70±0.02	*n.a.*	9.71±0.29	15.14±0.55	0.42±0.00	*n.a.*
	GPN	83.99±0.31	1.04±0.01	1.22±0.01	**29.25**±1.92	**83.96**±0.31	1.02±0.01	1.26±0.01	**100.00**±0.00
Coauthor Physics	APPNP	96.16±0.08	0.83±0.00	*n.a.*	*n.a.*	28.71±0.40	0.30±0.00	*n.a.*	*n.a.*
	VGCN	96.00±0.00	0.62±0.00	*n.a.*	*n.a.*	33.26±0.67	0.44±0.01	*n.a.*	*n.a.*
	RGCN	94.69±0.07	0.80±0.00	*n.a.*	*n.a.*	58.56±0.36	0.75±0.02	*n.a.*	*n.a.*
	VGCN-Dropout	95.90±0.05	0.61±0.00	0.57±0.00	*n.a.*	32.52±0.60	0.75±0.01	19.76±0.41	*n.a.*
	DropEdge	95.90±0.03	0.62±0.00	0.63±0.01	*n.a.*	30.53±0.58	0.53±0.01	12.71±0.29	*n.a.*
	VGCN-Energy	96.00±0.00	0.62±0.00	0.72±0.00	*n.a.*	33.26±0.67	0.44±0.01	0.23±0.00	*n.a.*
	VGCN-Ensemble	96.00±0.00	0.62±0.00	0.57±0.00	*n.a.*	31.70±0.21	0.71±0.01	41.30±0.30	*n.a.*
	VGCN-BNN	95.65±0.07	0.69±0.01	0.57±0.01	*n.a.*	32.95±0.60	1.17±0.04	44.88±0.80	*n.a.*
	GKDE-GCN	**96.61**±0.05	0.62±0.00	0.74±0.00	*n.a.*	28.84±0.31	6.53±0.18	0.22±0.00	*n.a.*
	GPN	92.70±0.11	0.63±0.00	0.60±0.00	**23.91**±1.06	**92.70**±0.11	0.62±0.00	0.62±0.00	**100.00**±0.00
OGBN Arxiv	APPNP	63.50±0.95	0.79±0.16	*n.a.*	*n.a.*	51.10±1.12	0.42±0.10	*n.a.*	*n.a.*
	VGCN	65.70±0.47	0.20±0.00	*n.a.*	*n.a.*	51.30±0.75	0.30±0.02	*n.a.*	*n.a.*
	VGCN-Dropout	65.30±0.70	0.21±0.00	0.37±0.03	*n.a.*	49.90±0.77	0.47±0.05	17.68±0.48	*n.a.*
	VGCN-Energy	65.70±0.47	0.20±0.00	2.32±0.38	*n.a.*	51.30±0.75	0.30±0.02	0.24±0.01	*n.a.*
	VGCN-Ensemble	**67.00**	0.20	0.19	*n.a.*	49.00	0.46	18.22	*n.a.*
	GKDE-GCN	65.20±0.49	0.76±0.05	0.76±0.05	*n.a.*	45.40±0.62	4.99±0.97	5.04±0.97	*n.a.*
	GPN	65.50±0.70	0.23±0.01	0.26±0.01	**46.84**±5.20	**65.50**±0.70	0.23±0.01	0.26±0.01	**48.97**±1.51

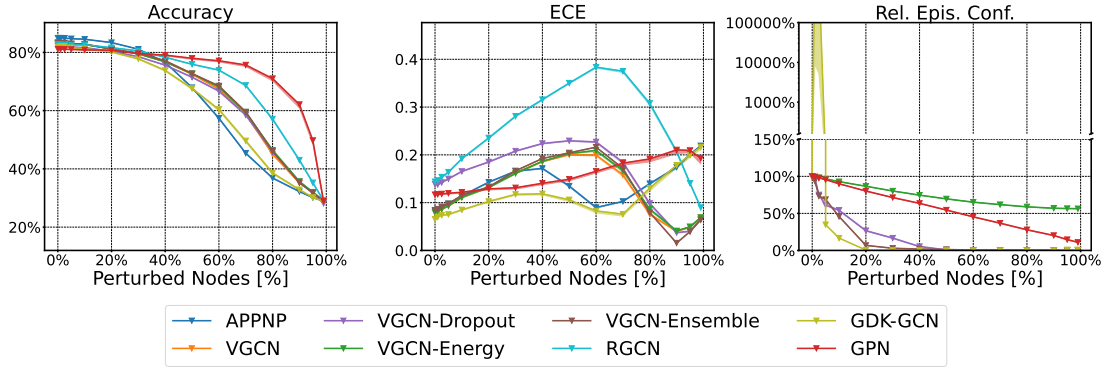
Table E.11: Accuracy and OOD detection scores on Bernoulli and unit Gaussian feature perturbations using AUC-APR. OOD-AUC-APR scores are given as *[Alea w/ Net] / [Epist w/ Net] / [Epist w/o Net]*. *n.a.* means either model or metric not applicable. Bold numbers indicate best results for Accuracy and OOD detection.

		Clean Graph		Leave-Out Classes						
	Model	**ID-ACC**	**ID-ECE**	**ID-ACC**	**OOD-AUC-ROC**			**OOD-AUC-PR**		
					Alea w/ Net	*Epist w/ Net*	*Epist w/o Net*	*Alea w/ Net*	*Epist w/ Net*	*Epist w/o Net*
CoraML	LP	78.41±0.00	64.12±0.00	86.40±0.00	83.78±0.00	80.86±0.00	*n.a.*	74.80±0.00	71.15±0.00	*n.a.*
	GKDE	72.88±0.00	56.46±0.00	83.02±0.00	74.46±0.00	71.86±0.00	*n.a.*	66.19±0.00	64.05±0.00	*n.a.*
	Matern-GGP	79.70±0.02	9.88±0.02	87.03±0.01	83.13±0.00	82.98±0.00	*n.a.*	71.42±0.00	71.04±0.14	*n.a.*
	GGP	79.04±0.01	21.67±0.01	88.65±0.00	81.49±0.00	82.03±0.00	*n.a.*	74.13±0.00	74.77±0.00	*n.a.*
	APPNP	**84.94**±0.02	8.27±0.10	**90.20**±0.02	83.71±0.06	*n.a.*	*n.a.*	78.77±0.07	*n.a.*	*n.a.*
	VGCN	83.14±0.03	7.96±0.16	89.66±0.05	81.70±0.07	*n.a.*	*n.a.*	75.67±0.10	*n.a.*	*n.a.*
	RGCN	82.79±0.06	14.39±0.17	88.66±0.03	80.37±0.11	*n.a.*	*n.a.*	76.97±0.11	*n.a.*	*n.a.*
	VGCN-Dropout	82.30±0.06	13.88±0.13	89.08±0.04	81.27±0.07	71.65±0.10	*n.a.*	75.55±0.12	60.65±0.11	*n.a.*
	DropEdge	83.07±0.04	13.93±0.11	89.03±0.03	83.55±0.05	75.48±0.12	*n.a.*	78.48±0.12	65.22±0.15	*n.a.*
	VGCN-Energy	83.14±0.03	7.96±0.16	89.66±0.05	81.70±0.07	83.15±0.07	*n.a.*	75.67±0.10	78.44±0.10	*n.a.*
	VGCN-Ensemble	83.41±0.01	8.45±0.01	89.87±0.00	81.85±0.00	74.24±0.00	*n.a.*	75.80±0.00	64.02±0.00	*n.a.*
	VGCN-BNN	82.83±0.06	15.66±0.13	88.49±0.04	82.18±0.23	73.18±1.10	*n.a.*	76.17±0.36	63.51±1.40	*n.a.*
	GKDE-GCN	81.91±0.19	**6.53**±0.14	89.33±0.04	82.23±0.08	82.09±0.18	*n.a.*	75.88±0.12	77.03±0.39	*n.a.*
	GPN	81.16±0.12	11.68±0.18	88.51±0.04	83.25±0.12	**86.28**±0.17	80.95±0.24	75.79±0.28	**79.97**±0.20	72.81±0.46
CiteSeer	LP	54.05±0.00	37.38±0.00	57.34±0.00	65.99±0.00	67.54±0.00	*n.a.*	48.12±0.00	48.59±0.00	*n.a.*
	GKDE	53.67±0.00	36.29±0.00	49.62±0.00	63.75±0.00	63.91±0.00	*n.a.*	56.74±0.00	56.79±0.00	*n.a.*
	Matern-GGP	53.25±0.03	12.76±0.03	53.83±0.06	59.57±0.00	59.56±0.05	*n.a.*	36.05±0.04	36.24±0.19	*n.a.*
	GGP	68.85±0.01	20.84±0.01	69.74±0.01	**74.56**±0.06	74.10±0.07	*n.a.*	50.39±0.08	48.74±0.08	*n.a.*
	APPNP	70.24±0.02	4.88±0.03	72.83±0.03	72.91±0.09	*n.a.*	*n.a.*	56.31±0.14	*n.a.*	*n.a.*
	VGCN	68.98±0.02	4.33±0.05	70.79±0.02	72.16±0.08	*n.a.*	*n.a.*	53.71±0.08	*n.a.*	*n.a.*
	RGCN	**70.57**±0.03	15.09±0.09	72.15±0.03	74.56±0.07	*n.a.*	*n.a.*	**58.63**±0.07	*n.a.*	*n.a.*
	VGCN-Dropout	68.41±0.03	7.31±0.07	70.44±0.06	71.31±0.08	60.05±0.12	*n.a.*	52.05±0.14	36.95±0.10	*n.a.*
	DropEdge	69.33±0.03	5.64±0.06	71.02±0.05	73.42±0.05	63.23±0.17	*n.a.*	55.70±0.10	39.38±0.14	*n.a.*
	VGCN-Energy	68.98±0.02	4.33±0.05	70.79±0.02	72.16±0.08	76.08±0.11	*n.a.*	53.71±0.08	58.35±0.17	*n.a.*
	VGCN-Ensemble	69.26±0.00	4.14±0.02	70.63±0.00	72.23±0.00	58.61±0.01	*n.a.*	54.04±0.00	38.93±0.01	*n.a.*
	VGCN-BNN	68.06±0.07	8.42±0.20	69.84±0.04	71.64±0.31	64.16±1.75	*n.a.*	52.60±0.47	46.72±1.76	*n.a.*
	GKDE-GCN	69.55±0.03	**3.88**±0.06	70.76±0.04	73.34±0.15	**76.19**±0.31	*n.a.*	54.25±0.16	59.07±0.42	*n.a.*
	GPN	66.13±0.17	7.42±0.22	69.79±0.10	72.46±0.27	70.74±0.26	66.65±0.29	55.14±0.46	50.52±0.34	44.93±0.31
PubMed	LP	78.40±0.00	45.07±0.00	89.18±0.00	**80.32**±0.00	79.64±0.00	*n.a.*	71.01±0.00	**72.98**±0.00	*n.a.*
	GKDE	77.10±0.01	40.02±0.01	88.16±0.00	69.66±0.00	68.47±0.00	*n.a.*	55.81±0.00	54.33±0.00	*n.a.*
	Matern-GGP	78.77±0.00	12.37±0.00	90.33±0.01	46.69±0.00	45.75±0.00	*n.a.*	39.85±0.00	39.63±0.00	*n.a.*
	GGP	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*
	APPNP	**86.88**±0.01	2.57±0.05	94.83±0.01	74.76±0.06	*n.a.*	*n.a.*	61.84±0.07	*n.a.*	*n.a.*
	VGCN	86.70±0.01	2.30±0.05	94.77±0.01	72.58±0.04	*n.a.*	*n.a.*	60.54±0.04	*n.a.*	*n.a.*
	RGCN	85.87±0.01	4.86±0.05	94.73±0.01	71.49±0.14	*n.a.*	*n.a.*	60.54±0.13	*n.a.*	*n.a.*
	VGCN-Dropout	86.49±0.01	5.53±0.06	94.72±0.01	71.10±0.04	67.27±0.06	*n.a.*	59.47±0.04	54.24±0.08	*n.a.*
	DropEdge	86.57±0.01	5.11±0.04	94.72±0.01	72.09±0.02	68.57±0.04	*n.a.*	59.84±0.03	54.95±0.06	*n.a.*
	VGCN-Energy	86.70±0.01	2.30±0.05	94.77±0.01	72.58±0.04	72.63±0.06	*n.a.*	60.54±0.04	60.63±0.10	*n.a.*
	VGCN-Ensemble	86.64±0.00	2.29±0.01	**94.88**±0.00	72.71±0.00	70.99±0.00	*n.a.*	60.47±0.00	59.31±0.00	*n.a.*
	VGCN-BNN	85.56±0.05	14.00±0.11	94.34±0.03	65.41±0.58	63.77±1.50	*n.a.*	53.23±0.40	54.36±1.54	*n.a.*
	GKDE-GCN	86.14±0.07	**1.36**±0.09	94.66±0.00	73.53±0.06	74.47±0.11	*n.a.*	61.36±0.04	61.96±0.27	*n.a.*
	GPN	84.10±0.26	4.31±0.09	94.08±0.02	71.84±0.08	73.91±0.20	71.20±0.15	57.92±0.10	67.19±0.25	59.72±0.18

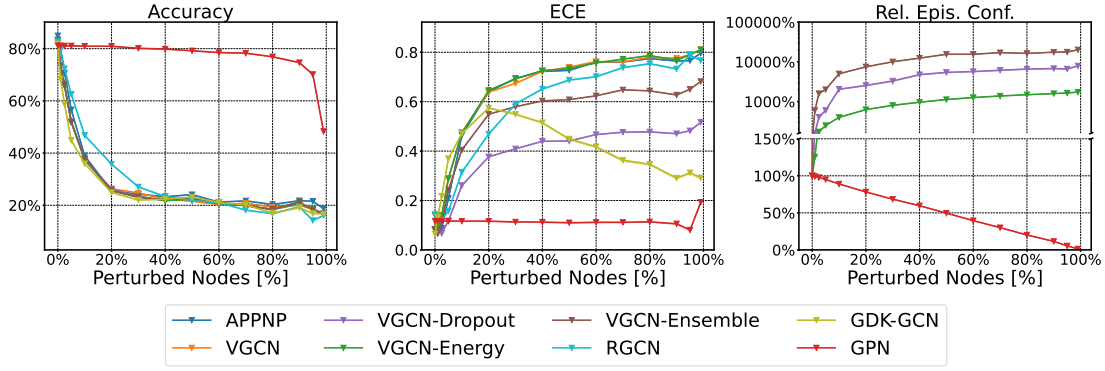
Table E.12: Accuracy and ECE scores on the clean graphs. Accuracy and OOD detection scores on Left-Out classes using AUC-ROC and AUC-PR scores. OOD-AUC-ROC and OOD-AUC-APR scores are given as *[Alea w/ Net] / [Epist w/ Net] / [Epist w/o Net]*. *n.a.* means model or metric not applicable and *n.f.* means not finished within our constraints. Bold numbers indicate best results for Accuracy, ECE and OOD detection.

		Clean Graph		Leave-Out Classes						
	Model	**ID-ACC**	**ID-ECE**	**ID-ACC**	**OOD-AUC-ROC**			**OOD-AUC-PR**		
					Alea w/ Net	*Epist w/ Net*	*Epist w/o Net*	*Alea w/ Net*	*Epist w/ Net*	*Epist w/o Net*
Amazon Computers	LP	79.49±0.00	69.49±0.00	83.28±0.00	86.74±0.00	83.88±0.00	*n.a.*	67.10±0.00	63.08±0.00	*n.a.*
	GKDE	63.49±0.00	38.93±0.00	71.41±0.00	75.14±0.00	73.58±0.00	*n.a.*	49.21±0.00	47.68±0.00	*n.a.*
	Matern-GGP	80.23±0.00	12.13±0.01	86.94±0.01	79.00±0.00	79.10±0.00	*n.a.*	49.57±0.00	49.91±0.04	*n.a.*
	GGP	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*
	APPNP	80.12±0.04	11.83±0.04	87.72±0.02	81.30±0.02	*n.a.*	*n.a.*	53.02±0.04	*n.a.*	*n.a.*
	VGCN	81.66±0.04	9.90±0.03	88.95±0.02	82.76±0.03	*n.a.*	*n.a.*	57.49±0.06	*n.a.*	*n.a.*
	RGCN	68.43±0.24	23.62±0.16	78.52±0.13	76.40±0.16	*n.a.*	*n.a.*	53.16±0.20	*n.a.*	*n.a.*
	VGCN-Dropout	81.29±0.05	11.52±0.03	88.54±0.02	81.99±0.04	72.90±0.04	*n.a.*	55.66±0.08	41.38±0.04	*n.a.*
	DropEdge	82.09±0.03	11.40±0.03	88.62±0.02	82.79±0.02	75.35±0.06	*n.a.*	56.77±0.02	43.94±0.07	*n.a.*
	VGCN-Energy	81.66±0.04	9.90±0.03	88.95±0.02	82.76±0.03	83.43±0.04	*n.a.*	57.49±0.06	60.64±0.10	*n.a.*
	VGCN-Ensemble	81.66±0.01	9.96±0.02	**89.00**±0.02	82.80±0.02	83.77±0.55	*n.a.*	57.46±0.04	57.62±0.84	*n.a.*
	VGCN-BNN	68.20±0.15	20.27±0.45	79.65±0.06	82.16±0.24	69.72±2.37	*n.a.*	58.10±0.53	52.08±2.48	*n.a.*
	GKDE-GCN	65.90±0.53	**9.04**±0.28	82.73±0.37	77.03±0.34	70.32±0.66	*n.a.*	49.81±0.44	45.92±0.90	*n.a.*
	GPN	**82.10**±0.29	9.22±0.19	88.48±0.07	82.49±0.17	**87.63**±0.18	74.55±0.24	56.78±0.38	**67.94**±0.28	48.03±0.34
Amazon Photos	LP	85.88±0.00	73.38±0.00	89.27±0.00	**94.24**±0.00	90.26±0.00	*n.a.*	**90.24**±0.00	85.55±0.00	*n.a.*
	GKDE	75.38±0.00	52.21±0.00	85.94±0.00	76.51±0.00	60.83±0.00	*n.a.*	66.72±0.00	59.09±0.00	*n.a.*
	Matern-GGP	86.10±0.01	**8.54**±0.02	88.65±0.00	87.26±0.00	86.75±0.00	*n.a.*	75.22±0.00	74.39±0.03	*n.a.*
	GGP	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*
	APPNP	**92.12**±0.01	13.68±0.02	**95.42**±0.01	77.45±0.10	*n.a.*	*n.a.*	67.50±0.12	*n.a.*	*n.a.*
	VGCN	90.95±0.01	10.37±0.03	94.24±0.01	82.44±0.07	*n.a.*	*n.a.*	72.60±0.11	*n.a.*	*n.a.*
	RGCN	81.00±0.42	40.20±0.25	87.59±0.69	75.25±0.23	*n.a.*	*n.a.*	67.53±0.29	*n.a.*	*n.a.*
	VGCN-Dropout	90.42±0.01	12.76±0.05	94.04±0.01	80.90±0.08	70.11±0.14	*n.a.*	70.55±0.12	53.16±0.13	*n.a.*
	DropEdge	91.03±0.01	12.25±0.02	94.22±0.01	82.48±0.09	71.67±0.11	*n.a.*	72.75±0.15	54.98±0.10	*n.a.*
	VGCN-Energy	90.95±0.01	10.37±0.03	94.24±0.01	82.44±0.07	79.64±0.07	*n.a.*	72.60±0.11	71.71±0.14	*n.a.*
	VGCN-Ensemble	90.94±0.01	10.38±0.02	94.28±0.01	82.72±0.03	88.53±0.39	*n.a.*	72.98±0.08	83.28±0.46	*n.a.*
	VGCN-BNN	84.51±0.23	29.42±0.32	91.88±0.19	72.03±0.38	64.10±1.98	*n.a.*	62.85±0.49	54.09±1.76	*n.a.*
	GKDE-GCN	79.74±0.99	12.84±0.31	89.84±0.73	73.65±1.13	69.09±0.81	*n.a.*	62.45±1.20	59.68±0.75	*n.a.*
	GPN	90.44±0.07	11.95±0.12	94.01±0.07	82.72±0.24	91.98±0.22	76.57±0.49	74.55±0.39	86.29±0.35	64.00±0.68
Coauthor CS	LP	83.34±0.00	76.67±0.00	82.89±0.00	86.64±0.00	86.40±0.00	*n.a.*	79.05±0.00	79.56±0.00	*n.a.*
	GKDE	79.27±0.00	70.74±0.00	78.84±0.00	79.32±0.00	77.59±0.00	*n.a.*	66.30±0.00	64.69±0.00	*n.a.*
	Matern-GGP	83.56±0.01	**6.18**±0.00	83.21±0.00	73.57±0.00	73.75±0.00	*n.a.*	62.05±0.00	61.74±0.00	*n.a.*
	GGP	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*
	APPNP	**92.96**±0.01	8.70±0.02	**93.51**±0.01	81.88±0.03	*n.a.*	*n.a.*	75.85±0.06	*n.a.*	*n.a.*
	VGCN	92.61±0.01	7.00±0.03	93.07±0.01	85.35±0.03	*n.a.*	*n.a.*	80.87±0.06	*n.a.*	*n.a.*
	RGCN	92.02±0.02	11.52±0.06	92.49±0.01	77.00±0.08	*n.a.*	*n.a.*	71.87±0.11	*n.a.*	*n.a.*
	VGCN-Dropout	92.30±0.01	12.41±0.05	92.83±0.01	84.04±0.03	73.09±0.10	*n.a.*	78.99±0.07	55.79±0.14	*n.a.*
	DropEdge	92.63±0.01	11.40±0.03	92.92±0.01	84.62±0.06	75.68±0.08	*n.a.*	79.74±0.08	58.85±0.11	*n.a.*
	VGCN-Energy	92.61±0.01	7.00±0.03	93.07±0.01	85.35±0.03	87.33±0.04	*n.a.*	80.87±0.06	82.79±0.11	*n.a.*
	VGCN-Ensemble	92.66±0.01	7.03±0.02	93.07±0.00	85.43±0.00	83.19±0.07	*n.a.*	80.88±0.01	72.27±0.14	*n.a.*
	VGCN-BNN	92.21±0.04	12.15±0.10	92.54±0.04	80.48±0.25	70.75±0.65	*n.a.*	73.64±0.35	54.41±0.76	*n.a.*
	GKDE-GCN	92.35±0.09	8.04±0.11	93.13±0.01	85.02±0.03	84.45±0.06	*n.a.*	80.15±0.07	77.90±0.12	*n.a.*
	GPN	86.88±0.10	18.92±0.11	88.21±0.10	69.49±0.39	**92.09**±0.20	88.84±0.31	55.41±0.46	**90.28**±0.18	86.54±0.42
Coauthor Physics	LP	90.75±0.00	70.75±0.00	95.39±0.00	91.78±0.00	90.03±0.00	*n.a.*	70.58±0.00	69.63±0.00	*n.a.*
	GKDE	87.75±0.00	56.04±0.00	93.30±0.00	87.02±0.00	84.64±0.00	*n.a.*	57.00±0.00	52.49±0.00	*n.a.*
	Matern-GGP	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*
	GGP	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*
	APPNP	95.56±0.00	1.57±0.01	**97.96**±0.00	90.37±0.02	*n.a.*	*n.a.*	61.46±0.05	*n.a.*	*n.a.*
	VGCN	95.59±0.00	**1.28**±0.02	**97.96**±0.00	90.29±0.02	*n.a.*	*n.a.*	63.63±0.09	*n.a.*	*n.a.*
	RGCN	95.33±0.00	5.11±0.07	97.92±0.00	75.62±0.13	*n.a.*	*n.a.*	56.27±0.13	*n.a.*	*n.a.*
	VGCN-Dropout	95.51±0.00	3.07±0.02	97.92±0.00	89.63±0.03	87.10±0.04	*n.a.*	62.53±0.10	51.19±0.12	*n.a.*
	DropEdge	95.59±0.00	2.81±0.01	97.92±0.00	90.56±0.02	88.76±0.04	*n.a.*	64.59±0.08	55.32±0.14	*n.a.*
	VGCN-Energy	95.59±0.00	1.28±0.02	97.96±0.00	90.29±0.02	91.08±0.05	*n.a.*	63.63±0.09	69.41±0.11	*n.a.*
	VGCN-Ensemble	95.58±0.00	1.29±0.01	97.96±0.00	90.35±0.00	92.39±0.00	*n.a.*	63.67±0.00	71.30±0.02	*n.a.*
	VGCN-BNN	95.46±0.02	5.64±0.09	97.94±0.01	90.73±0.21	90.09±0.50	*n.a.*	66.98±0.32	61.27±1.47	*n.a.*
	GKDE-GCN	**95.61**±0.00	1.51±0.02	97.95±0.00	87.38±0.09	84.62±0.19	*n.a.*	57.97±0.22	56.30±0.51	*n.a.*
	GPN(16)	94.32±0.02	10.61±0.03	97.40±0.01	85.20±0.17	**94.51**±0.15	89.63±0.24	61.89±0.20	**83.73**±0.31	66.44±0.65
OGBN Arxiv	LP	64.27	61.77	66.84	**80.04**	75.22	*n.a.*	65.21	**67.69**	*n.a.*
	GKDE	48.87	22.59	51.51	68.12	65.80	*n.a.*	47.22	45.23	*n.a.*
	Matern-GGP	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*
	GGP	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*	*n.f.*
	APPNP	71.46±0.09	3.96±0.06	75.47±0.15	65.21±0.14	*n.a.*	*n.a.*	43.23±0.06	*n.a.*	*n.a.*
	VGCN	71.89±0.08	2.18±0.10	75.61±0.11	64.91±0.28	*n.a.*	*n.a.*	42.72±0.32	*n.a.*	*n.a.*
	VGCN-Dropout	71.76±0.07	2.29±0.08	75.47±0.12	65.35±0.27	64.24±0.26	*n.a.*	43.09±0.30	41.58±0.23	*n.a.*
	VGCN-Energy	71.89±0.08	2.18±0.10	75.61±0.11	64.91±0.28	64.50±0.38	*n.a.*	42.72±0.32	42.41±0.39	*n.a.*
	VGCN-Ensemble	**72.59**	**2.10**	**76.12**	65.93	70.77	*n.a.*	43.84	50.63	*n.a.*
	VGCN-BNN	*n.a.*	*n.a.*	*n.a.*	*n.a.*	*n.a.*	*n.a.*	*n.a.*	*n.a.*	*n.a.*
	GKDE-GCN	68.99±0.40	8.43±0.43	73.89±0.33	68.84±0.18	72.44±0.50	*n.a.*	49.71±0.59	52.23±0.66	*n.a.*
	GPN	69.08±0.21	6.96±0.31	73.84±0.21	66.33±0.23	74.82±0.27	62.17±0.23	46.35±0.26	58.71±0.34	43.01±0.36

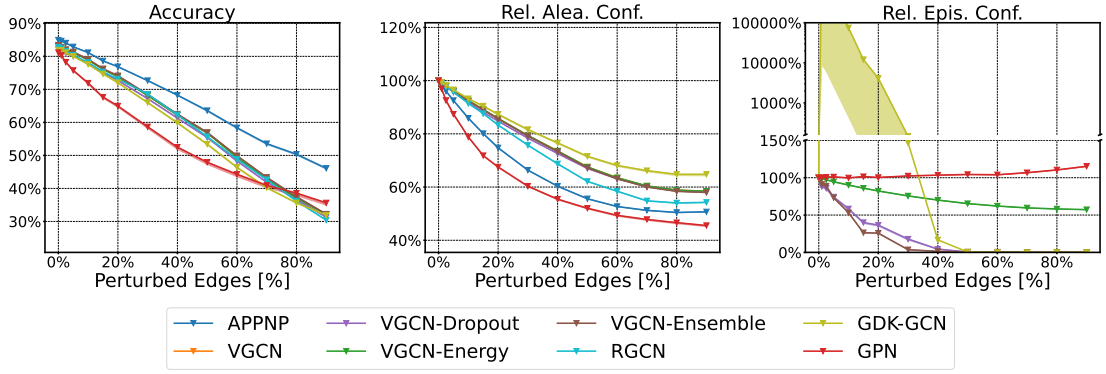
Table E.13: Accuracy and ECE scores on the clean graphs. Accuracy and OOD detection scores on Left-Out classes using AUC-ROC and AUC-PR scores. OOD-AUC-ROC and OOD-AUC-APR scores are given as *[Alea w/ Net] / [Epist w/ Net] / [Epist w/o Net]*. *n.a.* means either model or metric not applicable and *n.f.* means not finished within our constraints. Bold numbers indicate best results for Accuracy, ECE and OOD detection.



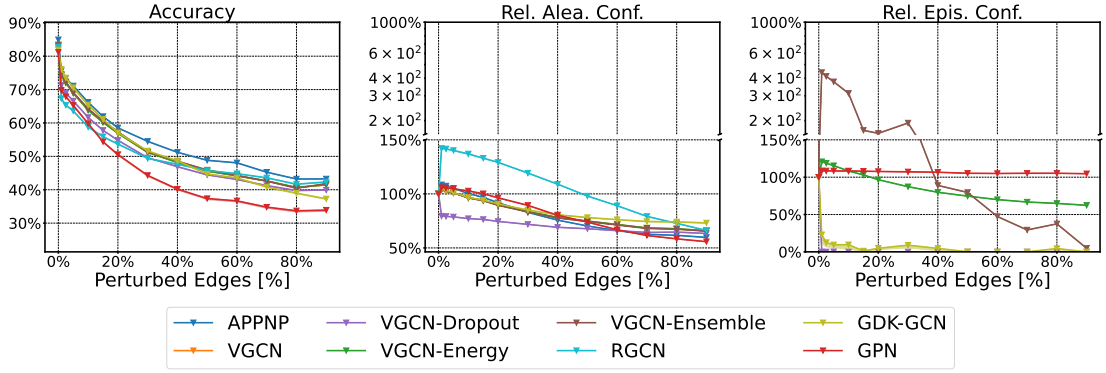
(a) Bernoulli Feature Shift



(b) Unit Gaussian Feature Shift

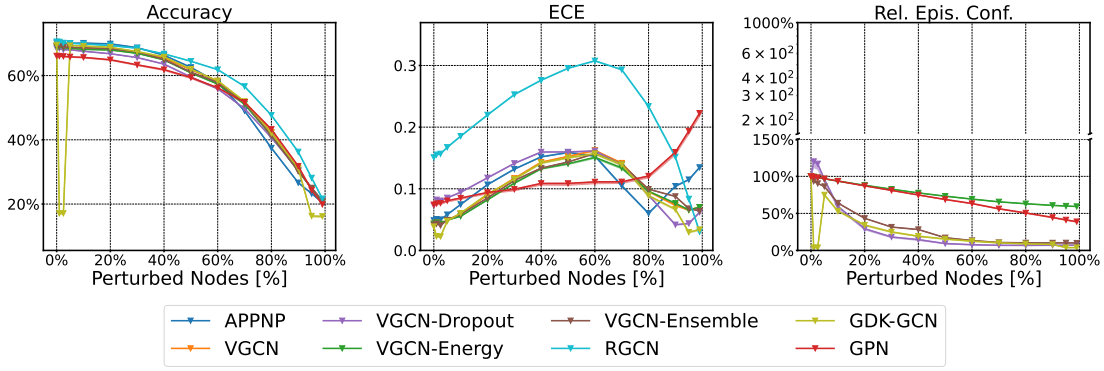


(c) Random Edge Shift

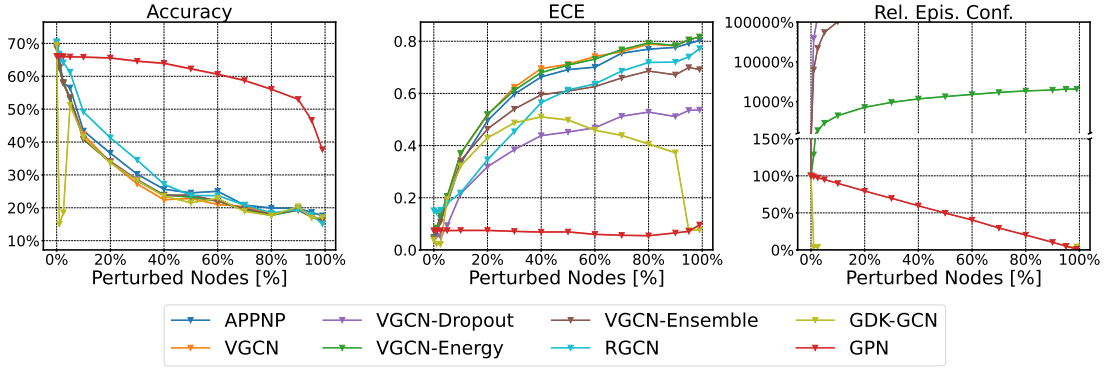


(d) DICE [431] Edge Shift

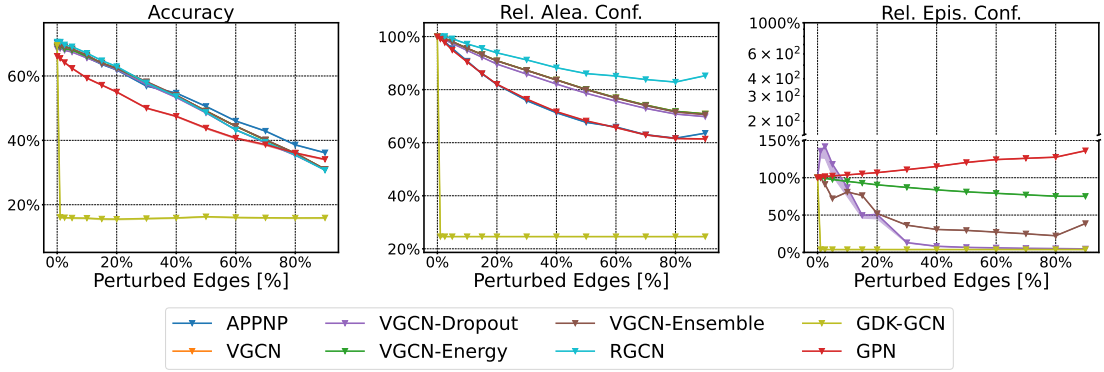
Figure E.2: Relative performance over different degrees of corruption of *CoraML*. For feature shifts, we perturb different fractions of nodes (whose features are replaced with either random vectors from a Bernoulli noise or a Unit Gaussian noise) and show accuracy, ECE, and relative average epistemic confidence. For edge shifts, we perturb different fractions of edges (by replacing them at random or using the global and untargeted DICE [431] attack) and show accuracy, relative average aleatoric confidence, and relative average epistemic confidence.



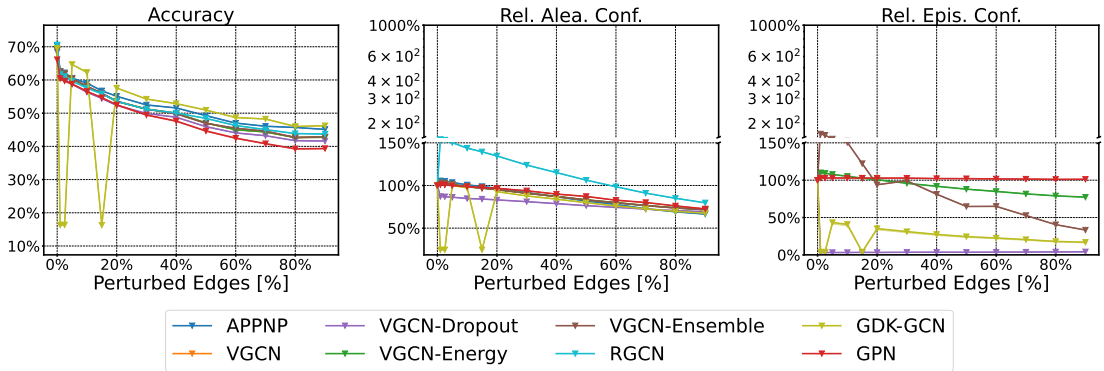
(a) Bernoulli Feature Shift



(b) Unit Gaussian Feature Shift

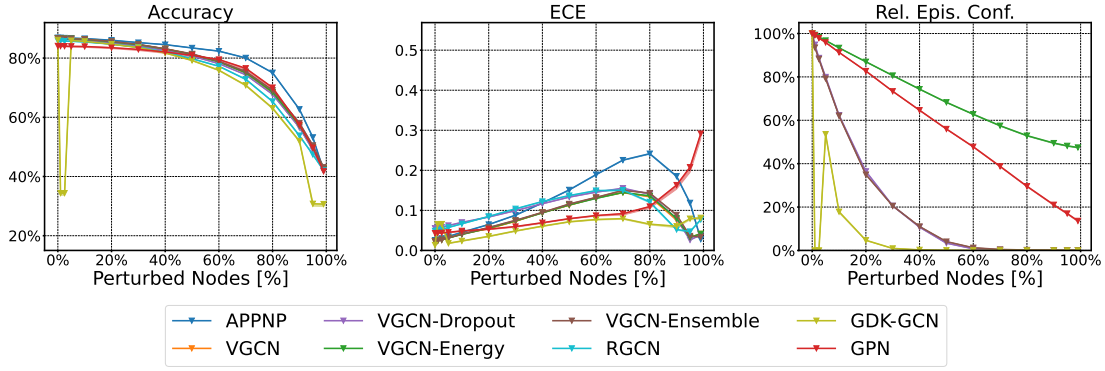


(c) Random Edge Shift

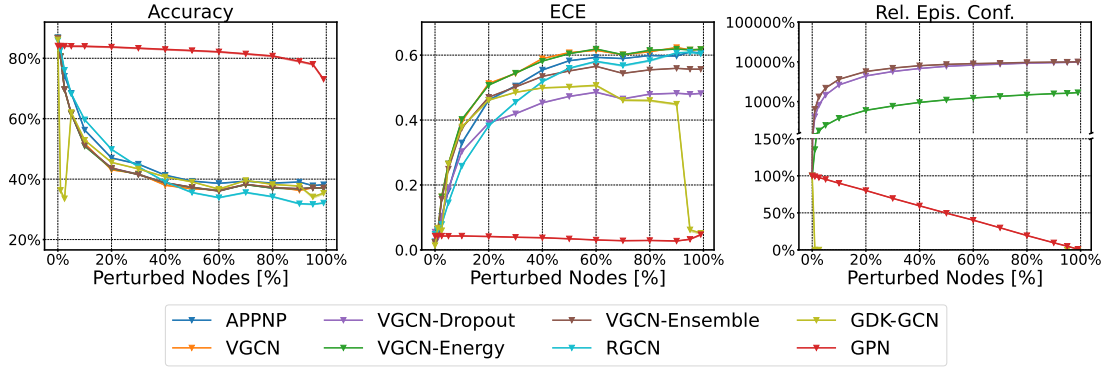


(d) DICE [431] Edge Shift

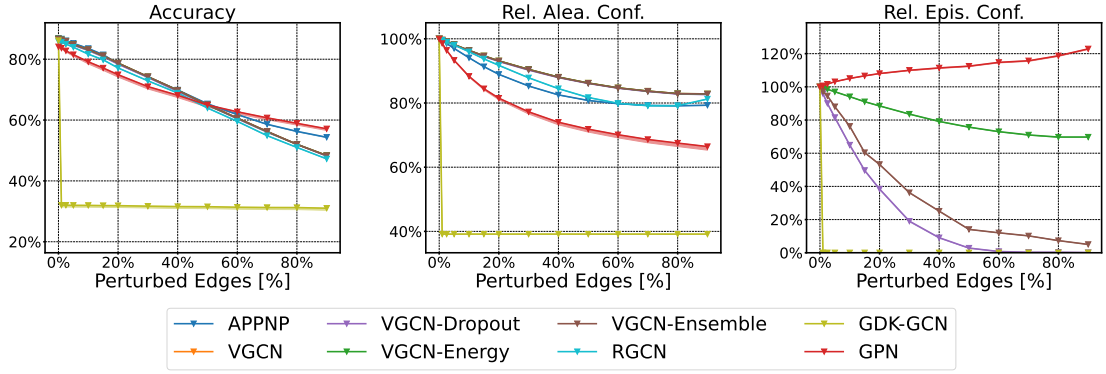
Figure E.3: Relative performance over different degrees of corruption of *CiteSeer*. For feature shifts, we perturb different fractions of nodes (whose features are replaced with either random vectors from a Bernoulli noise or a Unit Gaussian noise) and show accuracy, ECE, and relative average epistemic confidence. For edge shifts, we perturb different fractions of edges (by replacing them at random or using the global and untargeted DICE [431] attack) and show accuracy, relative average aleatoric confidence, and relative average epistemic confidence.



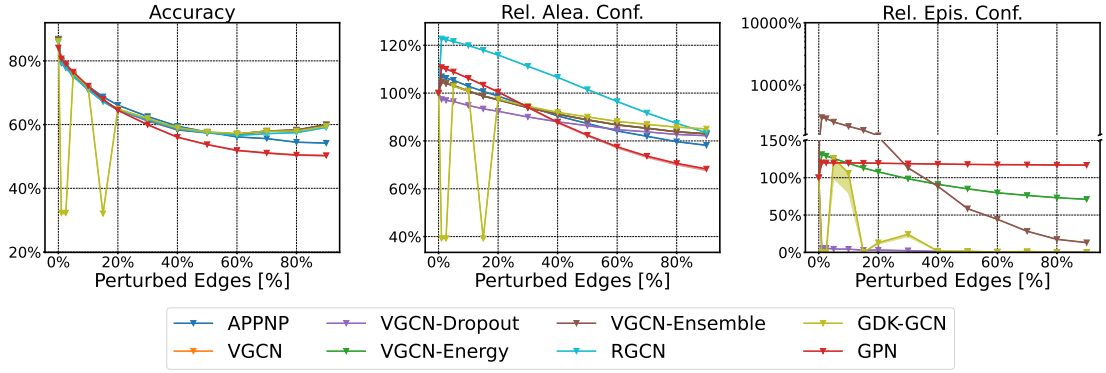
(a) Bernoulli Feature Shift



(b) Unit Gaussian Feature Shift

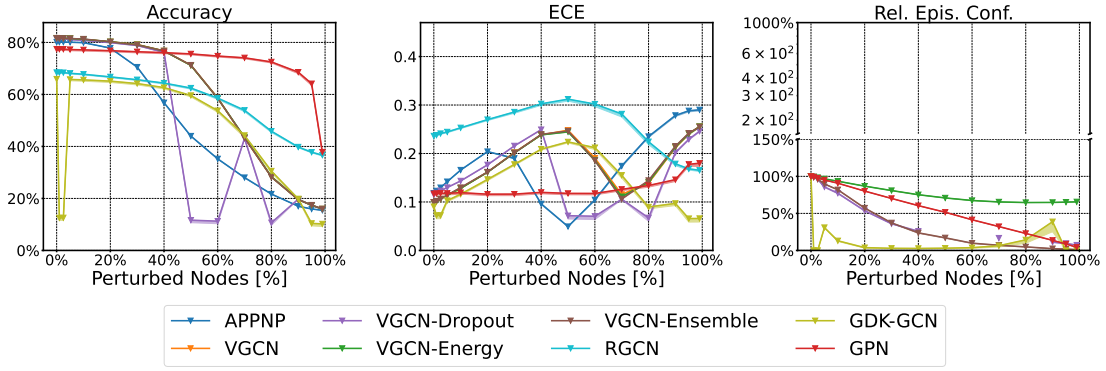


(c) Random Edge Shift

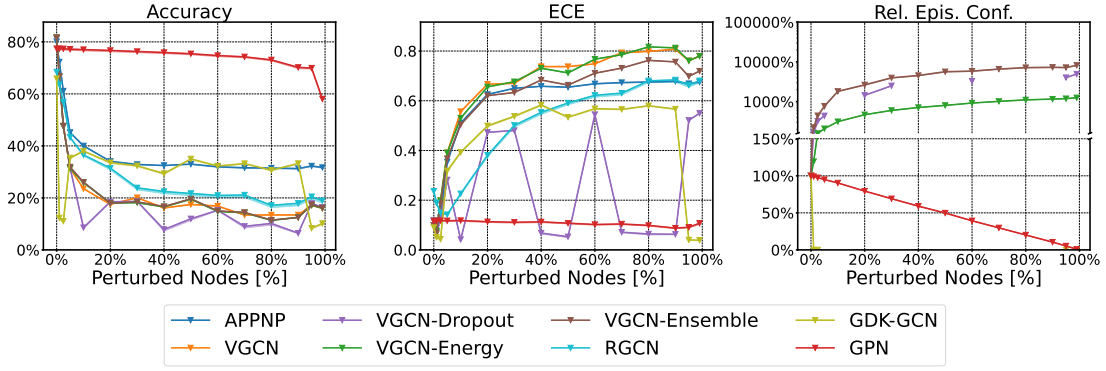


(d) DICE [431] Edge Shift

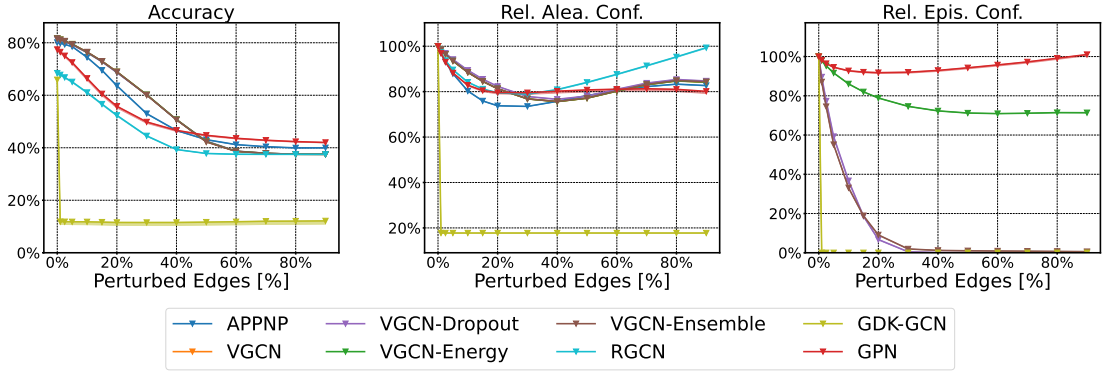
Figure E.4: Relative performance over different degrees of corruption of *PubMed*. For feature shifts, we perturb different fractions of nodes (whose features are replaced with either random vectors from a Bernoulli noise or a Unit Gaussian noise) and show accuracy, ECE, and relative average epistemic confidence. For edge shifts, we perturb different fractions of edges (by replacing them at random or using the global and untargeted DICE [431] attack) and show accuracy, relative average aleatoric confidence, and relative average epistemic confidence.



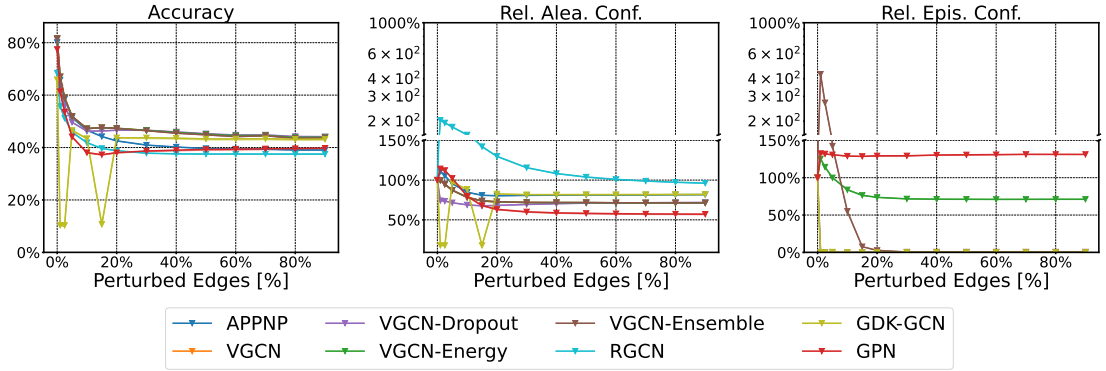
(a) Bernoulli Feature Shift



(b) Unit Gaussian Feature Shift

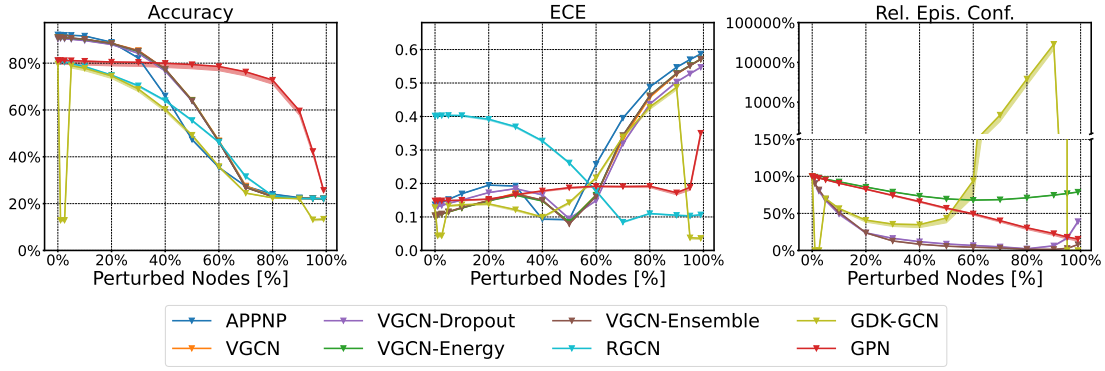


(c) Random Edge Shift

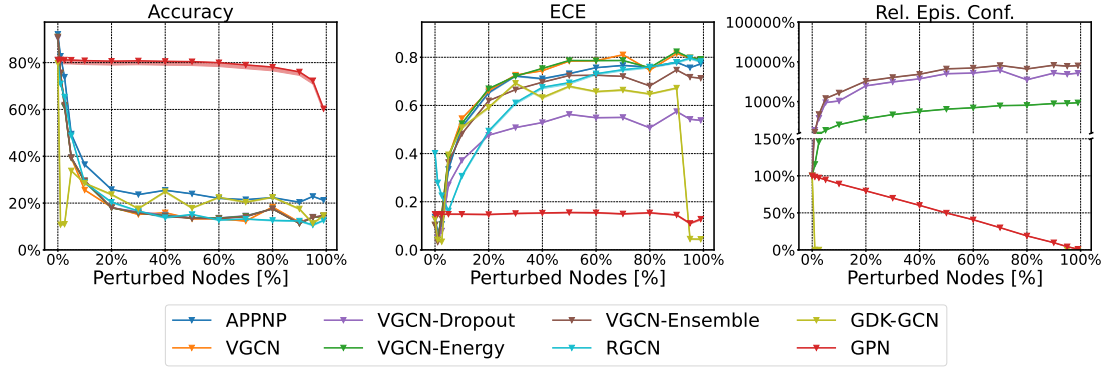


(d) DICE [431] Edge Shift

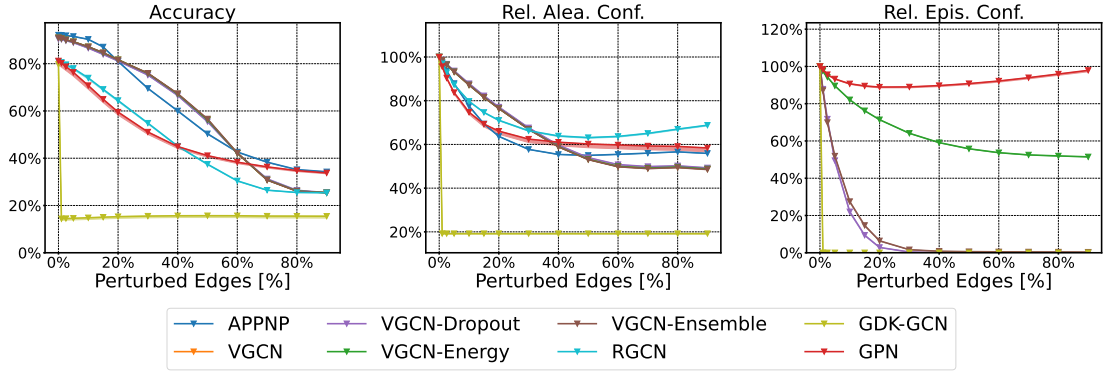
Figure E.5: Relative performance over different degrees of corruption of *Amazon Computers*. For feature shifts, we perturb different fractions of nodes (whose features are replaced with either random vectors from a Bernoulli noise or a Unit Gaussian noise) and show accuracy, ECE, and relative average epistemic confidence. For edge shifts, we perturb different fractions of edges (by replacing them at random or using the global and untargeted DICE [431] attack) and show accuracy, relative average aleatoric confidence, and relative average epistemic confidence.



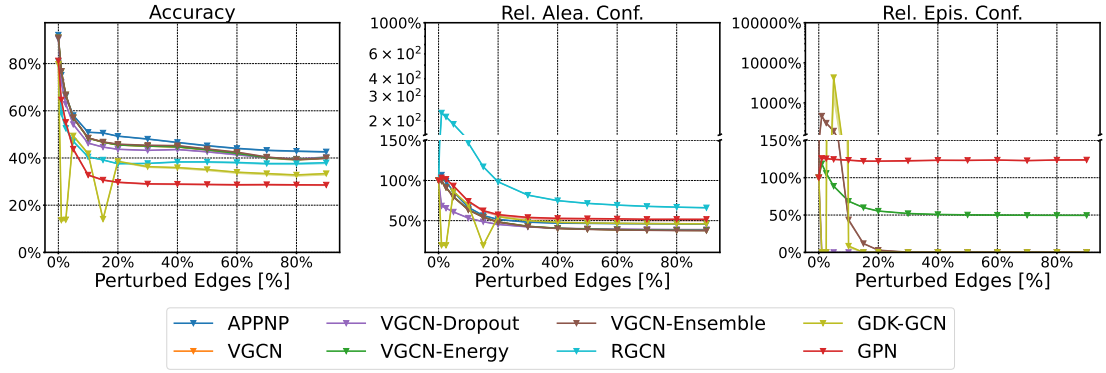
(a) Bernoulli Feature Shift



(b) Unit Gaussian Feature Shift

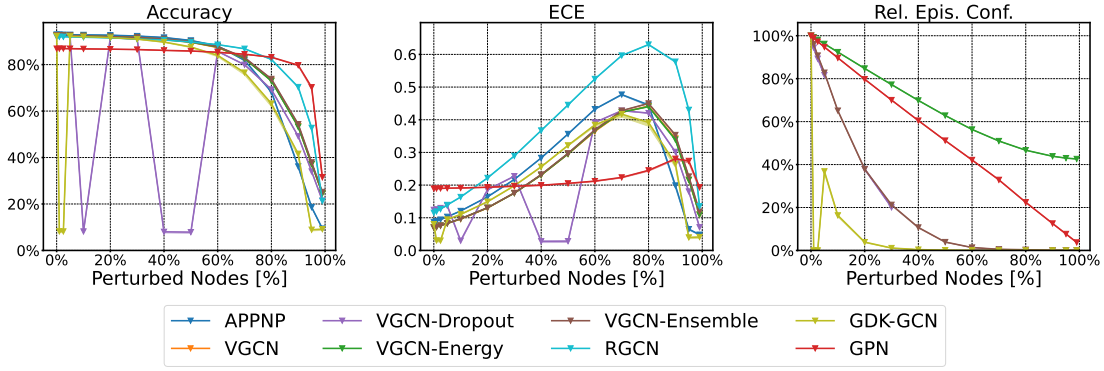


(c) Random Edge Shift

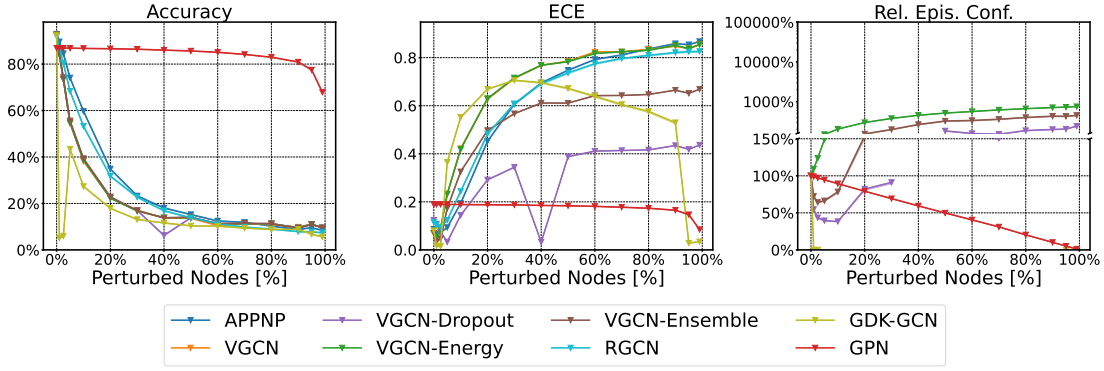


(d) DICE [431] Edge Shift

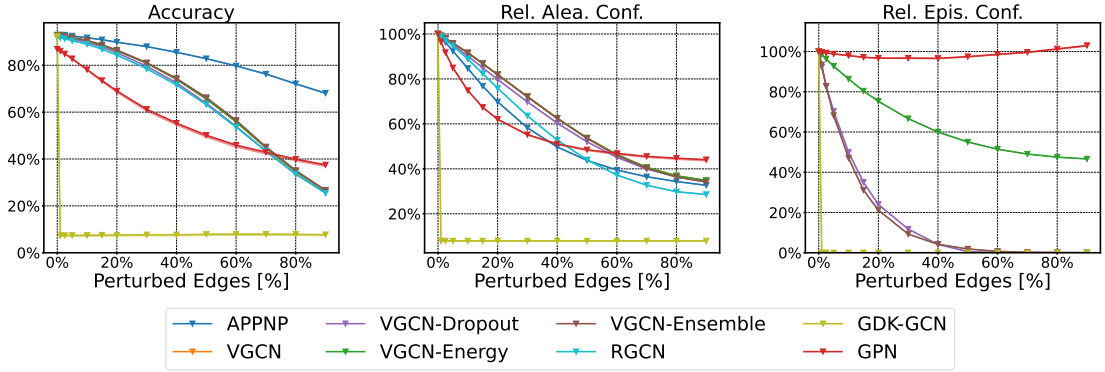
Figure E.6: Relative performance over different degrees of corruption of *Amazon Photos*. For feature shifts, we perturb different fractions of nodes (whose features are replaced with either random vectors from a Bernoulli noise or a Unit Gaussian noise) and show accuracy, ECE, and relative average epistemic confidence. For edge shifts, we perturb different fractions of edges (by replacing them at random or using the global and untargeted DICE [431] attack) and show accuracy, relative average aleatoric confidence, and relative average epistemic confidence.



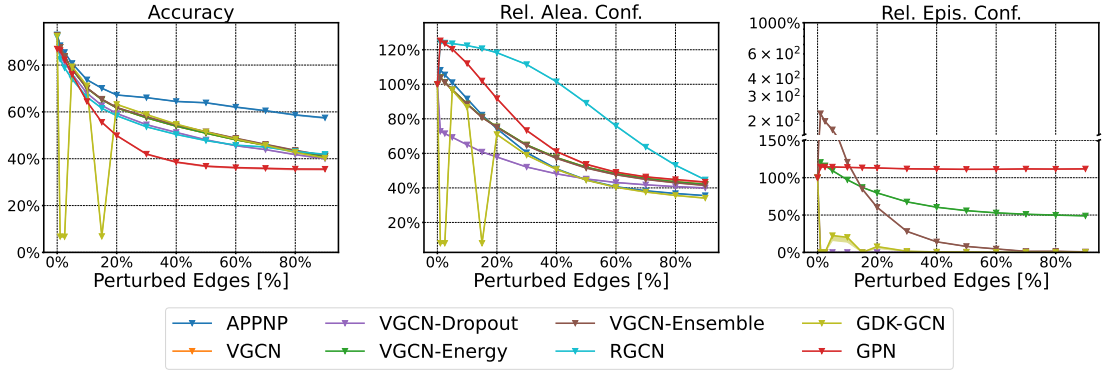
(a) Bernoulli Feature Shift



(b) Unit Gaussian Feature Shift

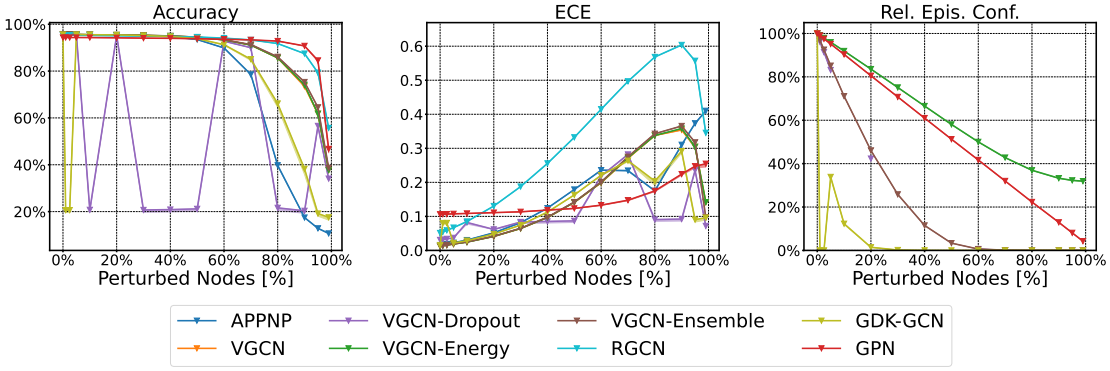


(c) Random Edge Shift

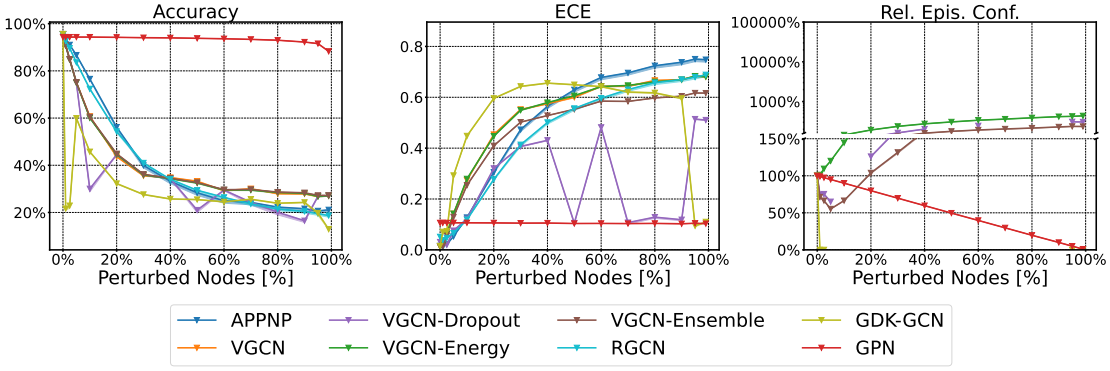


(d) DICE [431] Edge Shift

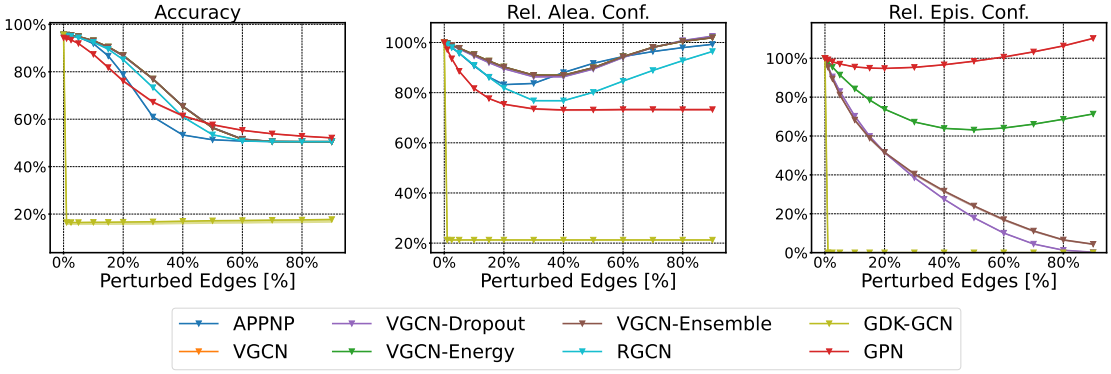
Figure E.7: Relative performance over different degrees of corruption of *Coauthor CS*. For feature shifts, we perturb different fractions of nodes (whose features are replaced with either random vectors from a Bernoulli noise or a Unit Gaussian noise) and show accuracy, ECE, and relative average epistemic confidence. For edge shifts, we perturb different fractions of edges (by replacing them at random or using the global and untargeted DICE [431] attack) and show accuracy, relative average aleatoric confidence, and relative average epistemic confidence.



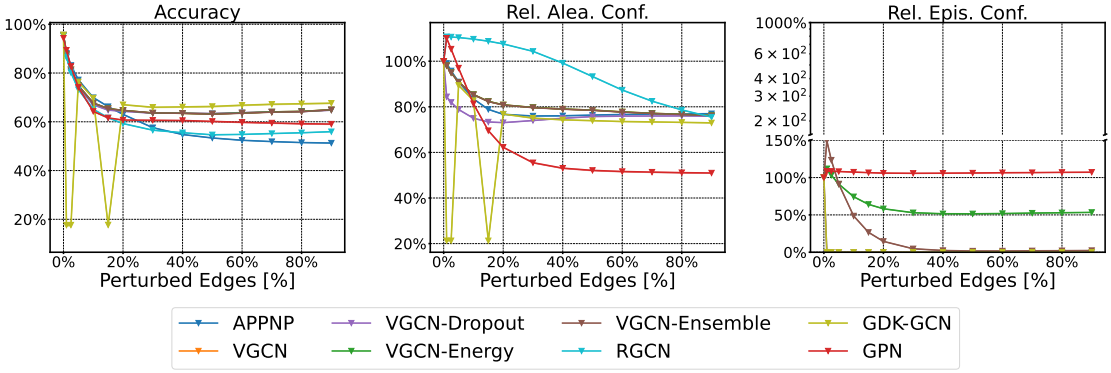
(a) Bernoulli Feature Shift



(b) Unit Gaussian Feature Shift

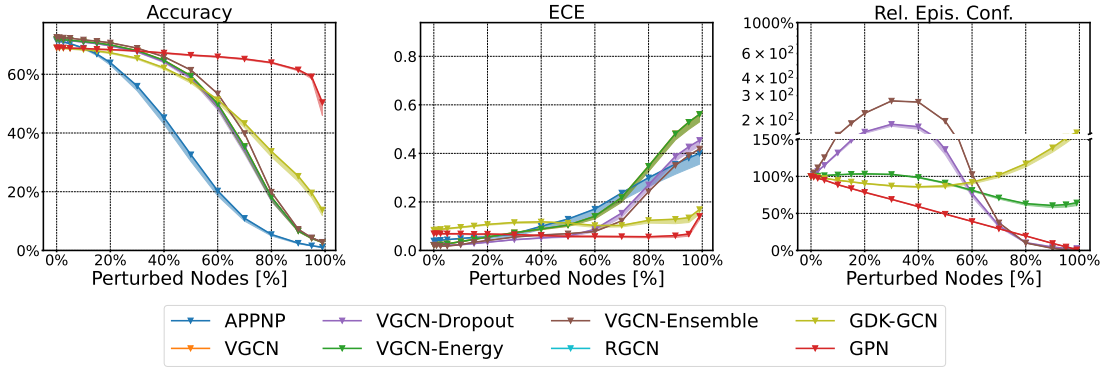


(c) Random Edge Shift

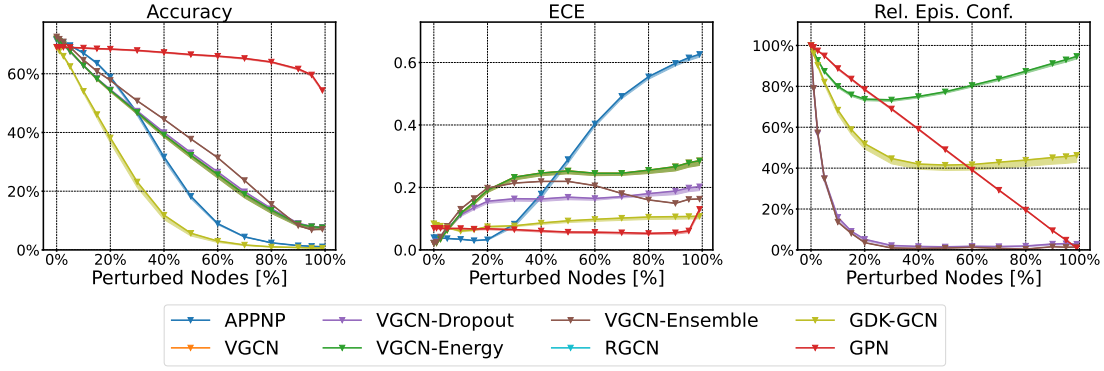


(d) DICE [431] Edge Shift

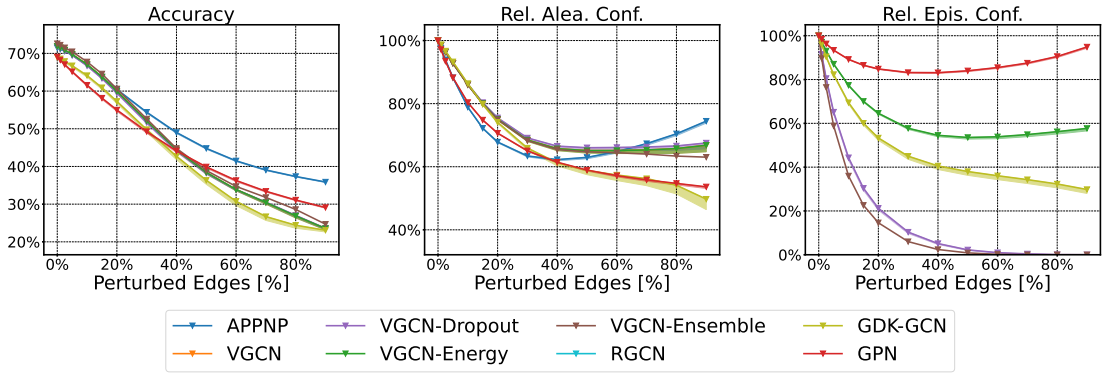
Figure E.8: Relative performance over different degrees of corruption of *Coauthor Physics*. For feature shifts, we perturb different fractions of nodes (whose features are replaced with either random vectors from a Bernoulli noise or a Unit Gaussian noise) and show accuracy, ECE, and relative average epistemic confidence. For edge shifts, we perturb different fractions of edges (by replacing them at random or using the global and untargeted DICE [431] attack) and show accuracy, relative average aleatoric confidence, and relative average epistemic confidence.



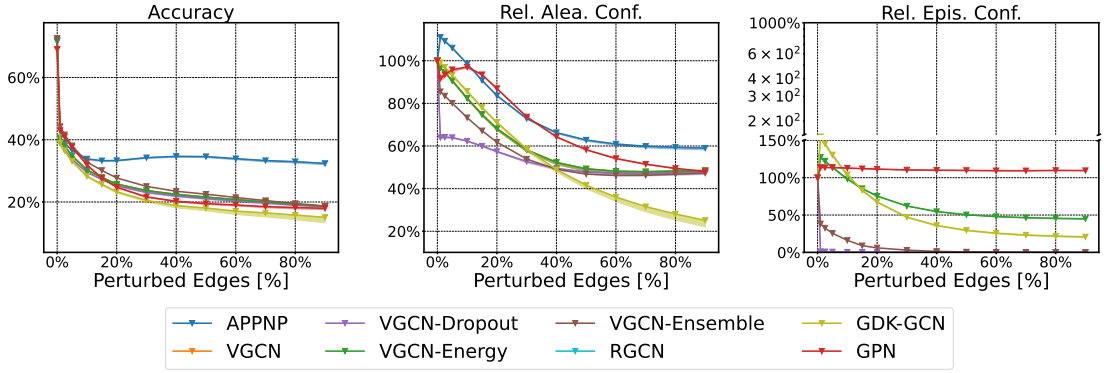
(a) Bernoulli Feature Shift



(b) Unit Gaussian Feature Shift



(c) Random Edge Shift

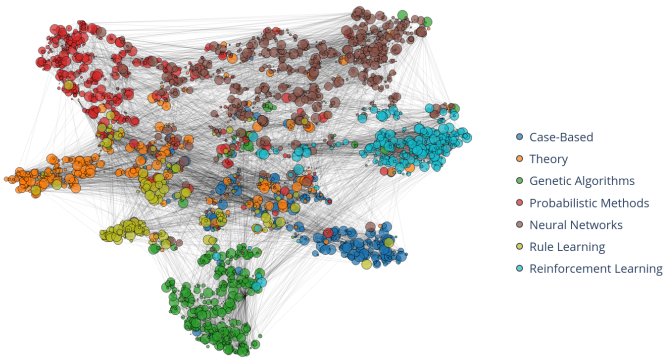


(d) DICE [431] Edge Shift

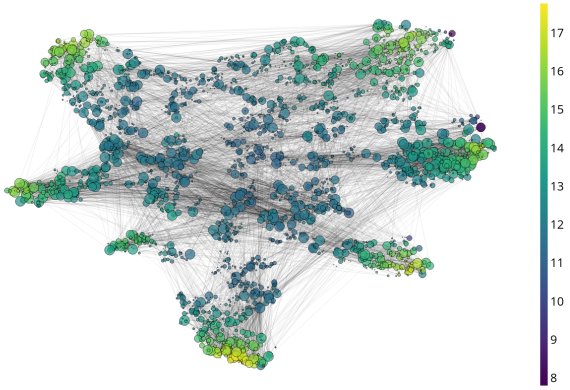
Figure E.9: Relative performance over different degrees of corruption of *OGBN Arxiv*. For feature shifts, we perturb different fractions of nodes (whose features are replaced with either random vectors from a Bernoulli noise or a Unit Gaussian noise) and show accuracy, ECE, and relative average epistemic confidence. For edge shifts, we perturb different fractions of edges (by replacing them at random or using the global and untargeted DICE [431] attack) and show accuracy, relative average aleatoric confidence, and relative average epistemic confidence.

Node	Node Representation
220	**Abstract:** IlliGAL Report No. 95006 July 1995 **Bag-of-Word:** ['1995', 'report', 'july']
197	**Abstract:** Report of the 1996 Workshop on Reinforcement **Bag-of-Word:** ['workshop', 'reinforcement', 'report', '1996']
193	**Abstract:** Report of the 1996 Workshop on Reinforcement **Bag-of-Word:** ['workshop', 'reinforcement', 'report', '1996']
258	**Abstract:** TIK-Report Nr. 11, December 1995 Version 2 (2. Edition) **Bag-of-Word:** ['version', '1995', 'report', '11']
2319	**Abstract:** We tend to think of what we really know as what we can talk about, and disparage knowledge that we can't verbalize. [Dowling 1989, p. 252] **Bag-of-Word:** ['knowledge', 'know', '1989', 'tend']
1945	**Abstract:** Reihe FABEL-Report Status: extern Dokumentbezeichner: Org/Reports/nr-35 Erstellt am: 21.06.94 Korrigiert am: 28.05.95 ISSN 0942-413X **Bag-of-Word:** ['reports', '94', 'report', '95']
293	**Abstract:** Keith Mathias and Darrell Whitley Technical Report CS-94-101 January 7, 1994 **Bag-of-Word:** ['technical', '1994', '94', 'cs', 'report', 'january']
99	**Abstract:** Multigrid Q-Learning Charles W. Anderson and Stewart G. Crawford-Hines Technical Report CS-94-121 October 11, 1994 **Bag-of-Word:** ['learning', 'technical', '1994', '94', 'cs', 'report', '11']
766	**Abstract:** Internal Report 97-01 **Bag-of-Word:** ['internal', 'report', '97']
992	**Abstract:** A Learning Result for Abstract **Bag-of-Word:** ['learning', 'result', 'abstract']
2030	**Abstract:** DRAFT March 16, 1998 available via anonymous ftp: site ftp.uwasa.fi directory cs/report94-1 file gaGPbib.ps.Z **Bag-of-Word:** ['available', '1998', 'cs', '16', 'anonymous', 'ftp', 'march', 'fi', 'site']
991	**Abstract:** In IEEE Transactions on Neural Networks, 7(1):97-106, 1996 Also available as GMD report 794 **Bag-of-Word:** ['available', 'networks', 'neural', 'report', 'ieee', '1996', '97']
74	**Abstract:** Empirical Comparison of Gradient Descent and Exponentiated Gradient Descent in Supervised and Reinforcement Learning Technical Report 96-70 **Bag-of-Words:** ['learning', 'technical', 'empirical', 'gradient', 'comparison', 'reinforcement', 'supervised', 'report', 'descent', '96']
865	**Abstract:** fl Partially supported by the Advanced Research Projects Agency (AFOSR 90-0083). y Partially supported by the Air Force Office of Scientific Research (AFOSR F49620-92-J-0499), the Advanced Research Projects Agency (ONR N00014-92-J-4015), and the Office of Naval Research (ONR N00014-91-J-4100). z Partially funded by the Air Force Office of Scientific Research (AFOSR F49620-92-J-0334) and the Office of Naval Research (ONR N00014-91-J-4100 and ONR N00014-94-1-0597). **Bag-of-Words:** ['fl', 'research', '90', 'partially', '94', 'advanced', 'projects', 'agency', 'supported', 'n00014', 'office', 'naval', 'force', 'air', 'scientific', 'afosr', '91', '92', 'onr', 'funded']
858	**Abstract:** Technical Report UMIACS-TR-97-77 and CS-TR-3843 Abstract **Bag-of-Words:** ['technical', 'abstract', 'cs', 'report', '97', 'tr']
1944	**Abstract:** Reihe FABEL-Report Status: extern Dokumentbezeichner: Org/Reports/nr-13 Erstellt am: 22.12.93 Korrigiert am: 02.02.94 ISSN 0942-413X **Bag-of-Words:** ['reports', '13', '94', 'report', '22', '93', '12']
1710	**Abstract:** Knowledge Systems Laboratory March 1995 Report No. KSL 95-32 **Bag-of-Words:** ['systems', 'knowledge', 'laboratory', '1995', 'report', 'march', '95']
2491	**Abstract:** Edward S. Orosz and Charles W. Anderson Technical Report CS-94-111 April 27, 1994 **Bag-of-Words:** ['technical', '1994', '94', 'cs', 'report', 'april']
221	**Abstract:** V. Scott Gordon and Darrell Whitley Technical Report CS-93-114 September 16, 1993 **Bag-of-Words:** ['technical', '1993', 'cs', 'report', '16', '93']
1874	**Abstract:** COINS Technical Report 94-61 August 1994 **Bag-of-Words:** ['technical', '1994', '94', 'report', 'august']

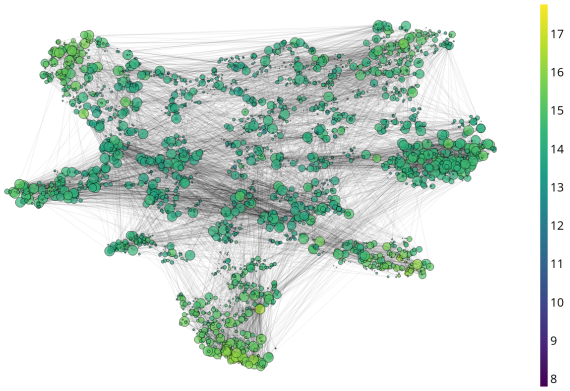
Node	Node Representation
1637	**Abstract:** A pervasive, yet much ignored, factor in the analysis of processing-failures is the problem of misorganized knowledge. If a systems knowledge is not indexed or organized correctly, it may make an error, not because it does not have either the general capability or specific knowledge to solve a problem, but rather because it does not have the knowledge sufficiently organized so that the appropriate knowledge structures are brought to bear on the problem at the appropriate time. In such cases, the system can be said to have forgotten the knowledge, if only in this context. This is the problem of forgetting or retrieval failure. This research presents an analysis along with a declarative representation of a number of types of forgetting errors. Such representations can extend the capability of introspective failure-driven learning systems, allowing them to reduce the likelihood of repeating such errors. Examples are presented from the Meta-AQUA program, which learns to improve its performance on a story understanding task through an introspective meta-analysis of its knowledge, its organization of its knowledge, and its reasoning processes. **Bag-of-Word:** ['retrieval', 'number', 'problem', 'learning', 'systems', 'representations', 'knowledge', 'representation', 'understanding', 'program', 'learns', 'time', 'make', 'examples', 'appropriate', 'general', 'error', 'analysis', 'presented', 'cases', 'performance', 'correctly', 'task', 'presents', 'improve', 'reasoning', 'likelihood', 'research', 'driven', 'processes', 'does', 'reduce', 'processing', 'organization', 'extend', 'solve', 'specific', 'meta', 'factor', 'declarative', 'failure', 'capability', 'context', 'allowing', 'structures', 'types', 'ignored', 'failures', 'story', 'errors', 'sufficiently', 'brought', 'organized', 'introspective', 'bear']
1375	**Abstract:** This paper describes an interactive planning system that was developed inside an Intelligent Decision Support System aimed at supporting an operator when planning the initial attack to forest fires. The planning architecture rests on the integration of case-based reasoning techniques with constraint reasoning techniques exploited, mainly, for performing temporal reasoning on temporal metric information. Temporal reasoning plays a central role in supporting interactive functions that are provided to the user when performing two basic steps of the planning process: plan adaptation and resource scheduling. A first prototype was integrated with a situation assessment and a resource allocation manager subsystem and is currently being tested. **Bag-of-Word:** ['intelligent', 'information', 'paper', 'planning', 'based', 'case', 'integrated', 'functions', 'describes', 'decision', 'allocation', 'architecture', 'support', 'reasoning', 'mainly', 'techniques', 'process', 'supporting', 'role', 'integration', 'exploited', 'currently', 'initial', 'user', 'tested', 'interactive', 'basic', 'developed', 'temporal', 'central', 'assessment', 'situation', 'steps', 'adaptation', 'operator', 'scheduling', 'constraint', 'performing', 'plays', 'provided', 'resource', 'prototype', 'metric', 'aimed', 'plan', 'inside']
2672	**Abstract:** Inductive learning systems are designed to induce hypotheses, or general descriptions of concepts, from instances of these concepts. Among the wide variety of techniques used in inductive learning systems, algorithms derived from nearest neighbour (NN) pattern classification have been receiving attention lately, mainly due to their incremental nature. Nested Generalized Exemplar (NGE) theory is an inductive learning theory which can be viewed as descent from nearest neighbour classification. In NGE theory, the induced concepts take the form of hyperrectangles in a n-dimensional Euclidean space. The axes of the space are defined by the attributes used for describing the examples. This paper proposes a fuzzified version of the original NGE algorithm, which accepts input examples given as feature/fuzzy value pairs, and generalizes them as fuzzy hyperrectangles. It presents and discusses a metric for evaluating the fuzzy distance between examples, and between example and fuzzy hyperrectangles; criteria for establishing the reliability of fuzzy examples, by strengthening the exemplar which makes the right prediction and weakening the exemplar which makes a wrong one and criteria for producing fuzzy generalizations, based on the union of fuzzy sets. Keywords : exemplar-based learning, nested generalized exemplar, nearest neighbour, fuzzy NGE. **Bag-of-Word:** Bag-of-Words ['paper', 'used', 'learning', 'systems', 'feature', 'theory', 'based', 'input', 'algorithm', 'generalizations', 'algorithms', 'given', 'hypotheses', 'form', 'attributes', 'examples', 'general', 'concepts', 'instances', 'evaluating', 'prediction', 'classification', 'space', 'keywords', 'presents', 'pattern', 'version', 'receiving', 'mainly', 'techniques', 'nature', 'sets', 'wide', 'right', 'makes', 'variety', 'attention', 'example', 'original', 'describing', 'distance', 'generalized', 'inductive', 'nearest', 'dimensional', 'descriptions', 'value', 'incremental', 'designed', 'viewed', 'discusses', 'derived', 'fuzzy', 'pairs', 'generalizes', 'producing', 'defined', 'descent', 'induced', 'nn', 'wrong', 'criteria', 'reliability', 'proposes', 'metric', 'induce', 'euclidean']



(a) Ground-Truth Classes

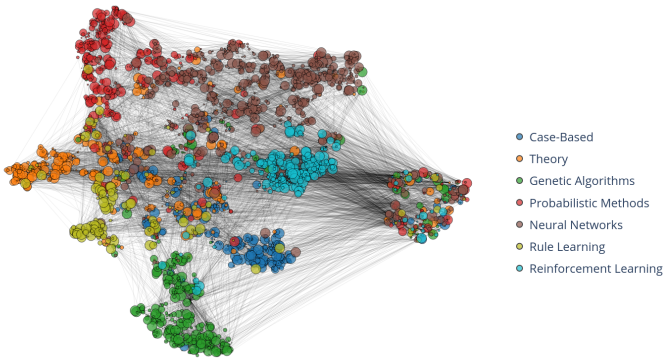


(b) Feature Evidence

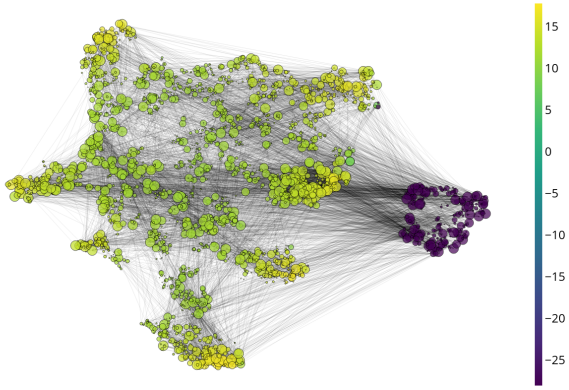


(c) Aggregated Evidence

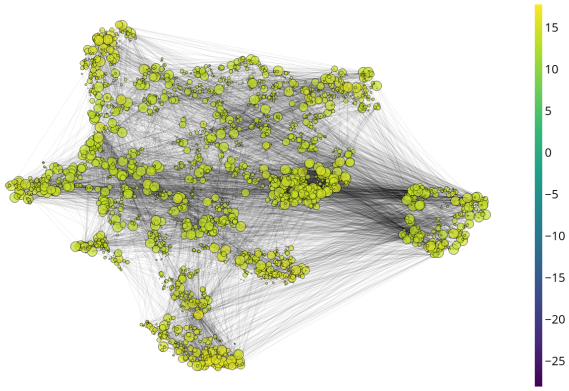
Figure E.10: Latent space visualizations on the clean CoraML graph. The ground-truth classes are shown in different colors. The feature and aggregated evidence are represented in log-scale.



(a) Ground-Truth Classes

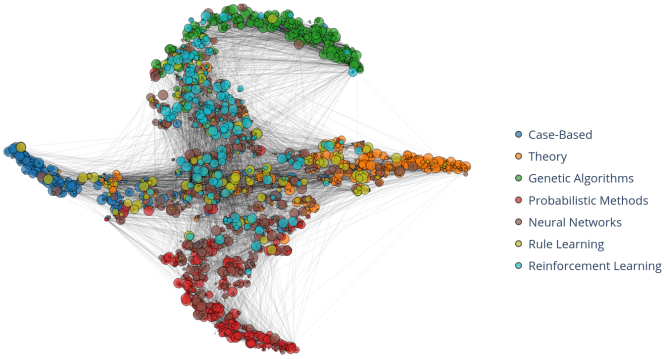


(b) Feature Evidence

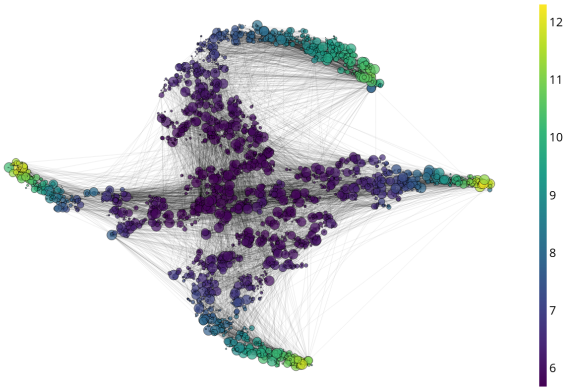


(c) Aggregated Evidence

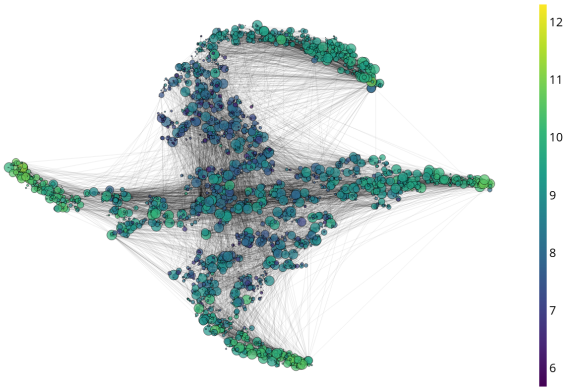
Figure E.11: Latent space visualizations on the CoraML where 10% of the nodes are perturbed with unit Gaussians. The ground-truth classes are shown in different colors. The feature and aggregated evidence are represented in log-scale.



(a) Ground-Truth Classes



(b) Feature Evidence



(c) Aggregated Evidence

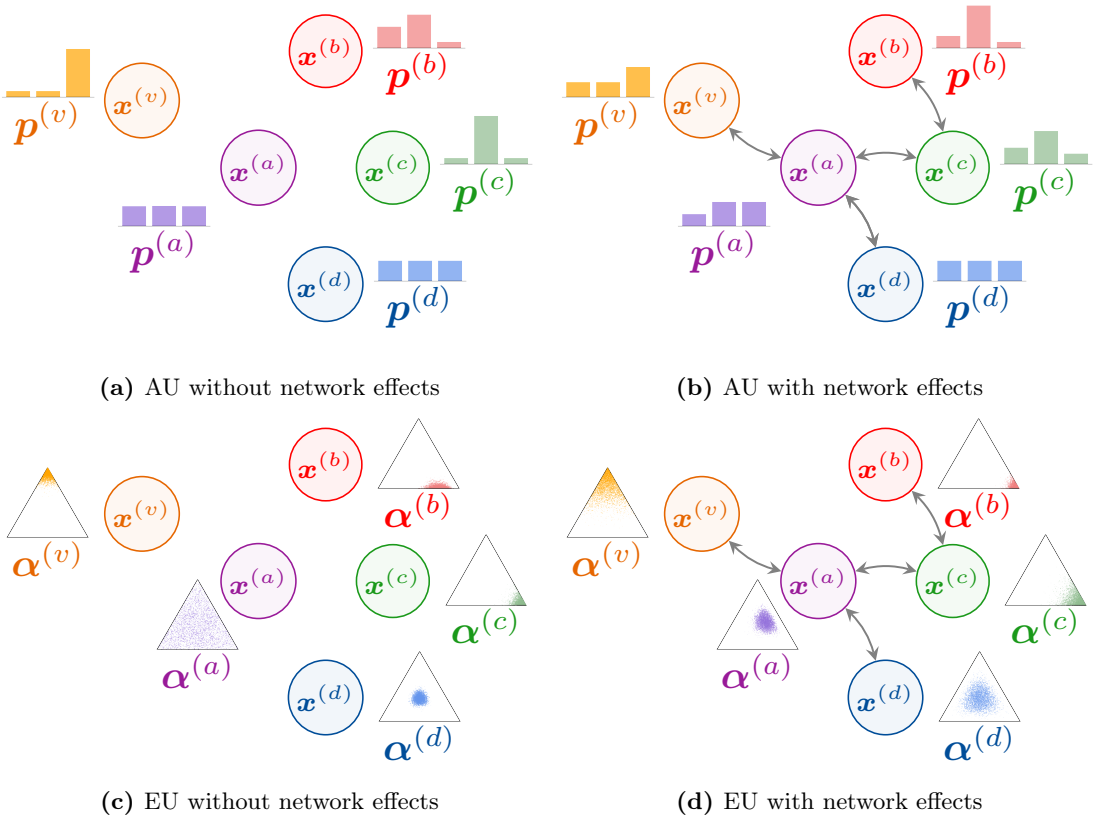
Figure E.12: Latent space visualizations on the CoraML with Left-Out classes experiments. The ground-truth classes are shown in different colors. Nodes of the classes *Neural Networks*, *Rule Learning*, and *Reinforcement Learning* have been left out for training but are kept in the graph. The feature and aggregated evidence are represented in log-scale.

Model	CoraML	CiteSeer	PubMed	Amazon C.	Amazon Ph.	Coauthor CS	Coauthor Ph.	OGBN Arxiv
APPNP	2.87 ± 0.01	2.97 ± 0.01	3.07 ± 0.02	6.53 ± 0.05	3.50 ± 0.03	6.67 ± 0.01	10.33 ± 0.04	$\mathbf{49.50 \pm 0.33}$
VGCN	$\mathbf{2.55 \pm 0.04}$	2.73 ± 0.02	2.70 ± 0.01	6.41 ± 0.09	3.96 ± 0.03	6.42 ± 0.02	12.27 ± 0.04	60.90 ± 0.04
VGCN-Dropout	24.40 ± 0.17	26.96 ± 0.25	30.43 ± 0.20	58.24 ± 0.20	36.56 ± 0.13	62.21 ± 0.26	120.68 ± 0.21	637.63 ± 0.43
VGCN-Energy	2.78 ± 0.03	2.78 ± 0.02	3.24 ± 0.03	6.03 ± 0.03	3.79 ± 0.01	6.59 ± 0.02	12.46 ± 0.03	62.45 ± 0.32
VGCN-Ensemble	23.18 ± 0.63	23.10 ± 0.02	27.22 ± 0.42	52.78 ± 0.65	32.81 ± 0.14	54.17 ± 0.46	108.58 ± 0.51	548.27
GKDE-GCN	2.61 ± 0.03	$\mathbf{2.18 \pm 0.01}$	$\mathbf{2.57 \pm 0.02}$	$\mathbf{3.54 \pm 0.04}$	$\mathbf{3.00 \pm 0.10}$	$\mathbf{4.92 \pm 0.02}$	$\mathbf{8.79 \pm 0.09}$	62.21 ± 0.34
GPN	9.35 ± 0.06	9.02 ± 0.02	14.13 ± 0.02	27.48 ± 0.09	14.52 ± 0.05	48.07 ± 0.15	35.94 ± 0.13	275.69 ± 0.91

Table E.16: Inference time (in ms) across different datasets evaluated on a single NVIDIA GTX 1080 Ti. Bold numbers indicate the fastest algorithm during inference for each dataset while red numbers indicate the slowest one.

Model	CoraML	CiteSeer	PubMed	Amazon C.	Amazon Ph.	Coauthor CS	Coauthor Ph.	OGBN Arxiv
APPNP	45.44	27.74	**54.92**	257.55	230.75	166.84	217.73	425.75
VGCN	47.28	32.98	55.77	211.79	174.00	167.77	194.53	**329.43**
VGCN-Dropout	47.28	32.98	55.77	211.79	174.00	167.77	194.53	**329.43**
VGCN-Energy	47.28	32.98	55.77	211.79	174.00	167.77	194.53	**329.43**
VGCN-Ensemble	472.82	329.84	557.65	2117.86	1740.00	1677.72	1945.34	3294.34
GKDE-GCN	46.48	**17.77**	523.41	354.66	473.71	247.01	475.40	60185.40
GKDE	5.12	5.04	54.72	61.27	21.24	103.34	320.98	59775.90
GPN	**10.20**	39.40	59.15	**81.59**	**64.72**	**32.80**	**93.38**	2393.03

Table E.17: Average training times (in s) for a single model and initializations across different datasets evaluated mostly on a single NVIDIA GTX 1080 Ti. Bold numbers indicate the fastest algorithm during training for each dataset while red numbers indicate the slowest one. The gray line shows the GKDE component of the GKDE-GCN approach as reference.



(a) AU without network effects

(b) AU with network effects

(c) EU without network effects

(d) EU with network effects

Figure E.13: Illustration of aleatoric uncertainty (AU) and epistemic uncertainty (EU) without and with network effects (i.e. i.i.d. inputs vs interdependent inputs). Each node of one color has the same features in all four cases. Network effects are visualized through edges between nodes which change the predicted distributions. The aleatoric uncertainty is high if the categorical distribution $\hat{y}^{(v)} \sim \text{Cat}(\boldsymbol{p}^{(v)})$ is flat. The epistemic uncertainty is high if the Dirichlet distribution $\boldsymbol{p}^{(v)} \sim \text{Dir}(\boldsymbol{\alpha}^{(v)})$ is spread out. Note that node with high epistemic certainty in the absence of network effects (e.g. orange) get less certain with neighbors being epistemically uncertain (purple). Epistemically uncertain nodes (purple) get more certain with certain neighbors on the other hand. Similar effects are shown for aleatoric uncertainty. For more details behind that reasoning, see our axiomatic approach in Section 8.3.1.

F Uncertainty Estimation for Sequential Data

F.1 Distributions

For reference, we give here the definition of the Dirichlet and Logistic-normal distribution.

F.1.1 Dirichlet distribution

The Dirichlet distribution with concentration parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$, where $\alpha_i > 0$, has the probability density function:

$$f(\boldsymbol{x}; \boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{F.1}$$

where Γ is a gamma function:

$$\Gamma(\alpha) = \int_0^\infty \alpha^{z-1} e^{-\alpha} dz$$

F.1.2 Logistic-normal distribution (LN)

The logistic normal distribution is a generalization of the logit-normal distribution for the multidimensional case. If $\boldsymbol{y} \in \mathbb{R}^C$ follows a normal distribution, $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then

$$\boldsymbol{x} = \left[\frac{e^{y_1}}{\sum_{i=1}^{C} e^{y_i}}, \ldots, \frac{e^{y_C}}{\sum_{i=1}^{C} e^{y_i}} \right]$$

follows a logistic-normal distribution.

F.2 Behavior of the min kernel

The desired behavior of the min kernel function can easily be illustrated by considering the gram matrix $\boldsymbol{K}$ and vector $\boldsymbol{k}$, which are required to estimate μ and σ^2 for a new time point τ. W.l.o.g. consider M pseudo points $\tau_1, \ldots, \tau_M$ such that $w_1 < \cdots < w_M$.

Since the new query point is observed we assign it weight 1. It follows:

$$\boldsymbol{k} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} \odot \begin{bmatrix} k(\tau_1, \tau) \\ k(\tau_2, \tau) \\ \vdots \\ k(\tau_M, \tau) \end{bmatrix}, \; \boldsymbol{K} = \begin{bmatrix} w_1 & w_1 & \dots & w_1 \\ w_1 & w_2 & \dots & w_2 \\ \vdots & \vdots & \ddots & \vdots \\ w_1 & w_2 & \dots & w_M \end{bmatrix} \odot \begin{bmatrix} k(\tau_1, \tau_1) & \dots & k(\tau_1, \tau_M) \\ k(\tau_2, \tau_1) & \dots & k(\tau_2, \tau_M) \\ \vdots & \ddots & \vdots \\ k(\tau_M, \tau_1) & \dots & k(\tau_M, \tau_M) \end{bmatrix} \tag{F.2}$$

Assuming $w_1 = 0$ returns $\boldsymbol{k}$ without the first row and $\boldsymbol{K}$ without the first row and column. Plugging them back into Eq. (9.1) we can see that the point τ_1 is discarded, as desired. In practice, the weights have values from interval $[0, 1]$ which in turn gives us the ability to *softly discard* points. This is shown in Fig. 9.3 we can see that the mean line does not have to cross through the points with weights < 1 and the variance can remain higher around them.

F.3 Computation of the approximation for the uncertainty cross-entropy of WGP-LN

Given true categorical distribution $\boldsymbol{p}_i^*$, and predicted $\boldsymbol{p}_i(\tau)$, the uncertainty cross-entropy can be calculated as in Eq. (9.4). For the WGP-LN model $\boldsymbol{p}_i(\tau) = \text{softmax}(\boldsymbol{z}_i(\tau))$, where $\boldsymbol{z}_i(\tau)$ are logits that come from a Gaussian process and follow a normal distribution $\mathcal{N}(\boldsymbol{\mu}_i(\tau), \boldsymbol{\Sigma}_i(\tau))$,. Therefore, $\exp(\boldsymbol{z}_i(\tau))$ follows a log-normal distribution. We will use this to derive an approximation of the loss. From now on, we omit τ from the equations. Mean and variance for $\sum_c^C \exp(\boldsymbol{z}_{c_i})$ are then:

$$\begin{aligned} \mathbb{E}\left[\sum\nolimits_c^C \exp(\boldsymbol{z}_{c_i})\right] &= \sum\nolimits_c^C \exp(\boldsymbol{\mu}_{c_i} + \boldsymbol{\sigma}_{c_i}^2/2) \\ \mathbf{Var}\left[\sum\nolimits_{c_i}^C \exp(\boldsymbol{z}_{c_i})\right] &= \sum\nolimits_{c_i}^C (\exp(\sigma_{c_i}^2) - 1)\exp(2\boldsymbol{\mu}_{c_i} + \boldsymbol{\sigma}_{c_i}^2) \end{aligned} \tag{F.3}$$

The expectation of the cross entropy loss given that logits are following a normal distribution is

$$\mathcal{L}_i^{\text{UCE}} = \mathbb{E}[\mathcal{L}_i^{\text{CE}}] = \mathbb{E}[\log(\exp(\boldsymbol{z}_{c_i}))] - \mathbb{E}\left[\log\left(\sum\nolimits_c^C \exp(\boldsymbol{z}_{c_i})\right)\right] \tag{F.4}$$

In general, given a random variable x, we can approximate expectation of $\log x$ by performing a second order Taylor expansion around the mean μ:

$$\begin{aligned} \mathbb{E}[\log x] &\approx \mathbb{E}\Big[\log\mu + \underbrace{\frac{(\log\mu)'}{1!}(x-\mu)}_{\mathbb{E}[x-\mu]=0} + \frac{(\log\mu)''}{2!}(x-\mu)^2\Big] \\ &\approx \mathbb{E}[\log\mu] - \frac{\mathbf{Var}[x]}{2\mu^2} \end{aligned} \tag{F.5}$$

Using Eq. (F.5) together with Eq. (F.3) and plugging into Eq. (F.4) we get a closed-form solution for the loss for event i:

$$\mathcal{L}_i^{\text{UCE}} \approx \mu_{c_i}(\tau_i^*) - \log\Big(\sum_c^C \exp(\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*)/2)\Big) + \frac{\sum_c^C (\exp(\sigma_c^2(\tau_i^*)) - 1)\exp(2\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*))}{2\Big(\sum_c^C \exp(\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*)/2)\Big)^2} \tag{F.6}$$

F.4 Non Expressiveness of RMTPP intensities

The intensity function has the following form in the RMTPP model [109]:

$$\log \lambda_0(t) = \boldsymbol{v}^T \cdot \boldsymbol{h}_i + w(t - t_i) + b \tag{F.7}$$

The variables $\boldsymbol{v}$, w and b are learned parameters and $\boldsymbol{h}_i$ is given by the hidden state of an RNN. The only dependence on t is $(t - t_i)$. RMTPP is then limited to monotonic intensity functions with respect to time.

F.5 Dirichlet Evolution

Our goal is to model the evolution of a distribution on a probability simplex. Fig. 9.1b shows this for two classes. In general, we can do the same for multiple classes. Fig. F.1 shows an example of the Dirichlet distribution for three classes, and how it changes over time. This evolution is the output of the FD-Dir model trained on the 3-G dataset, created to simulate the car example from Section 9.1 (see also Fig. F.6a in Appendix F.7). The three classes: *overtaking*, *breaking* and *collision* occur independently of each other at three different times. The represent the corners of the triangle in Fig. F.1.

We can distinguish three cases: (a) at first we are certain that the most likely class is *overtaking*; (b) as time passes, the most likely class becomes *breaking*, (c) and finally *collision*. After that, we are in the area where we have not seen any data and do not have a confident prediction (d).

(a) $\tau = 0$ **(b)** $\tau = 0.5$ **(c)** $\tau = 1.$ **(d)** $\tau = 2.$

Figure F.1: Dirichlet distribution at different time for the 3-G dataset with $\sigma = 1$.

F.6 Comparison of the classical cross-entropy and the uncertainty cross-entropy

F.6.1 Simple classification task

In this section, we do not consider temporal data. The goal of this experiment is to show the benefit of the uncertainty cross-entropy compare with the classical cross-entropy loss on a simple classification task. As a consequence, we do not consider RNN in this section. We use a simple two layers neural network to predict the concentration parameters of a Dirichlet distribution from the input vector.

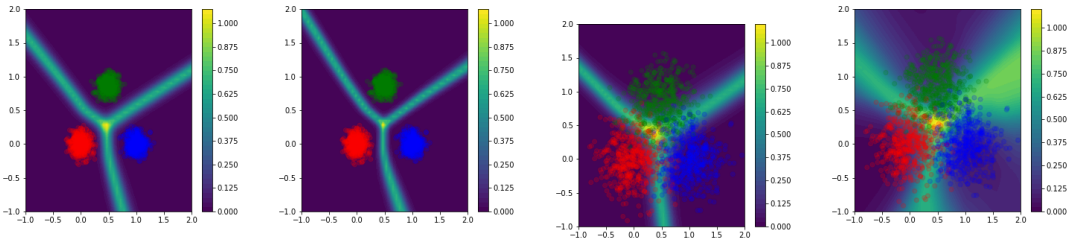
Set-up. The set-up is similar to [270] and consists of two datasets of 1500 instances divided in three equidistant 2-D Gaussians. One dataset contains non-overlapping classes (**NOG**) whereas the other contains overlapping classes (**OG**). Given one input x_i, we train simple two layers neural networks to predict the concentration parameters of a Dirichlet distribution $\mathbf{Dir}(\alpha_1(x_i), \alpha_2(x_i), \alpha_3(x_i))$ which model the uncertainty on the categorical distribution $\boldsymbol{p}(x_i)$. On each dataset, we train two neural networks. One neural network is trained with the classic cross-entropy loss $\mathcal{L}^{\text{CE}}$ which uses only the mean prediction $\bar{\boldsymbol{p}}(x_i)$. The second neural network is trained with the uncertainty cross-entropy loss plus a simple α-regularizer:

$$\mathcal{L}^{\text{UCE}} + \left|\alpha_0(x_i) - \sum_j \mathbb{1}_{x_j \in N_w(x_i)}\right| \tag{F.8}$$

where x_i is the input 2-D vector and $N_w(x_i) = \{x', ||x' - x_i||_2^2 < w\}$ is its euclidean neighbourhood of size w. We set $w = 10^{-5}$ for the non-overlapping Gaussians and $w = 10^{-2}$ for the overlapping Gaussians.

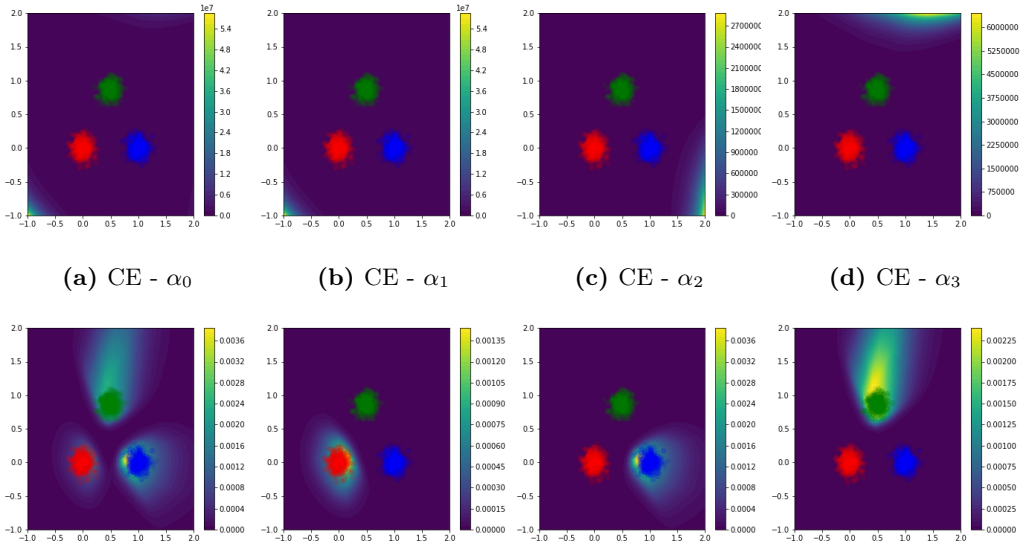
Results. The categorical entropy $-\sum_c p_c(x_i) \log p_c(x_i)$ is a good indicator to know how certain is the categorical distribution $\boldsymbol{p}(x_i)$ at point x_i. A high entropy meaning that the categorical distribution is uncertain. For non overlapping Gaussians (Figs. F.2a and F.2b), we remark that both losses learn uncertain categorical distribution only on thin borders. However, for overlapping Gaussians (See Figs. F.2c and F.2d),the uncertainty cross-entropy loss learns more uncertain categorical distributions because of the thicker borders.

Other interesting results are the concentration parameters learned by the two models (Figs. F.3 and F.4). The classic cross-entropy loss learns very high value for $\alpha_1(x_i), \alpha_2(x_i), \alpha_3(x_i)$ which does match with the true distribution of the data. In contrast, the uncertainty cross-entropy learn meaningful alpha values for both datasets (delimiting the in-distribution areas for α_0 and centred around the classes for the others).



(a) NOG - CE - Cat. Ent. **(b)** NOG - UCE - Cat. Ent. **(c)** OG - CE - Cat. Ent. **(d)** OG - UCE - Cat. Ent.

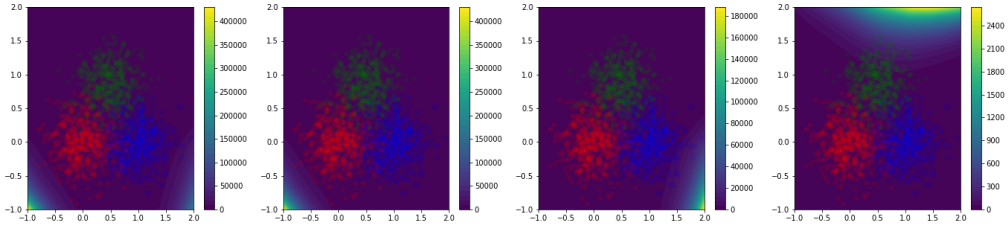
Figure F.2: The Figs. F.2a and F.2b plot the entropy of the categorical distribution learned on a classification task with three non-overlapping Gaussians. They show categorical entropy learned with the classic cross-entropy and learned with the uncertainty cross-entropy. The Figs. F.2c and F.2d plot the entropy of the categorical distribution learned on a classification task with three overlapping Gaussians. They show categorical entropy learned with the classic cross-entropy and learned with the uncertainty cross-entropy.



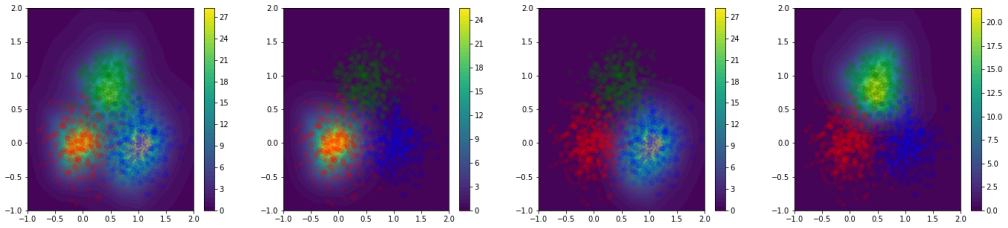
(a) CE - α_0 **(b)** CE - α_1 **(c)** CE - α_2 **(d)** CE - α_3

(e) UCE - α_0 **(f)** UCE - α_1 **(g)** UCE - α_2 **(h)** UCE - α_3

Figure F.3: Concentration parameters of the Dirichlet distribution on a classification task with three non-overlapping Gaussians. The Figs. F.3a to F.3d are α_0, α_1, α_2, α_3 learned with the classic cross-entropy. The Figs. F.3a to F.3d are α_0, α_1, α_2, α_3 learned with the uncertainty cross-entropy.



(a) CE - α_0 **(b)** CE - α_1 **(c)** CE - α_2 **(d)** CE - α_3



(e) UCE - α_0 **(f)** UCE - α_1 **(g)** UCE - α_2 **(h)** UCE - α_3

Figure F.4: Concentration parameters of the Dirichlet distribution on a classification task with three non-overlapping Gaussians. The Figs. F.3c, F.4a, F.4b and F.4d are α_0, α_1, α_2, α_3 learned with the classic cross-entropy. The Figs. F.3c, F.4a, F.4b and F.4d are α_0, α_1, α_2, α_3 learned with the uncertainty cross-entropy.

F.6.2 Asynchronous Event Prediction

In this section, we consider temporal data. The goal of this experiment is again to show the benefit of the uncertainty cross-entropy compared to the classical cross-entropy in the case of asynchronous event prediction.

Set-up. For this purpose, we use the same set-up describe in the experiment Anomaly detection & Uncertainty. We trained the model FD-Dir with three different type of losses: (1) The classical cross-entropy (CE), (2) The classical cross-entropy with regularization described in Section 9.2.3 (CE + reg) and (3) The classical uncertainty cross-entropy with regularization described in Section 9.2.3 (UCE + reg).

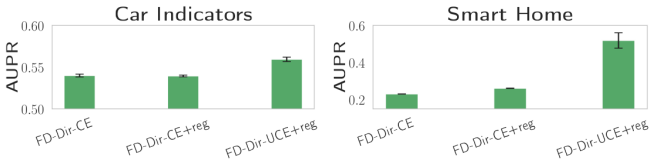


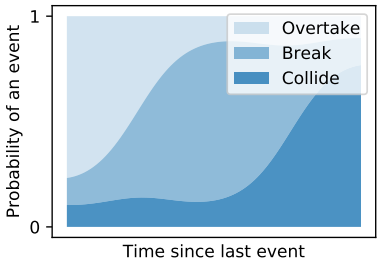
Figure F.5: Loss comparison in anomaly detection

Results. The results are shown in Fig. F.5. The loss UCE + reg consistently improves the anomaly detection based on the distribution uncertainty.

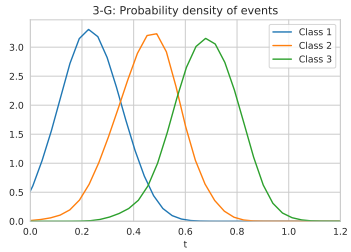
F.7 Datasets

In this section we describe the datasets in more detail. The time gap between two events $\tau_i^* = t_i - t_{i-1}$ is first log-transformed before applying min-max normalization: $\hat{\tau_i}^* = \frac{\tau_i' - \min(\tau_i^{*'})}{(\max(\tau_i^{*'}) - \min(\tau_i^{*'})}$ with $\tau_i^{*'} = \log(\tau_i^* + \epsilon)$, $\epsilon > 0$.

3-G. We use $C = 3$ and draw from a normal distribution $P(\tau|c_i) = \mathcal{N}(i + 1, 1.)$. This dataset tries to imitate the setting from Fig. F.6a as explained in Section 9.1. We generate 1000 events. Probability density is shown in Fig. F.6b. Models that are not taking time into account cannot solve this problem. Below is the code. We create the **Multi-G** dataset similarly.



(a) Car example explained in Section 9.1 where probabilities of events to occur change over time



(b) Probability density of events in K-Gaussians dataset. We can see that classes are independent of history.

```
def generate():
    data = np.zeros((1000, 2))
    for i in range(1000):
        i_class = np.random.choice(3, 1)[0]
        time = np.random.normal(i_class + 1, 1.)
        while time <= 0:
            time = np.random.normal(i_class + 1, 1.)
        data[i, 0] = i_class
        data[i, 1] = time
    return data
```

Car Indicators. A sequence contains signals from a single car during one ride. We remove signals that are perfectly correlated giving 6 unique classes in the end. Top 3 classes make up 33%, 32%, and 16% of a total respectively. From Fig. F.7 we can see that the setting is again asynchronous.

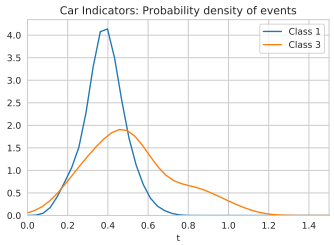


Figure F.7: Probability density of events in Car Indicators dataset for 2 selected classes. Time is log-transformed.

Graph. We generate graph G with 10 nodes and 48 edges between them. We assign variables μ and σ to each transition (edge) between events (nodes). The time it takes to make a transition between nodes i and j is drawn from normal distribution $\mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$. By performing a random walk on the graph we create 10 thousand events. This dataset is similar to K-Gaussians with the difference that a model needs to learn the relationship between events together with the time dependency. Parts of the trace are shown in Fig. F.8.

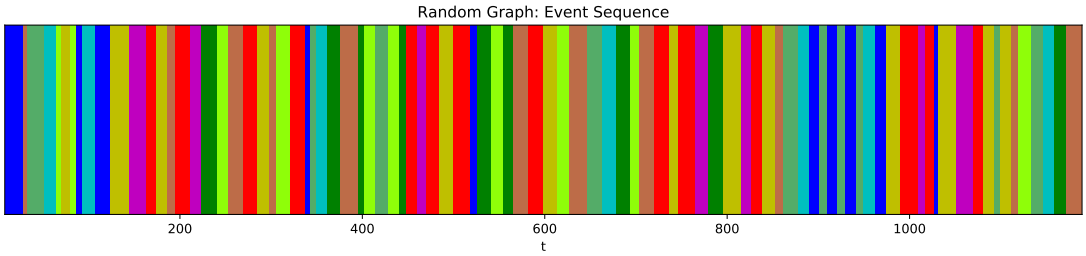


Figure F.8: Trace of events for random graph. Different colors represent different classes and width of a single column represents the time that passed.

F.8 Details of experiments

We test our models (**WGP-LN**, **FD-Dir** and **DPP**) against neural point process models (**RMTPP** and **Hawkes**) and simple baselines (**RNN** and **LSTM** – getting only history as an input, **F-RNN** and **F-LSTM** – having also the real time of the next event as an additional input; thus, they get a strong advantage!). We test on real world (**Stack Exchange**, **Car Indicators** and **Smart Home**) and synthetic datasets (**Graph**). We

show that our models consistently outperform all the other models when evaluated with class prediction accuracy and Time-Error.

F.8.1 Model selection

We apply the same tuning technique to all models. We split all datasets into train–validation–test sets ($60\% - 20\% - 20\%$), use the validation set to select a model and the test set to get final scores. For Stack Exchange dataset we split on users. In all other datasets we split the trace based on time. We search over dimension of a hidden state $\{32, 64, 128, 256\}$, batch size $\{16, 32, 64\}$ and L_2 regularization parameter $\{0, 10^{-3}, 10^{-2}, 10^{-1}\}$. We use the same learning rate 0.001 for all models and an Adam optimizer [219], run each of them 5 times for maximum of 100 epochs with early stopping after 5 consecutive epochs without improvement in the validation loss. For the number of points M we pick 3 for WGP-LN and 20 for FD-Dir. WGP-LN and FD-Dir have additional regularization (Eq. (9.7)) with hyperparameters α and β. For both models we choose $\alpha = \beta = 10^{-3}$. Model with the highest mean accuracy on the validation set is selected. We use GRU cell [80] for both of our models. We trained all models on GPUs (1TB SSD).

F.8.2 Results

Tables F.1 and F.2, together with Fig. F.9 show test results for *all* models on *all* datasets for Class accuracy and Time-Error.

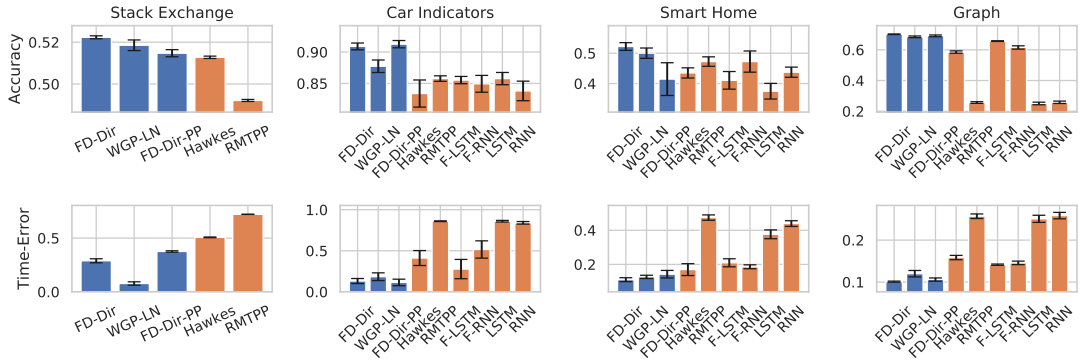


Figure F.9: Class accuracy (top) and Time-Error (bottom) comparison across datasets

F.8.3 Time Prediction with Point Processes

The benefit of the point process framework is the ability to get the point estimate for the time $\hat{\tau}$ of the next event:

$$\hat{\tau} = \int_0^\infty t q(\tau) dt \tag{F.9}$$

Table F.1: Class accuracy comparison for all models on all datasets

	Car Indicators	Graph	Smart Home	Stack Exchange
FD-Dir	0.909 ± 0.005	**0.701 ± 0.002**	**0.522 ± 0.013**	**0.522 ± 0.001**
Dir-PP	**0.912 ± 0.006**	0.691 ± 0.006	0.415 ± 0.054	0.515 ± 0.002
WGP-LN	0.877 ± 0.010	0.685 ± 0.005	0.500 ± 0.017	0.519 ± 0.003
Hawkes	0.834 ± 0.022	0.585 ± 0.008	0.435 ± 0.017	0.513 ± 0.001
RMTPP	0.858 ± 0.004	0.257 ± 0.005	0.472 ± 0.016	0.492 ± 0.000
F-LSTM	0.855 ± 0.006	0.657 ± 0.002	0.411 ± 0.029	-
F-RNN	0.849 ± 0.013	0.615 ± 0.011	0.472 ± 0.035	-
LSTM	0.858 ± 0.010	0.251 ± 0.008	0.375 ± 0.026	-
RNN	0.838 ± 0.016	0.258 ± 0.008	0.437 ± 0.017	-

Table F.2: Time-Error comparison for all models on all datasets

	Car Indicators	Graph	Smart Home	Stack Exchange
FD-Dir	**0.115 ± 0.040**	**0.101 ± 0.001**	**0.111 ± 0.011**	0.289 ± 0.019
WGP-LN	0.184 ± 0.047	0.120 ± 0.008	0.127 ± 0.010	**0.077 ± 0.016**
FD-Dir-PP	0.132 ± 0.031	0.106 ± 0.004	0.143 ± 0.022	0.375 ± 0.007
Hawkes	0.412 ± 0.091	0.158 ± 0.005	0.170 ± 0.035	0.507 ± 0.003
RMTPP	0.860 ± 0.004	0.257 ± 0.005	0.474 ± 0.016	0.721 ± 0.001
F-LSTM	0.277 ± 0.118	0.141 ± 0.002	0.209 ± 0.023	-
F-RNN	0.516 ± 0.105	0.146 ± 0.004	0.186 ± 0.011	-
LSTM	0.860 ± 0.010	0.251 ± 0.008	0.376 ± 0.026	-
RNN	0.841 ± 0.016	0.258 ± 0.008	0.439 ± 0.017	-

where

$$q(\tau) = \lambda_0(\tau) \exp\left(-\int_0^\tau \lambda_0(s)ds\right) \tag{F.10}$$

The usual way to evaluate the quality of this prediction is using an MSE score. As we have already discussed in Section 9.5, this is not optimal for our use case. Nevertheless, we did preliminary experiments comparing our neural point process model **FD-Dir-PP** to others. We use **RMTPP** [109] since it achieves the best results. On Car Indicators dataset our model has mean MSE score of 0.4783 while RMTPP achieves 0.4736. At the same time FD-Dir-PP outperforms RMTPP on other tasks (see Section 9.5).

G Uncertainty Estimation for Reinforcement Learning

G.1 Proofs

In this section, we show that deep kernel learning [421] and evidential models based on Posterior Networks [67, 69] are guaranteed to assign high epistemic uncertainty for (state) inputs far from (state) inputs observed during training under technical assumptions. In particular, the combination of DQN with deep kernel learning or evidential networks presented in Section 10.4 are guaranteed to assign high epistemic uncertainty for extreme input states. We assume that the encoder should use ReLU activations, which is common in deep learning, and that the rows of the linear transformations are independent, which is realistic for trained networks with no constant output [182].

Lemma 13. *[17] Let $\{Q_l\}_l^R$ be the set of linear regions associated to the piecewise ReLU network $f_\phi(\boldsymbol{x})$. For any $\boldsymbol{x} \in \mathbb{R}^D$, there exists $\delta^* \in \mathbb{R}^+$ and $l^* \in 1, \dots, R$ such that $\delta \cdot \boldsymbol{x} \in Q_{l^*}$ for all $\delta > \delta^*$.*

Lemma 14. *Let a (deep) encoder f_ϕ with piecewise ReLU activations. Let $f_\phi(\boldsymbol{x}) = V^{(l)}\boldsymbol{x} + a^{(l)}$ be the piecewise affine representation of the ReLU network f_ϕ on the finite number of affine regions $Q^{(l)}$ [17]. Suppose that $V^{(l)}$ have independent rows, then for almost any $\boldsymbol{x}$ we have $\|f_\phi(\delta \cdot \boldsymbol{x})\| \underset{\delta \to \infty}{\to} \infty$, i.e the norm of the latent representations $\boldsymbol{z}_\delta = f_\phi(\delta \cdot \boldsymbol{x})$ associated to the input $\delta \cdot \boldsymbol{x}$ goes to infinity.*

Proof. Let $\boldsymbol{x} \in \mathbb{R}^D$ be a non-zero input and f_ϕ be a ReLU network. Lemma 13 implies that there exists $\delta^* \in \mathbb{R}^+$ and $l \in \{1, ..., R\}$ such that $\delta \cdot \boldsymbol{x} \in Q^{(l)}$ for all $\delta > \delta^*$. Thus, $\boldsymbol{z}_\delta = f_\phi(\delta \cdot \boldsymbol{x}) = \delta \cdot (V^{(l)}\boldsymbol{x}) + a^{(l)}$ for all $\delta > \delta^*$. Note that for $\delta \in [\delta^*, +\infty]$, $\boldsymbol{z}_\delta$ follows an affine half line $S_{\boldsymbol{x}} = \{\boldsymbol{z} \,|\, \boldsymbol{z} = \delta \cdot (V^{(l)}\boldsymbol{x}) + a^{(l)}, \delta > \delta^*\}$ in the latent space. Further, note that $V^{(l)}\boldsymbol{x} \neq 0$ since $\boldsymbol{x} \neq 0$ and $V^{(l)}$ has independent rows. Therefore, we have $\|\boldsymbol{z}_\delta\| \underset{\delta \to \infty}{\to} +\infty$ □

Theorem 15. *Let a Deep Kernel Learning model parametrized with a (deep) encoder f_ϕ with piecewise ReLU activations, a set of K inducing points $\{\phi_k\}_{k=1}^K$ and a RBF, Matern or Rational Quadratic kernel $\kappa(\cdot, \cdot)$ [115, 356]. Let $f_\phi(\boldsymbol{x}) = V^{(l)}\boldsymbol{x} + a^{(l)}$ be the piecewise affine representation of the ReLU network f_ϕ on the finite number of affine regions $Q^{(l)}$ [17]. Suppose that $V^{(l)}$ have independent rows, then for almost any $\boldsymbol{x}$ we have $\sigma(f_\phi(\delta \cdot \boldsymbol{x})) \underset{\delta \to \infty}{\to} c$ where $c = \kappa(0, 0)$.*

Proof. Lemma 13 says that $\|\boldsymbol{z}_\delta\| \underset{\delta \to \infty}{\to} +\infty$ where $\boldsymbol{z}_\delta = f_\phi(\delta \cdot \boldsymbol{x})$. It implies that $\|\boldsymbol{z}_\delta - \boldsymbol{\phi}_k\| \underset{\delta \to \infty}{\to} \infty$ for all inducing point $\boldsymbol{\phi}_k$. Thus, we obtain $\kappa(\boldsymbol{z}_\delta, \boldsymbol{\phi}_k) \underset{\delta \to \infty}{\to} 0$ where $\kappa(\cdot, \cdot)$ is the

RBF, Matern or Rational Quadratic kernel [115, 356]. Since the variance of the predictive Gaussian distribution associated with the Gaussian process is $\sigma(f_\phi(\delta \cdot \boldsymbol{x})) = c - \boldsymbol{\kappa} \boldsymbol{C} \boldsymbol{\kappa}$ where $c = \kappa(f_\phi(\delta \cdot \boldsymbol{x}), f_\phi(\delta \cdot \boldsymbol{x})) = \kappa(0, 0)$, $\boldsymbol{\kappa}_k = \kappa(\boldsymbol{z}_\delta, \boldsymbol{\phi}_k)$ and $\boldsymbol{\kappa}_{k,k'} = \kappa(\boldsymbol{\phi}_k, \boldsymbol{\phi}_{k'})$. This gives the final result $\sigma(f_\phi(\delta \cdot \boldsymbol{x})) \underset{\delta \to \infty}{\to} c$ where $c = \kappa(0, 0)$. □

Theorem 15 implies that deep kernel learning on a latent space parametrized with a neural network is guaranteed to predict high uncertainty corresponding to the prior uncertainty far from training data. This includes the uncertainty predicted by the GP associated to each action a in the combination of DQN and deep kernel learning presented in Section 10.4. The uncertainty prediction $u_{\text{epist}}(s_t, a_t) = \mathbb{H}(\mathcal{N}(\mu(\boldsymbol{s}^{(t)}, a^{(t)}), \sigma(\boldsymbol{s}^{(t)}, a^{(t)})))$ becomes high for input states $s^{(t)}$ extremely different from the training environment, i.e. $\|s^{(t)}\| \to \infty$.

Theorem 16. *[69] Let a Natural Posterior Network model parametrized with a (deep) encoder f_ϕ with piecewise ReLU activations, a decoder g_ψ and the density $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$. Let $f_\phi(\boldsymbol{x}) = V^{(l)} \boldsymbol{x} + a^{(l)}$ be the piecewise affine representation of the ReLU network f_ϕ on the finite number of affine regions $Q^{(l)}$ [17]. Suppose that $V^{(l)}$ have independent rows and the density function $\mathbb{P}(\boldsymbol{z} \,|\, \boldsymbol{\omega})$ has bounded derivatives, then for almost any $\boldsymbol{x}$ we have $\mathbb{P}(f_\phi(\delta \cdot \boldsymbol{x}) \,|\, \boldsymbol{\omega}) \underset{\delta \to \infty}{\to} 0$, i.e the evidence becomes small far from training data.*

The proof of Theorem 16 is already given in [69], and relies also on Lemma 14 and the fact that a smooth density estimator should converge to 0 far from training data. Intuitively, it implies that the epistemic associated to each possible action a by the combination of DQN and posterior network becomes high for input states $s^{(t)}$ extremely different from the training environment i.e. $\|s^{(t)}\| \to \infty$. In particular, prior parameter takes over in the posterior update (i.e. $n^{\text{post}}(\boldsymbol{s}^{(t)}, a)) \to n^{\text{prior}}$, $\boldsymbol{\chi}^{\text{post}}(\boldsymbol{s}^{(t)}, a) \to \boldsymbol{\chi}^{\text{prior}}$)

G.2 Model Details

We train all models on a single GPU (NVIDIA GTX 1080 Ti or NVIDIA GTX 2080 Ti, 11 GB memory). All models use the same core architecture. They use a 2 layers MLP with 128 hidden units for the CartPole environment, a 2 layers MLP with 64 hidden units for the Acrobot environment and a 3 layers MLP with 128 hidden units for the LunarLander environment. All models are trained using 5 random seeds with the Adam optimizer [219]. For fair comparison, we use the same hyperparameters for the DQN architecture in all uncertainty models: the target network parameters are completely updated (i.e. $\tau = 1$) every 10 training iterations. The epsilon-greedy strategy start with $\epsilon = 1$ and decay till $\epsilon = 0.01$ after 1000 iteration steps. The discount factor is set to 0.99. Further, we use a batch size of 16, a replay size of 1000 and a maximum number of training iterations of 13000 for Cartpole, a batch size of 64, a replay size of 10000 and a maximum number of training iterations of 120000 for Acrobot, and a batch size of 128, a replay size of 10000 and a maximum number of training iterations of 300000 for LunarLander. For each type of uncertainty model, we performed a grid search for the learning rate in the range $[10^{-1}, 10^{-4}]$.

Each uncertainty method has also its own hyperparameters. We show an hyperparameter study in Appendix G.5.4 for the main hyper-parameters of each uncertainty

method. For the MC dropout model, we make a grid-search over the number of samples $n \in \{10, 20, 40, 80\}$ and the drop probability $p \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. In the main experiments, we use $n = 80$ and $p = .2$. For the ensemble model, we make a grid-search over the number of networks $n \in \{10, 20, 40, 80\}$. In the main experiments, we use $n = 80$. For the deep kernel learning model, we make a grid-search over the number of inducing points $n \in \{10, 20, 40, 80\}$, the latent dimension $H \in \{16, 32, 64\}$, the kernel type in RBF, RQ and Matern-$\frac{3}{2}$ Kernel, and an ELBO regularization factor in $\lambda \in [0, 1]$. In the main experiments, we use $n = 80$ inducing points, a latent dimension of $H = 64$, the RQ kernel, and a regularization factor of $\lambda = 0.1$. Further, we observed that adding a batch normalization layer right after the encoder $f_{\boldsymbol{\theta}}$ was stabilizing the training similarly to Charpentier et al. [67]. For the evidential network model based on posterior networks, we make a grid-search over the flow depth $d \in \{8, 16, 32\}$, the latent dimension $H \in \{8, 16, 32\}$. In the main experiments, we use a radial flow with depth $d = 8$ and a latent dimension of $H = 16$. Further, we observed that adding a batch normalization layer right after the encoder $f_{\boldsymbol{\theta}}$ was stabilizing the training similar to Charpentier et al. [67].

We provide the code at the anonymous link [1]. To conduct the experiments, we used Pytorch [341] with BSD license, Pytorch Lightning [125] with Apache 2.0 license and Weight&Biases [35]. Further, we also use GPytorch for to implement the deep kernel model [147].

G.3 Environment Details

We use OpenAI gym environments [54] with MIT license. We design the OOD environments such that they should not be relevant to the original training environment task, and thus being a reasonable failure mode. Further, we design a continuum of perturbed environments going from tasks very similar to the training environment to the tasks very different from the original environment. We distinguish between perturbations on the *state* space, the *action* space, and the *transition* dynamics to follow the MDP structure of the original environment. In contrast, [336, 295] mostly focus on perturbations on the environment parameters. We provide the code for the OOD and perturbed environment at the following anonymous link Footnote 1.

Cartpole [25] In this environment, the goal of the agent is to maintain a pole on a cart straight up. This environment has a discrete action space with 2 possible actions corresponding to apply the a force to the left or the right of the cart. This environment has a continuous state space with dimension 4 corresponds to. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center. The reward is +1 at every time step that the pole stays up. The maximum length of an episode is 200 steps. For the OOD environment, the input states are drawn from a Gaussian distribution with unit variance i.e. $\boldsymbol{s}^{(t),\text{pert}} \sim \mathcal{N}(0, 1)$. For the perturbed environments with perturbation strength ϵ, the action space is perturbed by randomly adding Gaussian noise to the scale of the force applied to the cart (i.e. $f^{\text{pert}} = (1 + x)f$ where $f \in \{-10, 10\}$ is default action force and $x \sim \mathcal{N}(0, \epsilon)$ is the

[1] `https://anonymous.4open.science/r/Aleatoric-Epistemic-Uncertainty-RL-DDB5/README.md`

perturbation), the state space is perturbed by adding Gaussian noise to the observation scale (i.e. $\boldsymbol{s}^{(t),\text{pert}} = (1+x)\boldsymbol{s}^{(t)}$ where $x \sim \mathcal{N}(0,\epsilon)$ is the perturbation), and the transition dynamic is perturbed by adding a uniform noise centered around the true dynamic parameters (i.e. $\nu^{\text{pert}} = (1+x)\nu$ where $x \sim U(-\epsilon,\epsilon)$ where ν is the original environment parameters) such as the gravity, pole length, pole weight.

Acrobot [402, 153] In this environment, the agent control a robot arm with two links and its goal is to move the end of the lower link up to a given height. This environment has a discrete action space with 3 possible actions corresponding to apply a positive torque, a negative torque or nothing. This environment has a continuous state space with dimension 6 corresponding to the 4 joint angles and the 2 angular velocities. The episode ends when the lower link of the robot arm is above a given height. The reward is -1 at every time step that the pole does not reach the expected height. The maximum length of an episode is 500 steps at most. For the OOD environment, the input states are drawn from a Gaussian distribution with unit variance (i.e. $\boldsymbol{s}^{(t),\text{pert}} \sim \mathcal{N}(0,1)$). For the perturbed environments with perturbation strength ϵ, the action space is perturbed by randomly sampling actions with probability $p = \frac{\epsilon}{2}$, the state space is perturbed by adding Gaussian noise to the observation scale (i.e. $\boldsymbol{s}^{(t),\text{pert}} = (1+x)\boldsymbol{s}^{(t)}$ where $x \sim \mathcal{N}(0,\epsilon)$), and the transition dynamic is perturbed by adding a uniform noise centered around the true dynamic parameters (i.e. $\nu^{\text{pert}} = (1+x)\nu$ where $x \sim U(-\epsilon,\epsilon)$ where ν is the original environment parameters) such as the lengths of the links, the masses of the links.

LunarLander [52] In this environment, the agent control a space ship and its goal is to land it on the surface of the moon. This environment has a discrete action space with 4 possible actions corresponding to apply a torque to the left, to the right, downward or nothing. This environment has a continuous state space with dimension 8 corresponding to the space ship coordinates. The reward is correlated with fast landing in the correct area without crashes. The episode ends when the spaceship is landed or crashed. For the OOD environment, the input states are drawn from a Gaussian distribution with unit variance (i.e. $\boldsymbol{s}^{(t),\text{pert}} \sim \mathcal{N}(0,1)$). For the perturbed environments with perturbation strength ϵ, the action space is perturbed by randomly sampling actions with probability $p = \frac{\epsilon}{2}$, the state space is perturbed by adding Gaussian noise to the observation scale (i.e. $\boldsymbol{s}^{(t),\text{pert}} = (1+x)\boldsymbol{s}^{(t)}$ where $x \sim \mathcal{N}(0,\epsilon)$), and the transition dynamic is perturbed by adding a uniform noise centered around the true dynamic parameters (i.e. $\nu^{\text{pert}} = (1+x)\nu$ where $x \sim U(-\epsilon,\epsilon)$ where ν is the original environment parameters) such as the lengths of the links, the masses of the links.

G.4 Metric Details

G.4.1 Training Time

We track the current reward, the epistemic uncertainty and the aleatoric uncertainty at every training step. The epistemic and aleatoric uncertainty are defined by the variance or the entropy of the epistemic and the aleatoric distributions (see Section 10.4). The two uncertainty types are then normalized between $[0,1]$ with min-max normalization to compute the relative epistemic and aleatoric uncertainty on the plots. The normalization

enable an easier comparison of the trend of the uncertainty estimates across methods. For all these experiments, we compute the mean and the standard error of the mean across 5 seeds for all results.

G.4.2 Testing Time

We save 20 model checkpoints at regular interval during the whole training. We evaluate then the 20 checkpointed models at testing time. First, we compute the in-distribution (ID) reward average over 10 episodes on the original training environment. Second, we compute the OOD detection scores by comparing the epistemic uncertainty of the ID and the OOD environment over 10 episodes each with the area under the receiver operating characteristic curve (AUC-ROC) and the area under the precision-recall curve (AUC-PR). Higher scores indicate better OOD detection performances. Third, we compute the averaged reward and epistemic uncertainty on the perturbed environment over 10 episodes. For all these experiments, we compute the mean and the standard error of the mean across 5 seeds for all results. Further, we also sampled 5 random perturbations for each perturbation strength.

G.5 Additional Experiments

G.5.1 Training Time

We show additional results on CartPole, Acrobot and LunarLander in Fig. G.1 to compare the performance of the uncertainty estimates of the four uncertainty methods at training time. The epistemic uncertainty estimates of PostNet decrease during training. Thus, PostNet empirically validate des. 10.3.1. Further, Ensembles and PostNet require a low number of finished episodes on CartPole and LunarLander. This translates for these two envionments into a safer learning with a lower number of restart of the systems.

We show additional results in Fig. G.2 to compare the performance of the sampling-epistemic and the sampling-aleatoric strategies at training time. The sampling-epistemic strategy consistently achieve a better sample efficiency. Thus, Ensemble, DropOut and PostNet empirically satisfy des. 10.3.2. Hence, disentangling aleatoric and epistemic uncertainty can speed learning in a training environment.

G.5.2 Testing Time

We show additional results in Fig. G.3 to compare the generalization and OOD detection performance of the uncertainty estimates of the four uncertainty methods at testing time. The models use the sampling-epistemic or the sampling-aleatoric strategy at both training and testing time. Further, we show other additional results for OOD detection by using the area under the precision-recall (AUC-PR) scores instead of the area under the receiver operating characteristic curve (AUC-ROC) in Fig. G.8. We observe that DKL and PostNet achieve very high OOD detection scores in most settings compared to DropOut and Ensemble. These *empirical* results align with the *theoretical* results stating

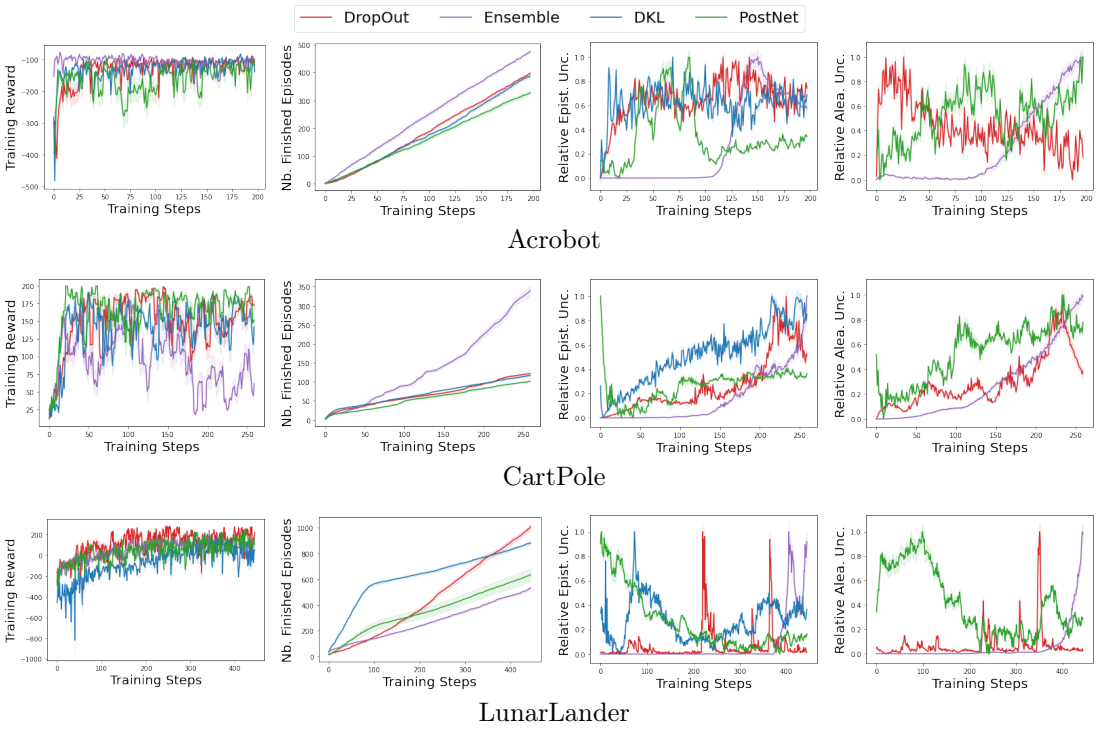


Figure G.1: Comparison of the training performance of the four uncertainty methods using epsilon-greedy strategies. Ideally, an uncertainty aware model should achieve high reward with few samples and with a decreasing epistemic uncertainty.

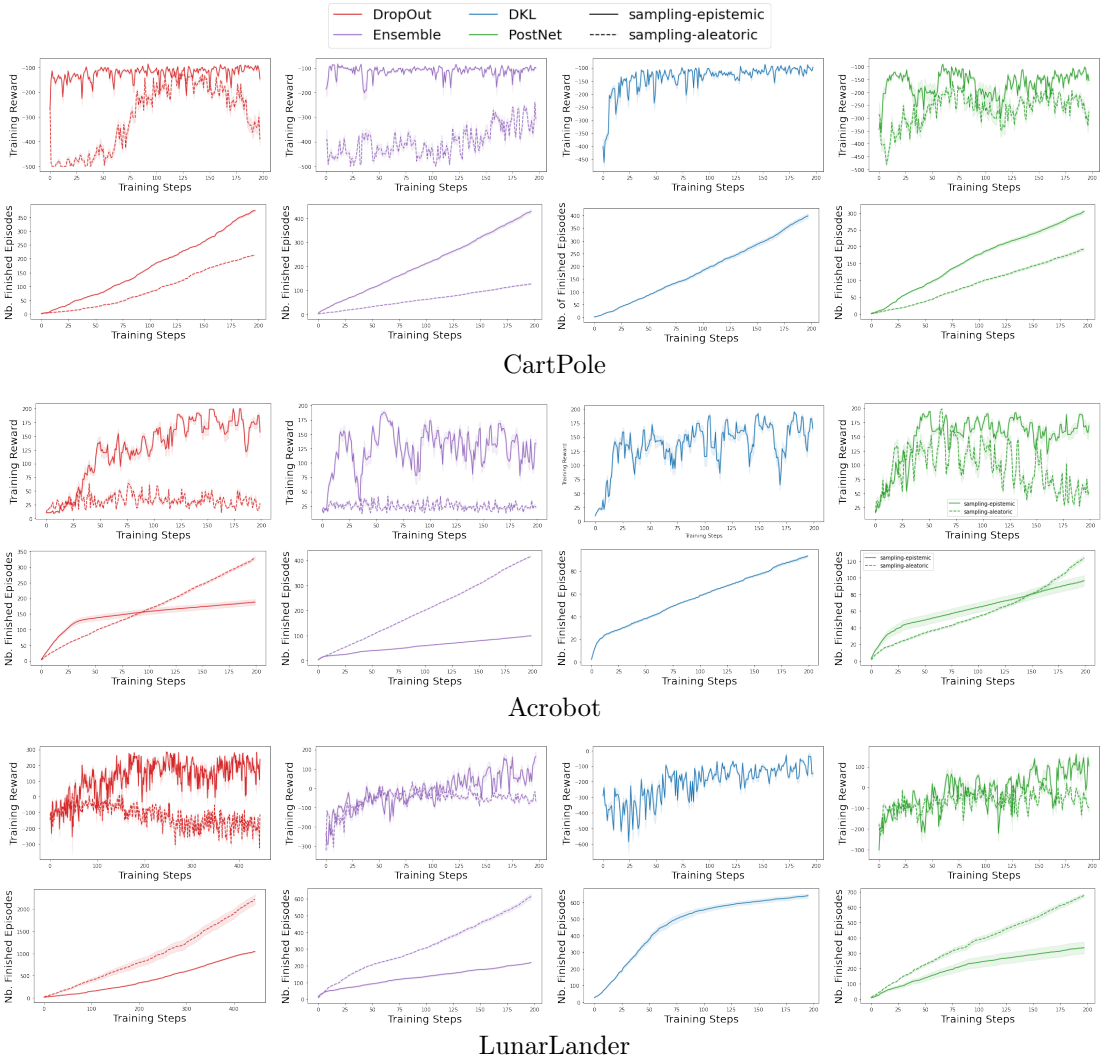


Figure G.2: Comparison of the training performance. The four uncertainty methods use the sampling-aleatoric or the sampling-epistemic at training time. Ideally, an uncertainty aware-model should high reward with few samples.

that DKL and PostNet should assign high uncertainty to states very different from states observed during training. Thus, DKL and PostNet validate des. 10.3.3. In particular, DKL and PostNet can reliably equip DQN with epistemic uncertainty estimates which can be used to flag anomalous OOD states.

We show additional results in Fig. G.4 to compare the performance of the sampling-epistemic and sampling-aleatoric strategies for each uncertainty model. All models use the same epsilon-greedy strategy at training time. We observe that the sampling-epistemic strategy is consistently better than sampling-aleatoric at testing time. The higher generalization capacity of the sampling-epistemic strategy aligns with [159] which recasts the problem of generalization in RL as solving an epistemic POMDP. These empirical results underline the need to disentangle both aleatoric and epistemic uncertainty for high reward performance at testing time.

We show additional results in Figs. G.5 to G.7 to compare the generalization and uncertainty performances of the sampling-epistemic and sampling-aleatoric strategies of each method on perturbed environments with state, action and transition dynamic perturbations. All methods achieve lower reward on environment with stronger perturbations. This is expected since a model cannot generalize to all new environments. The sampling-epistemic strategy achieves significantly better that the sampling-aleatoric strategy. The generalization capacity of the sampling-epistemic strategy aligns again with [159]. Thus, differentiating between aleatoric and epistemic uncertainty can improve generalization. Finally, only DKL and PostNet reliably assign higher epistemic uncertainty to most of the perturbation types. Therefore, DKL and PostNet have a good trade-off between generalization and detection of new perturbed environments.

Video: For a better visualization, we attach supplementary videos showing the landing performance, the reward performance, and the relatice epistemic uncertainty prediction of the PostNet model in the original LunarLander environments and two environments with perturbed states with perturbation strengths equal to 0.5 and 2.0. On the original environment, we observe that the space ship lands correctly with lower epistemic uncertainty after landing. On the perturbed environment with strength 0.5, we observe that the space ship avoids crashing but assigns higher epistemic uncertainty when moving further from the landing zone. Finally, on the perturbed environment with strength 2.0, we observe that the space ship assigns significantly higher epistemic uncertainty especially when approaching the floor before the crash.

G.5.3 Comparison with Vanilla DQN

We show additional results in Fig. G.9 to compare the sample efficiency and the generalization capacity of the uncertainty models with the vanilla DQN. The vanilla DQN is not eqquiped by default with uncertainty estimates. Therefore, it cannot be used for uncertainty tasks like OOD detection. For the sake of comparison, all models use the epsilon-greedy strategy. We observe that the vanilla DQN achieve significantly lower sample efficiency on CartPole. Further, it achieves less stable generalization performance on LunarLander. In contrast, the four uncertainty methods achieve higher generalization performance especially when using the sampling epistemic strategy (see Fig. G.4).

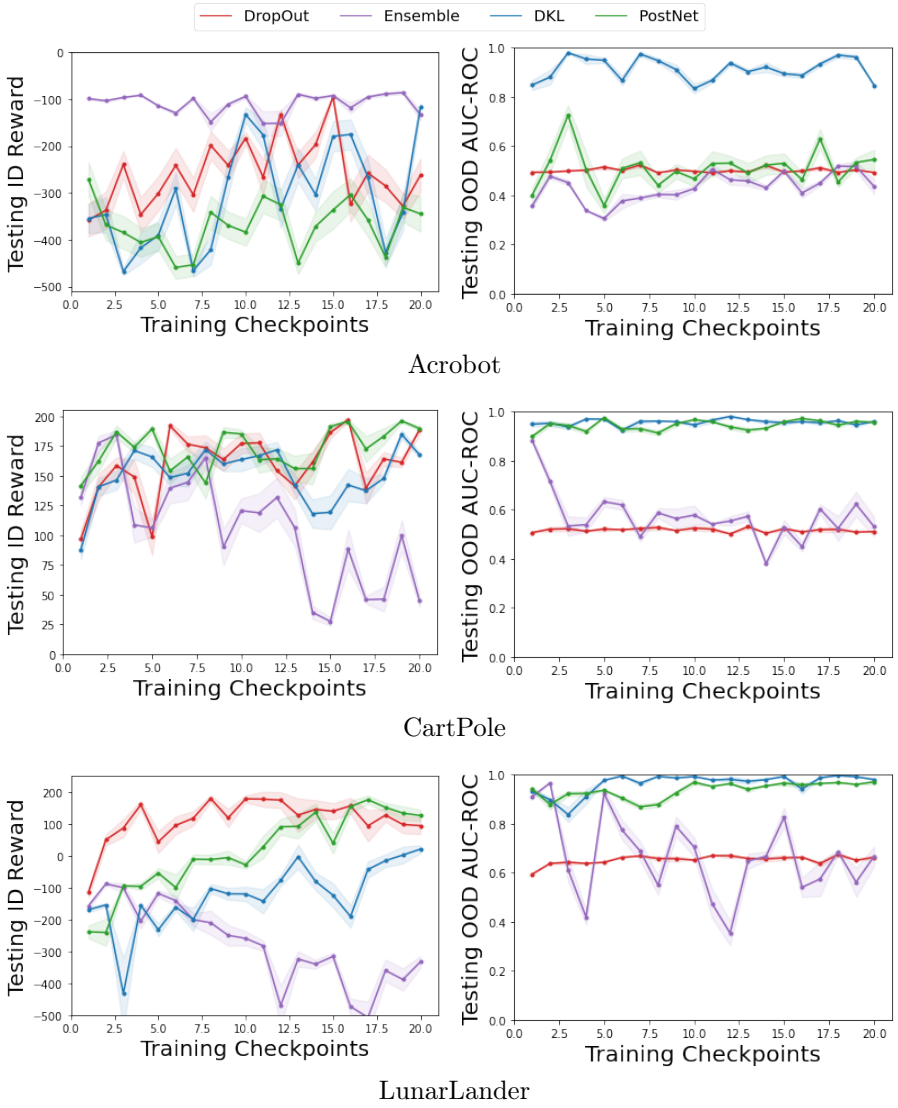


Figure G.3: Comparison of the testing performance of the four uncertainty methods using epsilon-greedy strategies at training and testing time. Ideally, an uncertainty aware model should achieve high reward and high OOD detection scores.

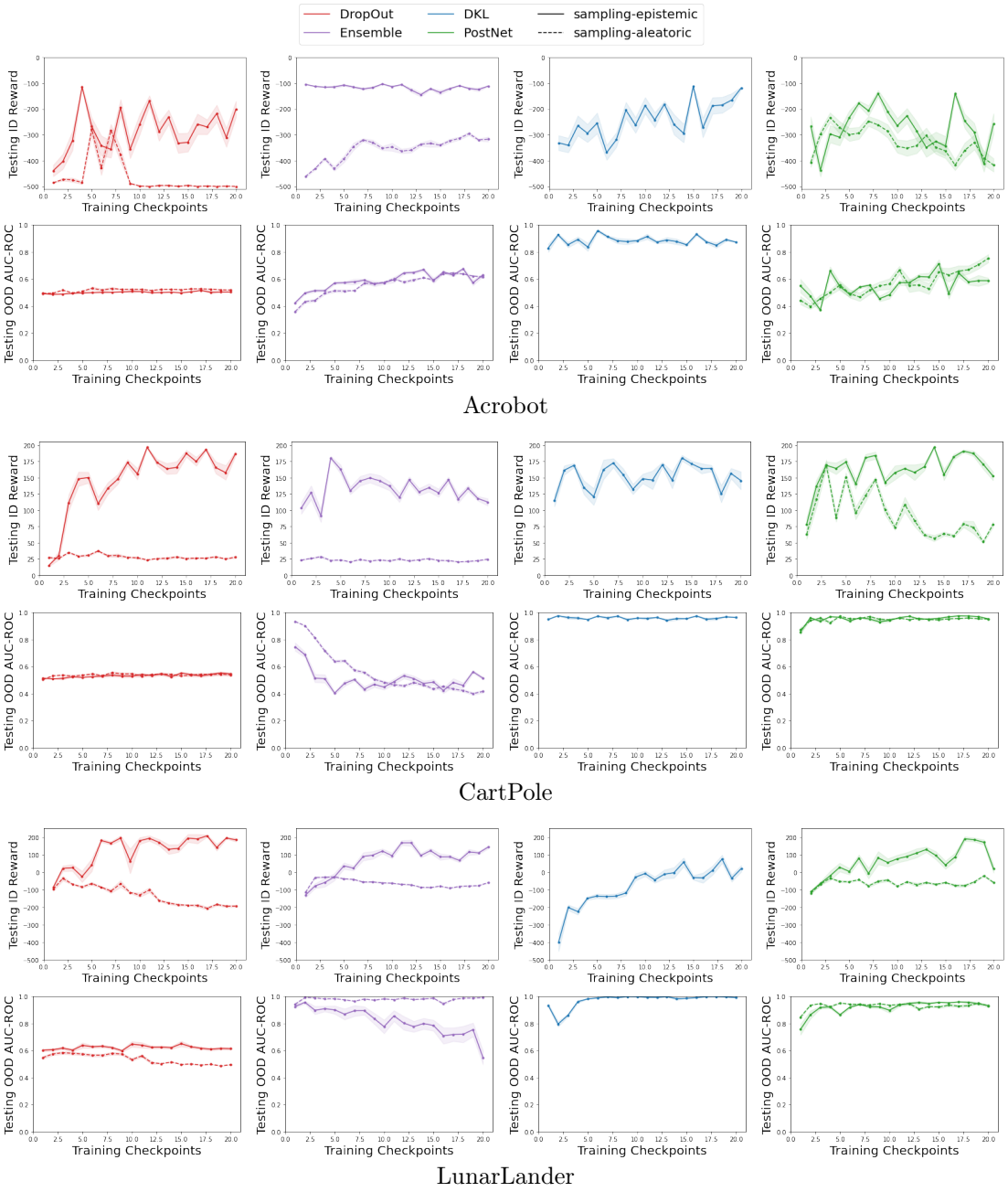


Figure G.4: Comparison of the testing reward and OOD performance. The four uncertainty methods use the sampling-aleatoric or sampling-epistemic strategies at both training and testing time. Ideally, an uncertainty aware model should achieve high testing reward and high OOD AUC-ROC detection score.

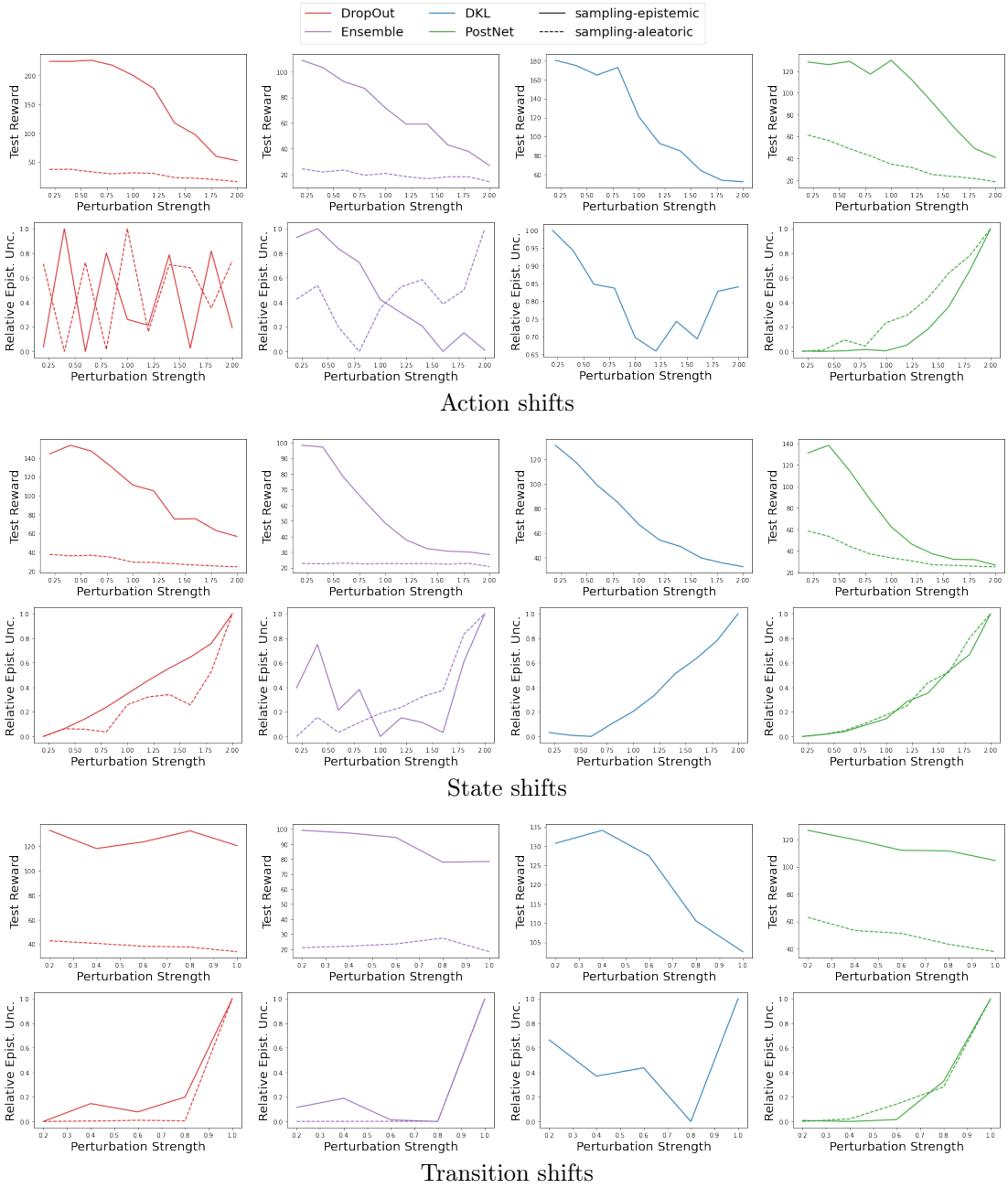


Figure G.5: Comparison of the testing performance and the epistemic uncertainty predictions on CartPole with perturbed actions, states, and transitions. The four uncertainty methods use the epsilon-greedy strategy at training time and the sampling-aleatoric or sampling-epistemic strategy at testing time. Ideally, an uncertainty aware model should maintain high reward while assigning higher epistemic uncertainty on more severe perturbations.

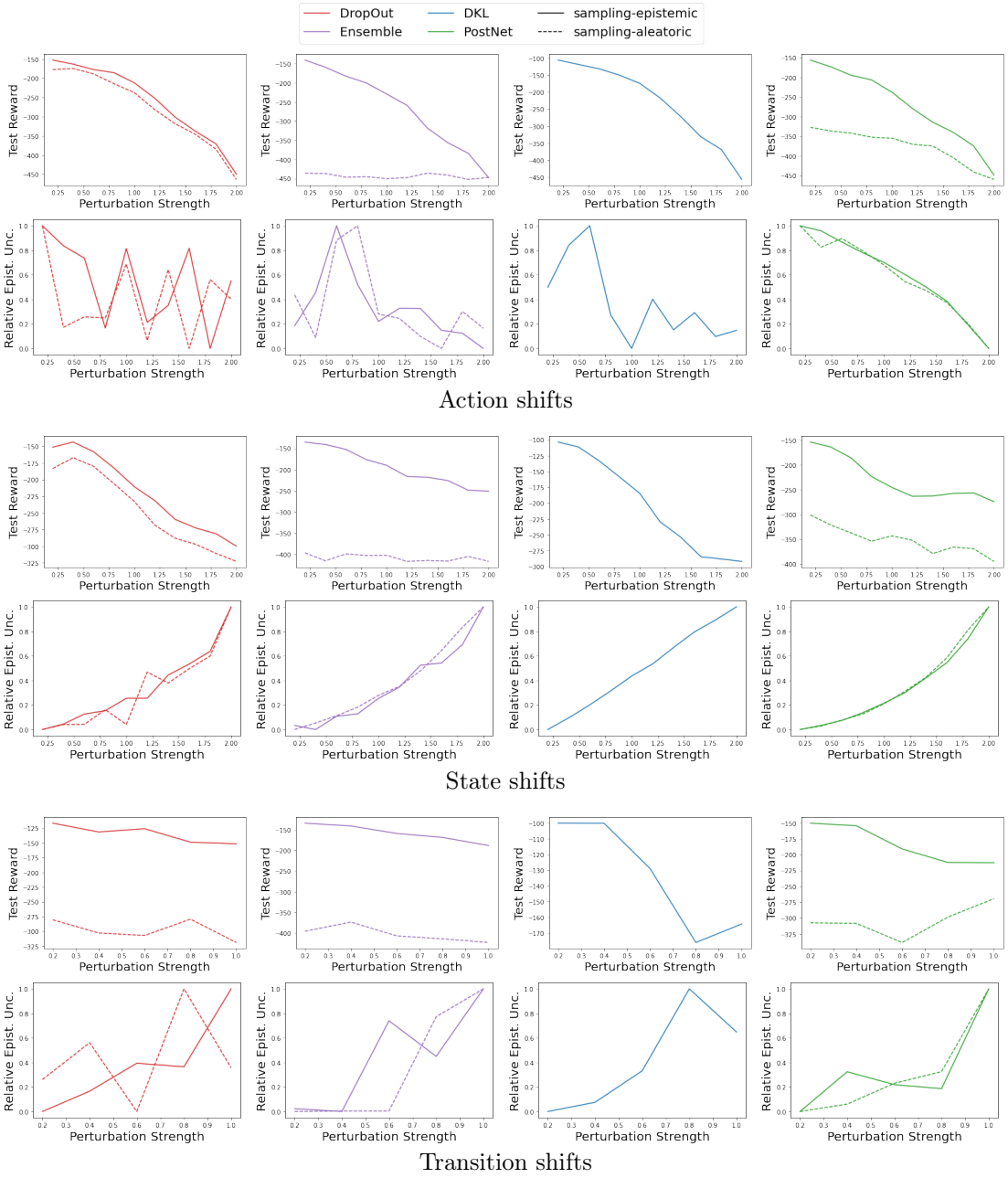


Figure G.6: Comparison of the testing performance and the epistemic uncertainty predictions on Acrobot with perturbed actions, states, and transitions. The four uncertainty methods use the epsilon-greedy strategy at training time and the sampling-aleatoric or sampling-epistemic strategy at testing time. Ideally, an uncertainty aware model should maintain high reward while assigning higher epistemic uncertainty on more severe perturbations.

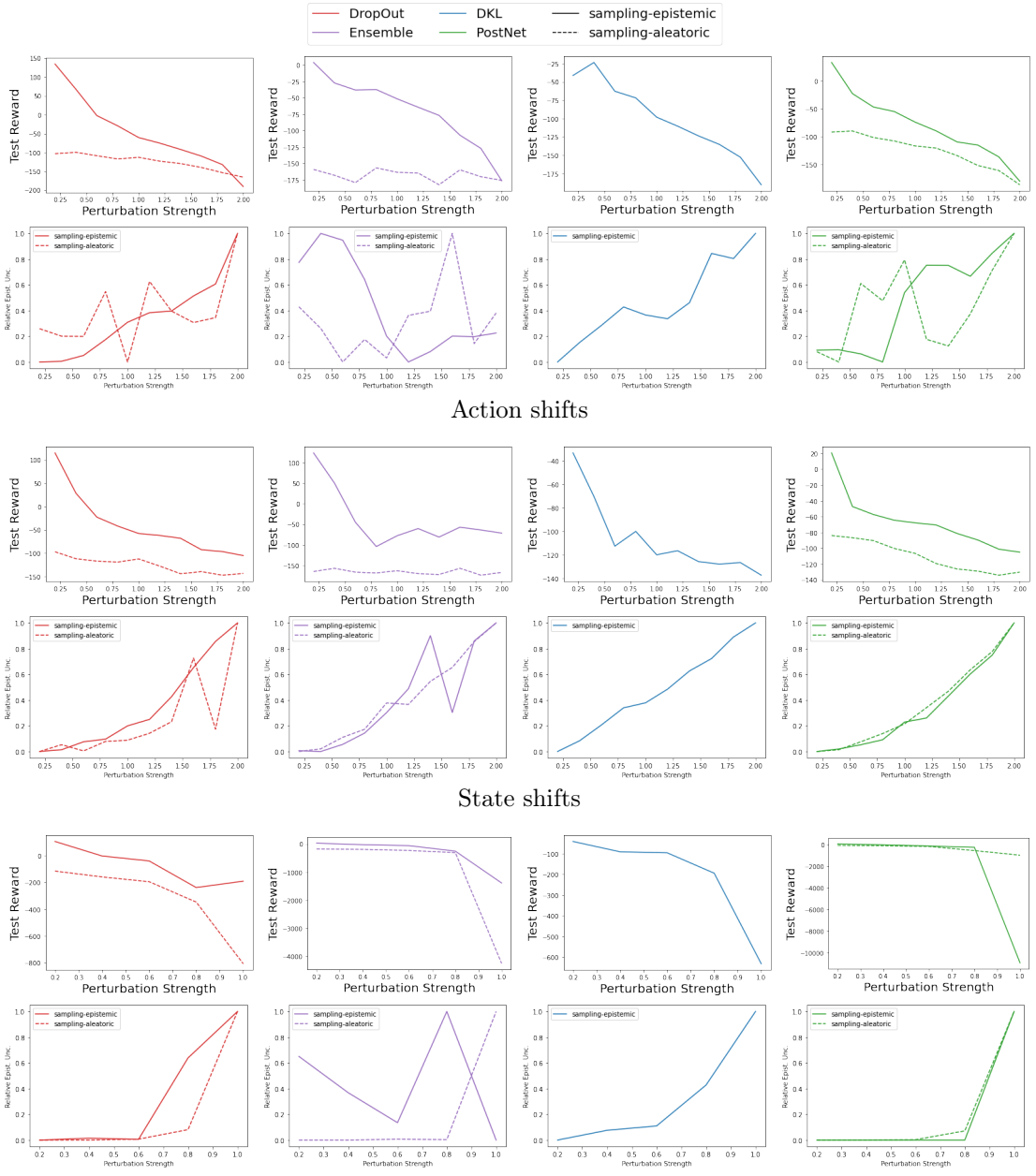


Figure G.7: Comparison of the testing performance and the epistemic uncertainty predictions on LunarLander with perturbed actions, states, and transitions. The four uncertainty methods use the epsilon-greedy strategy at training time and the sampling-aleatoric or sampling-epistemic strategy at testing time. Ideally, an uncertainty aware model should maintain high reward while assigning higher epistemic uncertainty on more severe perturbations.

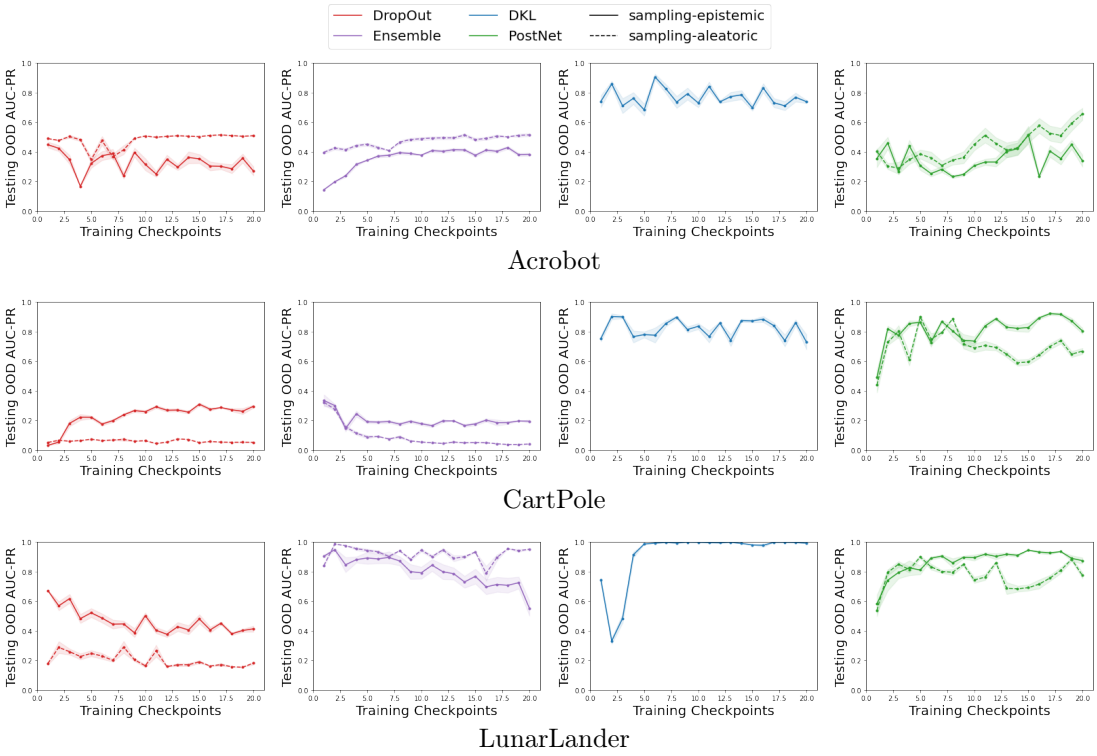


Figure G.8: Comparison of the OOD performance. The four uncertainty methods use the sampling-aleatoric or sampling-epistemic strategies at both training and testing time. Ideally, an uncertainty aware model should achieve high testing reward and high OOD AUC-PR detection score.

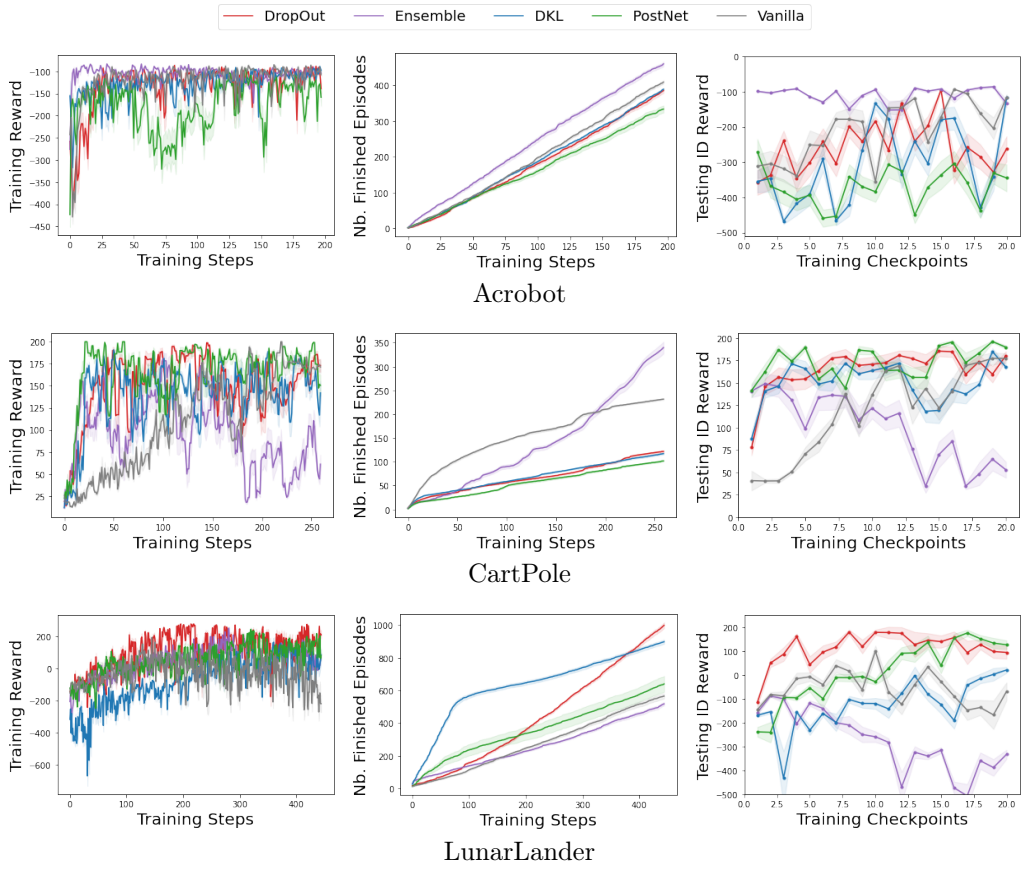


Figure G.9: Comparison of the vanilla DQN with the four uncertainty methods performance. All methods use the epsilon-greedy strategy. The vanilla DQN cannot be evaluated on uncertainty tasks.

These results underline the benefit of predicting and disentangling the aleatoric and the epistemic uncertainty for better sample efficiency and generalization performance.

G.5.4 Hyperparameter Selection

In this section, we present a hyperparameter study for each uncertainty method on the CartPole environment To this end, plot the testing reward and the OOD scores when varying the most important hyper-parameters. we at testing time. We show the hyper-parameter study for dropout when varying the number of samples n and the dropout probability p in Fig. G.10. We observe that a higher number of samples achieves a slightly better OOD detection score. Dropout is pretty insensitive to the dropout probability. We show the hyper-parameter study for ensemble when varying the number of networks n in Fig. G.11. While a higher number of networks is supposed to give higher prediction quality [246], Ensemble looks to give similar results for all number of networks. We

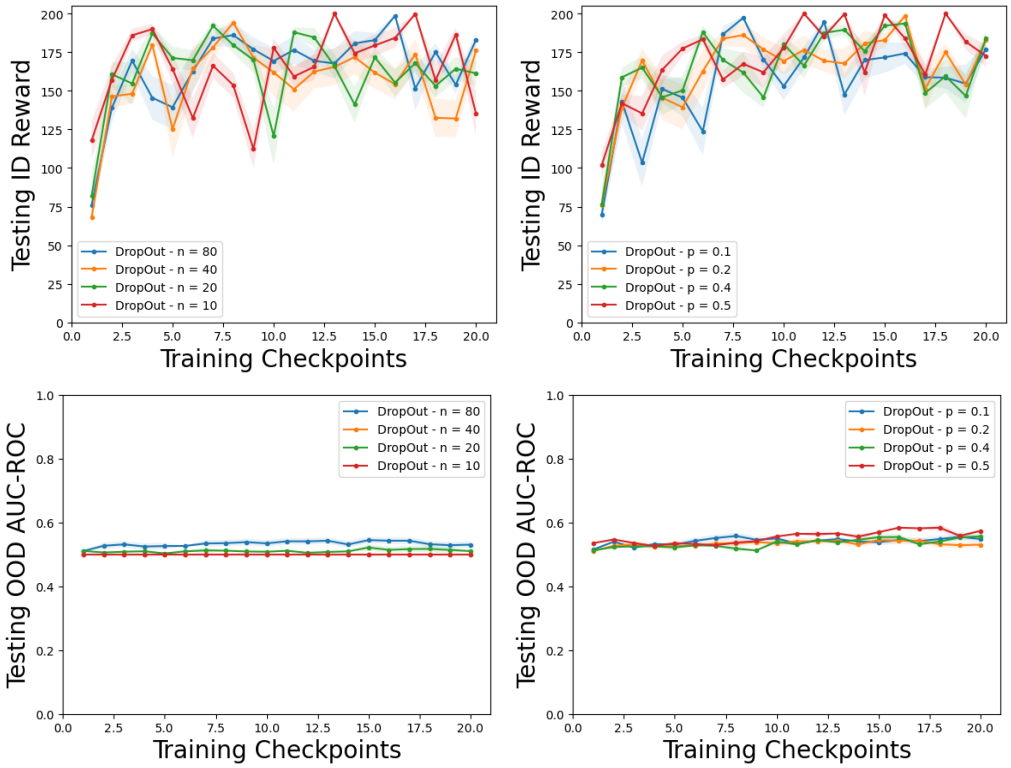


Figure G.10: Hyper-parameter study for DropOut w.r.t. the number of samples n and the dropout probability p. Ideally, an uncertainty aware-model should achieve high reward and high OOD detection scores.

show the hyper-parameter study for DKL when varying the number of inducing points n, the latent dimension H, the kernel type and the batch norm layer in Fig. G.12. The batch norm layer appears to improve the results similarly to [67]. It facilitates the match between the latent positions output by the encoder and the inducing points. The other hyperparameters consistently show good performances. We show the hyperparameter study for PostNet in Fig. G.13. Again, the batch norm layer appears to improve the result stability as observed in [67]. It facilitates the match between the latent positions output by the encoder and non-zero density regions learned by the normalizing flows. The other hyperparameters consistently show good performances.

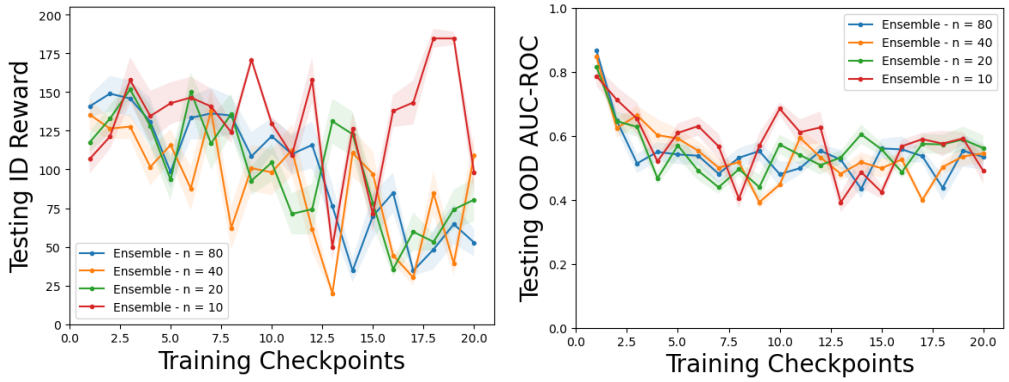


Figure G.11: Hyper-parameter study for Ensemble w.r.t. the number of networks n. Ideally, an uncertainty aware-model should achieve high reward and high OOD detection scores.

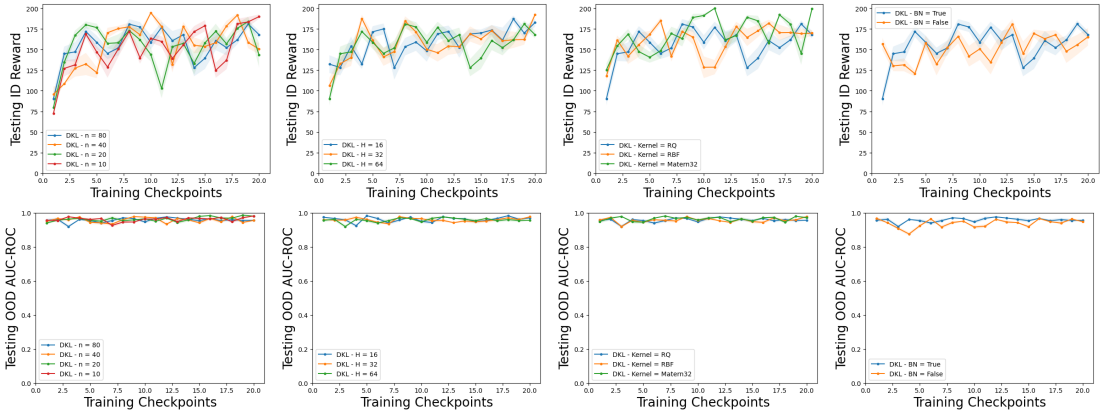


Figure G.12: Hyper-parameter study for DKL w.r.t. the number of inducing points n, the latent dimension H, the kernel type and the batch norm layer. Ideally, an uncertainty aware-model should achieve high reward and high OOD detection scores.

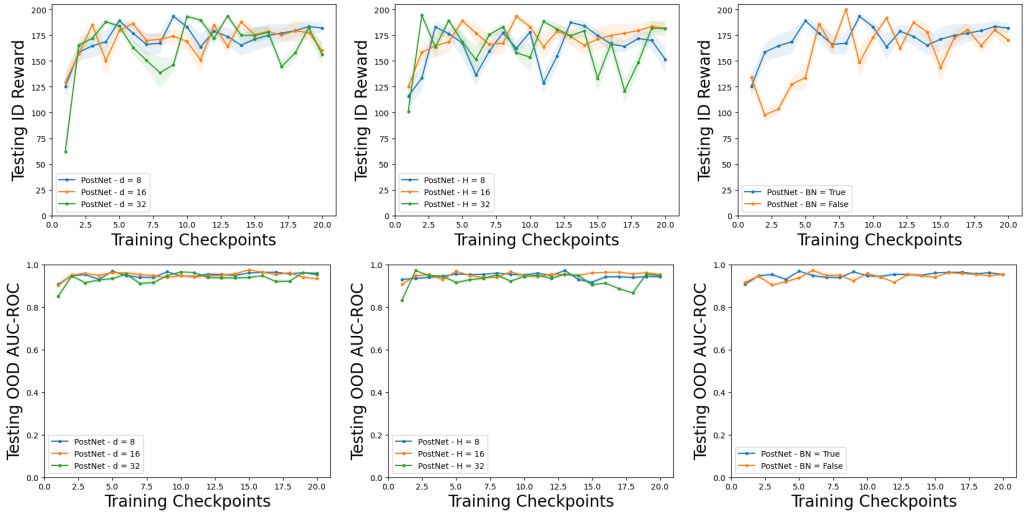


Figure G.13: Hyper-parameter study for PostNet w.r.t. the flow depth d, the latent dimension H and the the batch norm layer. Ideally, an uncertainty aware-model should achieve high reward and high OOD detection scores.