MAXIMILIAN SOELCH

UNCOVERING DYNAMICS

Learning and Amortized Inference for State-Space Models

Technische Universität München

Fakultät für Informatik

UNCOVERING DYNAMICS

Learning and Amortized Inference for State-Space Models

MAXIMILIAN JOHANNES GEORG SÖLCH

Vollständiger Abdruck der von der promotionsführenden Einrichtung Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. Daniel Cremers
Prüfer der Dissertation:	1.	Prof. Dr. Patrick van der Smagt
	2.	apl. Prof. Dr. Georg Groh

Die Dissertation wurde am 11. Mai 2021 bei der Technischen Universität München eingereicht und durch die promotionsführende Einrichtung Fakultät für Informatik am 30. August 2021 angenommen.

Maximilian Soelch
Uncovering Dynamics: Learning and Amortized Inference for State-Space Models (May 2021)

This thesis is typeset using $\LaTeX\,2_\varepsilon$, due to Donald Knuth, Leslie Lamport, and Frank Mittelbach, and based upon the `classicthesis` style and the `ArsClassica` package by André Miede and Lorenzo Pantieri, with minor modifications. Text and bold **mathematical** symbols are set in Palatino, other mathematical expressions in Euler, both courtesy of Hermann Zapf. Headlines are set in Iwona by Janusz M. Nowacki. All figures were created using Ti*k*Z by Till Tantau and `matplotlib` by John D. Hunter.

May this serve as evidence.

ABSTRACT

Understanding the dynamics that drive a system over time is a central pillar of many domains, such as physics or engineering. Uncovering these dynamics from data is thus immediately appealing, and this thesis provides two major contributions towards this goal.

Firstly, we efficiently learn sequential latent-variable models directly from data without supervision. We then highlight how this learning algorithm can be readily transferred to a concrete task such as tracking objects by casting it as an inference problem.

Secondly, we dissect our own as well as several closely related algorithms and models. We discover systematic theoretical and empirical failure scenarios in inference and learning caused by common practically motivated design choices. We provide intuitions for these failure cases and suggest practical strategies to avoid them.

Dynamics in this thesis are represented by state-space models due to their widespread appeal in adjacent disciplines. Learning state-space models becomes practical by amortizing costly state inference with learnable models, in our case neural networks.

Learned state-space models allow reliable predictions about the future based on past observations from the system. This leads from simply uncovering the dynamics to eventually planning, adjusting, and even controlling the system based on posterior beliefs. In our algorithms, these beliefs are represented by sets of samples.

This motivates the third and last main contribution of this thesis, a study of learnable set functions. Again, we empirically show that practical model design choices lead to models that are sensitive to different test scenarios or hyper-parameters, and we provide a variety of model components that alleviate this sensitivity.

ZUSAMMENFASSUNG

Eine der zentralen Säulen vieler Disziplinen, wie bspw. der Physik oder Ingenieurswissenschaften, ist es, die Dynamik zu verstehen, aufgrund der sich ein System über die Zeit entwickelt. Daher ist es von unmittelbarer Bedeutung, diese Dynamiken auf Basis von Daten freizulegen. Diese Arbeit steuert zwei wesentliche Aspekte zu diesem Ziel bei.

Erstens wird gezeigt, dass sequenzielle Modelle mit latenten Variablen effizient und unüberwacht direkt aus Daten gelernt werden können. Zudem wird herausgearbeitet, wie dieser Lernalgorithmus direkt auf konkrete Anwendungen, in diesem Fall das Tracking von Objekten, übertragen werden kann, indem Tracking als Bayes'sche Inferenz formuliert wird.

Zweitens werden die in dieser und verwandten Arbeiten vorgeschlagenen Algorithmen und Modelle einer genauen Untersuchung unterzogen. Dabei wird theoretisch wie empirisch gezeigt, dass übliche anwendungsorientierte Modellannahmen zu systematischen Fehlern in der Inferenz und dadurch beim Lernen führen. Um diese zu vermeiden, werden Intuitionen der Fehlfunktionen sowie praktische Strategien entwickelt.

Dynamiken werden im Rahmen dieser Arbeit aufgrund ihrer breiten Nutzung in angrenzenden Disziplinen durch Zustandsraummodelle repräsentiert. Zustandsraummodelle zu lernen wird dadurch ermöglicht, dass die Kosten für Zustandsinferenz mit Hilfe lernbarer Modelle, in diesem Falle neuronaler Netze, amortisiert werden.

Gelernte Zustandsraummodelle ermöglichen verlässliche Vorhersagen über die Zukunft auf Basis vergangener Beobachtungen des Systems. Dies führt dazu, dass nach erfolgreicher Freilegung der Dynamik schlussendlich Planung, Anpassung und Regelung des Systems gemäß A-posteriori-Wahrscheinlichkeiten möglich werden. Diese Wahrscheinlichkeiten werden in den vorgestellten Algorithmen mittels Mengen von Stichproben dargestellt.

Dadurch motiviert werden als dritter Beitrag dieser Arbeit lernbare Funktionen auf Mengen untersucht. Wie zuvor wird gezeigt, dass anwendungsorientierte Modellannahmen dazu führen, dass die resultierenden Modelle sehr sensitiv auf Änderungen der Testumgebung oder der Hyperparameter reagieren. Es werden Modellbausteine vorgeschlagen und diskutiert, die diese Sensitivität verringern.

PUBLICATIONS

This thesis is based on ideas that have appeared previously in the following publications and working papers:

Karl, Maximilian, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt (2017). "Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=HyTqHL5xg.

Akhundov, Adnan, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt (2019). *Variational Tracking and Prediction with Generative Disentangled State-Space Models*. arXiv: 1910.06205 `[cs, stat]`. URL: http://arxiv.org/abs/1910.06205.

Soelch, Maximilian, Adnan Akhundov, Patrick van der Smagt, and Justin Bayer (2019). "On Deep Set Learning and the Choice of Aggregations." In: *Artificial Neural Networks and Machine Learning - ICANN 2019: Theoretical Neural Computation - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part I*. Ed. by Igor V. Tetko, Vera Kurková, Pavel Karpov, and Fabian J. Theis. Vol. 11727. Lecture Notes in Computer Science. Springer, pp. 444–457. ISBN: 978-3-030-30487-4. DOI: 10.1007/978-3-030-30487-4\\_35.

Bayer, Justin, Maximilian Soelch, Atanas Mirchev, Baris Kayalibay, and Patrick van der Smagt (2021). "Mind the Gap When Conditioning Amortised Inference in Sequential Latent-Variable Models." In: *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net. URL: https://openreview.net/forum?id=a2gqxKDvYys.

The author's contribution to each paper is discussed on the preface page before the respective part.

The author has further contributed to the following papers. These papers will be mentioned in passing as related work where appropriate, but not discussed as a contribution of this thesis.

Soelch, Maximilian, Justin Bayer, Marvin Ludersdorfer, and Patrick van der Smagt (2016). *Variational Inference for On-Line Anomaly Detection in High-Dimensional Time Series*. arXiv: 1602.07109 `[cs, stat]`. URL: http://arxiv.org/abs/1602.07109.

Karl, Maximilian, Maximilian Soelch, Philip Becker-Ehmck, Djalel Benbouzid, Patrick van der Smagt, and Justin Bayer (2017). *Unsupervised Real-Time Control through Variational Empowerment*. arXiv: 1710.05101 `[stat]`. URL: http://arxiv.org/abs/1710.05101.

Mirchev, Atanas, Baris Kayalibay, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer (2019). “Approximate Bayesian Inference in Spatial Environments.” In: *Robotics: Science and Systems XV, University of Freiburg, Freiburg Im Breisgau, Germany, June 22-26, 2019*. Ed. by Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson. DOI: 10.15607/RSS.2019.XV.083.

ACKNOWLEDGMENTS

Special thanks are due to three people without whom my course would have been different:

Christian, you ignited the flame.

Justin, you taught me how to handle it.

Patrick, you always trusted me more than I did. You helped me thrive. Except for about five seconds in *Steinheil*, and I know you enjoyed that!

I would also like to thank my lab mates and students, first at TUM, later with Volkswagen MLRL: Adam, Adnan, Agneta, Alex, Alexandros, Alexej, Atanas, Baris, Benedikt, Botond, Daniela, Djalel, Eileen, Eva, Felix, Grady, Karolina, Markus, Marvin, Max, Nutan, Philip, Rachel, Richard, Sebastian, Simon, and Stephan.

I feel extremely fortunate to have been part of this amazing environment of support, creativity, and companionship.

Wiebke, you could close off that list of lab mates, but that would not do you justice. Thank you.

I would like to thank Steffen Seitz at TU Dresden for his collaboration and hospitality.

A special thanks goes to my family, my parents Hannelore and Wilhelm as well as my brother Alexander, for your continuous support. You did not question me once despite my repeated failure to relay what I was *really* working on all those years.

Sushi, thank you for always being there, no matter what. As this chapter closes, I am looking forward to our next one.

Little Roman, you won the race by a few days. I am glad you are here.

CONTENTS

ACRONYMS

AIR	Attend, Infer, Repeat
ANS	approximate neural smoothing
AVI	amortized variational inference
cdf	cumulative distribution function
CNN	convolutional neural network
DDPAE	decompositional disentangled predictive auto-encoder
DKF	deep Kalman filter
DKS	deep Kalman smoother
DMM	deep Markov model
DN	density network
DSAE	disentangled sequential auto-encoder
DVBF	deep variational Bayes filter
ELBO	evidence lower bound
EM	expectation maximization
ESS	effective sample size
E2C	embed to control
FIVO	filtering variational objective
GMM	Gaussian mixture model
GRU	gated recurrent unit
i. i. d.	independent and identically distributed
IS	importance sampling
IWAE	importance-weighted auto-encoder
KDE	kernel density estimate
KL	Kullback-Leibler divergence
KVAE	Kalman variational auto-encoder
LGS	linear Gaussian system
LSTM	long short-term memory network
LVM	latent-variable model
MC	Monte Carlo
MCO	Monte Carlo objective
MLP	multi-layer perceptron
NASMC	neural adaptive sequential Monte Carlo

NN	neural network
pdf	probability density function
RL	reinforcement learning
RNN	recurrent neural network
SGD	stochastic gradient descent
SIR	sequential importance resampling
SIS	sequential importance sampling
SQAIR	sequential Attend, Infer, Repeat
SRNN	stochastic recurrent neural network
SSM	state-space model
STORN	stochastic recurrent network
SUS	stochastic universal sampling
SVI	stochastic variational inference
UAV	unmanned aerial vehicle
VAE	variational auto-encoder
VI	variational inference
VRNN	variational recurrent neural network
VTSSI	variational tracking and state-space inference

NOTATION

x	a scalar
$\mathbf{x}$	a vector
$\mathbf{X}$	a matrix or tensor
$\mathcal{X}$	a set, typically the space of, e. g., all $\mathbf{x}$
$x_{a:b}$	an ordered sequence $(x_a, x_{a+1}, \dots, x_b)$ with integers a, b; if $b < a$, then $x_{a:b} \equiv \emptyset$ is empty
x_t	the t-th element in the *sequence* $x_{1:T}$
x_i	the i-th element in the *vector* $\mathbf{x}$
$x^{(i)}$	the i-th element in the (unordered) *set* $\left\{x^{(1)}, \dots, x^{(N)}\right\}$
$x_{t,ij}^{(n)}$	the element at row i, column j of matrix $\mathbf{X}_t^{(n)}$ from the sequence $\mathbf{X}_{1:T}^{(n)}$ from the set $\left\{\mathbf{X}_{1:T}^{(n)} \,\middle|\, n = 1, \dots, N\right\}$
$p(x)$ $p(x \mid z)$ $p(x, z)$	marginal, conditional, joint distribution over respective (conditional) random variables; p also denotes the probability density (mass) function; a distinction between random variable x and sample (realization) x is only made where necessary in context
$x \sim p(x)$	the random variable x follows the distribution $p(x)$ *or* the value x was sampled according to distribution $p(x)$
$p_\theta(\cdot)$	a distribution parametrized by a set of parameters θ
$\mathbb{E}_{p(x)}[\cdot]$	the expected value w. r. t. the distribution $p(x)$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	the corresponding probability density function
$\mathcal{U}[a, b]$	the uniform distribution on the interval $[a, b]$, $a < b$

Reserved letters:

x	observations, emissions
z	latent variables, states
u	control inputs
$\mathcal{D}$	a data set
t	an integer index for *some* arbitrary time step
T	an integer index for the *last* time step: the sequence length
$p(\cdot)$	a model (generative) distribution *or* the true distribution
$q(\cdot)$	an approximate (posterior) distribution
$\pi(\cdot)$	a proposal distribution

INTRODUCTION

Neural networks are everything and nothing to this thesis.

Which begs the question: what are neural networks, really? In the recent advent of interest—and hype—around artificial intelligence, neural networks play a central role. From neural networks reaching human capabilities in pattern recognition (Cireşan et al., 2011a,b; Krizhevsky et al., 2012; Schmidhuber, 2017) almost a decade ago, to beating the reigning Go world champion (Silver et al., 2016), to neural language models (Devlin et al., 2019) taking over Google's query processing within a year (Nayak, 2019; Schwartz, 2020), the feats continue to impress.

In this sense, neural networks have pushed the perception of what is possible, replacing and outperforming intricate hand-crafted algorithms by learning from data. And yet, the neural networks found in this thesis are rarely *deep* in the modern sense. Three feed-forward layers often do the trick! Neural networks in the following chapters can often be reduced to a mere footnote.[1]

Two common threads weave through the models and algorithms presented in this thesis.

The first thread is *structured* data. Where at least the early successes of deep learning were based on the availability of large amounts of *independent and identically distributed* (i. i. d.) data, the data sets in this thesis violate independence in one way or the other: with sequential data we explicitly want to understand the dependencies across time. Similarly, the assumption of set-valued inputs, i. e., data sets consisting of sets, the inherent symmetries in the data force conventional neural networks over their limits.

Which leads to the second thread: algorithms define networks. The neural networks found in this thesis are not meant to replace algorithms. Instead, known algorithms—or even theorems—are the solid foundation. Only then do we carefully replace selected components of these algorithms with neural building blocks. The intent is to get the best of both worlds: the algorithm serves as *inductive bias* for the neural network, the neural network *enhances* the algorithm.

These two threads are presented in four parts. Part I recalls the necessary basics to understand the contributions of this thesis. Part II shows how sequential Bayesian posterior algorithms can be turned into unsupervised learning algorithms for sequential latent-variable models in general and state-space models in particular; these algorithms are then applied to learn a neural state-space model for tracking. Part III shines a sobering light on some flaws in the early designs of inference

1 Maybe they should rather be called *stacked function approximators* (Clark, 2017)?

models for sequential latent-variable models—ironically, these flaws are a consequence of leaning into neural networks too much—and presents possible remedies, including a learning algorithm based on particle smoothing. Lastly, part IV investigates how neural models can be applied to set-valued data, where a mathematical categorization of *all* set functions is leveraged.

Without neural networks, this thesis would not be. Without neural networks, the algorithms could still be.

Part I

BACKGROUND

This part serves as a recap of the background knowledge required for the remainder of this thesis: basics of learning and sampling distributions in chapter 1, *latent-variable models* (LVMs) and *variational inference* (VI) in chapter 2, and sequential LVMs in chapter 3.

Supplementary material is collected in appendix A.

Much of the material was collected and honed as part of tutorials by the author for his peers and students.

It is largely based on textbooks (MacKay, 2002; Bishop, 2007; K. P. Murphy, 2012; Owen, 2013), particularly the most excellent summary on sequential Bayesian inference by Sarkka (2013). Other resources include Babb (2015), Doersch (2016), Dykeman (2016), and Blei et al. (2017). Original publications are referenced within the text.

The author has attributed original sources to maximal extent allowed by the eclectic nature of this part.

1 LEARNING AND SAMPLING

1.1 LEARNING DISTRIBUTIONS

A common thread throughout this thesis is learning parametrized probabilistic models from data. This section covers the basic techniques for implementing and learning such models.

1.1.1 Learning and Optimization

A central vehicle of this thesis are parametric models. In this space of models defined by all possible parameter sets θ, our goal is to infer *ideal* parameters for a certain scenario. This is formalized as an optimization problem

$$\arg\min_{\theta} \mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x}, \theta)]. \tag{1.1}$$

The scenario of interest is codified by the distribution $p(\mathbf{x})$. The ideal parameters are codified by the objective function f.

We refer to the process of inferring the ideal parameters as *learning* or *fitting*, where the former is usually understood as learning a model and the latter as fitting the parameters, two sides of the same coin. The frequently-used synonymous term *inference* is in this thesis henceforth exclusively reserved for *Bayesian* inference to avoid confusion between these concepts.

The distribution $p(\mathbf{x})$ is most often represented by a finite data set $\mathcal{D}$, which is typically thought of as *independent and identically distributed* (i. i. d.) samples of $p(\mathbf{x})$, i. e.,

$$\mathcal{D} = \left\{\mathbf{x}^{(i)} \,\middle|\, i = 1, \dots, N\right\}, \qquad \mathbf{x}^{(i)} \sim p(\mathbf{x}). \tag{1.2}$$

Since $p(\mathbf{x})$ is only available through the data set, we approximate eq. (1.1) as

$$\arg\min_{\theta} \sum_{i=1}^{N} f\left(\mathbf{x}^{(i)}, \theta\right). \tag{1.3}$$

This is called *empirical risk minimization*.

Our means of tackling the learning problem, eq. (1.1), is to use optimization techniques—most often first-order methods based on *stochastic gradient descent* (SGD)—on the best available approximation, the empirical risk minimization, eq. (1.3).

Neural Networks and Optimization

The models and algorithms presented in this thesis are in most cases implemented by neural networks. Yet, in almost all cases they are entirely abstracted, e. g., by *density networks* (DNs), section 1.1.4. That is, algorithms are not tied to specific neural architectures, or even neural networks to begin with, and can be discussed without considering them at all.

In this light, we assume a basic familiarity with neural network architectures, point the reader to introductory textbooks, e. g., Goodfellow et al. (2016), and refrain from further discussion. Insofar as specifics are relevant, they will be discussed in the respective sections.

In terms of nomenclature, *neural network* (NN) will serve as a general umbrella term for a neural architecture. We will call the vanilla feed-forward architecture—consecutive layers of affine transformations followed by point-wise nonlinearities—*multi-layer perceptrons* (MLPs). Similarly, *recurrent neural network* (RNN) is an umbrella term for vanilla RNNs as well as special variants like bidirectional RNNs (Schuster and Paliwal, 1997), *long short-term memory networks* (LSTMs; Hochreiter and Schmidhuber, 1997), or *gated recurrent units* (GRUs; Cho et al., 2014).

In a similar vein, we assume basic familiarity with *stochastic gradient descent* (SGD; Bottou, 2010). In particular, this includes the backpropagation algorithm and modern adaptive, moment-based first-order optimization algorithms such as Adam (Kingma and Ba, 2015).

In practice, these considerations are largely abstracted by software packages for automatic differentiation (Abadi et al., 2015; Al-Rfou et al., 2016; Paszke et al., 2019).

1.1.2 Monte Carlo Integration

In probabilistic learning problems, we often encounter expected values of some function g w. r. t. some distribution $p(\mathbf{z})$,

$$\mathbb{E}_{p(\mathbf{z})}[g(\mathbf{z})] = \int p(\mathbf{z}) g(\mathbf{z}) \, d\mathbf{z}. \tag{1.4}$$

Most objective functions in this thesis include one or several such expectations. Typically, these are not analytically tractable. Instead, we will use *Monte Carlo* (MC) integration to get estimates of the respective quantities.

The core idea behind MC integration is to approximate expected values by averaging $N \in \mathbb{N}$ sample evaluations,

$$\mathbb{E}_{p(\mathbf{z})}[g(\mathbf{z})] \approx \frac{1}{N} \sum_{i=1}^{N} g\left(\mathbf{z}^{(i)}\right), \quad \mathbf{z}^{(i)} \sim p(\mathbf{z}). \tag{1.5}$$

Here, g is some arbitrary function. We call $p(\mathbf{z})$ the *target distribution*. This estimate is unbiased.

1.1.3 Graphical Models

Often, we model systems where various quantities $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and their dependencies are at least partially known. For any such set, the joint distribution $p(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ factorizes as

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid \mathbf{x}_{1:i-1}), \tag{1.6}$$

but any permutation of the ordering of random variables is mathematically valid.

Graphical models are a useful tool to model structure and outside knowledge about the relation between the quantities. A graphical model is a joint probability distribution over a set of random vectors $\mathcal{V} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with an associated directed, acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The graph represents conditional dependencies in the sense that

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid \mathcal{P}(\mathbf{x}_i)), \tag{1.7}$$

where $\mathcal{P}(\mathbf{x}_i)$ denotes the set of variables corresponding to parent nodes of $\mathbf{x}_i$ in the graph. This is well-defined because the graph is acyclic. Note that the graph is not unique. In particular, by eq. (1.6) any fully-connected graph trivially yields a correct factorization of the joint according to eq. (1.7). We implicitly assume the graph is minimal—we spare a detailed discussion of *minimal* here and point out that for our purposes the graph is usually the starting point.

Beyond being a useful tool to conceptualize models, the most salient feature of graphical models within this thesis is that certain graph configurations translate to (conditional) independence between the respective nodes. This will be useful when discussing sequential LVMs in chapter 3 and parts II and III.

1.1.4 Density Networks

Graphical models are largely defined in terms of *conditional* distributions $p(\mathbf{x} \mid \mathbf{z})$. Here, we present a simple but common strategy to implement such distributions with deterministic functions (or function approximations) such as NNs.

First, we choose some parametric family of distributions, i. e., a set of distributions where each distribution can be described by a set of fixed distribution parameters ϑ. The most commonly used example in this thesis are Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, parametrized by mean $\boldsymbol{\mu}$ and covariance (matrix) $\boldsymbol{\Sigma}$ so that $\vartheta = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$.

Then, a function f_θ, where θ is the set of all learnable function parameters, e. g., weight matrices and biases for NNs, can now be leveraged to implement $p(\mathbf{x} \mid \mathbf{z})$ by mapping the input $\mathbf{z}$ to a set of valid distribution parameters, e. g.,

$$(\mu, \mathbf{\Sigma}) = f_\theta(\mathbf{z}) \quad \rightsquigarrow \quad p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mu, \mathbf{\Sigma}). \tag{1.8}$$

Depending on the application, some or all *distribution* parameters are constants in $\mathbf{z}$, and these constants may themselves be *learnable* parameters. Examples are priors (which have no input), or likelihood models with a shared variance modeling sensor noise.

In notation, all of these scenarios are often shortened to $p_\theta(\mathbf{x} \mid \mathbf{z})$ (or even just $p(\mathbf{x} \mid \mathbf{z})$) with no explicit mention of f_θ or ϑ. In fact, θ will often include all learnable parameters. This conflates the notion of the learnable *function* parameters θ and the functionally dependent, non-learnable *distribution* parameters $\vartheta = f_\theta(\mathbf{z})$.[1]

For mixtures of Gaussians, this concept was suggested by Bishop (1994) and dubbed mixture density networks. Following this, we refer to this slightly wider concept as *density networks* (DNs) in this thesis.[2]

1.1.5 Reparametrization

With density networks at hand, we need to be able to compute gradients for learning. As mentioned in section 1.1.2, we often encounter expected values of type

$$\mathbb{E}_{p_\theta(\mathbf{z})}[g(\mathbf{z})] \tag{1.9}$$

with some function g in our objectives. For SGD on such objectives we need to compute gradients

$$\nabla_\theta \mathbb{E}_{p_\theta(\mathbf{z})}[g(\mathbf{z})]. \tag{1.10}$$

Since the expectation is typically not analytically solvable, analytic gradients are equally unavailable. This is particularly true when $p_\theta(\mathbf{z})$ is a sophisticated density network.

MC approximation with a parametrized distribution on the other hand faces two challenges: firstly, in general gradients w. r. t. distribution parameters and expectations are not interchangeable:

$$\nabla_\theta \mathbb{E}_{p_\theta(\mathbf{z})}[g(\mathbf{z})] = \nabla_\theta \int p_\theta(\mathbf{z}) g(\mathbf{z}) \, d\mathbf{z} \tag{1.11}$$

$$= \int \nabla_\theta p_\theta(\mathbf{z}) g(\mathbf{z}) \, d\mathbf{z} + \underbrace{\int p_\theta(\mathbf{z}) \, \nabla_\theta g(\mathbf{z}) \, d\mathbf{z}}_{= \mathbb{E}_{p_\theta(\mathbf{z})}[\nabla_\theta \, g(\mathbf{z})]} \tag{1.12}$$

$$\implies \nabla_\theta \mathbb{E}_{p_\theta(\mathbf{z})}[g(\mathbf{z})] \neq \mathbb{E}_{p_\theta(\mathbf{z})}[\nabla_\theta g(\mathbf{z})]. \tag{1.13}$$

1 We will use the terms *weights* and *parameters* for learnable function parameters interchangeably, even outside a neural context.

2 The author acknowledges that neither is the output of a density network a density but a distribution or its parameters nor are we restricted to (neural) networks. With this naming convention we choose consistency with the literature over precision.

Secondly, a sample $\mathbf{z}$ depends on θ but $\nabla_\theta g(\mathbf{z})$ is generally not well-defined.

The key is to require the distribution $p_\theta(\mathbf{z})$ to permit *reparametrization*. A distribution is reparametrizable if its sampling process can be rewritten as a deterministic function r of the distribution parameters ϑ and a random sample $\boldsymbol{\epsilon}$ from an arbitrary base distribution $p(\boldsymbol{\epsilon})$ independent of ϑ, i. e.,

$$\mathbf{z} = r(\boldsymbol{\epsilon}, \vartheta), \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \quad \Longrightarrow \quad \mathbf{z} \sim p_\vartheta(\mathbf{z}). \tag{1.14}$$

Arguably the most frequently used example[3] is the Gaussian distribution $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$: for distribution parameters $\vartheta = \left\{\boldsymbol{\mu}, \boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^\top\right\}$, where $\mathbf{L}$ is a lower triangular matrix, e. g., from a Cholesky decomposition, and the standard Gaussian base distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, one can sample via

$$\mathbf{z} = r(\boldsymbol{\epsilon}, \boldsymbol{\mu}, \mathbf{L}) = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon} \implies \mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{1.15}$$

Such reparametrization makes gradients $\nabla_\vartheta g(\mathbf{z})$ well-defined since $g(\mathbf{z}) = g(r(\boldsymbol{\epsilon}, \vartheta))$ is a deterministic function in ϑ. This is particularly useful for SGD on MC estimates. Reparametrization allows rewriting expectations and thus gradients:

$$\mathbb{E}_{p_\vartheta(\mathbf{z})}[g(\mathbf{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[g(r(\boldsymbol{\epsilon}, \vartheta))] \tag{1.16}$$

$$\implies \nabla_\vartheta \mathbb{E}_{p_\vartheta(\mathbf{z})}[g(\mathbf{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_\vartheta g(r(\boldsymbol{\epsilon}, \vartheta))]. \tag{1.17}$$

The right-hand side can now be estimated without bias by MC integration. The extension to conditional distributions with functionally dependent distribution parameters $\vartheta = f_\theta(\mathbf{z})$ as found in density networks follows from standard rules of multivariate calculus.

1.2 SAMPLING DISTRIBUTIONS

1.2.1 Importance Sampling

In many interesting cases, the target distribution is intractable, cf. Bayesian posteriors in section 3.2. In such cases, even samples are hard to obtain, rendering Monte Carlo integration impossible. Here, *importance sampling* (IS) can be of help. The basic idea is to resort to a *proposal distribution* $\pi(\mathbf{z})$ that can be sampled easily and rephrase

$$\mathbb{E}_{p(\mathbf{z})}[g(\mathbf{z})] = \int p(\mathbf{z}) \frac{\pi(\mathbf{z})}{\pi(\mathbf{z})} g(\mathbf{z})\, d\mathbf{z} = \mathbb{E}_{\pi(\mathbf{z})}\left[\frac{p(\mathbf{z})}{\pi(\mathbf{z})} g(\mathbf{z})\right]. \tag{1.18}$$

Provided we can evaluate the *probability density functions* (pdfs) $p(\mathbf{z})$ and $\pi(\mathbf{z})$, eq. (1.18) is now a special case of eq. (1.4), with new objective function $\hat{g} = p \cdot g / \pi$, and can be estimated without bias as in eq. (1.5).

3 A helpful overview of other examples is due to Mohamed (2015).

We can even use importance sampling when p can only be evaluated up to a normalizing constant $Z > 0$, i. e.,

$$p(\mathbf{z}) = \frac{r(\mathbf{z})}{Z}. \tag{1.19}$$

The unknown normalizing constant can also be estimated by means of importance sampling:

$$Z = \int r(\mathbf{z})\, d\mathbf{z} = \mathbb{E}_{\pi(\mathbf{z})}\left[\frac{r(\mathbf{z})}{\pi(\mathbf{z})}\right]. \tag{1.20}$$

The ratio of unnormalized target pdf and proposal pdf is the (unnormalized) weight function

$$\hat{w}(\mathbf{z}) \equiv \frac{r(\mathbf{z})}{\pi(\mathbf{z})} \geqslant 0. \tag{1.21}$$

The notion of weights gives rise to the alternative interpretation of eq. (1.18) as *weighted samples*: for a given set of samples $\{\mathbf{z}^{(i)}\}$, compute their unnormalized weights

$$\hat{w}^{(i)} = \hat{w}\left(\mathbf{z}^{(i)}\right), \tag{1.22}$$

and then *self-normalize* to get the *normalized* weights

$$w^{(i)} = \frac{\hat{w}^{(i)}}{\sum_{j=1}^{N} \hat{w}^{(j)}} \in [0, 1] \qquad \left[\implies \sum_{i=1}^{N} w^{(i)} = 1\right]. \tag{1.23}$$

Despite being sampled from the proposal, the set of pairs of normalized weights and samples

$$\left\{\left(w^{(i)}, \mathbf{z}^{(i)}\right) \,\middle|\, \mathbf{z}^{(i)} \sim \pi(\mathbf{z})\right\}_{i=1,\dots,N} \tag{1.24}$$

can be viewed as a set of *weighted* samples from the target distribution $p(\mathbf{z})$. Starting from eq. (1.18), one can show

$$\begin{aligned}
&\mathbb{E}_{p(\mathbf{z})}[g(\mathbf{z})] && (1.25)\\
=&\mathbb{E}_{\pi(\mathbf{z})}\left[\frac{p(\mathbf{z})}{\pi(\mathbf{z})} g(\mathbf{z})\right] = \frac{1}{Z}\mathbb{E}_{\pi(\mathbf{z})}\left[\frac{r(\mathbf{z})}{\pi(\mathbf{z})} g(\mathbf{z})\right] && (1.26)\\
=&\frac{\mathbb{E}_{\pi(\mathbf{z})}\left[\frac{r(\mathbf{z})}{\pi(\mathbf{z})} g(\mathbf{z})\right]}{\mathbb{E}_{\pi(\mathbf{z})}\left[\frac{r(\mathbf{z})}{\pi(\mathbf{z})}\right]} \approx \frac{\frac{1}{N}\sum_{i=1}^{N} \hat{w}^{(i)} g\left(\mathbf{z}^{(i)}\right)}{\frac{1}{N}\sum_{j=1}^{N} \hat{w}^{(j)}} && (1.27)\\
=&\sum_{i=1}^{N} w^{(i)} g\left(\mathbf{z}^{(i)}\right). && (1.28)
\end{aligned}$$

The approximation in eq. (1.27) is based on approximating numerator and denominator independently by MC integration.

The vanilla MC integration by averaging as in eq. (1.5) has been replaced by a *weighted* average in eq. (1.28).

Remarks on Importance Sampling

The further loosening of assumptions on the target distribution comes at a cost: the estimator in eq. (1.28) is only *asymptotically* unbiased. This is due to eq. (1.27), where we estimated numerator and denominator separately with unbiased importance sampling, but Jensen's inequality tells us that

$$\mathbb{E}\left[\frac{1}{h(\mathbf{z})}\right] \geqslant \frac{1}{\mathbb{E}[h(\mathbf{z})]}, \tag{1.29}$$

i. e., inverting the unbiased denominator estimate introduces bias.

An important observation is that the weighted samples, eq. (1.24), are independent of the objective function g. This is another reason why they can be viewed as representing the target distribution p. In fact, the act of producing *representative* samples can be completely decoupled from an objective function. We will revisit this concept with *sequential importance sampling* (SIS) and *sequential importance resampling* (SIR), e. g., in section 3.3.2.2 when we introduce particle filters.

Technically, importance sampling works for any combination of target and proposal distribution, provided the support of the target distribution is a subset of the support of the proposal and the number of samples N grows sufficiently large. In practice however, for an IS estimator to be useful the proposal should be close to the target to supply representative samples.

To understand this, consider the random variable $\mathbf{y} \sim p(\mathbf{y})$, which has some variance σ^2. The average of N i. i. d. copies $\mathbf{y}^{(i)}$ has variance

$$\mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{y}^{(i)}\right] = \frac{\sigma^2}{N}. \tag{1.30}$$

On the other hand, a (self-normalized) weighted average[4] has variance

$$\mathrm{Var}\left[\frac{\sum_{i=1}^{N}\hat{w}^{(i)}\mathbf{y}^{(i)}}{\sum_{j=1}^{N}\hat{w}^{(j)}}\right] = \frac{\sum_{i=1}^{N}\left[\hat{w}^{(i)}\right]^2\sigma^2}{\left[\sum_{j=1}^{N}\hat{w}^{(j)}\right]^2} = \sum_{i=1}^{N}\left[w^{(i)}\right]^2\sigma^2. \tag{1.31}$$

Equating both right-hand sides, we see that the variance of a weighted average is equivalent to that of an unweighted average computed with

$$P_{\mathrm{ESS}} \equiv \frac{1}{\sum_{i=1}^{N}\left[w^{(i)}\right]^2} \in [1, N] \tag{1.32}$$

samples—typically referred to as the *effective sample size* (ESS).

For our purposes, we can set

$$\mathbf{y} = g(\mathbf{z}). \tag{1.33}$$

4 This weighted average assumes fixed, independent weights.

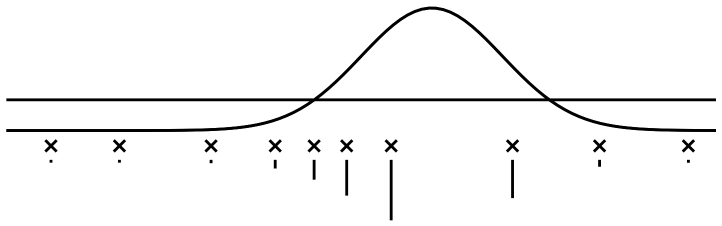


Figure 1.1: Example of *importance sampling* (IS): the target is a Gaussian distribution $\mathcal{N}(0.6, 0.1)$, the proposal is the uniform distribution $\mathcal{U}[0, 1]$. The *probability density functions* (pdfs) on the unit interval are shown. Ten proposal samples are drawn i. i. d., with self-normalized weights depicted by bars below the samples. The *effective sample size* (ESS) is roughly 3.9—less than half the sample size since the proposal disproportionally covers the tails of the target. N. B.: for illustration purposes the proposal does not cover the full support of the target, an IS estimate would have a small bias.

That is, when making an IS estimate with N samples from the proposal, the ESS estimates how many samples from the target distribution would yield a hypothetical MC estimate with equally high variance.

Observing that the effective sample size is at most N, this implies that an IS estimator has higher variance compared to the corresponding, usually intractable MC estimator. The extreme cases highlight this notion: when the proposal distribution is the target distribution, then $w^{(i)} = 1/N$ and the ESS is indeed N; when the proposal distribution is so off that a single sample dominates, i. e., the weights are one-hot, the ESS is indeed 1.

An illustrative example for IS and ESS is given in fig. 1.1.

1.2.2 Ancestral Sampling

In this thesis we are often interested in sampling joint distributions of variables. A general strategy for sampling such joint distributions is *ancestral sampling*. It builds upon any factorization of the joint distribution, e. g.,

$$p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = p(\mathbf{x}_3 \mid \mathbf{x}_2, \mathbf{x}_1) p(\mathbf{x}_2 \mid \mathbf{x}_1) p(\mathbf{x}_1). \tag{1.34}$$

Then we can sample from the joint distribution by sampling, in order,

$$\mathbf{x}_1 \sim p(\mathbf{x}_1), \qquad \mathbf{x}_2 \sim p(\mathbf{x}_2 \mid \mathbf{x}_1), \qquad \mathbf{x}_3 \sim p(\mathbf{x}_3 \mid \mathbf{x}_2, \mathbf{x}_1). \tag{1.35}$$

Despite being sampled individually from conditionals, the triplet $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ is now distributed according to the joint distribution.

Ancestral sampling is more of a sampling *strategy* than a technique. Its usefulness depends on how simple sampling from the conditionals is.

The term *ancestral sampling* highlights the connection to graphical models. Graphical models dictate the factorization of the joint distribution, and the joint distribution can then be sampled via ancestral sampling in topological order of the graph, i. e., starting from parent nodes and then child nodes as soon as all their ancestor nodes have been sampled. This strategy will be particularly useful when discussing *latent-variable models* (LVMs) later on.

In this context, it is also worth pointing out that, in addition to being distributed according to the joint distribution, each of the vectors of the triplet is also distributed according to its respective marginal distribution, i. e.,

$$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \sim p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \tag{1.36}$$

$$\Longrightarrow \mathbf{x}_1 \sim p(\mathbf{x}_1), \quad \mathbf{x}_2 \sim p(\mathbf{x}_2), \quad \mathbf{x}_3 \sim p(\mathbf{x}_3). \tag{1.37}$$

The converse is generally not true. This is a potentially computationally wasteful way of obtaining marginal samples, especially if we do not require samples from all marginals for the task at hand. However, this observation is useful when analyzing posteriors of a sequential LVM.

2 LATENT-VARIABLE MODELS

Throughout this thesis, we will learn probabilistic models for and from sequential data.

Our goal is to learn a distribution $p(\mathbf{x})$ of some random vector $\mathbf{x}$ from a data set of some $N \in \mathbb{N}$ samples $\mathcal{D} = \{\mathbf{x}^{(i)} \mid i = 1, \dots, N\}$, where each $\mathbf{x}^{(i)} \in \mathbb{R}^{d_x}$.[1] This puts us firmly into the realm of *unsupervised learning*, i. e., we only assume some representative data $\mathcal{D}$ but no supervision signals $y^{(i)}$ such as classification labels.

The method of choice for learning these so-called generative models $p(\mathbf{x})$ in this thesis are *latent-variable models* (LVMs), a class of probabilistic models. For alternatives such as stochastic processes, auto-regressive models, or direct density estimation via maximum likelihood, we refer the reader to, e. g., K. P. Murphy (2012).

While the bulk of this thesis discusses sequential models, it is worth studying non-sequential models first.

2.1 DEFINITIONS AND CONCEPTS

An LVM assumes that, beyond the observation $\mathbf{x}$, there also exists a latent random variable $\mathbf{z} \in \mathbb{R}^{d_z}$. Together, they follow a joint distribution

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z}), \tag{2.1}$$

where we call $p(\mathbf{z})$ the *prior* distribution and $p(\mathbf{x} \mid \mathbf{z})$ the *likelihood* distribution. From the joint distribution, a model for data $p(\mathbf{x})$ can be obtained via marginalization,

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) \, d\mathbf{z}. \tag{2.2}$$

It is worth noting that many joint distributions lead to the same marginal distribution; trivially, any model where $\mathbf{z}$ and $\mathbf{x}$ are independent, $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x})$, will have the appropriate marginal.

Thus, LVMs serve a conceptual purpose not immediately reflected by the definition: the essence of data is captured in the latent variable $\mathbf{z}$ and the corresponding prior distribution $p(\mathbf{z})$; the likelihood model $p(\mathbf{x} \mid \mathbf{z})$ captures how this essence is translated into an observation. The line between these concepts is often blurry yet drives the design of models and algorithms.

1 As is common in the literature, our notation will not distinguish between the random variable or vector $\mathbf{x}$ and the value it takes on unless specifically required from context.

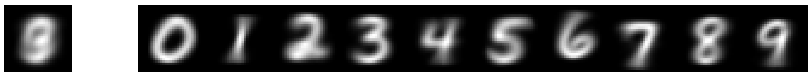


Figure 2.1: Per-pixel average digit of the MNIST test data set. The entire data set on the left, categorized by class label on the right.

As an example, consider the ubiquitous MNIST data set (LeCun et al., 2010), a data set of gray-scale images of handwritten digits. Figure 2.1 shows on the left the entire test set averaged, an approximation of the first moment of $p(\mathbf{x})$. Beyond being centered on the canvas, most information about the underlying data is lost in this statistic. It is unlikely to serve well as a building block of a useful model.

Contrast this with the right-hand side, which shows the average of the test set samples after they were categorized by their respective class labels. Here, the mean for each digit k would be more suitable grounds for the respective likelihood model $p(\mathbf{x} \mid z = k)$. This simple example illustrates one of the appeals of LVMs. By composing simple distributions—e. g., a Categorical distribution for the class label as well as a conditional Gaussian distribution for each digit class—we quickly arrive at a rich marginal distribution, a *Gaussian mixture model* (GMM).

The class membership is only one possible factor of variation to be modeled by a latent variable. Others might include the thickness of the strokes or the rotation of the digit. Which of these to model—and how—is a choice by the user or the learning algorithm that trades off faithfulness, complexity, and scalability of the resulting model. As the saying goes: "All models are wrong but some are useful." (Box, 1976, 1979) We will explore such trade-offs in the upcoming chapters of this thesis.

In learning LVMs from data, we face two core challenges.

Firstly, learning algorithms for the model: by design latent variables are missing during the learning process. It is thus not possible to learn the model via a standard maximum likelihood objective,

$$\arg\max_{\theta} \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})}[\ln p_\theta(\mathbf{x}, \mathbf{z})], \tag{2.3}$$

as the data set $\mathcal{D}$ does not contain latent variables. Here, $p_{\mathcal{D}}(\mathbf{x})$ denotes the data distribution, which may be empirical, and $p_\theta(\mathbf{x}, \mathbf{z})$ denotes a so far non-descript parametrized model with learnable parameters θ.

Likewise, it is frequently not possible to learn via maximum likelihood of the marginal model

$$\arg\max_{\theta} \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})}[\ln p_\theta(\mathbf{x})] = \arg\max_{\theta} \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})}\left[\ln \int p_\theta(\mathbf{x}, \mathbf{z})\, d\mathbf{z}\right]. \tag{2.4}$$

While this objective is at least well-defined in the sense that $\mathcal{D}$ does not need to contain latent variables, it requires solving the often intractable marginalization integral.

This leads to the second challenge for learning LVMs from data: inference. If the latent variables are not available in $\mathcal{D}$, can we at least infer a belief in them? This may help with the learning process, and given that $\mathbf{z}$ is interpreted as the essence of the respective observation $\mathbf{x}$, it is an interesting challenge in its own right.

Bayes' rule hands us a principled way of inferring the latent variable as the posterior

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}. \tag{2.5}$$

This brings the challenge full circle. Bayes' deceptively simple formula cannot hide the fact that the denominator is the same intractable marginal distribution that makes it challenging to learn an LVM.

2.2 VARIATIONAL INFERENCE

As it turns out, a key to solving these cyclically linked challenges of LVMs is *approximate* inference: the unavailable true posterior $p(\mathbf{z} \mid \mathbf{x})$ is replaced by a surrogate distribution $q(\mathbf{z})$—the *approximate posterior*. Now, one can show that

$$\begin{aligned}\ln p(\mathbf{x}) &\geqslant \ln p(\mathbf{x}) - \underbrace{\mathrm{KL}(q(\mathbf{z}) \,\|\, p(\mathbf{z} \mid \mathbf{x}))}_{\geqslant 0} && (2.6)\\ &= \mathbb{E}_{q(\mathbf{z})}\left[\ln \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}\right] && (2.7)\\ &= \mathbb{E}_{q(\mathbf{z})}[\ln p(\mathbf{x} \mid \mathbf{z})] - \mathrm{KL}(q(\mathbf{z}) \,\|\, p(\mathbf{z})) \equiv \mathcal{L}_{\mathrm{ELBO}}. && (2.8)\end{aligned}$$

Since $\ln p(\mathbf{x})$ is referred to as *evidence* in Bayesian terminology, either right-hand side expression in eqs. (2.6) to (2.8) defines the *evidence lower bound* (ELBO).

Given that we want to learn a model $p(\mathbf{x}, \mathbf{z})$, the joint pdf can typically be evaluated much more easily than the posterior. The approximate posterior $q(\mathbf{z})$ is likewise a model choice and thus tractable at least by MC integration. As a consequence, the ELBO is tractable even in scenarios where neither the marginal $p(\mathbf{x})$ nor the posterior $p(\mathbf{z} \mid \mathbf{x})$ are.

Further, the gap between evidence and ELBO is interpretable, it is precisely the posterior *Kullback-Leibler divergence* (KL)

$$\mathrm{KL}(q(\mathbf{z}) \,\|\, p(\mathbf{z} \mid \mathbf{x})) \equiv \int q(\mathbf{z}) \ln \frac{q(\mathbf{z})}{p(\mathbf{z})} \, d\mathbf{z}. \tag{2.9}$$

The gap can be tightened by finding better and better approximations to the intractable posterior. Conveniently, the posterior KL divergence in eq. (2.6) is bounded from below and $\ln p(\mathbf{x})$ is constant in q, so that for any $\mathbf{x}$

$$\arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q(\mathbf{z}) \,\|\, p(\mathbf{z} \mid \mathbf{x})) = \arg\max_{q \in \mathcal{Q}} \mathcal{L}_{\mathrm{ELBO}}(q, \mathbf{x}). \tag{2.10}$$

Here, $\mathcal{Q}$ is the set of available distributions for q. This set is called the *variational family*. If $p(\mathbf{z} \mid \mathbf{x}) \in \mathcal{Q}$, then and only then the gap will vanish, and the ELBO is in fact tight.

The ELBO can thus address the challenges we have identified with LVMs: per eq. (2.10), we can find good *approximate* posteriors by maximizing the tractable ELBO in q; in doing so, by eq. (2.6) the ELBO becomes a good proxy to maximum likelihood.

Classical inference by Bayes' rule is replaced by *optimization* in the space of distributions. This is a so-called variational problem, hence the term *variational inference* (VI) for this class of methods (Jordan et al., 1999; MacKay, 2002).

For an excellent, recent introduction and review of the vast field of VI, the reader is referred to Blei et al. (2017).

We will limit ourselves to the recently popularized approach of *stochastic variational inference* (SVI; Hoffman et al., 2013), predominantly to contrast it with the framework introduced in the following section. SVI leverages stochastic optimization: in its simplest form, for every data sample $\mathbf{x}^{(i)} \in \mathcal{D}$ a respective approximate posterior $q_i(\mathbf{z})$ is determined via SGD on the ELBO. Conceptually simple in this sense, it is a fairly flexible method, provided stochastic gradients of the ELBO can be obtained. This allows using more flexible approximate posteriors and generative models, especially compared to the previously predominant approaches.

While Hoffman et al. (2013) showed how to scale the method to "big" data sets in the order millions of samples, an inherent disadvantage remains: the parameters for each approximate posterior $q_i(\mathbf{z})$ either have to be stored or computed anew.

2.3 VARIATIONAL AUTO-ENCODERS

A recent approach that leverages the scalability of neural networks for learning generative models is the *variational auto-encoder* (VAE), independently and concurrently developed by Kingma and Welling (2014) and Rezende et al. (2014). We will briefly discuss the two main ingredients to VAEs: amortization and reparametrization.

AMORTIZATION A downside of the VI approaches as described in section 2.2 is scalability in the size of the data set: the respective approximate posterior for each data sample $\mathbf{x}^{(i)}$ needs to be computed individually in a possibly costly optimization procedure. This is particularly costly if the model parameters θ are updated: after every update, the approximate posteriors have to be optimized anew. This lack of scalability is the motivation for VAEs.

VAEs replace approximate inference via optimization with a density network with weights ϕ. In this work, we refer to it as the *inference* *network* though it is also known as the *recognition model*. The density network is shared between all samples; as discussed in section 1.1.4, it returns a member $q \in \mathcal{Q}$ as a function of a sample $\mathbf{x}^{(i)}$, usually by returning parameters of a distribution in a parametric family, e. g., mean and covariance for Gaussians. To reflect the shared nature of computations we write $q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})$ instead of $q_i(\mathbf{z})$ as above.

The VAE derives its name from deterministic auto-encoders, a neural architecture consisting of two subnetworks, encoder and decoder, linked via a low-dimensional representation. The inference network can be viewed as an encoder, with the low-dimensional representation being replaced with a stochastic latent variable. The role of the decoder is played by the likelihood $p_\theta(\mathbf{x} \mid \mathbf{z})$, which can also be implemented by a density network. From this lens, the prior $p_\theta(\mathbf{z})$ serves as regularizer of the auto-encoder.

It is worth noting that density networks do not render the approximate posteriors more flexible or accurate even when implemented via neural networks. In fact, a thought experiment reveals we might expect the opposite. Suppose we had an oracle optimizer: it would always return the optimal member of the variational family in sample-wise optimization. The same oracle optimizer might also provide us with the optimal network weights that minimize the ELBO on average. Even this optimal network can only return solutions as good as the optimal oracle approximations since both are constrained to the same variational family. Unless we also assume that the neural network can approximate the sample-wise process arbitrarily well, we must expect it to perform worse at least on some samples. We will formalize this intuition in section 2.4 and chapter 6.

The benefit is found elsewhere: density networks can exploit patterns among samples and their posteriors to arrive at approximate posteriors that are empirically *good enough* with a comparatively low—and fixed—amount of computation. All samples benefit from an improved set of network weights. On top, neural networks are well-suited for parallelization across large batches of samples. This approach is referred to as *amortization*.

REPARAMETRIZATION The scalability of *amortized variational inference* (AVI) stands and falls with the scalability of the underlying density networks. The VAE framework thus requires another ingredient: an objective function that can be optimized via SGD. The ELBO is the natural candidate. For SGD, we need to compute or estimate

$$\nabla_\phi \, \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\ln \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})}\right]. \tag{2.11}$$

As before, the expectation is typically not analytically solvable. This is particularly true if we implement the generative likelihood model $p_\theta(\mathbf{x} \mid \mathbf{z})$ with density networks.

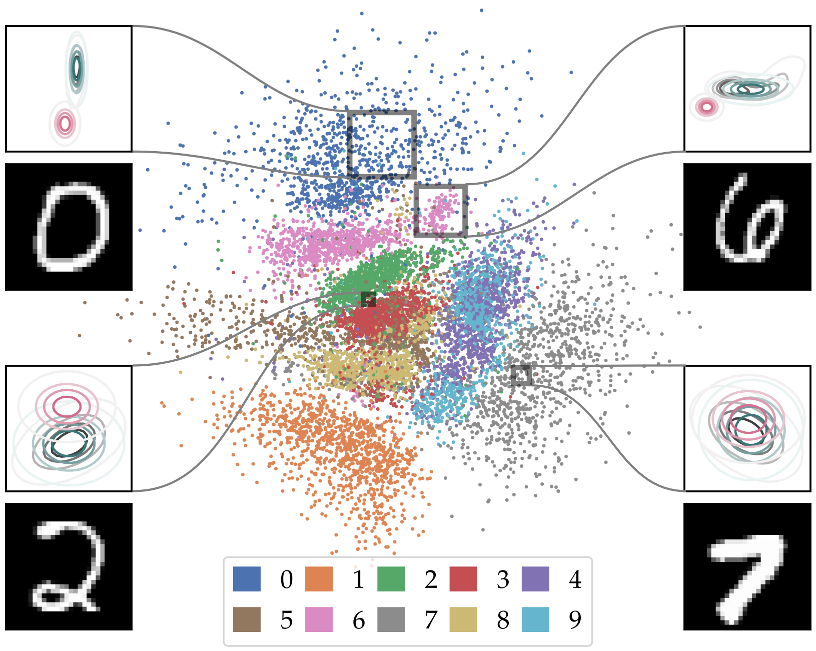


Figure 2.2: Latent space of a VAE with two latent dimensions trained on MNIST. For each posterior $q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})$ with $\mathbf{x}^{(i)}$ from the test set, a single sample colored by class label is shown. For four test set digits, the respective approximate posterior (red), optimal variational family member (green), and true posterior (black) are displayed. See section 2.3 for more details.

Overcoming this obstacle is the second building block of the VAE framework. The key observation is that we have to restrict the variational family $\mathcal{Q}$ to distributions that permit reparametrization, as discussed in section 1.1.5. With the help of reparametrization, the ELBO gradient in eq. (2.11) can be estimated without bias, and the entire VAE, inference network and generative model, can be trained jointly with SGD on the ELBO, *end-to-end*. VAEs are typically trained with just a single sample per posterior per gradient update.

EXAMPLE: VAE ON MNIST For a better understanding of VAEs, we explore a simple example on the MNIST data set. The inference model is a density network that returns Gaussians with restricted, diagonal covariance matrices. The likelihood model is a density network that returns independent Bernoullis per pixel. See appendix A.1 for details.

It should be noted that this is an illustrative example. The VAE was trained to convergence, but model and hyper-parameters were not tuned for optimal performance. Most notably, the latent space is constrained to two dimensions to facilitate visualization. Figure 2.2 shows one latent-space sample for each test set digit, colored by the respective class label.

The samples are spread unevenly, with more samples clustered in the center and a wider spread moving outwards. The samples can be viewed as representative samples from the mixture distribution

$$\frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} q_\phi\left(\mathbf{z} \,\middle|\, \mathbf{x}^{(i)}\right), \quad \mathbf{x}^{(i)} \in \mathcal{D}. \tag{2.12}$$

This *aggregate posterior* is regularized towards the prior since the inference network approximates the true posteriors and

$$p_\theta(\mathbf{z}) = \int p_\theta(\mathbf{z} \mid \mathbf{x}) p_\theta(\mathbf{x}) \, d\mathbf{x} \approx \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} p_\theta\left(\mathbf{z} \,\middle|\, \mathbf{x}^{(i)}\right). \tag{2.13}$$

The similarity between eqs. (2.12) and (2.13) explains the spread of samples roughly according to the prior, a standard Gaussian.

Yet, particularly between the clusters of classes we also see areas with lower number of samples than the prior would warrant. In these areas of low density of the aggregate posterior, the likelihood network would have to interpolate between digits from different classes. Where the likelihood density network is unable to do so because the interpolation would be too non-smooth, the VAE learns to avoid the respective latent-space regions. The notion of smoothness of the likelihood model has been explored further by N. Chen et al. (2018, 2019).

In a related manner, we see that clusters of similar digits, e. g., 4s and 9s, are located in similar regions of latent space. This is because the likelihood MLP can then interpolate smoothly, as is highlighted by the generative samples from the model depicted in fig. 2.3. These representative samples are generated to reflect the latent space as depicted in fig. 2.2, see appendix A.1 for details. These samples exhibit different factors of variation, such as boldness or orientation within class cluster, interpolation between classes with ambiguous hybrid samples, as well as spurious samples from the regions of low density of the aggregate posterior.

Figure 2.2 further shows four example digits. For each of these digits $\mathbf{x}$, the amortized approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ is computed and displayed in red. In green, the optimal member of the variational family of diagonal Gaussians as determined by SVI is shown. These examples exhibit imperfect inference via density networks to varying degree. Moreover, due to the low dimension of the latent space, the true posterior can be probed by exhaustive importance sampling. It is depicted in black. We observe that while on the one hand the true posteriors are not Gaussian, cf. the 6, the Gaussian assumption overall is fairly accurate. To some extent, this can be expected: through learning, the model $p_\theta(\mathbf{x}, \mathbf{z})$ can adjust to the variational family and be learned such that the corresponding posteriors can be approximated well enough by Gaussians. Conversely though, this is an implicit constraint to learning a more accurate model.

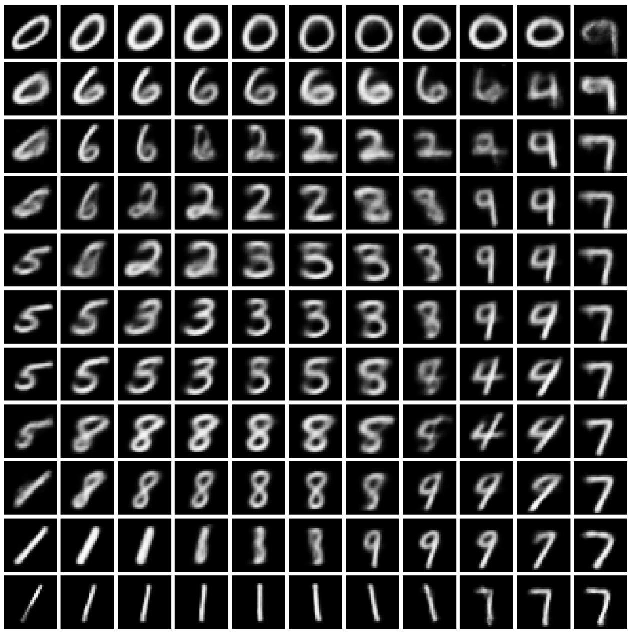


Figure 2.3: Representative generative mean samples from a VAE trained on MNIST. The samples cluster like the latent states in fig. 2.2 from the same model. One can observe factors of variation within a class cluster and similarity of samples on the border of class clusters. Further, some samples do not show a clear digit. Their respective latent states fall into a valley of the aggregate approximate posterior in fig. 2.2.

In the following section 2.4, we will take a more formal look at the *inference gaps* caused by the variational family and amortization.

2.4 INFERENCE GAPS

A central topic in VI are *inference gaps*. Loosely speaking, inference gaps characterize different modes of inference suboptimality of approximate inference. Several gaps are known in the literature and will be discussed here. We will discuss a new gap later in chapter 6.

2.4.1 Approximation Gap

The best possible approximate inference distribution for a sample observation $\mathbf{x}^{(i)}$ is the optimum

$$q_i^*(\mathbf{z}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\Big(q_i(\mathbf{z}) \,\Big\|\, p\Big(\mathbf{z} \,\Big|\, \mathbf{x}^{(i)}\Big)\Big) \tag{2.14}$$

within the *variational family* $\mathcal{Q}$.

The variational family is an assumption. Immediately, if the true posterior is not a member of the variational family, $p(\mathbf{z} \mid \mathbf{x}^{(i)}) \notin \mathcal{Q}$, the posterior divergence cannot vanish,

$$\mathrm{KL}\Big(q_i^*(\mathbf{z}) \,\Big|\Big|\, p\Big(\mathbf{z} \,\Big|\, \mathbf{x}^{(i)}\Big)\Big) > 0. \tag{2.15}$$

This irreducible gap between ELBO and log evidence $\ln p(\mathbf{x}^{(i)})$ is called *approximation gap*.

The approximation gap can be lowered by choosing a larger variational family. This choice has to trade off greater flexibility for the increased complexity of the optimization to obtain the optimum within the family as well as the practicability of the members of the variational family. We discuss the popular concept of normalizing flows in section 2.5.1

2.4.2 Amortization Gap

Amortization adds an additional gap on top of the approximation gap. Crucially, the optimization eq. (2.14) is per sample observation $\mathbf{x}^{(i)}$. Amortized variational inference replaces per-sample optimization with a typically neural function with learnable parameters ϕ that maps a sample observation $\mathbf{x}^{(i)}$ onto a member $q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})$ of the variational family. This immediate functional relationship is reflected by notation: $q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})$ instead of $q_i(\mathbf{z})$. The latter is only *indirectly* informed by $\mathbf{x}^{(i)}$ through optimization.

This leads to the *expected* ELBO

$$\arg\min_{\phi} \mathbb{E}_{p(\mathbf{x})}\left[\mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \,||\, p(\mathbf{z} \mid \mathbf{x})\right)\right] \tag{2.16}$$

as the objective, where instead of optimizing the member of the variational family for one specific observation $\mathbf{x}^{(i)}$ as in eq. (2.14), the function parameters ϕ of the approximate inference model are optimized such that *on average* the posterior KL is low. By definition, for an arbitrarily flexible function this would be the case if for all $\mathbf{x}^{(i)}$

$$q_\phi\Big(\mathbf{z} \,\Big|\, \mathbf{x}^{(i)}\Big) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\Big(q(\mathbf{z}) \,\Big|\Big|\, p\Big(\mathbf{z} \,\Big|\, \mathbf{x}^{(i)}\Big)\Big) = q_i^*(\mathbf{z}). \tag{2.17}$$

In general, one cannot expect a function approximation to solve eq. (2.14) accurately. The gap towards the log evidence widens:

$$\mathrm{KL}\Big(q_\phi\Big(\mathbf{z} \,\Big|\, \mathbf{x}^{(i)}\Big) \,\Big|\Big|\, p\Big(\mathbf{z} \,\Big|\, \mathbf{x}^{(i)}\Big)\Big) \geqslant \mathrm{KL}\Big(q_i^*(\mathbf{z}) \,\Big|\Big|\, p\Big(\mathbf{z} \,\Big|\, \mathbf{x}^{(i)}\Big)\Big). \tag{2.18}$$

This phenomenon was first discussed by Cremer et al. (2018), and the additional gap

$$\mathrm{KL}\Big(q_\phi\Big(\mathbf{z} \,\Big|\, \mathbf{x}^{(i)}\Big) \,\Big|\Big|\, p\Big(\mathbf{z} \,\Big|\, \mathbf{x}^{(i)}\Big)\Big) - \mathrm{KL}\Big(q_i^*(\mathbf{z}) \,\Big|\Big|\, p\Big(\mathbf{z} \,\Big|\, \mathbf{x}^{(i)}\Big)\Big) \geqslant 0 \tag{2.19}$$

is called the *amortization gap*.

The amortization gap is independent of the approximation gap in that a vanishing approximation gap does not imply vanishing amortization gap and vice versa. They have different causes: the variational family on the one hand and the limited capacity of the amortization model on the other.

Reducing the amortization gap can be achieved by, e. g., either using wider and deeper NNs for the inference network. Alternatively, the network design can be informed by inductive biases that make the network more suitable to amortize inference. We will see examples of this approach in parts II and III.

2.5 VAES AS A FRAMEWORK

VAEs can be viewed as a framework rather than an isolated model. It provides a very generic way to combine LVMs with auto-encoding and amortized inference in a principled yet scalable fashion. Any application that can be framed as an LVM immediately lends itself to this framework. This flexibility has spawned a host of subsequent research into improving and extending it, of which we will highlight three main threads of relevance to the remainder of the thesis.

2.5.1 Normalizing Flows

Motivated by lowering the approximation gap, an immediate candidate for extending the framework is the variational family. For simplicity, the family of Gaussians is often the default choice. It is tempting to utilize the flexibility of neural networks to overcome this restriction. A key observation is that learning a VAE does not require a closed-form distribution. Samples and pdf evaluations are sufficient to estimate gradients.

A natural idea is to use neural networks for a *change of variables*, i. e., using an invertible and differentiable function f and simply mapping samples $\boldsymbol{\epsilon}$ from a simple distribution $p(\boldsymbol{\epsilon})$ to obtain samples $\mathbf{z} = f(\boldsymbol{\epsilon})$ of a more complicated distribution.

In order to be useful for VAEs, we need to be able to at least evaluate the pdf of this new distribution. For a change of variables we know

$$p(\mathbf{z}) = p(\boldsymbol{\epsilon}) \left| \det \frac{\partial f}{\partial \boldsymbol{\epsilon}} \right|^{-1}. \tag{2.20}$$

The latter factor denotes the inverse determinant of the Jacobian matrix of f w. r. t. $\boldsymbol{\epsilon}$.

Being able to obtain samples and evaluate the pdf in this fashion would allow us to estimate the ELBO using a potentially richer variational family. Two challenges to employ neural networks for f remain. The function f is required to be invertible, which neural networks are not by default, and computing the Jacobian and its determinant in a naive way has cubic complexity for an arbitrary function.

Rezende and Mohamed (2015) suggested restricted neural layers that are invertible by default and their determinant can be computed in linear complexity. They call these constructs *normalizing flows*. Since then, a host of normalizing flows has been suggested (Kingma et al., 2016; Dinh et al., 2017; Papamakarios et al., 2017). Recent, comprehensive reviews are provided by Papamakarios et al. (2019) and Kobyzev et al. (2020).

2.5.2 Flexible Priors

Normalizing flows were initially developed to improve the flexibility of the variational family. In principle, they can be used just as well to devise more flexible priors. If the marginal distribution $p(\mathbf{x})$ is the object of interest, this option is of less interest since the added flexibility can equally well be represented by a flexible likelihood model $p(\mathbf{x} \mid \mathbf{z})$.

A more flexible prior is advisable in scenarios where the prior is of interest itself. Sequential LVMs, as discussed throughout this thesis, are such a case. More recent approaches provide flexible priors by introducing hierarchies of latent variables (Sønderby et al., 2016; Klushyn et al., 2019; Child, 2020). Their detailed discussion is beyond the scope of this thesis.

2.5.3 Alternative Bounds

Another pillar of research on the VAE framework is tweaking or replacing the ELBO as the learning objective. *Monte Carlo objectives* (MCOs) are a general strategy to derive lower bound objectives to the log marginal likelihood $\ln p(\mathbf{x})$ (Mnih and Rezende, 2016). Starting from an unbiased statistical estimator $\hat{p}(\mathbf{x})$ of $p(\mathbf{x})$, i. e.,

$$\mathbb{E}[\hat{p}(\mathbf{x})] = p(\mathbf{x}), \tag{2.21}$$

one can derive an MCO by applying Jensen's inequality:

$$\ln p(\mathbf{x}) = \ln \mathbb{E}[\hat{p}(\mathbf{x})] \geqslant \mathbb{E}[\ln \hat{p}(\mathbf{x})]. \tag{2.22}$$

The ELBO is an example of an MCO. With

$$\hat{p}(\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}, \quad \mathbf{z} \sim q(\mathbf{z}), \tag{2.23}$$

we obtain the ELBO via

$$\mathbb{E}_{q(\mathbf{z})}\left[\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}\right] = p(\mathbf{x}) \implies \mathbb{E}_{q(\mathbf{z})}\left[\ln \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}\right] \leqslant \ln p(\mathbf{x}). \tag{2.24}$$

Another important example are *importance-weighted auto-encoders* (IWAEs), due to Burda et al. (2016).[2] IWAEs are identical to VAEs, except they use a different MCO as the objective. Instead of the estimator in eq. (2.23), they use the natural, equally unbiased but less variant multi-sample estimator

$$\hat{p}(\mathbf{x}) = \sum_{k=1}^{K} \frac{p\left(\mathbf{x}, \mathbf{z}^{(k)}\right)}{q\left(\mathbf{z}^{(k)}\right)}, \quad \mathbf{z}^{(k)} \sim q(\mathbf{z}), \tag{2.25}$$

which leads to the MCO

$$\mathbb{E}_{q(\mathbf{z})}\left[\ln \sum_{k=1}^{K} \frac{p\left(\mathbf{x}, \mathbf{z}^{(k)}\right)}{q\left(\mathbf{z}^{(k)}\right)}\right]. \tag{2.26}$$

The estimator is familiar to us from importance sampling, cf. section 1.2.1, hence the naming. As should be expected, for $K = 1$ we recover the ELBO. Moreover, one can show that the bound is monotonically non-decreasing in K, i. e., the bound is guaranteed to get tighter with growing number of samples.

It should be noted that instead of interpreting IWAEs as suggesting a tighter bound as the objective, it may also be seen as enriching the variational family while sticking to the ELBO as the objective (Cremer et al., 2017).

Further bounds based on different divergences have been discussed, cf. appendix A.2, but will not be used throughout this thesis. Similarly, we briefly mention β-VAEs (Higgins et al., 2017), a variant of VAEs that scale the likelihood term to obtain better latent representations. An interesting intuition was later provided by Alemi et al. (2018) in terms of a Pareto frontier of models that trade off the complexity of latent representation $\mathbf{z}$ for the complexity of the likelihood model $p(\mathbf{x} \mid \mathbf{z})$. We refrain from further discussion but point out that balancing the terms of the ELBO—even online during learning—is an active direction of research (Klushyn et al., 2019).

2 Historically, IWAEs were suggested before (or at least independent of) MCOs.

3 SEQUENTIAL LATENT-VARIABLE MODELS

We now turn our attention to the study of *dynamical systems*. Here, this broad term denotes entities or collections thereof and their dynamic evolution over time as observed through sensors.

Such systems lend themselves to be modeled with LVMs: by distinguishing between observable and latent aspects of the dynamics, LVMs provide the tools for principled Bayesian inference about the state of a system.

We thus extend the theory discussed in chapter 2 to *sequences* of length $T \in \mathbb{N}$ to obtain sequential LVMs. We discuss sequences of latent variables $\mathbf{z}_{1:T} = (\mathbf{z}_1, \dots, \mathbf{z}_T)$ and observations $\mathbf{x}_{1:T} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$, respectively. Similar to the non-sequential (static) case, we are now interested in models of sequences of observations $\mathbf{x}_{1:T}$,

$$p(\mathbf{x}_{1:T}) = \int p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) \, d\mathbf{z}_{1:T}, \tag{3.1}$$

after marginalization of the sequence of latent variables $\mathbf{z}_{1:T}$.

REMARKS AND NOMENCLATURE

1. In this thesis, we will only discuss models with discrete time and fixed time interval Δt between subsequent steps in time.
2. Being such a vast field with widespread applications, different nomenclatures have evolved. Observations $\mathbf{x}_t$ are often also called *measurements*, emphasizing, e. g., the sensor setup, or *emissions* that are merely glimpses into and emitted by the underlying latent system. Likewise, latent variables—or just latents—$\mathbf{z}_t$ are also called (latent) states, to emphasize that they capture the actual state of the system as opposed to noisy, distorted, or projected measurements.
3. Systems are often studied with the intent to *control* them by means of control inputs $\mathbf{u}_t$ to the system (also called *actions* or more generally *inputs* or *conditions*). This changes the distributions of interest to, e. g., $p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T})$ or $p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})$. The theory presented throughout this thesis is largely unaffected by presence or absence of control signals. For notational brevity, we will discuss the control-free versions by default and consider control inputs where appropriate.

From one perspective, sequential LVMs are a special case of the theory presented in chapter 2. For instance, we can argue in terms of a prior

$p(\mathbf{z}_{1:T})$, a likelihood model $p(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T})$, or a posterior $p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$. All considerations from the static case apply.

At the same time, the joint distribution of a sequence can be factorized in a larger variety of interesting ways according to the chain rule of probability, e. g.,

$$\begin{aligned}
&p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) && (3.2)\\
&= p(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}) p(\mathbf{z}_{1:T}) && (3.3)\\
&= \prod_{t=1}^{T} p(\mathbf{x}_t, \mathbf{z}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) && (3.4)\\
&= \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{x}_{1:t-1}) p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}). && (3.5)
\end{aligned}$$

Where eq. (3.3) emphasizes the prior-likelihood nature of the model, eqs. (3.4) and (3.5) shed more light on the sequential nature of the model: in accordance with our notion of time and causality, later time steps are conditioned on earlier time steps; and the observation $\mathbf{x}_t$ at time t depends on the latent state $\mathbf{z}_t$.

All of these factorizations are equally true and of use in different situations. Any such choice sheds a different light on the model at hand. A particular choice of factorization becomes interesting once we impose assumptions on the model, typically by assuming independence of certain variables. For instance, we might assume that—given current and past latent states—the randomness of observations is only caused by sensor noise. The mathematical model could reflect this as independence of observations given states, and eq. (3.5) would become

$$\prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \cancel{\mathbf{x}_{1:t-1}}) p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}), \qquad (3.5a)$$

or even

$$\prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) p(\mathbf{z}_t \mid \cancel{\mathbf{x}_{1:t-1}}, \mathbf{z}_{1:t-1}) \qquad (3.5b)$$

if we also assume that the independent sensor noise does not feed back into the latent system.

3.1 STATE-SPACE MODELS

Equations (3.5), (3.5a), and (3.5b) are three examples of valid factorizations with increasingly stronger model assumptions. Upon closer inspection, we observe that all three factorizations consist of 2T unique factors. Despite imposing assumptions, we would still be required to specify a probabilistic model for each of the 2T factors, i. e., with increasing sequence length the number of model components increases linearly.

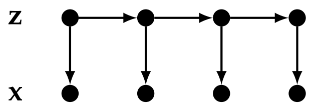


Figure 3.1: Graphical model of a *state-space models* (SSMs).

This is caused by all factors depending on an increasingly longer past. It is thus common to impose *Markov assumptions*—given the present, past and future are assumed independent. Two such Markov assumptions lead from eq. (3.5) to *state-space models* (SSMs):

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{x}_{1:t-1}) = p(\mathbf{x}_t \mid \mathbf{z}_t), \tag{3.6}$$

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) = p(\mathbf{z}_t \mid \mathbf{z}_{t-1}). \tag{3.7}$$

Plugging back into eq. (3.5), this leads to the simplified factorization

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = p(\mathbf{z}_1) \prod_{t=1}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t) \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_t). \tag{3.8}$$

We call $p(\mathbf{z}_1)$ the *initial state* (prior) distribution, $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)$ the *transition* model, and $p(\mathbf{x}_t \mid \mathbf{z}_t)$ the *emission* model, and $\mathbf{x}_t$ synonymously either emissions or observations. By further sharing both transition and emission model across time, we have reduced the modeling effort from 2T components to a fixed three components, independent of the length of sequences.[1]

In an SSM, the state $\mathbf{z}_t$ plays a powerful role: both the future state $\mathbf{z}_{t+1}$ and the current observation $\mathbf{x}_t$ depend solely on $\mathbf{z}_t$, i. e., it must encompass all relevant information about the underlying system to predict the observations as well as the future. This is also reflected in the graphical model shown in fig. 3.1: the present state $\mathbf{z}_t$ blocks all paths between past $(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})$, present $(\mathbf{x}_t)$, and future $(\mathbf{x}_{t+1:T}, \mathbf{z}_{t+1:T})$, rendering them conditionally independent.

3.2 SEQUENTIAL BAYESIAN POSTERIORS

Much like the factorization of the joint distribution is more nuanced, the notion of "the" posterior is less obvious than in the static case. The *joint* posterior

$$p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \frac{p(\mathbf{z}_{1:T}, \mathbf{x}_{1:T})}{p(\mathbf{x}_{1:T})} \propto p(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}) \tag{3.9}$$

is theoretically available by Bayes' rule. One could obtain more specific inferences by appropriate marginalizations. Even if these operations

1 In the literature, there exists the notion of *time-variant* systems, where transition or emission model are not or only partially shared across time. In this work, we will only study time-invariant SSMs.

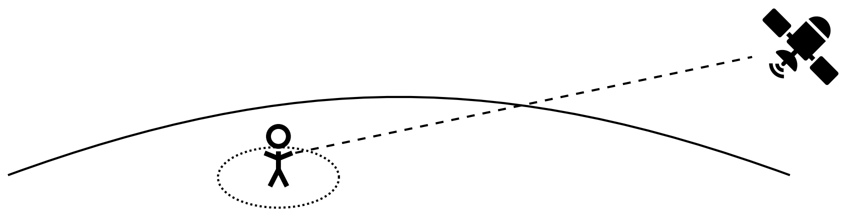


Figure 3.2: Raw satellite signals $\mathbf{x}_t$ observe the true position with a coarse accuracy of several meters. Consumer devices can reduce this error by orders of magnitude by various filtering techniques. With an accurate motion model, this error can be reduced further (Banville and Diggelen, 2016).

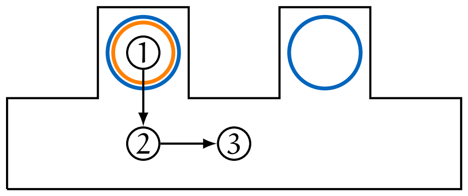


Figure 3.3: Filtering vs. smoothing for indoor localization. The agent perceives distances to walls in all directions, $\mathbf{x}_t$, akin to Lidar sensors. The true position is part of the latent state $\mathbf{z}_t$. The agent starts at position 1 at time $t = 1$ and subsequently moves to positions 2 and 3. The left and right room are indistinguishable due to symmetry, the filtering posterior $p(\mathbf{z}_1 \mid \mathbf{x}_1)$ is thus bimodal. Observations $\mathbf{x}_2$ and $\mathbf{x}_3$ break the symmetry by observing the entire hallway at the bottom, the smoothing posterior $p(\mathbf{z}_1 \mid \mathbf{x}_{1:3})$ is unimodal. Note that filtering and smoothing posterior from $t = 2$ onwards collapse to the same mode once the symmetry-breaking observation $\mathbf{x}_2$ arrives. Situations in which the same observation can be caused by very different latent states, like $\mathbf{x}_1$ in this example, are often called *perceptual aliasing*.

are tractable, this approach is not advisable as its computational time and space requirements are often prohibitive.

Considering the outstanding role of the state in SSMs, we are often interested in narrower inferences. This leads to the notions of (Bayesian) *filtering* and *smoothing*.

FILTERING Filtering asks the question: given all observations made so far, $\mathbf{x}_{1:t}$, what is my (posterior) belief in the current latent state $\mathbf{z}_t$ of the system? We are trying to determine

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:t}). \tag{3.10}$$

Filtering is relevant in systems where accurate estimates of the latent state of the system are required *ad hoc*, e. g., because downstream decisions need to be made online based on everything that has been observed *so far*. Filtering is thus very common in engineering disciplines (Julier and Uhlmann, 2004). A classic example is tracking as depicted in fig. 3.2.

SMOOTHING Smoothing tackles a similar question but with the benefit of hindsight: where filtering neglects the additional information provided by future observations $\mathbf{x}_{t+1:T}$ (assuming $t < T$), smoothing asks the question what our *post-hoc* belief in the latent state $\mathbf{z}_t$ given all available observations $\mathbf{x}_{1:T}$ is,

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:T}). \tag{3.11}$$

The benefit of hindsight can rule out hypotheses for $\mathbf{z}_t$ that were still credible in the filtering scenario. This comes at the cost of the mandatory delay to gather future observations (relative to $\mathbf{z}_t$).

A minimal example showcasing the relation between filtering and smoothing is depicted in fig. 3.3.

In many scenarios, additional information to a posterior like the additional observations for a smoother compared to a filter will lead to a "more certain" belief in terms of, e. g., variance or entropy, as is true for the example in fig. 3.3. This is not generally true, only *on average*. This subtle misconception is further discussed in appendix A.3.

3.3 INFERENCE IN STATE-SPACE MODELS

3.3.1 Building Blocks of SSM Posteriors

In trying to understand the extensions of the VAE as presented later in parts II and III, it is worth dissecting the joint, filtering, and smoothing posteriors, eqs. (3.9) to (3.11). Expressing them in terms of the basic building blocks of an SSM, the initial state distribution, the transition, and the emission model can inform the design of algorithms and approximations.

FORWARD FILTER The first building block is the *forward filter*

$$\alpha_t(\mathbf{z}_t) \equiv p(\mathbf{z}_t \mid \mathbf{x}_{1:t}), \tag{3.12}$$

as encountered in eq. (3.10). It follows a recursive structure:

$$\begin{aligned}
\alpha_1(\mathbf{z}_1) = p(\mathbf{z}_1 \mid \mathbf{x}_1) &= \frac{p(\mathbf{x}_1 \mid \mathbf{z}_1)p(\mathbf{z}_1)}{p(\mathbf{x}_1)} && (3.13)\\
&\propto p(\mathbf{x}_1 \mid \mathbf{z}_1)p(\mathbf{z}_1), && (3.14)
\end{aligned}$$

$$\begin{aligned}
\alpha_t(\mathbf{z}_t) &= \frac{p(\mathbf{x}_t, \mathbf{z}_t \mid \mathbf{x}_{1:t-1})}{p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})} && (3.15)\\
&= \frac{p(\mathbf{x}_t \mid \mathbf{z}_t)}{p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})} \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \underbrace{p(\mathbf{z}_{t-1} \mid \mathbf{x}_{1:t-1})}_{\alpha_{t-1}(\mathbf{z}_{t-1})} \, d\mathbf{z}_{t-1} && (3.16)\\
&\propto p(\mathbf{x}_t \mid \mathbf{z}_t) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \alpha_{t-1}(\mathbf{z}_{t-1}) \, d\mathbf{z}_{t-1}. && (3.17)
\end{aligned}$$

Crucially, all elements in this recursive definition only make use of initial state distribution, transition, and emission model—up to normalizing constants.

Equation (3.17) hints at a general skeleton for Bayesian filtering algorithms: the prediction-update cycle. The integral factor *predicts* with the transition model, based on the previous belief.[2] The emission model factor then *updates* the prediction according the latest observation to arrive at the new filtered belief. We will revisit the prediction-update cycle later.

BACKWARD FILTER A second building block for analyzing Bayesian posteriors in SSMs is the *backward filter*

$$\beta_t(\mathbf{z}_t) \equiv p(\mathbf{x}_{t+1:T} \mid \mathbf{z}_t), \tag{3.18}$$

i. e., the likelihood of future observations given the current state. Similar to the forward filter, it follows a recursive structure, this time starting from the back of the sequence:

$$\beta_{T-1}(\mathbf{z}_{T-1}) = p(\mathbf{x}_T \mid \mathbf{z}_{T-1}) = \int p(\mathbf{x}_T \mid \mathbf{z}_T)p(\mathbf{z}_T \mid \mathbf{z}_{T-1})\,\mathrm{d}\mathbf{z}_T, \tag{3.19}$$

$$\begin{aligned}\beta_t(\mathbf{z}_t) &= \int p(\mathbf{x}_{t+1:T}, \mathbf{z}_{t+1} \mid \mathbf{z}_t)\,\mathrm{d}\mathbf{z}_{t+1} &(3.20)\\ &= \int \beta_{t+1}(\mathbf{z}_{t+1})p(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1})p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)\,\mathrm{d}\mathbf{z}_{t+1}. &(3.21)\end{aligned}$$

Again, each recursive equation can be expressed using only the three SSM building blocks. Beyond the model components of an SSM, forward and backward filter are key tools for analyzing other interesting posterior quantities, as we will do in the following sections.

SMOOTHER With forward and backward filter, we can express the smoother

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:T}) \tag{3.22}$$

as known from eq. (3.11). From Bayes' rule, we get

$$\begin{aligned}p(\mathbf{z}_t \mid \mathbf{x}_{1:T}) &= \frac{p(\mathbf{z}_t \mid \mathbf{x}_{1:t})p(\mathbf{x}_{t+1:T} \mid \mathbf{z}_t)}{p(\mathbf{x}_{t+1:T} \mid \mathbf{x}_{1:t})} &(3.23)\\ &\propto \alpha_t(\mathbf{z}_t)\beta_t(\mathbf{z}_t). &(3.24)\end{aligned}$$

From the perspective of eq. (3.24), forward and backward filter act analogous to prior and likelihood for Bayesian posterior estimation of $\mathbf{z}_t$.

2 This marginalization of $\mathbf{z}_{t-1}$ is also called *Chapman-Kolmogorov equation*.

CONSECUTIVE-STATE JOINT POSTERIOR In the VAE framework, it is important that models and posteriors or their approximations are easy to sample. We thus investigate *posterior transitions*, which could allow us to sample the posterior step-wise via ancestral sampling:

$$p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t+1:T}). \tag{3.25}$$

Here, the independence from past observations $\mathbf{x}_{1:t}$ on the right-hand side is a direct consequence of the SSM assumptions. Given the state $\mathbf{z}_t$, future and past are independent. This can also be deduced from the graphical model.

An intermediate quantity for understanding the transitions is the posterior joint distribution of two consecutive states,

$$p(\mathbf{z}_t, \mathbf{z}_{t+1} \mid \mathbf{x}_{1:T}). \tag{3.26}$$

We can compute it as

$$\begin{aligned} p(\mathbf{z}_t, \mathbf{z}_{t+1} \mid \mathbf{x}_{1:T}) &= \frac{p(\mathbf{z}_t, \mathbf{z}_{t+1}, \mathbf{x}_{t+1:T} \mid \mathbf{x}_{1:t})}{p(\mathbf{x}_{t+1:T} \mid \mathbf{x}_{1:t})} && (3.27) \\ &\propto p(\mathbf{x}_{t+2:T} \mid \cancel{\mathbf{x}_{t+1}}, \mathbf{z}_{t+1}, \cancel{\mathbf{z}_t}, \cancel{\mathbf{x}_{1:t}}) && (\beta_{t+1}(\mathbf{z}_{t+1})) \\ &\quad \cdot p(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1}, \cancel{\mathbf{z}_t}, \cancel{\mathbf{x}_{1:t}}) && (p(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1})) \\ &\quad \cdot p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \cancel{\mathbf{x}_{1:t}}) && (p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)) \\ &\quad \cdot p(\mathbf{z}_t \mid \mathbf{x}_{1:t}). && (\alpha_t(\mathbf{z}_t)) \end{aligned}$$

The cancellations follow from the SSM assumptions: the remaining conditions block the path to the canceled conditions in the graphical model. As before, we have expressed the joint posterior distribution as a product of known distributions up to a normalizing constant.

POSTERIOR FORWARD TRANSITION Given the consecutive-state joint posterior and the smoother marginal, we can now compute the posterior forward transition

$$p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t+1:T}) \tag{3.28}$$

with Bayes' rule and inserting the previous results:

$$\begin{aligned} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t+1:T}) &= \frac{p(\mathbf{z}_t, \mathbf{z}_{t+1} \mid \mathbf{x}_{1:T})}{p(\mathbf{z}_t \mid \mathbf{x}_{1:T})} && (3.29) \\ &= \frac{\beta_{t+1}(\mathbf{z}_{t+1})}{\beta_t(\mathbf{z}_t)} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1}). && (3.30) \end{aligned}$$

It is worth noting that (i) the above is exact since the normalizing constants of numerator and denominator are identical, compare eqs. (3.23) and (3.27), and (ii) $\beta_t(\mathbf{z}_t)$ is a normalizing constant as the forward transition is a density in $\mathbf{z}_{t+1}$. This is a very interesting result: the posterior transition is the same as the prior transition up to reweighting that accounts for future observations.

These insights into the posterior forward transition will be the building block for a novel design of an approximation of eq. (3.28) in chapter 7.

POSTERIOR BACKWARD TRANSITION For completeness, the backward transition

$$p(\mathbf{z}_t \mid \mathbf{z}_{t+1}, \mathbf{x}_{1:T}) \tag{3.31}$$

follows a similar pattern:

$$p(\mathbf{z}_t \mid \mathbf{z}_{t+1}, \mathbf{x}_{1:T}) = \frac{p(\mathbf{z}_t, \mathbf{z}_{t+1} \mid \mathbf{x}_{1:T})}{p(\mathbf{z}_{t+1} \mid \mathbf{x}_{1:T})} \tag{3.32}$$

$$\propto \frac{\alpha_t(\mathbf{z}_t)}{\alpha_{t+1}(\mathbf{z}_{t+1})} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1}), \tag{3.33}$$

with normalization constant

$$p(\mathbf{x}_{t+2:T} \mid \mathbf{x}_{1:t+1}). \tag{3.34}$$

We will not use the backward transition further throughout this thesis, but it emphasizes the important role of forward and backward filter along with the three building blocks to understand posterior analysis of SSMs.

3.3.2 Kalman and Particle Filters

Since applications of SSMs are ubiquitous, there exists a vast body of literature solving or approximating posterior quantities for various scenarios and assumptions. In the following, we will give a brief introduction to Kalman filters (Kalman, 1960) and particle filters (Del Moral, 1996). Arguably two of the most popular algorithms for Bayesian filtering, they serve as a solid foundation for understanding the algorithms presented in parts II and III. Our discussion will be limited to the basics necessary for these discussions. For a more thorough introduction to a wide variety of related algorithms, the interested reader is referred to the excellent reference by Sarkka (2013).

3.3.2.1 *Kalman Filters*

Kalman filters are a special case of filtering for a restricted class of SSMs, so-called *linear Gaussian systems* (LGSs). LGSs assume

$$p(\mathbf{z}_1) \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \qquad \boldsymbol{\mu}_1 \in \mathbb{R}^{d_z}, \boldsymbol{\Sigma}_1 \in \mathbb{R}^{d_z \times d_z}, \tag{3.35}$$

$$p(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \sim \mathcal{N}(\mathbf{A}\mathbf{z}_{t-1}, \mathbf{Q}), \qquad \mathbf{A}, \mathbf{Q} \in \mathbb{R}^{d_z \times d_z}, \tag{3.36}$$

$$p(\mathbf{x}_t \mid \mathbf{z}_t) \sim \mathcal{N}(\mathbf{H}\mathbf{z}_t, \mathbf{R}), \qquad \mathbf{H} \in \mathbb{R}^{d_x \times d_z}, \mathbf{R} \in \mathbb{R}^{d_x \times d_x}, \tag{3.37}$$

with $\mathbf{\Sigma}_1$, $\mathbf{Q}$, and $\mathbf{R}$ symmetric and positive semidefinite. The transition and the emission model in eqs. (3.36) and (3.37) can be rephrased equivalently as

$$\mathbf{z}_t = \mathbf{A}\mathbf{z}_{t-1} + \boldsymbol{\epsilon}_{t-1}, \qquad \boldsymbol{\epsilon}_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \tag{3.38}$$

$$\mathbf{x}_t = \mathbf{H}\mathbf{z}_t + \boldsymbol{\delta}_t, \qquad \boldsymbol{\delta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}). \tag{3.39}$$

That is, in an LGS all involved random variables are Gaussian, and all conditional distributions are linear in their conditions. We call $\mathbf{A}, \mathbf{Q}$, and $\boldsymbol{\epsilon}$ *transition* matrix, covariance, and noise, respectively. Correspondingly, $\mathbf{H}, \mathbf{R}$ and $\boldsymbol{\delta}$ are called *emission* matrix, covariance, and noise, respectively. Within this work, we further assume that the matrices and covariances are *time-invariant*, i. e., they remain constant over time. Similarly, we assume transition and emission noises to be independent from each other and i. i. d. across time.

LGSs are a very friendly special case of SSMs in the sense that a lot of interesting posterior quantities are tractable and can be computed efficiently. The Kalman filter—the Bayesian filter algorithm for $p(\mathbf{z}_t \mid \mathbf{x}_{1:t})$—is arguably a wide-spread example.

The purpose of this section is thus less to review and dissect Kalman filters, as has been done numerous times over the past half-century, but to offer a perspective that helps understanding new concepts in parts II and III.

Recall the prediction-update cycle, eq. (3.17), which we rewrite as two equations:

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}) = \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1})\alpha_{t-1}(\mathbf{z}_{t-1})\, \mathrm{d}\mathbf{z}_{t-1}, \tag{prediction}$$

$$\alpha_t(\mathbf{z}_t) \propto p(\mathbf{x}_t \mid \mathbf{z}_t)p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}). \tag{update}$$

Both equations can be solved in closed form for LGSs, and the resulting algorithm is the Kalman filter.

PREDICTION Since the initial state distribution is Gaussian and the filter is defined recursively, we can conclude by induction through time that for any LGS the filter distribution is also Gaussian,

$$\alpha_{t-1}(\mathbf{z}_{t-1}) \sim \mathcal{N}\left(\boldsymbol{\mu}_{t-1}^{(f)}, \mathbf{\Sigma}_{t-1}^{(f)}\right), \tag{3.40}$$

for some mean $\boldsymbol{\mu}_{t-1}^{(f)}$ and covariance $\mathbf{\Sigma}_{t-1}^{(f)}$.

Then, inserting into eq. (3.38), standard rules for multivariate Gaussian distributions tell us that for $\mathbf{z}_{t-1} \sim \alpha_{t-1}(\mathbf{z}_{t-1})$

$$\mathbf{z}_t = \mathbf{A}\mathbf{z}_{t-1} + \boldsymbol{\epsilon}_{t-1} \sim \mathcal{N}\Big(\underbrace{\mathbf{A}\boldsymbol{\mu}_{t-1}^{(f)}}_{\equiv \boldsymbol{\mu}_t^{(p)}}, \underbrace{\mathbf{A}\mathbf{\Sigma}_{t-1}^{(f)}\mathbf{A}^\top + \mathbf{Q}}_{\equiv \mathbf{\Sigma}_t^{(p)}}\Big). \tag{3.41}$$

Adding $\mathbf{Q}$, the prediction step usually increases uncertainty—unless $\mathbf{A}$ contracts strongly, i. e., has eigenvalues of absolute values much smaller than 1. Figure 3.4 depicts this as the transition in state space from $\alpha_{t-1}(\mathbf{z}_{t-1})$ to $\mathbb{E}_{\alpha_{t-1}}[p(\mathbf{z}_t \mid \mathbf{z}_{t-1})]$.

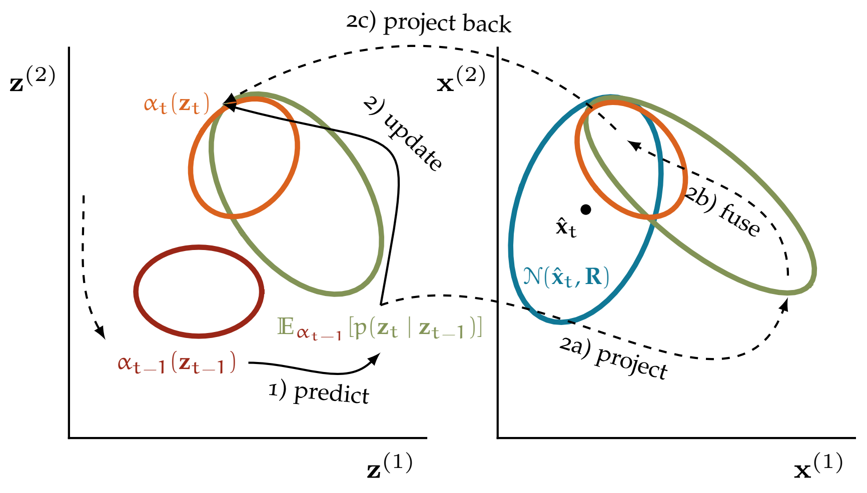


Figure 3.4: The prediction-update cycle of Kalman filters. Gaussian distributions are represented by ellipses around the means corresponding to 50% of probability mass. Starting from the current filtering posterior $\alpha_{t-1}(\mathbf{z}_{t-1})$, a prediction $\mathbb{E}_{\alpha_{t-1}}[p(\mathbf{z}_t \mid \mathbf{z}_{t-1})]$ is made in state space (left) according to eq. (3.41). The prediction is updated with the observation $\hat{\mathbf{x}}_t$ to obtain $\alpha_t(\mathbf{z}_t)$ according to eq. (3.42). If the emission matrix $\mathbf{H}$ is square and invertible, the update step can be broken down into (a) projecting the prediction into observation space (right), (b) fusion with the noisy observation $\mathcal{N}(\mathbf{x}_t, \mathbf{R})$, and (c) undoing the projection. Correspondence via projection between state and observation space is indicated by matching colors of the ellipses.

UPDATE The update step

$$\alpha_t(\mathbf{z}_t) \propto p(\mathbf{x}_t \mid \mathbf{z}_t) p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}) \tag{3.42}$$

is often tackled in one of two ways:

1. inserting the Gaussian pdfs and completing the square to normalize the right-hand side; or
2. noticing that the right-hand side is a joint Gaussian pdf in $\mathbf{z}_t$ and $\mathbf{x}_t$ and computing the marginal in $\mathbf{z}_t$ according to standard rules for multivariate Gaussians.

Both ways lead to closed-form solutions of eq. (3.42) and numerically stable Kalman filter implementations but are rather pedestrian. In the following, we will trade the generality of these solutions for a more intuitive derivation that will prove helpful in understanding the design decisions in section 4.2. The derivation is accompanied by the visualization in fig. 3.4.

To this end, we now additionally assume that the emission matrix $\mathbf{H}$ is square and invertible. Often, this is not the case, for instance with redundant sensors ($d_x > d_z$) or higher-order state components that are not measured by a sensor ($d_x < d_z$).

The motivation for this additional assumption is that we can now start from the back—the desired distribution $\alpha_t(\mathbf{z}_t) \sim \mathcal{N}\left(\boldsymbol{\mu}_t^{(f)}, \boldsymbol{\Sigma}_t^{(f)}\right)$—and project into observation space by a change of variables, which requires the invertible $\mathbf{H}$. At this point, $\boldsymbol{\mu}_t^{(f)}$ and $\boldsymbol{\Sigma}_t^{(f)}$ are unknown, but the projection follows the distribution

$$\mathbf{x}_t = \mathbf{H}\mathbf{z}_t \sim \mathcal{N}\left(\mathbf{H}\boldsymbol{\mu}_t^{(f)}, \mathbf{H}\boldsymbol{\Sigma}_t^{(f)}\mathbf{H}^\top\right), \qquad \mathbf{z}_t \sim \alpha_t(\mathbf{z}_t). \tag{3.43}$$

We can find equations for the missing parameters by combining information from two sources: the (projected) prediction, cf. eq. (3.41),

$$\mathbf{x}_t = \mathbf{H}\mathbf{z}_t \sim \mathcal{N}\left(\mathbf{H}\boldsymbol{\mu}_t^{(p)}, \mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top\right), \quad \mathbf{z}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_t^{(p)}, \boldsymbol{\Sigma}_t^{(p)}\right), \tag{3.44}$$

and the noisy measurement[3]

$$\mathbf{x}_t \sim \mathcal{N}(\hat{\mathbf{x}}_t, \mathbf{R}). \tag{3.45}$$

The fusion of these two sources of information is performed by multiplying the respective densities, as is depicted in fig. 3.4.

The product of densities of two Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$ and $\mathcal{N}(\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b)$ is proportional to the density of another Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$, with

$$\boldsymbol{\mu}_c = \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_a(\boldsymbol{\Sigma}_a + \boldsymbol{\Sigma}_b)^{-1}(\boldsymbol{\mu}_b - \boldsymbol{\mu}_a), \tag{3.46}$$

$$\boldsymbol{\Sigma}_c = \left(\boldsymbol{\Sigma}_a^{-1} + \boldsymbol{\Sigma}_b^{-1}\right)^{-1} \tag{3.47}$$

$$= \boldsymbol{\Sigma}_a(\boldsymbol{\Sigma}_a + \boldsymbol{\Sigma}_b)^{-1}\boldsymbol{\Sigma}_b. \tag{3.48}$$

By identifying $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ with the projected filter, eq. (3.43), and the two sources of information with $\mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$ and $\mathcal{N}(\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b)$, respectively, eqs. (3.46) and (3.47) provide us with a system of equations that we can solve for the missing parameters $\boldsymbol{\mu}_t^{(f)}$ and $\boldsymbol{\Sigma}_t^{(f)}$:

$$\mathbf{H}\boldsymbol{\mu}_t^{(f)} = \mathbf{H}\boldsymbol{\mu}_t^{(p)} + \mathbf{H}\mathbf{K}\left(\hat{\mathbf{x}}_t - \mathbf{H}\boldsymbol{\mu}_t^{(p)}\right) \tag{3.49}$$

$$= \mathbf{H}\left(\boldsymbol{\mu}_t^{(p)} + \mathbf{K}\left(\hat{\mathbf{x}}_t - \mathbf{H}\boldsymbol{\mu}_t^{(p)}\right)\right) \tag{3.50}$$

$$\mathbf{H}\boldsymbol{\Sigma}_t^{(f)}\mathbf{H}^\top = \left(\left(\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top\right)^{-1} + \mathbf{R}^{-1}\right)^{-1} \tag{3.51}$$

$$= \mathbf{H}\left(\boldsymbol{\Sigma}_t^{(p)} - \mathbf{K}\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\right)\mathbf{H}^\top \tag{3.52}$$

with

$$\mathbf{K} \equiv \boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top\left(\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top + \mathbf{R}\right)^{-1}. \tag{3.53}$$

The step in eqs. (3.51) and (3.52) is detailed in appendix A.4.

Since $\mathbf{H}$ is invertible, we can immediately solve eqs. (3.50) and (3.52) by multiplying with appropriate inverses from left and right:

$$\boldsymbol{\mu}_t^{(f)} = \boldsymbol{\mu}_t^{(p)} + \mathbf{K}\left(\hat{\mathbf{x}}_t - \mathbf{H}\boldsymbol{\mu}_t^{(p)}\right), \tag{3.54}$$

$$\boldsymbol{\Sigma}_t^{(f)} = \boldsymbol{\Sigma}_t^{(p)} - \mathbf{K}\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}. \tag{3.55}$$

Put together, eqs. (3.53) to (3.55) form the update step of the Kalman filter. Combined with the prediction step, eq. (3.41), this yields the Kalman filter algorithm summarized in algorithm 1.

3 Here, the notation distinguishes the *random variable* $\mathbf{x}_t$ from the *measured value* $\hat{\mathbf{x}}_t$.

Algorithm 1: Kalman Filter.

```
Input:  observations $\hat{\mathbf{x}}_{1:T}$;
        LGS parameters: initial state distribution parameters
        $\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1$, matrices $\mathbf{A}, \mathbf{Q}, \mathbf{H}, \mathbf{R}$, cf. eqs. (3.35) to (3.37)
Output: filter distributions $\alpha_t(\mathbf{z}_t) \sim \mathcal{N}\left(\boldsymbol{\mu}_t^{(f)}, \boldsymbol{\Sigma}_t^{(f)}\right)$

Initialize $\boldsymbol{\mu}_0^{(f)} = \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_0^{(f)} = \boldsymbol{\Sigma}_1$.
for $t = 1, \dots, T$ do
    Predict
        $\boldsymbol{\mu}_t^{(p)} = \mathbf{A}\boldsymbol{\mu}_{t-1}^{(f)}$,
        $\boldsymbol{\Sigma}_t^{(p)} = \mathbf{A}\boldsymbol{\Sigma}_{t-1}^{(f)}\mathbf{A}^\top + \mathbf{Q}$.
    Compute Kalman gain
        $\mathbf{K} = \boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top\left(\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top + \mathbf{R}\right)^{-1}$.
    Update
        $\boldsymbol{\mu}_t^{(f)} = \boldsymbol{\mu}_t^{(p)} + \mathbf{K}\left(\hat{\mathbf{x}}_t - \mathbf{H}\boldsymbol{\mu}_t^{(p)}\right)$,
        $\boldsymbol{\Sigma}_t^{(f)} = \boldsymbol{\Sigma}_t^{(p)} - \mathbf{K}\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}$.
end
```

A MINIMAL EXAMPLE To shed some more light on the Kalman filter, it is worth looking at a minimal example, a scalar latent random walk with white noise in the emission model:

$$p(z_t \mid z_{t-1}) = \mathcal{N}\left(z_t \mid z_{t-1}, \sigma_p^2\right), \tag{3.56}$$

$$p(x_t \mid z_t) = \mathcal{N}\left(x_t \mid z_t, \sigma_e^2\right). \tag{3.57}$$

That is, $\mathbf{A} = \mathbf{H} = 1$. Then eqs. (3.41) and (3.53) to (3.55) translate to

$$\mu_t^{(p)} = \mu_{t-1}, \tag{3.58}$$

$$\sigma_{t,(p)}^2 = \sigma_{t-1}^2 + \sigma_p^2, \tag{3.59}$$

$$k = \frac{\sigma_{t-1}^2 + \sigma_p^2}{\sigma_{t-1}^2 + \sigma_p^2 + \sigma_e^2} \in [0, 1], \tag{3.60}$$

$$\mu_t = \mu_{t-1} + k(x_t - \mu_{t-1}) = (1-k)\mu_{t-1} + kx_t, \tag{3.61}$$

$$\sigma_t^2 = \sigma_{t-1}^2 - k(\sigma_{t-1}^2 + \sigma_p^2). \tag{3.62}$$

With $k \in [0, 1]$, we see that the updated mean μ_t is a convex combination of the previous updated mean μ_{t-1} and the observation x_t. The factor k measures how much to trust the prediction vs. the new observation by relating the respective variances σ_p^2 and σ_e^2. If $\sigma_p^2 \ll \sigma_e^2$, i.e., the emission is much more trustworthy than the prediction, then $k \to 1$, and conversely if $\sigma_p^2 \gg \sigma_e^2$ then $k \to 0$.

FURTHER REMARKS $\mathbf{K}$ is the so-called *Kalman gain*. Its name is rooted in the control community where the Kalman filter originates.

Loosely speaking, a gain matrix quantifies the magnitude of effect a control input has on the system. Analogously, the Kalman gain quantifies how much effect the deviation $\hat{\mathbf{x}}_t - \mathbf{H}\boldsymbol{\mu}_t^{(p)}$ between observation and predicted observation has on the belief update, as is indicated by eq. (3.54) and highlighted by the scalar example.

It is worth stressing again that the detour to observation space is not necessary to arrive at this solution from a purely algebraic point of view. However, the concept of fusing different sources of information by multiplying their densities gives a good intuition of the Kalman filtering algorithm. It applies to similar scenarios: for instance, if there is no prediction but several independent sensor measurements of the same quantity, each with their individual noise model, principled sensor fusion can be achieved by multiplying the respective densities according to the same rules. This yields a weighted mean of the sensor readings, weighted with the relative trust in the individual sensors. This concept of fusion will be a building block of *deep variational Bayes filters* (DVBFs) in section 4.2.

The Kalman filter has been extended to nonlinear scenarios, most notably by the *extended* Kalman filter (Mc Gee et al., 1962) and the *unscented* Kalman filter (Julier and Uhlmann, 1997). Further, each of these algorithms can be extended to the smoothing scenario, pioneered by Rauch et al. (1965). The interested reader is once again referred to Sarkka (2013).

3.3.2.2 *Particle Filters*

Kalman filters fall on one end of the spectrum of Bayesian posterior algorithms in that they introduce strong assumptions, but deliver closed-form solutions to usually intractable inference problems. *Particle filters* in some sense fall on the opposite end of this spectrum. They attempt to reduce assumptions on the system to a minimum while still providing posterior samples and estimates.

Once again, we revisit the prediction-update recursion, eq. (3.17),

$$\alpha_t(\mathbf{z}_t) \propto p(\mathbf{x}_t \mid \mathbf{z}_t) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \alpha_{t-1}(\mathbf{z}_{t-1}) \, d\mathbf{z}_{t-1} \tag{3.63}$$

$$= p(\mathbf{x}_t \mid \mathbf{z}_t) \mathbb{E}_{\alpha_{t-1}(\mathbf{z}_{t-1})}[p(\mathbf{z}_t \mid \mathbf{z}_{t-1})]. \tag{3.64}$$

Particle filters are built upon the following idea: the filter α_t is recursively known up to a normalizing constant, and in general we cannot sample it. In section 1.2.1, we found how to obtain weighted samples via importance sampling in such a scenario.

BASIC ALGORITHM Since the unnormalized pdf of the filter depends on an expectation w. r. t. the previous filter, eq. (3.64), we sample recursively. We start from a weighted batch of particles representing α_1. Then we perform particle-wise prediction, which corresponds to an IS estimate of $\mathbb{E}_{\alpha_{t-1}(\mathbf{z}_{t-1})}[p(\mathbf{z}_t \mid \mathbf{z}_{t-1})]$. Last we update the weights

Algorithm 2: Particle Filter.

```
Input: observations $\mathbf{x}_{1:T}$; number of particles $P$;
       SSM initial, transition, and emission distribution;
       proposal distributions $\pi(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t})$;
       resampling criterion and technique
Output: filtered, weighted state particle trajectories
       $\left\{\left\{\left(w_t^{(p)}, \mathbf{z}_{1:t}^{(p)}\right)\right\}_{p=1,\ldots,P}\right\}_{t=1,\ldots,T}$

Initialize $w_0^{(p)} = 1/P, p = 1, \ldots, P$
for $t = 1, \ldots, T$ do
    for $p = 1, \ldots, P$ do
        Proposals $\mathbf{z}_t^{(p)} \sim \pi\left(\mathbf{z}_t \,\middle|\, \mathbf{z}_{1:t-1}^{(p)}, \mathbf{x}_{1:t}\right)$
        Updates $\gamma_t^{(p)} = p\left(\mathbf{x}_t \middle| \mathbf{z}_t^{(p)}\right) p\left(\mathbf{z}_t^{(p)} \middle| \mathbf{z}_{t-1}^{(p)}\right) / \pi\left(\mathbf{z}_t^{(p)} \middle| \mathbf{z}_{1:t-1}^{(p)}, \mathbf{x}_{1:t}\right)$
        Unnorm. weights $\hat{w}_t^{(p)} = w_{t-1}^{(p)} \gamma_t^{(p)}$
    end
    Renormalize weights $w_t^{(p)} = \hat{w}_t^{(p)} / \sum_{r=1}^{P} \hat{w}_t^{(r)}$
    if criterion is met then
        $\left\{\left(w_t^{(p)}, \mathbf{z}_{1:t}^{(p)}\right)\right\}$ = resample$\left(\left\{\left(w_t^{(p)}, \mathbf{z}_{1:t}^{(p)}\right)\right\}\right)$
    end
end
def resample$\left(\left\{\left(w_t^{(p)}, \mathbf{z}_{1:t}^{(p)}\right)\right\}\right)$:
    Save temporary copy $\hat{\mathbf{z}}_{1:t}^{(p)} = \mathbf{z}_{1:t}^{(p)}$
    Updated particle indexes $\{i_p\}$ = technique$\left(\left\{w_t^{(p)}\right\}\right)$
    for $p = 1, \ldots, P$ do
        Set $w_t^{(p)} = 1/P$
        Overwrite trajectories $\mathbf{z}_{1:t}^{(p)} = \hat{\mathbf{z}}_{1:t}^{(i_p)}$
    end
```

with the emission model. A more technical description is presented in algorithm 2.[4]

The particle filter is an implementation of a more general class of algorithms, *sequential importance sampling* (SIS). In this context, the samples $\mathbf{z}_t^{(i)}$ are referred to as *particles*, hence *particle filters*.

Since we assumed nothing beyond tractability of all pdfs of all components of the SSM and proposals that can be sampled, particle filters are a very versatile example of a Bayesian posterior algorithm. This versatility comes with strings attached, which we will discuss in the following.

4 We will not cover actual implementation details, such as operating in log space and making use of the *logsumexp* function.

NORMALIZING CONSTANTS Part of the appeal of particle filters is that we only need to know the filtering distribution up to a normalizing constant. At the same time, section 3.3 informs us that the normalizing constant Z_t of the filter $\alpha_t(\mathbf{z}_t)$ is $p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})$ so that

$$\prod_{t=1}^{T} Z_t = p(\mathbf{x}_{1:T}). \tag{3.65}$$

In eq. (1.20), we saw that the normalizing constant can be estimated as the average unnormalized weight. One can thus obtain an unbiased estimator of the joint marginal likelihood of the observations as a byproduct of particle filtering via

$$p(\mathbf{x}_{1:T}) \approx \prod_{t=1}^{T} \underbrace{\sum_{p=1}^{P} \hat{w}_t^{(p)}}_{\approx Z_t}. \tag{3.66}$$

A more detailed discussion of this estimator including a proof of unbiasedness even in the case of resampling can be found in Maddison et al. (2017).

PARTICLE DEGENERATION Algorithm 2 assumes a generic proposal $\pi(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t})$. Often, this is simplified to $\pi(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$ to guarantee an efficient online algorithm. It is even possible to implement a so-called *bootstrap* particle filter, which uses the *prior* transition model as a proposal. Many of the caveats of importance sampling transfer to sequential importance sampling. In particular, a successful implementation of a particle filter depends on a suitable proposal distribution.

This leads to the phenomenon of *particle degeneration*. Empirically, the set of particles tends to collapse after a certain time horizon, in the sense that only a small fraction of particles have a non-negligible weight. That implies a low ESS, which in turn is connected to high-variance estimators. To counter this phenomenon, SIS incorporates optional resampling steps. Generally speaking, the resampling step discards particles with negligible weights in favor of copies of particles with non-negligible weight. SIS with resampling is often called *sequential importance resampling* (SIR).

RESAMPLING TECHNIQUES In this work, we make use of two different resampling techniques. The first and arguably most common technique is *Categorical* resampling. The particles are resampled by drawing P indexes i. i. d. from a Categorical distribution with probabilities equal to the normalized weights. This way, particles with negligible weight are unlikely to survive.

The second technique is called *stochastic universal sampling* (SUS; Baker, 1987). Again, we draw P random indexes, this time not i. i. d. but correlated: where Categorical resampling allows the improbable

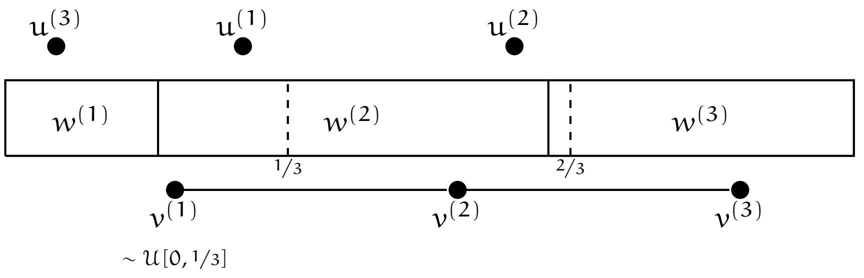


Figure 3.5: Categorical vs. *stochastic universal sampling* (SUS) with $P = 3$ particles. Weights $w^{(i)}$ correspond to their block length. Categorical sampling draws three i. i. d. samples $u^{(i)} \sim \mathcal{U}[0, 1]$. The third particle is not resampled despite having twice the weight of the first particle. SUS samples only one random sample $v^{(1)} = u \sim \mathcal{U}[0, 1/3]$, then adds 1/3 and 2/3 to arrive at $v^{(2)}$ and $v^{(3)}$, respectively. Particles with weight larger than 1/3 survive.

Algorithm 3: Categorical Resampling.

```
Input: current weights $\left\{w_t^{(p)}\right\}_{p=1,\dots,P}$
Output: resampled indexes $\{i_p\}_{p=1,\dots,P}$

for $p = 1, \dots, P$ do
    Sample $u_t^{(p)} \sim \mathcal{U}[0, 1]$.
    Set $i_p = \arg\min_{K \in \{1,\dots,P\}} \sum_{k=1}^{K} w_t^{(k)} > u_t^{(p)}$.
end
```

scenario that, e. g., the particle with highest weight is not among the resampled, SUS guarantees that every particle with $w_t^{(p)} > K/P$ for some $K \in \mathbb{N}$ is at least K times among the resampled particles.

The two techniques are shown in fig. 3.5 and algorithms 3 and 4.

RESAMPLING STRATEGIES Beyond the *technique*, we need to specify a *criterion* upon which to resample. An easy default would be to resample at every step, but this suffers from at least two drawbacks.

Firstly, resampling may be costly, so it may be advisable to limit resampling to a necessary minimum.

Secondly, particle filters have to maintain a posterior belief with a finite amount of particles. That is, if the belief contains multiple modes, each mode is in turn represented by an even lower amount of particles. Assume the following thought experiment: a particle filter with two particles ($P = 2$, arguably an audacious choice) and weights $w_t^{(1)} = w_t^{(2)} = 1/2$. Each particle represents a distinct mode of the posterior belief. Categorical resampling will delete one of the modes with a probability of 50 %. More generally, resampling too frequently can run the risk of deleting modes without need.

In this extreme scenario, the ESS is at its maximum P, which indicates that particle degeneration does not show. Indeed, a very common

Algorithm 4: Stochastic Universal Sampling (Baker, 1987).

```
Input: current weights $\left\{w_t^{(p)}\right\}_{p=1,\dots,P}$
Output: resampled indexes $\{i_p\}_{p=1,\dots,P}$

Sample $u \sim \mathcal{U}[0, 1/P]$.
for $p = 1, \dots, P$ do
    Set $v_t^{(p)} = u + (p-1)/P$.
    Set $i_p = \arg\min_{K \in \{1,\dots,P\}} \sum_{k=1}^{K} w_t^{(k)} > v_t^{(p)}$.
end
```

default criterion is for the ESS to drop below a threshold, often $P/2$. Empirically, this has shown to be a good compromise that avoids particle degeneration on the one hand and unprompted mode collapse on the other.

SIR tends to suffer less from particle degeneration.

3.4 THE SEQUENTIAL ELBO

The following two parts of this thesis revolve around applying the principles of VAEs to sequential LVMs. While we save the details of these adaptations to the later chapters, it is worth inspecting the natural objective function, the sequential ELBO

$$\mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}[\ln p(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T})] - \mathrm{KL}(q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) \parallel p(\mathbf{z}_{1:T})), \tag{3.67}$$

as the basis for all later discussion. It is superficially very similar to the static ELBO discussed in chapter 2. Since all three joint distributions in eq. (3.67) factorize differently, however, we inspect the ELBO closer under SSM assumptions as discussed in the previous sections.

LOG LIKELIHOOD

$$\mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}[\ln p(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T})] \tag{3.68}$$

$$= \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}[\ln p(\mathbf{x}_t \mid \mathbf{z}_t)] \qquad \text{(by eq. (3.6))} \tag{3.69}$$

$$= \sum_{t=1}^{T} \mathbb{E}_{\underline{q(\mathbf{z}_t|\mathbf{x}_{1:T})}}[\ln p(\mathbf{x}_t \mid \mathbf{z}_t)]. \tag{3.70}$$

The marginal in the last step is explained by the observation

$$\mathbb{E}_{p(\mathbf{x},\mathbf{y})}[f(\mathbf{x})] = \iint p(\mathbf{x},\mathbf{y}) f(\mathbf{x}) \, d\mathbf{y} \, d\mathbf{x} \tag{3.71}$$

$$= \int p(\mathbf{x}) f(\mathbf{x}) \int p(\mathbf{y} \mid \mathbf{x}) \, d\mathbf{y} \, d\mathbf{x} = \int p(\mathbf{x}) f(\mathbf{x}) \, d\mathbf{x} \tag{3.72}$$

$$= \mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})] \tag{3.73}$$

for in this case arbitrary random variables $\mathbf{x}$ and $\mathbf{y}$. That is, if variables in the joint distribution are not used in the function f, one may revert to an appropriate marginal. This is a general observation that does not impose assumptions on q.

Typically, we do not compute the expectations in closed form; rather, we estimate them with Monte Carlo sampling. Samples from the marginal can always be obtained by sampling the joint and dropping the superfluous variables. Thus, a simplification to the marginal is interesting only if samples of the marginal are in some way easier to obtain, e. g., with lower computational effort.

The likelihood is sometimes also called *reconstruction error*. This is motivated by the often-used Gaussian likelihood model. The mean $\mu(\mathbf{z}_t)$ can be interpreted as a reconstruction of the original input $\mathbf{x}_t$, and the Gaussian log pdf is essentially a Mahalanobis distance error between reconstruction and datum.

PRIOR DIVERGENCE

$$\begin{aligned}
&\mathrm{KL}(q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) \,\|\, p(\mathbf{z}_{1:T})) && (3.74)\\
&= \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\sum_t \ln \frac{q(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t+1:T})}{p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)}\right] && (3.75)\\
&= \sum_t \mathbb{E}_{q(\mathbf{z}_t,\mathbf{z}_{t+1}|\mathbf{x}_{1:T})}\left[\ln \frac{q(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t+1:T})}{p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)}\right] && (3.76)\\
&= \sum_t \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{1:T})}[\mathrm{KL}(q(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t+1:T}) \,\|\, p(\mathbf{z}_{t+1} \mid \mathbf{z}_t))]. && (3.77)
\end{aligned}$$

Here, the second line assumes that $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$ decomposes like the true posterior of an SSM. The prior KL thus decomposes into a sum of *expected* transition KLs.[5]

Even this very elemental analysis of the sequential ELBO hints at many different possible estimators. As we will see in parts II and III, further assumptions on the approximate posterior $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$ will lead to different algorithm designs.

Jointly looking at eqs. (3.70) and (3.77), we observe that the sequential ELBO decomposes into what can be described as *step-wise* ELBOs. Particularly the divergence term comes with an additional expectation, though. The widespread use of single-sample MC estimates of such expectations further blurs this distinction.

We examined the ELBO under SSM assumptions; similar principles apply under different assumptions of the LVM, cf. chapter 4.

5 For brevity, we have not considered $t = 1$ separately in eqs. (3.74) to (3.77). Due to the lack of ancestors, it is not an expected KL.

Part II

AUTO-ENCODING STATE-SPACE MODELS

This part is based on ideas that have appeared previously in the following publications and working papers:

Karl, Maximilian, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt (2017). "Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=HyTqHL5xg.

Akhundov, Adnan, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt (2019). *Variational Tracking and Prediction with Generative Disentangled State-Space Models*. arXiv: 1910.06205 [cs, stat]. URL: http://arxiv.org/abs/1910.06205.

The former is a shared contribution between Maximilian Karl and the author. Where his thesis (Karl, 2020) explained the more algorithmic implementation side, chapter 4 will complement the relation to prior and concurrent related work.

The latter is a collaboration between Adnan Akhundov and the author. Concept and model were conceived by the author. Experimental setup and execution along with modifications to the original model resulted in Adnan's Master thesis (Akhundov, 2018). Chapter 5 will focus on the connections between the two papers and the conclusions they allow for future work.

Direct quotes from the publications are highlighted in gray font color. Minor adaptations of the quotes to the style of this thesis are not explicitly highlighted.

Supplementary material is collected in appendix B.

4 NEURAL STATE-SPACE MODELS

In part I, we discussed VAEs and how to use them for unsupervised learning of generative LVMs. Then we discussed the peculiarities of sequential LVMs, in particular SSMs. In this chapter, we combine these ideas to understand how to learn SSMs with unsupervised algorithms, only from sampled observations

$$\mathcal{D} = \left\{\mathbf{x}_{1:T}^{(i)}\right\}_{i=1,\dots,N}. \tag{4.1}$$

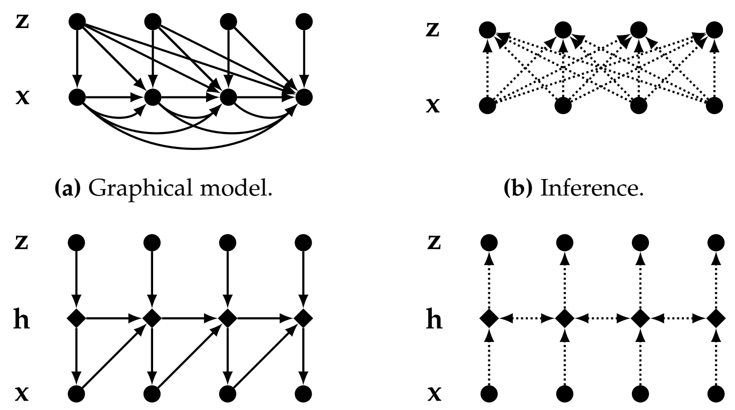
In VAEs, we assumed some relatively simple prior distribution $p(\mathbf{z})$ along with a likelihood density network $p(\mathbf{x} \mid \mathbf{z})$. Taken together, they form the LVM $p(\mathbf{x}, \mathbf{z})$. In the following, we will adapt this approach to sequential LVMs $p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$, with a strong focus on SSMs.

In some sense, this can be considered a special case of VAEs. As chapter 3 has shown, sequential LVMs add additional wrinkles over LVMs, in particular in relation to Bayesian inference. These require careful adaptation in the VAE framework.

Where the prior $p(\mathbf{z})$ in VAEs is typically fixed to a simple distribution and we only learn $p(\mathbf{x} \mid \mathbf{z})$ to complete the LVM, with sequential LVMs we are also interested in the prior $p(\mathbf{z}_{1:T})$. For instance, for SSMs the prior captures the entire dynamics of the system, so it is arguably even the more interesting component to be learned from data compared to the emission model

$$p(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_t). \tag{4.2}$$

EXTENDED GRAPHICAL MODELS In this chapter, we will discuss several neural implementations of sequential LVMs, often with an associated figure showing the graphical model, figs. 4.1, 4.2, 4.4, and 4.7 to 4.10. The pure graphical model, cf. section 1.1.3, is often insufficient to highlight important implementation aspects. We thus extend the graphical model by additional nodes: circular nodes (●) represent random variables or vectors; diamond nodes (◆) represent deterministic quantities like hidden variables of RNNs. The pure graphical model can be recovered by purging the deterministic nodes and adding edges between circular nodes if there existed a directed path between them via only deterministic nodes. We distinguish between the generative model $p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$ and the inference model $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$ by using solid graph edges ($\longrightarrow$) for the former and dotted edges ($\dashrightarrow$) for the latter. If parts of the generative model are reused in the inference model, the respective edges in the inference model will also be solid.



(a) Graphical model.

(b) Inference.

(c) Graphical model extended with computational nodes.

(d) Inference model extended with computational nodes.

Figure 4.1: *Top:* graphical LVM of *stochastic recurrent networks* (STORNs). *Bottom:* extended graphical models, with hidden states **h** of the RNN included. The hidden states between inference and generation are not shared.

4.1 SEQUENTIAL VARIATIONAL AUTO-ENCODERS

Shortly after the introduction of VAEs—and before tackling SSMs—variants using recurrent networks were proposed to handle sequential data. The first to do so were Bayer and Osendorfer (2014) with *stochastic recurrent networks* (STORNs).

STORNs are arguably best understood as the most immediate translation of the auto-encoding approach to inference from the static to the dynamic case: the feed-forward NNs of VAEs are replaced with RNNs. The inference density network is interpreted to return the approximation

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t \mid \mathbf{x}_{1:T}) \tag{4.3}$$

or, if the inference RNN is not bidirectional,

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t \mid \mathbf{x}_{1:t}). \tag{4.4}$$

That is, the states are assumed independent across time given observations. This constitutes a fairly strong assumption but makes maximally efficient use of standard RNN architectures for inference.

The learned LVM exploits RNN architectures equally efficient. The prior again assumes independence across time[1],

$$p(\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{z}_t). \tag{4.5}$$

1 More flexible priors have later been suggested by the author (Soelch et al., 2016).

In analogy to VAEs, the individual priors $p(\mathbf{z}_t)$ are assumed to be simple i. i. d. Gaussian distributions, usually with diagonal or even isotropic covariance.

The emission model is then implemented by a unidirectional RNN which at each time step takes the current state $\mathbf{z}_t$ and the previous emission $\mathbf{x}_{t-1}$ as input and returns the conditional

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{x}_{1:t-1}) \tag{4.6}$$

so that

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{z}_t) p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{x}_{1:t-1}). \tag{4.7}$$

The conditional in eq. (4.6) is best understood in terms of the (deterministic) hidden states $\mathbf{h}_{1:T}$ of the RNN , i. e.,

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{x}_{1:t-1}) = p(\mathbf{x}_t \mid \mathbf{h}_t(\mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{h}_{t-1})). \tag{4.8}$$

The graphical model and its extended version are depicted in fig. 4.1.

From this lens, STORNs may be viewed as step-wise VAEs where the encoders and decoders are linked through respective recurrent cells. The largest deviation from this point of view is a short-cut connection used only at training time. The feedback of emission $\mathbf{x}_{t-1}$ in eq. (4.8) is replaced by feeding the true observation from the data set to guide learning.

This notion of *step-wise* VAEs is further stressed by the the observation that the ELBO decomposes into step-wise terms. This is particularly true when using single-sample MC estimates of expectations, as is custom in the field, cf. also section 3.4.

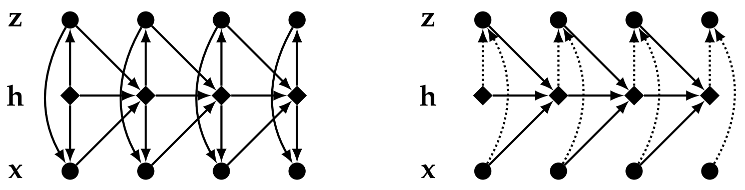
For further details, the reader is also referred to the thesis by Bayer (2015).

4.1.1 Variational Recurrent Neural Networks

STORNs are an interesting bridge in the history of ideas from VAEs on the one side and SSMs on the other. They have been discussed as such in the previous section. For the remainder of this thesis STORNs are of lesser interest. We close their discussion by briefly acknowledging further non-SSM LVMs that have been presented and applied in the wake of Bayer and Osendorfer (2014).

Most notably, Chung et al. (2015) suggest *variational recurrent neural networks* (VRNNs). VRNNs additionally make the prior depend on the hidden state of the generative RNN,

$$p(\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{z}_t \mid \mathbf{h}_t(\mathbf{z}_{t-1}, \mathbf{x}_{t-1}, \mathbf{h}_{t-1})), \tag{4.9}$$



(a) Graphical model extended with computational nodes.

(b) Inference model extended with computational nodes.

Figure 4.2: Graphical and inference models of *variational recurrent neural networks* (VRNNs) extended with deterministic hidden RNN states. The hidden states between inference and generation are shared, as indicated by the solid lines.

cf. fig. 4.2. In terms of connectivity in the graphical model, this is the most general sequential LVM possible. Further, the inference network and the generative RNN share weights. This coupling can be seen as an inductive bias for learning but prevents bidirectional inference as in STORNs, as has been addressed by Goyal et al. (2017).

4.2 DEEP VARIATIONAL BAYES FILTERS

In the previous section, we have highlighted attempts at learning models that strike a balance between widespread applicability—reflected by the near-complete graphical models—and efficient computability, exploiting RNNs to maximal extent. A disadvantage of this approach is that it is nigh impossible to separate the LVM as a probabilistic model on the one hand from its concrete neural implementation on the other hand, as is highlighted by the extended graphical models in figs. 4.1 and 4.2.

Considering their widespread use in control theory and the engineering disciplines, it was only a matter of time before the VAE framework would be applied to SSMs. The first attempt is due to Watter et al. (2015), dubbed *embed to control* (E2C). E2C attempts to learn a useful latent-space transition for subsequent control. The learning algorithm is inspired by VAEs, but the learned model is not strictly an SSM and the objective function needs to be augmented by several regularizing terms.

From the start, the motivation behind our contribution, *deep variational Bayes filters* (DVBFs), was a possible application to model-predictive control.[2] This dictates a core goal: learning a good prior, specifically a good transition density network $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$. The state needs to carry all relevant information so as to devise good policies. It is the *prior* that matters to us.

2 This would later be realized in Karl et al. (2017b), Becker-Ehmck et al. (2020), and Karl (2020).

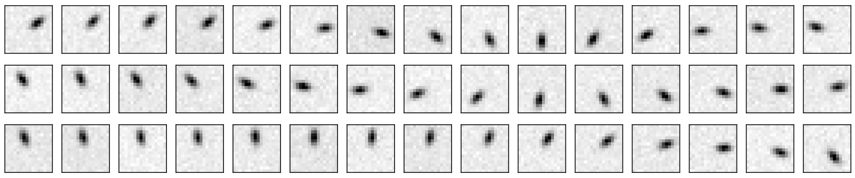


Figure 4.3: Pendulum example data as used in Karl et al. (2017a). Each row shows the observations $\mathbf{x}_{1:15}$ of a single training sequence from left to right.

In the literature preceding DVBFs, in particular STORN, VRNN, or E2C, we notice two patterns that hinder this. The first is the use of RNNs. Since RNNs have their own deterministic hidden states $\mathbf{h}_{1:T}$, the responsibility for carrying all relevant information is split between $\mathbf{z}_{1:T}$ and $\mathbf{h}_{1:T}$. The fact that the latter are not regularized by, e. g., the prior KL term of the ELBO undermines the role of the latent states $\mathbf{z}_{1:T}$.

The second pattern is that the prior and with it the transition only occur in the prior KL term of the ELBO. This means that the prior can only learn dynamics from data if the approximate posterior also does—otherwise more accurate dynamics *increase* the prior KL. Unfortunately, this is not guaranteed as the approximate posterior may focus on intra-step correlations, which may be sufficient for maximizing the ELBO.

To understand this, consider the synthetic pendulum camera data as depicted in fig. 4.3 (Karl et al., 2017a). The state space of the pendulum consists of two quantities, angle and angle velocity. The observation space even at this very moderate image resolution consists of 256 dimensions. An ill-posed model

$$p(\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{x}_t) = \prod_{t=1}^{T} \int p(\mathbf{x}_t, \mathbf{z}_t)\, d\mathbf{z}_t, \tag{4.10}$$

i. e., a *frame-wise* i. i. d. LVM, is very capable of representing the frames well (Klushyn et al., 2019). The angle alone is sufficient to describe the essence of each frame. The marginal benefit in terms of the ELBO of uncovering the dynamics in the form of the angular velocity is slim. Such a local minimum is hard to escape even for dynamic models.

These two patterns dictate two core design principles of DVBFs:

1. The approximate posterior $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$ should reuse the transition $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$.

2. To consolidate their singular role, the latent states $\mathbf{z}_{1:T}$ should be the only variables passing information through time.

The second point in particular dictates that the respective density networks should not be stateful in the sense that they involve deterministic dynamic hidden variables $\mathbf{h}_t$ like recurrent cells that would dilute the central role of the state $\mathbf{z}_t$. As section 4.3 shows, this is a common technique in concurrent and later approaches. Further, the downstream use in a control scenario motivated using a filtering approximate posterior,

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t). \tag{4.11}$$

Here, the factors have already been simplified according to SSM assumptions. This is a technically incorrect assumption motivated by the application. We discuss its ramifications in sections 4.2.3 and 4.4, and we later deconstruct it in chapter 6. Inserting into the sequential ELBO, eq. (3.67), we get

$$\sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}[\ln p(\mathbf{x}_t \mid \mathbf{z}_t) - \mathrm{KL}(q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t) \parallel p(\mathbf{z}_t \mid \mathbf{z}_{t-1}))]. \tag{4.12}$$

As is custom, the expectation is approximated by MC integration with a single sample, obtained via ancestral sampling according to eq. (4.11). Estimating the ELBO thus adds the two design constraints already known from VAEs: firstly, sampling the posterior, in this case the conditional distributions $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$; secondly, evaluating its pdf so as to at least estimate the prior KL term.

This sets us up to discuss deep variational Bayes filters. There exist two major variants, residual DVBFs and fusion DVBFs.[3]

4.2.1 Residual DVBF

The core idea of residual DVBFs is a residual formulation

$$\mathbf{z}_t = f(\mathbf{z}_{t-1}) + \boldsymbol{\epsilon}_t(\mathbf{z}_{t-1}) \tag{4.13}$$

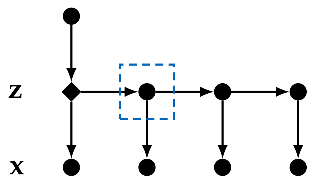
of the latent transition $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ with a *deterministic* function f and *centered* noise variables $\boldsymbol{\epsilon}_t$ that are independent across time. They are usually either identically distributed, $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \boldsymbol{\Sigma})$, or covariances are a function of the previous state, $\boldsymbol{\epsilon}_t(\mathbf{z}_{t-1}) \sim \mathcal{N}(0, \boldsymbol{\Sigma}(\mathbf{z}_{t-1}))$. The transition is implicitly defined as

$$p(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \sim \mathcal{N}(f(\mathbf{z}_{t-1}), \boldsymbol{\Sigma}_t). \tag{4.14}$$

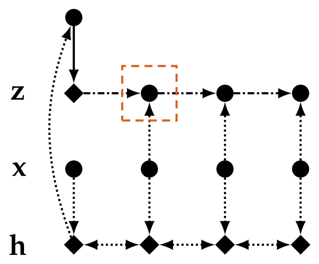
This formulation has the advantage that the approximate posterior transition $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$ can also be defined implicitly via a residual formulation:

$$\mathbf{z}_t = f(\mathbf{z}_{t-1}) + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim q(\boldsymbol{\epsilon}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t). \tag{4.15}$$

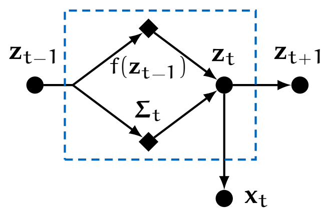
3 This chapter is based on residual DVBFs (Karl et al., 2017a). Fusion DVBFs were developed later in Karl et al. (2017b). The author's contributions to the latter paper are out of scope of this thesis. The term fusion DVBFs was coined by Becker-Ehmck et al. (2019); the term residual DVBFs is used in this thesis to distinguish the two variants.



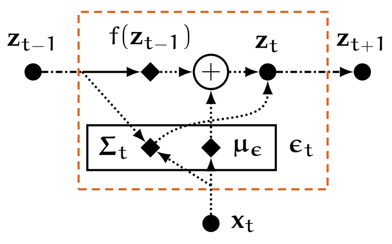
(a) Extended graphical model of DVBF.



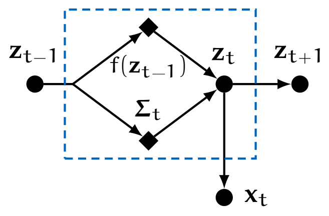
(b) Inference model of DVBF.



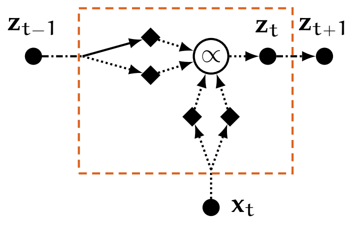
(c) Residual DVBF prior transition.



(d) Approximate posterior transition of residual DVBF.



(e) Fusion DVBF prior transition.



(f) Approximate posterior transition of fusion DVBF.

Figure 4.4: *Top*: extended graphical models and inference models of *deep variational Bayes filters* (DVBFs) for both variants. *Middle*: details for *residual* DVBFs. *Bottom*: details for *fusion* DVBFs. In the three inference plots, solid edges indicate a reuse of the respective prior component. Half-dotted edges (-·-·-·-►, cf. (b), (d), (f)) indicate partial reuse. Boxes in the top row correspond to boxes in the middle and bottom row. Plots (c) and (e) are identical.

The inference model $q(\boldsymbol{\epsilon}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$ is implemented by a density network. Prior and posterior density networks share the mean function f, which is made possible by reparametrization. The extended graphical and inference models are depicted in figs. 4.4a and 4.4b, with details on the prior and approximate posterior density networks in figs. 4.4c and 4.4d.

The residual formulation creates the desired coupling of prior and posterior, where both equally benefit of improvements of f. It also readily allows for estimating the ELBO. The approximate posterior $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$ can easily be sampled by ancestral sampling of $q(\boldsymbol{\epsilon}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$. This can be viewed from two perspectives. Either the states $\mathbf{z}_{1:T}$ are deterministic functions of the noise samples; or the particular process of combining the distribution parameters is part of the density network that implements $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$. The latter point of view is depicted in fig. 4.4d. The likelihood term can be estimated via MC integration. The prior KL term can be evaluated equally well as it is shift-invariant, i. e.,

$$\mathrm{KL}(q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t) \,\|\, p(\mathbf{z}_t \mid \mathbf{z}_{t-1})) \tag{4.16}$$
$$= \mathrm{KL}(q(\boldsymbol{\epsilon}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t) \,\|\, p(\boldsymbol{\epsilon}_t)). \tag{4.17}$$

The function f is generally an arbitrary function such as an MLP. In Karl et al. (2017a), a particular implementation

$$f(\mathbf{z}_{t-1}) = \mathbf{A}(\mathbf{z}_{t-1})\mathbf{z}_{t-1} \tag{4.18}$$

is suggested where a transition matrix $\mathbf{A}(\mathbf{z}_{t-1})$ is computed by mixing base matrices as a function of $\mathbf{z}_{t-1}$. The actual transition is then a matrix-vector product. This *fast-weight* approach (Schmidhuber, 1992; Ba et al., 2016) may be interpreted as returning a linearization of a nonlinear system in $\mathbf{z}_t$. This inductive bias is called a *locally linear* transition. For full details including Bayesian treatment of the base matrices (Blundell et al., 2015) and annealing techniques (Mandt et al., 2016), the reader is referred to Karl (2020).

4.2.2 Fusion DVBF

The key inspiration for fusion DVBFs is the prediction-update cycle

$$\alpha_t(\mathbf{z}_t) \propto p(\mathbf{x}_t \mid \mathbf{z}_t)\mathbb{E}_{\alpha_{t-1}(\mathbf{z}_{t-1})}[p(\mathbf{z}_t \mid \mathbf{z}_{t-1})] \tag{4.19}$$

known from standard Bayesian SSM posterior techniques, cf. section 3.3, which in this context translates to

$$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t) \propto p(\mathbf{x}_t \mid \mathbf{z}_t)\mathbb{E}_{q(\mathbf{z}_{t-1})}[p(\mathbf{z}_t \mid \mathbf{z}_{t-1})]. \tag{4.20}$$

The prediction step is easy to implement in typical style via single-sample MC integration. The immediate benefit of this algorithmic inspiration is the reuse of the prior transition.

The update step is less clear. The emission model $p(\mathbf{x}_t \mid \mathbf{z}_t)$ is a typically highly nonlinear density network, which makes closed-form normalization as with Kalman filters, section 3.3.2.1, difficult. Thus the rather pragmatic solution is to approximately *invert* the emission model $p(\mathbf{x}_t \mid \mathbf{z}_t)$ with a local inference density network $q(\mathbf{z}_t \mid \mathbf{x}_t)$ so that fusion takes place in latent space instead of observation space as with Kalman filters:

$$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t) \propto q(\mathbf{z}_t \mid \mathbf{x}_t)\mathbb{E}_{q(\mathbf{z}_{t-1})}[p(\mathbf{z}_t \mid \mathbf{z}_{t-1})]. \tag{4.21}$$

The benefit is that we can exploit the fusion mechanism for Gaussian distributions we established in eqs. (3.46) to (3.48) for closed-form normalization of eq. (4.21).[4] This fusion mechanism can be seen as a weighted average of a prediction and the local inference depending on their covariances, i. e., how much they can be believed.

Additionally, since $q(\mathbf{z}_t \mid \mathbf{x}_t)$ only makes use of the current observation $\mathbf{x}_t$, we achieve the goal of propagating information through time only via the transition.

Residual and fusion DVBFs are identical from a high level point of view, as is indicated by the extended graphical and inference model depictions in figs. 4.4a and 4.4b. Even the prior is identical, cf. figs. 4.4c and 4.4e. They largely differ in the details of the implementation of the approximate posterior transition $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}\, \mathbf{x}_t)$, as can be seen in figs. 4.4d and 4.4f.

The fact that Gaussian fusion strictly decreases variance combined with the single-sample MC integration of the prediction makes it necessary to decouple the variance of the prior $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ and the prediction $\mathbb{E}_{q(\mathbf{z}_{t-1})}[p(\mathbf{z}_t \mid \mathbf{z}_{t-1})]$. Ultimately, like with residual DVBFs, prior and posterior density networks share weights to produce the means but not the variances. As with residual DVBFs we spare these details and refer the reader to Karl (2020).

The resulting fusion mechanism bears resemblance to the concept of a *product of experts* (Hinton, 2002; Kurle et al., 2019). Products of experts apply to non-sequential LVMs with m (conditionally) independent measurements, the experts. That is, the emission model for $\mathbf{x} = \left(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\right)$ decomposes into factors,

$$p(\mathbf{x} \mid \mathbf{z}) = \prod_{i=1}^{m} p\left(\mathbf{x}^{(i)} \;\middle|\; \mathbf{z}\right), \tag{4.22}$$

in which case the posterior factorizes as

$$p(\mathbf{z} \mid \mathbf{x}) \propto p(\mathbf{z}) \prod_{i=1}^{m} \frac{p\left(\mathbf{z} \mid \mathbf{x}^{(i)}\right)}{p(\mathbf{z})}. \tag{4.23}$$

Like with fusion DVBF, the posterior is a product of densities for different sources of information on the latent variable.

4 This has the downside of restricting the variational family to those that allow for closed-form normalization like the family of Gaussian distributions.

4.2.3 Initial Inference

Both variants of DVBFs share a weakness we have ignored so far: inference of the initial state $\mathbf{z}_1$. Implementing the filtering scheme, eq. (4.11), by the letter would imply an initial inference model $q(\mathbf{z}_1 \mid \mathbf{x}_1)$. A single observation typically does not capture higher-order dynamical information, like velocities or forces. Again, the pendulum data in fig. 4.3 illustrate this point. The lack of information about the higher-order dynamic components of the state space would require very uncertain initial posteriors; $q(\mathbf{z}_1 \mid \mathbf{x}_1)$ essentially needs to fall back on the prior for all latent quantities that are not directly reflected by $\mathbf{x}_1$.

By itself this is not necessarily problematic, provided the density network $q(\mathbf{z}_1 \mid \mathbf{x}_1)$ is able to represent such distributions well; the uncertainty could be reduced in subsequent steps with new observations. Combined with the fact that the posterior beliefs are propagated through time by as few as a single sample, however, high uncertainty of $q(\mathbf{z}_1 \mid \mathbf{x}_1)$ is catastrophic: the probability to draw samples *only* from the wrong region in state space is high. This drastically increases the variance of ELBO estimates, which in turn translates to noisy stochastic gradients and negatively affects learning.

As a solution, both variants look at future observations to infer $\mathbf{z}_1$, i.e., $q(\mathbf{z}_1 \mid \mathbf{x}_1)$ is replaced by $q(\mathbf{z}_1 \mid \mathbf{x}_{1:\tau})$, which takes into account some horizon $\mathbf{x}_{1:\tau}$ with $1 < \tau \leqslant T$. The approximate posterior model of both variants is thus more correctly described as

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = q(\mathbf{z}_1 \mid \mathbf{x}_{1:\tau}) \prod_{t=2}^{T} q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t). \tag{4.24}$$

The initial inference model $q(\mathbf{z}_1 \mid \mathbf{x}_{1:\tau})$ is implemented as a density network, which could be feed-forward if the horizon τ is known and fixed a priori or recurrent (and possibly bidirectional) in the more general case. The latter case is depicted in the inference model in fig. 4.4b.

The necessity of this initial inference model is a testament to the intentionally misspecified posterior model, eq. (4.11), which, as discussed, is not faithful to the true SSM posterior factorization. In fact, through the backdoor it turns the *entire* inference model into a smoother. Since $q(\mathbf{z}_1 \mid \mathbf{x}_{1:\tau})$ depends on observations $\mathbf{x}_{1:\tau}$, so will all further inferred states $\mathbf{z}_{2:T}$.

From the application-driven point of view this compromise may still be desirable. It is not necessary to always feed all observations, $\tau = T$, as was originally done in Karl et al. (2017a). Indeed, at test time initial inference may be used during a smoothing burn-in phase after which DVBF—as designed—is used as a filter. A similar scheme is described in Becker-Ehmck et al. (2020).

There is a further problem for initial inference. State spaces are typically constrained, e. g., by mathematical or physical laws. In the

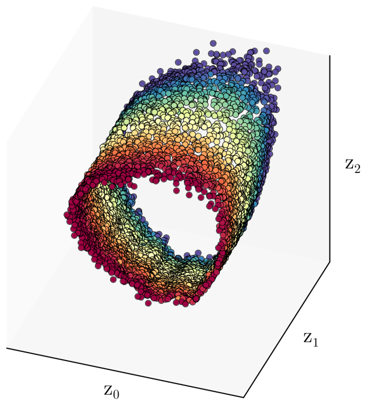


Figure 4.5: Test time samples of the approximate posterior $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$ of a DVBF trained on the image pendulum data presented in fig. 4.3. The color gradient indicates the ground truth angular velocity. The DVBF learns a two-dimensional manifold in the embedding three-dimensional space.
This figure has previously appeared in Karl et al. (2017a) and Karl (2020).

pendulum example, its angle is part of the state space. This translates either to a periodicity constraint on the dynamics or to an embedding of the angle as sine and cosine, a one-dimensional manifold in a two-dimensional embedding latent space.[5] The latter case is easier to implement for smooth density networks.

Indeed, DVBFs learn to extract such a barrel-shaped manifold for the pendulum data, fig. 4.5. Such embedded manifolds lead to states that are implausible.[6]

Gaussian transition density networks $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ or $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$ can stay close to such manifolds with their entire probability mass since the uncertainty between time steps tends to be low if Δt is small. The initial inference network, however, tends to have larger uncertainty, as discussed earlier. If implemented by a Gaussian density network, due to the sheer geometry of Gaussian distributions at least some of the probability mass may be far off the state-space manifold. To this end, DVBFs add an additional function g *after* sampling the density network $q(\mathbf{z}_1 \mid \mathbf{x}_{1:\tau})$, effectively performing a mapping onto the manifold. This mapping is shared between $p(\mathbf{z}_1)$ and $q(\mathbf{z}_1 \mid \mathbf{x}_{1:\tau})$, as is reflected in both figs. 4.4a and 4.4b.

Since g is typically not invertible, the prior KL term at $t = 1$ is computed before the mapping. The prior KL term is technically not invariant under g. This effect is ignored. A more principled, less

5 Technically, the embedding only needs to be homeomorphic—i. e., topologically equivalent—to the circular sine and cosine embedding. In the simplest case, that can mean a rescaling of sine and cosine.

6 The term *manifold* is used rather loosely here, not in the strict sense of Riemann manifolds. Indeed, it is an open question of research how to enforce the discovery of flexible proper manifolds in such latent spaces.

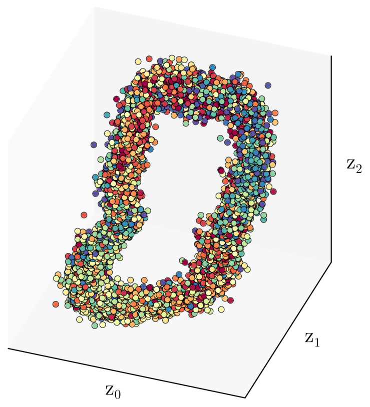


Figure 4.6: Test time samples of the approximate posterior $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$ of a *deep Kalman filter* (DKF) trained on the image pendulum data presented in fig. 4.3. Color indicates the ground truth angular velocity. The DKF only learns to encode the angle as a one-dimensional manifold but fails to capture angular velocity.
This figure has previously appeared in Karl et al. (2017a) and Karl (2020).

flexible approach would be to use normalizing flows or flexible priors, cf. sections 2.5.1 and 2.5.2.

4.3 CONCURRENT AND LATER MODELS

A number of publications have addressed learning SSMs or closely related models by auto-encoding and amortized inference. In this section, we review a number of key papers and relate them to DVBFs.

We also refer the interested reader to a concurrent review paper of this class of models due to Girin et al. (2020).

4.3.1 Deep Kalman Filters

Concurrently to DVBFs, Krishnan et al. (2015) suggested DKFs. Both models constitute the first attempts to learn neural SSMs by employing amortized inference.

The generative model of DKFs is a standard SSM, where transition and emission model are density networks. The inference model is reminiscent of the inference model of STORNs and as such more straightforward than with DVBFs: an RNN processes either[7] $\mathbf{x}_{1:t}$ or $\mathbf{x}_{1:T}$—depending on whether the RNN is bidirectional—and at each time step produces the approximate posterior $q(\mathbf{z}_t \mid \mathbf{x}_{1:T})$, that is

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t \mid \mathbf{x}_{1:T}). \tag{4.25}$$

7 The original paper suggests further baseline alternatives of lesser interest.

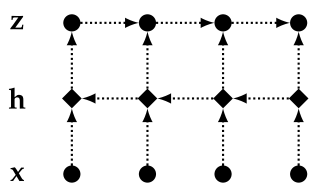


Figure 4.7: The inference model of *deep Kalman smoother* (DKS). The corresponding generative model, *deep Markov models* (DMMs), is omitted since it is a standard SSM known from fig. 3.1. Note that *deep Kalman smoothers* (DKSs) do not reuse the transition model for inference as indicated by the dotted lines.

Both generative and inference model have been depicted previously in figs. 3.1 and 4.1d.

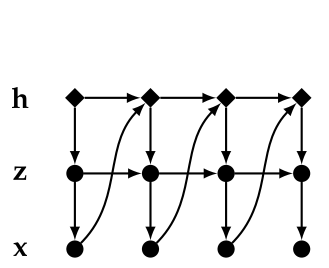
Interestingly, the authors notice that the true posterior is a smoother and provide the correct factorization according to eq. (3.25), yet their inference neglects the previous state in favor of the previous observations.

This turns out to be a crucial oversight. Empirically, DKFs consistently struggle to learn higher-order dynamic state-space components such as velocity that cannot be inferred from a single observation $\mathbf{x}_t$, cf. fig. 4.6. The transition is only learned through the prior KL in the ELBO. This does not provide sufficient inductive bias. The powerful recognition model can leverage intra-step correlations just as discussed earlier, which is sufficient to achieve high ELBO values.

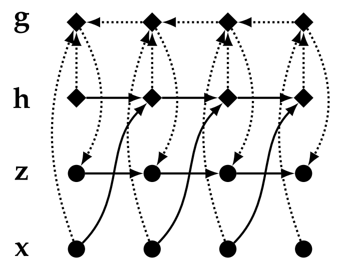
4.3.2 Deep Markov Models and Deep Kalman Smoothers

Krishnan et al. (2017) later also suggested *deep Markov models* (DMMs) as a generative model and *deep Kalman smoothers* (DKSs) as an inference model. DMMs are special cases of SSMs with a particular density network parametrization of the transitions $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)$. Rather than using an arbitrary neural network, they use a particular architecture inspired by *gated recurrent units* (GRUs; Chung et al., 2015)). The motivation is to provide an inductive bias towards linear and at least initially isometric transitions. From the theory of dynamical systems, it is known that isometric systems, i. e., systems with transition Jacobian eigenvalues close to the unit circle in the complex plane, are more *stable*, the exact same argument that motivates LSTMs and GRUs over vanilla RNNs.

The authors observe the true posterior factorization we know from eq. (3.25) and consider various inference models, eventually settling for the variant they call DKS. It is depicted in fig. 4.7. It consists of a unidirectional RNN that summarizes observations, running *backward* in time, as well as a specific combiner function that combines the previous state sample $\mathbf{z}_{t-1}$ and the summarized observations $\mathbf{h}_t(\mathbf{x}_{t:T})$ to obtain the distribution $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{t:T})$. Notably though, in contrast



(a) Graphical model extended with recurrent hidden units **h**.



(b) Inference model with additional hidden units **g**.

Figure 4.8: Graphical and inference model of *stochastic recurrent neural networks* (SRNNs). The connection between observations **x** and RNN hidden units **h** is as in the experiments of the original paper. In theory, the RNN could be fed other inputs, such as control signals. Solid lines in the inference model indicate that the generative model is reused.

to DVBFs the combiner function is completely independent of the prior transition function.

To the best of our knowledge, this is the first inference model in the suggested literature that correctly implements an approximation

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{t:T}) \tag{4.26}$$

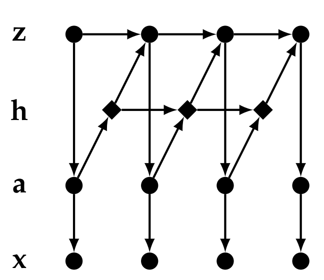
that is faithful to the true posterior factorization.

4.3.3 Stochastic Recurrent Neural Networks

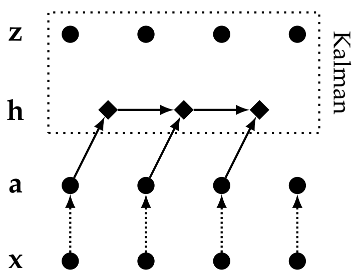
Fraccaro et al. (2016) suggest a hybrid model achieved by stacking probabilistic SSMs and deterministic RNNs in latent space called *stochastic recurrent neural networks* (SRNNs). The model is depicted in fig. 4.8.

In theory, the model is presented as an SSM where the state space consists of a probabilistic part **z** and a deterministic part **h**. In practice, however, the devil lies in the details: if there are no control signals—as we assume for the most part in this thesis—then the deterministic part of the dynamics will have no input and would always be the same. To overcome this, the model is modified to feed observations back into the latent system.[8] This violates SSM assumptions since the transition $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{1:t-1})$ is now no longer Markovian. SRNNs can thus rather be viewed as an alternative to the previously discussed non-SSM models such as STORN or VRNN.

8 This crucial detail is mentioned in the experimental section of Fraccaro et al. (2016).



(a) Graphical model extended with pseudo-observations **a**.



(b) Inference model with closed-form Kalman inference.

Figure 4.9: Graphical and inference model of *Kalman variational auto-encoders* (KVAEs). The model $p(\mathbf{z}_{1:T}, \mathbf{a}_{1:T})$ is restricted to be an LGS, its inference then closed-form as indicated by the box. Solid lines in the inference model indicate that the generative model is reused.

The deterministic part is the hidden state of an RNN. SRNNs orders the two parts: the probabilistic part depends on the deterministic part,

$$p(\mathbf{h}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{h}_t) p(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \mathbf{x}_{t-1}), \tag{4.27}$$

as indicated in fig. 4.8. The factor $p(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \mathbf{x}_{t-1})$ is a Dirac point mass distribution representing the deterministic propagation.

Inference then obeys the correct posterior factorization, eq. (3.25), and implements a smoothing posterior by a second inference RNN that runs backward in time, producing hidden states $\mathbf{g}_{1:T}$,

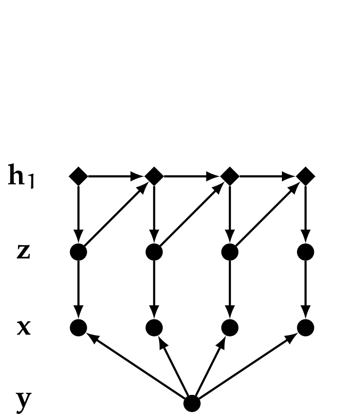
$$q(\mathbf{h}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \mathbf{x}_t) q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{g}_t(\mathbf{x}_t, \mathbf{h}_{t:T})). \tag{4.28}$$

4.3.4 Kalman Variational Auto-Encoders

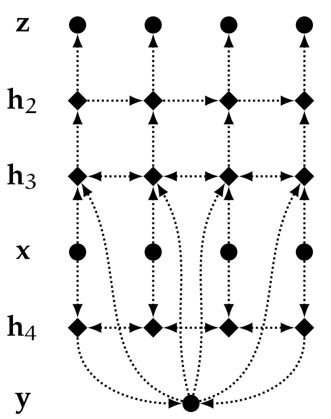
After SRNNs, Fraccaro et al. (2017) also suggested *Kalman variational auto-encoders* (KVAEs). This model is truer to its namesake than DKFs: the core idea is to leverage closed-form inference via Kalman filters (or smoothers). To this end, the authors assume an LGS in latent space. In order to represent nonlinear observations, they introduce the concept of pseudo-observations $\mathbf{a}_{1:T}$. Pseudo-observations are the virtual, intermediate, linear emissions of the LGS, which are then translated to real observations by means of a density network, i. e.,

$$p(\mathbf{x}_t \mid \mathbf{z}_t) = \int p(\mathbf{x}_t, \mathbf{a}_t \mid \mathbf{z}_t) \, d\mathbf{a}_t = \int p(\mathbf{x}_t \mid \mathbf{a}_t) p(\mathbf{a}_t \mid \mathbf{z}_t) \, d\mathbf{a}_t, \tag{4.29}$$

where $p(\mathbf{a}_t \mid \mathbf{z}_t)$ is a linear Gaussian distribution.



(a) Graphical model extended with recurrent hidden units **h**.



(b) Inference model with recurrent hidden units $\mathbf{h}_2$, $\mathbf{h}_3$, $\mathbf{h}_4$, distinct from another and $\mathbf{h}_1$.

Figure 4.10: Graphical and inference model of *disentangled sequential auto-encoders* (DSAEs).

In inference, the pseudo-observations are inferred like in a VAE, and then the states are inferred closed-form, i. e.,

$$q(\mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = p(\mathbf{z}_{1:T} \mid \mathbf{a}_{1:T}) \prod_{t=1}^{T} q(\mathbf{a}_t \mid \mathbf{x}_t), \tag{4.30}$$

where $p(\mathbf{z}_{1:T} \mid \mathbf{a}_{1:T})$ denotes the closed-form posterior as opposed to an approximation $q(\mathbf{z}_{1:T} \mid \mathbf{a}_{1:T})$.

Since LGSs are relatively restricted in scope, the latent dynamics of KVAEs may change with time. This is implemented by convex combinations of base matrices in a similar fashion to the locally linear transitions in DVBFs. The difference is that the weights are not a function of the previous state $\mathbf{z}_{t-1}$ but all previous pseudo-observations $\mathbf{a}_{1:t-1}$ by means of an RNN. During inference, the same function may be used—KVAEs originally use LSTMs—to infer the transition parameters from the amortized pseudo-observations. This model is depicted in fig. 4.9.

The authors argue that this difference to DVBFs is what allows them to use the closed-form inference. It should be noted, however, that this completely neglects the dependence of $\mathbf{a}_{1:t-1}$ on, among others, $\mathbf{z}_{t-1}$. Strictly speaking, $p(\mathbf{a}_{1:T}, \mathbf{z}_{1:T})$ is thus not a proper SSM, as is obvious from fig. 4.9. The inference as suggested by the authors does not account for that.

4.3.5 Disentangled Sequential Auto-Encoder

Li and Mandt (2018) suggest a model called *disentangled sequential auto-encoders* (DSAEs). Inspired by Hsu et al. (2017), DSAEs add a static variable $\mathbf{y}$ which is supposed to capture features that apply to the entire sequence, like the tone of voice for a speech signal. It is explicitly modeled to influence only the emissions but not the dynamics of the latent states,

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{y}) = p(\mathbf{y}) \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_t, \mathbf{y}) p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}). \tag{4.31}$$

Note how, by means of tying the states $\mathbf{z}_{1:T}$ via an RNN, this model is again not a strict SSM.

During inference, the static feature vector is first inferred from $\mathbf{x}_{1:T}$ through a bidirectional RNN. Then, the states $\mathbf{z}_{1:T}$ are inferred by first processing $\mathbf{y}$ and $\mathbf{x}_{1:T}$ with another bidirectional RNN and lastly processing the output of this RNN with a unidirectional RNN.

Both processes are depicted in fig. 4.10.

Interestingly, the model is designed to be smoothing in the sense that it uses future observations for inferring states. Like other models presented previously, it fails to feed back the eventual states into the inference. This is not reflected in the original publication.

4.3.6 Filtering Variational Objectives

A related approach was independently developed by Maddison et al. (2017), Le et al. (2018), and Naesseth et al. (2018). All three publications motivate their solution from slightly different angles but ultimately result in a similar algorithm. In this illustration, we will follow Maddison et al. (2017), who suggest a *filtering variational objective* (FIVO).

FIVO is an MCO, section 2.5.3, turning the unbiased estimator

$$p(\mathbf{x}_{1:T}) \approx \prod_{t=1}^{T} \sum_{p=1}^{P} \hat{w}_t^{(p)} \tag{4.32}$$

from the unnormalized weights $\hat{w}_t^{(p)}$ of particle filters, cf. eq. (3.66), into a lower bound

$$\ln p(\mathbf{x}_{1:T}) = \ln \mathbb{E}\left[\prod_{t=1}^{T} \sum_{p=1}^{P} \hat{w}_t^{(p)}\right] \geqslant \mathbb{E}\left[\ln \prod_{t=1}^{T} \sum_{p=1}^{P} \hat{w}_t^{(p)}\right]. \tag{4.33}$$

This lower bound is closely related to the bound used by IWAEs as discussed in section 2.5.3.

With FIVO, we can train SSMs. However, FIVO side-steps devising an explicit inference model by using particle filters. The inference model is reinterpreted as the proposal distribution for the particle filter.

We will revisit FIVO in section 7.3, where we will find that learning proposals via lower bounds can lead to biased solutions.

4.3.7 Hybrid Models

Sequential LVMs have sparked interest in the *reinforcement learning* (RL) literature. Their appeal is immediate: with high-dimensional sensors, such as cameras with highly redundant information, it is advisable to first reduce the sensor information to its core in order to devise more efficient policies. A comparably low-dimensional state space is a natural candidate. Karl et al. (2017b), later refined by Becker-Ehmck et al. (2020), showed how to utilize DVBFs for model-based RL.

Driven by application, we observe a surge of hybrid models in the field, i. e., models that are typically only partially motivated by VAEs and the models presented in this chapter. Often, they feature considerable deterministic components on top. For instance, Ha and Schmidhuber (2018a,b) suggest a model where a VAE is used to embed image streams frame by frame. The latent dynamics are then entirely learned through deterministic LSTMs without ties to VI.

Other approaches borrow ideas from the models mentioned earlier, such as using hybrids between RNNs and SSMs (Hafner et al., 2019), even significantly altering the ELBO to perceived needs (Gregor et al., 2019; Hafner et al., 2019), or partially setting the posterior to be the prior (A. X. Lee et al., 2020), a noisy variant of RNN-SSM hybrids.

4.4 CRITICAL DISCUSSION

The application of the VAE framework to sequential LVMs has sparked a plethora of models that have been used for applications as varied as anomaly detection (Soelch et al., 2016), music generation (Hennig et al., 2017), machine translation (Su et al., 2018), video generation (Denton and Fergus, 2018), or simultaneous localization and mapping (SLAM; Mirchev et al., 2019). As we have seen, they most recently found their way into model-based RL. In fact, chapter 5 tackles tracking as another application.

At the same time, we critically observe that in the roughly seven years since the publication of VAEs no single approach has crystallized as a reliable go-to method for sequential LVMs. VRNNs are often used as a reliable baseline, likely due to code availability, but are generally not used as an off-the-shelf solution.

For SSMs, no clear contender could be established as a standard solution. In fact, as our discussion of hybrid models for model-based RL shows, new models with minor adjustments and modifications are invented over and over. This constant reinvention of models and algorithms for very similar purposes and the rather liberal approach to, e. g., SSM assumptions motivates a critical reflection of possible causes. The following paragraphs identify recurring themes and gather possible hypotheses.

ASSUMPTIONS AND DETERMINISTIC COMPONENTS Many of the approaches presented in section 4.3 are motivated by SSMs and indeed often presented as neural SSM implementations. As we have seen in several cases, however, these assumptions are not implemented faithfully, often by adding feedback loops or entirely deterministic components of the state space that effectively render the latent transition stateful and thus non-Markovian.

Often, these changes are not apparent from the theoretical presentation and can only be found in experimental sections, appendices, or supplementary code. As a consequence, the repercussions of such choices are discussed and examined insufficiently.

Deterministic state components are a prime example. In a probabilistic, sequential LVM learned with the ELBO the prior is regularized via the prior KL term. That is, a complex prior model needs to be justified by equal improvement in the likelihood term of the ELBO. Deterministic dynamics, even partially deterministic, can side-step this balance: they do not appear in the prior KL term and are thus not regularized. Anything the probabilistic components capture can be represented by the deterministic components without the complexity penalty of the prior KL.

One needs to carefully evaluate to what extent ELBO values of these hybrid models can even be compared fairly. Their apparent closeness in absolute value between models is misleading here. Prior KL terms are typically orders of magnitude smaller than the likelihood term in absolute values. In other words, the likelihood term tends to dominate the ELBO.

POSTERIOR FACTORIZATION AND CONDITIONING Many of the density network designs for amortized sequential posteriors are motivated from the perspective of auto-encoding rather than Bayesian inference. The apparent design principle is the efficient exploitation of RNNs to implement $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$ as a mapping from observations $\mathbf{x}_{1:T}$ to states $\mathbf{z}_{1:T}$. With occasional exceptions—like deep Kalman smoothers, section 4.3.2—the density network designs end up not adhering to the *known* structure of the true Bayesian posterior, e. g., eq. (3.25) for SSMs. While the posterior design is necessarily an approximation, the effects of such violations were insufficiently studied.

The amortization of posteriors via density networks further adds an axis to the design space. A density network has an explicit input-output relationship between the random variable and its conditions. This also implies that designs can actively choose—and in particular choose *not*—to use certain inputs. DVBFs are a prime example: motivated by a filtering approach, the amortized posterior transition is a density network $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$. In comparison to the true posterior transition $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{t:T})$, it is actively modeled to ignore the future observations $\mathbf{x}_{t+1:T}$.

LOOSE COUPLING OF PRIOR AND POSTERIOR Another common thread in the literature is to approximate the posterior $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$ *as a whole*, i. e., the density networks that implement the approximate posterior share little or no weights with the density networks that implement the generative model $p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$.

This is not warranted by the literature on sequential Bayesian posteriors. The prediction-update cycle we know from section 3.3 can serve as an illustration: the Bayesian posterior can be written as a recursive function of prior components and subsequent normalization.

Completely ignoring such insights leads to the arguably most interesting component, the prior $p(\mathbf{z}_{1:T})$ capturing the latent dynamics, often being an afterthought in learning, which is dominated by fitting the approximate posterior. We have alluded to this point when motivating the design principles of DVBFs. Our results comparing DKFs and DVBFs are a strong indication that reusing prior components for the posterior can be beneficial for learning better priors.

DATA Much like none of the models has proven to be an off-the-shelf solution, the community has not settled for a shared set of benchmark data sets. This stands in contrast to VAEs, where standard data sets such as MNIST are widely accepted. This lack of standard hinders comparability.

Further, the lack of standardized data sets makes it difficult to evaluate common properties of these data sets. The closest to an emerging community standard are RL environments (Todorov et al., 2012; Brockman et al., 2016). These environments are typically simulations of deterministic environments, with little to no noise both in simulated dynamics and sensor readings. It is questionable to what extent a probabilistic treatment with, e. g., SSMs would be beneficial for such data. This goes to show that the choice of data can severely bias the results for any given model, and the choice of data is often not motivated transparently.

BELIEF REPRESENTATIONS Some assumptions that were made in the literature on VAEs have found their way into the sequential models without further scrutiny. A central example is the approximation of expected values with single-sample MC integration, an almost universal feature of the models presented here with FIVO as a notable exception.

In a setting like the VAE, where the posterior is by default rather simple and in particular unimodal, this approach is not harmful and more than justified by speed-up in the learning phase. In many dynamical systems, however, assuming approximate posteriors so simple that a single sample could represent them well may limit the class of systems which can be represented and learned well by such learning algorithms. Well-known phenomena like perceptual aliasing—

recall the localization example in fig. 3.3—become more challenging to resolve.

A potential remedy would be to use richer belief representations than single-sample approximations. The challenge here is to provide a belief representation that can be sampled and evaluated efficiently—so that the ELBO remains a viable objective—or propagated through time in some fashion that allows efficient learning.

The hypotheses of this section are of a more inductive nature, observing patterns in the existing literature and speculating about plausible causes. In part III, we will test multiple of these hypotheses and suggest possible remedies.

5 CASE STUDY: VARIATIONAL TRACKING

Chapter 4 has introduced a general-purpose learning algorithm for state-space models with deep variational Bayes filters. In this chapter, we will present an application of deep variational Bayes filters to the problem of object tracking in sequential, visual data. Many relevant and concrete perception tasks can be solved given sufficient engineering efforts (Pulford, 2005; Cadena et al., 2016). Adaptation of conceptually simple frameworks to specific scenarios requires the exploitation of constraints to achieve satisfying performance. In tracking, e. g., different target representations (point, bounding box), observations (depth, color), and partial models (appearance, motion) need to be incorporated.

In recent years, learning methods and in particular deep neural networks have enhanced or even replaced hand-crafted perception pipelines, promising competitive performance in the presence of rich data sets. These approaches can loosely be put into three categories. First, components of existing pipelines are replaced by neural components, leaving major parts untouched (Dosovitskiy et al., 2015; Schulter et al., 2017; Yang et al., 2018). Second, complete pipelines are replaced with learnable counterparts, often inspired by the previously dominant solutions (Krizhevsky et al., 2012; Kahou et al., 2017; Kosiorek et al., 2017; Gordon et al., 2018; Parisotto et al., 2018). Third, the data generating process is formulated as a latent variable model and the task of interest expressed as Bayesian inference. The last scenario immediately allows us to apply the techniques discussed in the previous chapters.

Multi-object tracking has been the primal concern of many works (Pulford, 2005). Bewley et al. (2016) propose using a detector and a subsequent state-space model, showing the promise of such methods outside a deep learning context. Neiswanger et al. (2014) formulate tracking as a mixture of Dirichlet processes operating on top of a feature extraction pipeline without the need for supervision signals. A series of works considers tracking via end-to-end supervised learning (Kahou et al., 2017; Kosiorek et al., 2017; Ning et al., 2017; Gordon et al., 2018), showing that it is possible to represent trackers with neural architectures when annotated data are available. Importantly, the learning algorithms presented in this chapter are unsupervised. Tracking does not emerge from explicit supervision via labels but implicitly by rephrasing it as Bayesian inference.

In video prediction the central concern is the prediction of future frames in a video stream (Srivastava et al., 2015; Babaeizadeh et al., 2018; Denton and Fergus, 2018; A. X. Lee et al., 2018; Steenkiste et al., 2018). This can be expressed as inference in the underlying generative model but without a focus on tracking. This is the starting point of our method, which is based on the approaches discussed in chapter 4.

Beyond highlighting an interesting application, this chapter serves two major purposes in the context of this thesis. The first concerns the type of LVMs used for this application. Up to this point, all discussed LVMs could be considered *black-box* in the sense that we had limited or no assumptions on the interpretation of the latent variables or states that were learned. In the pendulum example, post-hoc analysis showed that the latent variables indeed corresponded to the ground truth and formed a plausible manifold. Yet, the numerical values of a single state $\mathbf{z}_t$ are not immediately interpretable, and, e. g., axis alignment of learned states and ground truth quantities is coincidental. The only inductive bias in this example is to fix the number of latent dimensions to $d_{\mathbf{z}} = 3$. This chapter is a conscious departure. The latent states of the tracking model are much more structured, and we examine how this added structure in turn informs the design of both generative and inference model.

Secondly, many of the challenges for learning LVMs we identified in section 4.4 recur tangibly in this line of work. As such, this chapter bridges the gap between DVBFs on the one hand and subsequent developments discussed in part III.

5.1 SCENE UNDERSTANDING

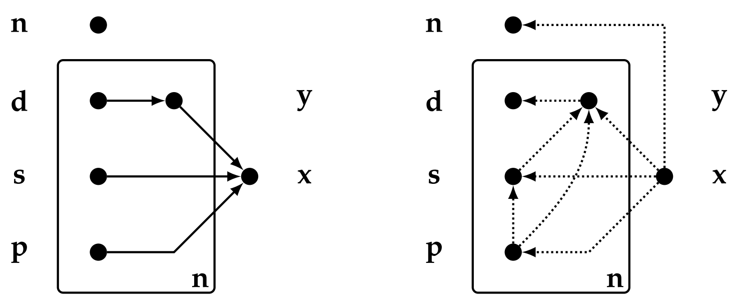
To begin, we discuss the work of Eslami et al. (2016), who suggest *Attend, Infer, Repeat* (AIR). AIR is a special-purpose variant of VAEs for scene understanding. Scene understanding here refers to the process of decomposing an image into a set of conceptually similar objects. The core idea of our model presented later is to add dynamic consistency to frame-wise scene understanding. In this formulation (approximate) Bayesian inference means tracking objects between frames—our target application.

AIR implements scene understanding as a VAE that imposes structure on the generative latent-variable model: it assumes scenes of $n \in \mathbb{N}_0$ conceptually similar objects $\mathbf{y}^{(i)}$ defined by a set of properties $\mathbf{z}^{(i)} = \left\{\mathbf{p}^{(i)}, \mathbf{s}^{(i)}, \mathbf{d}^{(i)}\right\}$, comprised of the position $\mathbf{p} \in \mathbb{R}^2$, size of the object $\mathbf{s} \in \mathbb{R}^2$, and a content description vector $\mathbf{d} \in \mathbb{R}^d$.

The rather baroque generative model is

$$p\left(\mathbf{x}, \mathbf{n}, \left\{\mathbf{y}^{(i)}, \mathbf{p}^{(i)}, \mathbf{s}^{(i)}, \mathbf{d}^{(i)}\right\}\right) \tag{5.1}$$

$$= p(\mathbf{n}) p\left(\mathbf{x} \,\middle|\, \left\{\mathbf{y}^{(i)}, \mathbf{p}^{(i)}, \mathbf{s}^{(i)}\right\}\right) \prod_{i=1}^{\mathbf{n}} p\left(\mathbf{y}^{(i)}, \mathbf{d}^{(i)}, \mathbf{p}^{(i)}, \mathbf{s}^{(i)}\right), \tag{5.2}$$



(a) Graphical model of AIR. **(b)** Inference model of AIR.

Figure 5.1: Graphical and inference model of *Attend, Infer, Repeat* (AIR). The box indicates $\mathbf{n}$-fold repetition of the graphical model, one set of position $\mathbf{p}$, size $\mathbf{s}$, and description $\mathbf{d}$ for each of the $\mathbf{n}$ distinct objects. The bold font type of the integer $\mathbf{n} \in \mathbb{N}_0$ hints at the vector-valued encoding in the actual implementation. In practice, position $\mathbf{p}$ and size $\mathbf{s}$ are estimated jointly, i. e., their depicted inference order here is arbitrary.

with i. i. d. object priors

$$p(\mathbf{y}, \mathbf{d}, \mathbf{p}, \mathbf{s}) = p(\mathbf{y} \mid \mathbf{d})p(\mathbf{d})p(\mathbf{p})p(\mathbf{s}), \tag{5.3}$$

and the inference model is

$$q\left(\mathbf{n}, \left\{\mathbf{y}^{(i)}, \mathbf{p}^{(i)}, \mathbf{s}^{(i)}, \mathbf{d}^{(i)}\right\} \,\middle|\, \mathbf{x}\right) \tag{5.4}$$

$$= q(\mathbf{n} \mid \mathbf{x}) \prod_{i=1}^{\mathbf{n}} q\left(\mathbf{y}^{(i)}, \mathbf{d}^{(i)}, \mathbf{p}^{(i)}, \mathbf{s}^{(i)} \,\middle|\, \mathbf{x}\right) \tag{5.5}$$

with per-object posterior density network[1]

$$q(\mathbf{y}, \mathbf{d}, \mathbf{p}, \mathbf{s} \mid \mathbf{x}) = q(\mathbf{p}, \mathbf{s} \mid \mathbf{x})q(\mathbf{y} \mid \mathbf{p}, \mathbf{s}, \mathbf{x})q(\mathbf{d} \mid \mathbf{y}). \tag{5.6}$$

For better understanding, the graphical and inference model are depicted in fig. 5.1.

The name *Attend, Infer, Repeat* is derived from the procedure by which its inference operates. After determining the number of objects, $q(\mathbf{n} \mid \mathbf{x})$, the objects are inferred one at a time by attending to their position and size, $q(\mathbf{p}, \mathbf{s} \mid \mathbf{x})$, inferring the contents, $q(\mathbf{y} \mid \mathbf{p}, \mathbf{s}, \mathbf{x})q(\mathbf{d} \mid \mathbf{y})$, and repeating the process for all objects. Crucially, this sequential inference process imposes an order—albeit arbitrary—on the objects.

This setup is in several ways a departure from vanilla VAEs. Position and size vectors $\mathbf{p}$ and $\mathbf{s}$ are by design immediately interpretable in relation to the original scene $\mathbf{x}$. In fact, the generative model uses them without further processing as the inputs to an invertible spatial transformer network (Jaderberg et al., 2015), which is used as a density network to implement

$$p\left(\mathbf{x} \,\middle|\, \left\{\mathbf{y}^{(i)}, \mathbf{p}^{(i)}, \mathbf{s}^{(i)}\right\}\right), \tag{5.7}$$

1 The per-object posteriors are technically linked via a deterministic RNN hidden state without feedback. We refrain from a detailed discussion in the interest of brevity and refer the interested reader to Eslami et al. (2016) and Akhundov (2018).

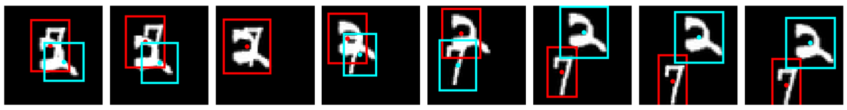


Figure 5.2: Application of AIR to individual frames of a sequence. The color frames indicate inference of the object positions and sizes. Their color indicates the order of inference.
A similar figure has previously appeared in Akhundov et al. (2019).

the process that pastes the objects into the scene. The inverse is then the density network that implements the corresponding inference

$$q\left(\mathbf{y}^{(i)} \,\middle|\, \mathbf{x}, \mathbf{p}^{(i)}, \mathbf{s}^{(i)}\right) \tag{5.8}$$

which extracts an object into a canonical representation $\mathbf{y}^{(i)}$.

Only the description vectors $\mathbf{d}$ resemble the less interpretable latent variables of VAEs. Indeed, the pair of generative density network $p(\mathbf{y} \mid \mathbf{d})$ and its corresponding inference density network $q(\mathbf{d} \mid \mathbf{y})$ along with the prior $p(\mathbf{d})$ arguably constitute a VAE within the latent states.

An interesting observation is that the more complex graphical model of AIR compared to VAEs induces conditional independence patterns between the latent variables, like

$$q(\mathbf{d} \mid \mathbf{y}, \mathbf{p}, \mathbf{s}, \mathbf{x}) = q(\mathbf{d} \mid \mathbf{y}). \tag{5.9}$$

This also informs their order of inference—an aspect to inference that is usually absent in vanilla VAEs.

We refrain from further implementation details of AIR and refer the reader to Akhundov (2018) and Akhundov et al. (2019) for a detailed review, including two adjustments to the original model by Eslami et al. (2016), which empirically stabilize the notoriously difficult training of AIR (Kosiorek et al., 2018).

5.1.1 Dynamic Scene Composition

A straightforward way to leverage AIR for the tracking problem would be to use AIR for independent, frame-wise scene understanding. Several problems of this approach become immediately apparent, as fig. 5.2 shows.

The order of attention in AIR is arbitrary. Empirically, it learns a spatial policy for attention order, e. g., left-to-right, top-to-bottom (Eslami et al., 2016). With moving objects, this inevitably leads to permutations in object discovery order between frames.

In a single frame, AIR cannot distinguish between multiple overlapping objects and non-overlapping regular objects since it is not equipped with a semantic understanding of the difference between the two or any other prior information as to the appearance of the objects it is supposed to detect.

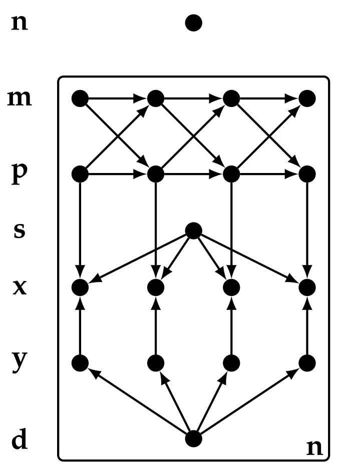


Figure 5.3: The graphical model of *variational tracking and state-space inference* (VTSSI). Note that the number of objects $\mathbf{n}$ is fixed for a sequence $\mathbf{x}_{1:T}$. Similarly, description $\mathbf{d}$ and size $\mathbf{s}$ remain constant across time and only vary between objects.

A third downside of using AIR independently on frames is that, after tracking, prediction is not possible by design. The core idea to tackle all three challenges is to integrate the structured generative model of AIR with SSM dynamics. This connects frames in the generative model. Subsequently, we adjust the inference model to explicitly take temporal consistency into account.

We add an explicit motion random variable $\mathbf{m}_t^{(i)}$ to the latent space of frame $\mathbf{x}_t$. It captures higher-order motion description, e. g., velocities, accelerations, or curve radii. This allows us to define Markov transition priors

$$p\left(\mathbf{p}_t^{(i)}, \mathbf{m}_t^{(i)} \,\middle|\, \mathbf{p}_{t-1}^{(i)}, \mathbf{m}_{t-1}^{(i)}\right) \tag{5.10}$$

for state prediction in the next frame given the current state. Apart from this addition, the likelihood model of AIR is reused at every time step,

$$p\left(\mathbf{x}_t \,\middle|\, \left\{\mathbf{y}^{(i)}, \mathbf{p}_t^{(i)}, \mathbf{s}^{(i)}\right\}\right), \tag{5.11}$$

as can be seen from fig. 5.3. This implies we assume a fixed number of objects for a sequence of frames $\mathbf{x}_{1:T}$, and descriptions and sizes of these objects are constant.

5.2 TRACKING AS INFERENCE

Naive frame-wise application of AIR is insufficient for tracking. Our suggested inference model addresses these points and ties in the new dynamic state components with three distinct inference components, which we will explain in the following section. We call the resulting sequential auto-encoding model *variational tracking and state-space inference* (VTSSI).

5.2.1 Inferring Consistent Labels

The first component aims at preventing label switches. One solution would be to aim at a matching of objects between frames. This would still require AIR inference evaluations for every frame. On top, the matching adds additional complexity, especially considering that the process should be differentiable to allow gradient-based learning.

Instead, we leverage the description $\mathbf{d}$. At this point, we assume we have achieved a description from any source. This may be a single evaluation of AIR on the first frame; alternatively a consensus from several frames, as will be explained further down.

Based on an object description $\mathbf{d}$ we try to find the corresponding object in subsequent frames. In comparison to AIR, this reverses the inference order of object *position* $\mathbf{p}$ and *description* $\mathbf{d}$. This prevents label switches while reducing the number of applications of the computationally relatively more expensive AIR component from T to one. The implementation is inspired by the fast-weights approach (Schmidhuber, 1992; Ba et al., 2016): we compute convolution kernels from $\mathbf{d}$. From the resulting features of frame $\mathbf{x}_t$ and the previous position $\mathbf{p}_{t-1}$ the updated position $\mathbf{p}_t$ is inferred. Since its task is to *find* a previously seen object, we call this component FIND.

FIND may be interpreted as a density network

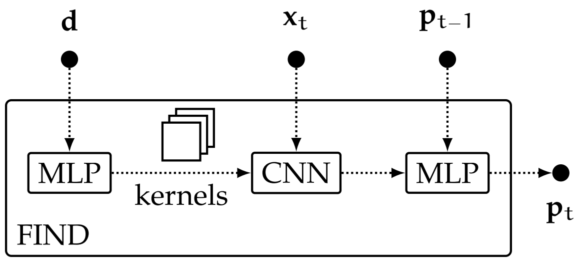
$$q(\mathbf{p}_t \mid \mathbf{p}_{t-1}, \mathbf{d}, \mathbf{x}_t). \tag{5.12}$$

The component is depicted in fig. 5.4a.

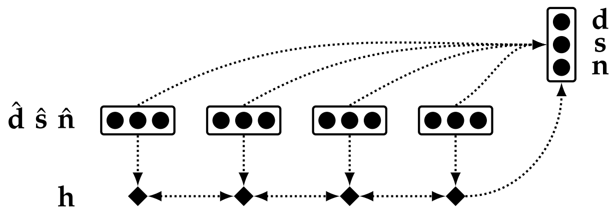
5.2.2 Inferring Overlapping Objects

FIND, combined with AIR on the first frame, will find an object in every frame and can thus already serve as a tracking algorithm. However, this assumes that AIR is able to extract an object description $\mathbf{d}$ from the first frame $\mathbf{x}_1$. Since we do not impose AIR with prior information as to the appearance of the objects it is supposed to detect, it will not learn to single out overlapping objects and find a single description for the resulting overlap. As a consequence, for the combination of AIR and FIND to work, we would need to impose the rather strong assumption that objects must not overlap in the first frame.

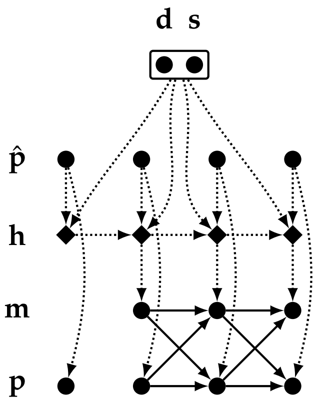
We introduce the second component RECT (for rectification) to relax this assumption: rather than relying on AIR's object description from the first frame, an RNN processes the inference output of AIR on the first K frames, where K is a hyper-parameter. This net reaches a consensus $\mathbf{z}$ from the K sets $\hat{\mathbf{z}}_{1:K}$ of latent variables from applications



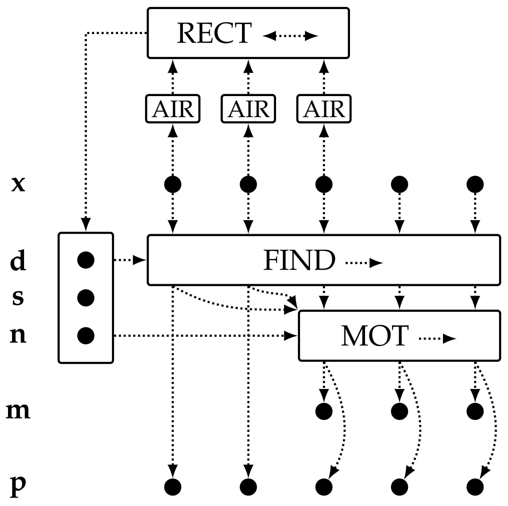
(a) The FIND density network $q(\mathbf{p}_t \mid \mathbf{p}_{t-1}, \mathbf{d}, \mathbf{x}_t)$. It uses the description $\mathbf{d}$ to produce fast-weight convolution kernels for a CNN, which *rediscovers* the object described by $\mathbf{d}$ in frame $\mathbf{x}_t$ and subsequently updates the position estimate $\mathbf{p}_t$.



(b) The RECT inference model that computes a single consensus between several sets of descriptions $\hat{\mathbf{d}}$, sizes $\hat{\mathbf{s}}$, and $\hat{\mathbf{n}}$ variables by means of a bidirectional RNN.



(c) The MOT inference model to infer motion variables after a burn-in phase (one frame in the depiction). The prior transition is reused for $\mathbf{p}$ and $\mathbf{m}$.



(d) FIND, RECT, and MOT—abstracted to their interfaces—integrated into the full VTSSI inference model. RECT computes a consensus on three frames. MOT has a two-frame burn-in. FIND and MOT are unidirectional, RECT is bidirectional.

Figure 5.4: The components FIND, RECT, and MOT individually, and combined into *variational tracking and state-space inference* (VTSSI).

of AIR on the first K frames, e. g., by means of weighted averaging. Finally, we use the more robust consensus $\mathbf{d} \in \mathbf{z}$ as the input to the FIND module.

The RECT inference component is depicted in fig. 5.4b.

5.2.3 Inferring Motion

FIND and RECT are designed to deal with label switches and object overlap. As described, the focus of these two components is the consistency of objects across time.

The last component of VTSSI deals with the added state-space dynamics of the generative model. This inference component, dubbed MOT, aims at establishing dynamic consistency of single objects, inferring the motion variables $\mathbf{m}_t$ and thereby refining the position estimates $\mathbf{p}_t$.

To infer the motion variable, we feed the object position proposals $\hat{\mathbf{p}}_{1:T}$ from FIND to an RNN. After M frames, where M is at least the order of dynamics assumed, the RNN provides inferred motion proposals $\hat{\mathbf{m}}_{M+1:T}$. Both position and motion proposals are fused with prior predictions $\tilde{\mathbf{p}}_t^{(i)}$ and $\tilde{\mathbf{m}}_t^{(i)}$ from the transition prior. The fusion is achieved by averaging.

This inference process is inspired by DVBFs. In particular, reusing the prior transition is an integral part of the design. As we will show empirically—cf. section 5.3—this allows faithful multi-step object-level prediction. Similarly, the explicit burn-in phase of M frames is the initial inference model of DVBFs in new clothes. The application, tracking, is inherently a filtering application. Where DVBFs opt for kickstarting the inference by a smoothing inference of the initial state, VTSSI makes this deficiency explicit and only infers motion as soon as it is reasonably possible from past observations.

The MOT inference component is depicted in fig. 5.4c.

5.2.4 Combined Inference

Combining all suggested modules, we arrive at the full architecture, which we call *variational tracking and state-space inference* (VTSSI). It processes initial frames $\mathbf{x}_{1:K}$ separately with AIR; reaches a consensus with RECT; uses this consensus in FIND to determine positions; refines the position estimates with dynamic information by exploiting MOT. This procedure is depicted in fig. 5.4d.[2]

An interesting feature of VTSSI is its modularity: rather than using the full model with all suggested components, we can choose to use

2 To acknowledge Adnan Akhundov's contribution to this collaboration and in the interest of space, we refer the reader to Akhundov (2018) and the appendices of Akhundov et al. (2019) for full implementation details.

only some of them, depending on the downstream task, for a more efficient model. We will investigate this in the following section.

VTSSI, arguably more than any model discussed so far, tightly combines concrete neural architectures and density networks. The components have defined interfaces and can be combined in any way that returns estimates of the desired quantities. Only the ultimate position estimate is interpreted as an approximate posterior with a corresponding prior. Intermediate estimates, for instance, from AIR or FIND components before being processed by MOT, are not considered in a Bayesian way in the sense that they do not have priors.

5.2.5 DDPAE and SQAIR

Two related approaches to ours have been suggested in the literature: *decompositional disentangled predictive auto-encoders* (DDPAEs; Hsieh et al., 2018) and *sequential Attend, Infer, Repeat* (SQAIR; Kosiorek et al., 2018). Both approaches use attention-based amortized inference to decompose video sequences of moving objects into per-object latent state sequences. Like VTSSI, both approaches borrow the likelihood model of AIR, cf. section 5.1.

DDPAE focuses on faithful prediction of the tail $\mathbf{x}_{K+1:T}$ of a sequence from its head $\mathbf{x}_{1:K}$. As a consequence, it is trained on a lower bound to the conditional $p(\mathbf{x}_{K+1:T} \mid \mathbf{x}_{1:K})$ rather than the joint $p(\mathbf{x}_{1:T})$. This also leads to architectural differences: in contrast to VTSSI and SQAIR, DDPAE does not auto-encode the entire sequence but follows a seq2seq-inspired approach (Sutskever et al., 2014). The sequence head $\mathbf{x}_{1:K}$ is only used for inference and never reconstructed. Conversely, the latent states $\mathbf{z}_{K+1:T}$ of the sequence tail $\mathbf{x}_{K+1:T}$ are never inferred from data but predicted from the head. Both inference and prediction are implemented by RNNs. DDPAE further models interactions between objects by means of another recurrence that connects inference of individual objects.

SQAIR introduces two inference components: PROP and DISC. PROP handles object propagation between frames. Two recurrent cells update the position, then (based on the new position) update description and presence. DISC discovers new objects. It works much akin to inference in AIR, except that the inference of a new object is informed by the latent states of existing objects from propagation to avoid duplicate discovery. Relying on AIR to this extent, SQAIR inherits its inability to handle overlapping objects in inference for the first time step and assumes non-overlapping first frame. SQAIR can, in principle, support entering and exiting objects at arbitrary frames.

Contrasting DDPAE and SQAIR with VTSSI, we conclude that all models share the same ancestor AIR, specifically the non-dynamic part of latent space design and resulting likelihood model. A major distinctive feature of VTSSI is enhancing the state space with an explicit motion variable **m**, capturing the dynamics of motion. This extra variable turns the position transition fully Markov and the overall model into a proper state-space model. In contrast, both DDPAE and SQAIR use recurrent cell states in the transition model, which need to capture the motion information. This reduces the interpretability of the latent state as the role of the recurrent state is unclear for each specific model and rules out regularization via priors.

All three models implement significantly different inference procedures for the sequential case. Our modular framework focuses on robust inference even in challenging scenarios to allow for accurate long-term prediction even for complicated nonlinear motion. Object interaction (as in DDPAE) or entering and exiting objects (as in SQAIR) are not considered but could be introduced by adding new or modified components to the VTSSI framework.

5.3 EXPERIMENTS

We conducted two sets of experiments. The first aims at understanding the modularity of VTSSI and the added benefit of each component. The second then examines differences between VTSSI and the related baselines DDPAE and SQAIR.

5.3.1 Evaluating Components of VTSSI

With the first set of experiments, we study five model variants:

1. AIR,
2. FIND (based on AIR),
3. RECT/FIND (i. e., VTSSI without MOT),
4. FIND/MOT (i. e., VTSSI without RECT), and
5. full VTSSI.

We trained these variants on four flavors of Moving MNIST—we used several variants with different features to perform targeted studies of the components of VTSSI: the data show either linear or elliptic motion, and either the first frame is guaranteed to contain only non-overlapping digits or not. Details on the data set can be found in appendix B.1.

We evaluated object counting accuracy as a proxy for robustness towards overlapping digits. Further, we report the accuracy of the position inference against ground truth as well as prediction accuracy for the two models that make use of MOT (all other models cannot generate coherent sequences by design). The results can be found in table 5.1. We can make several interesting observations:

Table 5.1: Quantitative tracking and prediction results with variants of *variational tracking and state-space inference* (VTSSI) on 10 000 test set trajectories. Counting accuracy refers to the average percentage of frames for which the amount of present objects is determined correctly. Inference and prediction errors refer to the average per-frame Euclidean distance (unit: pixels) from the inferred or predicted object center to the ground truth, respectively.
A similar table has previously appeared in Akhundov et al. (2019).

		AIR		FIND		RECT/FIND		FIND/MOT			VTSSI		
motion	overlap	count acc.	inf. err.	count acc.	inf. err.	count acc.	inf. err.	count acc.	inf. err.	pred. err.	count acc.	inf. err.	pred. err.
linear	✗	97.6 %	5.95	99.9 %	1.02	99.9 %	1.13	99.9 %	1.29	3.49	99.9 %	1.04	3.44
	✓	97.2 %	5.62	91.5 %	2.74	99.7 %	1.23	92.7 %	3.00	5.20	99.5 %	1.11	3.54
elliptic	✗	97.3 %	5.16	99.9 %	0.97	99.9 %	1.10	99.9 %	0.85	2.84	99.9 %	1.03	2.58
	✓	96.7 %	4.83	91.0 %	2.13	99.5 %	1.19	89.6 %	2.28	4.37	99.5 %	1.08	2.68

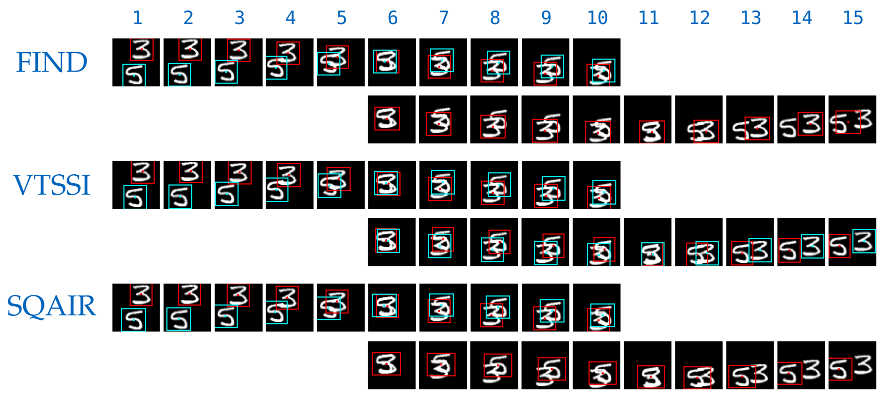


Figure 5.5: Qualitative example of challenging overlap. We pick one sequence $\mathbf{x}_{1:15}$ and show inference results of FIND, VTSSI, and SQAIR on two subsequences of length ten, $\mathbf{x}_{1:10}$ and $\mathbf{x}_{6:15}$. Applied to $\mathbf{x}_{1:10}$, FIND, VTSSI, and SQAIR successfully infer object properties. Applied to $\mathbf{x}_{6:15}$, FIND and SQAIR, relying on AIR for discovery, only recognize one object, and are unable to correct. VTSSI recognizes both digits, despite overlap of the two digits in all $K = 5$ first frames.
A similar figure has previously appeared in Akhundov et al. (2019).

FIND drastically improves the inference accuracy when the first frame is sufficiently clean to identify objects. In fact, FIND is on a par with VTSSI in these scenarios despite being much more lightweight. AIR suffers from label switches and recounting every frame. The results for FIND drop significantly when the assumption of non-overlapping objects in the first frame is removed. This can be mitigated by the introduction of RECT. We hypothesize that the slight drop in performance compared to FIND on non-overlapping first frames hints at room for improvement with the consensus mechanism of RECT.

RECT is very robust w. r. t. overlapping objects, as fig. 5.5 highlights. FIND, SQAIR (Kosiorek et al., 2018), and the RECT-based VTSSI successfully tackle the sequence on the left side with a clean first frame. When these models do not get access to the first five frames but start with the cluttered frames 6 and higher, FIND and SQAIR are

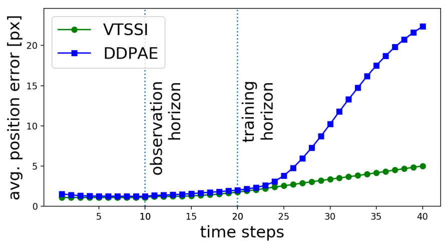


Figure 5.6: Test set prediction errors of DDPAE vs. VTSSI on data used in the original publication. Details in section 5.3.2.
This figure has previously appeared in Akhundov et al. (2019).

unable to recover from the wrong count in the first frame. This is a consequence of AIR's inability to deal with overlapping frames. We note that VTSSI succeeds despite RECT only accessing $K = 5$ frames, i. e., frames 6 to 10 in this case, all of which have overlapping objects.

MOT by itself generally does not lead to improved *inference* over FIND. When combined with RECT to form VTSSI, however, generative accuracy increases, even for scenarios where RECT is not strictly necessary—being able to predict helps inference. The full VTSSI handles all variants equally well. It performs well on linear and nonlinear motion, with slight advantage on the nonlinear, but smooth elliptic movements compared to discontinuous bouncing behavior, which is more difficult to predict.

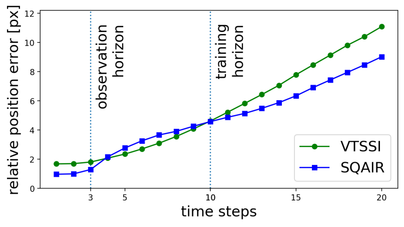
We conclude that each component fulfills its designated purpose: in the absence of FIND, we observe label switching; in the absence of RECT, overlapping objects cannot be disentangled reliably; in the absence of MOT, prediction is impossible, but even inference performance drops slightly.

Our evaluation also suggests that we can take advantage of the modular composition of VTSSI. For inference, FIND and RECT are the decisive factors. If prediction is not necessary, we can reliably train and use a simpler model.

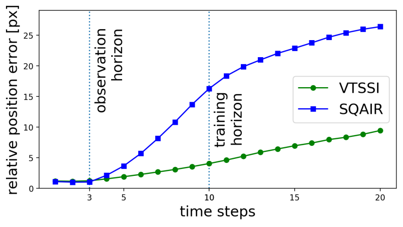
5.3.2 Comparison to DDPAE and SQAIR

On top of our ablation studies in section 5.3.1, we study VTSSI against the baselines DDPAE and SQAIR. The experiments investigate the robustness in inference and prediction, particularly over longer horizons. We build upon the Moving MNIST data sets previously studied with the baselines.

PREDICTION Hsieh et al. (2018) provide a data generation process for DDPAE. We built a training and test set from this process to ensure fair comparability, cf. appendix B.1. We trained both models with $T = 20$ and $K = M = 10$, as in the original publication. Starting from inferences with $K = 10$, we tested the position prediction error for



(a) VTSSI vs. SQAIR: SQAIR data.



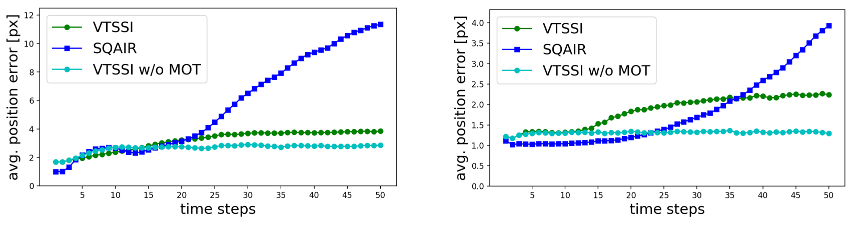
(b) VTSSI vs. SQAIR: our linear data.

Figure 5.7: Test set prediction of SQAIR vs. VTSSI on its data set and our data set. The models perform inference on three observations (first vertical line), the observation horizon. After that, object trajectories are sampled generatively without access to further observations and beyond training sequence length (second vertical line).
This figure has previously appeared in Akhundov et al. (2019).

$T > 20$, probing the generalization of the learned predictions. The average performance across a test set of 10 000 sequences can be seen in fig. 5.6. DDPAE and VTSSI are equally faithful to ground truth within the training horizon. However, the recurrent prediction cell of DDPAE is unable to generalize beyond the training horizon, it seems to severely overfit on the training horizon. This is particularly remarkable given DDPAE's loss is tailored towards prediction.

As with DDPAE, we tried to compare VTSSI to SQAIR on its original data set. Kosiorek et al. (2018) also provide a data generation process. We used the same data generation process, except we removed the noise, which turned prediction comparisons in the confined frames futile. We trained both models with $T = 10$ and $K = M = 3$, as in the original publication. On these data, we find SQAIR and VTSSI to perform equally well, with slight advantage for VTSSI within the training horizon and for SQAIR outside the training horizon, cf. fig. 5.7a. We noticed a subtle but crucial difference in the generation process of these data against the data we used for the results in, e. g., table 5.1: the data generation implements bouncing of the walls in terms of the top left corner of the tight bounding box, i. e., an object bounces in-frame on the top and left border and out-of-frame otherwise.

To examine the effect, we trained SQAIR on the linear data set suggested in section 5.3.1 with clean first frames, i. e., not SQAIR's original data set but well within SQAIR's assumptions. These data do *not* generate bouncing behavior in terms of bounding boxes but the actual object appearance. The result can be seen in fig. 5.7b. When required to model bouncing behavior, SQAIR falls short of VTSSI. SQAIR defines object positions in terms of bounding box corners, not the center (as DDPAE and VTSSI do). We believe that this generally makes it harder to learn accurate object dynamics except when the data set reflects this model assumption. This may lead to instabilities in the recurrent motion propagation cell of SQAIR. Using the object



(a) VTSSI vs. SQAIR: SQAIR data. **(b)** VTSSI vs. SQAIR: our linear data.

Figure 5.8: Test set inference error of SQAIR vs. VTSSI on a noise-free version of its own data set and our data set. Details in section 5.3.2.
This figure has previously appeared in Akhundov et al. (2019).

center makes it easier to use a simpler Markov transition. Specific motion behavior of an object can be saved into the motion variable **m**.

INFERENCE AND TRACKING With the same models and data as in our evaluation of prediction, we also examined tracking performance of SQAIR and VTSSI. The results can be seen in fig. 5.8. On both data sets, we see that the tracking performance of SQAIR drops drastically after around 20 steps. In contrast, VTSSI keeps a constant error over long horizons.

We also added one of the models discussed in section 5.3.1, VTSSI without the MOT component, which is not necessary for pure tracking. Rather than training a new, reduced model separately, this model is achieved by using the full VTSSI model. At test time, the outputs of its FIND component are directly evaluated. Unaffected by prediction errors, this model achieves even more reliable tracking performance.

5.4 DISCUSSION

In the context of this thesis, the results of section 5.3 are interesting from several points of view.[3]

Firstly, there are aspects inherently interesting to VTSSI. In the relatively young field of amortized inference, structured latent spaces like that of AIR and VTSSI are still the exception. The experiments are a proof of concept that this approach can be applied to a specific task, in this case tracking, because it can be cast as an instance of Bayesian inference. Moreover, VTSSI highlights how such highly structured latent spaces require a more careful design of the respective (approximate) inference model. Case in point: FIND is a feed-forward component, where the baselines use black-box RNNs. We believe this leads to the

3 We took the editorial liberty to exclude some results and discussion compared to Akhundov et al. (2019) in order to keep this chapter concise and connected to the surrounding chapters while reflecting the collaborative nature of the underlying publication. We refer the interested reader to the publication for more detail.

higher robustness, and as a side effect VTSSI trains significantly faster than SQAIR. Using reference implementations of the original authors, our model required at least an order of magnitude less wall clock time until convergence. The amount of parameters was roughly equal and most were used by the AIR base model.

Our ablation study of the different variants of VTSSI also shows the benefit of embracing a less black-box, more algorithmic inference design in terms of interfaces. Depending on the downstream application, the inference model may be chosen or adapted.

From a higher-level point of view, the results highlight several aspects of our critical discussion of the state of the art in sequential LVMs in section 4.4—for better or worse.

For instance, the baselines DDPAE and SQAIR are both examples of hybrid models where the dynamics are handled by deterministic RNNs and thus avoid Bayesian regularization against a prior. VTSSI explicitly avoids this and uses a proper SSM as a prior, and moreover it couples prior and posterior by reusing the SSM during inference. The result is a significantly increased robustness in both prediction and inference. Most strikingly, both baselines struggle when the presented sequences are longer than those used during training even when the underlying data generation process is otherwise identical. This underlines our previous discussion. Such choices haven been studied insufficiently. Our experiments indeed hint at insufficiently learned dynamics priors with the baselines, which are the basis of the robust prediction of VTSSI.

At the same time, VTSSI itself strikes different compromises that can be viewed critically in hindsight, especially in the light of the discussion in section 4.4. Most notable is its steadfast focus on *filtering*. This is inherited from DVBFs, since tracking—when interpreted as Bayesian inference—is a filtering task. Where DVBFs evaded the dilemma of the true posterior being of smoothing type by introducing the smoothing initial inference, VTSSI doubles down on filtering. The result is a compromise where the motion variables are only inferred after a burn-in phase, which means that no prior KL terms for the motion variables before that point—which are present in the generative model—will be used. None of these solutions is inherently preferable since both are approximations born out of necessity. This necessity warrants a closer look though, which will be the starting point for part III.

Part III

LEARNING BY SMOOTHING

Chapter 6 is based on ideas that have appeared previously in the following publication:

Bayer, Justin, Maximilian Soelch, Atanas Mirchev, Baris Kayalibay, and Patrick van der Smagt (2021). "Mind the Gap When Conditioning Amortised Inference in Sequential Latent-Variable Models." In: *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net. URL: https://openreview.net/forum?id=a2gqxKDvYys.

This publication is a collaboration between Justin Bayer and the author. First authorship is shared with equal contribution.

Chapter 7 is previously unpublished work.

Direct quotes from the publication are highlighted in gray font color. Minor adaptations of the quotes to the style of this thesis are not explicitly highlighted.

Supplementary material is collected in appendix C.

6 THE CONDITIONING GAP

Part II explored how to apply the VAE framework to sequential LVMs and specifically SSMs. In section 4.4, we discussed common model design assumptions and inaccuracies in our as well as related work.

In this chapter, we take a closer look at one of these highlighted design decisions for inference models shared by both DVBFs and VTSSI: a filtering inference density network, i. e., a density network that is restricted to past and present observations.

In both cases we required amendments to the respective inference models to balance out this assumption. DVBFs have a specialized initial inference network that introduces future observations almost through the backdoor. VTSSI acknowledges the assumption more directly with an explicit burn-in phase for inference of state components that correspond to higher-order dynamics.

These amendments are symptoms of a deeper issue, which we will study in this chapter. The introduction of a direct functional relationship between conditions and the distribution of the (approximate) posterior via density networks allows to be *selective* in terms of the conditions that are fed to the inference density network. This would not be the case when optimizing posterior distribution parameters directly.

This phenomenon is not restricted to sequential LVMs by any means. Examples for non-sequential LVMs are just much more rare in the literature (Kurle et al., 2019). VAE implementations can be selective about inference density network inputs; though counter-intuitive, they could, for instance, choose to ignore half the pixels of an input image $\mathbf{x}$. Sequential LVMs merely have a more established history of and use for different flavors of Bayesian inference like filtering and smoothing. The more varied density network design is an almost logical consequence. In contrast to the example of missing pixels, a filtering inference model may be conceptually simpler, and part II has shown several applications.

We call this phenomenon *partial conditioning* and the resulting approximate posteriors *partially conditioned*. Table 6.1 collects examples of partial conditioning in the literature discussed in chapter 4. As we will show in this chapter, it has effects on the posterior approximation distinct from both approximation and amortization gap known in the literature and discussed in section 2.4. While partial conditioning has been acknowledged in the literature (Krishnan et al., 2017), its effect has hitherto not been studied systematically. This chapter will study the emerging *conditioning gap* theoretically and empirically.

Table 6.1: Overview of partial conditions for sequential inference networks $q(\mathbf{z}_t \mid \mathcal{C}_t)$ in the literature. $\overline{\mathcal{C}}_t$ denotes missing conditions vs. the true posterior according to the respective graphical model. DKF acknowledges the true factorization but does not use $\mathbf{z}_{t-1}$ in any experiments.
This table has previously appeared in Bayer et al. (2021).

Model	$\mathcal{C}_t$	$\overline{\mathcal{C}}_t$
STORN (Bayer and Osendorfer, 2014)	$\mathbf{x}_{1:T}$	$\mathbf{z}_{1:t-1}$
VRNN (Chung et al., 2015)	$\mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}$	$\mathbf{x}_{t+1:T}$
DKF (Krishnan et al., 2015)	$\mathbf{x}_{1:T}$	$\mathbf{z}_{t-1}$ (exper.)
DKS (Krishnan et al., 2017)	$\mathbf{z}_{t-1}, \mathbf{x}_{t:T}$	$\emptyset$
DVBF (Karl et al., 2017a)	$\mathbf{x}_t, \mathbf{z}_{t-1}$	$\mathbf{x}_{t+1:T}$
Planet (Hafner et al., 2019)	$\mathbf{x}_t, \mathbf{z}_{t-1}$	$\mathbf{x}_{t+1:T}$
SLAC (A. X. Lee et al., 2020)	$\mathbf{x}_t, \mathbf{z}_{t-1}$	$\mathbf{x}_{t+1:T}$

6.1 A NEW INFERENCE SUBOPTIMALITY

While partial conditioning is most prevalent with sequential LVMs, we analyze it for general LVMs. For the following discussion, we split the observed variables into two disjoint sets of included conditions $\mathcal{C}$ and excluded conditions $\overline{\mathcal{C}}$, i.e., $\mathbf{x} = \mathcal{C} \dot{\cup} \overline{\mathcal{C}}$.

Take DVBFs as an example. The true posterior forward transition would be $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{t:T})$, a DVBF implements $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$ instead, i.e., $\mathcal{C} = \{\mathbf{z}_{t-1}, \mathbf{x}_t\}$ and $\overline{\mathcal{C}} = \{\mathbf{x}_{t+1:T}\}$.

Amortized posteriors are learned by maximizing the *expected* ELBO,

$$\arg\max_{\phi} \mathbb{E}_{p(\mathbf{x})}\left[\ln p(\mathbf{x}) - \mathrm{KL}\big(q_\phi(\mathbf{z} \mid \mathcal{C}) \,||\, p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})\big)\right] \tag{6.1}$$

$$= \arg\min_{\phi} \mathbb{E}_{p(\mathbf{x})}\left[\mathrm{KL}\big(q_\phi(\mathbf{z} \mid \mathcal{C}) \,||\, p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})\big)\right]. \tag{6.2}$$

In eq. (6.1), we deliberately choose the intractable form of the ELBO, which is equivalent to the expected posterior KL, eq. (6.2). The latter is more useful for the following theoretical analysis.

Notice how, in the sequential case that is more interesting to this thesis, the posterior KL decomposes across time as

$$\mathbb{E}_{p(\mathbf{x}_{1:T})}\left[\mathrm{KL}\big(q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) \,||\, p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})\big)\right] \tag{6.3}$$

$$= \sum_{t=1}^{T} \mathbb{E}_{p(\mathcal{C}_t, \overline{\mathcal{C}}_t)}\left[\mathrm{KL}\big(q_\phi(\mathbf{z}_t \mid \mathcal{C}_t) \,||\, p(\mathbf{z}_t \mid \mathcal{C}_t, \overline{\mathcal{C}}_t)\big)\right], \tag{6.4}$$

which again justifies focusing on the general case first, if only for the reduced notation clutter.

Analyzing the amortized objective in eq. (6.2) reveals that

$$\begin{aligned}
&\mathbb{E}_{p(\mathbf{x})}\left[\mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathcal{C}) \,||\, p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right)\right)\right] && (6.5)\\
&= \mathbb{E}_{p(\mathcal{C})}\left[\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathcal{C})}\left[\ln \frac{q_\phi(\mathbf{z} \mid \mathcal{C})}{p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right)}\right]\right]\right] && (6.6)\\
&= \mathbb{E}_{p(\mathcal{C})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathcal{C})}\left[\ln q_\phi(\mathbf{z} \mid \mathcal{C}) - \mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\ln p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right)\right]\right]\right] && (6.7)\\
&= \mathbb{E}_{p(\mathcal{C})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathcal{C})}\left[\ln \frac{q_\phi(\mathbf{z} \mid \mathcal{C})}{\exp\left(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\ln p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right)\right]\right)}\right]\right] && (6.8)\\
&= \mathbb{E}_{p(\mathcal{C})}\left[\mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathcal{C}) \,||\, q_{\mathcal{C}}(\mathbf{z})\right) - \ln \mathcal{F}(\mathcal{C})\right], && (6.9)
\end{aligned}$$

where

$$q_{\mathcal{C}}(\mathbf{z}) \propto \exp\left(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\ln p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right)\right]\right) \tag{6.10}$$

with normalizing constant

$$\mathcal{F}(\mathcal{C}) = \int \exp\left(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\ln p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right)\right]\right) d\mathbf{z}. \tag{6.11}$$

Since $\mathcal{F}$ is constant in q_ϕ, it is easy to see that $q_{\mathcal{C}}(\mathbf{z})$ is the *theoretically* optimal solution of eq. (6.2). Analyzing this optimum further, we make two unfortunate observations:

First, from eq. (6.10), it is immediately clear that the partially-conditioned approximate posterior is in general not equal to the fully-conditioned true posterior,

$$q_{\mathcal{C}}(\mathbf{z}) \neq p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right). \tag{6.12}$$

Secondly, a straightforward reformulation of eq. (6.10),

$$\begin{aligned}
q_{\mathcal{C}}(\mathbf{z}) &\propto \exp\left(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\ln p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right)\right]\right) && (6.13)\\
&= \exp\left(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\ln p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right)\frac{p(\mathbf{z} \mid \mathcal{C})}{p(\mathbf{z} \mid \mathcal{C})}\right]\right) && (6.14)\\
&= p(\mathbf{z} \mid \mathcal{C}) \exp\left(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\ln \frac{p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right)}{p(\mathbf{z} \mid \mathcal{C})}\right]\right), && (6.15)
\end{aligned}$$

shows that, similarly, the partially-conditioned *approximate* posterior is in general not equal to the partially-conditioned *true* posterior,

$$q_{\mathcal{C}}(\mathbf{z}) \neq p(\mathbf{z} \mid \mathcal{C}). \tag{6.16}$$

That is, even if $q_\phi(\mathbf{z} \mid \mathcal{C})$ is capable of representing the desired true posteriors $p(\mathbf{z} \mid \mathcal{C})$ or $p\left(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}\right)$, they will in general not be optimal.

Since the optimum of our objective was shifted, a new inference suboptimality

$$\mathbb{E}_{p\left(\mathcal{C},\overline{\mathcal{C}}\right)}[\mathrm{KL}(q_{\mathcal{C}}(\mathbf{z}) \,\|\, p(\mathbf{z} \mid \mathbf{x}))] \geqslant 0 \tag{6.17}$$

is introduced, which we call the *conditioning gap*. In fact, as we show in appendix C.1, assuming either of eqs. (6.12) and (6.16) to hold true implies

$$p(\overline{\mathcal{C}} \mid \mathbf{z}, \mathcal{C}) = p(\overline{\mathcal{C}} \mid \mathcal{C}), \tag{6.18}$$

i. e., the missing conditions $\overline{\mathcal{C}}$ are conditionally *independent* of the state $\mathbf{z}$ given the used conditions $\mathcal{C}$. Unless this is the case, the inequality in eq. (6.17) is strict and the conditioning gap will not vanish.

Establishing this new gap immediately raises several questions:

1. Can we gain a better understanding of the new optimal approximate posterior $q_{\mathcal{C}}(\mathbf{z})$?
2. Are there potential reasons why the effect had not been discussed in the literature yet?
3. How is the conditioning gap related to the well-established approximation and amortization gaps, cf. section 2.4?
4. Does the provable inference suboptimality also affect the learning of the generative model?
5. How large is the effect of the conditioning gap in practice?

The remainder of this chapter will tackle these questions in order.

6.2 UNDERSTANDING THE CONDITIONING GAP

6.2.1 Understanding the Optimal Approximate Posterior

The first step is a close inspection of the optimal approximate posterior

$$q_{\mathcal{C}}(\mathbf{z}) \propto \exp\Big(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\big[\ln p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})\big]\Big). \tag{6.19}$$

Interestingly, this expression bears superficial similarity to the true partially-conditioned posterior

$$\begin{aligned} p(\mathbf{z} \mid \mathcal{C}) &= \mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\big[p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})\big] \qquad (6.20)\\ &= \exp\Big(\ln\Big(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\big[p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})\big]\Big)\Big). \qquad (6.21) \end{aligned}$$

To understand the difference, consider a uniform discrete $p(\overline{\mathcal{C}} \mid \mathcal{C})$ for the sake of the argument. In this case, the expectation is an average over all missing conditions. The true partially-conditioned posterior $p(\mathbf{z} \mid \mathcal{C})$ is then a mixture distribution of all plausible fully-conditioned posteriors $p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})$. The optimal approximate posterior, because of the logarithm inside the sum, is a *product* of plausible fully-conditioned posteriors.

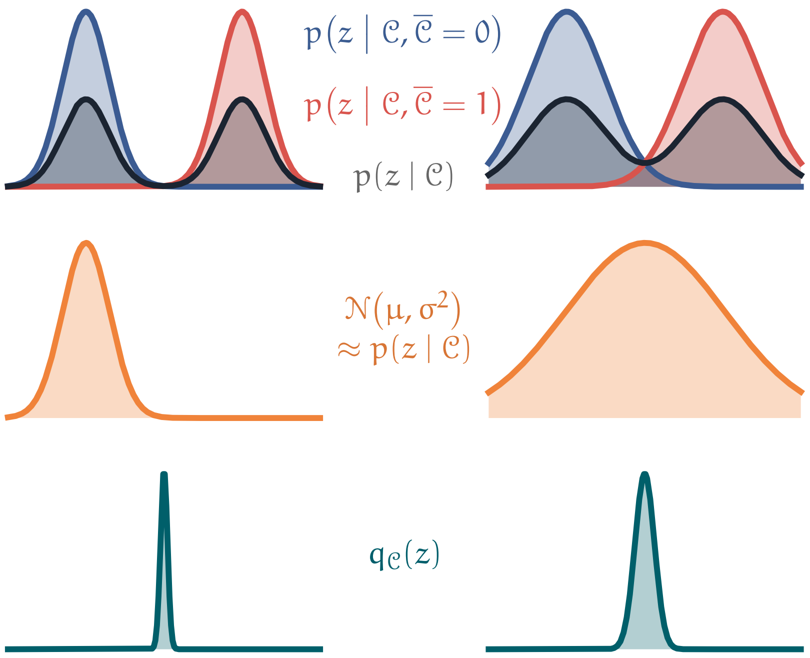


Figure 6.1: Illustration of the effect of partial conditioning on two examples. Consider a latent-variable model $p(\mathcal{C}, \overline{\mathcal{C}} \mid z)p(z)$ with scalar latent variable z and binary $\overline{\mathcal{C}}$. We omit $\overline{\mathcal{C}}$ from the *amortized* approximate posterior $q(z \mid \mathcal{C})$. *Left:* the true fully-conditioned Gaussian posteriors barely overlap. *Right:* the true fully-conditioned posteriors overlap. *Top*: the true fully-conditioned posteriors $p(z \mid \mathcal{C}, \overline{\mathcal{C}} = 0)$ and $p(z \mid \mathcal{C}, \overline{\mathcal{C}} = 1)$ as well as their average, the true partially-conditioned posterior $p(z \mid \mathcal{C})$. *Middle*: Variational Gaussian approximation $\mathcal{N}(\mu, \sigma^2)$ to the marginal posterior, which was obtained by stochastic gradient descent on the reverse, mode-seeking KL-divergence (Hoffman et al., 2013). *Bottom*: The optimal $q_{\mathcal{C}}(\mathbf{z})$ obtained by optimizing the ELBO with a partially-conditioned amortized approximate posterior w. r. t. q. It is located far away from the modes, sharply peaked and shares little mass with the true fully- or partially-conditioned posteriors, as well as the approximate partially-conditioned posterior.

A similar figure has previously appeared in Bayer et al. (2021).

Such distributions occur, e. g., in products of experts (Hinton, 2002) or Gaussian sensor fusion (K. P. Murphy, 2012), as we have discussed previously in the context of fusion DVBFs in section 4.2.2. Products behave differently from mixtures: an intuition due to Welling (2007) is that a factor in a product can single-handedly *veto* a sample while each term in a mixture can only *pass* it.

This intuition is highlighted in fig. 6.1. We can see that $q_{\mathcal{C}}(z)$ is located between the modes and sharply peaked. It shares almost no mass with either of the posteriors $p(z \mid \mathcal{C}), p\big(z \bigm| \mathcal{C}, \overline{\mathcal{C}} = 1\big)$, or $p\big(z \bigm| \mathcal{C}, \overline{\mathcal{C}} = 0\big)$. The best *Gaussian* approximate marginal posterior on the other hand either covers one or two modes, depending on the width of the two full posteriors—a much more reasonable approximation.

It is worth stressing that in this simple example, a single bit missing in the condition is sufficient for the *theoretical* optimum to not be a desirable approximation of *any* true posterior.

Arguably, the properties of the optimal shared posterior are surprising. Partial conditioning means inferring based on less information. One could expect the less informed approximate posterior to be more uncertain, the way $p(\mathbf{z} \mid \mathcal{C})$ is on average more uncertain than $p\big(\mathbf{z} \bigm| \mathcal{C}, \overline{\mathcal{C}}\big)$, cf. appendix A.3. The opposite is true.

A very reduced scalar linear Gaussian example highlights this. The target distribution is the standard Gaussian $p(\mathbf{x}) \sim \mathcal{N}(0, 1)$. We assume the latent variable model $p_a(x, z) = \mathcal{N}(x \mid az, 0.1) \cdot \mathcal{N}(z \mid 0, 1)$ with free parameter $a \geqslant 0$. This implies

$$p_a(x) = \mathcal{N}\big(x \bigm| 0, 0.1 + a^2\big), \tag{6.22}$$

$$p_a(z \mid x) = \mathcal{N}\Big(z \Bigm| a\big(0.1 + a^2\big)^{-1} x, \big(1 + 10a^2\big)^{-1}\Big). \tag{6.23}$$

With $a^* = \sqrt{0.9}$, we recover the target distribution with posterior

$$p_{a^*}(z \mid x) = \mathcal{N}\Big(z \Bigm| \sqrt{0.9}x, 0.1\Big). \tag{6.24}$$

Next, we introduce the variational approximation $q(z) = \mathcal{N}(z \mid \mu_z, \sigma_z^2)$. With the only condition x missing, this is a deliberately extreme case of partial conditioning where $\mathcal{C} = \emptyset$ and $\overline{\mathcal{C}} = \{x\}$. Note that the true posterior is a member of the variational family, i. e., no approximation gap. We maximize the *expected* ELBO

$$q_a^{\mathcal{C}}(z) = \arg\max_{\mu_z, \sigma_z} \mathbb{E}_{p(x)}\left[\mathbb{E}_{q(z)}\left[\ln \frac{p_a(x, z)}{q(z)}\right]\right], \tag{6.25}$$

i. e., all observations from p share the same approximation q. One can show that

$$q_a^{\mathcal{C}}(z) = \mathcal{N}\Big(z \Bigm| 0, \big(100a^2 + 1\big)^{-1}\Big). \tag{6.26}$$

This distribution has extremely low variance, in particular much lower than $p_{a^*}(z \mid x) \equiv p\big(z \bigm| \mathcal{C}, \overline{\mathcal{C}}\big)$ or $p(z) \equiv p(z \mid \mathcal{C})$, the true fully- and partially-conditioned posteriors. We immediately see that $q_a^{\mathcal{C}}(z)$ is generally equal to neither of them. This simple example highlights how poor shared posterior approximations can become.

Further, inserting $q_a^{\mathcal{C}}(z)$ back into the expected ELBO and optimizing for a reveals that the maximum likelihood model parameter a^* is not optimal. In other words, $p_{a^*}(x, z)$ does not optimize the expected ELBO in p—despite being the maximum likelihood model.

6.2.2 Vanishing Conditioning Gap

With these theoretical results in mind, it is reasonable to wonder why the effects of the conditioning gap have not been reported in the literature. After all, they seem to be at odds with the overall positive results including successful applications to applied problems as discussed in part II.

While there may be factors like publication bias (Song et al., 2010) at play on a meta level, there is reason to believe that the types of systems typically studied in the literature are not as affected by the conditioning gap, as the previous sections may suggest. To understand this, we take a closer look at the circumstances for the conditioning gap to disappear,

$$p\big(\overline{\mathcal{C}} \mid \mathbf{z}, \mathcal{C}\big) = p\big(\overline{\mathcal{C}} \mid \mathcal{C}\big) \iff \overline{\mathcal{C}} \perp \mathbf{z} \mid \mathcal{C}. \tag{6.27}$$

At first glance, this may seem like an extreme scenario, but there may be cases where these circumstances are at least approximately true and the gap is small. We highlight two such cases of sequential LVMs.

First, where the partially- and the fully-conditioned posterior correspond to the prior transition, i. e.,

$$p(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \approx p(\mathbf{z}_t \mid \mathcal{C}_t) \approx p\big(\mathbf{z}_t \mid \mathcal{C}_t, \overline{\mathcal{C}}_t\big).$$

This is, for example, the case for deterministic dynamics where the transition is a single point mass.

Second, the case where the observations are sufficient to explain the latent state, i. e.,

$$p(\mathbf{z}_t \mid \mathbf{x}_t) \approx p(\mathbf{z}_t \mid \mathcal{C}_t) \approx p\big(\mathbf{z}_t \mid \mathcal{C}_t, \overline{\mathcal{C}}_t\big).$$

A common case are systems with perfect state information.

We conjecture that the mentioned safe cases are overrepresented in the studied data sets. For example, environments for reinforcement learning such as the OpenAI Gym or MuJoCo environments (Todorov et al., 2012; Brockman et al., 2016) feature deterministic dynamics.

Since it is reasonable to assume that the conditioning gap is negligible even in these relevant standard benchmarks, it is less surprising that its effects have not yet been reported. Indeed, for such scenarios it may even be favorable to use the theoretically incorrect approaches

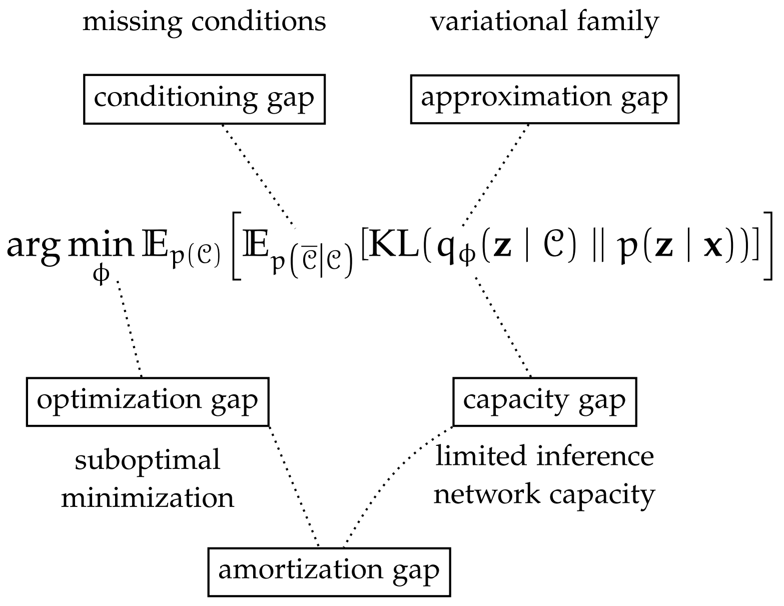

missing conditions
variational family
conditioning gap
approximation gap
$\arg\min_{\phi} \mathbb{E}_{p(\mathcal{C})}\left[\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}[\mathrm{KL}(q_\phi(\mathbf{z} \mid \mathcal{C}) \,\|\, p(\mathbf{z} \mid \mathbf{x}))]\right]$
optimization gap
capacity gap
suboptimal minimization
limited inference network capacity
amortization gap

Figure 6.2: High-level breakdown of the sources of different gaps in (amortized) variational inference as they relate to the expected posterior KL. The posterior KL—equivalent to the ELBO—is the better theoretical tool for understanding the differences between the gaps. Recall that $\mathbf{x} = \mathcal{C} \dot{\cup} \overline{\mathcal{C}}$ and thus $p(\mathbf{x}) = p(\mathcal{C}, \overline{\mathcal{C}})$.

as their practical advantages outweigh the negative impact of the conditioning gap.

Our empirical analysis in section 6.3 will focus on data sets that deviate from these predominant scenarios in the literature.

6.2.3 Relation to Other Gaps

The conditioning gap requires *amortized* VI. Density networks used for amortization introduce the notion of conditions as network inputs. It is their use that enables partial conditioning in the first place.

Seeing that the inference suboptimality is caused by amortization, a natural conclusion would be to subsume the conditioning gap as part of the amortization gap, as discussed in section 2.4. Indeed, this turned out to be a common concern among the reviewers of Bayer et al. (2021).[1] The argument has a certain appeal. From a practical perspective, the root cause for both the amortization and the conditioning gap is the inference density network. Considering the gaps as separate has the air of describing two sides of the same medal.

Establishing the conditioning gap as a separate phenomenon is justified if countermeasures would not emerge from counteracting the amortization gap. In this light, it is worth revisiting the previously

1 Reviews and rebuttals are public at https://openreview.net/forum?id=a2gqxKDvYys.

known gaps—including the approximation gap—and understanding their relation to the conditioning gap.

First, we will briefly discuss the approximation gap. Its root cause is the restricted variational family. If we attempt to approximate, e. g., a non-Gaussian posterior with a Gaussian approximation, this introduces an approximation gap. This gap can be countered by widening the variational family to include more faithful classes of distributions. Our results in section 6.1, however, never assumed a particular distribution of $q(\mathbf{z} \mid \mathcal{C})$, $p(\mathbf{z} \mid \mathcal{C})$, or $p\left(\mathbf{z} \;\middle|\; \mathcal{C}, \overline{\mathcal{C}}\right)$. Widening the variational family would not help closing the conditioning gap at all. Even worse, we showed that the desired solutions $p(\mathbf{z} \mid \mathcal{C})$ or $p\left(\mathbf{z} \;\middle|\; \mathcal{C}, \overline{\mathcal{C}}\right)$ are generally not optimal in the objective even if they are members of the variational family.

The amortization gap is caused by the use of density networks for inference. Empirically, the mapping from conditions to *the* optimal member of the variational family cannot be represented perfectly by the inference density network. This discrepancy is the amortization gap and is caused by two factors. Firstly, while neural networks are theoretically universal function approximators, in practice their finite amount of parameters and particular architecture restricts the class of functions a neural network can represent well. We refer to this as the neural network having a *limited capacity*. The second cause for an amortization gap is that the density network is learned by imperfect first-order optimization methods.[2] Countering this gap thus involves either the fine art of increasing the capacity of the inference density network, or the optimization method to learn its parameters, or likely a combination of both.

Like with the approximation gap, we observe that neither of these interventions affects the conditioning gap at all. The amortization gap quantifies how much the inference density network misses the optimum. The conditioning gap on the other hand is caused by the optimum being shifted, recall fig. 6.1. Even in a hypothetical scenario where the inference density network is a true universal approximator—infinite capacity—and an oracle optimizer that guarantees optimal parameters for a given objective, the shifted optimum implies that the conditioning gap remains present. The fact that the countermeasures to reduce the amortization gap have no effect on the conditioning gap are a first strong indication that it is worth discussing it as separate phenomenon.

This is backed by another observation. Both approximation and amortization gaps are defined on the level of individual data samples. For a particular $\mathbf{x}$, the true posterior $p(\mathbf{z} \mid \mathbf{x})$ is not a member of the

2 One might argue that the amortization gap itself consists of two separate gaps, a *capacity* gap and an *optimization* gap. Making this distinction is potentially interesting because a non-amortized VI method like SVI also suffers from the optimization gap but not from the capacity gap. A full classification—and value judgment of its usefulness—is out of scope for this thesis and left to future research.

variational family; for a particular $\mathbf{x}$, the inference density network misses the optimal member of the variational family. The conditioning gap on the other hand cannot be characterized from this point of view. To define the conditioning gap, all plausible observations $\mathbf{x}$ that share the same condition $\mathcal{C}$ always need to be considered. All these observations necessarily share the same optimal $q_{\mathcal{C}}(\mathbf{z})$. The conditioning gap can only be thought of in terms of expectations w. r. t. $p(\mathbf{x})$. No neural network can avoid the foul compromise that is the new optimum. In this sense, the cause of the conditioning gap is much less isolated in the inference density network than the amortization gap. From this vantage point, it becomes clear that the conditioning gap cannot be a part of the amortization gap—say, in addition to optimization and capacity gap.

The discussion of this section is summarized in fig. 6.2.

Two main routes to counter the conditioning gap come to mind. The first is a careful analysis of the system to be learned. If there is reason to suspect that it falls, e. g., into one of the two benevolent categories discussed earlier, one may simply choose to ignore the conditioning gap in favor of a practically useful inference density network.

The second is to decouple learning the inference model from learning the generative model. That is to say, for learning the generative model one should attempt to reduce the conditioning gap by feeding all relevant conditions to the inference density network even if the resulting density network is not useful for downstream application. In a second step, after the generative model has converged, one can then seek to fit a separate inference model based on the conditions that are available in the downstream application. Here, it is likely advisable to use a different objective than a lower bound to $p(\mathbf{x})$ or $p(\mathbf{x}_{1:T})$ but a more targeted objective that explicitly seeks to approximate the true posterior that matches the intended application. Efficient curricula along these lines are left to future research. The bottom line is that, depending on the system to be learned, the conditioning gap is a strong argument against currently popular joint end-to-end learning of inference and generative model.

6.2.4 The Conditioning Gap and the Generative Model

The analysis so far has been centered on the inference model and inference suboptimality caused by the conditioning gap. Adding the generative model—which we usually try to fit in conjunction in the VAE framework—back into the picture begs the question how much it is affected by the conditioning gap.

A simple variational calculus argument, appendix C.2, reveals that, when trained with partially-conditioned approximate posteriors, maximum-likelihood models and ELBO-optimal models are generally not the same. While this may not be surprising, the interesting question is how large the discrepancy is. We hypothesize that during the learning process the generative model adapts in a way that reduces the conditioning gap at the cost of overall model quality. That is to say, instead of learning an approximately optimal model in the sense of maximum likelihood, a model with high likelihood subject to low conditioning gap is learned.

For sequential LVMs, we believe this to lead to inferior *predictive* models when future observations are part of the missing conditions. The reason is that in this scenario all plausible futures are mapped to the same optimal yet foul compromise. It is thus not possible for this belief to carry relevant information to predict the future observations. We will put this hypothesis to a test in our empirical study.

6.2.5 Further Comments

It should be noted that the analysis of section 6.1 hinges on the posterior KL as the objective. While it is a natural choice given its duality to the ELBO, our objective of choice in this work, it is by no means the only option. It is an open question to what extent other divergences (Ranganath et al., 2014; Li and Richard E. Turner, 2016), cf. also appendix A.2, suffer from similar conditioning gaps and if these lead to practical learning algorithms.

Theoretical studies of biases of learning algorithms based on the ELBO are numerous in the literature (Richard Eric Turner and Sahani, 2011; Nowozin, 2018; Huang and Courville, 2019). These studies are largely orthogonal to our perspective here, which is deeply rooted in amortized inference in particular.

However, we want to address one study by Lai et al. (2019). Their study sets out to examine the role of sequential LVMs. They conclude that the added stochasticity is not only unhelpful but potentially even harmful to the performance of the overall system. Crucially, their experimental protocol restricts the inference density networks in a way that induces a conditioning gap, of which the study was unaware. We conjecture that such assumptions lead to a collapse of the model where the latent variables merely help to explain the data local in time, i. e., intra-step correlations. In the light of the conditioning gap, their results need to be re-examined.

6.3 EMPIRICAL STUDY

We studied the conditioning gap and its impact on learning SSMs with amortized VI. In section 6.2.2 we have discussed that the conditioning gap is likely to be less relevant in commonly studied data sets. We thus looked at three different data sets: *unmanned aerial vehicle* (UAV) trajectories with imperfect state information, section 6.3.2, a sequential version of the MNIST data set, section 6.3.3, and traffic flow, section 6.3.4.

6.3.1 Experimental Setup

We implement an SSM

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_t) p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) \tag{6.28}$$

with density networks. Controls $\mathbf{u}_{1:T}$ are used in the UAV scenario. Like with residual DVBFs, section 4.2.1, we choose a residual formulation of the transition.

This is to ease the implementation of comparable inference models

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \tag{6.29}$$

$$= q(\mathbf{z}_1 \mid \mathbf{x}_{1:k}, \mathbf{u}_{1:k}) \prod_{t=2}^{T} q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{1:m}, \mathbf{u}_{1:m}) \tag{6.30}$$

that allow us to examine the conditioning gap for different configurations of k and m. Similar to residual DVBFs, the inference density networks implement

$$q(\boldsymbol{\epsilon}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_{1:m}, \mathbf{u}_{1:m}), \tag{6.31}$$

and then

$$q(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_{1:m}, \mathbf{u}_{1:m}) \tag{6.32}$$

is implicitly defined through the residual formulation.

The initial inference model $q(\mathbf{z}_1 \mid \mathbf{x}_{1:k}, \mathbf{u}_{1:k})$ is also inspired by DVBFs in that we may allow a sneak peek of k steps ahead even when the inference model generally mimics filtering.

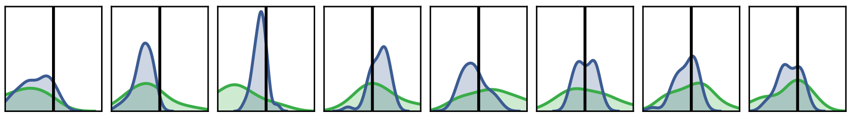
Concretely, we consider the configurations:

1. $k = m = T$. We refer to this as fully-conditioned inference.
2. $k = 1, m = t$. We refer to this as partially-conditioned inference.
3. $k > 1, m = t$. We refer to this as semi-conditioned inference.

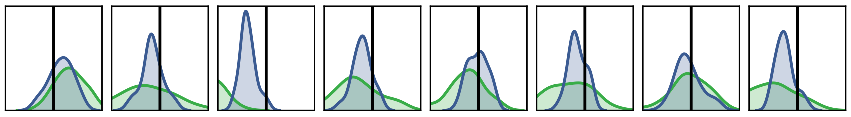
The semi-conditioned inference model is motivated by the same arguments as the initial inference model of DVBFs. It is a less severe form of partial conditioning. If the conditioning gap is relevant in practice, we should see the first fully-conditioned configuration outperforming the latter two.

In practice, the translation from eq. (6.31) to eq. (6.32) is yet another density network

$$q(\mathbf{z}_t \mid f(\mathbf{z}_{t-1}, \mathbf{u}_{t-1}), \mathbf{h}_t) \tag{6.33}$$



(a) x-axis



(b) y-axis

Figure 6.3: Posterior-predictive check of prefix-sampling on the Blackbird data set. Possible futures are sampled from the model after having observed a prefix $\mathbf{x}_{1:t}$. The state at the end of the prefix is inferred with a bootstrap particle filter. Each plot shows a kernel density estimate of the distribution over the final location $\mathbf{x}_T$, once for the semi-conditioned model in green and for the fully-conditioned in blue. The true value is marked as a vertical, black line. The fully-conditioned model assigns higher likelihood in almost all cases and is more concentrated around the truth.
This figure has previously appeared in Bayer et al. (2021).

Table 6.2: ELBO values for models with differently conditioned variational posteriors for various data sets. Presented values are averages over ten samples from the inference model. The standard deviations were negligible. Higher is better.
This table has previously appeared in Bayer et al. (2021).

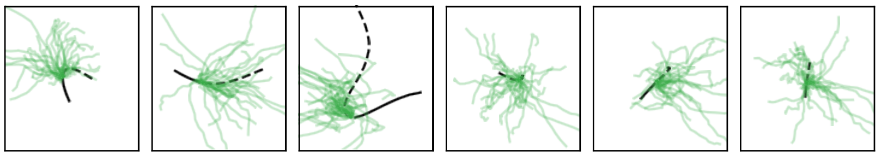
	UAV		Traffic Flow	
	val	test	val	test
partial	-	-	−2.91	−2.97
semi	1.47	2.13	−2.73	−2.75
full	2.03	2.41	−2.69	−2.78

that uses the prior mean prediction $f(\mathbf{z}_{t-1}, \mathbf{u}_{t-1})$. This is to maximally exploit the prior during inference and avoid duplicate learning of dynamics. The deterministic feature vector $\mathbf{h}_t$ is computed differently depending on the configuration. The interface via the feature vector allows us to easily change the conditioning. For the fully-conditioned configuration, $\mathbf{h}_{1:T}$ is the output of a *bidirectional* RNN with inputs $\mathbf{x}_{1:T}$ and $\mathbf{u}_{1:T}$. For the the other two configurations, $\mathbf{h}_1$ is computed with an MLP with inputs $\mathbf{x}_{1:k}$ and $\mathbf{u}_{1:k}$, and $\mathbf{h}_{2:T}$ are computed with a *unidirectional* RNN with inputs $\mathbf{x}_{1:T}$ and $\mathbf{u}_{1:T}$.[3]

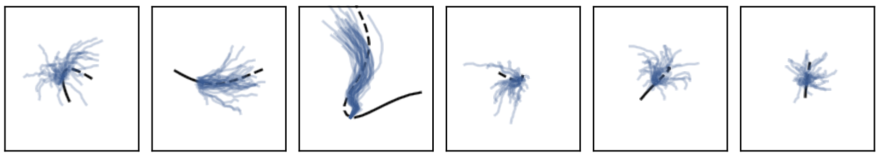
6.3.2 UAV Trajectories

With these models, we learn UAV trajectories from the Blackbird data set (Antonini et al., 2018). By discarding the rotational information,

3 For further implementation details, we refer to the appendices of Bayer et al. (2021).



(a) semi-conditioned



(b) fully-conditioned

Figure 6.4: Comparison of prefix-sampling for a top-down view of UAV data. Possible futures $\hat{\mathbf{x}}^{(i)}_{k+1:T}$ (colored lines) are sampled from the model after having observed a prefix $\mathbf{x}_{1:k}$ (solid black line) and then compared to the true suffix $\mathbf{x}_{k+1:T}$ (dashed line). The state at the end of the prefix is inferred with a bootstrap particle filter.
A similar figure has previously appeared in Bayer et al. (2021).

we create a system with imperfect state information. This creates a system in which, according to our previous discussion, we expect a conditioning gap because the observation $\mathbf{x}_t$ does not contain the full dynamic state. Each observation $\mathbf{x}_t \in \mathbb{R}^3$ is the location of an *unmanned aerial vehicle* (UAV) in a fixed global frame. The conditions $\mathbf{u}_t \in \mathbb{R}^{14}$ consist of IMU readings, rotor speeds, and pulse-width modulation. The emission model was implemented as a Gaussian density network with fixed, hand-picked standard deviations, where the mean corresponds to the first three state dimensions:

$$p(\mathbf{x}_t \mid \mathbf{z}_t) = \mathcal{N}\big(\mu = \mathbf{z}_{t,1:3}, \sigma^2 = [0.15, 0.15, 0.075]\big). \tag{6.34}$$

This approach is reminiscent of the explicit latent variables in VTSSI.

We leave out the partially-conditioned case as it cannot infer the higher-order derivatives necessary for rigid-body dynamics. A sneak peek of $k = 7$ for the semi-conditioned model is theoretically sufficient to infer those moments.

Fully-conditioned models outperform semi-conditioned ones on the test set ELBO, as can be seen in table 6.2. We evaluated the models on prefix-sampling, i. e., the predictive performance of

$$p(\mathbf{x}_{t+1:T} \mid \mathbf{x}_{1:t}, \mathbf{u}_{1:T}). \tag{6.35}$$

To restrict the analysis to the found parameters of the generative model only, we inferred the filter distribution $p(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{u}_{1:t})$ using a bootstrap particle filter, cf. section 3.3.2.2. By not using the respective approximate posteriors, we ensure fairness between the different models since the fully-conditioned model would otherwise be at an advantage. Inference would have already taken into account the observations to be predicted. Samples from the predictive distribution were obtained

Table 6.3: Results for row-wise MNIST. We report the ELBO as a lower bound on the log likelihood and the KL divergence of the digit distribution induced by the model from a uniform distribution.
This table has previously appeared in Bayer et al. (2021).

Distribution	Log-Likelihood ↑	KL ↓
data	-	0.002
partial	$\geqslant -98.99 \pm 0.06$	0.098
full	$\geqslant -88.45 \pm 0.05$	0.015
vhp + rewo (Klushyn et al., 2019)	$\geqslant -82.74$	-
iwae (L=2) (Klushyn et al., 2019)	$\geqslant -82.83$	-

via ancestral sampling of the generative model. Representative samples are shown in fig. 6.4. We performed a posterior-predictive check for both models, where we compare the densities of the final observations $\mathbf{x}_T$ obtained from prefix sampling in fig. 6.3. Both evaluations qualitatively illustrate that the predictions of the fully-conditioned approach concentrate more around the true values. In particular the partially-conditioned model struggles more with long-term prediction.

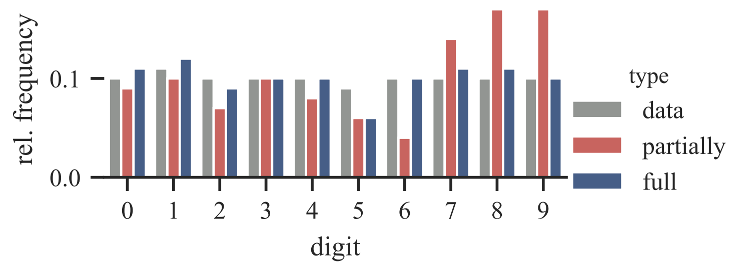
6.3.3 Row-Wise MNIST

We transformed the MNIST data set into a sequential data set by considering one row in the image plane per time step, from top to bottom. This results in stochastic dynamics: similar initial rows can result in a 3, 8, 9, or 0, future rows are very informative. Before all experiments, each pixel was binarized by sampling from a Bernoulli with a rate in $[0, 1]$ proportional to the corresponding pixel intensity.

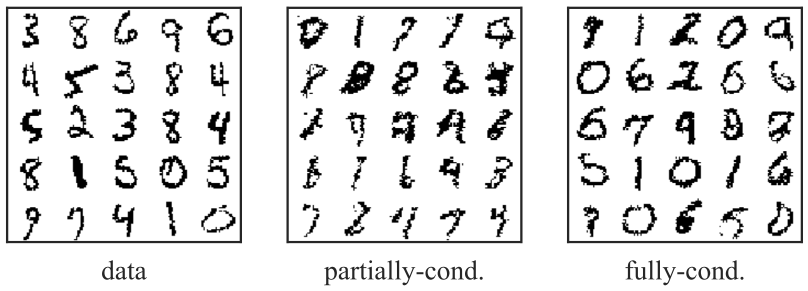
The setup was identical to that of section 6.3.2, except that a density network returning a Bernoulli was used. No conditions $\mathbf{u}_{1:T}$ and a short sneak-peek ($k = 1$) were used. The fully-conditioned model outperforms the partially-conditioned by a large margin, placing it significantly closer to state-of-the-art performance, see table 6.3. This is supported by samples from the model, see fig. 6.5. Note that state-of-the-art results cannot be expected to be achieved since an SSM is arguably not an ideal generative model of handwritten digits.

For qualitative evaluation, we used a state-of-the-art classifier[4] to categorise 10 000 samples from each of the models. If the data distribution is learned well, we expect the classifier to behave similarly on both data and generative samples, i. e., yield uniformly distributed class predictions. We report KL divergences from the uniform distribution of the class prediction distributions in table 6.3. A bar plot of the induced class distributions can be found in fig. 6.5. Only the fully-conditioned model is able to nearly capture a uniform distribution.

4 https://github.com/keras-team/keras/blob/2.4.0/examples/mnist_cnn.py



(a) Class distributions of the respective image distributions induced by a state-of-the-art classifier. The data distribution is close to uniform, except for 5. The fully-conditioned model yields too few 5s and is close to uniform for the other digits. The partially-conditioned model only captures the right frequencies of 1 and 3.



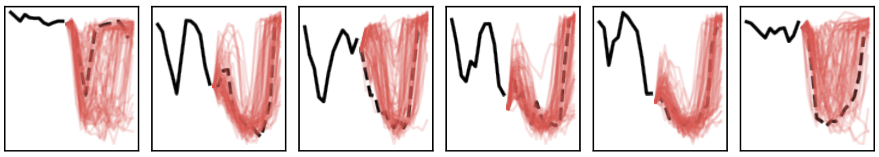
(b) Comparison of generative sampling on row-wise MNIST. Samples from the data distribution are shown on the left. The middle and right show samples from models with a partially- and a fully-conditioned approximate posterior, respectively.

Figure 6.5: Results for the row-wise MNIST data.
A similar figure has previously appeared in Bayer et al. (2021).

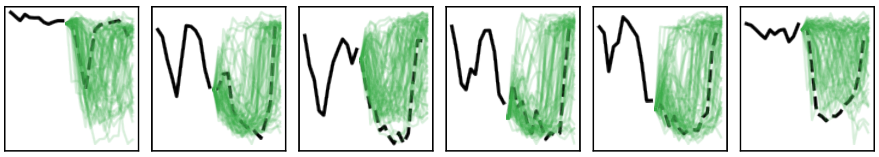
6.3.4 Traffic Flow

We consider the Seattle loop data set (Cui et al., 2019, 2020) of average speed measurements of cars at 323 locations on motorways near Seattle, from which we selected a single one (42). The dynamics of this system are highly stochastic, which is of special interest for our study. Even though all days start out very similar, traffic jams can emerge suddenly. In this scenario the emission model was a Gaussian density network conditioned on the whole latent state. We compare partially-, semi- ($k = 7$) and fully-conditioned models. The results are shown in table 6.2. While the fully-conditioned posterior emerges as the best choice on the validation set, the semi-conditioned and the fully-conditioned one are on par on the test set. We suspect that the *sneak peek* is sufficient to fully capture a sensible initial state approximation.

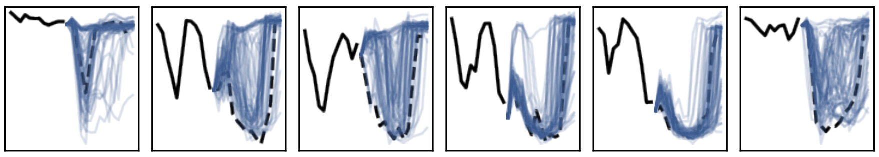
We performed a qualitative evaluation of this scenario as well in the form of prefix sampling. Given the first $t = 12$ observations, the task is to predict the remaining ones for the day, compare section 6.3.2. We show the results in fig. 6.6. The fully-conditioned model clearly shows more concentrated yet multi-modal predictive distributions. The partially-condition model concentrates too much, and the semi-conditioned one too little.



(a) partially-conditioned



(b) semi-conditioned



(c) fully-conditioned

Figure 6.6: Comparison of prefix-sampling. Same as fig. 6.4 but for the traffic flow data.
A similar figure has previously appeared in Bayer et al. (2021).

6.4 DISCUSSION

In this chapter, we have gathered strong theoretical and empirical evidence that the common practice of under-conditioning amortized inference models harms learning. In particular, our empirical study shows that not only inference is affected but also the generative learning that is learned in conjunction.

New learning algorithms that minimize—or altogether avoid—the conditioning gap need to be devised. For many applications, such as model-based control or RL with SSMs, learning the model may need to be split from learning to infer. Neither part is straightforward; we will consider one model learning algorithm without conditioning gap in chapter 7.

Considering the overall promising results of part II, the results of this chapter may appear anticlimactic. Yet, they also allow for a more optimistic interpretation: by acknowledging and countering the conditioning gap, one may find the basis upon which learning sequential LVMs becomes feasible in broader classes of systems, particularly systems with more inherent uncertainty. Future research may lead to model designs and learning algorithms that are less tailored to a particular application as many of the algorithms discussed in part II, establishing the VAE framework more firmly for dynamical systems.

7 APPROXIMATE NEURAL SMOOTHING

Chapter 6 explicitly outlines how sequential VAEs suffer from the conditioning gap when the approximate posterior density network uses different inputs compared to what the true posterior conditions prescribe. This upcoming chapter combines the insights from part II and chapter 6 into a new learning algorithm for SSMs.

From these experiences, we derive a set of design principles. The inference model

1. must not exhibit a conditioning gap, it is thus smoothing;
2. should reuse generative SSM components to maximum extent;
3. should not impose Gaussian restrictions out of the gate.

As a consequence of these principles, the inference model needs some way of handling beliefs that are not Gaussian. The most straightforward approach that is largely agnostic to certain distributions is importance sampling, which leads to a particle smoothing approach. This has the added benefit of not only maintaining a full posterior belief at all times, albeit approximated by particles, but in the process does away with the single-sample MC integrations.

The remainder of this chapter fleshes out the design principle into an inference model and subsequently a learning algorithm for SSMs and concludes with a proof of concept.

7.1 FAITHFUL APPROXIMATE SMOOTHING

Much like the design of DVBFs was inspired by Bayesian filters in general and the Kalman filter in particular, cf. chapter 4, we once again turn our attention to Bayesian *smoothers* for inspiration.

The crucial building blocks are already known from the background section 3.3.1. According to eq. (3.25), the Bayesian SSM posterior factorizes as

$$p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t+1:T}). \tag{7.1}$$

With eqs. (3.14), (3.24), and (3.30), we can refine the factors further:

$$\begin{aligned} p(\mathbf{z}_1 \mid \mathbf{x}_{1:T}) &\propto p(\mathbf{z}_1)p(\mathbf{x}_1 \mid \mathbf{z}_1)\beta_1(\mathbf{z}_1), & (7.2) \\ p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t+1:T}) &\propto p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)p(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1})\beta_{t+1}(\mathbf{z}_{t+1}). & (7.3) \end{aligned}$$

This view on the posterior is immediately appealing for designing an approximation since it makes use of the SSM components. Reusing these components creates a level of entanglement between posterior approximation and generative model that may provide a useful inductive bias for model learning. We observed similar advantages with DVBFs in chapter 4.

In an eventual approximation the only source of the *approximation* gap is how well the backward filter β is approximated. The proportionality and the fact that β is not a distribution but a likelihood function make it at least theoretically more plausible to make the lower bound tight because it is not hampered by variational family choices.

In this light, eqs. (7.2) and (7.3) pose two challenges to a concrete implementation. The first is the proportionality of both equations. This does not necessarily pose a problem to *inference* as seen with, e. g., particle filters. Yet, pdf evaluations are not straightforward, which needs to be accounted for during *learning* with the ELBO. We will investigate this in section 7.2

The second is the backward filter $\beta_t(\mathbf{z}_t)$. We observe that we can reuse initial state distribution, transition, and emission models by approximating the posterior based on eqs. (7.2) and (7.3).[1] The only unknown—and in fact intractable—component is the backward filter $\beta_t(\mathbf{z}_t)$. This leads to the core idea of this new approach: instead of an explicit closed-form approximate posterior we *implicitly* define an approximate posterior

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t+1:T}), \tag{7.4}$$

with

$$\begin{aligned} q(\mathbf{z}_1 \mid \mathbf{x}_{1:T}) &\propto p(\mathbf{z}_1)p(\mathbf{x}_1 \mid \mathbf{z}_1)b_1(\mathbf{z}_1), && (7.5) \\ q(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t+1:T}) &\propto p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)p(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1})b_{t+1}(\mathbf{z}_{t+1}), && (7.6) \end{aligned}$$

where we use an *approximate* backward filter $b_t \approx \beta_t$. Using SIS akin to particle filters with target q, we require neither distribution nor normalizing constants to obtain samples.

The backward filter

$$\beta_t(\mathbf{z}_t) = p(\mathbf{x}_{t+1:T} \mid \mathbf{z}_t) \tag{7.7}$$

is a function, not a distribution in $\mathbf{z}_t$. It comes, however, with a demanding constraint: as a likelihood function of the future observations it is required to be non-negative and integrate to 1 over the observations. However, since eqs. (7.5) and (7.6) are already proportional, we

1 We note that, in contrast to the reuse in, e. g., DVBFs, the prior here needs not be reparemetrizable as long as we can evaluate its pdf.

may relax the latter.[2] Instead of directly approximating β_t with a normalized variational distribution $\hat{\beta}_t$, we learn a proportional positive function

$$b_t(\mathbf{z}_t) \propto \hat{\beta}_t(\mathbf{z}_t) \approx \beta_t(\mathbf{z}_t). \tag{7.8}$$

From this viewpoint, even $\hat{\beta}_t$ is now implicit to our method. Like q, and in contrast to b_t, it is never explicitly implemented.

7.1.1 Approximate Neural Smoothing

Aside from these considerations, we have not assumed a particular implementation of q or b_t. To implement an SIS approach, we need to provide a specific implementation for the proposals as well as the approximate backward filter. The latter has not been used in the previous literature, and both differ from the generative model in that they require to work with inputs of variable size.

Both need to operate on a variable number of future observations. This lends itself to RNNs:

$$\mathbf{f}_{1:T} = \mathrm{RNN}_b(\mathbf{x}_{1:T}), \tag{7.9}$$
$$\ln b_t(\mathbf{z}_t) = \mathrm{MLP}_b(\mathbf{z}_t, \mathbf{f}_{t+1}), \tag{7.10}$$

$$\mathbf{g}_{1:T} = \mathrm{RNN}_\pi(\mathbf{x}_{1:T}), \tag{7.11}$$
$$\pi(\mathbf{z}_1 \mid \mathbf{x}_{1:T}) = \mathrm{MLP}_\pi(\mathbf{g}_1), \tag{7.12}$$
$$\pi(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{x}_{t:T}) = \mathrm{MLP}_\pi(\mathbf{g}_t, \mathbf{z}_t). \tag{7.13}$$

The RNNs in eqs. (7.9) and (7.11) compute fixed-size future summaries $\mathbf{f}_t \in \mathbb{R}^{d_f}$ and $\mathbf{g}_t \in \mathbb{R}^{d_g}$ *backwards in time*. For instance, $\mathbf{f}_t$ is always a function of only $\mathbf{x}_{t:T}$ to avoid data leakage by ensuring congruence of inputs with the definition of a backward filter, eq. (7.7). This also requires an uninformed $\mathbf{f}_T$, which is set to be a learnable variable.

By using the same networks at all time steps, we can share weights between all approximations. We can even resort to using only one RNN between proposal and approximate backward filter to summarize the future observations. However, due to reasons discussed in section 7.3.1 this should be done with care.

Concretely, eqs. (7.12) and (7.13) are implemented by density networks returning distribution parameters as discussed in section 1.1.4.

Equation (7.10) could be implemented by a vanilla network since, as discussed, b_t is only proportional to a likelihood function, and the subsequent exponentiation ensures non-negativity. In early experiments, we found that constraining it to be a (log) probability density in $\mathbf{z}_t$ to be favorable. This can be implemented like the proposal.

2 Technically, the approximation needs to have finite integral in the future observations $\mathbf{x}_{t+1:T}$. We did not encounter any problems with neglecting this tricky constraint.

Such an approach is further justified by the observation that in the simple case of an LGS the true backward filter can be written as

$$\beta_t(\mathbf{z}_t) = \mathcal{N}(\mathbf{x}_{t+1:T} \mid \mathbf{B}_t\mathbf{z}_t, \boldsymbol{\Sigma}_t) \tag{7.14}$$

for appropriate matrices $\mathbf{B}_t$ and $\boldsymbol{\Sigma}_t$, cf. appendix C.4. Despite being a likelihood function, this is a bell-curve function in $\mathbf{z}_t$. As such, it is proportional to a Gaussian pdf in $\mathbf{z}_t$, and thus can be represented by the proposed implementation.

The resulting algorithm is shown in algorithm 5. It bears close resemblance to particle filters in algorithm 2. The major differences are an adapted update function

$$\gamma_t(\mathbf{z}_t, \mathbf{z}_{t-1}) = \frac{p(\mathbf{x}_t \mid \mathbf{z}_t)p(\mathbf{z}_t \mid \mathbf{z}_{t-1})b_t(\mathbf{z}_t, \mathbf{x}_{t+1:T})}{\pi(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{1:T})}, \tag{7.15}$$

which, compared to the particle filter, adds the approximate backward filter to the numerator. Further, the proposal $\pi(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_{1:T})$ may now also use future observations. We call this new amortized inference algorithm *approximate neural smoothing* (ANS).

7.2 ESTIMATING THE SEQUENTIAL ELBO

In order to use ANS for learning, we need to estimate the sequential ELBO

$$\mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\ln \frac{p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})}\right]. \tag{7.16}$$

Due to the implicit nature of our approximation q even a partial evaluation of, e. g., the prior KL like with DVBFs is not possible. However, SIS equips us with (weighted) samples of q so that an MC estimate is possible.

We thus turn our attention to the log ratio in eq. (7.16). We denote the step-wise normalizing constants of the approximate posterior as

$$Z_t(\mathbf{z}_{t-1}) = \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1})p(\mathbf{x}_t \mid \mathbf{z}_t)b_t(\mathbf{z}_t)\, d\mathbf{z}_t, \tag{7.17}$$

$$Z_1 = \int p(\mathbf{z}_1)p(\mathbf{x}_1 \mid \mathbf{z}_1)b_1(\mathbf{z}_1)\, d\mathbf{z}_1. \tag{7.18}$$

Our deliberate choice of reusing the generative model as much as possible for the approximate posterior now has an interesting effect on this log ratio. All components of the generative model cancel:

$$\ln \frac{p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})} \tag{7.19}$$

$$= \ln \frac{p(\mathbf{z}_1)\prod_{t=1}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)\prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_t)Z_t(\mathbf{z}_{t-1})}{p(\mathbf{z}_1)\prod_{t=1}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)b_t(\mathbf{z}_t)\prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_t)}. \tag{7.20}$$

Algorithm 5: Approximate Neural Smoothing.

```
Input:  observations $\mathbf{x}_{1:T}$; number of particles $P$;
        SSM initial, transition, and emission distribution;
        proposal distributions $\pi(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_{1:T})$;
        resampling criterion and technique
Output: weighted state particle trajectories
        $\left\{\left\{\left(w_t^{(p)}, \mathbf{z}_{1:t}^{(p)}\right)\right\}_{p=1,\dots,P}\right\}_{t=1,\dots,T}$

Initialize $w_0^{(p)} = 1/P, p = 1, \dots, P$
for $t = 1, \dots, T$ do
    for $p = 1, \dots, P$ do
        Proposals $\mathbf{z}_t^{(p)} \sim \pi\left(\mathbf{z}_t \;\middle|\; \mathbf{z}_{1:t-1}^{(p)}, \mathbf{x}_{1:T}\right)$
        Updates
        $\gamma_t^{(p)} \equiv \gamma_t\left(\mathbf{z}_t^{(p)}, \mathbf{z}_{t-1}^{(p)}\right) = \frac{p\left(\mathbf{x}_t \middle| \mathbf{z}_t^{(p)}\right) p\left(\mathbf{z}_t^{(p)} \middle| \mathbf{z}_{t-1}^{(p)}\right) b_t\left(\mathbf{z}_t^{(p)}, \mathbf{x}_{t+1:T}\right)}{\pi\left(\mathbf{z}_t^{(p)} \middle| \mathbf{z}_{1:t-1}^{(p)}, \mathbf{x}_{1:T}\right)}$
        Unnorm. weights $\hat{w}_t^{(p)} = w_{t-1}^{(p)} \gamma_t^{(p)}$
    end
    Renormalize weights $w_t^{(p)} = \hat{w}_t^{(p)} / \sum_{r=1}^{P} \hat{w}_t^{(r)}$
    if criterion is met then
        $\left\{\left(w_t^{(p)}, \mathbf{z}_{1:t}^{(p)}\right)\right\} = \texttt{resample}\left(\left\{\left(w_t^{(p)}, \mathbf{z}_{1:t}^{(p)}\right)\right\}\right)$
    end
end
def resample$\left(\left\{\left(w_t^{(p)}, \mathbf{z}_{1:t}^{(p)}\right)\right\}\right)$:
    Save temporary copy $\hat{\mathbf{z}}_{1:t}^{(p)} = \mathbf{z}_{1:t}^{(p)}$
    Updated particle indexes $\{i_p\} = \texttt{technique}\left(\left\{w_t^{(p)}\right\}\right)$
    for $p = 1, \dots, P$ do
        Set $w_t^{(p)} = 1/P$
        Overwrite trajectories $\mathbf{z}_{1:t}^{(p)} = \hat{\mathbf{z}}_{1:t}^{(i_p)}$
    end
```

We are left with[3]

$$\mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\ln \frac{p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})}\right] \tag{7.21}$$

$$= \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\ln Z_1 + \sum_{t=1}^{T-1} \ln \frac{Z_{t+1}(\mathbf{z}_t)}{b_t(\mathbf{z}_t)}\right] \tag{7.22}$$

$$= \sum_{t=1}^{T} \underbrace{\mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{x}_{1:T})}[\ln Z_t(\mathbf{z}_{t-1})]}_{A} - \underbrace{\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{1:T})}[\ln b_t(\mathbf{z}_t)].}_{B} \tag{7.23}$$

The latter term B can immediately be estimated via importance sampling with the weighted particles. The log normalizing constants in A require some thought. The outer expectation can be approximated with the weighted particles as well. We further observe that

$$Z_t\left(\mathbf{z}_{t-1}^{(p)}\right) = \mathbb{E}_{\pi\left(\mathbf{z}_t \middle| \mathbf{z}_{t-1}^{(p)}, \mathbf{x}_{t:T}\right)}\left[\gamma_t\left(\mathbf{z}_t, \mathbf{z}_{t-1}^{(p)}\right)\right]. \tag{7.24}$$

This can be estimated as

$$Z_t\left(\mathbf{z}_{t-1}^{(p)}\right) \approx \frac{1}{N} \sum_{n=1}^{N} \gamma_t\left(\mathbf{z}_t^{(n,p)}, \mathbf{z}_{t-1}^{(p)}\right) \tag{7.25}$$

by drawing N proposals[4]

$$\mathbf{z}_t^{(n,p)} \sim \pi\left(\mathbf{z}_t \;\middle|\; \mathbf{z}_{t-1}^{(p)}, \mathbf{x}_{t:T}\right). \tag{7.26}$$

Putting things together, we get

$$\mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{x}_{1:T})}[\ln Z_t(\mathbf{z}_{t-1})] \tag{7.27}$$

$$= \mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{x}_{1:T})}\left[\ln \mathbb{E}_{\pi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T})}[\gamma_t(\mathbf{z}_t, \mathbf{z}_{t-1})]\right] \tag{7.28}$$

$$\approx \sum_{p=1}^{P} w_{t-1}^{(p)} \ln\left(\frac{1}{N}\sum_{n=1}^{N} \gamma_t\left(\mathbf{z}_t^{(n,p)}, \mathbf{z}_{t-1}^{(p)}\right)\right). \tag{7.29}$$

In total, we can get an estimator of the ELBO as

$$\sum_{t=1}^{T}\left[\underbrace{\sum_{p=1}^{P} w_{t-1}^{(p)} \ln\left(\frac{1}{N}\sum_{n=1}^{N}\gamma_t^{(n,p)}\right)}_{\approx A} - \underbrace{\sum_{p=1}^{P} w_t^{(p)} \ln b_t^{(p)}}_{\approx B}\right], \tag{7.30}$$

where

$$\gamma_t^{(n,p)} = \gamma_t\left(\mathbf{z}_t^{(n,p)}, \mathbf{z}_{t-1}^{(p)}\right), \quad b_t^{(p)} = b_t\left(\mathbf{z}_t^{(p)}\right). \tag{7.31}$$

In eq. (7.30), note the different time indexes of weights in A and B. They prevent the backward filter term from canceling with its contribution to the update term.

3 Equation (7.23) defines $\ln b_T(\mathbf{z}_T) = 0$ for ease of notation.
4 To start the process at $t = 1$, $N \cdot P$ particles are drawn i. i. d.

Equation (7.30) actually estimates a lower bound to the ELBO. The reason is that the unbiased estimator of the normalizing constant, eq. (7.25), is the argument of a logarithm so that Jensen's inequality kicks in. This bias disappears asymptotically with decreasing variance of the inner estimator increases as $N \to \infty$.

It should be noted that resampling requires categorical sampling for which we cannot easily use reparametrization for computing parameter gradients. Stopping the gradients through these decisions has been shown to bias the gradient estimates. In their analysis of FIVO, Maddison et al. (2017) verify that the bias drastically reduces the variance compared to corrected unbiased gradients. We follow this line of argument and trade off a small gradient bias for reduced gradient variance.

7.3 A LINEAR GAUSSIAN EXAMPLE

To put ANS and the proposed estimator to a test, we conduct an experiment on a two-dimensional LGS. Details on the system and data used are gathered in appendix C.3. Using an LGS has several advantages:

1. Since we know the ground-truth system, we can plug it into the learning algorithm and isolate the (approximate) backward filter and proposal in our evaluation.

2. Many quantities of interest can be computed in closed form for LGSs. This includes the true backward filter of an LGS, cf. appendix C.4. The log marginal likelihood $\ln p(\mathbf{x}_{1:T})$ is also tractable, which allows us to evaluate the approximation gap.

Learning approximate backward filter and proposal with the estimator in eq. (7.30) in this controlled scenario creates an undesired result: consistently negative approximation gaps. That is, the ELBO estimates are *higher* than the known log marginal likelihood when they should be lower bounds. This holds true even on held-out data, and since the generative model is set to the ground-truth model overfitting can be ruled out as a reason.

We thus turn our attention to the estimator in eq. (7.30). It is important to recall that this is an IS estimator. For IS estimators to work, it is crucial that the proposal is designed to cover the tails of the target distribution. Ionides (2008) provides a good overview with examples.

When learning the proposal along with the model and approximate backward filter with eq. (7.30), such advice is ignored. Instead, the proposal is learned to *maximize* the estimator. The crux is that maximizing an estimator is not necessarily equivalent to maximizing the quantity to be estimated, the ELBO. It is thus reasonable to assume that the proposal learns to exploit this discrepancy and delivers what is asked: a *maximized* estimator rather than an *accurate* estimator.

We put this hypothesis to a test: recall the *filtering variational objective* (FIVO) from our discussion in section 4.3.6. FIVO is also a particle-based IS estimate of a lower bound to the log marginal likelihood $\ln p(\mathbf{x}_{1:T})$. In contrast to our proposed method, the proposal is the only learnable component when fixing the generative model to the ground truth LGS, i. e., we can isolate the proposal for study. Indeed, we find that the FIVO estimator suffers from the same phenomenon of systematically overestimating the true log marginal likelihood even though it should be a lower bound.[5]

7.3.1 Learning the Proposal

This motivates obtaining a proposal by other means. Specifically, we use *neural adaptive sequential Monte Carlo* (NASMC; Gu et al., 2015). The idea is to learn a proposal by maximizing

$$\mathrm{KL}(q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) \,\|\, \pi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})). \tag{7.32}$$

Compared to VI, this KL is reversed, the proposal is the second argument. This is to encourage *mode-covering* behavior, which is beneficial to the quality of the IS estimate. It is further justified by the observation that the number of samples required should be at the minimum as large as the *exponentiated* KL in eq. (7.32) to make reasonable estimates likely (Chatterjee and Diaconis, 2015). In other words, a proposal optimized with the NASMC objective will learn to achieve more accurate estimates with fewer particles.

This mode-covering KL is hard to evaluate especially since the target distribution is not available, but the key insight by Gu et al. (2015) is that evaluation is not necessary. Stochastic gradients w. r. t. proposal parameters φ can be computed from the particles obtained in the ELBO estimation, in our case

$$\begin{aligned}
&-\nabla_\varphi \,\mathrm{KL}\big(q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) \,\big\|\, \pi_\varphi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})\big) && (7.33)\\
&= \sum_t \mathbb{E}_{q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})}[\nabla_\varphi \ln \pi_\varphi(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{t:T})] && (7.34)\\
&\approx \sum_t \sum_p w_t^{(p)} \nabla_\varphi \ln \pi_\varphi\Big(\mathbf{z}_t^{(p)} \,\Big|\, \mathbf{z}_{t-1}^{(p)}, \mathbf{x}_{t:T}\Big). && (7.35)
\end{aligned}$$

Concretely, we update generative and inference model parameters θ and ϕ with gradient steps in the ELBO estimate, eq. (7.30), as before.

5 In our preliminary experiments, this only holds true when the proposal of FIVO uses future observations via a bidirectional RNN. With the additional information the proposal exploits the discrepancy between estimator and target quantity. As a side note, this is an empirical counterexample to the claim that a smoothing proposal does not provide benefit in terms of maximizing the estimator (Maddison et al., 2017). The fact that this had not been found may be related to our results of chapter 6.

Proposal parameters φ are updated according to eq. (7.35).[6] This adds a substantial consideration to the learning algorithm: how to balance training of the proposal vs. all other learnable components, in this case the approximate backward filter but later also the generative model. The challenge here is that the target distribution for the proposal, the approximate posterior, is learnable and thus a moving target and the proposal needs to catch up. At the same time, gradient steps in the *proposal* parameters do not benefit the model, which adds to the computational cost of learning. This leads to a trade-off of learning stability and learning time. Such a trade-off is particularly difficult because it is hard to detect the quality of the proposal or the estimator on the fly, in particular in scenarios where no ground truth log marginal likelihoods exist to verify the plausibility of estimated ELBO values.

7.3.2 Results

In our LGS example, we employ a rather pragmatic learning algorithm, with one update step in the proposal parameters for each update step in the model parameters. This schedule proves to remove the spurious ELBO over-estimation, and the approximation gap vanishes. As the approximate backward filter implementation is able to represent the true backward filter in this controlled scenario, this result is to be expected.

Figure 7.1 qualitatively contrasts the proposal and its target—the approximate posterior—with the ground truth posterior. We see that ANS is able to approximate the ground truth well. In addition, the prior and the ground truth filter obtained by Kalman filtering, section 3.3.2.1, are depicted. We see that these differ significantly, showing that the addition of the backward filter is beneficial for inference, and that the LGS was chosen to be sufficiently volatile for future observations to provide benefit.

7.4 FURTHER EXPERIMENTS AND DISCUSSION

7.4.1 Outlook: MNIST

We also conducted experiments on nonlinear data, in this case the row-wise MNIST data familiar from section 6.3.3. While we do not have the benefit of closed-form log marginal likelihoods as with LGSs,

6 In practice with auto-differentiation frameworks, one needs to ensure that the proposal is not optimized w. r. t. either weights or particles as these depend on the proposal parameters φ. This is achieved, e. g., by stopping gradients *through both* in eq. (7.35) or any comparable way of achieving the gradient of the proposal density function directly. Further, in this setting proposal and model components must not share parameters.

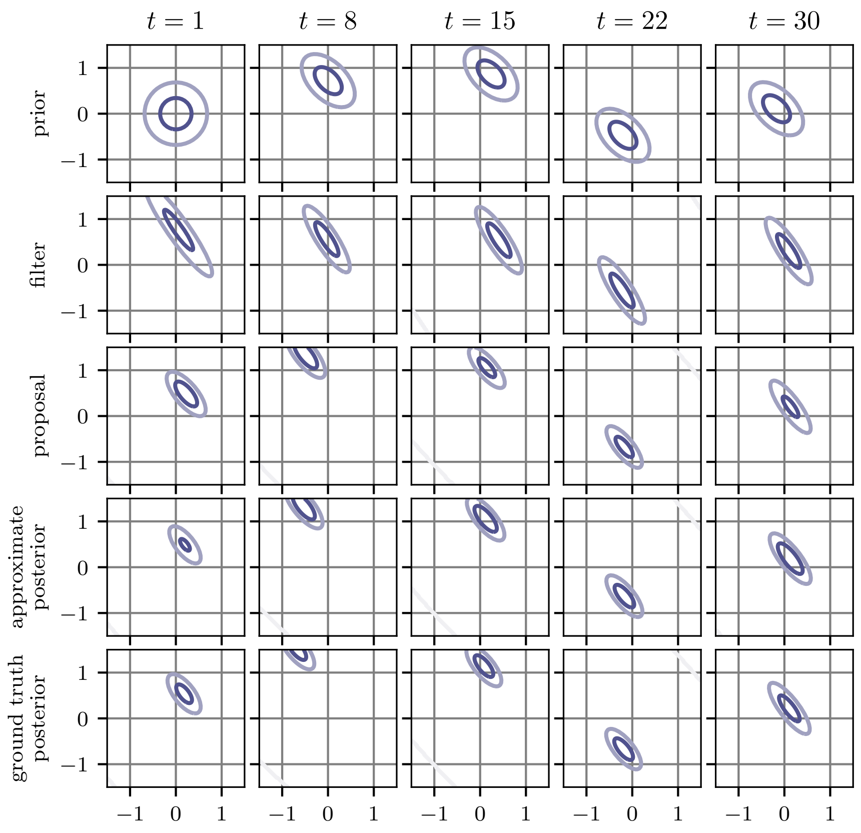


Figure 7.1: Qualitative comparison of various distributions in latent space for one sequence at the respective time steps from left to right. *Top to bottom:* the prior distribution; the ground truth Bayesian filter computed with the Kalman filtering algorithm; the proposal distribution $\pi(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{1:T})$; the unnormalized approximate posterior distribution $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{t:T})$; the ground truth smoothing posterior $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{t:T})$.

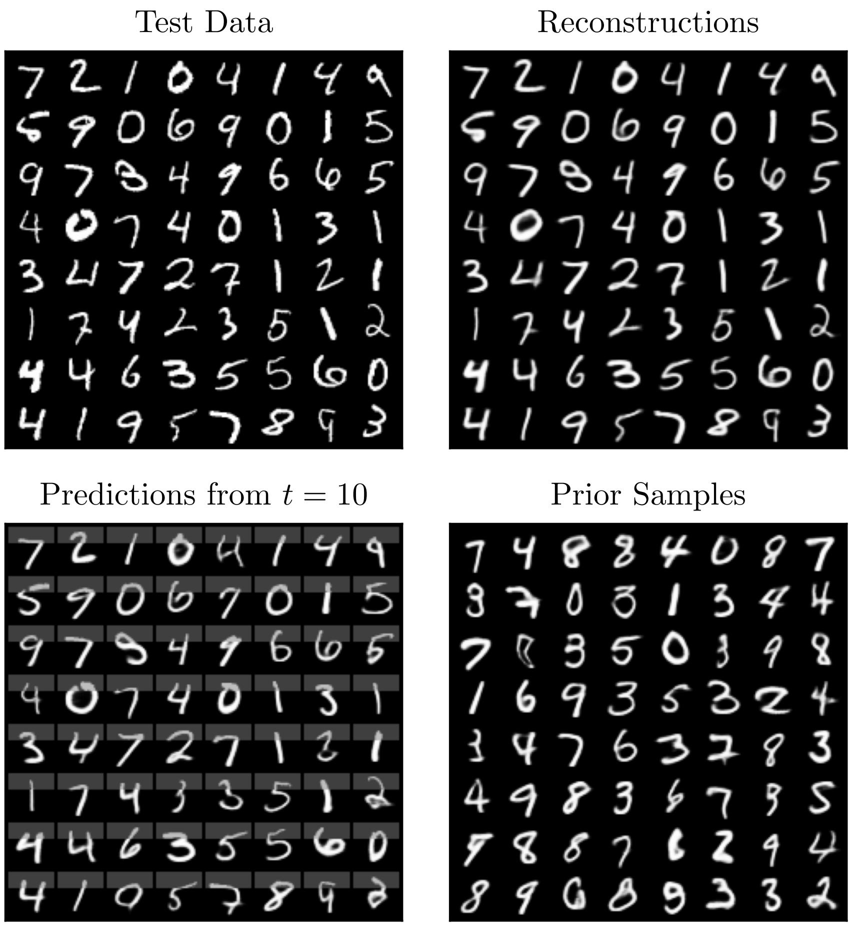


Figure 7.2: Qualitative results on row-wise MNIST data. The top row shows a batch of test data and the respective mean reconstructions from the posterior belief. The bottom row examines the prior. On the left are predictions with the prior, starting from the approximate posterior belief after $t = 10$ rows. The beliefs are those of the test batch in the top row. To distinguish ground truth from prediction, the ground truth rows are rescaled towards gray values. On the right, we show purely generative samples.

MNIST is so well-established as a benchmark that we at least know *reasonable* ELBO values.

We experience the same phenomenon of overestimating the ELBO, though not consistently. This is a challenge for quantitative evaluation since ELBO estimates are not trustworthy and hence not useful for model selection and evaluation.

Motivated by the positive impact for LGS data, we explored whether NASMC would lead to similar stabilization. The results were mixed. While NASMC drastically reduced the number of experiments with clearly implausible ELBO estimates, it also dramatically increased training time. In fact, none of the experiments with NASMC achieved comparable results to those of section 6.3.3.

We speculate that there are two main reasons for this.

The first is that striking a balance between learning the model via ELBO and the proposal via NASMC is more delicate in this case. Too few updates of the proposal parameters negate the effect of using NASMC; too many render learning prohibitively expensive. It appears that a more sophisticated schedule for balancing the two objectives would be necessary.

The second insight is derived from experiments that do not use NASMC. In this case, we can share weights between the backward RNN of the proposal and the backward filter. This appears to be beneficial to learning.

For these reasons, a quantitative analysis is not possible at this point. Nonetheless, a qualitative analysis of a successful run produces interesting results summarized in figs. 7.2 and 7.3. This run did *not* use NASMC; its final ELBO estimate was on a par with the results from section 6.3.3—keeping in mind that this should be interpreted with care. Details on the experiment can be found in appendix C.5.

The model exhibits a few desirable properties. Figure 7.2 shows that reconstructions are impeccable in comparison to data while prior samples are diverse and plausible at first glance. While the algorithm eludes deeper quantitative analysis for now, this serves as a promising proof of concept.

The remaining plot of fig. 7.2 shows the interplay between inference and generative model. The inference model is fed the test data and infers a belief up until time step $t = 10$, the tenth row, indicated by the gray background. From there on, the remaining rows are produced by sampling from the belief and subsequently exclusively from the prior. Again, we see that the resulting digits are plausible.

Contrasting the predictions with the true continuations of the data, we observe significant variability. This indicates that the learned latent system is dynamic and stochastic. The belief and the subsequent steps in the prior have not degenerated into a virtually deterministic system. Keep in mind that the belief at $t = 10$ was indirectly informed by the entire digit by means of the backward filter. This is an encouraging

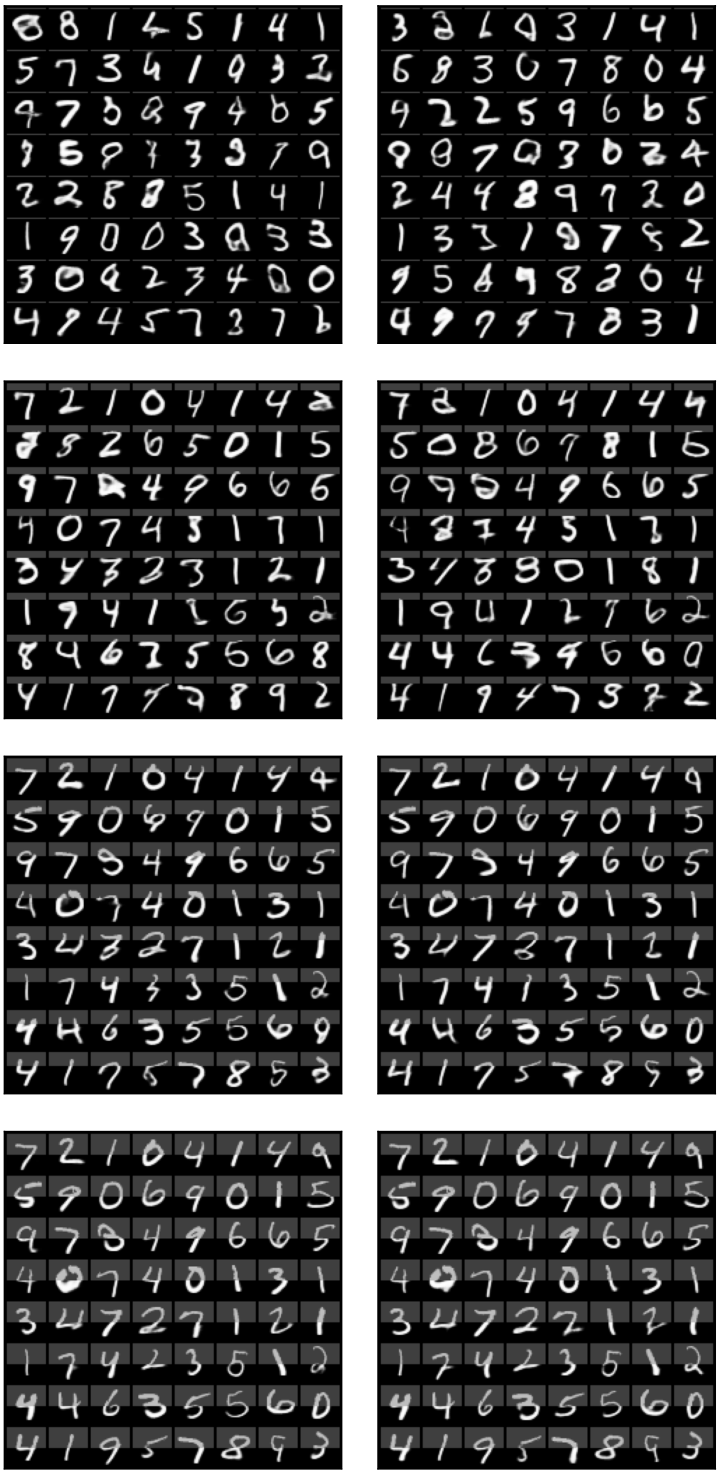


Figure 7.3: This plot reiterates the prediction plot in the bottom left of fig. 7.2, starting the prediction with the prior from different prefixes of length t. From top to bottom $t = 1, 5, 10, 15$. The columns are two independent predictions based on the same belief to examine sample diversity. The growing prefix is highlighted by rescaling their pixel grayscale values from $[0, 1]$ to $[0.25, 0.75]$. The test batch is the same as in fig. 7.2.

result, indicating that the learning algorithm is able to pick up inherent stochasticity in the system and avoids over-emphasizing the inference model.

Figure 7.3 explores this notion further. Starting from the same test batch, we investigate different prefix lengths before predicting with the prior and show two predictive samples per prefix length. Contrasting the samples from short prefixes with data, we can see that the posterior does not memorize the digit by letting the backward filter determine the perfect sample. With increasing prefix length, the variability between the predictive samples decreases, but never disappears. We also notice that the samples show a very consistent style, e. g., the stroke width within a sample remains plausible. This indicates that such global properties are propagated in the state, while local variations within the same style are driven by transition noise.

While the analysis remains qualitative at this point, these results certainly warrant further research into improving and stabilizing the learning of proposals.

7.4.2 Relation to Importance-Weighted Auto-Encoders

Our analysis of learned proposals is sufficient to motivate the use of methods like NASMC to learn the proposal. From a higher point of view, however, it does not provide insight into what exactly causes the overestimation.

We speculate further analysis would be interesting to *importance-weighted auto-encoders* (IWAEs), cf. section 2.5.3. IWAEs are similar to our proposed method in the sense of providing an IS approach to learning VAEs. In fact, Maddison et al. (2017) claim that FIVO without resampling is the same as IWAE, except for a sequential model.

But there are also differences to our approach. Where our model has potentially three moving parts—generative, proposal, and inference model—FIVO and IWAEs have only two, generative and proposal model. Our ELBO estimate in eq. (7.30) has the overall structure

$$\mathbb{E}_{\pi(\mathbf{z})}\left[\frac{q(\mathbf{z})}{\pi(\mathbf{z})}\ln\frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right] \tag{7.36}$$

whereas the IWAE bound has the structure

$$\mathbb{E}_{\pi(\mathbf{z})}\left[\ln\frac{p(\mathbf{x},\mathbf{z})}{\pi(\mathbf{z})}\right]. \tag{7.37}$$

That is, with IWAEs, there is an application of Jensen's inequality *after* importance sampling. A better understanding of these similarities and differences may hold the key to improved learning algorithms in the presence of proposal distributions.

The theory of IWAE bounds has been studied to some extent. Rainforth et al. (2018) found that, counterintuitively, increasing numbers of particles may hinder proposal learning. The reason is that a higher number of particles lowers the influence of the proposal: with a higher number of particles, a wider range of proposals lead to good results, leading in return to lower gradient magnitudes. Rainforth et al. (2018) show that the signal-to-noise ratio of the gradient estimate indeed *decreases* in the number of particles P.

It is unclear to what extent our results translate to the IWAE setting. We believe this to be worth of further study since our results show that estimated bound values are potentially misleading for evaluation and model selection. A deeper analysis is left to future research.

7.4.3 Discussion

ANS merges many of the previous threads of this thesis.

Based on the findings of chapter 6, it eliminates the conditioning gap by using a fully-conditioned inference algorithm for learning a generative model.

Even more, this inference model is derived by analyzing the Bayesian smoothing posterior of a state-space model. This is to maximally exploit inductive biases of proper algorithms and without relying on specific neural architectures or even neural networks in the first place. Only when quantities become generally intractable are they approximated with density networks, in this case the backward filter.

In doing so, ANS picks up the design principles of DVBFs and even takes them a crucial step further. Where DVBFs would only reuse the prior transition, ANS makes use of the entire generative SSM, including the emission model.

As we have seen in the preliminary experiments of this chapter, there is light and shadow. The experiments serve as a proof of concept. The approximate backward filter is learned well enough so that the posterior forward transition is approximated well. While the nonlinear results are not yet conclusive, the probed model shows promising properties.

At the same time, we observed challenges with the proposal distribution. NASMC shows promise as a remedy, but the nonlinear experiments also show that it is not straightforwardly integrated into a learning algorithm. Managing two objectives simultaneously ups the complexity of the overall learning algorithm significantly. This challenge needs to be addressed before ANS can be considered as a general-purpose algorithm.

To this end, further research beyond the scope of this thesis is required.

Future Work

To conclude this chapter, we can identify and speculate about possible paths forward which can already be identified at this stage.

One possible path is to better understand what causes instabilities of the estimator. As discussed in section 7.4.2, this may be of interest even beyond ANS. It could be interesting to probe to what extent proposal and backward filter may still share weights and what objective should be used for these shared weights.

Another possible route is to establish a learning algorithm that balances the two objectives more gracefully. A promising approach was recently suggested for VAEs by Rezende and Viola (2018) and Klushyn et al. (2019). They established a constrained optimization framework in which the ELBO is reinterpreted as the Lagrangian of a constrained optimization program. This leads to a learning algorithm that dynamically adjusts the relative importance of the likelihood and prior KL terms of the ELBO. This framework could be extended to include the proposal so that the resulting learning algorithm can adjust the relative importance of learning the model vs. learning the proposal. As with the previous suggestion, such approaches are possibly of interest beyond ANS. A principled, yet practical way of imposing constraints to sequence models may be beneficial beyond the challenge of learning proposals.

Part IV

SET-VALUED NEURAL FUNCTIONS

This part is based on ideas that have previously appeared in the following publication:

Soelch, Maximilian, Adnan Akhundov, Patrick van der Smagt, and Justin Bayer (2019). "On Deep Set Learning and the Choice of Aggregations." In: *Artificial Neural Networks and Machine Learning - ICANN 2019: Theoretical Neural Computation - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part I*. Ed. by Igor V. Tetko, Vera Kurková, Pavel Karpov, and Fabian J. Theis. Vol. 11727. Lecture Notes in Computer Science. Springer, pp. 444–457. ISBN: 978-3-030-30487-4. DOI: 10.1007/978-3-030-30487-4\\_35.

Direct quotes from this publication are highlighted in gray font color. Minor adaptations of the quotes to the style of this thesis are not explicitly highlighted.

Supplementary material is collected in appendix D.

8 AGGREGATION FUNCTIONS IN DEEP SET LEARNING

8.1 A MOTIVATING EXAMPLE

This last part shifts focus towards functions with set-valued inputs—set functions. This is nominally a departure from sequential LVMs, the central topic of this thesis, but there are connections on multiple levels.

To illustrate this, consider an agent with a belief in its own (scalar) position z along a line. The agent might need to take different actions if a certain position has been reached, e. g., perform a task in a specific place or avoid the danger of falling off a cliff.

As we have learned, e. g., in chapters 3 and 7, such beliefs can be represented by a set of particles. Then any policy acting on this belief must be a function of this belief, i. e., a set function. For the sake of the argument, we consider a simple task: answering the question whether the current position is more likely to exceed some threshold τ or not; that is which of the mutually exclusive events $z > \tau$ and $z \leqslant \tau$ is more probable.

The particle-based belief does not immediately answer this question, but it can inform the decision. If the belief is assumed to be Gaussian, then the particle mean—a function of the set of particles—is a good feature. Since a Gaussian is symmetric around its mean, its relative position to the threshold is a strong indicator to answer the question.

With a non-Gaussian belief, this feature becomes more error-prone. A more sophisticated feature derived from the particle representation of the belief is a committee: each particle submits an educated guess on whether the threshold was exceeded (e. g., by determining whether according to this particle the threshold is exceeded). The overall prediction is then determined by a majority vote among the particles.

Even for this feature, it is easy to imagine systematic failure cases. The most suitable features are tailored towards the beliefs that are likely to occur. This motivates replacing such hand-made features with learnable set functions which leverage learnable feature extraction.

A simple example underlines this.[1] We assume the true beliefs are scalar mixtures of two Gaussians, with one component mean above threshold $\tau = 0$ and one below. We represent each belief by a meager five samples. Figure 8.1 shows some beliefs and the respective samples.

1 The underlying neural architecture will be explained in the following sections. The full experimental details are collected in appendix D.1.

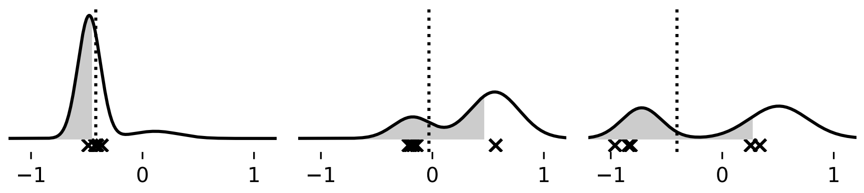


Figure 8.1: Three examples of mixtures of Gaussians for the motivating example. The graph shows the pdf. The shaded area indicates 50 % of the probability mass, i. e., the task is to predict whether the shaded area stretches across 0. Particles are depicted as crosses, the dotted line is their mean. Examples were picked to show failure cases of the hand-crafted features in the two right plots.

Table 8.1: Test set accuracies for sign prediction of the median of a mixture of two Gaussians, section 8.1.

FEATURE	VARIANT	ACCURACY
hand-crafted	particle mean	0.8150
	committee	0.8292
neural	mean aggregation	0.8348
	max aggregation	0.8328
ground truth	mixture weight	0.9124

We now compare the mean predictor, the committee predictor, and a learned neural predictor for whether $z > 0$ is more likely or not. The results are reported in table 8.1. Even in this rather simple setting, the neural predictor can outperform manual features out of the box—no hyper-parameters needed to be tuned. For completeness, we show another predictor that decides based on the true weight of the belief components. It outperforms all other predictors, but it also leverages information that is not contained in the particle-based belief and is difficult to estimate from only five particles but emphasizes that learned set functions are not strictly superior.

This simple example establishes a direct link to the previous chapters. From a higher level, there are further connections.

Firstly, with both sequential models and set functions, the data have structure that needs to be adhered. With sequential models, the temporal order is key. Analogously, with set functions it is important to respect that sets are unordered—a notion that we will formalize in section 8.2.2. In both cases, the additional structure in data requires distinct treatment in the models to tackle them.

The remedy shows further high-level connections to the previous chapters. The neural architectures are informed by algorithms. The same principle has been applied to set functions. The *Deep Set* neural architecture (Zaheer et al., 2017) is entirely motivated by a provable mathematical property of set functions as presented in section 8.2.3 and theorem 8.1.

NOTATION AND TERMINOLOGY To ease the discussion of sets of sets, minor adjustments in notation and terminology are required to avoid ambiguity. The elements $\mathcal{Z} \in \mathcal{D}$ of a data set—themselves sets—are called *populations*. The elements $\mathbf{z}_i \in \mathfrak{Z}$ of a population $\mathcal{Z} \subset \mathfrak{Z}$ are called *particles* from the *particle space* $\mathfrak{Z}$, alluding to the connection to particle-based algorithms discussed earlier. Each population further has a tensor-valued[2] representation $\mathbf{Z} \equiv \mathcal{Z}$, where $\mathbf{Z} \in \mathbb{R}^{p \times d}$ is created by concatenating $p = |\mathcal{Z}|$ particles of dimension d in a new tensor axis. Implicitly, this representation assumes that $\mathbf{z} \in \mathfrak{Z} \subset \mathbb{R}^d$. Such a representation is only unique up to the order of particles in the tensor. Equivalent representations are denoted by $\mathbf{Z}_\pi$, where π denotes the permutation along the concatenation axis that transforms $\mathbf{Z}$ into $\mathbf{Z}_\pi$. That is, generally $\mathbf{Z} \neq \mathbf{Z}_\pi$ but $\mathbf{Z} \equiv \mathcal{Z} \equiv \mathbf{Z}_\pi$. Populations could technically be *multi-sets*—sets that may hold duplicate entries. All presented results are valid either way. For clarity, we omit further discussion of multi-sets.

8.2 STATE OF THE ART

8.2.1 Order Matters

Sets that have been transformed into a tensor representation can be *processed* by a variety of neural networks. The problem is that the resulting mapping is in most cases not a proper set function, usually for one of two reasons.

Firstly, the result typically changes with a permutation of representation from $\mathbf{Z}$ to $\mathbf{Z}_\pi$. This can be mitigated by regularization methods such as shuffling the tensor randomly at training time but does not guarantee identical results for all permutations.

Secondly, many feed-forward NNs are unable to handle sets of variable size. This is not true for RNNs, where the set can be interpreted as an input sequence of arbitrary length. However, in the context of sequence-to-sequence problems it has been shown that the result is sensitive to the order of particles (Vinyals et al., 2016). The challenge is that most NNs are explicitly *not* designed to ignore the ordering.

The suggested solution by Vinyals et al. (2016) is an architecture that decouples the set represented as tensor $\mathbf{M}$—the memory—from the inputs to the neural component. Rather, from a high-level point of view the RNN produces a query $\mathbf{q}_t$ to the memory per step; the memory responds with an activation a_t that is aggregated from the

2 As has become custom in the community, we use the term *tensor* loosely as a synonym for multi-dimensional arrays, not the generalization of linear transforms.

memory, neglecting the order; the aggregation is then fed back to produce the next query:

$$\mathbf{q}_t = \text{LSTM}(\mathbf{q}_{t-1}, \mathbf{a}_{t-1}) \tag{8.1}$$

$$\hat{w}_{i,t} = \text{attention}(\mathbf{m}_i, \mathbf{q}_t) \qquad \left(= \mathbf{m}_i^\top \mathbf{q}_t\right) \tag{8.2}$$

$$\mathbf{w}_t = \text{softmax}(\hat{\mathbf{w}}_t) \tag{8.3}$$

$$\mathbf{a}_t = \sum w_{i,t} \mathbf{m}_i \tag{8.4}$$

$$\mathbf{a} = \mathbf{a}_T. \tag{8.5}$$

This architecture is called *read-process-write*. It is a proper set function that can handle sets of arbitrary size by design. Yet, it is unclear what kind of functions can be expressed by this architecture.

To the best of our knowledge, this model has only been discussed in its sequence-to-sequence context. We will revisit and refine this architecture in section 8.3.3.

8.2.2 Invariance and Equivariance

We have defined how sets $\mathcal{Z}$ can be translated into tensor representations $\mathbf{Z}$. This brings us closer to the more familiar tensor-to-tensor scheme, but we also found that a naive application of MLPs does not yield proper set functions. The key issue is the non-unique translation of sets into tensors. A proper set function needs to be invariant to any particular translation. This concept of invariance is formalized by

Definition 8.1 (Invariance)**.** *A function* $f \colon \mathcal{P}(\mathfrak{Z}) \to \mathcal{Y}$ *is* order-invariant *if for any permutation* π *and input* $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\} \in \mathcal{P}(\mathfrak{Z})$

$$f(\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}) = f\left(\left\{\mathbf{z}_{\pi(1)}, \ldots, \mathbf{z}_{\pi(N)}\right\}\right).$$

If it is clear from the context, we will call such functions invariant*. When the input is represented as a tensor, a function is invariant if for all tensor representations* $\mathbf{Z}$ *and permutations* π*, this definition can also be expressed as*

$$f(\mathbf{Z}) = f(\mathbf{Z}_\pi).$$

The attentive reader may have spotted that the first definition is somewhat circular: the argument of right- and left-hand side are identical since both are sets of the same elements which are not affected by the permutation. If defined on sets, f is automatically invariant. Some ordering of the particles is required to even define functions that are *not* invariant, e. g., a function that returns the *first* particle. The key notion of definition 8.1 is thus the invariance to arbitrary ordering of set elements from a particular representation on which f operates, e. g., from assigning indices, or in the tensor-valued representation. The latter motivates the equivalent definition of invariance in terms of $\mathbf{Z}$ and $\mathbf{Z}_\pi$.

The function $f\colon \mathcal{P}(\mathfrak{Z}) \to \mathcal{Y}$ in definition 8.1 assumes a range $\mathcal{Y}$ of fixed dimension, irrespective of the size of the input set. Alternatively, we might be interested in mapping a set to a set of outcomes of equal size, i. e., a function $g\colon \mathfrak{Z}^p \to \mathcal{Y}^p$. This might be the case if we want to simultaneously make a prediction for each particle, taking into account the entire set. In this case, a related concept to *invariance* emerges.

Definition 8.2 (Equivariance)**.** *A function* $g\colon \mathfrak{Z}^p \to \mathcal{Y}^p$ *is* equivariant *if input permutation results in equivalent output permutation, i. e., for any* $\mathbf{Z}$ *and* $\mathbf{Z}_\pi$

$$g(\mathbf{Z}_\pi) = (g(\mathbf{Z}))_\pi.$$

Equivariance guarantees that we can interpret the function g—which is defined in terms of tensors $\mathbf{Z} \in \mathfrak{Z}^p$—as a function on sets.

8.2.3 Deep Sets

We have seen that naive implementations of set functions with neural networks will not yield *invariant* functions. To benefit from the universal approximation capabilities of NNs for invariant set functions, a deeper understanding of invariance is required.

The seminal work by Zaheer et al. (2017) proved a defining structural property of order-invariant functions:

Theorem 8.1 (Deep Sets, Zaheer et al. (2017))**.** *A function* f *on populations* $\mathcal{Z}$ *from* countable *particle space* $\mathfrak{Z}$ *is invariant if and only if there exists a decomposition,*

$$f(\mathcal{Z}) = \rho\left(\sum_{\mathbf{z} \in \mathcal{Z}} \phi(\mathbf{z})\right),$$

with appropriate functions ϕ *and* ρ*. Following Wagstaff et al. (2019), we call such functions* sum-decomposable*.*

In other words, every invariant function is decomposable into a per-particle embedding ϕ, a sum operation that aggregates particles and thus ensures invariance, followed by a final processing of the invariant sum with ρ. The key advantage of this decomposition is that both ϕ and ρ are no longer set functions but operate on vector-valued input. As a consequence, it is much more straightforward to use off-the-shelf NNs in their place.

As Wagstaff et al. (2019) point out, the devil is in the details: the restriction to *countable* particle spaces in theorem 8.1 limits its practical use. They highlight severe pathologies for *uncountable* input spaces:

1. There exist invariant functions that have no sum decomposition.

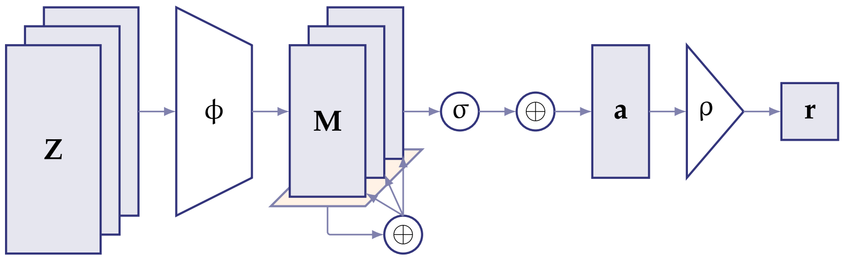


Figure 8.2: Deep Set architecture, eqs. (8.7) to (8.10), with a single equivariant layer, eq. (8.6). Aggregation functions are depicted by ⊕.
A similar figure has previously appeared in Soelch et al. (2019).

2. There exist sum decompositions that are everywhere-discontinuous.

3. Even common functions such as $\max(\mathcal{Z})$ cannot be *continuously* decomposed when the dimension of the image space of the embedding ϕ is smaller than the population size $p = |\mathcal{Z}|$.

As a consequence they refine theorem 8.1 to

Theorem 8.2 (Uncountable Particle Spaces, Wagstaff et al. (2019))**.** *A* continuous *function* f *on* finite *populations* $\mathcal{Z}$, $|\mathcal{Z}| \leqslant p$, *is invariant if and only if it is sum-decomposable via* $\mathbb{R}^p$.

That is, for arbitrary continuous f, the image space of ϕ has to have at least dimension p, which is both necessary and sufficient. More restrictive in scope than theorem 8.1, its assumptions are more applicable in practice where most function approximators—neural networks, Gaussian processes—are continuous.

A generic invariant neural architecture emerges from theorems 8.1 and 8.2 by using neural networks for ρ and ϕ, respectively. In practice, to allow for higher-level particle interaction during the embedding ϕ, *equivariant* neural layers (Zaheer et al., 2017) are introduced:

$$\text{equivariant}(\mathbf{Z}) = \sigma(\mathbf{Z} - \mathbf{1}\alpha(\mathbf{Z})), \tag{8.6}$$

where $\sigma(\cdot)$ denotes a per-particle feed-forward layer, and $\alpha(\cdot)$ denotes an aggregation. Aggregations—our object of study—induce invariance by mapping a population to a fixed-size description, typically, e. g., sum, mean, or max. The full architecture is

$$\mathbf{m}_i = \text{embed}(\mathbf{z}_i), \tag{8.7}$$
$$\mathbf{C} = \text{combine}(\mathbf{M}), \qquad \left(\mathbf{M} = \left[\mathbf{m}_i^\top\right]\right), \tag{8.8}$$
$$\mathbf{a} = \text{aggregate}(\mathbf{C}), \tag{8.9}$$
$$\mathbf{r} = \text{process}(\mathbf{a}), \tag{8.10}$$

with ϕ implemented by a per-particle embedding followed by an equivariant combination function consisting of equivariant layers. These two steps are equivariant by design. Summation is replaced by a generic aggregation operation. This aggregation induces invariance of the overall architecture. Qi et al. (2017a) and Zaheer et al. (2017) suggest the max operation as an alternative to summation. Lastly, ρ can be implemented by arbitrary functions since the aggregation in eq. (8.9) is already invariant. This framework is depicted in fig. 8.2.

8.2.4 Related Work

Beyond the highlighted connections to the previous chapters, set functions have a wide field of applications: depth vision with 3D point clouds, probability distributions represented by finite samples, or operations on unstructured sets of tags (Póczos et al., 2013; Reed et al., 2016; Wang et al., 2019).

This motivated research into order-invariant neural architectures (Guttenberg et al., 2016; Ravanbakhsh et al., 2016; Vinyals et al., 2016; Edwards and Storkey, 2017). From this, the Deep Set framework as outlined in the previous sections emerged, proving that many interesting invariant functions allow for a sum decomposition (Qi et al., 2017a; Zaheer et al., 2017; Wagstaff et al., 2019).

Several papers introduce and discuss a Deep Set framework for dealing with set-valued inputs (Qi et al., 2017a; Zaheer et al., 2017). A driving force behind research into order-invariant neural networks are point clouds (Qi et al., 2017a,b, 2018), where such architectures are used to perform classification and semantic segmentation of objects and scenes represented as point clouds in $\mathbb{R}^3$. It is further shown that a max decomposition allows for arbitrarily close approximation (Qi et al., 2017a).

Generative models of sets have been investigated: in an extension of VAEs, the inference of latent population statistics resembles a Deep Sets architecture (Edwards and Storkey, 2017). Generative models of point clouds are proposed by Achlioptas et al. (2018) and Yi et al. (2019).

Permutation-invariant neural networks have been used for predicting dynamics of interacting objects (Guttenberg et al., 2016). The authors propose to embed the individual object positions in pairs using a feed-forward neural network. Similar pairwise approaches have been investigated by X. Chen et al. (2014) and Chang et al. (2017) and applied to relational reasoning by Santoro et al. (2017).

Weighted averages based on attention have been proposed and applied to multi-instance learning (Ilse et al., 2018). Several works have focused on higher-order particle interaction, suggesting computationally efficient approximations of Janossy pooling (R. L. Murphy et al., 2019) or proposed set attention blocks as an alternative to equivariant layers (J. Lee et al., 2019).

8.3 THE CHOICE OF AGGREGATION

8.3.1 The Role of Aggregations in Deep Set Architectures

Section 8.2 has established the Deep Set architecture as proposed by Zaheer et al. (2017), along with minor extensions. The key ingredient that renders this type of architecture invariant is the summation between embedding and processing step. The architecture *inherits* its invariance from the sum function—the prototypical, downright trivial invariant function.

Yet, in practice we see it replaced by, e. g., the mean or max function. In comparison, summation has unfavorable properties. For instance, the former two have their result bounded when the embedding is bounded. The result of summation can instead grow arbitrarily large with growing population size. This may cause a downstream neural processing function ρ to saturate. Such practical considerations motivate a looser definition of the Deep Set architecture with a more general *aggregation* function at its core, cf. eqs. (8.7) to (8.10).

In this context, aggregation is a rather loose term. It is strictly speaking simply an invariant function, which turns the connection to theorems 8.1 and 8.2 on its head: invariant functions are invariant because they are invariant. For our purposes, an aggregation function is a function that is loosely speaking "obviously" invariant and thus lends itself to be deployed in a Deep Set architecture as the central, invariance-inducing component between embedding and processing.

We hypothesize that this rather imprecise nature led to a focus on either embedding or processing in the previous literature. In the following, we will study aggregations. After discussing desirable properties and extending the theory around aggregation functions, we will suggest multiple alternatives, including learnable recurrent aggregation functions. Studying them in several experimental settings, we will find that the choice of aggregation impacts not only the performance but also hyper-parameter sensitivity and robustness to varying population sizes. In the light of these findings, we will argue for new evaluation techniques for neural set functions.

8.3.2 Sum Isomorphism

We start with a simple but useful observation:

Corollary 8.1 (Sum Isomorphism)**.** *Theorems 8.1 and 8.2 can be extended to aggregations of the form $\alpha_g = g \circ \sum \circ g^{-1}$, i. e., summations in an isomorphic space.*

Proof. From $\rho \circ \sum \circ \phi = (\rho \circ g^{-1}) \circ g \circ \sum \circ g^{-1} \circ (g \circ \phi)$, sum decompositions can be constructed from α_g-decompositions and vice versa. □

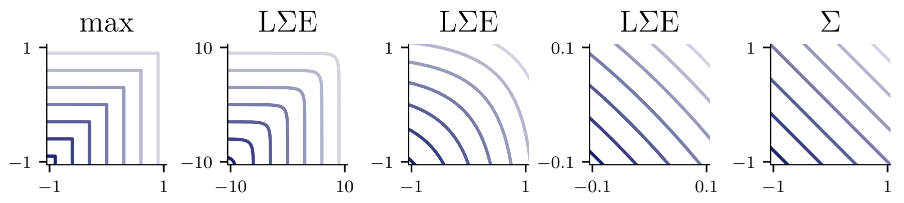


Figure 8.3: Contour plots on two inputs for max, logsumexp (LΣE) on three ranges, and sum (Σ). For large ranges, LΣE acts like max, shifting towards sum with decreasing input range.
A similar figure has previously appeared in Soelch et al. (2019).

Corollary 8.1 justifies, e. g., mean with

$$g((z_1, \dots, z_{n+1})) = (z_1, \dots, z_n)/z_{n+1}, \tag{8.11}$$

$$g^{-1}(\mathbf{z}) = (\mathbf{z}^\top, 1)^\top, \tag{8.12}$$

but also logsumexp (LΣE) with $g = \ln$. In that light, there is an interesting case to be made for LΣE: depending on the input magnitudes, LΣE can behave akin to max or like a linear function akin to summation, cf. fig. 8.3. Operating in log space, LΣE further exhibits *diminishing returns*: N identical scalar particles z_i yield

$$\mathrm{L\Sigma E}(\{z_i\}) = \ln(N) + z_1. \tag{8.13}$$

The larger N, the smaller the output change from additional particles. As discussed previously with sum vs. mean aggregation, this may be numerically favorable particularly with large sets. While we will not investigate this idea further, we point out that the concept of diminishing returns is also related to possibly desirable asymptotic statistical properties. With an increasing amount of particles, we might want to expect some notion of convergence in the result akin to asymptotic consistency.

8.3.3 Learnable Aggregation Functions

In the Deep Set architecture as suggested by Zaheer et al. (2017), cf. section 8.2.3, the aggregation is the only non-learnable component. While understandable in the light of its origin in theorems 8.1 and 8.2, there is no inherent reason why this should be the case. The key to a learnable aggregation is finding a function that is learnable and useful but simple enough to be integrated as an aggregation function.

Here, we recall the read-process-write architecture by Vinyals et al. (2016), cf. section 8.2.1. While it predates the Deep Set architecture, it has been overlooked[3] in the wake of the more universal architecture. Elaborating on this idea, we suggest *recurrent aggregations*:

3 For instance, Zaheer et al. (2017) only briefly mention the sequence-to-sequence and order aspect without acknowledging the suggested invariant architecture.

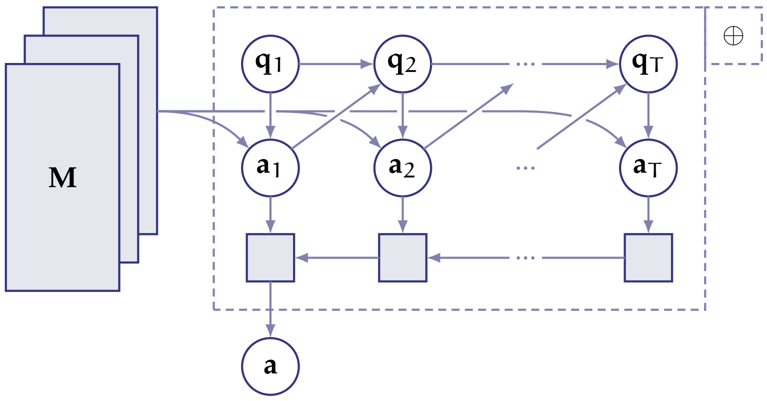


Figure 8.4: Recurrent aggregation function, eqs. (8.14) to (8.18). Queries to memory are produced in a forward pass, responses aggregated in a backward pass. This backward pass introduces short cuts for stable gradients.
A similar figure has previously appeared in Soelch et al. (2019).

Definition 8.3 (Recurrent and Query Aggregation)**.** *A* recurrent aggregation *is a function* $f(\mathcal{Z}) = \mathbf{a}$ *that can be written recursively as:*

$$\mathbf{q}_t = \text{query}(\mathbf{q}_{t-1}, \mathbf{a}_{t-1}) \tag{8.14}$$

$$\hat{w}_{i,t} = \text{attention}(\mathbf{m}_i, \mathbf{q}_t) \tag{8.15}$$

$$\mathbf{w}_t = \text{normalize}(\hat{\mathbf{w}}_t) \tag{8.16}$$

$$\mathbf{a}_t = \text{reduce}(\{w_{i,t}\mathbf{m}_i\}) \tag{8.17}$$

$$\mathbf{a} = g(\mathbf{a}_{1:T}), \tag{8.18}$$

where $\mathbf{m}_i = \phi(\mathbf{z}_i)$ *is an embedding of the input population* $\{\mathbf{z}_i\}$ *and* $\mathbf{q}_1$ *is a constant. We further call the special case* $T = 1$*, i. e., a single query* $\mathbf{q} \equiv \mathbf{q}_1$*, a* query aggregation*.*

As long as reduce is invariant and normalize is equivariant, recurrent and query aggregations are invariant. This architectural block is depicted in fig. 8.4.

Building upon eqs. (8.1) to (8.5), recurrent aggregations introduce two modifications: firstly, we replace a weighted sum by a general weighted aggregation—giving us a rich combinatorial toolbox on the basis of simple invariant functions.

Secondly, we add post-processing of the step-wise results $\mathbf{a}_{1:T}$. In practice, we use another recurrent network layer that processes $\mathbf{a}_{1:T}$ in reversed order. Without this modification, later queries tend to be more important as their result is not as easily forgotten by the forward recurrence. The reversed-order processing reverses this effect so that the first queries tend to be more important, and the overall architecture is more robust to common fallacies of recurrent architectures, in particular unstable gradients (Hochreiter, 1991).

Observing eq. (8.17), we note that our learnable aggregation functions wrap around the previously discussed simpler non-learnable aggregations. A major benefit is that the inputs are weighted—sum becomes weighted average, for instance. This also allows the model to effectively exploit nonlinearities as discussed with LΣE.

8.3.4 Further Remarks

DIVIDE-AND-CONQUER AGGREGATION A straightforward strategy to obtain "obviously" invariant aggregations leverages commutative and associative *binary* operations like addition and multiplication. Commutativity and associativity guarantee that we aggregate a population two particles at a time, and no matter the order, the result remains the same.

This principle can be generalized to divide-and-conquer operations, where the set is first divided up into its atoms and then aggregated piecemeal by conquering. This sort of aggregation is invariant if conquering is invariant to division. This rationale allows for a much wider class of aggregations. Examples are logical operators such as *any* or *all*, but also sorting. Sorting further generalizes max and min, and any percentile, e. g., median. We mention this interesting class of aggregations for completeness but will not discuss it further—it does not fit our neural framework, which requires aggregations to be differentiable.

UNIVERSAL APPROXIMATION The key promise of *universal* approximation (Kolmogorov, 1957; Hecht-Nielsen, 1988; Hornik et al., 1989) is that a family of approximators, e. g., neural nets or neural sum decompositions, is dense within a wider family of interesting functions. The universality granted by theorems 8.1 and 8.2, through constructive proofs, hinges on sum aggregation. Corollary 8.1 grants flexibility but does not apply to arbitrary aggregations, like max or the suggested learnable aggregations.[4] It remains open to what extent the sum can be replaced. It is worth noting that the embedding dimension constraint of theorem 8.2 is rarely met, trading theoretical guarantees for test-time performance.

INVARIANCE RECURSION Both the broadened definition of the Deep Set architecture and the recurrent aggregation are defined in terms of some other invariant aggregation function, cf. eqs. (8.9) and (8.17). This deviation from sum decompositions implies a *recursion of invariance*: the invariance of the overall architecture is relayed to invariance of a component of a component etc. For instance, a recurrent aggregation could be the aggregation function used inside a higher-level recurrent aggregation. Similarly, *equivariant* functions or layers require an arbitrary aggregation, which can include a recurrent aggregation or even an entire, nested Deep Set architecture.

4 Note that max allows for arbitrary approximation (Qi et al., 2017a).

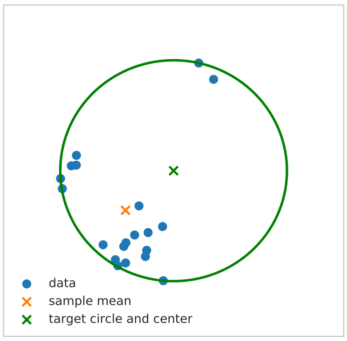


Figure 8.5: Minimal enclosing circle example population.
This figure has previously appeared in Soelch et al. (2019).

While we restrict ourselves to at most one level of recursion in this work, this perspective highlights the considerable amount of creative freedom the Deep Set architecture grants once it is detached from theorems 8.1 and 8.2 At the same time, it shows a need to gain a better understanding of aggregation functions such as inductive biases in practical settings, much like feed-forward neural nets are usually replaced with architectures targeted towards the task at hand.

8.4 EXPERIMENTS

We consider three simple aggregations: mean (or weighted sum), max, and LΣE. These are used in equivariant layers and final aggregations and may be be wrapped into a recurrent aggregation. This combinatorially large space of configurations is tested in four experiments described in the following sections.

8.4.1 Mininmal Enclosing Circle

In this supervised experiment, we are trying to predict the minimal enclosing circle of a population of size 20 from a GMM. A sample population with target circle is depicted in fig. 8.5. The sample mean does not approximate the center of the minimal enclosing circle well, and the correct solution is defined by at least three particles. The models are trained by minimizing the *mean squared error* (MSE) towards the center and radius of the true circle, computable in linear time (Welzl, 1991).

Results are given in table 8.2. Each row shows the best result out of 180 runs, 20 runs for each of the 9 combinations of aggregations. We can see that both recurrent equivariant layers and recurrent aggregations improve the performance, with equivariant layers granting the larger performance boost. The challenge lies mostly in a better approximation of the center.

Table 8.2: Minimal enclosing circle results.
A similar table has previously appeared in Soelch et al. (2019).

recurrent equiv./aggr.	mean squared errors best	radius	center	median best
✗ / ✗	0.71	0.06	0.66	1.57
✗ / ✓	1.02	0.14	0.88	1.30
✓ / ✗	0.54	0.08	0.47	0.87
✓ / ✓	0.42	0.09	0.33	0.58

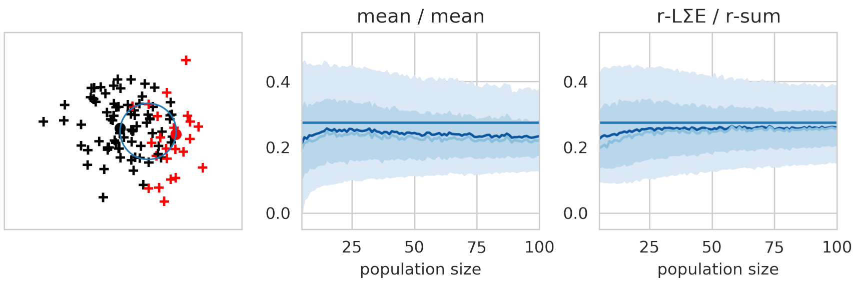


Figure 8.6: GMM mixture weights problem. *Left*: Example population. *Middle and Right*: Estimator development for increasing populations size for a non-learnable and a learnable model, with 50 % and 90 % empirical confidence intervals.
This figure has previously appeared in Soelch et al. (2019).

The top row indicates that an entirely non-recurrent model performs better than its counterpart with recurrent aggregation (second row). To test for a performance outlier, we compute a bootstrap estimate of the expected peak performance when only performing 20 experiments: we subsample all available experiments (with replacement) into several sets of 20 experiments, recording the best performance in each batch. The last column in table 8.2 reports the median of these best batch performances. The result shows increased robustness to hyper-parameters despite having more hyper-parameters.

8.4.2 Gaussian Mixture Weights

In this experiment, our goal is to estimate the mixture weights of a Gaussian mixture model directly from particles. The GMM populations of size 100 in our data set are sampled as follows: each mixture consists of two components; the mixture weights are sampled from $[.05, .95]$; the means span a diameter of the unit circle, their position is drawn uniformly at random; component variances are fixed to the same diagonal value such that the clusters are not linearly separable. An example population is shown in fig. 8.6. The model outputs

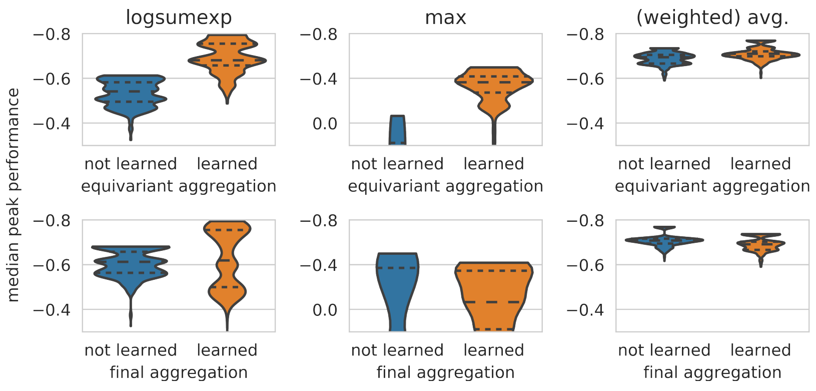


Figure 8.7: Robustness analysis. Metric is the score ratio of the true mixture weight under a neural model compared to expectation maximization (negative sign indicates EM is outperformed; the more negative, the better). Each violin shows the peak performance distribution for batches of five experiments. Top row: equivariant layer aggregations. Bottom row: final aggregations.
This figure has previously appeared in Soelch et al. (2019).

concentrations a and b of a Beta distribution. We train to maximize the log likelihood of the smaller ground truth weight under this Beta distribution. At training time, for every gradient step the batch population size N is chosen randomly, with $p(N = n) \propto n$. In fig. 8.6, we show how an estimator based on the learned model behaves with growing population size.

We were again interested in the robustness of the models. We compare to *expectation maximization* (EM)—the classic estimation technique for mixture weights—as a baseline by gathering 100 estimates each from EM and the model for each population size by subsampling (with replacement) the original population. Then we compare the likelihood of the true weight under a *kernel density estimate* (KDE) of these estimates. The final metric is the log ratio of the scores under the two KDEs. Then, as in the previous section, we compute the peak performance for batches of five experiments in order to see which configurations of models consistently perform well.

The results of this analysis are shown in fig. 8.7. The top row indicates that learnable equivariant layers lead to a significant performance boost across all reduction operations. Note that the y-axis is in log scale, indicating *multiples* of improvements over the EM baseline. We note that LΣE benefits most drastically from learnable inputs. Notably, the middle column, which depicts max-type aggregations, indicates that this type of aggregation significantly falls behind the alternatives. Note that we had to scale the y-axes to even show the violins and that a significant amount of *peak* performances perform *worse* than EM (indicated by sign flip of the metric).

Table 8.3: Test set accuracy on ModelNet40 classification. Sorted by accuracy for $|\mathcal{Z}| = 1000$, the population size at training time. The models show a high range of robustness to lowered population sizes.
This table has previously appeared in Soelch et al. (2019).

	Equivariant layer type & aggregation type									
	max	max	max	max	max	r-sum	max	r-max	r-LΣE	q-sum
$\lvert\mathcal{Z}\rvert$	max	r-LΣE	r-sum	q-max	q-sum	r-sum	r-max	r-max	r-LΣE	q-sum
1000	0.873	0.858	0.857	0.838	0.835	0.820	0.817	0.812	0.780	0.775
100	0.665	0.753	0.730	0.695	0.684	0.719	0.453	0.220	0.640	0.603
50	0.470	0.628	0.584	0.524	0.513	0.610	0.355	0.146	0.519	0.468

8.4.3 Point Clouds

The previous experiment extensively tested the effect of aggregations in controlled scenarios. To test the effect of aggregations on a more realistic data set, we tackle classification of point clouds derived from the ModelNet40 benchmark data set (Wu et al., 2015). The data set consists of CAD models describing the surfaces of objects from 40 classes. We sample point cloud populations uniformly from the surface. The training is performed on 1000 particles. For this experiment, we fixed all hyper-parameters—including optimizer parameters and learning rate schedules—as described by Zaheer et al. (2017), and only exchanged the aggregation functions in the equivariant layers and the final aggregation.

The results for the 10 best configurations are summarized in table 8.3. The original model (max/max column) performs best in the training scenario ($|\mathcal{Z}| = 1000$, first row)—as expected on hyper-parameters that were optimized for the model. Otherwise, learnable final aggregations outperform all non-learnable aggregations. We further observe that max-type aggregations in equivariant layers seem crucial for good final performance. This contrasts the findings from section 8.4.2. We believe this to be a result of either (i) the hyper-parameters being optimized for max-type equivariant layers or (ii) the classification task (as opposed to a regression task), favoring max-normalized embeddings that amplify discriminative features.

The second and third row highlight an insufficiently investigated problem with invariant neural architectures: the top-performing model overfits to the training population size. Despite sharing all hyper-parameters except the aggregations, the test scenarios with fewer particles show that learnable aggregation functions generalize favorably. Compare the first two columns: both drops for the original model are comparable to the total drop for the learnable model.

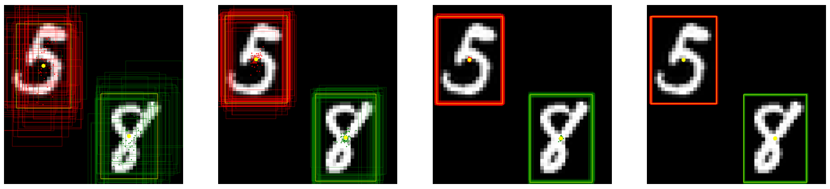


Figure 8.8: Spatial attention example. Each pane shows multiple test time bounding box samples for 5, 20, 200, 1000 particles.
This figure has previously appeared in Soelch et al. (2019).

8.4.4 Spatial Attention

In the previous experiments, we investigated models trained in isolation on supervised tasks. Here, we will test the performance as a building block of a larger model, trained end-to-end and unsupervised. The data consist of canvases containing multiple MNIST digits, cf. fig. 8.8. These data are known from AIR and VTSSI in chapter 5. We plug an invariant model as the localization module, which repeatedly attends to the input image, at each step returning the bounding box of an object. To turn a canvas into a population, we interpret the gray-scale image as a two-dimensional density and create populations by sampling 200 particles proportional to the pixel intensities. As we discussed in chapter 5, AIR is a major bottleneck for VTSSI. Remarkably, the set-based approach requires an order of magnitude fewer weights and consequently has a significantly lower memory footprint compared to the original model, which repeatedly processes the entire image.

The task is challenging in several ways: the loss is a lower bound to the likelihood of the input canvas, devoid of localization information. The intended localization behavior needs to emerge from interaction with downstream components of the overall model. As with enclosing circles, the bounding box center is correlated with the sample mean of isolated particles from one digit. However, depending on the digit, this can be inaccurate.

As fig. 8.9 indicates, the order-invariant architecture on 200 particles (as in training, vertical line) can serve as a drop-in replacement, performing on a par or slightly improved compared to the original model baseline, indicated by the vertical line. This is remarkable, with the original model being notoriously hard to train (Kosiorek et al., 2018).

We investigate the performance of the model when the population size varies. We observe that the effect on performance varies with different aggregation functions. Learnable aggregation functions exhibit strictly monotonic performance improvements. This is reflected by tightening bounding boxes for increasing population sizes, fig. 8.8. Similar behavior cannot be found reliably for non-learnable aggregations. Note that we can trade off performance and inference speed *at*

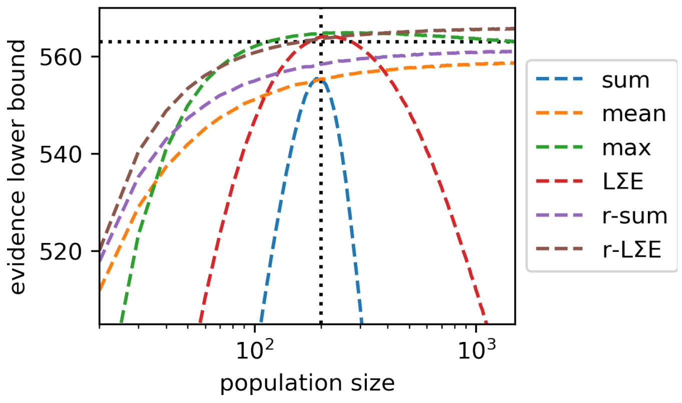


Figure 8.9: Test-time evidence lower bound values against various population sizes. Dashed vertical line: training population size. Dashed horizontal line: best baseline model.
This figure has previously appeared in Soelch et al. (2019).

test time by varying the population size, depending on the needs of the application.

Lastly, we note that in both this and the point cloud experiment, section 8.4.3, learnable LΣE-aggregations performed well. We attribute this to the properties of diminishing returns and sum-max-interpolation amplified by weighted inputs, cf. section 8.4.

8.5 DISCUSSION

Our analysis of Deep Set architectures reveals that aggregation functions play a more crucial role than previously acknowledged. Firstly, we have shown that the aggregation function of a Deep Set architecture is also a learnable component, like embedding and processing. Secondly, we provided theoretical and empirical results that can guide the choice of aggregation in future applications. Depending on the task at hand—classification vs. regression—different aggregation functions may be warranted. And even when the peak performance is equal for different aggregation functions, we found that they can exhibit very different secondary properties. Aggregations wildly differ in sensitivity to hyper-parameters, a concern that is not reflected in peak results. Some aggregations, particularly learnable aggregations, scale more gracefully with varying population sizes at test time.

Tying this coda chapter back in with the remainder of this thesis, we found that deep set architectures can be integrated well as density networks with LVMs and trained in unsupervised fashion and end-to-end as part of the pipeline of AIR. To the best of our knowledge, this had not been shown before. In the light of our motivating example and the particle-based approach presented in chapter 7, further research into combining Deep Set architectures with sequential LVMs, e. g., as policies, is warranted.

CONCLUSION

This thesis shows that the variational auto-encoder framework can be adapted to sequential latent-variable models in order to efficiently uncover dynamics via unsupervised learning from data. With DVBFs, VTSSI, and ANS, we have successfully showcased an evolution of algorithms to learn state-space models. Their inference models tackle even complex semantic tasks—such as tracking with VTSSI—as a sequential Bayesian inference problem by amortizing the previously prohibitive computational cost of inference.

Naturally, new questions arise with the success of these models. ANS itself is a reaction to our findings on the conditioning gap, which was found upon close examination of assumptions made by DVBFs, VTSSI, and many related earlier models. Even ANS, despite encouraging initial results, is not an off-the-shelf solution for learning SSM—at least, not yet. We are confident that the insights and suggestions of this thesis are suitable grounds for overcoming the growing pains of this still nascent field.

A key message of this thesis is that structured data like sequences are best tackled with equally structured models. The more surgically learnable building blocks like neural networks are injected, the more efficiently and robustly they learn. As we have shown, this insight transfers readily to a different mode of structured data, namely sets. Here, a theorem—rather than an algorithm—provides the leverage for learning neural set functions. Our deliberations have shown that the theory is insufficiently developed, and the resulting models can have wildly different secondary virtues, such as robustness to changes in the test scenario or hyper-parameters.

We started with the tongue-in-cheek truism that neural networks were everything and nothing to this thesis. Eight chapters in, we want to close by reflecting on just how much truth there is to the statement.

All contributions of this thesis make strong use of neural networks in one way or the other. If it were not for neural networks, we would not be asking some of the questions tackled in this thesis. And yet, neural networks are a mere vehicle in many regards. The algorithms presented transcend neural networks in the sense that they start from an established algorithm or theorem, and neural networks only come into play when it comes to implementing the algorithms. More so, when we uncover failings of the presented algorithms, they can be traced back to an unexpected show of idiosyncrasies of the involved neural networks. If there were a more flexible, more easily learned alternative to neural networks, it could be hot-swapped with limited or no adjustments. In this light, it is revealing that chapter 3—the core background chapter—does not even mention neural networks.

Algorithms and theory can serve as a phenomenal inductive bias for whatever model one hopes to fit. The better the respective template is understood when deriving a model or respective learning algorithm, the more efficiently we can make use of learnable components. If the reader were to take away one thing and one thing only from this thesis, it should be this.

APPENDICES

A BACKGROUND

A.1 DETAILS ON VAE ON MNIST EXAMPLE

Model and Training Details

The VAE consists of

1. the approximate posterior implemented by an MLP with two hidden layers of 512 units and ReLU activations returning a member of the variational family of diagonal two-dimensional Gaussians, the covariance diagonal is rectified with the exponential function (summary: 784-512(ReLU)-512(ReLU)-4(2+2(exp));
2. the likelihood model is implemented by an MLP with two hidden layers of 512 units and ReLU activations returning the logit of a Bernoulli distribution for each pixel (summary: 2-512(ReLU)-512(ReLU)-784).

The 70 000 samples in the data set were split (55 000, 5000, 10 000) into training, validation, and test set. The model was trained with mini-batch SGD with the Adam optimizer (Kingma and Ba, 2015) on the ELBO, with batch size 50 and step size 10^{-4}. The training lasted for 3000 epochs, the evaluated model was selected based on the validation ELBO.

Generative Samples

The generative samples in fig. 2.3 are sampled with the following procedure.

First, we span an evenly-spaced grid in the square area

$$[\mathrm{cdf}(-2.5), \mathrm{cdf}(2.5)]^2 \subset [0, 1]^2, \tag{A.1}$$

where cdf refers to the cumulative distribution function of the standard Gaussian distribution.

Then, we transform each two-dimensional grid point into a latent state $\mathbf{z}$ by pointwise application of the inverse cdf, akin to component-wise inverse transform sampling, except the uniform samples are grid-based instead of random.

This procedure is a compromise between a grid-based evaluation for a consistent arrangement of the samples and an evaluation with latent states $\mathbf{z}$ faithful to the spread of probability mass in the prior. The resulting latent states are shown in fig. A.1.

Lastly, the samples shown in fig. 2.3 are created by applying the likelihood model $p_\theta(\mathbf{x} \mid \mathbf{z})$. The figure depicts means in gray scale.

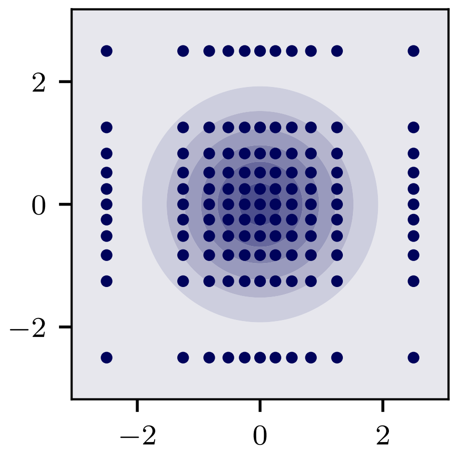


Figure A.1: The latent states used to produce the samples in fig. 2.3, see appendix A.1 for details. Contour plot of the standard Gaussian pdf in the background.

A.2 RENYI DIVERGENCES AND BOUNDS

The KL divergence can be viewed as special cases of the Renyi divergence

$$D_\alpha(p \parallel q) \equiv \frac{1}{\alpha - 1} \ln \int p(\mathbf{z})^\alpha q(\mathbf{z})^{1-\alpha} \, d\mathbf{z} \geqslant 0 \tag{A.2}$$

with parameter $\alpha > 0$ (Li and Richard E. Turner, 2016). One can show that

$$\lim_{\alpha \to 1} D_\alpha(p \parallel q) = \mathrm{KL}(p \parallel q). \tag{A.3}$$

Similar to the ELBO, one may derive a variational bound

$$\begin{aligned} \ln p(\mathbf{x}) &\geqslant \ln p(\mathbf{x}) - D_\alpha(q(\mathbf{z}) \parallel p(\mathbf{z} \mid \mathbf{x})) && \text{(A.4)} \\ &= \frac{1}{1-\alpha} \ln \mathbb{E}_{q(\mathbf{z})}\left[\left(\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}\right)^{1-\alpha}\right] && \text{(A.5)} \\ &\equiv \mathcal{L}_\alpha. && \text{(A.6)} \end{aligned}$$

The bound is continuous and non-increasing in α. We can devise a straightforward estimator

$$\mathcal{L}_{\alpha,K} \equiv \frac{1}{1-\alpha} \ln \sum_{k=1}^{K} \left(\frac{p\left(\mathbf{x}, \mathbf{z}^{(k)}\right)}{q\left(\mathbf{z}^{(k)}\right)}\right)^{1-\alpha}, \quad \mathbf{z}^{(k)} \sim q(\mathbf{z}). \tag{A.7}$$

We recognize the IWAE bound as the special case $\alpha = 0$. The estimator is biased by Jensen's inequality due to the expectation inside the logarithm.

For a fixed α, the estimator has interesting properties:

- The bias vanishes with increasing K, i. e.,

$$\lim_{K \to \infty} \mathbb{E}[\mathcal{L}_{\alpha,K}] = \mathcal{L}_\alpha. \tag{A.8}$$

- It is non-decreasing in K if $\alpha \leqslant 1$ and non-increasing in K if $\alpha \geqslant 1$. This allows for sandwiching $\ln p(\mathbf{x})$ with two different values of α and increasing K.
- For $K = 1$ and all α,

$$\mathcal{L}_{\alpha,1} = \ln \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}. \tag{A.9}$$

  $\mathcal{L}_{\alpha,1}$ is a single-sample ELBO estimator. That is, for low K the estimator is biased towards $\mathcal{L}_1$, the ELBO.

A.3 BAYESIAN UPDATES, FUSION, UNCERTAINTY

In the case of Gaussians, eq. (3.47) shows that the fusion of two Gaussian distributions leads to reduced variance. For Gaussians, this directly translates to reduced entropy—loosely speaking, we reduce uncertainty.

A Bayesian update in a Gaussian model, like the update step in a Kalman filter, will thus reduce uncertainty. It is tempting to assume that this is always the case since it fits the intuition that more information leads to better results. More information leads to more accurate beliefs—always assuming the model is specified correctly—but that need not translate to reduced uncertainty. In this section, we will provide counterexamples and counterarguments.

Most straightforward is a technical counterexample for a product of pdfs. Consider the two beta distributions $B(0.5, 1.5)$ and $B(1.5, 0.5)$. Multiplying their pdfs and renormalizing yields the pdf of the Beta distribution $B(1, 1)$, otherwise known as the uniform distribution $\mathcal{U}[0, 1]$. This is of course not a coincidence: the two source distributions were chosen to be inverse to one another in the sense that their pdfs f and g fulfill $f \propto {}^{1}\!/_{g}$. The fixed interval of the Beta distribution allows this construction, the inverse of a Gaussian pdf is not proportional to a valid pdf. On a fixed interval, the uniform distribution is always the distribution with highest variance or entropy. That is, the fusion of these two pdfs indeed increased uncertainty, in this case to the maximally possible extent.

A similar counterexample for a Bayesian update is a textbook i. i. d. sequence of Bernoulli coin tosses. After N tosses all showing up heads, the posterior is $B(1, N+1)$ (starting from a uniform prior). A subsequent tail toss updates the posterior to $B(2, N+1)$. It is easy to verify that the entropy for, e. g., $N = 9$ increases after the tail toss. Intuitively, it questions the strong belief in a large bias of the coin.

On a general level, the law of total variance is instructive[1]:

$$\mathrm{Var}[\mathbf{x}] = \mathbb{E}_{p(\mathbf{y})}[\mathrm{Var}[\mathbf{x} \mid \mathbf{y}]] + \underbrace{\mathrm{Var}\left[\mathbb{E}_{p(\mathbf{y})}[\mathbf{x} \mid \mathbf{y}]\right]}_{\geqslant 0} \tag{A.10}$$

$$\implies \mathbb{E}_{p(\mathbf{y})}[\mathrm{Var}[\mathbf{x} \mid \mathbf{y}]] \leqslant \mathrm{Var}[\mathbf{x}]. \tag{A.11}$$

The variance is reduced *on average*, but for a single realization of $\mathbf{y}$ it may be larger.

A similar result can be shown for entropy:

$$\mathbb{H}(\mathbf{x}) = \mathbb{H}(\mathbf{x} \mid \mathbf{y}) + \underbrace{\mathrm{MI}(\mathbf{x}, \mathbf{y})}_{\geqslant 0} \geqslant \mathbb{H}(\mathbf{x} \mid \mathbf{y}), \tag{A.12}$$

where $\mathbb{H}$ denotes entropy and MI mutual information. As in the case of variance, the key insight is that the conditional entropy on the right-hand side is an expected value w. r. t. $p(\mathbf{y})$, i. e., entropy is reduced on average but not necessarily for all realizations of $\mathbf{y}$.

A.4 A MATRIX IDENTITY FOR KALMAN FILTERS

In eqs. (3.51) and (3.52), it is claimed that

$$\left(\left(\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top\right)^{-1} + \mathbf{R}^{-1}\right)^{-1} = \mathbf{H}\left(\boldsymbol{\Sigma}_t^{(p)} - \mathbf{K}\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\right)\mathbf{H}^\top. \tag{A.13}$$

The key to proving this identity is Woodbury's matrix inversion lemma (K. P. Murphy, 2012, Corollary 4.3.1). For matrices $\mathbf{E}, \mathbf{F}, \mathbf{G}, \mathbf{J}$, with $\mathbf{E}, \mathbf{J}$ invertible, the lemma is

$$\left(\mathbf{E} - \mathbf{F}\mathbf{J}^{-1}\mathbf{G}\right)^{-1} = \mathbf{E}^{-1} + \mathbf{E}^{-1}\mathbf{F}\left(\mathbf{J} - \mathbf{G}\mathbf{E}^{-1}\mathbf{F}\right)\mathbf{G}\mathbf{E}^{-1}. \tag{A.14}$$

Identifying

$$\mathbf{E} = \left(\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top\right)^{-1}, \qquad \mathbf{J} = -\mathbf{R}, \qquad \mathbf{F} = \mathbf{G} = \mathbf{I}_{d_x}, \tag{A.15}$$

with the identity matrix $\mathbf{I}_{d_x}$ of rank d_x, we can prove the claim:

$$\left(\left(\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top\right)^{-1} + \mathbf{R}^{-1}\right)^{-1} \tag{A.16}$$

$$= \mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top + \mathbf{H}\underbrace{\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top\left(-\mathbf{R} - \mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top\right)^{-1}}_{=-\mathbf{K}}\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top \tag{A.17}$$

$$= \mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top - \mathbf{H}\mathbf{K}\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\mathbf{H}^\top \tag{A.18}$$

$$= \mathbf{H}\left(\boldsymbol{\Sigma}_t^{(p)} - \mathbf{K}\mathbf{H}\boldsymbol{\Sigma}_t^{(p)}\right)\mathbf{H}^\top. \tag{A.19}$$

1 Admittedly though, the notational conflation of random variable and realization makes this equation hard to parse.

B AUTO-ENCODING STATE-SPACE MODELS

The following appendix is adopted from Akhundov et al. (2019). Minor adaptations in style and spelling have not been highlighted.

B.1 MOVING MNIST DATA SET

Our data sets consist of 50 000 training, 10 000 validation, and 10 000 test sequences with a variable number of MNIST digits moving within 50×50 frames. The length of the sequences is 20. The number of digits in each sequence is sampled uniformly at random from $\{0, 1, 2\}$ but is fixed for each sequence. MNIST digits for each sequence are sampled uniformly at random from the original MNIST data set. The MNIST digits in our test set are sampled only from the MNIST test set whereas the ones in our training and validation sets are sampled only from the MNIST training set.

Four versions of our data set are determined by combination of two factors, depending on

- whether digit motion is linear or elliptic and
- whether two digits in the first frame are allowed to overlap.

The digits are placed at a random position in the initial frame with the condition of residing within the frame. In the non-overlapping first frame data set two digits are not allowed to overlap in the first frame, i. e., they may not share non-zero intensity pixels but may still overlap in further frames.

In the data set with linear motion, a random velocity vector is sampled for each digit and kept constant during motion, except flipping the components of the velocity at the edges of the frame: when at least one pixel of the digit goes out of frame after a motion step, the digit bounces off the edge.

In the data set with elliptic motion, a random elliptic trajectory is sampled for each digit such that a digit stays within the frame while moving along it. Angular velocity of each individual object is also sampled randomly and kept constant throughout the sequence.

As the velocity magnitudes are sampled from uniform distributions, while objects are moving their positions take on fractional values. Instead of rounding the position to the nearest integer pixel and pasting the same constellation of pixels as in the original digit at a new discrete position, we maintain the real position values and through bilinear interpolation smoothen the digit motion. We believe that this makes our data sets closer to real video sequences, in which object motion is typically smooth.

DDPAE DDPAE and VTSSI models with the prediction performance reported in fig. 5.6 were trained on the data generated by the script from the official DDPAE repository[1]. The test set was also generated by the DDPAE script because the original Moving MNIST data set lacks ground truth position annotation. It is worth mentioning that VTSSI was trained on 50 000 20-frame sequences whereas DDPAE was trained on streaming data (with every batch being randomly generated). The performance of both models reported in fig. 5.6 is evaluated on the test set.

SQAIR SQAIR and VTSSI models with the prediction performance reported in figs. 5.7a and 5.7b were trained on three different data sets corresponding to the two figures. SQAIR data corresponding to fig. 5.7a was generated by the data generation script from the official SQAIR repository[2], without noise and acceleration in digit motion. Our linear data corresponding to fig. 5.7b is comprised of 10-frame sequences structurally similar to our non-overlapping linear data set, with the exception of all frame edges being virtually shifted 3 pixels away from the center. This is to allow the digits going deeper out of frame before bouncing (for higher similarity with SQAIR's data). Model performance reported in fig. 5.7 is evaluated on hold-out test sets.

B.1.1 Evaluation Details

The accuracies reported in table 5.1 are computed by dividing the number of sequences where the number of objects is correctly inferred by the total number of sequences in the test set. AIR's accuracy is computed per frame as it may infer different numbers of objects from different frames of a single sequence, e. g., when the objects are highly overlapping.

The position error reported in table 5.1 and figs. 5.6 and 5.7 is computed as a distance in pixels between the ground truth object position and the positions inferred or predicted by the model. Ground truth object positions in all data sets correspond to the geometric centers of the tight bounding boxes around the object. The positions inferred or predicted by the models are translated into pixel coordinates before being compared with the ground truth positions. The position error is computed per inferred object and not per sequence: i. e., if there are two objects in one sequence, those are treated as two

1 https://github.com/jthsieh/DDPAE-video-prediction
2 https://github.com/akosiorek/sqair

different subjects of comparison. When there are multiple possible matchings between ground truth and inferred objects, we pick the matching that minimizes the summed distance error on a prefix of a sequence. Observation horizons of the models are used as the length of matching-determining prefixes (e. g., 10 in VTSSI vs. DDPAE and 3 in VTSSI vs. SQAIR evaluation).

At test time, DDPAE and VTSSI replace random variables in the computational graph by their modes. This proves to yield more accurate one-shot long-term predictions of object motion. As the SQAIR code from the official repository samples generative trajectories randomly, this would give a comparative disadvantage to SQAIR. For this reason, during evaluation we have modified the SQAIR code to replace all random variables by their modes, the same way as DDPAE and VTSSI do. This modification substantially improved the prediction performance metrics of SQAIR. We also modified the configuration of the trained SQAIR models to avoid dropping the objects from the sequence even when they disappear behind an edge of a frame. After this change SQAIR always preserved the objects inferred from the first frame throughout the sequence.

C LEARNING BY SMOOTHING

C.1 OPTIMAL PARTIALLY-CONDITIONED POSTERIORS

In this section, we show that assuming either $q_{\mathcal{C}}(\mathbf{z}) = p(\mathbf{z} \mid \mathcal{C})$ or $q_{\mathcal{C}}(\mathbf{z}) = p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})$ implies $p(\overline{\mathcal{C}} \mid \mathbf{z}, \mathcal{C}) = p(\overline{\mathcal{C}} \mid \mathcal{C})$. Here, we will make use of the rewritten form of the optimal shared partially-conditioned posterior

$$q_{\mathcal{C}}(\mathbf{z}) \propto p(\mathbf{z} \mid \mathcal{C}) \exp\left(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\ln \frac{p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})}{p(\mathbf{z} \mid \mathcal{C})}\right]\right) \tag{C.1}$$

from eq. (6.15).

Firstly,

$$\begin{aligned}
&q_{\mathcal{C}}(\mathbf{z}) = p(\mathbf{z} \mid \mathcal{C}) && \text{(C.2)}\\
\implies &\exp\left(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\ln \frac{p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})}{p(\mathbf{z} \mid \mathcal{C})}\right]\right) = 1 && \text{(C.3)}\\
\implies &\frac{p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})}{p(\mathbf{z} \mid \mathcal{C})} = \frac{p(\overline{\mathcal{C}} \mid \mathbf{z}, \mathcal{C})}{p(\overline{\mathcal{C}} \mid \mathcal{C})} = 1 && \text{(C.4)}\\
\implies &p(\overline{\mathcal{C}} \mid \mathbf{z}, \mathcal{C}) = p(\overline{\mathcal{C}} \mid \mathcal{C}). && \text{(C.5)}
\end{aligned}$$

Secondly,

$$\begin{aligned}
&q_{\mathcal{C}}(\mathbf{z}) = p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}}) && \text{(C.6)}\\
\implies &\exp\left(\mathbb{E}_{p(\overline{\mathcal{C}}|\mathcal{C})}\left[\ln \frac{p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})}{p(\mathbf{z} \mid \mathcal{C})}\right]\right) \propto \frac{p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})}{p(\mathbf{z} \mid \mathcal{C})} && \text{(C.7)}\\
\implies &\frac{p(\mathbf{z} \mid \mathcal{C}, \overline{\mathcal{C}})}{p(\mathbf{z} \mid \mathcal{C})} = \frac{p(\overline{\mathcal{C}} \mid \mathbf{z}, \mathcal{C})}{p(\overline{\mathcal{C}} \mid \mathcal{C})} \text{ is constant w. r. t. } \overline{\mathcal{C}} && \text{(C.8)}\\
\implies &p(\overline{\mathcal{C}} \mid \mathbf{z}, \mathcal{C}) = p(\overline{\mathcal{C}} \mid \mathcal{C}). && \text{(C.9)}
\end{aligned}$$

C.2 PROOF OF SUBOPTIMAL GENERATIVE MODEL

We investigate whether a maximum likelihood solution

$$p^{\star} = \arg\min_{p} \mathbb{E}_{\hat{p}(\mathbf{x}_{1:T})}[-\ln p(\mathbf{x}_{1:T})] \tag{C.10}$$

is a minimum of the expected negative ELBO. From calculus of variations (Gelfand and Fomin, 2003), we derive necessary optimality conditions for maximum likelihood and the expected negative ELBO as

$$0 \overset{!}{=} \frac{d\mathbb{E}_{\hat{p}(\mathbf{x}_{1:T})}[-\ln p(\mathbf{x}_{1:T})]}{dp} \alpha \frac{d\mathcal{G}}{dp}, \tag{C.11}$$

$$0 \overset{!}{=} \frac{d\mathbb{E}_{\hat{p}(\mathbf{x}_{1:T})}[-\ln p(\mathbf{x}_{1:T})]}{dp} + \frac{d\mathbb{E}_{\hat{p}(\mathbf{x}_{1:T})}[\mathrm{KL}]}{dp} + \lambda \frac{d\mathcal{G}}{dp}, \tag{C.12}$$

respectively. $\mathcal{G}$ is a constraint functional ensuring that p is a valid density, λ and α are Lagrange multipliers. KL refers to the posterior divergence in eq. (6.2). Equating (C.11) and (C.12) and rearranging gives

$$\frac{d\mathbb{E}_{\hat{p}(\mathbf{x}_{1:T})}[\mathrm{KL}]}{dp} + (\alpha - \lambda)\frac{d\mathcal{G}}{dp} = 0. \tag{C.13}$$

Equation (C.13) is a necessary and sufficient condition (Erven and Harremoës, 2014) that the KL divergence is minimized *as a function of* p, which happens when $p(\mathbf{z}_t \mid \mathcal{C}_t, \overline{\mathcal{C}}_t) = q(\mathbf{z}_t \mid \mathcal{C}_t)$ for all t.

C.3 DETAILS ON THE EXAMPLE LINEAR GAUSSIAN SYSTEM

The example LGS

$$p(\mathbf{z}_1) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{C.14}$$

$$\mathbf{z}_{t+1} = \mathbf{A}\mathbf{z}_t + \boldsymbol{\epsilon}_{t+1}, \qquad \boldsymbol{\epsilon}_{t+1} \sim \mathcal{N}(0, \mathbf{Q}), \tag{C.15}$$

$$\mathbf{x}_t = \mathbf{H}\mathbf{z}_t + \boldsymbol{\delta}_t, \qquad \boldsymbol{\delta}_t \sim \mathcal{N}(0, \mathbf{R}), \tag{C.16}$$

has parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{A}, \mathbf{Q}, \mathbf{H}, \mathbf{R}$ chosen as follows.

The latent space as well as the observation space are of dimension $d_\mathbf{x} = d_\mathbf{z} = 2$.

The latent transition matrix $\mathbf{A}$ is designed to be rotating and slightly contracting. This translates to a scaled rotation matrix

$$\mathbf{J} = 0.95 \cdot \begin{bmatrix} \cos(20^\circ) & -\sin(20^\circ) \\ \sin(20^\circ) & \cos(20^\circ) \end{bmatrix}, \tag{C.17}$$

a real-valued Jordan block for a (pair of) complex eigenvalues indicating rotation.

$\mathbf{A}$ is similar (in the sense of linear mappings) to $\mathbf{J}$ with change of basis

$$\mathbf{M} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad \Longrightarrow \quad \mathbf{A} = \mathbf{M}\mathbf{J}\mathbf{M}^{-1}. \tag{C.18}$$

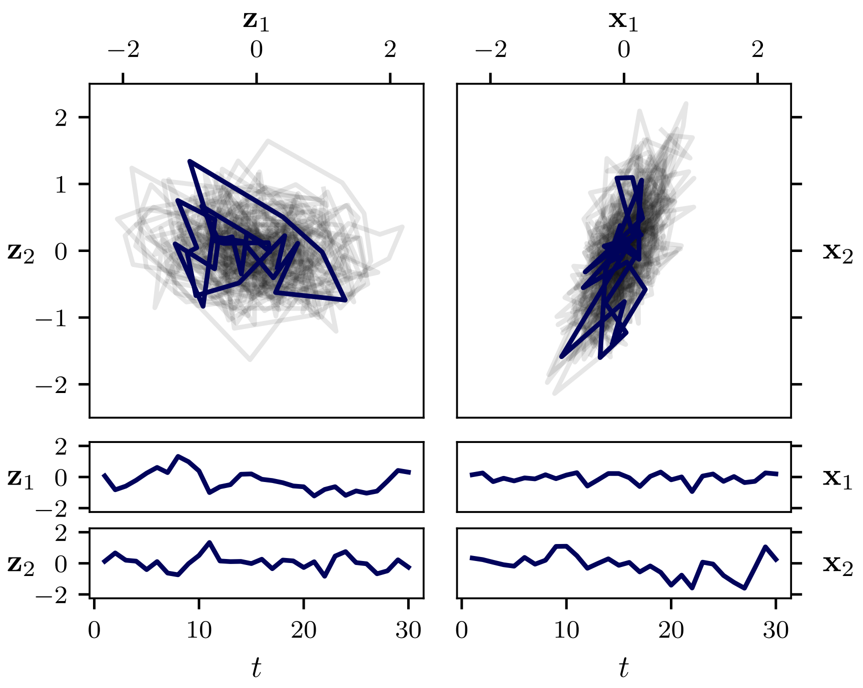


Figure C.1: 25 test set sequences from the LGS described in appendix C.3. Left: ground truth states. Right: ground truth observations Top: superposition of the 25 sequences, with a single highlighted sequence. Middle and bottom: the individual features of the highlighted sequence plotted against time.

The transition covariance matrix features correlation,

$$\mathbf{Q} = 0.5 \cdot \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}. \tag{C.19}$$

The observation model is a full-rank matrix that mixes latent states and adds white noise with

$$\mathbf{H} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{C.20}$$

The initial state distribution is

$$\mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, 0.7 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right). \tag{C.21}$$

The entire system is scaled down so that states and observations are in a range immediately suitable for neural networks without further normalization. This is achieved by scaling down state distributions with a factor 0.6 and emission distributions with 0.25. The covariance matrices are multiplied by the squared scaling factors.

The final system is

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} 0.252 & 0 \\ 0 & 0.252 \end{bmatrix}, \tag{C.22}$$

$$\mathbf{A} \approx \begin{bmatrix} 0.7306 & -0.3899 \\ 0.1950 & 0.3407 \end{bmatrix}, \qquad \mathbf{Q} = \begin{bmatrix} 0.18 & -0.09 \\ -0.09 & 0.18 \end{bmatrix}, \tag{C.23}$$

$$\mathbf{H} = \begin{bmatrix} 0.25 & 0.5 \\ 0.75 & 1 \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} 0.0625 & 0 \\ 0 & 0.0625 \end{bmatrix}. \tag{C.24}$$

We use 500 sequences of length $T = 30$ each for training, validation, and test set. A subsample of 25 test sequences is depicted in fig. C.1.

C.4 LINEAR GAUSSIAN BACKWARD FILTER

For the LGS

$$p(\mathbf{z}_1) \sim \mathcal{N}(\mu, \Sigma), \tag{C.25}$$

$$\mathbf{z}_{t+1} = \mathbf{A}\mathbf{z}_t + \epsilon_{t+1}, \qquad \epsilon_{t+1} \sim \mathcal{N}(0, \mathbf{Q}), \tag{C.26}$$

$$\mathbf{x}_t = \mathbf{H}\mathbf{z}_t + \delta_t, \qquad \delta_t \sim \mathcal{N}(0, \mathbf{R}), \tag{C.27}$$

the backward filter for $t = 1, \ldots, T-1$ can be written as

$$\beta_t(\mathbf{z}_t) = \mathcal{N}(\mathbf{x}_{t+1:T} \mid \mathbf{B}_t\mathbf{z}_t, \Sigma_t), \tag{C.28}$$

$$\mathbf{B}_t = \begin{bmatrix} \mathbf{HA} \\ \vdots \\ \mathbf{HA}^{T-t} \end{bmatrix}, \tag{C.29}$$

$$\Sigma_t = \begin{bmatrix} \mathbf{R} & & \\ & \ddots & \\ & & \mathbf{R} \end{bmatrix} + \left[\Sigma_t^{(i,j)}\right]_{i,j=1,\ldots,T-t}, \tag{C.30}$$

$$\Sigma_t^{(i,j)} = \begin{cases} \sum_{\tau=1}^{j} \mathbf{HA}^{i-\tau}\mathbf{Q}\left(\mathbf{A}^{j-\tau}\right)^\top \mathbf{H}^\top, & i \geqslant j \\ \left(\Sigma_t^{(j,i)}\right)^\top, & i < j \end{cases} \tag{C.31}$$

or recursively

$$\mathbf{B}_t = \begin{bmatrix} \mathbf{H} \\ \mathbf{B}_{t+1} \end{bmatrix} \mathbf{A}, \tag{C.32}$$

$$\Sigma_t = \begin{bmatrix} \mathbf{R} + \mathbf{HQH}^\top & \mathbf{HQB}_{t+1}^\top \\ \mathbf{B}_{t+1}\mathbf{QH}^\top & \Sigma_{t+1} + \mathbf{B}_{t+1}\mathbf{QB}_{t+1}^\top \end{bmatrix}. \tag{C.33}$$

Proof. By induction backward in time.

BASE CASE $t = T - 1$

$$\beta_{T-1}(\mathbf{z}_{T-1}) = p(\mathbf{x}_T \mid \mathbf{z}_{T-1}) \tag{C.34}$$

We know

$$\begin{aligned} \mathbf{x}_T &= \mathbf{H}\mathbf{z}_T + \delta_T & \delta_T &\sim \mathcal{N}(0, \mathbf{R}) & \text{(C.35)} \\ &= \mathbf{H}(\mathbf{A}\mathbf{z}_{T-1} + \boldsymbol{\epsilon}_T) + \delta_T & \boldsymbol{\epsilon}_T &\sim \mathcal{N}(0, \mathbf{Q}) & \text{(C.36)} \\ &\sim \mathcal{N}\Big(\mathbf{H}\mathbf{A}\mathbf{z}_{T-1}, \mathbf{H}\mathbf{Q}\mathbf{H}^\top + \mathbf{R}\Big), & & & \text{(C.37)} \end{aligned}$$

which follows from (i) linear mappings of Gaussians and (ii) independence of δ_T and $\boldsymbol{\epsilon}_T$. The distribution in eq. (C.37) complies with eqs. (C.28) to (C.31).

INDUCTION STEP $t + 1 \to t$ By definition,

$$\mathbf{x}_{t+1} = \mathbf{H}\mathbf{z}_{t+1} + \delta_{t+1}, \qquad \delta_{t+1} \sim \mathcal{N}(0, \mathbf{R}). \tag{C.38}$$

By the induction hypothesis,

$$\mathbf{x}_{t+2:T} = \mathbf{B}_{t+1}\mathbf{z}_{t+1} + \zeta_{t+1} \qquad \zeta_{t+1} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{t+1}). \tag{C.39}$$

In an SSM $\mathbf{x}_{t+1}$ and $\mathbf{x}_{t+2:T}$ are conditionally independent given $\mathbf{z}_{t+1}$. With

$$\boldsymbol{\eta}_{t+1} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \boldsymbol{\Sigma}_{t+1} \end{bmatrix}\right), \tag{C.40}$$

$$\boldsymbol{\epsilon}_{t+1} \sim \mathcal{N}(0, \mathbf{Q}), \tag{C.41}$$

this allows us to write

$$\begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{x}_{t+2:T} \end{bmatrix} = \begin{bmatrix} \mathbf{H} \\ \mathbf{B}_{t+1} \end{bmatrix} \mathbf{z}_{t+1} + \boldsymbol{\eta}_{t+1} \tag{C.42}$$

$$= \begin{bmatrix} \mathbf{H} \\ \mathbf{B}_{t+1} \end{bmatrix} (\mathbf{A}\mathbf{z}_t + \boldsymbol{\epsilon}_{t+1}) + \boldsymbol{\eta}_{t+1}, \tag{C.43}$$

which is a Gaussian with mean

$$\begin{bmatrix} \mathbf{H} \\ \mathbf{B}_{t+1} \end{bmatrix} \mathbf{A}\mathbf{z}_t =: \mathbf{B}_t\mathbf{z}_t \tag{C.44}$$

and covariance

$$\boldsymbol{\Sigma}_t := \begin{bmatrix} \mathbf{H} \\ \mathbf{B}_{t+1} \end{bmatrix} \mathbf{Q} \begin{bmatrix} \mathbf{H} \\ \mathbf{B}_{t+1} \end{bmatrix}^\top + \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \boldsymbol{\Sigma}_{t+1} \end{bmatrix} \tag{C.45}$$

$$= \begin{bmatrix} \mathbf{R} + \mathbf{H}\mathbf{Q}\mathbf{H}^\top & \mathbf{H}\mathbf{Q}\mathbf{B}_{t+1}^\top \\ \mathbf{B}_{t+1}\mathbf{Q}\mathbf{H}^\top & \boldsymbol{\Sigma}_{t+1} + \mathbf{B}_{t+1}\mathbf{Q}\mathbf{B}_{t+1}^\top \end{bmatrix}. \tag{C.46}$$

This proves the recursions in eqs. (C.32) and (C.33). Further, it is easy to see that eq. (C.44) implies eq. (C.29) under induction hypothesis.

It remains to show that eq. (C.46) implies eqs. (C.30) and (C.31) under the induction hypothesis. From eq. (C.46), we see that

$$\mathbf{\Sigma}_t^{(1,1)} = \mathbf{R} + \mathbf{H}\mathbf{Q}\mathbf{H}^\top \tag{C.47}$$

fulfills eqs. (C.30) and (C.31). Further, by the induction hypothesis

$$\begin{bmatrix} \mathbf{\Sigma}_t^{(1,1)} \\ \vdots \\ \mathbf{\Sigma}_t^{(T-t,1)} \end{bmatrix} \tag{C.48}$$

$$= \mathbf{B}_{t+1}\mathbf{Q}\mathbf{H}^\top = \begin{bmatrix} \mathbf{H}\mathbf{A} \\ \vdots \\ \mathbf{H}\mathbf{A}^{T-(t+1)} \end{bmatrix} \mathbf{Q}\mathbf{H}^\top \tag{C.49}$$

$$= \left[\mathbf{H}\mathbf{A}^{i-1}\mathbf{Q}\mathbf{H}^\top\right]_{i=2,\dots,T-t}, \tag{C.50}$$

which complies with eqs. (C.30) and (C.31). By symmetry, the same holds for

$$\begin{bmatrix} \mathbf{\Sigma}_t^{(1,2)} & \cdots & \mathbf{\Sigma}_t^{(1,T-t)} \end{bmatrix} = \mathbf{H}\mathbf{Q}\mathbf{B}_{t+1}^\top. \tag{C.51}$$

By induction hypothesis,

$$\mathbf{B}_{t+1}\mathbf{Q}\mathbf{B}_{t+1}^\top = \begin{bmatrix} \mathbf{H}\mathbf{A} \\ \vdots \\ \mathbf{H}\mathbf{A}^{T-(t+1)} \end{bmatrix} \mathbf{Q} \begin{bmatrix} \mathbf{H}\mathbf{A} \\ \vdots \\ \mathbf{H}\mathbf{A}^{T-(t+1)} \end{bmatrix}^\top \tag{C.52}$$

$$= \left[\mathbf{H}\mathbf{A}^{i}\mathbf{Q}(\mathbf{A}^{j})^\top\mathbf{H}^\top\right]_{i,j=1,\dots,T-(t+1)}. \tag{C.53}$$

Consequently, again by induction hypothesis, eq. (C.46), and assuming $T-2 \geqslant i \geqslant j \geqslant 1$,

$$\mathbf{\Sigma}_t^{(i+1,j+1)} \tag{C.54}$$

$$= \mathbf{\Sigma}_{t+1}^{(i,j)} + \mathbf{H}\mathbf{A}^{i}\mathbf{Q}(\mathbf{A}^{j})^\top\mathbf{H}^\top \tag{C.55}$$

$$= \mathbf{H}\mathbf{A}^{i}\mathbf{Q}(\mathbf{A}^{j})^\top\mathbf{H}^\top + \sum_{\tau=1}^{j} \mathbf{H}\mathbf{A}^{i-\tau}\mathbf{Q}\left(\mathbf{A}^{j-\tau}\right)^\top\mathbf{H}^\top \tag{C.56}$$

$$= \sum_{\tau=0}^{j} \mathbf{H}\mathbf{A}^{i-\tau}\mathbf{Q}\left(\mathbf{A}^{j-\tau}\right)^\top\mathbf{H}^\top \tag{C.57}$$

$$= \sum_{\tau=1}^{j+1} \mathbf{H}\mathbf{A}^{i+1-\tau}\mathbf{Q}\left(\mathbf{A}^{j+1-\tau}\right)^\top\mathbf{H}^\top, \tag{C.58}$$

which complies with eqs. (C.30) and (C.31). The case $j > i$ follows by symmetry, which concludes the argument. □

C.5 DETAILS ON ROW-WISE MNIST EXPERIMENTS

The data in this experiment are similar to those in section 6.3.3, i. e., each row of pixels of an MNIST digit is interpreted as a time step. The only difference is that this experiment uses gray-scale values instead of static binarization. All distributions and density networks of the model are Gaussian, except for $p(\mathbf{x}_t \mid \mathbf{z}_t)$, which is a pixel-wise Bernoulli distribution. The pixels are independent given $\mathbf{z}_t$. Backward filter and proposal share the same backward RNN. The transition mean can be residual, and the added residual is scaled down with a learnable scalar. The transition proposal density network gets the prior mean as input in order to avoid duplicate learning of the dynamics. The initial prior is fixed to a standard Gaussian. All weights are jointly trained on the ELBO estimator, NASMC is not used. The full set of hyper-parameters of the model is gathered in table C.1.

Table C.1: Hyper-parameters for qualitative MNIST experiment with ANS.

Parameter				Value
backward filter	full covariance			True
	hidden activation			elu
	n components			1
	n layers			3
	shared covariance			False
	units			64
dims	future			32
	observation			28
	particle			8
	proposal			2
	state			32
	time			28
optimizer	generative	clip norm		0.5
		cls		Adam
		kwargs	betas	(0.3, 0.999)
			lr	0.0001
proposal	init	full covariance		False
		hidden activation		elu
		n layers		2
		shared covariance		False
		units		64
	rnn	cls		RNN
		kwargs	hidden size	32
			input size	28
			num layers	1
	transit	full covariance		False
		hidden activation		softsign
		n layers		3
		residual		True
		residual scale init		0.01
		shared covariance		False
		units		128
		use prior		True
resampling	criterion			effective
	rel min n particle			0.35
	technique			iid
ssm	emission fn	hidden activation		elu
		n layers		2
		units		128
	init state dist	full covariance		False
		learnable loc		False
		learnable scale		False
	transition fn	full covariance		False
		hidden activation		softsign
		n layers		2
		residual		True
		residual scale init		0.1
		shared covariance		False
		units		128
training	batch size			64
	use nasmc			False

D SET-VALUED NEURAL FUNCTIONS

D.1 DETAILS ON MOTIVATING EXAMPLE

Data

We created $N = 20\,000$ independent Gaussian mixtures of two components with the following properties:

1. The two means were drawn randomly and independently from $\mathcal{U}[-0.8, 0]$ and $\mathcal{U}[0, 0.8]$, respectively.
2. The scales were drawn randomly and independently from the uniform distribution $\mathcal{U}[0.1, 0.3]$.
3. The mixture weight of the negative component was randomly drawn uniformly from the set $[0.1, 0.4] \cup [0.6, 0.9]$. The mixture weight of the positive component was then set to add up to 1.

Due to the symmetry of the parameter choices, the resulting classification problem is balanced. At the same time, the restrictions are strong enough to be leveraged by a Deep Set architecture.

Of each mixture, $P = 5$ particles were drawn i. i. d. The populations were split $(10\,000, 5000, 5000)$ into training, validation, and test set.

Models

The embedding dimension is $d = 32$. The embedding was performed by an MLP with two hidden layers of 128 units each with softsign activation functions and d linear output units, followed by a simple equivariance subtracting the particle mean from the output, followed by another MLP with two hidden layers of 128 units with softsign activation functions and d linear output units.

The aggregation was performed by either a mean or a max operation.

The processing was performed by an MLP with three hidden layers of 128 units each with softsign activation functions, mapping to a prediction probability normalized with the sigmoid function.

Training

The models were optimized by SGD on the (negative) binary cross entropy loss with the Adam optimizer (step size 10^{-4}). Training ran for

500 epochs with a batch size of 1000. Model selection was performed based on the prediction *accuracy* on the validation set. No hyper-parameter search was performed.

BIBLIOGRAPHY

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. URL: https://www.tensorflow.org/.

Achlioptas, Panos, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas (2018). "Learning Representations and Generative Models for 3D Point Clouds." In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 40–49. URL: http://proceedings.mlr.press/v80/achlioptas18a.html.

Akhundov, Adnan (2018). "Unsupervised Object Tracking with Variational Inference." Master thesis. Munich: Technische Universität München.

Akhundov, Adnan, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt (2019). *Variational Tracking and Prediction with Generative Disentangled State-Space Models*. arXiv: 1910.06205 [cs, stat]. URL: http://arxiv.org/abs/1910.06205.

Alemi, Alexander A., Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy (2018). "Fixing a Broken ELBO." In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 159–168. URL: http://proceedings.mlr.press/v80/alemi18a.html.

Antonini, Amado, Winter Guerra, Varun Murali, Thomas Sayre-McCord, and Sertac Karaman (2018). "The Blackbird Dataset: A Large-Scale Dataset for UAV Perception in Aggressive Flight." In: *Proceedings of the 2018 International Symposium on Experimental Robotics, ISER 2018, Buenos Aires, Argentina, November 5-8, 2018*. Ed. by Jing Xiao, Torsten Kröger, and Oussama Khatib. Vol. 11. Springer Proceedings in Advanced Robotics. Springer, pp. 130–139. DOI: 10.1007/978-3-030-33950-0\\_12.

Ba, Jimmy, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu (2016). "Using Fast Weights to Attend to the Recent Past." In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, pp. 4331–4339. URL: https://proceedings.neurips.cc/paper/2016/hash/9f44e956e3a2b7b5598c625fcc802c36-Abstract.html.

Babaeizadeh, Mohammad, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine (2018). "Stochastic Variational Video

Prediction." In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=rk49Mg-CW.

Babb, Tim (2015). *How a Kalman Filter Works, in Pictures*. URL: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/.

Baker, James E. (1987). "Reducing Bias and Inefficiency in the Selection Algorithm." In: *Proceedings of the 2nd International Conference on Genetic Algorithms, Cambridge, MA, USA, July 1987*, pp. 14–21.

Banville, Simon and Frank Diggelen (2016). "Precise GNSS for Everyone: Precise Positioning Using Raw GPS Measurements from Android Smartphones." In: *GPS World* 27, pp. 43–48.

Bayer, Justin (2015). "Learning Sequence Representations." PhD thesis. Munich: Technische Universität München. URL: https://nbn-resolving.org/urn:nbn:de:bvb:91-diss-20151102-1256381-1-9.

Bayer, Justin and Christian Osendorfer (2014). *Learning Stochastic Recurrent Networks*. arXiv: 1411.7610 [cs, stat]. URL: http://arxiv.org/abs/1411.7610.

Bayer, Justin, Maximilian Soelch, Atanas Mirchev, Baris Kayalibay, and Patrick van der Smagt (2021). "Mind the Gap When Conditioning Amortised Inference in Sequential Latent-Variable Models." In: *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net. URL: https://openreview.net/forum?id=a2gqxKDvYys.

Becker-Ehmck, Philip, Maximilian Karl, Jan Peters, and Patrick van der Smagt (2020). *Learning to Fly via Deep Model-Based Reinforcement Learning*. arXiv: 2003.08876 [cs, stat]. URL: http://arxiv.org/abs/2003.08876.

Becker-Ehmck, Philip, Jan Peters, and Patrick van der Smagt (2019). "Switching Linear Dynamics for Variational Bayes Filtering." In: *Proceedings of the 36th International Conference on Machine Learning*. International Conference on Machine Learning (ICML) 2019. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 553–562. URL: http://proceedings.mlr.press/v97/becker-ehmck19a.html.

Bewley, Alex, ZongYuan Ge, Lionel Ott, Fabio Tozeto Ramos, and Ben Upcroft (2016). "Simple Online and Realtime Tracking." In: *2016 IEEE International Conference on Image Processing, ICIP 2016, Phoenix, AZ, USA, September 25-28, 2016*. IEEE, pp. 3464–3468. DOI: 10.1109/ICIP.2016.7533003.

Bishop, Christopher M. (1994). *Mixture Density Networks*. URL: http://publications.aston.ac.uk/id/eprint/373/.

– (2007). *Pattern Recognition and Machine Learning, 5th Edition*. Information Science and Statistics. Springer. ISBN: 978-0-387-31073-2. URL: https://www.worldcat.org/oclc/71008143.

Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe (2017). "Variational Inference: A Review for Statisticians." In: *Journal of the American Statistical Association* 112.518, pp. 859–877. ISSN: 0162-1459. DOI: 10.1080/01621459.2017.1285773.

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra (2015). "Weight Uncertainty in Neural Network." In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Ed. by Francis R. Bach and David M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1613–1622. URL: http://proceedings.mlr.press/v37/blundell15.html.

Bottou, Léon (2010). "Large-Scale Machine Learning with Stochastic Gradient Descent." In: *19th International Conference on Computational Statistics, COMPSTAT 2010, Paris, France, August 22-27, 2010 - Keynote, Invited and Contributed Papers*. Ed. by Yves Lechevallier and Gilbert Saporta. Physica-Verlag, pp. 177–186. DOI: 10.1007/978-3-7908-2604-3\\_16.

Box, George E. P. (1976). "Science and Statistics." In: *Journal of the American Statistical Association* 71.356, pp. 791–799. ISSN: 0162-1459. DOI: 10.1080/01621459.1976.10480949.

– (1979). "Robustness in the Strategy of Scientific Model Building." In: *Robustness in Statistics*. Ed. by Robert L. Launer and Graham N. Wilkinson. Academic Press, pp. 201–236. ISBN: 978-0-12-438150-6. DOI: 10.1016/B978-0-12-438150-6.50018-2.

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). *OpenAI Gym*. arXiv: 1606.01540 [cs]. URL: http://arxiv.org/abs/1606.01540.

Burda, Yuri, Roger B. Grosse, and Ruslan Salakhutdinov (2016). "Importance Weighted Autoencoders." In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1509.00519.

Cadena, Cesar, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard (2016). "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age." In: *IEEE Trans. Robotics* 32.6, pp. 1309–1332. DOI: 10.1109/TRO.2016.2624754.

Chang, Michael, Tomer Ullman, Antonio Torralba, and Joshua B. Tenenbaum (2017). "A Compositional Object-Based Approach to Learning Physical Dynamics." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=Bkab5dqxe.

Chatterjee, Sourav and Persi Diaconis (2015). "The Sample Size Required in Importance Sampling." In: *The Annals of Applied Probability* 28, pp. 1099–1135. DOI: 10.1214/17-AAP1326.

Chen, Nutan, Francesco Ferroni, Alexej Klushyn, Alexandros Paraschos, Justin Bayer, and Patrick van der Smagt (2019). "Fast Approximate Geodesics for Deep Generative Models." In: *Artificial Neural Networks and Machine Learning - ICANN 2019: Deep Learning - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part II*. Ed. by Igor V. Tetko, Vera Kurková, Pavel Karpov, and Fabian J. Theis. Vol. 11728. Lecture Notes in Computer Science. Springer, pp. 554–566. DOI: 10.1007/978-3-030-30484-3\\_45.

Chen, Nutan, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick van der Smagt (2018). "Metrics for Deep Generative Models." In: *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*. Ed. by Amos J. Storkey and Fernando Pérez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, pp. 1540–1550. URL: http://proceedings.mlr.press/v84/chen18e.html.

Chen, Xu, Xiuyuan Cheng, and Stéphane Mallat (2014). "Unsupervised Deep Haar Scattering on Graphs." In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, pp. 1709–1717. URL: https://proceedings.neurips.cc/paper/2014/hash/892c91e0a653ba19df81a90f89d99bcd-Abstract.html.

Child, Rewon (2020). *Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images*. arXiv: 2011.10650 [cs]. URL: http://arxiv.org/abs/2011.10650.

Cho, Kyunghyun, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A Meeting of SIGDAT, a Special Interest Group of the ACL*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. ACL, pp. 1724–1734. DOI: 10.3115/v1/d14-1179.

Chung, Junyoung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio (2015). "A Recurrent Latent Variable Model for Sequential Data." In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, pp. 2980–2988. URL: http://papers.nips.cc/paper/5653-a-recurrent-latent-variable-model-for-sequential-data.

Cireşan, Dan C., Ueli Meier, Jonathan Masci, Luca M. Gambardella, and Jürgen Schmidhuber (2011a). "Flexible, High Performance Convolutional Neural Networks for Image Classification." In: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press, pp. 1237–1242. ISBN: 978-1-57735-514-4. DOI: 10.5591/978-1-57735-516-8/IJCAI11-210.

Cireşan, Dan C., Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber (2011b). "A Committee of Neural Networks for Traffic Sign Classification." In: *The 2011 International Joint Conference on Neural Networks, IJCNN 2011, San Jose, California, USA, July 31 - August 5, 2011*. IEEE, pp. 1918–1921. DOI: 10.1109/IJCNN.2011.6033458.

Clark, Jack (2017). *Let's Battle the Hype of AI by Coming up with Boring Alternate Terms! I'll Start. Deep Learning ===> Stacked Function Approximators*. URL: https://twitter.com/jackclarkSF/status/838986258542026752.

Cremer, Chris, Xuechen Li, and David Duvenaud (2018). "Inference Suboptimality in Variational Autoencoders." In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1086–1094. URL: http://proceedings.mlr.press/v80/cremer18a.html.

Cremer, Chris, Quaid Morris, and David Duvenaud (2017). "Reinterpreting Importance-Weighted Autoencoders." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=Syw2ZgrFx.

Cui, Zhiyong, Kristian Henrickson, Ruimin Ke, and Yinhai Wang (2020). "Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting." In: *IEEE Transactions on Intelligent Transportation Systems* 21.11, pp. 4883–4894. DOI: 10.1109/TITS.2019.2950416.

Cui, Zhiyong, Ruimin Ke, Ziyuan Pu, and Yinhai Wang (2019). *Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-Wide Traffic Speed Prediction*. arXiv: 1801.02143 [cs]. URL: http://arxiv.org/abs/1801.02143.

Del Moral, Pierre (1996). "Non Linear Filtering: Interacting Particle Solution." In: *Markov Processes and Related Fields* 2, pp. 555–580.

Denton, Emily and Rob Fergus (2018). "Stochastic Video Generation with a Learned Prior." In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1182–1191. URL: http://proceedings.mlr.press/v80/denton18a.html.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of* *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/n19-1423.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). "Density Estimation Using Real NVP." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=HkpbnH9lx.

Doersch, Carl (2016). *Tutorial on Variational Autoencoders*. arXiv: 1606.05908 `[cs, stat]`. URL: http://arxiv.org/abs/1606.05908.

Dosovitskiy, Alexey et al. (2015). "FlowNet: Learning Optical Flow with Convolutional Networks." In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, pp. 2758–2766. DOI: 10.1109/ICCV.2015.316.

Dykeman, Isaac (2016). *Conditional Variational Autoencoders*. URL: https://ijdykeman.github.io/ml/2016/12/21/cvae.html.

Edwards, Harrison and Amos J. Storkey (2017). "Towards a Neural Statistician." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=HJDBUF5le.

Erven, Tim van and Peter Harremoës (2014). "Rényi Divergence and Kullback-Leibler Divergence." In: *IEEE Transactions on Information Theory* 60.7, pp. 3797–3820. DOI: 10.1109/TIT.2014.2320500.

Eslami, S. M. Ali, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E. Hinton (2016). "Attend, Infer, Repeat: Fast Scene Understanding with Generative Models." In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, pp. 3225–3233. URL: https://proceedings.neurips.cc/paper/2016/hash/52947e0ade57a09e4a1386d08f17b656-Abstract.html.

Fraccaro, Marco, Simon Kamronn, Ulrich Paquet, and Ole Winther (2017). "A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 3601–3610. URL: https://proceedings.neurips.cc/paper/2017/hash/7b7a53e239400a13bd6be6c91c4f6c4e-Abstract.html.

Fraccaro, Marco, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther (2016). "Sequential Neural Models with Stochastic Layers." In: *Ad-*

vances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, pp. 2199–2207. URL: https://proceedings.neurips.cc/paper/2016/hash/208e43f0e45c4c78cafadb83d2888cb6-Abstract.html.

Gelfand, I. M. and S. V. Fomin (2003). *Calculus of Variations*. Trans. by Jerry Silverman. Mineola, N.Y: Dover Publications Inc. 240 pp. ISBN: 978-0-486-41448-5.

Girin, Laurent, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda (2020). *Dynamical Variational Autoencoders: A Comprehensive Review*. arXiv: 2008.12595 [cs, stat]. URL: http://arxiv.org/abs/2008.12595.

Goodfellow, Ian J., Yoshua Bengio, and Aaron C. Courville (2016). *Deep Learning*. Adaptive Computation and Machine Learning. MIT Press. ISBN: 978-0-262-03561-3. URL: http://www.deeplearningbook.org/.

Gordon, Daniel, Ali Farhadi, and Dieter Fox (2018). "Re(^{3}): Real-Time Recurrent Regression Networks for Visual Tracking of Generic Objects." In: *IEEE Robotics Autom. Lett.* 3.2, pp. 788–795. DOI: 10.1109/LRA.2018.2792152.

Goyal, Anirudh, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio (2017). "Z-Forcing: Training Stochastic Recurrent Networks." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 6713–6723. URL: http://papers.nips.cc/paper/7248-z-forcing-training-stochastic-recurrent-networks.

Gregor, Karol, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber (2019). "Temporal Difference Variational Auto-Encoder." In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. arXiv: 1806.03107. URL: https://openreview.net/forum?id=S1x4ghC9tQ.

Gu, Shixiang, Zoubin Ghahramani, and Richard E. Turner (2015). "Neural Adaptive Sequential Monte Carlo." In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, pp. 2629–2637. URL: https://proceedings.neurips.cc/paper/2015/hash/99adff456950dd9629a5260c4de21858-Abstract.html.

Guttenberg, Nicholas, Nathaniel Virgo, Olaf Witkowski, Hidetoshi Aoki, and Ryota Kanai (2016). *Permutation-Equivariant Neural Net-*

works Applied to Dynamics Prediction. arXiv: 1612.04530 [cs, stat]. URL: http://arxiv.org/abs/1612.04530.

Ha, David and Jürgen Schmidhuber (2018a). “Recurrent World Models Facilitate Policy Evolution.” In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, pp. 2455–2467. URL: http://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution.

– (2018b). *World Models*. arXiv: 1803.10122 [cs, stat]. URL: http://arxiv.org/abs/1803.10122.

Hafner, Danijar, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson (2019). “Learning Latent Dynamics for Planning from Pixels.” In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 2555–2565. URL: http://proceedings.mlr.press/v97/hafner19a.html.

Hecht-Nielsen, Robert (1988). “Theory of the Backpropagation Neural Network.” In: *Neural Networks* 1 (Supplement-1), pp. 445–448. DOI: 10.1016/0893-6080(88)90469-8.

Hennig, Jay A., Akash Umakantha, and Ryan C. Williamson (2017). *A Classifying Variational Autoencoder with Application to Polyphonic Music Generation*. arXiv: 1711.07050 [cs, stat]. URL: http://arxiv.org/abs/1711.07050.

Higgins, Irina, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2017). “Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework.” In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=Sy2fzU9gl.

Hinton, Geoffrey E. (2002). “Training Products of Experts by Minimizing Contrastive Divergence.” In: *Neural Computation* 14.8, pp. 1771–1800. ISSN: 0899-7667. DOI: 10.1162/089976602760128018.

Hochreiter, Sepp (1991). “Untersuchungen Zu Dynamischen Neuronalen Netzen.” Diploma thesis. Munich: Technische Universität München. URL: http://www.bioinf.jku.at/publications/older/3804_2.pdf.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long Short-Term Memory.” In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

Hoffman, Matthew D., David M. Blei, Chong Wang, and John Paisley (2013). “Stochastic Variational Inference.” In: *Journal of Machine*

Learning Research 14.4, pp. 1303–1347. ISSN: 1533-7928. URL: http://jmlr.org/papers/v14/hoffman13a.html.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer Feedforward Networks Are Universal Approximators." In: *Neural Networks* 2.5, pp. 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8.

Hsieh, Jun-Ting, Bingbin Liu, De-An Huang, Fei-Fei Li, and Juan Carlos Niebles (2018). "Learning to Decompose and Disentangle Representations for Video Prediction." In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, pp. 515–524. URL: https://proceedings.neurips.cc/paper/2018/hash/496e05e1aea0a9c4655800e8a7b9ea28-Abstract.html.

Hsu, Wei-Ning, Yu Zhang, and James R. Glass (2017). "Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 1878–1889. URL: http://papers.nips.cc/paper/6784-unsupervised-learning-of-disentangled-and-interpretable-representations-from-sequential-data.

Huang, Chin-Wei and Aaron Courville (2019). *Note on the Bias and Variance of Variational Inference*. arXiv: 1906.03708 `[cs, stat]`. URL: http://arxiv.org/abs/1906.03708.

Ilse, Maximilian, Jakub M. Tomczak, and Max Welling (2018). "Attention-Based Deep Multiple Instance Learning." In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 2132–2141. URL: http://proceedings.mlr.press/v80/ilse18a.html.

Ionides, Edward L (2008). "Truncated Importance Sampling." In: *Journal of Computational and Graphical Statistics* 17.2, pp. 295–311. DOI: 10.1198/106186008X320456.

Jaderberg, Max, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu (2015). "Spatial Transformer Networks." In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, pp. 2017–2025. URL: http://papers.nips.cc/paper/5854-spatial-transformer-networks.

Jordan, Michael I., Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul (1999). "An Introduction to Variational Methods for Graphical Models." In: *Machine Learning* 37.2, pp. 183–233. DOI: 10.1023/A:1007665907178.

Julier, Simon J. and Jeffrey K. Uhlmann (1997). "New Extension of the Kalman Filter to Nonlinear Systems." In: *Signal Processing, Sensor Fusion, and Target Recognition VI*. Signal Processing, Sensor Fusion, and Target Recognition VI. Vol. 3068. International Society for Optics and Photonics, pp. 182–193. DOI: 10.1117/12.280797.

– (2004). "Unscented Filtering and Nonlinear Estimation." In: *Proceedings of the IEEE* 92.3, pp. 401–422. ISSN: 1558-2256. DOI: 10.1109/JPROC.2003.823141.

Kahou, Samira Ebrahimi, Vincent Michalski, Roland Memisevic, Christopher Joseph Pal, and Pascal Vincent (2017). "RATM: Recurrent Attentive Tracking Model." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pp. 1613–1622. DOI: 10.1109/CVPRW.2017.206.

Kalman, Rudolph Emil (1960). "A New Approach to Linear Filtering and Prediction Problems." In: *Journal of Basic Engineering* 82.1, pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552.

Karl, Maximilian (2020). "Unsupervised Control." PhD thesis. Munich: Technische Universität München. URL: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20200331-1484075-1-4.

Karl, Maximilian, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt (2017a). "Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=HyTqHL5xg.

Karl, Maximilian, Maximilian Soelch, Philip Becker-Ehmck, Djalel Benbouzid, Patrick van der Smagt, and Justin Bayer (2017b). *Unsupervised Real-Time Control through Variational Empowerment*. arXiv: 1710.05101 [stat]. URL: http://arxiv.org/abs/1710.05101.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980.

Kingma, Diederik P., Tim Salimans, Rafal Józefowicz, Xi Chen, Ilya Sutskever, and Max Welling (2016). "Improving Variational Autoencoders with Inverse Autoregressive Flow." In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, pp. 4736–4744. URL: https://proceedings.

neurips.cc/paper/2016/hash/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Abstract.html.

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes." In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. URL: http://arxiv.org/abs/1312.6114.

Klushyn, Alexej, Nutan Chen, Richard Kurle, Botond Cseke, and Patrick van der Smagt (2019). "Learning Hierarchical Priors in VAEs." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 2866–2875. URL: http://papers.nips.cc/paper/8553-learning-hierarchical-priors-in-vaes.

Kobyzev, I., S. Prince, and M. Brubaker (2020). "Normalizing Flows: An Introduction and Review of Current Methods." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2020.2992934.

Kolmogorov, Andrei Nikolaevich (1957). "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition." In: *Doklady Akademii Nauk SSSR* 114, pp. 953–956. ISSN: 0002-3264. URL: https://zbmath.org/?q=an%3A0090.27103.

Kosiorek, Adam R., Alex Bewley, and Ingmar Posner (2017). "Hierarchical Attentive Recurrent Tracking." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 3053–3061. URL: https://proceedings.neurips.cc/paper/2017/hash/752d25a1f8dbfb2d656bac3094bfb81c-Abstract.html.

Kosiorek, Adam R., Hyunjik Kim, Yee Whye Teh, and Ingmar Posner (2018). "Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects." In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, pp. 8615–8625. URL: https://proceedings.neurips.cc/paper/2018/hash/7417744a2bac776fabe5a09b21c707a2-Abstract.html.

Krishnan, Rahul G., Uri Shalit, and David Sontag (2015). *Deep Kalman Filters*. arXiv: 1511.05121 [cs, stat]. URL: http://arxiv.org/abs/1511.05121.

Krishnan, Rahul G., Uri Shalit, and David A. Sontag (2017). "Structured Inference Networks for Nonlinear State Space Models." In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence,* *February 4-9, 2017, San Francisco, California, USA*. Ed. by Satinder P. Singh and Shaul Markovitch. AAAI Press, pp. 2101–2109. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14215.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a Meeting Held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, pp. 1106–1114. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

Kurle, Richard, Stephan Günnemann, and Patrick van der Smagt (2019). "Multi-Source Neural Variational Inference." In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, the Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, the Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, pp. 4114–4121. DOI: 10.1609/aaai.v33i01.33014114.

Lai, Guokun, Zihang Dai, Yiming Yang, and Shinjae Yoo (2019). "Re-Examination of the Role of Latent Variables in Sequence Modeling." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 7812–7822. URL: https://proceedings.neurips.cc/paper/2019/hash/d0ac1ed0c5cb9ecbca3d2496ec1ad984-Abstract.html.

Le, Tuan Anh, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood (2018). "Auto-Encoding Sequential Monte Carlo." In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=BJ8c3f-0b.

LeCun, Yann, Corinna Cortes, and CJ Burges (2010). "MNIST Handwritten Digit Database." In: *ATT Labs [Online]* 2. URL: http://yann.lecun.com/exdb/mnist.

Lee, Alex X., Anusha Nagabandi, Pieter Abbeel, and Sergey Levine (2020). "Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model." In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. URL: https://proceedings.neurips.cc/paper/2020/hash/08058bf500242562c0d031ff830ad094-Abstract.html.

Lee, Alex X., Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine (2018). *Stochastic Adversarial Video Prediction*. arXiv: 1804.01523 [cs]. URL: http://arxiv.org/abs/1804.01523.

Lee, Juho, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh (2019). "Set Transformer: A Framework for Attention-Based Permutation-Invariant Neural Networks." In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 3744–3753. URL: http://proceedings.mlr.press/v97/lee19d.html.

Li, Yingzhen and Stephan Mandt (2018). "Disentangled Sequential Autoencoder." In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 5656–5665. URL: http://proceedings.mlr.press/v80/yingzhen18a.html.

Li, Yingzhen and Richard E. Turner (2016). "Rényi Divergence Variational Inference." In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, pp. 1073–1081. URL: http://papers.nips.cc/paper/6208-renyi-divergence-variational-inference.

MacKay, David J. C. (2002). *Information Theory, Inference & Learning Algorithms*. USA: Cambridge University Press. ISBN: 978-0-521-64298-9.

Maddison, Chris J., Dieterich Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Whye Teh (2017). "Filtering Variational Objectives." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 6573–6583. URL: https://proceedings.neurips.cc/paper/2017/hash/fa84632d742f2729dc32ce8cb5d49733-Abstract.html.

Mandt, Stephan, James McInerney, Farhan Abrol, Rajesh Ranganath, and David M. Blei (2016). "Variational Tempering." In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*. Ed. by Arthur Gretton and Christian C. Robert. Vol. 51. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 704–712. URL: http://proceedings.mlr.press/v51/mandt16.html.

Mc Gee, L. A., S. F. Schmidt, and G. L. Smith (1962). *Application of Statistical Filter Theory to the Optimal Estimation of Position and Velocity* *on Board a Circumlunar Vehicle*. URL: http://archive.org/details/nasa_techdoc_19620006857.

Mirchev, Atanas, Baris Kayalibay, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer (2019). "Approximate Bayesian Inference in Spatial Environments." In: *Robotics: Science and Systems XV, University of Freiburg, Freiburg Im Breisgau, Germany, June 22-26, 2019*. Ed. by Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson. DOI: 10.15607/RSS.2019.XV.083.

Mnih, Andriy and Danilo Jimenez Rezende (2016). "Variational Inference for Monte Carlo Objectives." In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 2188–2196. URL: http://proceedings.mlr.press/v48/mnihb16.html.

Mohamed, Shakir (2015). *Machine Learning Trick of the Day (4): Reparameterisation Tricks*. URL: http://blog.shakirm.com/2015/10/machine-learning-trick-of-the-day-4-reparameterisation-tricks/.

Murphy, Kevin P. (2012). *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning Series. The MIT Press. ISBN: 978-0-262-01802-9.

Murphy, Ryan L., Balasubramaniam Srinivasan, Vinayak A. Rao, and Bruno Ribeiro (2019). "Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs." In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=BJluy2RcFm.

Naesseth, Christian A., Scott W. Linderman, Rajesh Ranganath, and David M. Blei (2018). "Variational Sequential Monte Carlo." In: *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*. Ed. by Amos J. Storkey and Fernando Pérez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, pp. 968–977. URL: http://proceedings.mlr.press/v84/naesseth18a.html.

Nayak, Pandu (2019). *Understanding Searches Better than Ever Before*. URL: https://blog.google/products/search/search-language-understanding-bert/.

Neiswanger, Willie, Frank D. Wood, and Eric P. Xing (2014). "The Dependent Dirichlet Process Mixture of Objects for Detection-Free Tracking and Object Modeling." In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*. Vol. 33. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 660–668. URL: http://proceedings.mlr.press/v33/neiswanger14.html.

Ning, Guanghan, Zhi Zhang, Chen Huang, Xiaobo Ren, Haohong Wang, Canhui Cai, and Zhihai He (2017). "Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking." In: *IEEE International Symposium on Circuits and Systems, ISCAS 2017, Baltimore, MD, USA, May 28-31, 2017*. IEEE, pp. 1–4. DOI: 10.1109/ISCAS.2017.8050867.

Nowozin, Sebastian (2018). "Debiasing Evidence Approximations: On Importance-Weighted Autoencoders and Jackknife Variational Inference." In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=HyZoi-WRb.

Owen, Art B. (2013). *Monte Carlo Theory, Methods and Examples*. URL: https://statweb.stanford.edu/~owen/mc/.

Papamakarios, George, Iain Murray, and Theo Pavlakou (2017). "Masked Autoregressive Flow for Density Estimation." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 2338–2347. URL: http://papers.nips.cc/paper/6828-masked-autoregressive-flow-for-density-estimation.

Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan (2019). *Normalizing Flows for Probabilistic Modeling and Inference*. arXiv: 1912.02762 [cs, stat]. URL: http://arxiv.org/abs/1912.02762.

Parisotto, Emilio, Devendra Singh Chaplot, Jian Zhang, and Ruslan Salakhutdinov (2018). "Global Pose Estimation with an Attention-Based Recurrent Network." In: *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, pp. 237–246. DOI: 10.1109/CVPRW.2018.00061.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 8024–8035. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Póczos, Barnabás, Aarti Singh, Alessandro Rinaldo, and Larry A. Wasserman (2013). "Distribution-Free Distribution Regression." In: *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*. Vol. 31. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 507–515. URL: http://proceedings.mlr.press/v31/poczos13a.html.

Pulford, G. W. (2005). “Taxonomy of Multiple Target Tracking Methods.” In: *Sonar and Navigation IEE Proceedings - Radar* 152.5, pp. 291–304. ISSN: 1350-2395. DOI: 10.1049/ip-rsn:20045064.

Qi, Charles Ruizhongtai, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas (2018). “Frustum PointNets for 3D Object Detection from RGB-D Data.” In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, pp. 918–927. DOI: 10.1109/CVPR.2018.00102.

Qi, Charles Ruizhongtai, Hao Su, Kaichun Mo, and Leonidas J. Guibas (2017a). “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.” In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pp. 77–85. DOI: 10.1109/CVPR.2017.16.

Qi, Charles Ruizhongtai, Li Yi, Hao Su, and Leonidas J. Guibas (2017b). “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.” In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 5099–5108. URL: https://proceedings.neurips.cc/paper/2017/hash/d8bf84be3800d12f74d8b05e9b89836f-Abstract.html.

Rainforth, Tom, Adam R. Kosiorek, Tuan Anh Le, Chris J. Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh (2018). “Tighter Variational Bounds Are Not Necessarily Better.” In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 4274–4282. URL: http://proceedings.mlr.press/v80/rainforth18b.html.

Ranganath, Rajesh, Sean Gerrish, and David M. Blei (2014). “Black Box Variational Inference.” In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*. Vol. 33. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 814–822. URL: http://proceedings.mlr.press/v33/ranganath14.html.

Rauch, H. E., F. Tung, and C. T. Striebel (1965). “Maximum Likelihood Estimates of Linear Dynamic Systems.” In: *AIAA Journal* 3.8, pp. 1445–1450. ISSN: 0001-1452. DOI: 10.2514/3.3166.

Ravanbakhsh, Siamak, Jeff Schneider, and Barnabas Poczos (2016). *Deep Learning with Sets and Point Clouds*. arXiv: 1611.04500 [cs, stat]. URL: http://arxiv.org/abs/1611.04500.

Reed, Scott E., Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee (2016). “Generative Adversarial Text to Image Synthesis.” In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June*

19-24, 2016. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1060–1069. URL: http://proceedings.mlr.press/v48/reed16.html.

Rezende, Danilo Jimenez and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows." In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Ed. by Francis R. Bach and David M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1530–1538. URL: http://proceedings.mlr.press/v37/rezende15.html.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models." In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. Vol. 32. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1278–1286. URL: http://proceedings.mlr.press/v32/rezende14.html.

Rezende, Danilo Jimenez and Fabio Viola (2018). *Taming VAEs*. arXiv: 1810.00597 [cs, stat]. URL: http://arxiv.org/abs/1810.00597.

Al-Rfou, Rami et al. (2016). *Theano: A Python Framework for Fast Computation of Mathematical Expressions*. arXiv: 1605.02688. URL: http://arxiv.org/abs/1605.02688.

Santoro, Adam, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Tim Lillicrap (2017). "A Simple Neural Network Module for Relational Reasoning." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 4967–4976. URL: https://proceedings.neurips.cc/paper/2017/hash/e6acf4b0f69f6f6e60e9a815938aa1ff-Abstract.html.

Sarkka, Simo (2013). *Bayesian Filtering and Smoothing*. Cambridge: Cambridge University Press. ISBN: 978-1-139-34420-3. DOI: 10.1017/CBO9781139344203.

Schmidhuber, Jürgen (1992). "Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks." In: *Neural Computation* 4.1, pp. 131–139. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.1.131.

– (2017). *History of Computer Vision Contests Won by Deep CNNs on GPU*. URL: http://people.idsia.ch/~juergen/computer-vision-contests-won-by-gpu-cnns.html.

Schulter, Samuel, Paul Vernaza, Wongun Choi, and Manmohan Chandraker (2017). "Deep Network Flow for Multi-Object Tracking." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pp. 2730–2739. DOI: 10.1109/CVPR.2017.292.

Schuster, Mike and Kuldip K. Paliwal (1997). "Bidirectional Recurrent Neural Networks." In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681. DOI: 10.1109/78.650093.

Schwartz, Barry (2020). *Google: BERT Now Used on Almost Every English Query*. URL: https://searchengineland.com/google-bert-used-on-almost-every-english-query-342193.

Silver, David et al. (2016). "Mastering the Game of Go with Deep Neural Networks and Tree Search." In: *Nature* 529.7587 (7587), pp. 484–489. ISSN: 1476-4687. DOI: 10.1038/nature16961.

Soelch, Maximilian, Adnan Akhundov, Patrick van der Smagt, and Justin Bayer (2019). "On Deep Set Learning and the Choice of Aggregations." In: *Artificial Neural Networks and Machine Learning - ICANN 2019: Theoretical Neural Computation - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part I*. Ed. by Igor V. Tetko, Vera Kurková, Pavel Karpov, and Fabian J. Theis. Vol. 11727. Lecture Notes in Computer Science. Springer, pp. 444–457. ISBN: 978-3-030-30487-4. DOI: 10.1007/978-3-030-30487-4\\_35.

Soelch, Maximilian, Justin Bayer, Marvin Ludersdorfer, and Patrick van der Smagt (2016). *Variational Inference for On-Line Anomaly Detection in High-Dimensional Time Series*. arXiv: 1602.07109 [cs, stat]. URL: http://arxiv.org/abs/1602.07109.

Sønderby, Casper Kaae, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther (2016). "Ladder Variational Autoencoders." In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, pp. 3738–3746. URL: http://papers.nips.cc/paper/6275-ladder-variational-autoencoders.

Song, Fuijan et al. (2010). "Dissemination and Publication of Research Findings: An Updated Review of Related Biases." In: *Health Technology Assessment (Winchester, England)* 14.8, pp. iii, ix–xi, 1–193. ISSN: 2046-4924. DOI: 10.3310/hta14080.

Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhutdinov (2015). "Unsupervised Learning of Video Representations Using LSTMs." In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Ed. by Francis R. Bach and David M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 843–852. URL: http://proceedings.mlr.press/v37/srivastava15.html.

Steenkiste, Sjoerd van, Michael Chang, Klaus Greff, and Jürgen Schmidhuber (2018). "Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and Their Interactions." In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceed-*

ings. OpenReview.net. URL: https://openreview.net/forum?id=ryH20GbRW.

Su, Jinsong, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang (2018). "Variational Recurrent Neural Machine Translation." In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, pp. 5488–5495. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16791.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks." In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, pp. 3104–3112. URL: https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "MuJoCo: A Physics Engine for Model-Based Control." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*. IEEE, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

Turner, Richard Eric and Maneesh Sahani (2011). "Two Problems with Variational Expectation Maximisation for Time Series Models." In: *Bayesian Time Series Models*. Ed. by A. Taylan Cemgil, David Barber, and Silvia Chiappa. Cambridge: Cambridge University Press, pp. 104–124. ISBN: 978-0-521-19676-5. DOI: 10.1017/CBO9780511984679.006.

Vinyals, Oriol, Samy Bengio, and Manjunath Kudlur (2016). "Order Matters: Sequence to Sequence for Sets." In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1511.06391.

Wagstaff, Edward, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne (2019). "On the Limitations of Representing Functions on Sets." In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 6487–6494. URL: http://proceedings.mlr.press/v97/wagstaff19a.html.

Wang, Yue, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon (2019). "Dynamic Graph CNN for Learning on Point Clouds." In: *ACM Trans. Graph.* 38.5, 146:1–146:12. DOI: 10.1145/3326362.

Watter, Manuel, Jost Tobias Springenberg, Joschka Boedecker, and Martin A. Riedmiller (2015). "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images." In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, pp. 2746–2754. URL: https://proceedings.neurips.cc/paper/2015/hash/a1afc58c6ca9540d057299ec3016d726-Abstract.html.

Welling, Max (2007). "Product of Experts." In: *Scholarpedia* 2.10, p. 3879. ISSN: 1941-6016. DOI: 10.4249/scholarpedia.3879.

Welzl, Emo (1991). "Smallest Enclosing Disks (Balls and Ellipsoids)." In: *New Results and New Trends in Computer Science, Graz, Austria, June 20-21, 1991, Proceedings [on Occasion of h. Maurer's 50th Birthday]*. Ed. by Hermann A. Maurer. Vol. 555. Lecture Notes in Computer Science. Springer, pp. 359–370. DOI: 10.1007/BFb0038202.

Wu, Zhirong, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao (2015). "3D ShapeNets: A Deep Representation for Volumetric Shapes." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, pp. 1912–1920. DOI: 10.1109/CVPR.2015.7298801.

Yang, Nan, Rui Wang, Jörg Stückler, and Daniel Cremers (2018). "Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry." In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss. Vol. 11212. Lecture Notes in Computer Science. Springer, pp. 835–852. DOI: 10.1007/978-3-030-01237-3\\_50.

Yi, Li, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J. Guibas (2019). "GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, pp. 3947–3956. DOI: 10.1109/CVPR.2019.00407.

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola (2017). "Deep Sets." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 3391–3401. URL: https://proceedings.neurips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html.