TUM School of Computation, Information and Technology
Technische Universität München

Deep Learning for Clinical Decision Support Systems in Chest Radiography

Alessandro Benjamin Wollek

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

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr. rer. nat. Georg Groh

Prüfer der Dissertation:

1. Priv.-Doz. Dr. rer. nat. Tobias Lasser
2. Prof. Dr. rer. nat. Michael Ingrisch,
   LMU München
3. Prof. Philippe Després, Ph.D.,
   Université Laval Québec

Die Dissertation wurde am 14.06.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 25.09.2023 angenommen.

Acknowledgments

Starting my doctoral studies in 2020 during the beginning of the COVID-19 pandemic fully remotely was rocky but thanks to all the people around me I was able to quickly adapt to the field of radiology and grow into our team. I would like to take this opportunity to thank some of the people that supported me through this journey explicitly.

First, I am very grateful for the support by my supervisor Tobias Lasser. His trust in my abilities, even when I felt discouraged after heading down another dead end, his thorough feedback during our meetings, and his advice were essential for making the work presented in this document possible. Thank you. Also, I would like to thank my mentor Georg Groh for his support and teaching me so much about NLP.

I would like to thank our team at the computational imaging and inverse problems group: Jonas for the Emacs configuration procrastination sessions and C++ discussions; David for the Emacs vs. VIM debates; Josué for responding to my guild.ai questions on the respective forum; Theo for discussions on computer vision; Erdal for our career planning conversations. Thank you all for the inspiring discussions, projects, and events we had throughout these years. Also, thank you John for always being up for a coffee break and discussing the different forms of attention and dropout. Thank you Theresa, not only for our collaboration but also for your support. I would also like to thank our partners at the LMU Klinikum: Sardi, Melia, Saša, Theresa, Bastian, and Michael for their support throughout our project, all these studies were not possible without you. Overall, I do not even want to imagine enduring a Ph.D. without such colleagues and collaborators, thank you all!

This whole academic journey would have not been possible without my family. I am lucky and deeply grateful to have such loving parents and siblings. Thank you for being there for me and supporting/sopportare me.

Abstract

Chest radiography is the most commonly administered imaging modality world wide today. It enables radiologists to quickly screen for various pathologies with comparatively small exposure to ionizing radiation. However, due to their usefulness in clinical practice, more images are being acquired than radiologists can evaluate, resulting in unprocessed queues.

Over the past decade, with the raise of fast parallel computing hardware and large image data sets, recent deep learning-based image recognition models have shown impressive results in tackling image-based tasks. Similarly, text-processing deep learning models have come a long way from rudimentary text completion to multi-domain chatbots in the past years. Consequently, in recent years, the research community has been investigating the use of deep learning-based models for automatic disease classification to assist radiologists and improve patient care.

In this dissertation, crucial components of chest radiography processing pipelines were investigated, from automatic anonymization and annotation of thoracic radiology reports to the processing of chest radiographs for pathology classification using deep neural networks. To increase the size of available chest X-ray data sets, this works presents two methods for automatic thoracic radiology report annotation, a rule-based and a deep learning-based approach, and an interface for manual creation of a reference standard. To simplify cross-institutional data sharing, an anonymization algorithm for German radiology reports is presented. Furthermore, the effect of image resolution and the application of windowing on image classification performance is investigated. Additionally, the feasibility of vision transformers for chest X-ray classification is shown, and an attention-based interpretability method is tested both quantitatively and qualitatively. Finally, the effect of other radiographs, so-called out-of-distribution images, on chest X-ray classification models is studied and a method for improving robustness against such images is proposed.

Zusammenfassung

Die Thoraxradiographie ist die am häufigsten verwendete bildgebende Modalität weltweit. Sie ermöglicht es Radiologen schnell nach verschiedenen Pathologien zu suchen und dabei vergleichsweise wenig ionisierende Strahlung einzusetzen. Aufgrund ihrer Nützlichkeit in der klinischen Praxis werden jedoch mehr Röntgenbilder erstellt, als von Radiologen ausgewertet werden können, was zu unverarbeiteten Warteschlangen führt.

In den letzten zehn Jahren haben sich aufgrund der zunehmenden Verbreitung von leistungsstarken Grafikkarten und großer Bilddatensätze neuere, auf Deep Learning basierende Bilderkennungsmodelle als wirksames Mittel zur Lösung bildbasierter Aufgaben erwiesen. Ähnlich haben sich Textverarbeitungsmodelle auf Basis von Deep Learning in den letzten Jahren stark weiterentwickelt. Infolgedessen haben Forscher in den letzten Jahren vermehrt untersucht, wie auf Deep Learning basierende Modelle zur automatischen Klassifizierung von Krankheiten eingesetzt werden können, um Radiologen bei der Patientenversorgung zu unterstützen.

In dieser Dissertation wurden essentielle Komponenten der automatischen Verarbeitung von Thoraxröntgenbildern untersucht, von der automatischen Anonymisierung und Annotation von Thorax-Radiologiebefunden bis zur Verarbeitung von Thoraxröntgenbildern zur Pathologieklassifikation mit Hilfe von Deep-Neural-Networks. Um die Größe der verfügbaren Datensätze zu erhöhen, werden in dieser Arbeit zwei Methoden für die automatische Befundannotation vorgestellt – eine regelbasierte und eine Deep-Learning-basierte Methode sowie eine Anwendung zur manuellen Erstellung eines Referenzstandards. Um den klinikübergreifenden Datenaustausch zu vereinfachen, wird ein Anonymisierungsalgorithmus für deutsche Radiologiebefunde vorgestellt. Darüber hinaus wird die Auswirkung von Bildauflösung und Anwendung von Fensterung auf die Güte der Bildklassifikation untersucht. Außerdem wird die Machbarkeit von Vision-Transformern für die Röntgenthoraxklassifikation gezeigt und eine auf Aufmerksamkeit basierende Interpretierbarkeitsmethode sowohl quantitativ als auch qualitativ getestet. Schließlich wird die Wirkung von anderen Röntgenaufnahmen, sogenannten Out-of-Distribution-Bildern, auf Röntgenthoraxklassifikationsmodelle untersucht und eine Methode zur Verbesserung der Robustheit gegenüber solchen Bildern vorgeschlagen.

Contents

List of Figures

List of Tables

Part I

Introduction

1 Introduction

1.1 On Medical Imaging

Medical imaging, starting with the discovery of X-rays in 1895 [1], has revolutionized medical diagnostics by enabling the visualization of internal structures of the human body for clinical analysis and medical intervention [2]. X-ray imaging provides a non-invasive procedure for inspecting the human body, enabling faster and more precise diagnoses to treat medical conditions. Before the advent of medical imaging, doctors had to rely mostly on patient-reported symptoms and recently invented rudimentary diagnostic tools such as stethoscopes [3]. When the participants of the German medical-physical society witnessed Röntgen's discovery in January 1896, they were quickly convinced of its medical benefits for medical diagnoses [4]. Even one of the earliest X-ray images, a radiograph of Anna Bertha Röntgen's hand (see Figure 1.1), provided a clear depiction of the bone structure of the hand. This image exemplifies the immediate medical usefulness of medical imaging. Today, medical imaging is an essential part of modern medicine, with different imaging techniques and acquisition methods such as X-ray imaging, computed tomography (CT), magnetic resonance imaging (MRI), or ultrasound, used to visualize the human body. This dissertation focuses on the most frequently performed imaging modality, chest radiography [5, 6].

Figure 1.1: X-ray image captured by Wilhelm Conrad Röntgen in 1895, depicting the hand of his wife Anna Bertha Röntgen. It is one of the earliest examples of X-ray imaging, demonstrating its potential for medical diagnostics by revealing the bones of the hand.

1.2 Introduction to Chest Radiography

X-rays are a type of electromagnetic radiation that was first discovered by Wilhelm Conrad Röntgen in 1895, known as "Röntgenstrahlen" in German [1]. During his experiments with cathode ray tubes, Röntgen observed that X-rays could penetrate materials

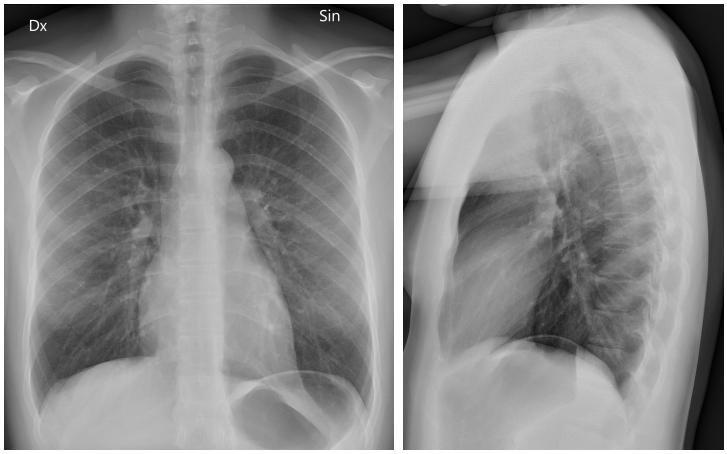

Figure 1.2: Frontal (PA) and lateral chest X-ray.

that visible light could not, such as wood and books. He also discovered that X-rays were partially absorbed by materials of varying densities. The medical potential of X-rays was immediately apparent, as they could penetrate soft tissues while being absorbed by denser structures like teeth and bones. The use of X-rays for imaging, known as radiography, revolutionized medicine by providing a low-cost and efficient imaging modality that has become the most frequently used modality in medical imaging [5, 6].

Chest X-rays, as depicted in Figure 1.2, are commonly used to diagnose and screen for various symptoms and medical conditions, such as pneumonia, heart failure, and pneumothorax, as well as to locate medical devices [7]. In this imaging modality, the patient is positioned between an X-ray source and a detector, and an image is acquired by directing the X-ray beam through the body. Typically, both a frontal and a lateral image are acquired, as seen in Figure 1.2. In the frontal view, the patient stands facing the detector (posterior-anterior, PA), but if this is not possible, they are positioned facing the source (anterior-posterior, AP). In some cases, they may be required to lie on their back (supine) or stomach (prone), for example, in intensive care. The simplicity of this technique results in a cheap and fast imaging modality. It is important to note that chest radiographs involve a low dose of ionizing radiation, which has potential harmful effects over prolonged exposure, equivalent to 28 days of natural radiation exposure [8], making them a low-risk procedure.

1.3 Decision Support Systems in Radiology

Clinical decision support systems (CDSS) are computer applications designed to aid clinicians in making diagnostic and therapeutic decisions in patient care [9]. CDSS can alert clinicians of potential medication interactions, provide access to patient health records, and assist in interpreting medical images more accurately [10, 11]. CDSS also have applications in order scheduling [12], computer-aided diagnosis [13, 14], or worklist prioritization [15]. They can complement radiologists in interpreting medical images, potentially leading to improved diagnostic accuracy and faster review times [16, 17, 14].

Other examples of CDSS in radiology are early detection of time critical diseases, where a CDSS can prioritize incoming images for faster review [15, 18, 19] or assisting in the training of radiologists [20].

The presented use cases are not just potential applications but many are already applied in the clinical routine. Today, radiologists are using speech recognition software for faster report generation and image diagnosis systems. In the US alone, over 300 medical devices using artificial intelligence (AI) have been approved by the Food and Drug Administration (FDA), with growing numbers year-over-year [21, 22][1]. We can summarize the potential impact of (AI-based) CDSS as Curtis P. Langlotz phrased it [23]: "Radiologists who use AI will replace radiologists who don't."

1.4 Why do we need AI in radiology?

This dissertation focuses on the use of deep neural networks for chest radiographs and corresponding radiology reports. While the previous section advocated for CDSS to improve diagnoses and clinical practice, the primary motivation for embracing AI-based decision support systems in radiology is the global shortage of radiologists. In the United States, the number of medical students becoming radiologists has decreased [24] and those who do move to urban counties, leaving rural areas understaffed [25]. The consequences are delays and backlogs for image diagnosis, reducing the quality of care, and risking patient health. These problems are a global phenomenon, reported by, for example the U.K. National Health Service [26] or the U.S. Department of Veterans Affairs [27]. This shortage is even worse in resource-poor countries, such as Rwanda [28] or Liberia [29]. Here, AI-based decision support systems can not only augment the work of radiologists by providing more efficient diagnoses but ease the stress of workforce shortages. As deep learning-based models continuously prove to excel in computer vision [30], they are becoming a versatile backbone for radiology software [31, 11].

1.5 Outline

This dissertation focuses on the data pipeline necessary for applying deep learning on chest radiographs for aiding radiologists in their diagnoses. Chapter 2 introduces the basic concepts of deep learning and the general computer vision architectures used throughout this thesis. Besides introducing the theoretical deep learning aspects, it explains the implementation of deep learning models in PyTorch and PyTorch Lightning and the development using `guild.ai`. Chapter 3 introduces recent deep learning advances for chest X-ray interpretation and presents publicly available chest X-ray data sets. Chapter 4 proposes methods to anonymize and extract radiological labels from German free text reports for training deep learning models. Chapters 5 and 6 analyze the effect of image windowing and resolution on chest X-ray classification performance, respectively. Chapter 7 introduces a method to improve interpretability of chest X-ray classifiers that

[1] `https://www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-and-machine-learning-aiml-enabled-medical-devices`

helps radiologists judge a model's prediction. Chapter 8 tackles the problem of incorrect input data and how chest X-ray classifiers can be augmented to become more robust and less prone to outliers. Chapter 9 discusses deep learning for clinical decision support in radiology and finally, Chapter 10 concludes this doctoral thesis.

2 Concepts of Deep Learning

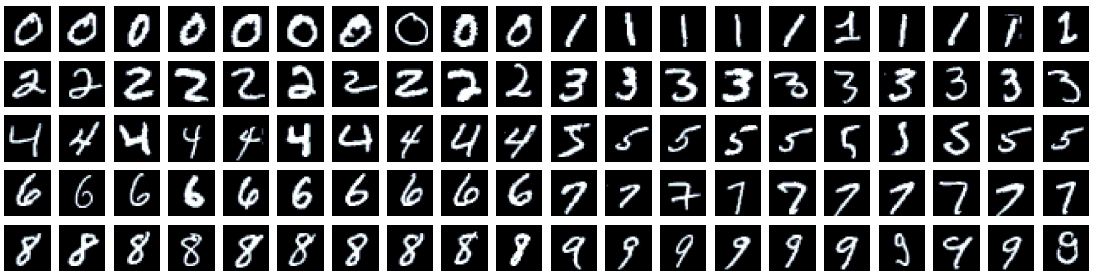

Figure 2.1: Handwritten digits of the MNIST data set [32]. While every digit has a distinct shape, each written digit displays a variation. For example, the number seven is written with and without a horizontal middle-line.

2.1 Introduction

In 1966, several undergraduate students at the Massachusetts Institute of Technology (MIT) were tasked in the summer vision project of the Artificial Intelligence Group to "construct a significant part of a visual system" [33]. With the primary goal to segment an image into objects, background area, and "chaos". This problem was not solved during that summer[1]. While various computer vision algorithms have been developed in the past 50 years[2], recent works show that many vision problems can be practically solved using deep neural networks [35].

In this chapter, we lay the foundation on how to derive a prediction from an image, for example, how to recognize the digits in Figure 2.1. This chapter begins with the basics of neural networks, covers PyTorch and PyTorch Lightning, and finally introduces the tooling required for experimentation and prototyping. A deeper review of the mathematical foundations of deep neural networks is provided in [36] and [37]. Good references for the broader field of machine learning are Kevin Murphy's [38, 39] and Bishop's textbooks [40].

[1] Relevant xkcd: `https://xkcd.com/1425`. This web comic was published in 2014. Five years later, such an application is completely possible.

[2] An introduction to computer vision algorithms is presented in [34].

Deep neural networks are the first method to scale to various tasks on unstructured data, such as images[3], across applications [42]. The main difficulty of unstructured data, for example, an image, lies in converting them into a useful representation. In other words, encoding the input as a vector in an effective way is difficult. Given a two-dimensional array, representing black and white pixels of images of handwritten digits (see Figure 2.1). The question is how to identify the digit using a computer program.

Intuitively, we know that each number has a specific shape. The challenge is, that these shapes are similar only in an abstract fashion. It is non trivial to specify the characteristics of each digit in code. What is simple, however, is to provide examples for each class. Shifting the problem from specifying the characteristics of a class to "learning" a representation.

The main difference between conventional programming and deep learning is that a programmer specifies how a task is performed using an algorithm, whereas in deep learning we provide examples of a (related) task and the model learns the details using guided feedback. For example, if we wish to train a model to identify handwritten numbers we could set up a training data set with examples for each digit. The input would be an image of the digit, see Figure 2.1, and the output the probability of each digit. As only one digit is shown per image, we could enforce that the sum of all predicted digit probabilities must equal 1 and the expected prediction should be a 100 % confidence for the displayed digit. In the following sections, we will discuss how to convert an image of a digit to a digital number using deep learning using the MNIST data set [32] as an example.

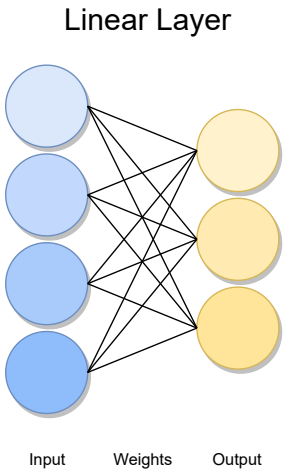

Figure 2.2: A linear layer is one of the fundamental building blocks of many modern neural networks. Mathematically, it is a matrix multiplication of the input by a weight matrix.

2.2 Deep Learning Essentials

2.2.1 Basic Building Blocks

This section provides an overview of the components used in the deep learning architectures discussed in the second part of this dissertation, which includes both convolutional neural networks and vision transformers. As most deep learning architectures use similar components with varying configurations, this section describes many elements used by other deep learning architectures in computer vision. As these models are primarily implemented in PyTorch, a more extensive list of deep learning building blocks can be found in the official PyTorch documentation[4].

[3] However, a neural network is not necessarily the best method for every use case. For example, tabular data have remained a challenge [41].

[4] `https://pytorch.org/docs/stable/nn.html`

2.2.1.1 Linear Layer

One of the basic building blocks of neural networks are simple matrix multiplications, called linear, fully-connected or dense layers. In a fully-connected layer every input neuron is connected to an output neuron, as seen in Figure 2.2. The importance of each connection, called weights, form the (weight) matrix. To determine the output of a fully-connected layer we multiply the input vector by the weight matrix. Additionally, a bias term is added to the output. The bias term allows for modeling a shift in the output and can be used as additional parameter to improve performance.

Mathematically, a deep neural network can be defined as:

$$\hat{y} = f(x, \theta).$$

Here, x represents the input, θ the trainable parameters, $\hat{y}$ the prediction, and f the model architecture. A model consisting of only a linear layer can be defined as:

$$\hat{y} = xW^T + b,$$

where W and b represent the trainable weights and bias. In PyTorch, a linear layer can be used as follows:

```
import torch
import torch.nn as nn

x = torch.randn(20) # random tensor
fc = nn.Linear(in_features=20, out_features=30, bias=True)
y = fc(x)
```

Here, the `in_features` and `out_features` define the input and output dimensions.

Before training the model, the parameters require an initial value (initialization). Empirically, it has been shown that the initialization significantly affects training performance [43]. In PyTorch, the weights are initialized randomly, drawn from a uniform distribution $\mathcal{U}(\sqrt{-1/n}, \sqrt{1/n})$, where n is the number of input dimensions [44]. This choice of initialization was developed for a specific activation function, the rectified linear unit (ReLU), see Section 2.2.1.2.

2.2.1.2 Activation Functions

To model non-linear relationships between input and output, non-linear functions, called activation functions, are applied to the output of a layer. The most commonly used activation functions are presented below. The combination of multiple layers with activations in between enables deep neural networks to model more complex functions and solve problems such as the XOR problem[5] [36, pp. 167]. For a comprehensive list of current activation functions see the official PyTorch documentation[6].

[5] For a description of the limitations of single-layer neural networks, see [45, p. 189].

[6] `https://pytorch.org/docs/stable/nn.html#non-linear-activations-weighted-sum-nonlinearity`

Sigmoid Function The sigmoid function, shown in Figure 2.3, is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}},\ x \in \mathbb{R},$$

and maps a real input value to a value in the range between 0 and 1. This makes the function suitable for representing the likelihood of binary classification problems.

The sigmoid function is commonly used for binary classification problems, where the classification problem can be expressed as "yes or no" or "present or not present". For example, we can model a neural network that should predict if a chest X-ray shows signs of a pneumothorax, a critical lung disease, using a sigmoid function as final step, representing the likelihood of a pneumothorax.

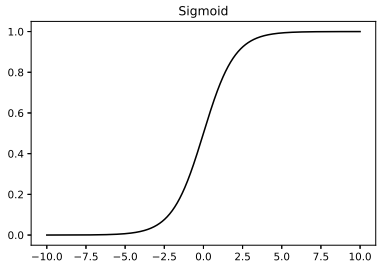

Figure 2.3: The "s-shaped" sigmoid function maps the real numbers to the range between 0 and 1.

While the output of a sigmoid function can be interpreted as probability, it is not necessary the case that a value of 0.5 means the model will be correct 50 % of the time. However, there are several transformations to solve this problem. Further details regarding such calibration can be found, for example, in [46].

Softmax Function The extension of the sigmoid function to classification problems with multiple classes is the softmax function. Given a vector $\mathbf{x} \in \mathbb{R}^n$, where n is the number of classes, the softmax function

$$\sigma(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

maps every component of $\mathbf{x} \in \mathbb{R}^n$ to the interval $(0, 1)$ and normalizes them so that $\sum_{i=1}^{n} \sigma(\mathbf{x})_i = 1$. Therefore, similarly to the sigmoid, the output of the softmax can be interpreted as probability distribution over all n classes. The softmax function magnifies the probability of a single class due to the exponential operation in the formula. This makes the softmax function suitable for problems where a single class is correct, i.e., multi-class classification. For example, predicting the digit visible in an MNIST image.

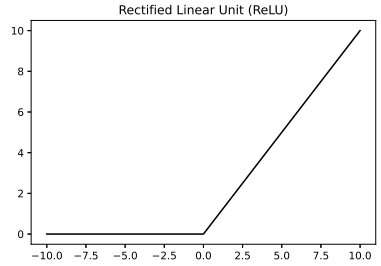

Figure 2.4: The rectified linear unit (ReLU) function is used to convert negative numbers to zero.

Rectified Linear Unit A computationally more efficient activation function without the need for exponential calculations is the rectified linear unit (ReLU) [47, 48, 30]. It is defined as

$$f(x) = \max\{0, x\},$$

as illustrated in Figure 2.4. The ReLU activation is commonly used in between layers. One significant advantage of ReLU, compared to the sigmoid function, is that it does not suffer from the vanishing gradient problem [36, p. 191], where the gradient of large or small inputs becomes close to zero, causing the learning process to slow down [48]. This is because the gradient of the ReLU is one for positive values, and thus, it does not have a saturation effect. However, as negative values have a gradient of zero, which stops the learning process, modifications have been proposed [49, 44].

2.2.1.3 Learning

Using linear layers and activation functions, we can finally create a simple one-layer neural network with a hidden layer[7]. In PyTorch, we can define our one-layer neural network as follows:

```
class OneLayerNN(nn.Module):

    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, digit):
        out = self.fc1(digit)
        hidden = nn.functional.relu(out)
        out = self.fc2(hidden)
        return out
```

This model expects a vector of dimension 784 as input and returns a ten dimensional vector. Each of the 784 dimensions represents a single pixel of a 28×28 pixel MNIST image, and each of the ten output dimensions represents a digit. In the following, we cover how to train such a neural network to predict the correct digit[8].

Loss Functions To train the one-layer neural network, we measure the discrepancy between the model's actual and the expected prediction. This error term is used to update the model parameters using back-propagation (Section 2.2.1.3) and an optimization algorithm (Section 2.2.1.3). The error, or "loss", is calculated using a loss function. The choice depends on the training task and the model output. The most commonly used loss functions for classification problems are presented next.

Cross-Entropy For multi-class image classification, where an image displays a single class out of multiple possibilities, a common choice is the negative log likelihood loss, defined as the negative sum of all predictions. To form a prediction, the model's output is passed through a softmax layer and then the logarithm is applied to the predicted

[7] Hidden, because the output of the layer is not the model's output.

[8] For a visual mathematical introduction to deep learning, I recommend 3B1B's video series `https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi`

probability of the expected class. For computational efficiency, softmax and negative log likelihood are often combined as so-called cross-entropy loss.

Binary Cross-Entropy For binary or multi-label prediction problems, for example, when an image can be assigned to multiple classes, we often use a cross-entropy variant called binary cross-entropy. In this case, the target is a binary vector with each element representing the probability of the class. For computational efficiency, the sigmoid and the negative log likelihood are combined. The formula for the n^{th} class given the output of the model x_n and the binary label y_n is

$$l_n = -\left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n))\right].$$

Gradient Descent How does a deep neural network actually learn? Typically, the task is to minimize the error measured by the loss function, as it represents the difference between the prediction and the desired outcome. That means we want to adjust the weights and biases so that the network's predictions become more accurate. In other words, how do we have to modify the weights to minimize the error the most? The negative gradient points in the direction of steepest descent, which gives us the fastest improvement in the loss function for a small change in the weights and biases[9]. Therefore, we obtain the best update of our weights by subtracting the respective gradient, so-called gradient descent [50]. Note, however, that we do not know for how much we should follow the direction. This degree is conventionally called "learning rate" and is one of several "hyper-parameters" that can be optimized before the training [10].

As data sets are large and do not necessarily fit into memory, a neural network is typically trained using batches of the training data. Like a random sample can be considered an estimate of the data set, the gradient of a batch can be considered an estimate of the complete gradient. Updating the model's weights according to the approximate gradient based on batches is called stochastic gradient descent (SGD)[11]. The algorithm can be written in pseudo-code as follows:

```
for batch in batches:
    images, labels = batch
    predictions = model(images)
    loss = loss_function(predictions, labels)
    gradients = compute_gradient(model, loss)
    model.weights -= learning_rate * mean(gradients)
```

Back-Propagation In 1986, Rumelhart, Hinton, and Williams proposed the back-propagation algorithm to train neural networks [51]. The weights of the model are

[9] A beautiful visual explanation narrated by 3B1B can be found at `https://www.youtube.com/watch?v=_-02ze7tf08`

[10] Another hyper-parameter could be, for example, the size of the hidden layer of our network.

[11] "True" SGD uses only a single sample to estimate the gradient. However, passing multiple samples in parallel is computationally more efficient on the GPU, and the average of their gradients provides a more stable gradient [36, pp. 274].

updated according to their partial derivatives with respect to the error term calculated by the loss function using gradient descent. This can be implemented efficiently by first calculating the partial derivatives of the last neurons with respect to the loss and then work backwards using the chain rule for the preceding neurons.

One of the major features of PyTorch is the automatic differentiation or "autograd" module[12]. By defining the forward (function) and backward pass (partial gradient) for a single layer, PyTorch can automatically apply the chain rule to calculate the partial gradients for a respective layer in a model. As most elementary functions and common layers, including their backward passes, are already defined, in practice, they can be combined to form more complex functions without the need to define the backward pass explicitly. However, we could define, for example, a linear layer as follows:

```
class Linear(torch.autograd.Function):
    @staticmethod
    def forward(ctx, input, weight, bias):
        ctx.save_for_backward(input, weight, bias)
        return torch.matmul(input, weight.t()) + bias

    @staticmethod
    def backward(ctx, grad_output):
        input, weight, bias = ctx.saved_tensors
        grad_input = torch.matmul(grad_output, weight)
        grad_weight = torch.matmul(grad_output.t(), input)
        grad_bias = torch.sum(grad_output, dim=0)
        return grad_input, grad_weight, grad_bias
```

Mathematically, the forward pass of the linear layer is

$$\hat{y} = W^\top x + b$$

for a single input x, defined in the `forward` method.

For the backward pass, we need the propagated gradient from the loss (`grad_output` in PyTorch) to calculate the gradient for each part of the layer:

$$\begin{aligned}
\frac{\partial L}{\partial x} &= W \frac{\partial L}{\partial \hat{y}} \\
\frac{\partial L}{\partial W} &= x \frac{\partial L}{\partial \hat{y}}^\top \\
\frac{\partial L}{\partial b} &= \frac{\partial L}{\partial \hat{y}}^\top \mathbf{1},
\end{aligned}$$

where L is the loss and $\frac{\partial L}{\partial \hat{y}}$ the gradient with respect to the prediction. Since we can combine layers using the chain rule, we can abstract this loss as the gradient with respect to the output, `grad_output`[13]. The gradient with respect to the input $\frac{\partial L}{\partial x}$ becomes the `grad_output` for the preceding layer.

[12] `https://pytorch.org/docs/stable/autograd.html`

[13] A complete step-by-step derivation can be found at `https://web.archive.org/web/20230228075518/http://cs231n.stanford.edu/handouts/linear-backprop.pdf`

ADAM Using the components presented in the previous section, we are able to finally train our neural network. To speed-up model training, we can utilize a more sophisticated optimization algorithm. Practically, there are several details to be considered for improving the basic SGD algorithm, for example, how should we choose the learning rate? The negative gradient itself gives only the direction of steepest descent in the close neighborhood, not necessarily the direction to the minimum. Following the steepest descent too quickly by multiplying the negative gradient with a large scalar could lead to overshooting but a tiny learning rate could lead to unnecessary long training times.

One improvement to a fixed learning rate, is to follow a direction with larger steps if the gradients keep pointing towards the same direction, similar to a ball gaining momentum [52]. Additionally, adapting the learning rate for each weight independently instead of using a global learning rate improves convergence[14].

A more recent, popular algorithm for faster gradient-based optimization is ADAM (Adaptive Moment Estimation) [53][15]. It uses exponential moving averages of the gradient and squared gradient as estimates of its mean and variance for faster convergence. The default ADAM implementation in PyTorch is based on AdamW [54] combining ADAM with L2 regularization for more stable training.

Finally, combining model and optimization algorithm we can train our neural network[16]. First, we initialize our one-layer neural network, and ADAM with the model's parameters (weights) and specify the learning rate (`lr=1e-3`). Next, we shuffle the training data to avoid learning just the sequence of labels. Then, we pass a batch of images to the model and calculate the cross-entropy loss of predicted and actual labels. As PyTorch accumulates the gradients by default, we clear the stored gradients before calculating the partial gradients in the backwards pass. By calling the optimizer, we update the model's weights. In other words, the model learns. The following code implements a bare-bone training procedure. To improve the results, we iterate over the training data set multiple times. One iteration over all training data is called an epoch.

```
one_layer_nn = OneLayerNN()  # initialize model
# initialize optimizer
optim = torch.optim.Adam(params=one_layer_nn.parameters(), lr=1e-3)
shuffled_order = torch.randperm(len(train))  # shuffle training data
for i in range(len(shuffled_order) // batch_size):
      batch = train[shuffled_order[i * batch_size:(i + 1) * batch_size]]
      y_true = batch[::batch_size]
      y_pred = one_layer_nn(batch[:batch_size:])
      loss = torch.nn.functional.cross_entropy(y_pred, y_true)
      optim.zero_grad()  # do not accumulate gradients
      loss.backward()  # perform backward pass
      optim.step()  # update the model weights
```

To avoid memorizing concepts that do not generalize to all handwritten digits (overfitting), we split the training data into training and test data. To get an even better

[14] Hinton's lecture on RMSProp (unpublished): `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`

[15] Other gradient-based optimization algorithms are presented in [36, Sections 8.3 - 8.6].

[16] Complete notebook available at `https://www.kaggle.com/code/fold10/mnist-1-layer-nn`

estimate of how good the model would perform on unseen data, we further split the training data into training and validation data. During our experiments, we monitor the results on the validation data. The predictions on the validation data are not incorporated into the training procedure, thereby simulating unseen data for the model. However, as the development choices are made on the validation results, the developer of the model "overfits" onto the validation set. Therefore, only after completing the development of the model it is evaluated on the test set. On the MNIST data set, the one-layer neural network achieved a test accuracy, the fraction of correct predictions, of 95.88 % or an error rate of 4.12 %. In other words, our one-layer neural network correctly recognized over 95 % of all test images.

2.2.1.4 Convolutional Layer

Every output element of a linear layer considers every input. In our MNIST example, this resulted in an input dimension of 764 (28×28). Considering a single 224×224 pixel, image this would result in an input dimension of over 50 000, which would have been a problem before the widespread availability of graphics processing units (GPUs). Nowadays, it is technically feasible to train large-scale networks consisting of linear layers [55]. However, as a linear layer considers every input, it is more difficult to calculate abstract representations of an image, filtering out noise.

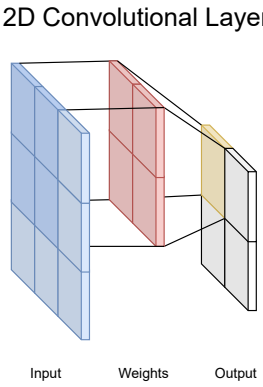

Figure 2.5: A two-dimensional convolutional layer.

To create a more efficient layer, specialized for images, LeCun proposed a convolutional layer (see Figure 2.5)[56]. The intuition behind the (2D) convolutional layer is that a local two-dimensional patch is encoded by a set of weights (called kernel). By re-using the kernel and applying it across the image, it is trained to recognize specific image characteristics, such as straight lines, textures, or objects, independent of their location[17]. To encode different textures, several kernels are applied in parallel. By stacking several convolutional layers, more concrete concepts can be encoded, as the stacked input covers a larger portion of the image and is effectively encoded by preceding layers [57][18].

Mathematically, the operation behind a convolutional layer is the similar cross-correlation: the sum of the element-wise multiplication of an input patch and kernel results in one output value, as shown in Figure 2.5. If the image has multiple channels, the kernel must have the same number of channels. In other words, both the image and kernels are tensors. As additional configuration, we can define the stride, how much the kernel is shifted between its applications, and padding. Padding refers to options for extending the image so that the kernel always has enough input values, with the padded

[17] An interactive example is shown at `https://commons.wikimedia.org/wiki/File:2D_Convolution_Animation.gif#/media/File:2D_Convolution_Animation.gif`

[18] A series of peer-reviewed articles investigating and visualizing learned concepts of different neural network layers can be found on `https://distill.pub/`.

pixels filled with different values, such as all zeros or mirroring the edge of the image. A convolutional layer can be used in PyTorch as follows:

```
conv = nn.Conv2d(in_channels, out_channels,
                 kernel_size, stride=stride,
                 padding=padding)
```

We can replace the first fully-connected layer with two convolutions from our one-layer neural network and obtain[19]:

```
class TwoLayerCNN(nn.Module):

    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels=1,  out_channels=9,
                               kernel_size=(3, 3))
        self.conv2 = nn.Conv2d(in_channels=9,  out_channels=5,
                               kernel_size=(3, 3))
        self.fc1 = nn.Linear(in_features=2880, out_features=10)

    def forward(self, digit):
        out = nn.functional.relu(self.conv1(digit))
        out = nn.functional.relu(self.conv2(out))
        out = self.fc1(out.flatten(1))
        return out
```

By replacing the first linear layer with two convolutional layers, we are able to reduce the number of trainable parameters from 101,770 to 29,310, thereby increasing training speed. Additionally, using convolutional layers also improved the classification error rate from 4.12 % to 2.07 %.

2.2.1.5 Pooling Layers

The convolutional neural network (CNN) presented in the previous section had a classification layer (the final linear layer) with a high-dimensional input of 2,880. Pooling multiple values is a common technique in convolutional neural networks to reduce the dimensionality and therefore computational cost. Two common pooling variants are maximum and average pooling. Similar to a convolutional layer, a kernel is applied over a portion of the image in a sliding-window. Stride and padding are defined analogously. The max-pool operation returns the maximum of the input, the average pool the average. An example is displayed in Figure 2.6, where a 2×2 filter is applied to the input. For the 2×2 input in the red rectangle, max-pool returns $\max\{67, 23, 95, 89\} = 95$ and average pool $\sum \frac{i}{4}, i \in \{67, 23, 95, 89\} = 68.5$.

[19] A complete example is available at `https://www.kaggle.com/fold10/mnist-convnet`

Max-pool is often used as part of a convolutional block. For example, the AlexNet architecture uses a convolutional layer, followed by a ReLU activation and a max-pool layer [30]. Average pooling can be used as final layer before the classification to allow for varying image resolutions, called global average pooling, where the kernel size equals the input width and height. Architectures like ResNet [58] and DenseNet [59] use this technique. A pooling layer can be used as follows in PyTorch.

```
tensor = torch.Tensor([[[67, 23],[95, 89]]])
maxpool = nn.MaxPool2d((2, 2))
maxpool(tensor)  # 95

avgpool = nn.AvgPool2d((2, 2))
avgpool(tensor)  # 68.5
```

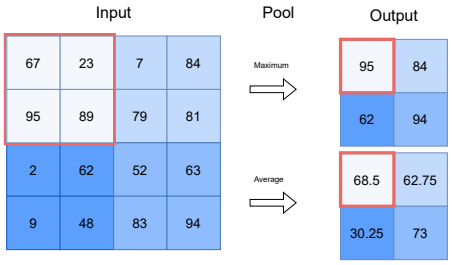

Figure 2.6: Commonly used pooling operations: max-pool and average pool. All pooling operations use a sliding window to down-sample the input. Here, a 2×2 filter is applied to the input. Max-pool returns the maximum, while average pool the mean of the elements.

Including 3×3 max pooling after each convolution reduces the input dimension of the classification layer of the MNIST CNN to 320 with a total of 3710 trainable parameters, less than 1 % of the original one-layer neural network. It also improved the error rate further to 1.67 %[20].

2.2.1.6 Batch Normalization

The training of deep neural networks is a complex process, as each input of a layer is affected by changes in all preceding layers. Consequently, the choice of initialization of a layer is dependent on its depth. Furthermore, as the weights get updated during the training, the data passing through a layer may experience a shift in distribution [60]. This can negatively affect the training, as a shift in distribution can make it unstable.

To stabilize training and increase training speed, Ioffe and Szegedy proposed to normalize the output of a layer before the next layer in a batch-wise fashion, called batch normalization [60]. To counteract the distribution shift of data passing through a layer, as the weights get updated over time, they normalize the output to have zero mean and unit variance after each batch. As the normalization could interfere with the learning process, they add a learnable scaling and shifting term to revert the normalization, if necessary.

In recent convolutional neural networks, batch normalization is commonly placed between the convolutional layer and the activation function [58, 59, 61]. To use batch normalization in PyTorch, one can use the `nn.BatchNorm2d` module. For example:

```
# batch containing a single 2x2x1 tensor
tensor = torch.Tensor([[[[67, 23], [95, 89]]]])
batchnorm = nn.BatchNorm2d(1)  # one channel
output = batchnorm(tensor)  # approximatively zero mean and unit variance
```

[20] A complete example is available at `https://www.kaggle.com/fold10/mnist-convnet`

By applying batch normalization to our MNIST CNN we obtain the following model[21].

```
class ImprovedTwoLayerCNN(nn.Module):

    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels = 1,  out_channels=9,
                               kernel_size=(3, 3))
        self.norm1 = nn.BatchNorm2d(9)
        self.pool = nn.MaxPool2d((3, 3))
        self.conv2 = nn.Conv2d(in_channels = 9,  out_channels=5,
                               kernel_size=(3, 3))
        self.norm2 = nn.BatchNorm2d(5)
        self.fc1 = nn.Linear(in_features=320, out_features=10)

    def forward(self, digit):
        out = nn.functional.relu(self.norm1(self.conv1(digit)))
        out = self.pool(nn.functional.relu(self.norm2(self.conv2(out))))
        out = self.fc1(out.flatten(1))
        return out
```

The improved two-layer CNN has 28 additional trainable parameters due to batch normalization and achieved a test error rate of 1.59 % compared to 1.67 % without batch normalization.

2.2.1.7 Skip Connections

Bach normalization combined with ReLU activations addressed the vanishing gradient problem [60] allowing models to become deeper. The vanishing gradient problem can occur when the gradients of the activation function are close to zero so the gradients for the preceding layers become too small to train the network further. This is a particular problem for saturating activation functions like tanh or sigmoid but can be addressed with ReLU activations [30].

These advancements allowed He et al. to train very deep neural networks. While they assumed that deeper networks perform better, they discovered that they actually performed worse than shallower models even though the shallower networks solution space is a subspace of the deeper one [58]. They explained their findings that the model has difficulties passing the information to deeper layers, similarly to an identity function. Therefore, they added identity functions to the network explicitly, so-called skip connections.

We can reproduce these findings by adding a depth parameter to our CNN. We refactor the convolutional block, convolution, batch normalization, and ReLU, as a `ConvLayer class`. By fixing the number of output channels to 10, we can stack them easily. Only the first convolutional layer has a single input channel, due to the binary image format. Instead of calculating the number of input channels of the final classification layer, we

[21] A complete example is available at `https://www.kaggle.com/fold10/mnist-convnet`

use a `LazyLinear` layer, that computes the necessary input dimension automatically during initialization. With the following setup, we can train the model with one, ten, fifty, and one hundred convolutional layers[22]:

```
class ConvLayer(nn.Module):
    def __init__(self, in_channels):
        super().__init__()
        self.conv = nn.Conv2d(in_channels, 10, kernel_size=3,
                              stride=1, padding=1, bias=False)
        self.norm = nn.BatchNorm2d(10)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, digit):
        return self.relu(self.norm(self.conv(digit)))

class DeepCNN(nn.Module):
    def __init__(self, depth):
        super().__init__()
        self.features = nn.Sequential(ConvLayer(in_channels=1))
        for i in range(1, depth):
            self.features.add_module(f"conv_{i}", ConvLayer(in_channels=10))
        self.pool = nn.AvgPool2d((4, 4))
        self.classifier = nn.LazyLinear(out_features=10)

    def forward(self, digit):
        out = self.pool(self.features(digit))
        return self.classifier(out.flatten(1))
```

The trained model performed best with 10 convolutional layers, as shown in Table 2.1. Going deeper, the performance deteriorated quickly. The 100-layer CNN had an error rate of 37.3 %, which is 14 times worse than the one-layer CNN.

Layers	CNN	+ Skip Connections
1 layer	1.94 %	1.93 %
10 layers	0.67 %	0.87 %
50 layers	1.03 %	0.76 %
100 layers	27.3 %	0.72 %

Table 2.1: Going deeper with convolutional networks. Error rates of the MNIST CNN with and without skip connections containing 1, 10, 50, and 100 convolutional layers. Adding skip connections stabilized the performance.

We can include skip connections, as proposed by the ResNet model, by adding the input and output of the layer starting from the second layer, as shown below. Introducing skip connections does not affect the number of the trainable parameters.

```
class ConvResLayer(ConvLayer):
    def forward(self, input):
```

[22]The complete notebook is available at `https://www.kaggle.com/fold10/mnist-convnet`

```
        output = self.norm(self.conv(input1))
        return self.relu(input + output)

class DeepResCNN(nn.Module):
    def __init__(self, depth):
        super().__init__()
        self.features = nn.Sequential(ConvLayer(in_channels=1))
        for i in range(1, depth):
            self.features.add_module(f"conv_{i}", ConvResLayer(10))
        self.pool = nn.AvgPool2d((4, 4))
        self.classifier = nn.LazyLinear(out_features=10)

    def forward(self, digit):
        out = self.pool(self.features(digit))
        return self.classifier(out.flatten(1))
```

While the error rates varied slightly across runs due to non deterministic initialization, the effect of skip connections on the 50- and 100-layer CNN remained striking. Both 1- and 10-layer versions performed similarly, as shown in Table 2.1. The 50- and 100-layer CNNs with skip connections performed similarly to the 10-layer CNN, without skip connections, the performance degraded quickly with the additional layers.

2.2.1.8 Transformer

In recent years, best performing deep learning models on text (language models) have been based on the transformer architecture [62], such as the famous GPT architecture [63, 64, 65, 66]. A transformer layer consists of skip connections, a linear layer, and an attention layer. The attention layer is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

where Q, K, and V are matrices of queries, keys, and values respectively, and d_k is the dimension of the key vector. Hereby are queries, keys, and values different representations of the same input[23]. Intuitively, the softmax creates a probability distribution of the input, where the probability can be interpreted as "attention": what is the importance of each word (key) with respect to a word (query)? This importance is then multiplied with the word (value)[24].

Due to the success of transformer-based architectures in the natural language processing domain, researchers started applying attention and transformers to images. However, computational efficiency was a major obstacle, as applying attention to every pixel results in quadratic computational costs.

[23] In the literature, this is referred to as self-attention.

[24] An illustrated, detailed explanation of the transformer can be found at `https://jalammar.github.io/illustrated-transformer/`

One popular architecture that addressed this problem is the Vision Transformer [67] by applying attention not to single pixels but larger patches. The Vision Transformer architecture is presented in Section 2.3.2. In this section, we replace the convolutions of our MNIST CNN with a simplistic transformer, as shown in Figure 2.7.

Similar to our previous deep CNN, modern deep learning architectures are designed to be scaled easily by increasing their depth. We achieve this in our transformer model by designing transformer blocks with the same input and output dimension. Therefore, we can simply stack multiple transformer blocks ($L\times$). Our transformer block consists of an attention layer and a linear layer. Both layers use skip connections, to allow deeper models, and normalization, to stabilize the training procedure [68]. To reduce computational complexity of the attention layer, we divide the input image into sixteen 7×7 pixel patches.

The attention layer can be implemented as follows, utilizing the einsum notation for easier matrix manipulations[25]. First, the input is embedded by a linear layer as query, key, and value (`to_qkv`) and then processed by the attention function. Having encapsulated the attention layer as a separate class, the transformer layer is straight-forward to implement.

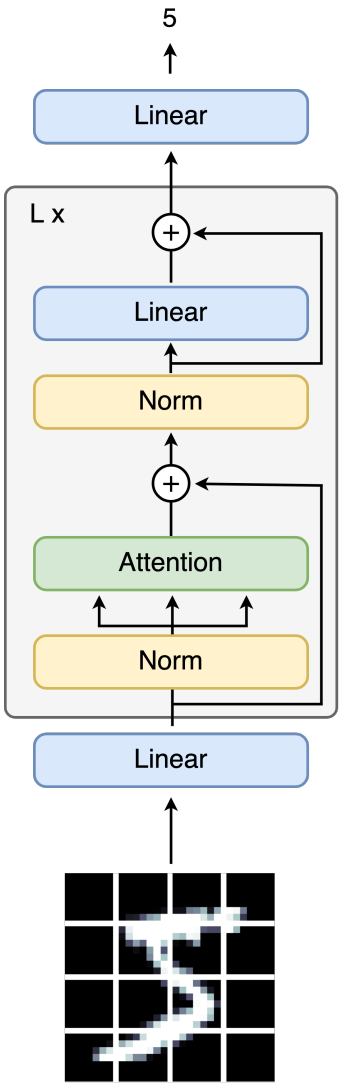

Figure 2.7: A simple transformer architecture.

```
class Attention(nn.Module):
    def __init__(self, dim):
        super().__init__()
        self.scale = dim ** -0.5
        # dim * 3 for Q,K,V
        self.to_qkv = nn.Linear(dim, dim * 3,
                                bias = False)

    def forward(self, x):
        b, n, _, h = *x.shape, 2
        qkv = self.to_qkv(x).chunk(3, dim = -1)
        q, k, v = map(lambda t: rearrange(t,
            'b n (h d) -> b h n d', h = h), qkv)
        dots = einsum('b h i d, b h j d -> b h i j',
                      q, k) * self.scale
        attn = dots.softmax(dim=-1)
        out = einsum('b h i j, b h j d -> b h i d',
                     attn, v)
        out = rearrange(out, 'b h n d -> b n (h d)')
        return out
```

[25] https://github.com/arogozhnikov/einops

```
class Transformer(nn.Module):
    def __init__(self):
        super().__init__()
        # input = output dimension = 32
        self.attention = Attention(32)
        self.norm1 = nn.LayerNorm(32)
        self.fc1 = nn.Linear(32, 32)
        self.norm2 = nn.LayerNorm(32)

    def forward(self, x):
        out = nn.functional.relu(
            self.attention(self.norm1(x)) + x)
        out = nn.functional.relu(
            self.fc1(self.norm2(out)) + out)
        return out
```

Similar to the previous DeepResCNN model, a deeper model improved the classification performance before saturating: at depth 1, the model achieved an error rate of 13.47 %, at depth 10, 3.82 %, and at depth 20, 3.87 %, respectively[26]. These numbers already show that the DeepResCNN performed significantly better than the vision transformer. However, scaling studies showed that CNNs performed worse than vision transformers when pre-trained on hundreds of millions of images [67]. In other words, vision transformers seem to better utilize large data sets. Both results can be explained by the nature of vision transformers. CNNs apply a fixed kernel to a local region. Vision transformers, on the other hand, use the complete input and learn to attend to specific regions [69]. This lack of inductive bias helps the transformer model to learn more precise data transformations. Since these must be learned in the first place, a vision transformer requires more training data. Without the data, the fixed kernels of a CNN are a good approximation.

2.2.2 Data Augmentation

Data augmentation is a technique to synthetically create more data. Besides increasing the amount of data, data augmentation encourages the model to learn a form of invariance. For example, the model should be able to accurately predict the digit in an image despite variations in location or tilt (see Figure 2.8). There are several techniques for data augmentation and not every technique is suitable for each application. In this section, we will review common data augmentation techniques used for our MNIST example and chest X-ray classification. A more general review can be found in [70] and [71].

For the MNIST data set, we can incorporate geometric transformations to improve generalization. Two suitable techniques are rotating the image, simulating a tilted handwriting, and shifting it. When applying these data transformations, one must be mindful of the effect on the label. Another common technique, flipping the image horizontally,

[26] The complete notebook can be found at `https://www.kaggle.com/fold10/mnist-vision-transformer-vit`

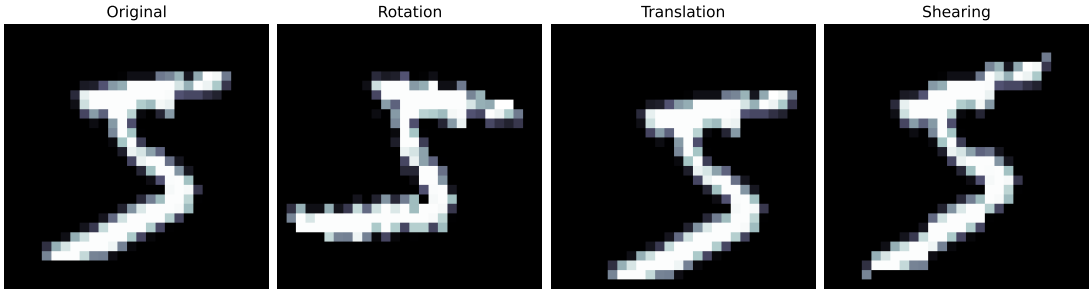

Figure 2.8: Data augmentation applied to an image of a handwritten digit. Moderate rotation, translation, and shearing alter the shape of the digit but not its value.

would convert a 6 to a 9 and vice-versa. For chest X-rays, a horizontal or vertical flip should not affect the prediction of the model. Also, as chest X-rays are high resolution images and are therefore resized, cropping the image is a common transformation.

In PyTorch, we can use the transform module of the TorchVision library[27] or specific libraries such as albumentations[28] for data augmentation. For our MNIST example, we can generate a random rotation (see Figure 2.8) as follows:

```
import torchvision.transforms as T

# define transformation pipeline
transforms = T.Compose([
    # torch vision expects three input channels
    T.Lambda(lambda x: x.expand(3, -1, -1)), # create a three channel view
    T.RandomRotation(degrees=35),
    T.Lambda(lambda x: x[:1,:,:]) # collapse to one channel
])
```

For the MNIST example, adding augmentations barely improved the performance, as the model's performance is already close to optimal. The 10-layer DeepResCNN achieved an error rate of 0.79 % with data augmentation compared to 0.87 % without. However, across runs, the performance of both versions was similar[29]. For more complex images, such as chest X-rays, including data augmentations in our experiments improved the model performance significantly.

Typically, data augmentations are only applied during training. However, another use of data augmentations is during testing, so-called test time augmentations. Here, predictions for multiple augmented versions of the same image are averaged to get a more robust prediction. Consequently, test time augmentations are especially common in deep learning competitions [30, 58].

[27] `https://pytorch.org/vision/stable/index.html`
[28] `https://albumentations.ai/`
[29] The complete notebook can be found at `https://www.kaggle.com/code/fold10/mnist-convnet`.

2.3 Architectures

This section presents the backbone architectures used throughout this thesis. These architectures are built from components presented in the previous section and are commonly used for transfer learning, i.e., the models were trained on a large data set like ImageNet [72] and then fine-tuned on a target data set, for example, chest radiographs.

2.3.1 DenseNet

Extending the principle of skip connections further, Huang et al. proposed to connect every layer of a "DenseBlock" inside their DenseNet CNN to all preceding layers using skip connections [59]. This allowed them create deeper models with the same number of parameters, improving the performance. Additionally, instead of adding the outputs they concatenated them. Therefore, to reduce the dimension between DenseBlocks they introduced transition and bottleneck segments.

2.3.2 Vision Transformer

As transformer-based architectures outperformed conventional recurrent networks on natural language tasks, researchers rushed to adapt the architecture to the image domain. The difficulty was, however, that the standard self-attention algorithm has a quadratic runtime as every word attends every other word. While computationally feasible for text, it limited the image resolution and depth of the network considerably when applied to pixels naively. Therefore, instead of applying the attention directly on pixels, Dosovitskiy et al. proposed to use 16×16 pixel patches [73]. Their vision transformer (ViT) architecture is mostly identical to the transformer developed in the previous section, with the difference of using multiple attention "heads" in parallel (multi-head attention), similarly to using multiple kernels for convolutional layers. Similar to our MNIST experiments, the vision transformer model performed worse than state-of-the-art CNNs on smaller data sets but when pre-trained on 20 - 300 million images the ViT outperformed convolution based architectures.

2.4 Development of Deep Learning Models

2.4.1 Introduction

Deep learning research papers often lack detailed descriptions of the methods, recipes, and tools used for developing deep learning-based software. To address this gap, this section describes the methodologies we employed in our research.

In general, I like the approach of Jeremy Howard when introducing the development of neural networks: focusing on practical approaches [74] and teaching deep learning development[30] with a top-down approach using interactive development tools like Jupyter

[30] `fast.ai`

notebooks[31]. The benefit is to quickly start coding deep learning models and to lower the barrier of entry with the downside of accepting code that has been abstracted away in a more rigid framework.

After getting acquainted with simple deep neural networks, to break out of more rigid frameworks, and, as current research models are mostly implemented in PyTorch[32], I recommend the official PyTorch tutorials. Again, by using Jupyter notebooks this should be an interactive, exploratory form of learning. Quickly, one will notice the standard PyTorch structure: dataset, data loader, train-, validation- and test loops sprinkled with logging, checkpoints, and development plots. Understanding these concepts is crucial for adapting and developing deep learning models.

The following sections provide a more in-depth introduction to deep learning development, continuing with our PyTorch implementation of an MNIST classifier. To provide more clarity, complete code examples are provided alongside. The referenced notebooks can be run directly in a browser.

2.4.2 PyTorch

2.4.2.1 Data Set

The PyTorch data set encapsulates the access and transformation of the raw data. A PyTorch data set implementation has to define the `__len__` and `__getitem__` methods, defining how many elements exist and how to access a specific element of the data set. Because data sets are used for multiple models, the PyTorch data set should be as general as possible. In our group I introduced a separate data abstraction layer to provide a general interface across data sets, including features like caching, reducing the PyTorch implementation to the bare essentials. Similarly, we can create a PyTorch data set for the MNIST example as follows[33]:

```
class Dataset(torch.utils.data.Dataset):
    def __init__(self, images, labels, transform):
        super().__init__()
        self.images = images
        self.labels = labels
        self.transform = transform

    def __len__(self):
        return len(self.images)

    def __getitem__(self, idx):
        # data augmentations
        return self.transform(self.images[idx]), self.labels[idx]

train_dataset = Dataset(train_images, train_labels, transform=transform)
```

[31] jupyter.org
[32] pytorch.org
[33] The complete example is available at https://www.kaggle.com/code/fold10/mnist-convnet.

```
val_dataset = Dataset(val_images, val_labels, transform=transform)
test_dataset = Dataset(test_images, test_labels, transform=transform)
```

The benefit of this approach is that all data sets: training, validation and testing can reuse the same class without custom code.

2.4.2.2 Transforms

The transformation of an image to PyTorch tensor is handled in a separate `Transforms` class. For example, we can implement the data augmentations introduced in the previous section as follows:

```
import torchvision.transforms as T

class Transforms():
    def __init__(self):
        self.transform = T.Compose([
            T.ToTensor(),
            T.Lambda(lambda x: x.expand(3, -1, -1)),
            T.RandomRotation(degrees=5),
            T.RandomPerspective(distortion_scale=0.6, p=1.0),
            T.Lambda(lambda x: x[:1,:,:])
        ])

    def __call__(self, input):
        return self.transform(input)
```

2.4.2.3 Data Loader

The PyTorch data loader takes a data set and provides efficient data access, creates batches, and other utilities. For our MNIST example, we can implement it as follows. Note, that for the training data loader we specify that the data set should be shuffled for every epoch. This ensures that the model does not simply memorize the sequence.

```
from torch.utils.data import DataLoader
train_loader = DataLoader(dataset=train_dataset, batch_size=100, shuffle=True)
val_loader = DataLoader(dataset=val_dataset, batch_size=100, shuffle=False)
test_loader = DataLoader(dataset=test_dataset, batch_size=100, shuffle=False)
```

2.4.2.4 Train/Validation/Test

Given model and data loader, we can create the training and validation loop. To speed up the training, we can train the model on a GPU by moving model and data to a specific `device`. Using the data loader, the code for accessing images and labels is much more readable compared to the `OneLayerNN` training routine in Section 2.2.1.3.

```
def train(cnn, train_loader, val_loader, cuda=False):
    device = "cuda:0" if cuda else "cpu"
    cnn.to(device)
```

```
    optim = torch.optim.Adam(params=cnn.parameters(), lr=1e-3)
    val_predictions = []
    for epoch in range(15):
        for (images, y_true) in train_loader:
            y_true = y_true.flatten(1).to(device)
            y_pred = cnn(images.to(device))
            loss = torch.nn.functional.cross_entropy(y_pred, y_true)
            optim.zero_grad()
            loss.backward()
            optim.step()
        # No need to compute the gradients during validation
        with torch.no_grad():
            for (images, y_true) in val_loader:
                y_true = y_true.flatten(1).to(device)
                y_pred = cnn(images.to(device))
                loss = torch.nn.functional.cross_entropy(y_pred, y_true)
                val_predictions.extend(torch.argmax(y_pred, axis=1).cpu())
    return val_predictions
```

We can test our model and calculate accuracy and error rate as follows:

```
def test(cnn, labels, data_loader, cuda=False):
    device = "cuda:0" if cuda else "cpu"
    predictions = []
    with torch.no_grad():
        for (images, y_true) in data_loader:
                y_true = y_true.flatten(1).to(device)
                y_pred = cnn(images.to(device))
                loss = torch.nn.functional.cross_entropy(y_pred, y_true)
                predictions.extend(torch.argmax(y_pred, axis=1).cpu())
        total = len(labels)
        print("Accuracy", sum(predictions == labels).item() / total)
        print("Error Rate", sum(predictions != labels).item() / total * 100)
```

2.4.3 PyTorch Lightning

The previous section presented the typical implementation of a neural network in PyTorch with the example of the MNIST data set. From a high-level, perspective deep learning code consists of two main parts: model and data. However, the PyTorch syntax is comparatively verbose and repetitive. For example, the validation and test loop are functionally identical and mostly independent of the actual model and data set. One attempt to streamline deep learning development and improve code readability and reproducibility is PyTorch Lightning[34]. Besides reducing boiler-plate code the framework makes it simple to, for example, log metrics, train the model on multiple GPUs, or create checkpoints of the trained model. For most models, only the model-specific training steps must be developed. The outer training loop, for example, is already provided by PyTorch Lightning.

[34] https://lightning.ai/pytorch-lightning/

This section introduces the PyTorch Lightning framework with the example of MNIST image classification. In my work, I mostly used PyTorch Lightning version 1.8, changes to the current version are documented online[35].

2.4.3.1 Data Module

The data module encapsulates all data related aspects of the deep learning code, mainly the PyTorch data set and the PyTorch data loader but also extracting the compressed data set or splitting the data into training, validation, and test data sets. Conceptually, a data module consists of two different parts: single use functions, e.g., downloading and splitting the data called during data preparation (`prepare_data`), and actions that need to be done, for example, for every GPU in a distributed training scheme, like assigning training data (`setup`). The full documentation is available online[36].

The data module embodies data preparation, data set initialization, and data loading, as shown in the following code sample[37].

```
class DataModule(pl.LightningDataModule):
    def prepare_data(self):
        train_val_images = self.read_image_data("train-images.idx3-ubyte")
        train_val_labels = self.read_labels("train-labels.idx1-ubyte")
        self.train_images = train_val_images[:int(0.9 * len(train_val_images))]
        self.train_labels = train_val_labels[:len(self.train_images)]

        self.val_images = train_val_images[len(self.train_images):]
        self.val_labels = train_val_labels[len(self.train_images):]

        self.test_images = self.read_image_data("t10k-images.idx3-ubyte")
        self.test_labels = self.read_labels("t10k-labels.idx1-ubyte")

    def setup(self, stage):
        self.train_dataset = Dataset(self.train_images, self.train_labels,
                                     Transforms())
        self.val_dataset = Dataset(self.val_images, self.val_labels,
                                   Transforms())
        self.test_dataset = Dataset(self.test_images, self.test_labels,
                                    Transforms())

    def train_dataloader(self):
        return DataLoader(self.train_dataset, batch_size=100, shuffle=True)

    def val_dataloader(self):
        return DataLoader(self.val_dataset, batch_size=100)

    def test_dataloader(self):
```

[35] https://lightning.ai/docs/pytorch/stable/upgrade/from_1_8.html
[36] https://lightning.ai/docs/pytorch/stable/data/datamodule.html
[37] The complete notebook is available at https://www.kaggle.com/fold10/mnist-image-classification-with-pytorch-lightning/.

```
        return DataLoader(self.test_dataset, batch_size=100)

    def read_image_data(self, path):
        # reads the image data from disk
        ...

    def read_labels(self, path):
        # reads the meta data from disk
        ...
```

Most of the necessary code was already implemented in the `Dataset`, `Transforms`, and `DataLoader` classes. Only the data preparation, loading the data from disk and splitting it into training, validation, and testing, was previously defined elsewhere. Using PyTorch Lighting, it is now defined in the `prepare_data` method. Overall, everything related to the data was incorporated into the `DataModule`.

2.4.3.2 Lightning Module

The lightning module complements the data module and encloses all model related code[38]. It provides access to the deep learning model, optimizer, and defines the training procedure. Additional development utilities, such as logging the training progress, are also defined here. A lightning module for our two layer CNN is provided in the following:

```
class Model(pl.LightningModule):
    def __init__(self, learning_rate):
        super().__init__()
        self.save_hyperparameters()
        self.model = CNN()
        self.loss = torch.nn.functional.cross_entropy
        # Store the output for evaluation
        self.val_outputs = []
        self.test_outputs = []

    def forward(self, images):
        return self.model(images)

    def _step(self, stage, batch, batch_idx):
        images, y_true = batch
        y_pred = self(images)
        loss = self.loss(y_pred, y_true.flatten(1))
        self.log(f"loss/{stage}", loss, prog_bar=True)
        return {"loss": loss,
                "labels": y_true,
                "preds": torch.argmax(y_pred.detach(), axis=1)}

    def training_step(self, batch, batch_idx):
```

[38] The full documentation is available online at `https://lightning.ai/docs/pytorch/stable/common/lightning_module.html`.

```
        return self._step("train", batch, batch_idx)

    def validation_step(self, batch, batch_idx):
        self.val_outputs.append(self._step("val", batch, batch_idx))
        return self.val_outputs[-1]

    def test_step(self, batch, batch_idx):
        self.test_outputs.append(self._step("test", batch, batch_idx))
        return self.test_outputs[-1]

    def _epoch_end(self, stage, outputs):
        predictions = torch.hstack([o["preds"] for o in outputs])
        labels = torch.vstack([o["labels"] for o in outputs]).argmax(axis=1)
        total = len(labels)
        accuracy = sum(predictions == labels).item() / total * 100
        error_rate = sum(predictions != labels).item() / total * 100
        self.log_dict({f"accuracy/{stage}": accuracy,
                       f"error_rate/{stage}": error_rate}, prog_bar=True)

    def on_validation_epoch_end(self):
        return self._epoch_end("val", self.val_outputs)

    def on_test_epoch_end(self):
        return self._epoch_end("test", self.test_outputs)

    def configure_optimizers(self):
        return torch.optim.Adam(self.model.parameters(),
                                lr=self.hparams.learning_rate)
```

For our simple MNIST CNN, the actions that are performed in the training, validation, and test loop can be abstracted as a general step function. This function creates the prediction for a given batch of images, calculates and logs the loss and returns loss, labels, and predictions. For later evaluation at the end of an epoch, we store the validation and test predictions. Again, as the evaluation is the same for both training stages, we define a common function(`_epoch_end`) to calculate and log accuracy and error rate. Finally, in the `configure_optimizers` method, the optimizer is defined. In summary, the lightning model incorporates everything necessary regarding the deep learning model.

2.4.3.3 Trainer

The trainer class manages the actual training procedure. Here, we can adjust, for example, the number of epochs or the device used for training. Given data module, model, and trainer we can finally train our model:

```
datamodule = DataModule()
model = Model(learning_rate=1e-3)
trainer = pl.Trainer(max_epochs=15, accelerator="gpu", devices=1)
trainer.fit(model, datamodule)
trainer.test(model, datamodule)
```

PyTorch Lightning transfers model and data automatically to the specified device and generates a progress bar to keep track of the training procedure. Overall, by using PyTorch Lightning, we can wrap all code into specific classes and are able to train and test our model with five lines of code.

2.4.3.4 Command Line Interface

While the trainer can be called manually, for example in a Jupyter notebook, a better approach is to call it in a Python module and specify the hyper-parameters as command line arguments.

```
if __name__ == "__main__":
    parser = ArgumentParser()
    parser = add_argparse_args(parser)
    parser = DataModule.add_argparse_args(parser)
    parser = Model.add_argparse_args(parser)
    parser = pl.Trainer.add_argparse_args(parser)
    datamodule = DataModule.from_argparse_args(args)
    model = Model.from_argparse_args(args)
    trainer = pl.Trainer.from_argparse_args(args)
    trainer.fit(model, datamodule)
    trainer.test(model, datamodule)
```

Using the `ArgumentParser`[39], the arguments of trainer, lightning and data module can be specified in the terminal. PyTorch Lightning already implements the `add_argparse_args` functionality for the different classes[40]. Additional arguments can be added as follows:

```
class Model(pl.LightningModule):
    @staticmethod
    def add_argparse_args(parent_parser):
        parser = parent_parser.add_argument_group("Model")
        parser.add_argument("--learning_rate", type=float, default=1e-3)
        return parent_parser

    @classmethod
    def from_argparse_args(cls, *args, **kwargs):
        """Pass the ArgParser's args to the constructor."""
        return pl.utilities.argparse.from_argparse_args(cls, *args, **kwargs)
    ...
```

Now, the model can be trained by passing the arguments, e.g., the learning rate, as command line arguments: `python main.py --learning_rate=0.01`.

[39] `https://docs.python.org/3/library/argparse.html`
[40] The latest PyTorch Lightning version replaced the ArgumentParser with an internal Lightning CLI.

2.4.4 Experiment Management with Guild

Guild.ai[41] is an open-source machine learning management package. In the previous sections, we developed a deep learning model using PyTorch and PyTorch Lightning. For software development, code changes are tracked using version control software like git[42]. When prototyping deep learning models, the code base might not change between runs, as only the hyper-parameters are tuned via the terminal. Furthermore, side-effects like binary model checkpoints or plots are not suited for code version control software. While there are existing solutions like DVS[43] to track data using version control, in my experience, a better approach is proposed by guild.

Guild creates snapshots of the code used for training a model and stores data artifacts along-side it. Furthermore, it uses a configuration file, `guild.yml`, to manage hyper-parameters and provides a CLI to initiate model training. Thus, guild enables reproducible builds. A sample configuration file for our MNIST example is provided below[44]. The file defines an operation, `train`, the necessary source code, and the entry point (`main.py`). Additional files that are necessary are specified as `required`, for example, we link the data. Combined with PyTorch Lightning, we can define the arguments of the argument parser either in the configuration or directly in the CLI.

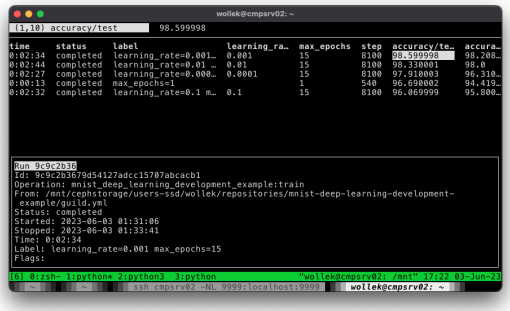

Figure 2.9: Comparing different training runs in the terminal using guild compare.

```
- model: mnist_deep_learning_development_example
  sourcecode:
    - "*.py"
  operations:
    train:
      main: mnist_deep_learning_development_example.main
      requires:
        - file: "mnist-dataset"
          target-type: link
      flags:
        max_epochs:
          default: 15
        learning_rate:
```

Given the configuration file, we can run, for example, a grid search to find the best learning rate using

[41] `https://guild.ai/`
[42] `https://git-scm.com/`
[43] `https://dvc.org/`
[44] The complete project can be found at `https://github.com/AlessandroW/mnist-pytorch-lightning`

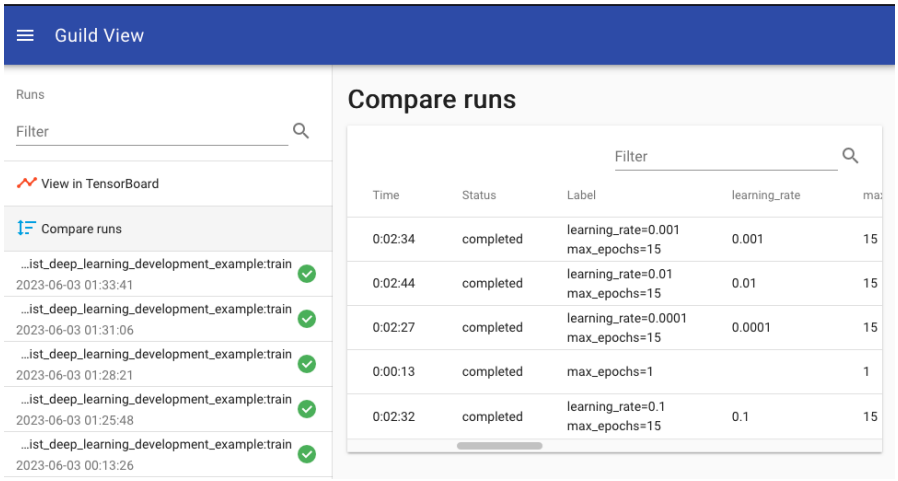

Figure 2.10: Inspecting and comparing different training runs in the browser using guild view.

```
guild run train learning_rate=[0.1,0.01,0.001,0.0001].
```

To compare the effect of different learning rates, we can, for example, compare the results in the terminal (`guild compare`), in a web interface with additional TensorBoard[45] support, or interactively in a Jupyter notebook.

```
import guild.ipy as guild
# All runs
runs = guild.runs()
# Runs filtered by status and operation
selected_runs = runs.loc[
  (runs.operation == "mnist_deep_learning_development_example:train") &
  (runs.status == "completed")]

scalars = selected_runs.scalars()
# Get the lowest test error rate for each completed run
scalars.loc[scalars.tag == "error_rate/test"][
  ["run", "min_val"]].sort_values("min_val")
```

	run	min_val
348	9c9c2b3679d54127adcc15707abcacb1	1.40
693	9370509cc22946d0aac401b91a39174d	1.67
1383	6b20cb24964e435997196064a2d29b74	3.31
1038	24c14e7356594226b9896d11fcd05e19	3.93

We get a similar output in the terminal by running `guild compare`, shown in Figure 2.9. Alternatively, we can inspect the different training runs in the browser by running `guild view`, see Figure 2.10. The browser view allows to inspect the training process in TensorBoard. As guild hooks into the training process, everything that was logged by PyTorch Lighting is visualized in TensorBoard, as show in Figure 2.11.

[45] `https://www.tensorflow.org/tensorboard`

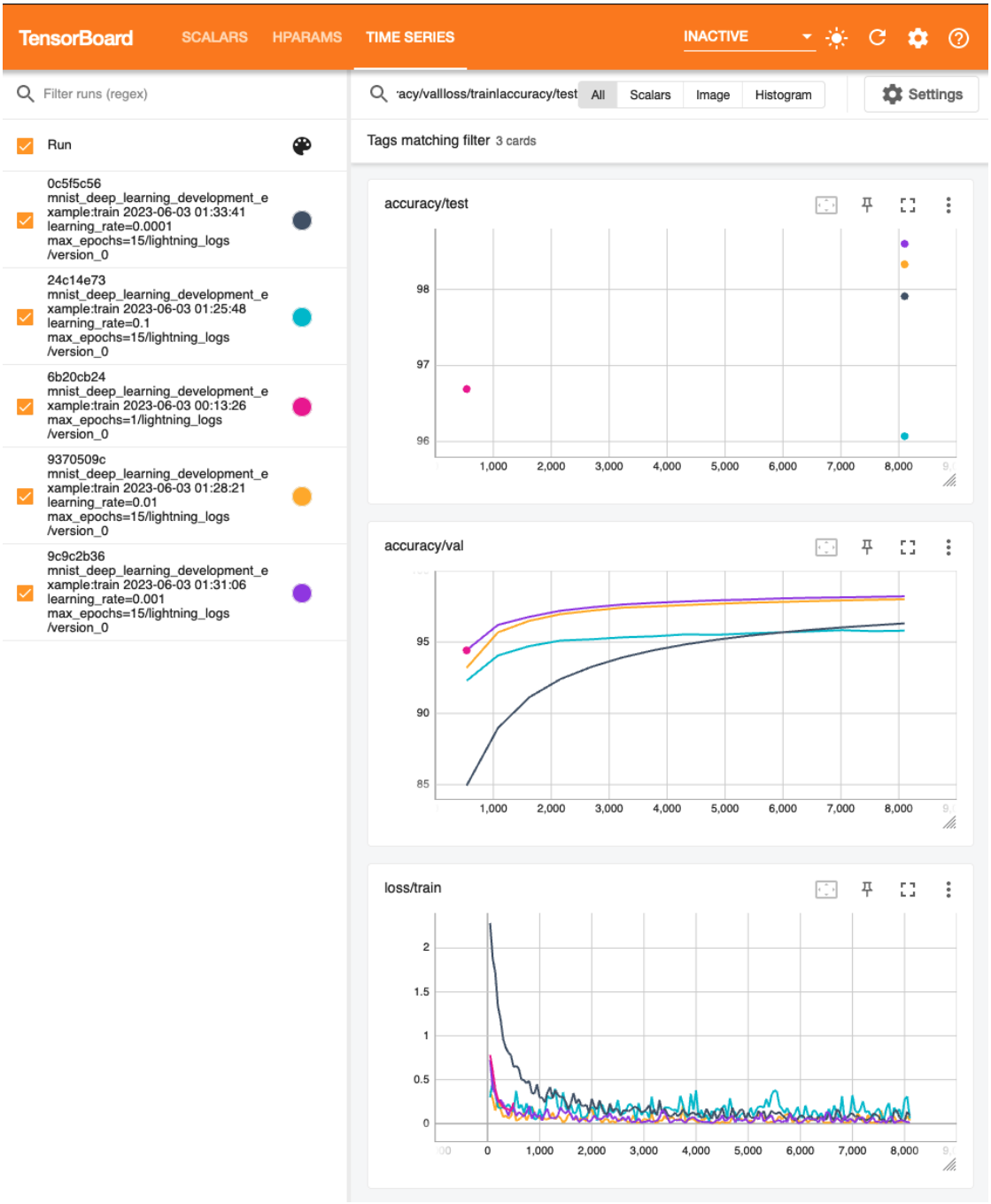

Figure 2.11: Comparing different training runs using TensorBoard.

3 Background: Deep Learning for Chest X-Ray Diagnosis

3.1 Public Chest X-Ray Data Sets

The prerequisite for training neural networks capable of interpreting chest radiographs are such data sets. This section presents commonly used publicly available chest X-ray data sets. An overview is presented in Table 3.1.

Data Set	Year	# Findings	# Samples	Annotation	Reports
JSRT	2000	1	247	L	N/A
PLCO	2000-2022	N/A	85,421	L, Position	N/A
Open-i	2016	177	7,470	L	Available
CXR14	2017	14	112,120	L, BBoxes	N/A
CheXpert	2019	14	224,316	L	N/A
MIMIC-CXR	2019	14	377,110	L	Available
SIIM-ACR	2020	3	5,302	L, Segmentations	N/A
PadChest	2020	174	160,868	L, Position	Available
VinBigData	2020	28	18,000	L, BBoxes	N/A
CANDID-PTX	2021	3	19,237	L, Segmentations	Available
BRAX	2022	14	40,967	L	Available

Table 3.1: Commonly used publicly available chest X-ray data sets. L = labels, CXR14 = Chest X-ray 14, BBoxes = bounding boxes.

3.1.1 Japanese Society of Radiological Technology

One of the earliest chest X-ray data sets, released in 2000, is from the Japanese Society of Radiological Technology, or JSRT in short [75]. The data sets consists of 247 chest radiographs out of which 154 display lung nodules: 100 malignant, 54 benign. All images were digitized with a 2048×2048 pixel resolution and 12-bit depth.

3.1.2 Prostate, Lung, Colorectal and Ovarian Data Set

The Prostate, Lung, Colorectal, and Ovarian (PLCO) data set[1] is based on a randomized, controlled trial whether specific screening exams reduce mortality from the respective forms of cancer [76]. The lung data set contains 85,421 chest radiographs from 56,071

[1] https://cdas.cancer.gov/plco/

patients acquired during screening procedures. The images have a resolution of 2500 × 2100 pixels with 16-bit gray-scale values. Some of the image annotations mention not only visible pathologies but also their location, for example, upper left lung.

3.1.3 Open-I

The public Open-i data set from the Indiana Network for Patient care contains 7470 chest radiographs (PA and lateral view) from 3955 corresponding radiology reports and patients [77] The labels were extracted manually from radiology reports by matching the impressions and findings section of the reports against medical subject headings (MeSH)[2] and RadLex[3] codes, resulting in 177 total findings. The images have a resolution of 512 × 512 pixels.

3.1.4 Chest X-ray 8, Chest X-ray 14

One of the first openly accessible, large-scale chest X-ray data set is the chest X-ray 8 data set containing 112,120 frontal view chest radiographs from 32,717 patients [78]. Initially only eight labels were provided (chest X-ray 8) but later, the labels were extended to 14 (chest X-ray 14, CXR14). The 14 labels are atelectasis, cardiomegaly, consolidation, edema, effusion, emphysema, fibrosis, hernia, infiltration, mass, nodule, pleural thickening, pneumonia, and pneumothorax. Additionally, the authors provided a small sub set (983 images) with bounding box annotations for the chest X-ray 8 annotations. The images have a resolution of 1024 × 1024 pixels. In contrast to later data sets, the test set was not manually labeled by radiologists but automatically, like the rest of the data set making the evaluation of the labeling performance difficult. Hence, the labels of this data set were criticized for being noisy and that chest X-ray 14 labels pneumonia, consolidation, and infiltration are difficult to distinguish on the radiograph itself, without further clinical information [79].

3.1.5 CheXpert

The CheXpert (Chest eXpert) data set contains 224,316 chest radiographs from 65,240 patients including frontal and lateral views [80]. Like the Chest X-ray 14 data set, the images were labeled according to 14 classes, although different ones. The 14 data set labels are: atelectasis, cardiomegaly, consolidation, edema, enlarged cardiomegaly, fracture, lung lesion, lung opacity, no finding, pleural effusion, pleural other, pneumonia, pneumothorax, and support devices. In contrast to the Chest X-ray 14 data set, the labeling algorithm was released alongside with the data set. The labels were extracted using a rule-based algorithm, which is explained in more detail in Chapter 4.2. Chest X-ray 14 and CheXpert have the following labels in common: atelectasis, cardiomegaly, consolidation, edema, pneumonia, and pneumothorax.

[2] `https://www.nlm.nih.gov/mesh/meshhome.html`
[3] https://radlex.org/

3.1.6 MIMIC-CXR

The MIMIC-CXR data set contains 377,110 images (frontal and lateral view) from 227,835 studies from 65,379 patients including frontal and lateral views [81]. In contrast to other data sets, both images and the corresponding radiology reports were released. The images were labeled using the CheXpert labeler, hence both data sets share the same labels.

3.1.7 Society for Imaging Informatics in Medicine – American College of Radiology

The chest X-ray data set from the Society for Imaging Informatics in Medicine – American College of Radiology (SIIM-ACR) contains 15302 images [82]. The images were randomly selected from the Chest X-ray 14 data set based on their labels. One third was labeled as showing signs of pneumothorax (5,302 images), another third as normal (5,000 images), and the remainder (5,000 images) was labeled with other classes. The authors re-labeled the images manually and segmented pneumothoraces and chest tubes.

3.1.8 Pathology Detection in Chest Radiographs

The Spanish Pathology Detection in Chest Radiographs (PadChest) data set contains 160,868 images (frontal and lateral view) from 109,931 studies from 67,625 patients with their corresponding radiology reports, 39,039 images were labeled manually [83]. The radiology reports were first mapped to the unified medical language system (UMLS) before labeling [84]. The remaining images were annotated automatically using four different deep learning-based models trained on the manual annotations. The images were labeled according to 174 findings and 104 anatomic locations. The labels were organized into hierarchical trees, for example, “chronical tuberculosis” is also “tuberculosis”.

3.1.9 VinBigData

The VinBigData set from the corresponding Kaggle competition[4] contains 18,000 frontal chest radiographs [85]. The images were manually labeled according to 28 findings and further annotated with bounding boxes.

3.1.10 CANDID-PTX

The CANDID-PTX data set contains 19,237 chest radiographs containing pneumothorax, rib fracture, and chest tube segmentations with the corresponding free-text reports [86].

[4] `https://www.kaggle.com/c/vinbigdata-chest-xray-abnormalities-detection`

3.1.11 BRAX

The BRAX data set contains 40,967 chest radiographs from 34,959 patients from a Brazilian hospital [87]. The labels were extracted automatically using a modified version of the CheXpert labeler.

3.2 Chest X-Ray Classification Models

Research on deep learning for chest X-ray classification is heavily dependent on publicly available data. Presented in the previous section, all openly available chest X-ray data sets contain images and labels. Additionally, some smaller data sets (less than 20,000 images) include bounding box annotations (Chest X-ray 14 and VinBigData) or segmentations (SIIM-ACR and CANDID-PTX). As deep learning models required large data sets, most research focused on the multi-label image classification task [88]. This section presents some key publications in the field, a more exhaustive review of the literature can be found in [88].

One commonly used chest X-ray classification baseline is CheXnet [13]. It was one of the first models trained on the Chest X-ray 14 data set for lung disease classification. The model uses a DenseNet-121 as backbone, where the last layer has been replaced with a fully-connected layer with a 14 dimensional output, matching the chest X-ray 14 classes. The authors showed in an extended publication that the architecture performs similar to radiologists [89].

Ke and Ellsworth et al. investigated the choice of model architecture on CheXpert classification performance, concluding that DenseNet, ResNet, and Inception perform better than EfficientNet, MNASNet, and MobileNet [90]. These results are in line with the choice of model architecture in the chest X-ray classification literature [13, 91, 14, 92, 93]. The authors explained their findings with the observation that more recent ImageNet image classification architectures (EfficientNet, MNASNet, and MobileNet) were explicitly optimized for the target data set, potentially overfitting the architecture to it and making them less suitable for transfer learning. They also investigated the importance of pre-training on ImageNet on chest X-ray classification performance with CNNs compared to training them from scratch and demonstrated that transfer learning improved classification performance on the CheXpert data set [90].

While current state-of-the-art ImageNet image classification models are based on the transformer architecture, they perform similar to CNNs for chest X-ray classification [94, 95], probably due to the lack of very large data sets [94]. Xiao et al. demonstrated that pre-training on ImageNet was not necessary for ViTs, achieving better results using self-supervised than ImageNet pre-training [95].

Lacking large chest X-ray data sets with bounding boxes annotations or segmentations for supervised training, several studies investigated related approaches for predicting disease locations. For example, Li et al. modeled chest X-ray classification as a multi-instance learning problem predicting class probabilities for image patches and used them to predicted class probability and location [96]. Similarly, Gadgil et al. leveraged saliency maps (see Chapter 7) to predict segmentations [97].

Part II

Contributions

4 Chest Radiology Reports

4.1 Anonymization of Radiology Reports

To the best of my knowledge, no German chest radiography data set with corresponding free text report is currently publicly available. Although digital data sets are available in many German clinics, data curation and annotation is a time intensive procedure. Most importantly, before releasing a data set it must comply with strict privacy regulations such as the General Data Protection Regulation (GDPR) [98]. Before any sensitive data can be processed outside the clinic, for example, for research purposes, the GDPR requires prior anonymization. We argue that most publicly available chest X-ray data sets lack corresponding free text radiology reports, as only the anonymization of chest radiographs is already automated, by stripping identifiable patient metadata[1]. Free text reports, on the other hand, may contain patient information such as name and date of birth or information about clinical employees that must be identified and anonymized first. As the manual anonymization of large data sets is unfeasible due to time restrictions, together with Martina Hermsdorf, we developed a deep learning-based German radiology report anonymization model.

Our proposed radiology report anonymization pipeline consisted of several steps: timestamps and identifiers were replaced using regular expressions, and names using a pre-trained model trained for named entity recognition (NER)[2], i.e., classifying words, for example, as names [100]. As the model was trained on general German texts, it misclassified medical terms like the David operation as person names. To reduce such false positive classifications, words classified as names were compared against a medical key word data base[3].

We evaluated the algorithm by comparing it against a manually anonymized data set of 150 reports. All dates and identifiers were correctly anonymized. Person name anonymization resulted in an average precision of 0.931 and recall of 0.962. In absolute terms, out of 928 names in the test set 45 were not anonymized. While the results are already very promising, it is unclear if a single instance of failed anonymization is legally allowable for such sensitive information. For example, the algorithm used to anonymize the MIMIC-CXR radiology reports did not detect 8 of 9,778 words containing personal information in a manually annotated test set of 2,238 reports. To improve recall without sacrificing precision, future work could aim to fine-tune a NER model on chest radiology reports.

[1] With the exception of burned-in information. However, due to the contrast differences between background and text, a simple machine learning model should be able to redact this information [99].

[2] `https://huggingface.co/flair/ner-german-large`

[3] `https://flexikon.doccheck.com`

4.2 German CheXpert Chest X-ray Radiology Report Labeler

Most of the work presented in this section has been submitted as part of an article to the Röfo journal on June 1st, 2023, the paper is currently under review (as of June 2023) [101].

4.2.1 Introduction

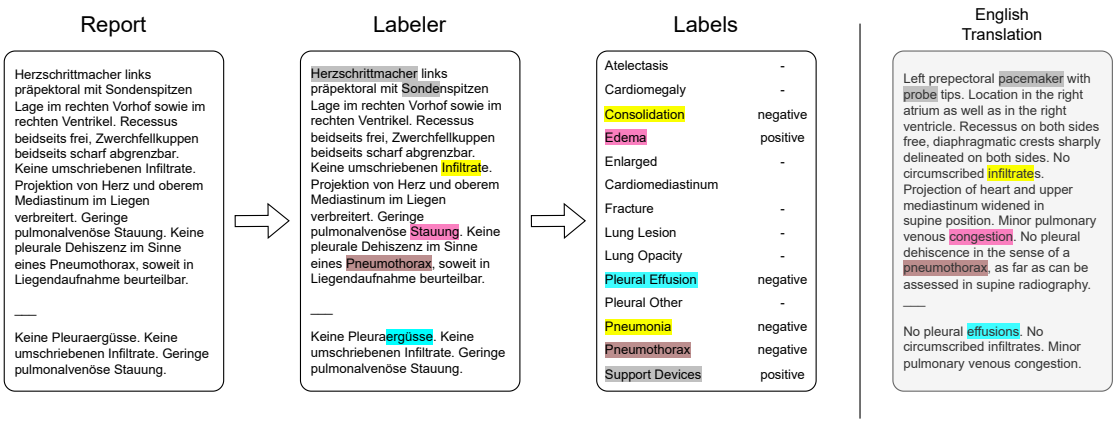

Figure 4.1: Automated labeling of German thoracic radiology reports. A report is passed to the report labeler and converted to 14 labels, motivated by CheXpert. The labeler detects each class according to class-specific phrases and converts them to positive, negative, or uncertain labels.

Chest X-rays are a frequently used and essential tool for detecting lung pathologies, like pneumothorax [5, 6]. The accurate interpretation of chest X-rays can be essential for the early detection, timely diagnosis, and effective treatment of these conditions. However, due to the large number of radiological images, radiology departments in many countries and regions are understaffed or overworked, ultimately risking the quality of care [102, 26, 24].

Recently, deep learning models used in decision support systems have achieved performance levels in chest X-ray diagnosis of pathologies like pneumonia that are comparable to those of radiologists [14, 89]. The integration of such models into clinical systems could reduce repetitive work, decrease workload, and improve the diagnostic accuracy of radiologists.

One of the reasons for the recent surge of innovation based on deep learning models is the availability of large data sets. For example, the release of the ImageNet data set and the according image classification competition, led to huge improvements in the computer vision domain [103, 30, 59, 72]. Similarly, the release of the chest X-ray 14 data set [78] sparked the development of chest X-ray classification models like CheXnet [13]. While the number of images used by modern deep learning architectures increased over the years, large publicly available data sets required for new architectures such as Vision Transformers [104] are missing in radiology, limiting the use of advanced models [94] and inhibiting advances in automated chest X-ray diagnosis.

Radiology departments around the world create large amounts of chest X-ray image data with corresponding reports during clinical routine. Despite the existence of huge numbers of radiological imaging studies and their radiological reports stored in the Picture Archiving and Communication Systems (PACS) of numerous clinics, only few are used for the development of new deep learning models, due to missing infrastructure, data privacy considerations, and required time, among others.

Unlike commonly used image data sets, such as ImageNet, chest X-ray data sets obtained from clinical routine require expert annotation due to the specialized knowledge required to understand the images. This annotation task falls on radiologists, who possess the necessary training and expertise to accurately interpret the X-rays. While decision support systems for chest X-ray diagnosis aim to reduce the workload of radiologists, a significant challenge arises from the need for radiologists to perform the time-consuming task of data annotation. This creates a "chicken-and-egg" problem, where the development of decision support systems depends on large, annotated data sets, yet creating these data sets requires significant time and effort from radiologists.

To reduce the amount of time needed for data annotation, natural language processing systems have been created for extracting structured labels from free-text radiology reports. Such systems can be primarily categorized as rule-based or deep learning-based approaches, each of which has its own benefits and limitations. Rule-based systems, for instance, are easier to implement, require no computationally intensive training, provide higher explainability, and can be easily updated with new rules and classes by anyone. On the other hand, deep learning-based approaches primarily rely on large language models, and thus have the potential to produce more accurate label predictions but require more computational resources and larger (manually) annotated data sets. Furthermore, they can only be developed and maintained by experts.

Recent public chest X-ray data sets such as chest X-ray 14, CheXpert [80] and MIMIC-CXR [99] were created by converting existing radiological reports to class labels automatically using rule-based systems. For example, the CheXpert labeler converts an existing report to the thirteen classes: atelectasis, cardiomegaly, consolidation, edema, enlarged cardiomediastinum, fracture, lung lesion, lung opacity, pleural effusion, pleural other, pneumonia, pneumothorax, support devices, and an additional "no finding" class. To minimize development time, the CheXpert labeler was used to annotate the MIMIC-CXR data set as well. Moreover, this labeler has been adapted and ported to process reports in other languages, such as Brazilian [87] and Vietnamese [105]. The process of labeling consists of three stages (see Figure 4.3): In the first stage, mention extraction, the labeler scans the input text for phrases defined in class-specific lists. For example, the phrase list for pneumothorax contains phrases such as "pneumothorax" and "pleural dehiscence". Next, extracted mentions are classified as positive, negative, or uncertain during the second stage (mention classification). Finally, an observation label is created by aggregating all its mentions together (mention aggregation). If a report happens to mention no observation, except support devices, the report is instead labeled as "no finding".

For German radiology reports, Nowak et al. investigated different approaches for training a deep-learning based labeling model [106]. In contrast to the CheXpert labeler,

their model predicted only the six observations: pulmonary infiltrates, pleural effusion, pulmonary congestion, pneumothorax, regular position of the central venous catheter (CVC) and misplaced position of the CVC. So far, neither source code nor model weights were released.

In this work, we propose an automatic labeler for German thoracic reports based on the CheXpert algorithm (shown in Figure 4.1). Our contributions are:

- We created a rule-based labeling algorithm for converting German thoracic radiology reports to CheXpert labels.
- We propose a web-based annotation tool for radiologists to adapt the labeler to new phrases used in a specific clinic and create a ground truth data set.
- We demonstrated that our proposed labeler performs similarly to radiological report labelers in other languages. In addition, we showed that a pneumothorax classifier trained on weakly labeled data outperforms models trained solely on publicly available data, and competitively to manually labeled data.

4.2.2 Materials and Methods

4.2.2.1 Data Collection

We retrospectively identified thoracic radiology reports from 2020 to 2021 in our institutional PACS and randomly selected 900 reports for the creation of a reference standard and 186 reports for phrase collection and development. In the following, we refer to this data set as data set 1 (DS 1). Initially, two radiologists, one board-certified radiologist with more than ten years' experience (B.S.), and one first year radiology resident (S.H.) from Klinikum der Ludwig-Maximilians-Universität München compiled a list of common phrases for each of the fourteen CheXpert classes. During the following data annotation process the list of phrases was expanded, including positive, negative, and uncertain phrases.

4.2.2.2 Data Annotation

To make the labeling of data set 1 as efficient and accurate as possible, we built a multi-user web-based labeling interface. The design and implementation respect patient data privacy by running the process locally in a secure environment.

The annotation tool, shown in Figure 4.2, displays the view position and report text on the left side of the screen, with four selectable labeling options available per pathology on the right. These options conform to the original CheXpert architecture and include positive, negative, uncertain, and none, which is used if the specific class was not mentioned. Radiologists can add new class-specific phrases by selecting "add new" and mark and comment on a report for later review. Before saving the annotations, the application highlights the phrases that were recognized by the labeler but were marked as "none" and prompts for a phrase if a class was selected during annotation, but not recognized by the labeler, thereby improving the phrase lists.

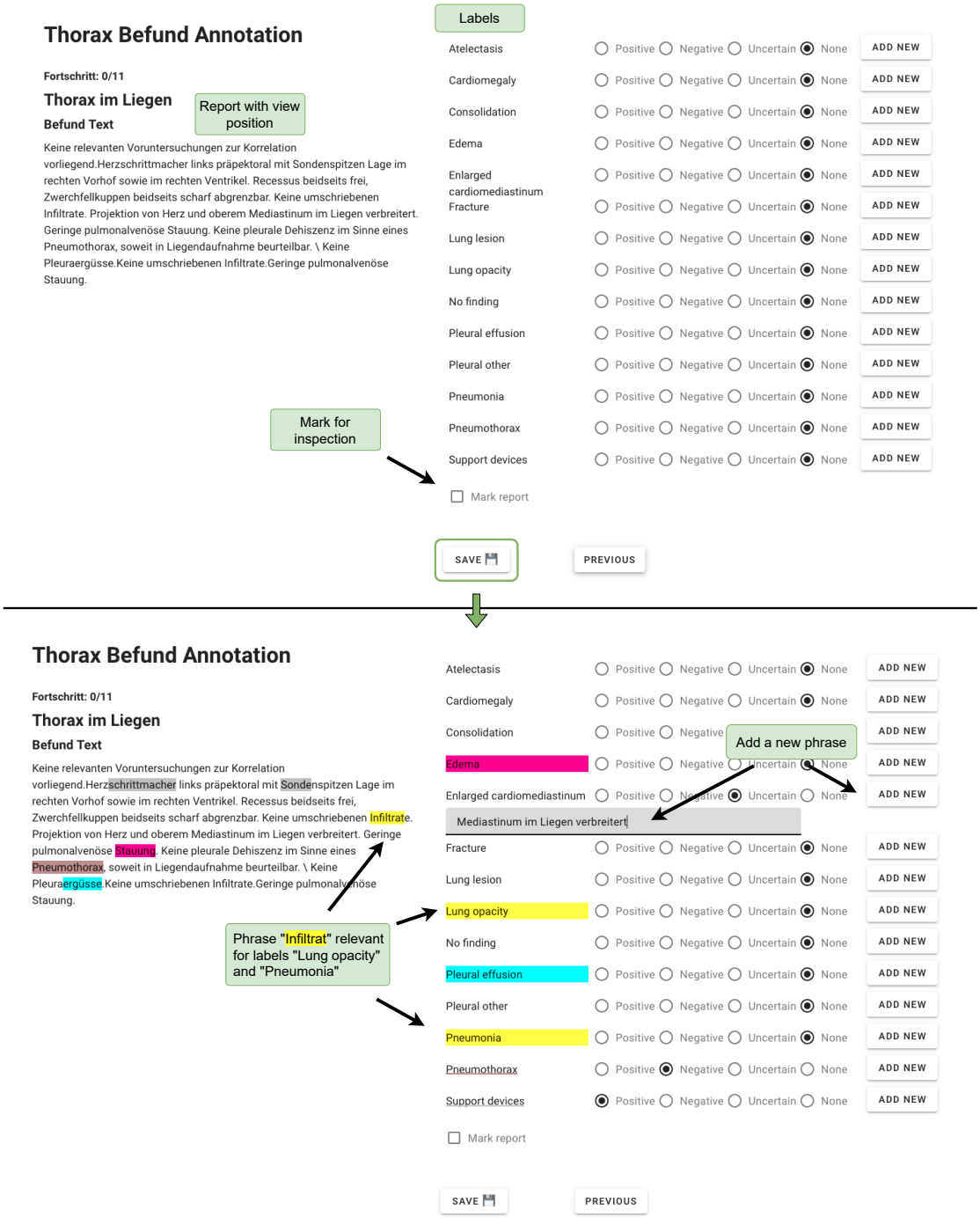

Figure 4.2: Report annotation web interface. Top: On the left side view position and report are displayed, on the right the 14 labels can be selected. Additionally, new phrases can be added by clicking "ADD NEW" and a report can be marked for later inspection. Bottom: After clicking "SAVE" the tool highlights the matching phrases with their corresponding labels and asks for a phrase when the selected class was not found by the labeler. Clicking "SAVE" again will save the annotation and load the next report.

Dataset	Data Set 1 (DS 1)					
Split	Development			Test		
Reports	186			900		
Class	P	U	N	P	U	N
Atelectasis	29	17	1	203	50	2
Cardiomegaly	34	56	41	166	338	248
Consolidation	17	28	115	210	23	552
Edema	61	3	74	259	11	478
Enlarged Cardiom.	39	42	52	206	273	277
Fracture	11	1	12	61	4	75
Lung Lesion	11	1	1	37	11	12
Lung Opacity	31	27	112	275	20	484
No Finding	24	-	-	121	-	-
Pleural Effusion	72	7	90	411	49	390
Pleural Other	11	3	-	53	18	1
Pneumonia	4	48	114	52	142	578
Pneumothorax	27	1	147	62	11	786
Support Devices	108	-	17	523	2	101

Dataset	Data Set 2 (DS 2)					
Split	Training		Validation		Test	
Reports	4507		660		1267	
Class	P	N	P	N	P	N
Pneumothorax	1122	3385	204	456	326	941

Table 4.1: Data sets with data splits and annotated classes used in this study. Data set 1 class annotations were acquired using our proposed annotation interface from free text reports. Data set 2 class annotations were acquired from reports and radiographs [107]. Enlarged Cardiom. = Enlarged Cardiomediastinum, P = Positive, U = Uncertain, N = Negative.

To evaluate the labelers performance and expand the class pattern list, one first year radiology resident (S.H.) from Klinikum der Ludwig-Maximilians-Universität München annotated the 1086 randomly selected radiology reports of data set 1 using our proposed annotation interface. The resulting class distribution is listed in Table 1.

4.2.2.3 Report Labeler

In German radiology reports, two distinct types of negations were identified: expressions that contain phrases like "nicht" or "kein" ("no", "not") and are observation-independent, which can be resolved by the German NegEx algorithm [108]. The other class comprises medical terms that lack any negations but convey the lack of an observation, for example, "Herz normal groß" ("regular heart size"). As the CheXpert architecture addresses only negated observations, we extended the architecture by using multiple phrase files (positive, negative, uncertain) per observation.

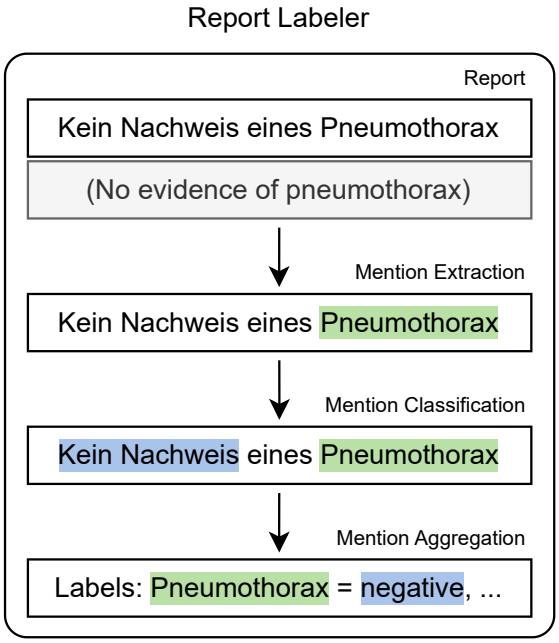

Figure 4.3: Labeling flow from our proposed report labeler based on the CheXpert architecture. The report is first matched against a set of class-specific phrases. Afterwards, each match is classified as positive, negative, or uncertain. If the report did not match any phrase it is labeled as no finding in the final stage. English translation provided below the German report excerpt.

As the original mention classification stage depends on an extensive rule set created for English report texts, our labeler utilizes a modified version of the German NegEx algorithm to classify German mentions instead. In the first step, the labeling algorithm identifies negation phrases such as "kann ausgeschlossen werden" ("can be excluded"), and uncertainty phrases, such as "unwahrscheinlich" ("unlikely"), based on a set of rules and marks them as pre- or post-negation/uncertainty phrases.

To identify whether the classification of a mention is affected by negation/uncertainty terms, a cut-off radius determines how many words before and after the mention are taken into consideration, following the German NegEx algorithm. If the relevant region around the mention does not contain any known negation/uncertainty phrases, the mention is classified as positive. If either a pre-negation or post-negation is found near the mention, it is classified as negative. Finally, if there is an uncertainty phrase in the surrounding region, the mention is classified as uncertain.

To form the final label for each observation, the results from mention classification are aggregated as shown in Figure 4.4. The following rules are applied to derive the labels:

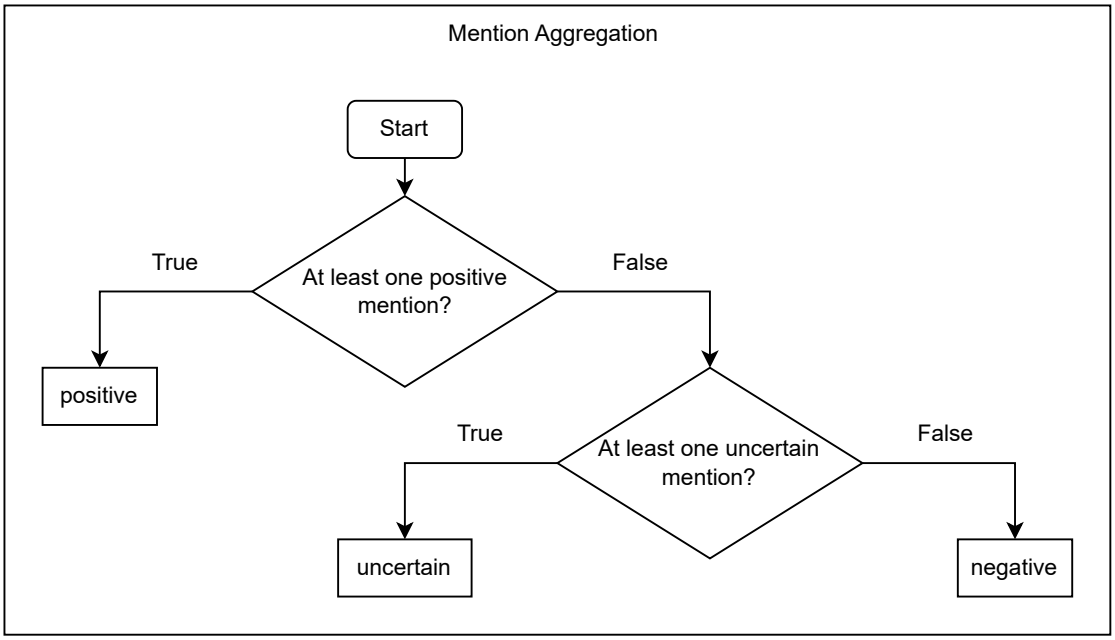

Figure 4.4: Derivation of class labels by aggregating all classified mentions per observation. Since an observation can be mentioned multiple times in a report, they must be aggregated for classification.

1. Observations with at least one positive mention are assigned a positive label.

2. Observations with no positive mentions and at least one uncertain mention, are labeled as uncertain.

3. Observations with no positive or uncertain mention or at least one negative mention are classified as negative.

The "no finding" label follows a different logic. Initially, a report is labeled as "no finding". The label is changed to negative if any of the other observations (excluding "support devices") are labeled as positive or uncertain.

The main benefit of automated label extraction is time savings. Our proposed algorithm features low memory consumption and enables parallel labeling of multiple reports using multi-threading. Using twelve threads the algorithm labeled 100 reports on average in 1.84 s $\pm$ 27.3 ms on a workstation equipped with an Intel i7-6800K CPU with a clock speed of 3.40GHz.

4.2.2.4 Label Extraction (DS 1)

Label extraction performance was measured by comparing extracted and annotated labels on DS 1 on three tasks: mention extraction, negation detection, and uncertainty detection. Regarding the mention extraction task, unlabeled findings ("none") were considered as negative, annotated ("positive", "negative", or "uncertain") as positive. For

negation detection, findings annotated as negative were considered as positive, others as negative. For uncertainty detection, annotations were classified analogously. The phrase lists were optimized on the development subset of DS 1. Phrases that were collected during the test subset annotation were discarded to avoid overfitting.

4.2.2.5 Pneumothorax Classification (DS 2)

To measure the effect of automatically extracted labels on downstream model training and classification performance we extracted pneumothorax labels from the radiology reports of an additional internal data set [107]. In the following, we refer to this data set as data set 2 (DS 2). This data set consists of 6434 frontal chest radiographs and their reports, out of which 1568 have been labeled as pneumothorax. Unlike DS 1, the labels are based on radiographs rather than solely reports, and as such, no uncertainty or "none" label are available. We converted the extracted labels to binary labels by considering uncertain cases as positive. For comparison, we applied the same conversion to DS 1 labels and annotations. Additionally, "none" annotations were considered as negative.

We used a DenseNet-121 pre-trained on ImageNet as backbone for our network. We replaced the final fully connected layer with a single output when fine-tuned on DS 2. We replaced the final softmax activation with a sigmoid. We used ADAM with a learning rate of 0.003 and a batch size of 32 and trained for 10 epochs. For our experiments, we selected the best checkpoint based on the validation area under the receiver operating characteristic curve (AUC). All images were normalized according to the ImageNet mean and standard deviation and resized to 224x224 pixels. For data augmentation we applied ten-crop. For our experiments, we compared a DenseNet-121 fine-tuned on the chest X-ray 14 data set (CheXnet) and fine-tuned on DS 2. When fine-tuning on DS 2 we trained with either radiologists' annotations (annotated) or automatically extracted labels (extracted).

4.2.2.6 Statistical Evaluation

We evaluated the labeler's performance using F1 score, precision, and recall regarding mention extraction, negation detection, and uncertainty detection by comparing the extracted to the annotated labels from DS 1. We evaluated pneumothorax classification performance using receiver operating characteristics (ROC) and AUC. Because our study is exploratory and involves multiple comparisons, we refrained from providing P values and provide 95 % confidence intervals calculated using the non-parametric bootstrap method with 10,000-fold resampling at the image level. The labeler performance with respect to the binary pneumothorax labels of DS 2 was measured using sensitivity and specificity. For comparison of DS 1 and DS 2, we converted DS 1 labels and annotations to binary labels and measured sensitivity and specificity. The statistical analyses in this study were done using NumPy version 1.24.2 and Scikit-Learn version 1.2.2.

Data Set 1	Mention Extraction			Negation			Uncertainty		
Findings	F1	R	P	F1	R	P	F1	R	P
Atelectasis	0.968	0.96	0.976	N/A	N/A	N/A	0.648	0.7	0.603
Cardiomegaly	0.813	0.71	0.952	0.627	0.528	0.771	0.683	0.551	0.898
Consolidation	0.933	0.919	0.947	0.884	0.802	0.984	0.4	0.609	0.298
Edema	0.993	0.996	0.991	0.965	0.941	0.989	0.48	0.545	0.429
Enlarged Cardiom.	0.867	0.807	0.937	0.678	0.569	0.84	0.725	0.607	0.902
Fracture	0.838	0.856	0.821	0.713	0.554	1.0	N/A	N/A	N/A
Lung Lesion	0.8	0.833	0.769	0.917	0.917	0.917	0.385	0.455	0.333
Lung Opacity	0.92	0.915	0.926	0.851	0.743	0.994	0.364	0.6	0.261
No Finding	0.238	1.0	0.135	N/A	N/A	N/A	N/A	N/A	N/A
Pleural Effusion	0.99	0.985	0.995	0.948	0.938	0.958	0.5	0.429	0.6
Pleural Other	0.864	0.792	0.95	N/A	N/A	N/A	0.8	0.778	0.824
Pneumonia	0.902	0.829	0.988	0.862	0.771	0.976	0.705	0.612	0.833
Pneumothorax	0.995	0.999	0.991	0.981	0.978	0.985	0.353	0.273	0.5
Support Devices	0.939	0.92	0.96	0.842	0.762	0.939	N/A	N/A	N/A

Table 4.2: F1 Score, precision and recall for the three evaluation tasks of our report labeler: mention extraction, negation detection, and uncertainty detection for each finding. Labels were extracted from DS 1 and compared to manual annotations. Enlarged Cardiom. = Enlarged Cardiomediastinum, F1 = F1 Score, R = Recall, P = Precision.

4.2.3 Results

4.2.3.1 Label Extraction (DS 1)

The mention extraction, negation detection, and uncertainty detection results are shown in Table 4.2. Excluding the special case "no finding", mention extraction F1 score ranged from 0.8 to 0.995, negation detection F1 score from 0.624 to 0.981, and the uncertainty detection F1 score from 0.353 to 0.725. The special case "no finding" covers both reports that describe a normal chest radiograph and is the default label when the labeler does not find anything. Since blank "none" labels are considered negative for the mention extraction task, the precision reflects the labeler not finding any mention in the report. Results marked as "N/A" have insufficient samples for calculation.

Commonly, chest X-ray classification models are trained on binary labels. Following Irvin et al. [80], we treat uncertain labels as positive and obtain sensitivity and specificity results as reported in Table 4.3.

4.2.3.2 Pneumothorax Label Extraction (DS 2)

The labeler extracted pneumothorax labels from DS 2 reports with a sensitivity of 0.997 [95 % CI: 0.994, 0.999] and specificity of 0.991 [95 % CI: 0.988, 0.994], see Table 4.3. Differences between pneumothorax sensitivity and specificity on DS 1 and DS 2 can be explained by the underlying annotation. Uncertain DS 1 annotations were considered as positive, missing ("none") annotations as negative.

Findings	Data Set 1 Sensitivity	Specificity
Atelectasis	0.944 [0.915-0.970]	0.988 [0.978-0.995]
Cardiomegaly	0.680 [0.639-0.721]	0.909 [0.880-0.936]
Consolidation	0.952 [0.923-0.978]	0.892 [0.868-0.914]
Edema	0.970 [0.948-0.989]	0.946 [0.928-0.963]
Enlarged Cardiom.	0.767 [0.727-0.803]	0.793 [0.754-0.831]
Fracture	0.954 [0.897-1.000]	0.959 [0.945-0.972]
Lung Lesion	0.792 [0.667-0.900]	0.986 [0.978-0.993]
Lung Opacity	0.979 [0.962-0.993]	0.859 [0.831-0.886]
No Finding	0.736 [0.653-0.813]	0.983 [0.974-0.991]
Pleural Effusion	0.965 [0.947-0.981]	0.968 [0.951-0.984]
Pleural Other	0.789 [0.688-0.881]	0.998 [0.994-1.000]
Pneumonia	0.874 [0.825-0.920]	0.977 [0.966-0.987]
Pneumothorax	0.819 [0.727-0.904]	0.979 [0.969-0.988]
Support Devices	0.902 [0.876-0.927]	0.906 [0.876-0.935]

Findings	Data Set 2 Sensitivity	Specificity
Pneumothorax	0.997 [0.994, 0.999]	0.991 [0.988, 0.994]

Table 4.3: Sensitivity and specificity for the extracted labels compared to the reference annotations on DS 1 and DS 2 with corresponding 95 % confidence intervals. To create binary labels, uncertain labels/annotations were considered positive, "none" negative. Enlarged Cardiom. = Enlarged Cardiomediastinum.

4.2.3.3 Pneumothorax Classifier

The ROC curves and corresponding AUC values for the pneumothorax classification models trained on our internal data set with manually annotated labels or extracted labels and trained on the chest X-ray 14 data set are shown in Figure 4.5. Training with manually annotated labels from multiple readers performed best with an AUC of 0.934 [95 % CI: 0.918, 0.949], followed by the model trained with labels extracted automatically with our labeler with an AUC of 0.858 [95 % CI: 0.832, 0.882]. The CheXnet model trained on chest X-ray 14 data performed worst with an AUC of 0.728 [95 % CI: 0.694, 0.760].

4.2.4 Discussion

In this study, we proposed an automatic label extraction algorithm for German thoracic radiology reports. Our deep learning model trained on extracted labels demonstrated strong improvements compared to the CheXnet model (0.728 vs.0.858 AUC) and competitive performance compared to training with manually annotated data (0.858 vs. 0.934 AUC), as shown in Figure 4.5. This indicates a promising alternative to manual annotation of the training data, especially as the training data set size can be easily

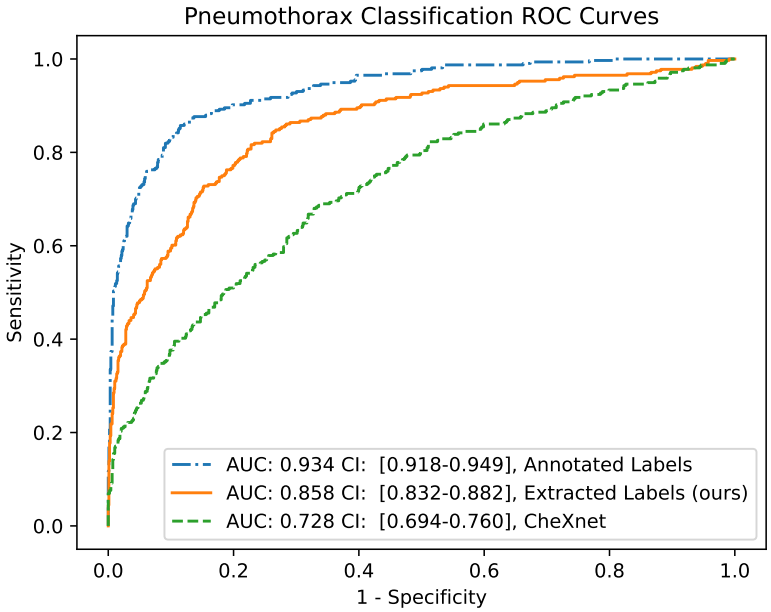

Figure 4.5: Receiver operating characteristic (ROC) curves and areas under the ROC curve (AUC) for pneumothorax classification on chest radiograph on our internal data set (DS 2). The model was trained on public data (CheXnet), on the DS 2 training data with either manual annotation (Annotated Labels) or labels extracted using our report labeler (Extracted Labels).

scaled with our proposed method. We expect better performance with larger training data sets, allowing for the use of more advanced model architectures, as larger training data sets generally improve image classification performance [104].

Although the extracted pneumothorax labels from DS 2 had a high label sensitivity and specificity of over 99 % (see Table 4.3), the larger classification AUC difference by the deep learning model trained on manual and extracted labels could be explained by the effect of noisier labels, making it harder to generalize. Creating class labels from radiological reports will always be inferior to the additional inspection of the image and a manual annotation. While pneumothorax label specificity is similar on both data sets, the sensitivity is considerably lower on data set 1, with a larger confidence interval. We interpret this difference as the effect of converting uncertain predictions to positives, as the uncertainty detection F1 score is comparatively low (see Table 4.2). While greater annotation quality resulted in better label extraction performance, it must be balanced with the time required to create such annotations. The results of our work show that our proposed labeler is a promising tool for clinical data scientists to create data sets.

Our label extraction algorithm was successful in identifying corresponding labels in DS 1 across all classes. The results are in line with other methods proposed in the literature [78, 80, 99]. Based on our experience, collecting labeling phrases using the proposed interface and the labeler results, we assume that the method can be easily applied to radiology reports from other clinics. Hence, additional classes can be incorporated quickly.

Furthermore, the annotation speed can be greatly improved by running the labeler first. Multiple readers could lower the risk of overlooking classes missed by the labeler.

During the process of annotating radiological reports based on the 14 CheXpert class labels, the radiologists commented that not all class labels were equally simple to annotate. In particular, the class "pleural other" was considered too vague for meaningful evaluation. Although the CheXpert labels were chosen based on the glossary of terms for thoracic imaging from the Fleischner Society [109], some of these labels lacked clear definitions, which could lead to inconsistent annotation, particularly when multiple annotators are involved, especially the "uncertain" classification is arguably too vague to be effectively used for modeling. To address these issues, future work could leverage the proposed annotation tool to refine and expand the CheXpert classes, ensuring that the labels are clearly defined and precise.

Images from a single clinic cannot be representative for the global population. Most chest X-ray data sets that are currently publicly available, such as Chest X-ray 14, CheXpert, or MIMIC-CXR stem from U.S. clinics. By establishing a set of shared class labels and developing chest X-ray report labels for other languages, models build on multi-institutional data sets will be more robust and general. We hope that our work motivates further research in other languages.

One limitation of our work is that we evaluated the effect of automatically extracted labels on chest X-ray classification performance only for the pneumothorax case, not for others. Future work will evaluate the model on all fourteen classes. Another limitation is that the proposed labeler cannot handle semantically equivalent words due to its rule-based nature. In a follow-up work we plan to replace it with a more sophisticated language model. Finally, we observed that few radiology reports described several images. Hence, extracted labels might refer not to the chest radiograph but another image.

In conclusion, we showed that extracting CheXpert labels automatically from German chest X-ray radiology reports are a promising substitute for manual annotation. A pneumothorax model trained on these extracted labels demonstrated competitive performance compared to manually annotated data.

4.3 Automated Labeling of German Chest X-Ray Radiology Reports using Deep Learning

A pre-print of most of the methods and results presented in this chapter has been submitted to ArXiv and is being prepared for publication (as of June 2023).

4.3.1 Introduction

Radiologists are in short supply worldwide, and deep learning models hold promise for addressing this shortage, for example, as decision-support systems (see Section 1.4). However, training such models often requires large data sets that are expensive and time-consuming to manually label. To reduce the amount time for obtaining labeled data sets, automatic label extraction from radiology reports is a compelling option. Unfortunately, label extraction from radiology reports itself is a challenging task, for example, due to semantically similar words and missing annotated data.

Recent developments in the natural language processing (NLP) domain have proposed models that generate dense word vector representations [110, 111, 112, 63], which have shown to be effective in training deep learning models for a wide range of tasks such as translation [62] and named entity recognition [100]. Similar to the computer vision domain, these language models can be pre-trained on a general, large corpus and then fine-tuned on a target corpus that might be otherwise too small for training [74].

In the medical domain, language models have been successfully applied to extract labels from unstructured radiology reports. Smit et al. [113] improved upon their rule-based labeler for English radiology reports by using a BERT [112] language model as backbone. Similarly, Nowak et al. [106] investigated the use of BERT for German radiology reports. They compared a rule-based labeler to a deep learning model, trained with 18,000 manually annotated reports, rule-based extracted labels, and a combination of both.

In this work, we explore the potential of weak supervision of a deep learning-based label prediction model, using a rule-based labeler. In contrast to Nowak et al., we focus on the classes of the CheXpert data set [80] (see Section 3.1.5), allowing for comparison with Smit et al. [113]. Our study builds upon previous work that used rule-based strategies to extract labels [101], presented in the previous section. We conduct extensive experiments on a dataset of internal radiology reports, and our results demonstrate the effectiveness of our approach.

4.3.2 Materials and Methods

4.3.2.1 Data Collection

We retrospectively identified 66,071 thoracic radiology reports from 2017 to 2021 in our institutional PACS. Additionally we used 1,091 thoracic radiology reports from 2020-2021 that were manually annotated by a first-year radiology resident from Klinikum der Ludwig-Maximilians-Universität München in a previous study [101].

Split	Train			Test		
Finding	Positive	Uncertain	Negative	Positive	Uncertain	Negative
Atelectasis	220	54	2	12	13	1
Cardiomegaly	184	368	266	16	25	25
Consolidation	205	45	627	23	6	41
Edema	297	9	521	24	5	34
Enlarged Cardiom.	223	295	305	22	19	26
Fracture	63	3	79	9	2	8
Lung Lesion	44	7	8	5	5	5
Lung Opacity	278	41	565	28	6	35
No Finding	1	0	0	1	0	0
Pleural Effusion	455	45	451	29	11	32
Pleural Other	57	16	1	7	5	0
Pneumonia	52	173	649	5	16	45
Pneumothorax	83	7	871	5	5	66
Support Devices	590	1	107	43	1	12

Table 4.4: Overview of the class distributions of the manually annotated training and test data. Enlarged Cardiom. = enlarged cardiomediastinum.

The training and test set label distributions of the manually annotated reports is reported in Table 4.4. Since annotated "no finding" reports describe normal appearing chest radiographs, there are no negative or uncertain annotations. To ensure sufficient representation of each class during testing, we selected 78 of the 1,091 manually annotated reports. This ensured that each class was mentioned by at least five reports. To increase the number of training samples, we favored test samples with multiple annotations. In cases where the entire data set contained less than five samples for a specific class, half of the samples were designated for testing. None of the manually annotated reports used for testing were part of the 66,071 reports.

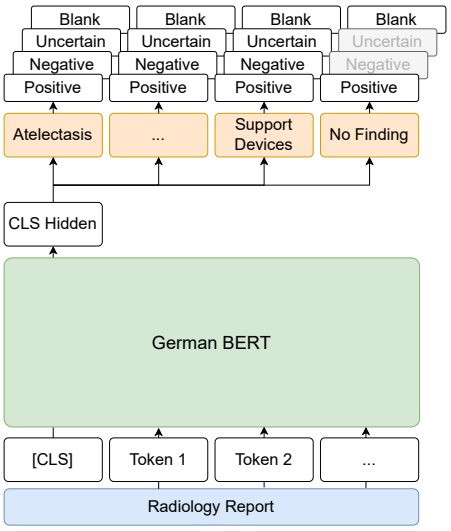

Figure 4.6: Architecture of our proposed label extraction model.

4.3.2.2 Model Architecture

Following Smit et al. [113] we used a pre-trained BERT [112] model as backbone for our label extraction model. The objective of the model is to predict the fourteen CheXpert labels: atelectasis, cardiomegaly, consolidation, edema, enlarged cardiomediastinum, fracture, lung lesion, lung opacity, pleural effusion, pleural other, pneumonia, pneumothorax, support devices, and "no finding" given a German radiology report [80]. The model re-

ceives the report as input and assigns one of the classes: blank, positive, negative, or uncertain to each of the 13 observations, mirroring the manual annotation. The blank represents no mention of the class in the report. For the special case "no finding", which corresponds to the absence of any of the findings, the labeler must predict only blank or positive.

We modified the BERT architecture by using 14 linear heads, as illustrated in Figure 4.6 [113]. Each head is dedicated to capture one of the 14 labels. For transfer learning we use the pre-trained "bert-base-german-cased" BERT model[4] trained on German texts, such as the German Wikipedia corpus [114] with a sequence length of 512.

To predict the classes of the 14 findings, the radiology reports were first tokenized. Of all tokenized reports, a single report in the training data, and none in the test data consisted of more than 512 tokens. The overflowing report consisted of 579 tokens and described multiple images in each report. We considered only the first 512 tokens of this report. After tokenization, the reports were processed by the model. Subsequently, the hidden state of the class (CLS) token from the final layer was used as the input for each of the 14 linear heads, predicting the class of each finding via a softmax.

The model was fine-tuned using cross-entropy loss, AdamW [54] optimization with default parameters ($\beta_1 = 0.9, \quad \beta_2 = 0.999$), a learning rate of 2e-5, and a batch size of 8. The individual cross-entropy losses for the 14 observations were aggregated before calculating the final loss. To monitor model performance, we periodically evaluated the model on the validation set and saved the best checkpoint across all 14 observations.

4.3.2.3 Experiments

We evaluated the deep learning-based labeler on three tasks: First, we investigated the benefit of weak labels on label extraction performance. The labels were created using the rule-based model proposed in a previous study [101] (see Section 4.2). For validation, we randomly sampled 1000 reports, and for internal testing 5000 reports, without patient overlap. We used the remaining 60,071 reports for training. The manually labeled reports were used only for final testing.

Run	Training	Validation	Test
25 %	203	51	78
50 %	406	101	78
75 %	608	152	78
100 %	810	203	78

Table 4.5: Training, validation, and test splits for the different training runs on manually annotated data.

As a second experiment, we trained the model solely on manually annotated reports. To assess the importance of the number of annotated reports, we split the training data into four quarters and trained on increasingly larger fractions, as reported in Table 4.5.

[4] https://huggingface.co/bert-base-german-cased

Finally, we fine-tuned the best weakly supervised model on the manually annotated reports. Again, we trained the model on increasing fractions of the manually annotated training data set, as reported in Table 4.5

For all three experimental settings we measured mean F1 scores for the three tasks of mention extraction, negation detection, and uncertainty detection by comparing model predictions with manually annotated test reports. Following the original CheXpert publication [80], findings labeled as "blank" were considered as negative for the mention extraction task and the other classes ("positive", "negative", or "uncertain") as positive. Regarding negation detection, only the "negative" classification was considered positive, and for uncertainty detection, only the "uncertain" class was considered positive.

4.3.3 Results

4.3.3.1 Weakly Supervised Training

When trained only with reports labeled by the German CheXpert labeler, the model achieved a mean F1 score of 91 % for mention extraction, 82 % for negation detection, and 52 % for uncertainty detection. Note that although the model was trained on automatically extracted labels, it was tested with manually annotated labels, as described in Table 4.4.

4.3.3.2 Supervised Training

	Supervised Training		
Run	Mention Extraction	Negation Detection	Uncertainty Detection
25 %	**0.87**	0.69	0.59
50 %	0.82	0.72	0.43
75 %	**0.87**	0.81	**0.61**
100 %	0.84	**0.83**	0.57

Table 4.6: Mean F1 scores when trained on increasing fractions of manually annotated reports. Highest F1 scores are highlighted in bold.

The results obtained, when trained solely on increasing fractions of manually annotated reports, are reported in Table 4.6. Mention extraction F1 scores ranged from 82 % to 84 %. Negation detection F1 scores increased from 69 % to 83 % when increasing the amount of training data. Mean uncertainty detection F1 scores ranged from 43 % to 61 %. Using 75% of the training data performed better than using all training data for both mention extraction and uncertainty detection.

4.3.3.3 Weakly-Supervised Pre-Training

The effect of pre-training with automatically labeled reports first and then fine-tuning on varying amounts of manually annotated data is reported in Table 4.7. Mention extraction results ranged from 93 % to 94 % mean F1 score, with a slightly higher score obtained

	Weakly-Supervised Pre-Training		
Run	Mention Extraction	Negation Detection	Uncertainty Detection
25 %	0.93	0.88	0.56
50 %	**0.94**	0.87	**0.64**
75 %	**0.94**	**0.89**	0.54
100 %	**0.94**	**0.89**	0.61

Table 4.7: Mean F1 scores when pre-trained on weakly labeled reports followed by fine-tuning on increasing fractions of manually annotated reports. Highest F1 scores are highlighted in bold.

with more training data. Similarly, negation detection F1 scores improved when using more manually annotated training data, F1 scores ranged from 0.88 % to 89 %. Mean uncertainty detection F1 scores ranged from 56 % to 64 %.

4.3.3.4 Comparison of Rule-Based and Deep Learning-Based Label Extraction

	Mention Extraction		Negation Detection		Uncert. Detection	
Finding	RB	DL	RB	DL	RB	DL
Atelectasis	**0.982**	0.963	**1.0**	N/A	**0.769**	0.700
Cardiomegaly	0.667	**0.955**	0.649	**0.898**	0.571	**0.809**
Consolidation	0.950	**0.979**	0.746	**0.911**	0.400	0.400
Edema	**0.992**	0.985	0.939	**0.955**	**0.600**	N/A
Enlarged Cardiom.	0.820	**0.933**	**0.800**	0.776	0.500	**0.821**
Fracture	0.900	0.900	0.545	**0.857**	N/A	N/A
Lung Lesion	0.857	**0.938**	0.889	0.889	0.182	**0.714**
Lung Opacity	0.936	**0.952**	0.667	**0.853**	**0.316**	N/A
No Finding	**0.025**	N/A	N/A	N/A	N/A	N/A
Pleural Effusion	0.973	**0.974**	0.954	**0.955**	0.556	**0.706**
Pleural Other	**0.857**	0.737	N/A	N/A	**0.500**	0.333
Pneumonia	0.922	**0.964**	0.892	**0.966**	0.600	**0.688**
Pneumothorax	0.987	**0.994**	**0.964**	0.950	**0.571**	0.333
Support Devices	**0.972**	0.962	**0.800**	0.762	N/A	N/A
Mean	0.91	**0.94**	0.82	**0.89**	0.51	**0.61**

Table 4.8: Comparison of rule-based and deep learning-based label extraction on mention extraction, negation detection, and uncertainty detection measured with the F1 score. The deep learning-based model was first trained on the weakly labeled reports and then fine-tuned on all manually labeled training data. Highest mean F1 scores are highlighted in bold. RB = rule-based labeler, DL = deep learning-based labeler, Uncert. Detection = uncertainty detection, Enlarged Cardiom. = enlarged cardiomediastinum.

To assess the benefit of employing a deep learning model for label extraction, we compared the results of the rule-based German CheXpert labeler with the proposed deep learning-based model. The deep learning-based model was first pre-trained on the

weakly labeled reports and then fine-tuned on the manually labeled data (100 % run). The results are reported in Table 4.8.

Across all thee tasks, the deep learning model performed better. For mention extraction, our proposed deep learning labeler had a mean F1 score of 94 % compared to 91 % of the rule-based labeler. For negation and uncertainty detection, the improvement of using a deep learning-based labeler compared to a rule-based model was even greater, with 89 % vs. 82 % mean F1 score for negation detection, and 61 % vs. 51 % mean F1 score for uncertainty detection.

4.3.4 Discussion

In this work, we proposed a deep learning-based CheXpert label prediction model. Our model was pre-trained on reports labeled by a rule-based German CheXpert model and fine-tuned on only a thousand manually labeled reports. On average, it significantly outperformed the rule-based model, on all three tasks, as shown in Table 4.8. Our results show that the improvements of deep learning-based label extraction compared to rule-based transfer from English to German radiology reports [113].

Similar to Nowak et al., the deep learning-based model outperformed the rule-based model on German reports. Apart from using different data sets, a direct comparison is made difficult, as Nowak et al. considered both uncertain and negative mentions as negative labels. Furthermore, their rule-based labeler achieved only an average classification F1 score of 75.1 %, compared to their deep learning-model with 95.5 %. Consequently, they observed that their automatically labeled reports did not improve the deep learning model performance. In contrast, our rule based-labeler served as strong baseline with an average mention extraction F1 score of 91 % and negation detection F1 score of 82 % (see Table 4.8). Consequently, pre-training with weak supervision improved the performance compared to only training on manually annotated data alone. For example, mean mention extraction F1 score improved from 84 % to 94 % when using all data.

In contrast to Nowak et al., our model was trained on only approximately one thousand manually labeled reports, compared to a total of 18,000 [106]. While they showed that increasing the amount of manually annotated training data improved mean F1 scores from 70.9 % to 95.5 % when increasing training data from 500 to 14,580 samples, annotating all 18,000 samples took 197 hours. Based on their results, we assume that increasing the number of manually annotated samples, could further improve our model.

Our study has several limitations. First, due to the limited number of available manually annotated reports, most data was used for training. A future study with more manually annotated data could both improve model performance and reduce the variation of test scores. Another limitation is that the labels were created only by a single radiologist, possibly introducing label biases or errors made due to annotation fatigue.

In conclusion, we demonstrated a significant improvement in German radiology report labeling using our proposed deep learning-based labeler, achieving a new state-of-the-art on this data set. Our results provide evidence of the benefits of employing a deep learning-based model, even in scenarios with sparse data, and the use of the rule-based labeler as a tool for weak supervision.

5 WindowNet: Learnable Windows for Chest X-ray Classification

A pre-print of most of the methods and results presented in this chapter has been submitted to ArXiv and is being prepared for publication (as of June 2023).

5.1 Introduction

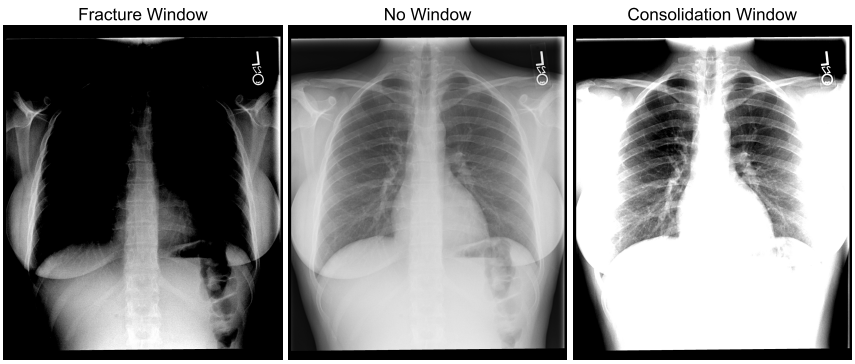

Figure 5.1: Applying a windowing operation enhances the contrast of particular structures of an image. The left window improved fracture, and the right window consolidation classification performance of our model compared to no windowing (middle) on the MIMIC data set.

Chest X-rays (CXR) are commonly acquired with a high resolution and bit depth. For example, the images of the MIMIC data set [99] (see Section 3.1.6) have approximately a 2500×3056 pixel resolution with 12-bit depth gray values. To reduce file size, these images are often compressed to a lower resolution, for example, 1024×1024 and lower bit depth, for example, 8-bit [78].

Under optimal conditions the human eye can differentiate between 700 and 900 shades of gray, or 9 to 10 bit-depth [115]. To better differentiate subtle lines, radiologists apply a windowing operation to the image: they increase the contrast by limiting the range of gray tones. These window ranges can be described using their center (window level) and width (window width). Formally, the windowing operation applied to a pixel value

px can be defined as:

$$\begin{aligned} window(px) &= \min(U, \max(L, px)), \\ U &= WL + WW/2, \\ L &= WL - WW/2. \end{aligned}$$

Where U is the upper limit and L the lower limit of the window defined by the window level WL and window width WW.

For computed tomography (CT) images, the gray values are calibrated according to the Hounsfield unit (HU) scale [4, p. 155]. A HU value of -1000 corresponds to air, 0 HU to distilled water at standard pressure and temperature, and, for example, bones range from 400 HU to 3000 HU [4, p. 155]. On the screen, air is displayed in black, whereas bones appear white or light gray. For example, for a chest CT image, one could apply a window utilized to highlight the lung by showing everything below -700 HU as black and above -600 HU as white [116, p. 379]. Consequently, larger gray tone intervals can be used for the specified range, enhancing the contrast.

For CT images, several studies showed that windowing improves classification performance of classification models and that it can be used as trainable parameter of the network [117, 118, 119, 120]. While for chest radiographs no quantitative scale like the Hounsfield Unit exists, radiologists still window these image for enhanced contrasts. This observation leads to following research questions: does windowing affect chest X-ray classification performance and if yes, can windowing improve it? An example of applying a window operation to a chest radiograph ist shown in Figure 5.1. To the best of my knowledge, so far chest X-rays are commonly processed by a deep learning model without applying any windowing operation [13, 94]. This chapter investigates the effect of windowing on chest X-ray classification and proposes a model, WindowNet, that learns optimal windowing settings.

5.2 Methods

The MIMIC data set (see Section 3.1.6) provides chest radiographs in the original Digital Imaging and Communications in Medicine (DICOM) format with 12-bit depth gray values. We replaced the last fully connected layer of a pre-trained DenseNet-121 [59] with a fully-connected layer with a 14-dimensional output to train it on this data set and predict all 14 classes. We trained the models with binary cross-entropy loss, AdamW optimization [54] with a learning rate of 1e-4, and a batch size of 32. For all experiments, we resized the images to 224×224 pixel resolution, divided the learning rate by a factor of ten if the validation loss did not improve in three consecutive epochs, and stopped the training if the validation loss did not improve after five consecutive epochs. We selected the best checkpoint based on the mean validation AUC. After windowing, we normalized the image according to the ImageNet mean and standard deviation.

5.2.1 Bit-Depth

As applying a windowing operation in our experiments required a higher initial bit-depth than conventionally used for chest X-ray image classification, we first tested the effect of bit-depth on classification performance. We compared a DenseNet-121 trained on 8-bit and 12-bit depth images and measured class-wise AUC scores. In both settings, we passed the full image to the model, without any windowing operation.

5.2.2 Windowing

To investigate wether windowing has an effect on classification performance, we trained a pre-trained DenseNet-121 with a single windowing operation applied to the chest radiographs with 12-bit depth using windows determined by a grid-search. We trained the model with a window level of 100, and with levels ranging from 250 to 3500 in steps of 250. All levels were combined with window widths of 500, 1000, 1500, 2000, and 3000. For evaluation, we compared the mean AUC of each model to the baseline with no windowing, i.e., a window level of 2048 and width of 4096.

5.2.3 WindowNet

When inspecting radiographs, radiologists dynamically adjust the window settings depending on the region of interest. In other words, they use multiple window settings for a single image. To test if multiple windows improve classification performance we extend the DenseNet-121 architecture to incorporate a learnable windowing module. In the following, we refer to this model as WindowNet.

Similar to Lee et al. [121], we implemented the windowing operation as 1×1 convolution with clamping. Typically, the single channel chest radiographs are duplicated to comply with the three channels required by models pre-trained on natural photographs. As the pre-trained DenseNet-121 expects three input channels, we add an additional 1×1 convolution to account for it. After windowing, the image is normalized according to the ImageNet mean and standard deviation and passed to the pre-trained DenseNet-121. The windowing module can be implemented as follows:

```
class ConvWindow(nn.Module):
    """Windowing using 1x1 convolutions with clamping."""
    def __init__(self, widths, level, upper=255):
        super().__init__()
        self.conv = nn.Conv2d(1, len(widths), kernel_size=(1, 1), stride=1)
        self.conv2 = nn.Conv2d(len(widths), 3, kernel_size=(1, 1), stride=1)
        # initialize windows
        for i, (width, level) in enumerate(zip(widths, levels)):
            self.conv.weight[i].data.fill_(255/width)
            self.conv.bias[i].data.fill_(-255/width * (level - width / 2))
        self.upper = upper

    def forward(self, x):
        out = torch.clamp(self.conv(x), min=0, max=self.upper)
```

```
        out = self.conv2(out)
        return out

    def get_width_and_level(self):
        WW = (self.upper / self.conv.weight.detach()).flatten()
        WL = -(self.conv.bias.detach()
               / self.conv.weight.detach().flatten()) + WW / 2
        return WW, WL
```

We initialized the learnable windows with the set of top the three windows per pathology, found during the grid-search experiment. The selection was based on the validation results. The following 13 windows (level, width) were selected: (100, 3000), (1250, 1000), (1500, 3000), (1750, 2000), (1750, 3000), (2000, 2000), (2250, 2000), (2250, 3000), (2500, 2000), (2500, 3000), (2750, 3000), (3250, 1000), (750, 3000), and no window (2048, 4096). For comparison, we train the model without clamping in the ConvWindow layer and default initialization.

5.3 Results

5.3.1 Bit-Depth

Finding	12 Bit	8 Bit
Atelectasis	0.749	0.751
Cardiomegaly	0.774	0.770
Consolidation	0.742	0.740
Edema	0.833	0.831
Enlarged Cardiomediastinum	0.701	0.691
Fracture	0.710	0.664
Lung Lesion	0.682	0.680
Lung Opacity	0.690	0.680
No Finding	0.797	0.789
Pleural Effusion	0.879	0.883
Pleural Other	0.831	0.823
Pneumonia	0.698	0.659
Pneumothorax	0.828	0.802
Support Devices	0.888	0.868
Mean	0.772	0.759

Table 5.1: Effect of bit-depth on chest X-ray classification performance. A higher bit-depth improved AUC values for most classes.

The classification AUCs, when trained with 8-bit depth and 12-bit depth, are shown in Table 5.1. Training with 12-bit depth slightly improved classification performance (0.772 vs. 0.759 AUC).

5.3.2 Windowing

Finding	No Window	Best Window
Atelectasis	0.749 (2048, 4096)	**0.757** (2750, 3000)
Cardiomegaly	0.774 (2048, 4096)	**0.786** (1750, 3000)
Consolidation	0.742 (2048, 4096)	**0.744** (2500, 3000)
Edema	0.833 (2048, 4096)	**0.841** (1750, 3000)
Enlarged Cardiom.	0.701 (2048, 4096)	**0.734** (2250, 3000)
Fracture	**0.710** (2048, 4096)	0.706 (1000, 3000)
Lung Lesion	0.682 (2048, 4096)	**0.720** (2500, 3000)
Lung Opacity	**0.690** (2048, 4096)	**0.690** (2250, 3000)
No Finding	0.797 (2048, 4096)	**0.804** (2500, 3000)
Pleural Effusion	0.879 (2048, 4096)	**0.888** (2500, 3000)
Pleural Other	0.831 (2048, 4096)	**0.850** (2750, 3000)
Pneumonia	**0.698** (2048, 4096)	0.690 (1750, 3000)
Pneumothorax	0.828 (2048, 4096)	**0.832** (1750, 3000)
Support Devices	0.888 (2048, 4096)	**0.889** (2750, 3000)
Mean	0.772 (2048, 4096)	**0.775** (2500, 3000)

Table 5.2: Effect of windowing on chest X-ray classification AUCs. For each finding, the best performing window and the baseline using no windowing is reported. Highest AUCs values are highlighted in bold. Enlarged Cardiom. = enlarged cardiomegaly.

The results of training with windowed chest X-rays are reported in Table 5.2. They demonstrate that windowing improved chest X-ray classification AUCs for most classes (12/14). Furthermore, the class-wise AUC values suggest that the baseline without windowing is a good baseline across all classes but not optimal. Across all grid-searched windows, a window width of 3000 performed best.

To study different window settings across all findings, we compared the four best-performing windows to no windowing at all. The results are show in Table 5.3. Again, the class-wise AUC values suggest that the baseline performance, without windowing, is similar to the best windows, but is not optimal for every class.

5.3.3 WindowNet

The WindowNet results are shown in Table 5.4, comparing no windowing to our proposed WindowNet model. Overall, WindowNet outperformed the baseline model trained without windowing with a mean AUCs of 81.2 % compared to 79 % AUC. This finding holds for all classes, except fracture, where no windowing was slightly better (61.9 % vs. 61.5 % AUC). The windows learned after training are shown in Table 5.5.

	No Window	Window 1	Window 2	Window 3	Window 4
Level	2048	2500	1750	2750	2250
Width	4096	3000	3000	3000	3000
Finding					
Atelectasis	0.749	0.756	0.753	0.749	**0.757**
Cardiomegaly	0.774	0.783	**0.786**	0.774	0.777
Consolidation	0.742	**0.744**	0.743	0.742	0.740
Edema	0.833	0.830	**0.841**	0.833	0.831
Enlarged Cardiom.	0.701	**0.710**	0.700	0.701	0.686
Fracture	**0.710**	0.695	0.670	**0.710**	0.669
Lung Lesion	0.682	**0.720**	0.710	0.682	0.700
Lung Opacity	**0.690**	0.683	0.686	**0.690**	0.684
No Finding	0.797	**0.804**	0.800	0.797	0.798
Pleural Effusion	0.879	**0.888**	0.883	0.879	0.885
Pleural Other	0.831	0.841	0.820	0.831	**0.850**
Pneumonia	**0.698**	0.686	0.690	**0.698**	0.683
Pneumothorax	0.828	0.822	**0.832**	0.828	0.809
Support Devices	0.888	0.887	0.887	0.888	**0.889**
Mean (Validation)	0.804	**0.807**	0.802	0.805	0.803
Mean (Test)	0.772	**0.775**	0.772	0.772	0.768

Table 5.3: Best single window settings for chest X-ray classification. The class-wise AUCs of the four best performing windows (Window 1-4) and the baseline without windowing are reported. Additionally, mean AUCs from the validation split are provided. Highest AUC values are highlighted in bold. Enlarged Cardiom. = enlarged cardiomegaly.

5.4 Discussion

In this study, we investigated the importance of windowing as a pre-processing step, which is motivated by the practice of radiologists manually adjusting window settings during chest X-ray inspection.

First, we analyzed the effect of bit-depth on image classification, as public datasets are commonly down-scaled to 8-bit. Our results indicate that a higher bit-depth (12-bit) improves performance (77.2 % vs. 75.9 % AUC), as seen in Table 5.1.

At native 12-bit depth, we were able to investigate the potential usefulness of windowing as a pre-processing step. Testing different windowing settings in a grid-search revealed that, except for pneumonia and fracture, training with a distinct window performed better than no windowing at all (see Table 5.2). These results suggest that windowing is a useful pre-processing step for improving chest X-ray classification performance.

To further improve the performance gains obtained from windowing, we proposed WindowNet, a model that learns optimal windowing settings. Our experiments demonstrate our proposed model outperforms the baseline model without windowing, achiev-

Finding	No Windowing	WindowNet
Atelectasis	0.812	**0.829**
Cardiomegaly	0.814	**0.827**
Consolidation	0.808	**0.823**
Edema	0.891	**0.897**
Enlarged Cardiom.	0.745	**0.764**
Fracture	**0.619**	0.615
Lung Lesion	0.701	**0.744**
Lung Opacity	0.726	**0.745**
No Finding	0.855	**0.859**
Pleural Effusion	0.909	**0.918**
Pleural Other	0.721	**0.793**
Pneumonia	0.731	**0.750**
Pneumothorax	0.830	**0.886**
Support Devices	0.897	**0.918**
Mean	0.790	**0.812**

Table 5.4: Comparison of baseline ("No Windowing") and WindowNet AUCs for chest X-ray classification. Enlarged Cardiom. = enlarged cardiomegaly.

ing a mean AUC of 81.2 % compared to 77.2 % for the baseline 12-bit model. The improvement in performance provides evidence that our model is an effective solution for automatically determining the best window settings for chest X-ray images.

While our study's results are promising, limitations include the evaluation of a data set from a single institution. Further research is needed to establish generalizability to other data sets and institutions.

In conclusion, we believe our work offers an important contribution to the field of computer vision and radiology by demonstrating that windowing can significantly improve chest X-ray classification performance, as shown by our proposed model, WindowNet.

Window	Ininitialized (Level, Width)	Learned (Level, Width)
0	(100, 3000)	(90, 2780)
1	(750, 3000)	(820, 3310)
2	(1250, 1000)	(1060, 850)
3	(1500, 3000)	(1130, 2280)
4	(1750, 2000)	(1820, 2080)
5	(1750, 3000)	(2200, 3770)
6	(2000, 2000)	(1980, 1980)
7	(2048, 4096)	(2010, 4120)
8	(2250, 3000)	(2420, 3230)
9	(2250, 2000)	(2280, 2030)
10	(2500, 3000)	(2700, 3240)
11	(2500, 2000)	(2040, 1630)
12	(2750, 3000)	(3250, 3540)
13	(3250, 1000)	(3450, 1060)

Table 5.5: Initial and learned windows after training, rounded to the nearest ten.

6 Chest X-Ray Resolution

A pre-print of most of the methods and results presented in this chapter has been submitted to ArXiv and is being prepared for publication (as of June 2023).

6.1 Introduction

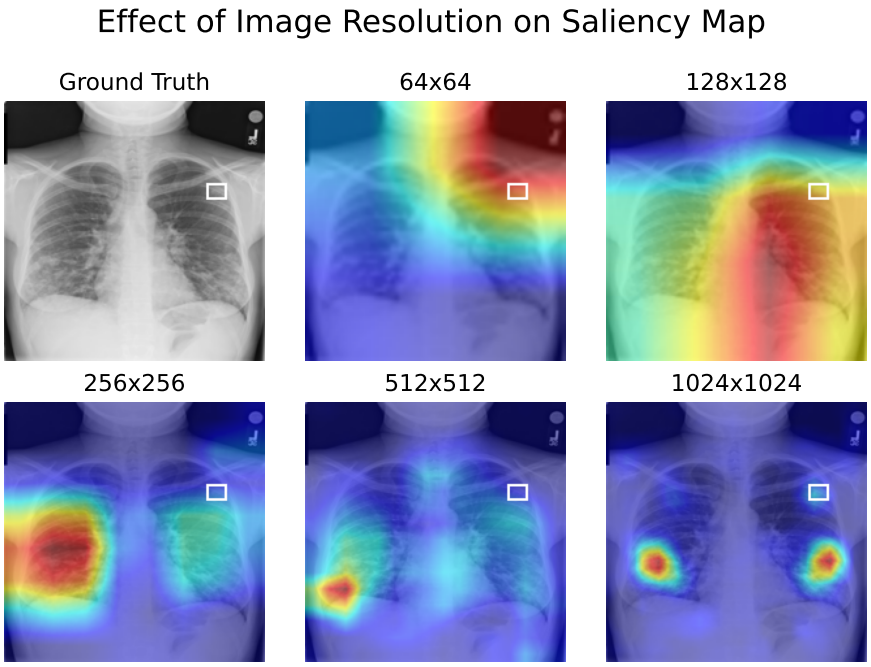

Figure 6.1: GradCAM saliency maps for different image resolutions. The annotated nodule bounding box is overlayed in white.

Since AlexNet, images processed by deep learning models are often resized to 224×224 pixels during training [30, 59, 58, 61], mostly for computational reasons. Training at a lower resolution requires less memory and consequently models train faster. Tan and Le studied the importance of image resolution on image classification accuracy on ImageNet [61]. Valuing image resolution as a parameter similar to network depth or

width. For chest radiographs, Sabottke and Spieler tested different image resolutions (32×32 up to 600×600 pixels) for chest X-ray classification [122]. In their experiments, maximum classification AUCs were obtained between 256×256 and 448×448 pixels resolution. Chest radiographs, on the other hand, have often a resolution of 2500×3500 pixels. While the images of the chest X-ray 14 data set [78] (see Section 3.1.4) were down-scaled to 1024×1024 pixels, it is the only large publicly available data set that also contains bounding boxes for eight pathologies. In this chapter, we investigate the effect of image resolution on chest X-ray classification performance.

6.2 Methods

Subsequent models were trained on the chest X-ray 14 data set containing 112,120 frontal view chest radiographs from 32,717 patients [78]. The test data set, contains a small sub set (983 images) with bounding box annotations for the eight findings: atelectasis, effusion, mass, cardiomegaly, infiltration, pneumonia, nodule, and pneumothorax. We used the development/test split provided by the authors and split the development split into Before model training, images were resized to 64×64, 128×128, 256×256, 512×512, and the full 1024×1024 pixel resolution.

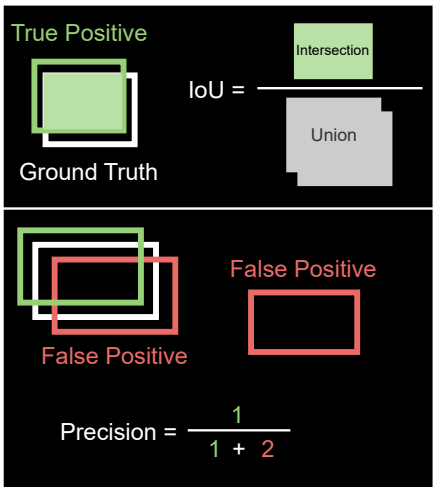

Figure 6.2: Evaluation of saliency map localization with precision at specific intersection over union. A prediction must overlap sufficiently, determined by the intersection over union (IoU) threshold. Only a single overlapping detection is considered a true positive. The precision is the number of true positives divided by the number of detections.

6.2.1 Chest X-ray Classification

For classification, a DenseNet-121 [59] pre-trained on the ImageNet [72] data set was used. To predict the 14 chest X-ray 14 classes, we replaced the last fully-connected layer with one with 14 output dimensions. We trained the model with binary cross entropy loss and used AdamW [123] for optimization.

6.2.2 Object Detection

Given the bounding box annotations for eight of the 14 findings, we investigated the effect of image resolution on predicted bounding boxes. We created binary segmentations from GradCAM [124] saliency maps generated by the penultimate layer by first normalizing them and then applying a threshold of 0.5. To create bounding boxes, we extracted the connected components and calculated the surrounding bounding boxes.

For each predicted, and annotated bounding box we calculated the intersection over union (IoU)

$$\text{IoU}(A, B) = \frac{A \cap B}{A \cup B},$$

shown in Figure 6.2, and the class-wise mean precision. The precision is defined as the number of true positives divided by the number of detections[1]. Since the chest X-ray 14 data set contains only one segmentation per sample, the precision is either 0 or 1 over the number of detections. A detection was considered positive, if the IoU was at least 0.1 (IoU$\geq$0.1). Out of multiple sufficiently overlapping detection only one was considered as a true positive.

6.3 Results

6.3.1 Chest X-ray Classification

Finding	64×64	128×128	256×256	512×512	1024×1024
Atelectasis	0.760	0.800	0.810	0.807	**0.821**
Cardiomegaly	0.858	0.900	0.906	**0.909**	0.908
Consolidation	0.752	0.787	**0.797**	0.794	**0.797**
Edema	0.866	0.869	0.885	0.878	**0.891**
Effusion	0.845	0.873	0.877	0.874	**0.879**
Emphysema	0.824	0.884	0.900	0.913	**0.937**
Fibrosis	0.748	0.803	0.816	0.821	**0.850**
Hernia	0.865	**0.926**	0.903	0.895	0.916
Infiltration	0.678	0.707	**0.714**	0.699	**0.714**
Mass	0.765	0.827	**0.830**	0.813	0.829
Nodule	0.669	0.719	0.761	0.780	**0.803**
Pleural Thickening	0.730	0.751	0.757	0.763	**0.796**
Pneumonia	0.688	0.743	0.760	0.760	**0.769**
Pneumothorax	0.799	0.839	0.858	0.859	**0.877**
Mean	0.775	0.816	0.827	0.826	**0.842**

Table 6.1: Chest X-ray classification AUCs for different image resolutions. Highest values are highlighted in bold.

Per-class AUC scores are provided in table 6.1. Unsurprisingly, the model trained on only 64×64 pixel images scored the lowest, with a mean AUC of 77.5 %. The highest resolution, 1024×1024 pixels, performed best with a mean AUC of 84.2 %, followed by 256×256 pixels with a mean AUC of 82.7 %.

6.3.2 Object Detection

When increasing image resolution, the generated GradCAM saliency maps became more detailed due to the larger output size before the global average pooling layer. An example is shown in Figure 6.1.

[1] For a more general explanation of object detection metrics, such as IoU and average precision, we refer to [125].

Finding	64×64	128×128	256×256	512×512	1024×1024
Atelectasis	0.001	0.117	0.120	0.361	**0.395**
Effusion	0.004	0.036	0.310	0.353	**0.356**
Mass	0.004	0.003	0.330	0.417	**0.508**
Cardiomegaly	0.114	0.804	**0.946**	0.792	0.242
Infiltration	0.008	**0.283**	0.084	0.201	0.200
Pneumonia	0.095	0.370	0.189	0.394	**0.402**
Nodule	0.000	0.000	0.000	0.150	**0.326**
Pneumothorax	0.003	0.061	**0.391**	0.180	0.198

Table 6.2: Mean precision at intersection over union ≥ 10 % of chest pathology bounding boxes and binary saliency maps.

Mean precision @ IoU ≥ 0.1 results are shown in Table 6.2. For most classes, the highest resolution, 1024×1024 pixels, had the highest precision, except for cardiomegaly, infiltration, and pneumothorax. Where 256×256 (cardiomegaly, pneumothorax) and 128×128 (infiltration) performed best.

The mean precision @ IoU ≥ 0.1 results show that, although the saliency maps were smaller they were more precise. Ignoring the number of false positives, i.e., how many saliency map-based segmentations were generated for a single ground truth and focusing on the maximum IoU per sample draws a similar picture, as shown in Table 6.3. While the highest resolution, 1024×1024, was the most precise except for cardiomegaly, infiltration, and pneumothorax, the highest average IoU was only highest for mass, nodule, and pneumothorax. For example, on average cardiomegaly bounding boxes and generated bounding boxes at 256×256 resolution had, on average, an IoU of 60.1 % compared, the highest resolution was worst with only 11.4 % IoU. We interpret these results, due to the nature of cardiomegaly bounding boxes that are the largest for this data set and that saliency maps for the highest resolution are small(er) than the annotatd bounding boxes. This hypothesis is supported by the results that at lower resolutions (128 - 512) cardiomegaly mean precision was significantly higher than for the highest resolution. The findings are inverted for the smallest pathology, lung nodules. Here, the bounding boxes from the highest resolution have a significantly higher mean precision and maximum IoU than all other resolutions. While these results suggest that a lower resolution improves classification performance of larger pathologies, the test AUCs in Table 6.1 show that the effect on classification performance is only noticable for the class hernia. For example, for pneumothorax the highest resolution achieved the highest mean AUC. We interpret these results, due to the kind of pathology. Signs of a pneumothorax, for example, could be only a long curve. Hence, the surrounding bounding box would cover a much larger area than the actual pathology.

6.4 Discussion

In this chapter, we studied the importance of chest X-ray resolution on image classification performance. The classification results (see Table 6.1) suggest that overall a higher

Finding	64×64	128×128	256×256	512×512	1024×1024
Atelectasis	0.030	0.049	0.071	**0.196**	0.177
Effusion	0.053	0.057	0.158	**0.209**	0.168
Mass	0.034	0.066	0.124	0.216	**0.340**
Cardiomegaly	0.126	0.280	**0.609**	0.264	0.114
Infiltration	0.100	**0.164**	0.092	0.133	0.055
Pneumonia	0.052	0.106	0.172	**0.206**	0.133
Nodule	0.003	0.008	0.023	0.073	**0.228**
Pneumothorax	0.050	0.080	0.126	0.109	**0.168**

Table 6.3: Mean max intersection over union

resolution improves image classification performance. The highest available resolution, 1024×1024, performed best for all findings except cardiomegaly, hernia, and mass. For these, it performed second to best. While Sabottke and Spieler achieved maximum AUCs between 256×256 and 448×448 pixels resolution, they tested only up to 600×600 pixels. We observed a slight decline in AUC from 256×256 to 512×512 pixel resolution for most (9/14) classes. These findings are in line with the conclusion of Tan and Le that their highest tested resolution 600×600 was not optimal. However, our results show that an even higher image resolution, 1024×1024 pixels, improved chest X-ray classification performance. Similar results were shown for image classification accuracy on ImageNet [61].

In summary, we investigated the effect of image resolution on chest X-ray classification and localization. Our results showed that a higher resolution of 1024×1024 performed best.

7 Saliency Maps

7.1 Introduction

Image classification models are trained to predict a class, they do not explain the reason for their prediction. This reasoning, however, is crucial for human acceptance. Without modifying the model architecture, one can use the intermediate representations of an image passing through the network and the gradient with respect to a specific class to visualize what parts of an image were important. These methods are commonly referred to as saliency maps. One popular method to create visual explanations for a model's decision-making process is gradient-weighted class activation mapping (GradCAM). GradCAM saliency maps are created by multiplying the class-dependent gradients with the activations of a specific layer. A more sophisticated approach can be applied when using vision transformers instead of convolutional neural networks. There, attention and gradients are combined to generate more precise saliency maps.

This chapter investigates the use of saliency maps on chest X-ray classification from a quantitative perspective, as well as, studying the subjective usefulness of saliency maps for radiologists to assess the quality of a model's prediction.

Most of the methods and results of this chapter have been published in Radiology Artificial Intelligence by Wollek and Graf et al. [94].

7.2 Attention-based Saliency Maps Improve Interpretability of Pneumothorax Classification

Artificial Intelligence (AI) has the potential to improve medical processes [126, 127] by increasing human performance and speed, but its black-box nature leads to big hurdles for decision- making support [128, 129]. Radiologists can confirm or reject AI-based identification of an important finding, such as a potentially life-threatening pneumothorax, most optimally when the location is provided; evaluation becomes more difficult and time-consuming when the area in question remains to be located [130]. Even when deep learning models perform similarly to radiologists [89, 14], erroneous decisions made by the models differ in nature from those made by human experts [14]. Therefore, radiologists working with such a model need appropriate methods to understand its strengths and limitations. A saliency map addresses this problem by highlighting locations that are relevant to the model's decision-making process (7.1). Predictions based on insufficient evidence, such as a chest tube as sole evidence for pneumothorax, allow the radiologist to confidently reject the model's prediction. Current state-of-the-art image classifiers are based on the vision transformer (ViT) [67, 131], Compared with convolutional neural

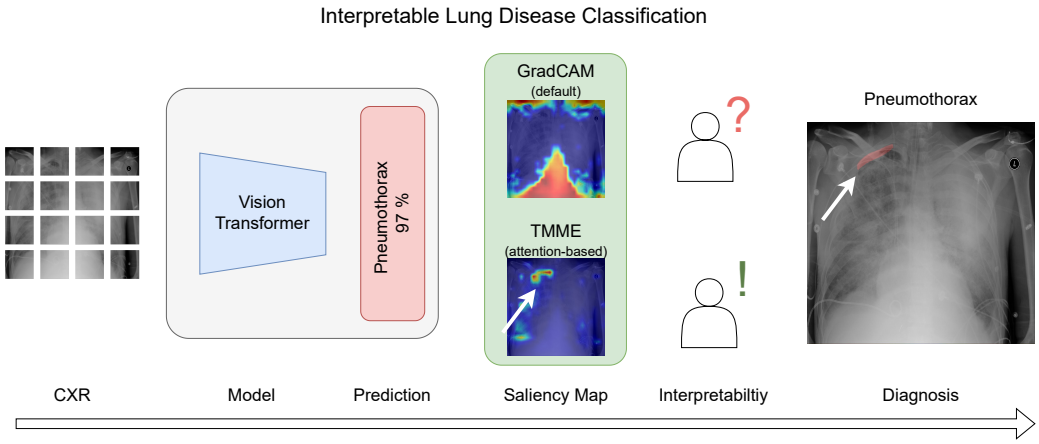

Figure 7.1: Diagram of interpretable lung disease classification pipeline used in this study. After classifying a chest radiograph (CXR) using a vision transformer network, an attention-based saliency map, transformer multi-modal explainability (TMME), is generated to aid the radiologist in interpreting the modelś prediction. The proposed saliency map based on TMME was compared with the conventional gradient-weighted class activation map (GradCAM).

networks, ViTs leverage the attention mechanism that is structurally easier to interpret, as they capture the importance of spatial token relationships (e.g., image patches) to model predictions. We created attention-based saliency maps using transformer multi-modal explainability (TMME) [132]. TMME is an extension of the roll-out mechanism, the matrix multiplication of all attention maps. The attention heads are merged by taking the minimum or mean. Skip-connections of attention layers are accounted for by adding an identity matrix to the merged attention matrix. Additionally, all heads are multiplied with their derivatives conditioned on a chosen class. Negative values are set to zero before applying roll-out. As baseline, we applied the frequently used gradient-weighted class activation mapping (GradCAM) technique [124].

To date, there has been limited research on the use of saliency maps for medical purposes [133, 134, 135, 136, 137, 138, 139], which all focus on comparing saliency maps and segmentation rather than explaining the basis for a model's decision. There is a need to investigate the reliability of saliency maps on chest radiographs (CXRs) and their usefulness to radiologists. To the best of our knowledge, medical studies have not yet been conducted on attention-based explainers. The aim of this exploratory study was to evaluate the feasibility of ViTs for CXR classification and interpretability of attention-based saliency maps for clinical decision support using the example of pneumothorax classification.

7.2.1 Materials and Methods

7.2.1.1 Data Sets

This retrospective study used five public CXR data sets: CheXpert [80], Chest X-Ray 14 [78], MIMIC CXR [81], VinBigData [140], and Society for Imaging Informatics in

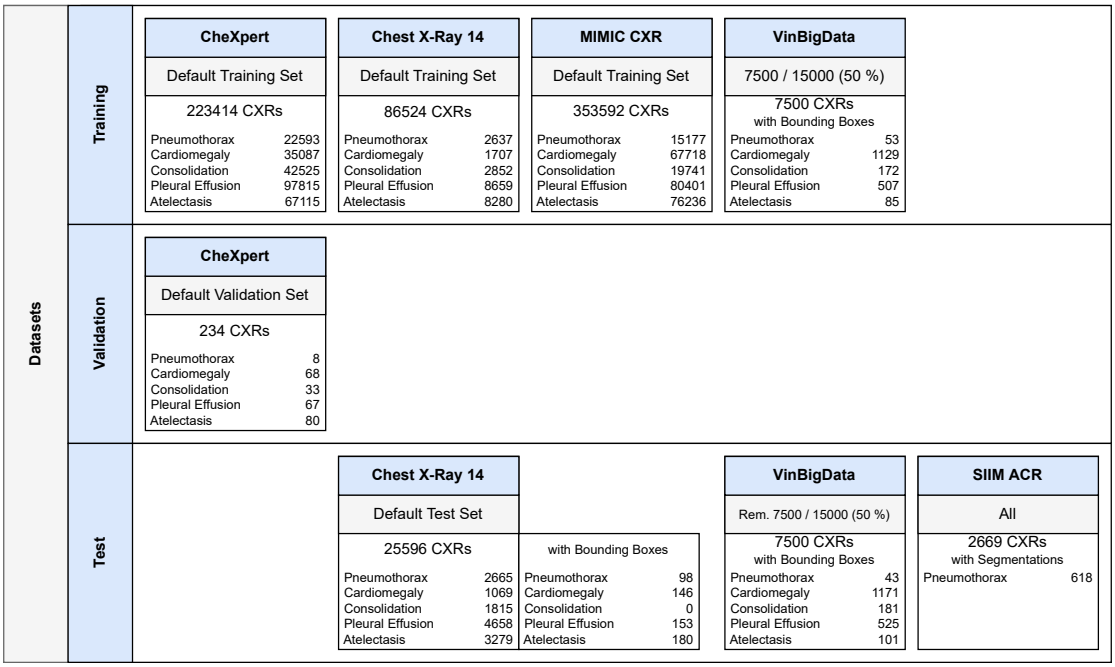

Figure 7.2: Data sets with data splits and classes used in this study. CXR = chest radiograph, SIIM-ACR = Society for Imaging Informatics in Medicine-American College of Radiology

Medicine- American College of Radiology (SIIM-ACR) [82] (7.2). CheXpert consists of 224,316 CXR of 65,240 patients, Chest X-Ray 14 of 112,120 CXR of 30,805 patients, MIMIC CXR of 377,110 CXR of 65,379 patients, VinBigData of 15,000 labeled CXR, and SIIM-ACR of 2669 pneumothorax CXR. Institutional review board review and patient informed consent were not required for this study. We trained on CheXpert, Chest X-Ray 14, MIMIC CXR and VinBigData, validated on the holdout set of CheXpert, and tested on the data sets containing image annotations: the holdout set of Chest X-Ray 14, 7500/15000 (50%) of the VinBigData set, and all SIIM-ACR data. All sets were split without patient overlap and are publicly accessible. Further data set details are described in the respective sections: CheXpert, Section 3.1.5, Chest X-Ray 14, Section 3.1.4, MIMIC CXR, Section 3.1.6, VinBigData, Section 3.1.9, and SIIM-ACR, Section 3.1.7.

7.2.1.2 Model Training and Development

We fine-tuned the ViT variant 'deit_base_distilled_patch16_224' [131] on the overlapping training data set classes: pneumothorax, cardiomegaly, consolidation, pleural effusion and atelectasis (7.2). We replaced the final softmax layer with a sigmoid. We used stochastic gradient descent (SGD) with a learning rate of 0.001 and a batch size of 64 and trained for 500 epochs with early stopping. We randomly oversampled the images. During training, we applied random affine translations (-15 ° – 15 °, translate 0.05, scale 0.9 – 1.05) and random horizontal flip. All images were normalized according to the ImageNet mean and standard deviation and resized to 224 × 224 pixels. We

compared our network with a pre- trained DenseNet-121 [59], commonly used for CXR classification [89, 13]. Our model was fine-tuned using the same data and augmentations, SGD with a learning rate of 0.001 and batch size of 32, and trained for 30 epochs with early stopping.

Saliency Map Evaluation Metrics

Positive and Negative Perturbation Test We continuously removed patches (16x16 pixels) of the input image according to their saliency map importance and then generated a new saliency map. We removed the most important patches in the positive perturbation test and least important patches in the negative perturbation test. Model confidence was measured while removing image patches AUC was calculated. We followed the procedure by Chefer et al. [141], but instead of blackening the removed patches, we replaced them with patches of an image with zero predictive confidence of the network, guaranteeing a convergence to zero instead of random predictions after most pixels were replaced.

Sensitivity-n The sensitivity-n test [142] applies random masks onto the input image. Masks lowering the confidence should correlate with a good saliency map. We used 200 random masks consisting of n random tokens (16x16 pixels). The Hadamard products of saliency map $\times$ mask and the change in confidence were compared with the Pearson product- moment correlation coefficients [143]. Scores were averaged, and the test was repeated with different numbers of tokens n, logarithmically distributed between 1 and half the total number tokens. We computed the AUC between token number and correlation score. We used the test images that were predicted as a pneumothorax by the ViT.

Effective Heat Ratio We used the effective heat ratio (EHR) [142] to show that the saliency maps highlight regions that align with prior medical knowledge. The test first produces a binary mask of the saliency map given a threshold. The EHR is the fraction between the threshold area inside the ground truth and the complete threshold area. The thresholds were computed in equidistant steps. We computed the AUC over all EHRs and thresholds.

Intra-architecture repeatability and interarchitecture reproducibility Following Arun et al. [133], we compared the similarity between saliency maps of different models. We compared the mean structural similarity index measure (SSIM) of 1000 saliency maps in the SIIM-ACR dataset on pneumothorax CXRs.

7.2.1.3 User Study

To evaluate clinical usefulness, we surveyed one board-certified radiologist with more than ten years' experience, one fourth year and one first year radiology resident from

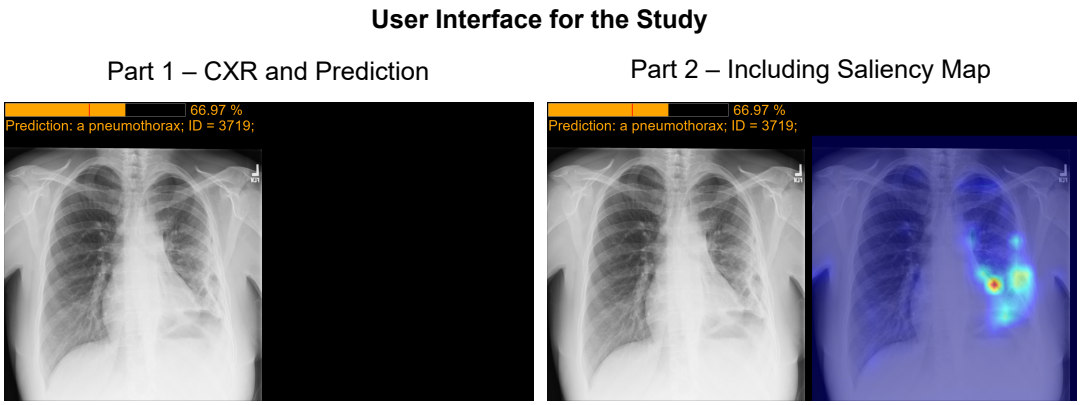

Figure 7.3: User interface for the radiologists in the study. First, they were shown chest radiographs (CXRs) with and without a present pneumothorax and the vision transformer (ViT) prediction score. In the second step, a saliency map was additionally shown. For both parts, radiologists had to detect if a pneumothorax was present and then determine if the saliency map was (subjectively) useful for aiding detection.

Klinikum der Ludwig-Maximilians-Universität München (hereafter referred to as "radiologists"). We compared GradCAM and TMME on images with and without pneumothorax. We had to limit our investigation to pneumothorax detection due to available segmentations and study participation time. We sampled 160 images from the SIIM-ACR data set, 110 with and 50 without pneumothorax. We included 70 TMME, 70 GradCAM, 10 artificial saliency maps based on segmentations and 10 random saliency maps generated by applying Gaussian-blur and multiplying repetitively with Perlin-noise (Table 7.1). We calibrated model predictions using histogram binning [143]. In two consecutive parts, the radiologists assessed the presence of pneumothorax when first given the CXR and model prediction and then when additionally given a saliency map (7.3). Furthermore, they rated usefulness of the saliency maps on a scale from 1-5 (strongly disagree – strongly agree).

Model Prediction Saliency Map	True Positives	False Negatives	True Negatives	False Positives
GradCAM	30	15	10	15
TMME	30	15	10	15
Artificial	10	10	0	0

Table 7.1: Distribution of test images for the user study.

7.2.1.4 Statistical Analysis

Model performance was assessed using receiver operating characteristics (ROC), sensitivity, and specificity at maximum F1-score with 95 % confidence intervals (calculated using the non-parametric bootstrap method with 10,000-fold resampling at the image

level). We compared model classification performance using the area under the ROC curve (AUC). AUC comparison was performed using fast implementation [144] of the non-parametric approach of DeLong et al. [145]. Since our analysis is exploratory and involves multiple comparisons P values are provided but significance claims are not made. Radiologist pneumothorax classification performance with and without saliency maps was measured using sensitivity and specificity. Statistical analyses were performed using NumPy[1] version 1.21.5 and SciPy[2] version 1.7.3 in this study.

7.2.2 Results

7.2.2.1 CXR Classification Performance

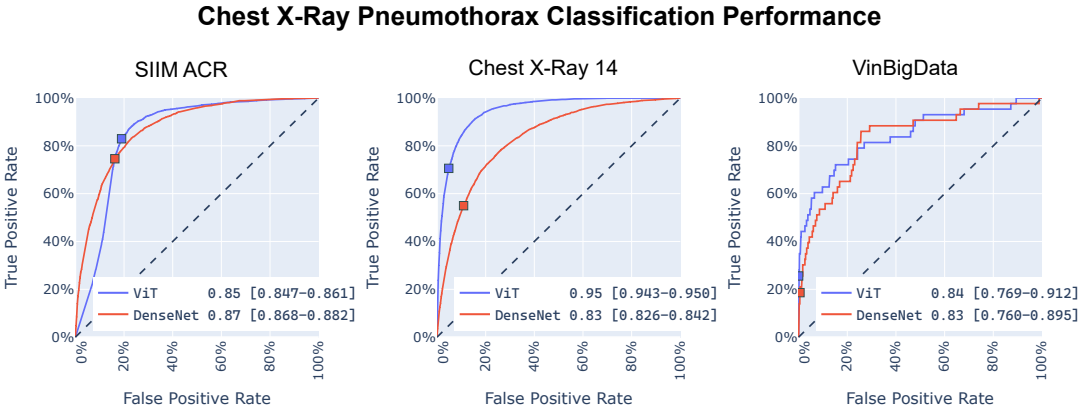

Figure 7.4: Receiver operating characteristic (ROC) curves and areas under the ROC curves [95 % CIs] for pneumothorax classification by vision transformer (ViT) and DenseNet models. The ROC curves are computed on hold-out sets. The dots show the maximized F1-scores. SIIM-ACR = Society for Imaging Informatics in Medicine-American College of Radiology

As the focus of this study lies on pneumothorax, we report the performance results of the other classes, atelectasis, cardiomegaly, consolidation, and pleural effusion, only briefly here. The ROC curves and corresponding AUCs with 95 % CIs are shown at the end of this section (Figure 7.7 - Figure 7.11). For these classes, the ViT outperformed the DenseNet on both the Chest X-Ray 14 and VinBigData data sets. Regarding pneumothorax classification, the ViT achieved a higher AUC than DenseNet on Chest X-Ray 14 (AUCs, 0.95 [95 % CI: 0.94, 0.95] vs. 0.83 [95 % CI: 0.83, 0.84]; $p<0.001$). We found no evidence of a difference between ViT and DenseNet performance on VinBigData (AUCs, 0.84 [95 % CI: 0.77, 0.91] vs. 0.83 [95 % CI: 0.76, 0.90]; $p = 0.67$). The DenseNet pneumothorax classification resulted in a higher AUC than ViT on the SIIM-ACR dataset (AUCs, 0.87 [95 % CI: 0.87, 0.88] vs. 0.85 [95 % CI: 0.85, 0.86]; $p<0.001$; Figure 7.4). F1-score, sensitivity, and specificity are reported in Table 7.2.

[1] `https://numpy.org`
[2] `https://scipy.org`

Data Set	Model	F1 Score	Sensitivity	Specificity
SIIM-ACR	ViT	**66 %**[65 %-68 %]	**83 %**[82 %-85 %]	81 %[80 %-82 %]
	CNN	64 % [63 %-66 %]	75 % [71 %-78 %]	**84 %** [82 %-86 %]
CXR14	ViT	**67 %**[65 %-68 %]	**71 %**[68 %-73 %]	**95 %**[95 %-96 %]
	CNN	44 % [43 %-46 %]	55 % [50 %-63 %]	89 % [86 %-91 %]
VinBigData	ViT	**40 %**[26 %-56 %]	**26 %**[13 %-40 %]	100 % [100 %-100 %]
	CNN	19 % [11 %-33 %]	19 % [5 %-30 %]	100 % [98 %-100 %]

Table 7.2: Comparison of vision transformer (ViT) and DenseNet (CNN) performance for pneumothorax classification on chest radiographs. Maximized F1-scores and the resulting sensitivity and specificity. Data in brackets are 95 % CIs. Higher values between the two models are marked in bold. SIIM-ACR = Society for Imaging Informatics in Medicine-American College of Radiology

7.2.2.2 Saliency Map Evaluation

The following tests measure how well each saliency method highlights important regions for the ViT's prediction. Across all saliency map tests and data sets, TMME performed better than GradCAM (Figure 7.5). On the positive perturbation test, where a lower value is better, as the most important tokens are removed first, TMME and GradCAM measured 0.04 (95 % CI: 0.036, 0.044) vs. 0.12 (95 % CI: 0.110, 0.135) on SIIM-ACR, 0.03 (95 % CI: 0.030, 0.037) vs. 0.12 (95 % CI: 0.111, 0.137) on Chest X-ray 14, and 0.02 (95 % CI: 0.013, 0.037) vs. 0.09 (95 % CI: 0.048, 0.135) on VinBigData, respectively. The negative perturbation test results, where a higher value is better, as the most important tokens are removed last, were 0.67 (95 % CI: 0.65, 0.69) for TMME vs. 0.37 (95 % CI: 0.34, 0.39) for GradCAM on SIIM-ACR, 0.55 (95 % CI: 0.52, 0.57) vs. 0.25 (95 % CI: 0.23, 0.27) on Chest X-ray 14, and 0.73 (95 % CI: 0.60-0.85) vs. 0.55 (95 % CI: 0.40-0.67) on VinBigData. Sensitivity-n (higher is better) values for TMME compared with GradCAM were 14 (95 % CI: 12.59, 15.32) vs. 6 (95 % CI: 4.44, 6.57) on SIIM-ACR, 11 (95 % CI: 9.85, 13.14) vs. 4 (95 % CI: 2.83, 5.17) on Chest X-ray 14, and 9 (95 % CI: 1.21, 17.98) vs. 7 (95 % CI: 0.55, 13.00) on VinBigData datasets, respectively. TMME achieved an EHR (higher is better) of 0.16 (95 % CI: 0.142, 0.171) vs. 0.11 (95 % CI: 0.099, 0.122) for GradCAM on SIIM-ACR, 0.26 (95 % CI: 0.199, 0.318) vs. 0.14 (95 % CI: 0.100, 0.193) on Chest X-ray 14, and 0.33 (95 % CI: 0.237, 0.434) vs. 0.22 (95 % CI: 0.126, 0.275) on VinBigData datasets. For intra-architecture repeatability, TMME had an average SSIM score of 0.57 (95 % CI: 0.562, 0.578) vs. a GradCAM SSIM score of 0.12 (95 % CI: 0.105, 0.126). Comparing different models, TMME had an inter-architecture reproducibility SSIM score of 0.47 (95 % CI: 0.465, 0.481) vs. 0.08 (95 % CI: 0.074, 0.091) for GradCAM (Figure 7.12).

During visual evaluation of the produced saliency maps, we detected that on some images, both GradCAM and TMME saliency maps highlighted confounders, such as a chest tube, instead of pneumothorax (Figure 7.6).

7.2.2.3 User Study

ViT Prediction	Method	Not Useful	Neither	Useful
False Negatives	Artificial	26	0	4
	GradCAM	23	7	15
	TMME	**18**	5	**22**
False Positives	GradCAM	29	4	12
	TMME	**22**	4	**19**
True Negatives	GradCAM	20	1	9
	TMME	**12**	7	11
True Positives	Artificial	**2**	9	19
	GradCAM	37	7	46
	TMME	34	9	**47**

Table 7.3: Cumulative votes assessing saliency map usefulness for pneumothorax detection by the three radiologists in the user study. Most useful and least not useful votes are marked in bold.

Given the frontal CXR and the ViT's prediction alone, the radiologists achieved a sensitivity of 64% (211/330; 95 % CI: 59 %, 69 %) and specificity of 84% (126/150; 95 % CI: 78 %, 90 %). Sensitivity improved to 65% (216/330; 95 % CI: 60 %, 70 %) and specificity remained at 84% (126/150; 95 % CI: 78 %, 90 %) after additionally showing the saliency map. Excluding the model's incorrect predictions resulted in a sensitivity of 77% (162/210; 95 % CI: 72 %, 83 %) before viewing the saliency map and 79% (165/210; 95 % CI: 73 %, 84 %) after; specificity was 93% (56/60; 95 % CI: 86 %, 98 %) before and 93% (56/60; 95 % CI: 87 %, 99 %) after viewing the saliency map. Showing GradCAM saliency maps did not improve sensitivity or specificity.

TMME saliency maps improved sensitivity from 61% (83/135; 95 % CI: 53 %, 70 %) to 64% (86/135; 95 % CI: 55 %, 71 %) (Figure 7.13). The artificial saliency maps, based on ground truth segmentations, were rated useful in 63 % (19/30) true positive cases and not useful in most false negative cases (26/30, 87 %) (Table 7.3). Both TMME and GradCAM were rated useful in 47/90, 52% and 46/90, 51% true positive predictions, respectively. Overall, TMME was rated useful (99/210, 47%) more often than GradCAM (82/210, 39%), especially for false model predictions (TMME: 41/90, 45% vs. GradCAM: 27/90, 30%).

7.2.3 Discussion

In this study, we trained a ViT on several CXR data sets to generate attention-based saliency maps to evaluate the performance of attention-based saliency maps compared with GradCAM and their usefulness in assisting radiologists in detecting pneumothoraces. We found that ViTs achieve similar results on CXR classification compared with convolutional neural networks (CNN; AUCs ranged from 0.84 to 0.95 for ViT, and from 0.83 to 0.87 for CNN), despite the limited amount of training data, as ViTs require more

training data than CNNs, encouraging further research. Given more training data we expect the ViTs to significantly outperform CNN-based architectures, as shown in other computer vision tasks (Figure 7.14).

Furthermore, we found that attention-based TMME saliency maps outperformed GradCAM across all metrics. This result was further confirmed by a difference in usefulness for pneumothorax detection by radiologists seeing TMME saliency maps compared with GradCAM or no saliency map, suggesting that attention-based saliency maps are more useful in clinical decision support than GradCAM. We attribute the drastically different results of GradCAM and TMME in some cases, where GradCAM visualization did not allow interpretation, to the inner workings of each method: GradCAM uses a single layer, while TMME combines the attention of multiple layers of the network, making it more robust. Saliency maps run the risk of human tendency to over-interpret a given visual result [146] and must be evaluated systematically. Model or data biases, such as chest tubes implying a pneumothorax, can be discovered using saliency maps. This phenomenon was observed in the literature before [107] and highlights the importance of increasing the interpretability of such models. These insights can then be used to improve both data quality and architectural choices. While the results suggest that the radiologists were biased by incorrect model predictions, the saliency map improved the readers' sensitivity and could be used to prevent confirmation bias [147]. In a post-study discussion, participants reported that the saliency maps helped them to better assess the strengths and weaknesses of the model, such as the model's focus on the presence of a chest tube in predicting pneumothorax probability. Despite the small number of participants in our study, improved reader sensitivity with attention-based saliency maps encourages further research. There were several limitations in our study. First, due to the unavailability of segmentation masks for lung diseases besides pneumothorax, the saliency maps could not be analyzed as precisely using the effective heat ratio. In these cases, we used the available but less precise bounding boxes. Second, we believe that model performance, particularly in pneumothorax classification, and the resulting saliency maps were limited by the small amount of available data, even when pooling data from multiple institutions (Figure 7.14). ViTs have been shown to substantially outperform CNN-based architectures given enough data [67]. Third, while our work indicates that the saliency maps have a positive effect on radiologist pneumothorax classification sensitivity, it is limited by the number of samples and participants. A more thorough assessment of the effect of saliency maps on the diagnostic performance of radiologists requires more study participants and a structured reading to mitigate confirmation biases. This would also allow for assessment saliency map usefulness for different levels of radiologist expertise. Fourth, following the protocol by Dosovitskiy et al. [67], radiographs were resized to 224×224 pixels, making subtle pneumothoraces potentially undetectable. In conclusion, we investigated the possible use of saliency maps based on ViTs for CXR classification. We showed that ViTs achieved similar results compared with conventional CNNs. The attention-based saliency map, TMME, performed better than the baseline GradCAM, supporting the use of ViTs for CXR classification. Radiologists rated attention-based saliency maps useful in more cases than GradCAM.

We showed saliency maps improve model understanding for both AI developers and radiologists, aiding in detection of model biases.

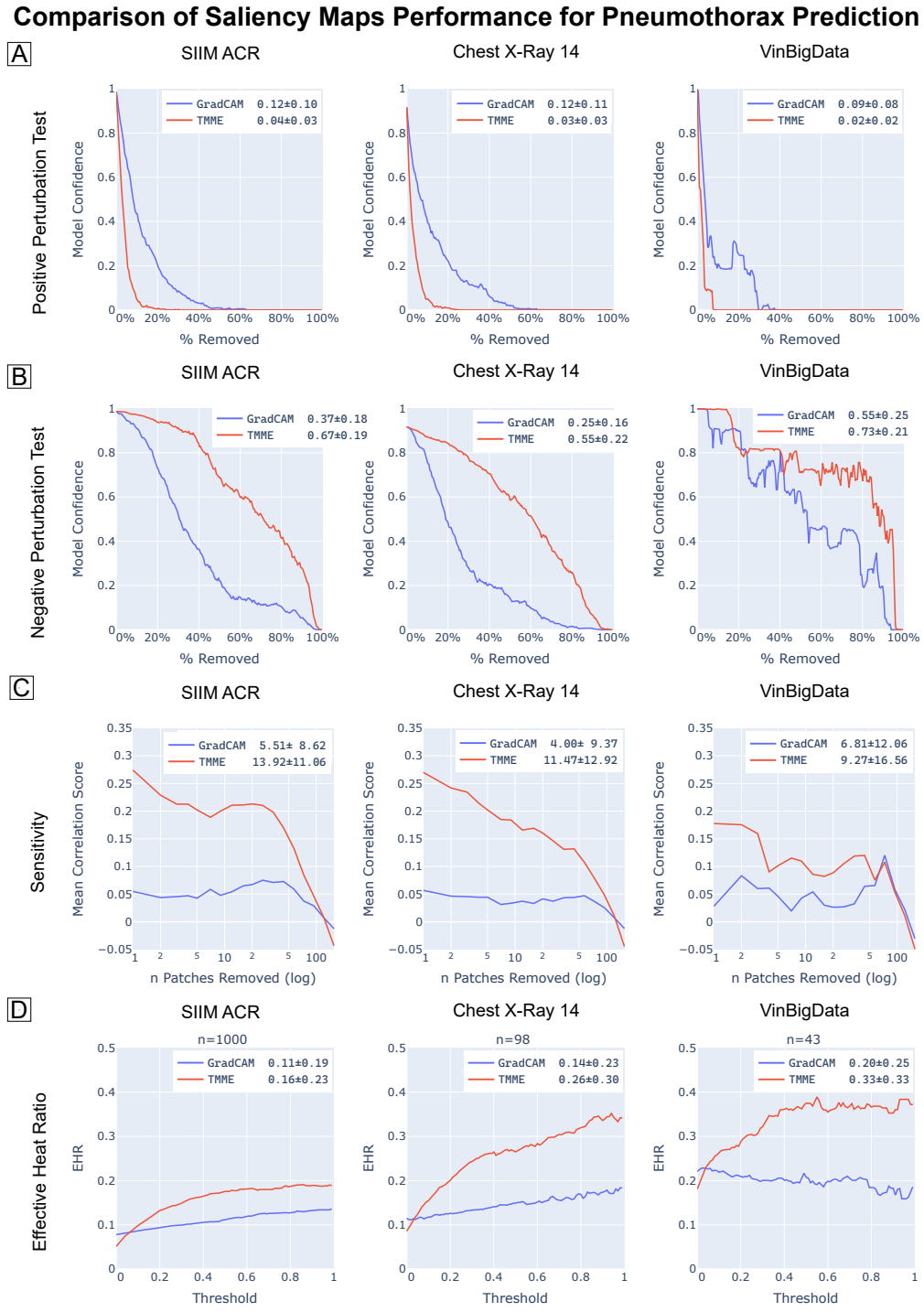

Figure 7.5: Saliency map performance comparison for pneumothorax prediction. Across all tests and data sets transformer multi-modal explainability (TMME) performed significantly better than gradient-weighted class activation mapping (GradCAM); Saliency metric values ± SD values are the integrals over the respective curves and provided in each graph. **A**: Positive perturbation test; a low value is better. **B**: Negative perturbation test; a high value is better. **C**: Sensitivity-n; a high value is better. **D**: Effective heat ratio (EHR); a high value is better. SIIM-ACR = Society for Imaging Informatics in Medicine-American College of Radiology

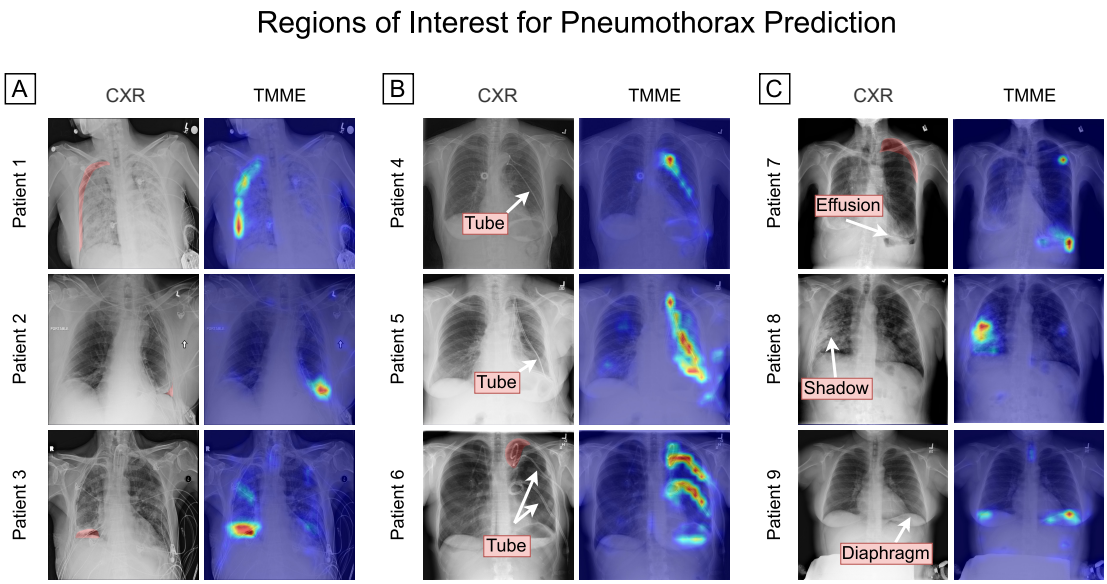

Figure 7.6: Images with vision transformer pneumothorax prediction from SIIM-ACR with transformer multi-modal explainability (TMME) visualization. Ground truth pneumothorax segmentation is highlighted in red. False positives, without pneumothorax, have no inpainting. **A**: Examples where TMME highlights the pneumothorax. **B**: Examples with chest tube highlighting. **C**: Pneumothorax prediction based on other pathologies (plural effusion, lung shadow) or the thoracic diaphragm. CXR = chest radiograph, SIIM-ACR = Society for Imaging Informatics in Medicine-American College of Radiology

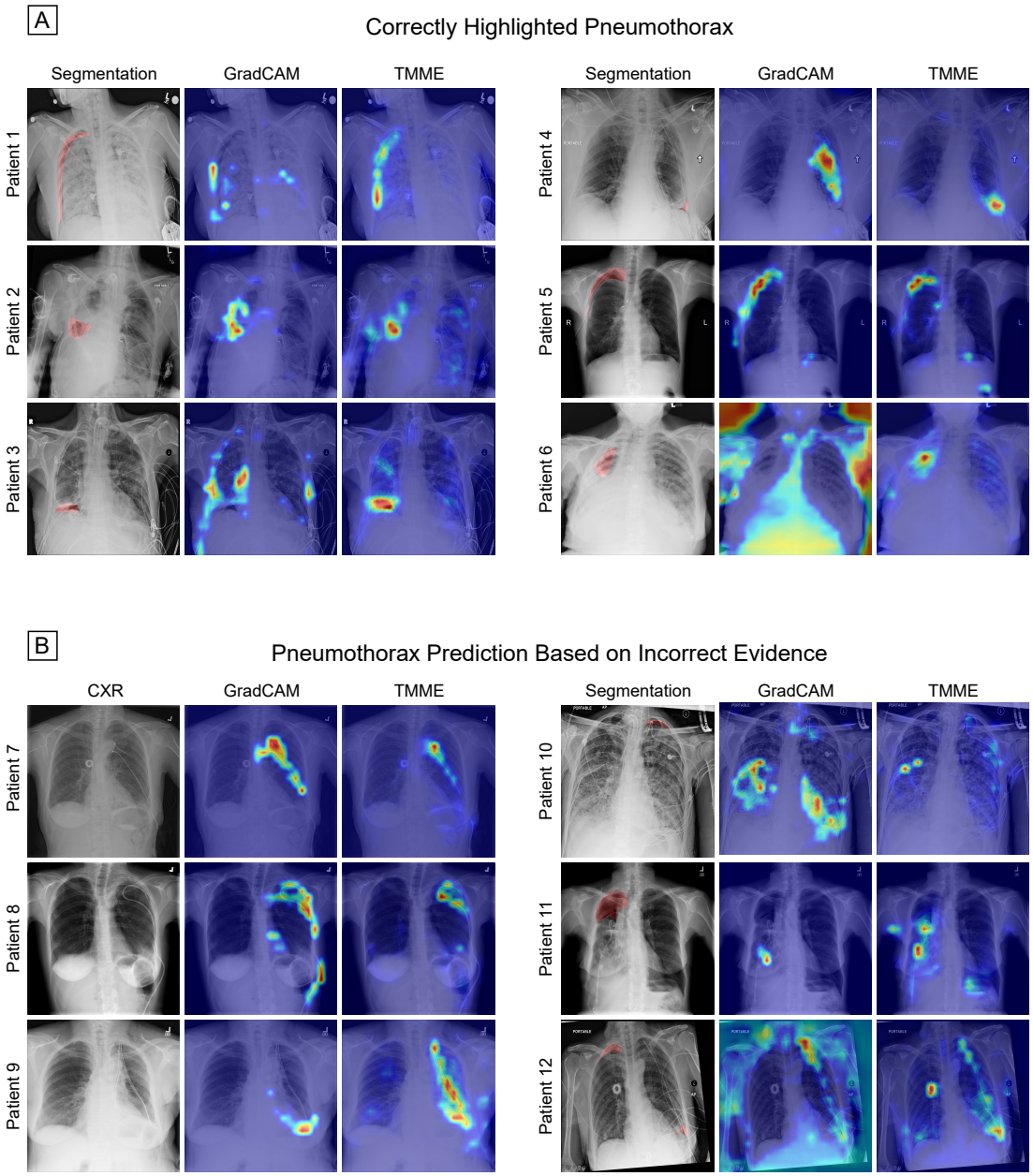

Figure 7.7: Saliency maps of pneumothorax predictions based on clinically correct and incorrect evidence. **A**: True positive examples where transformer multi-modal explainability (TMME) highlighted the pneumothorax. **B**: False positives (left) and true positives (right) where the TMME highlights incorrect evidence for pneumothorax. We sampled 50 random true positive and 25 false positive predictions to generate these images.

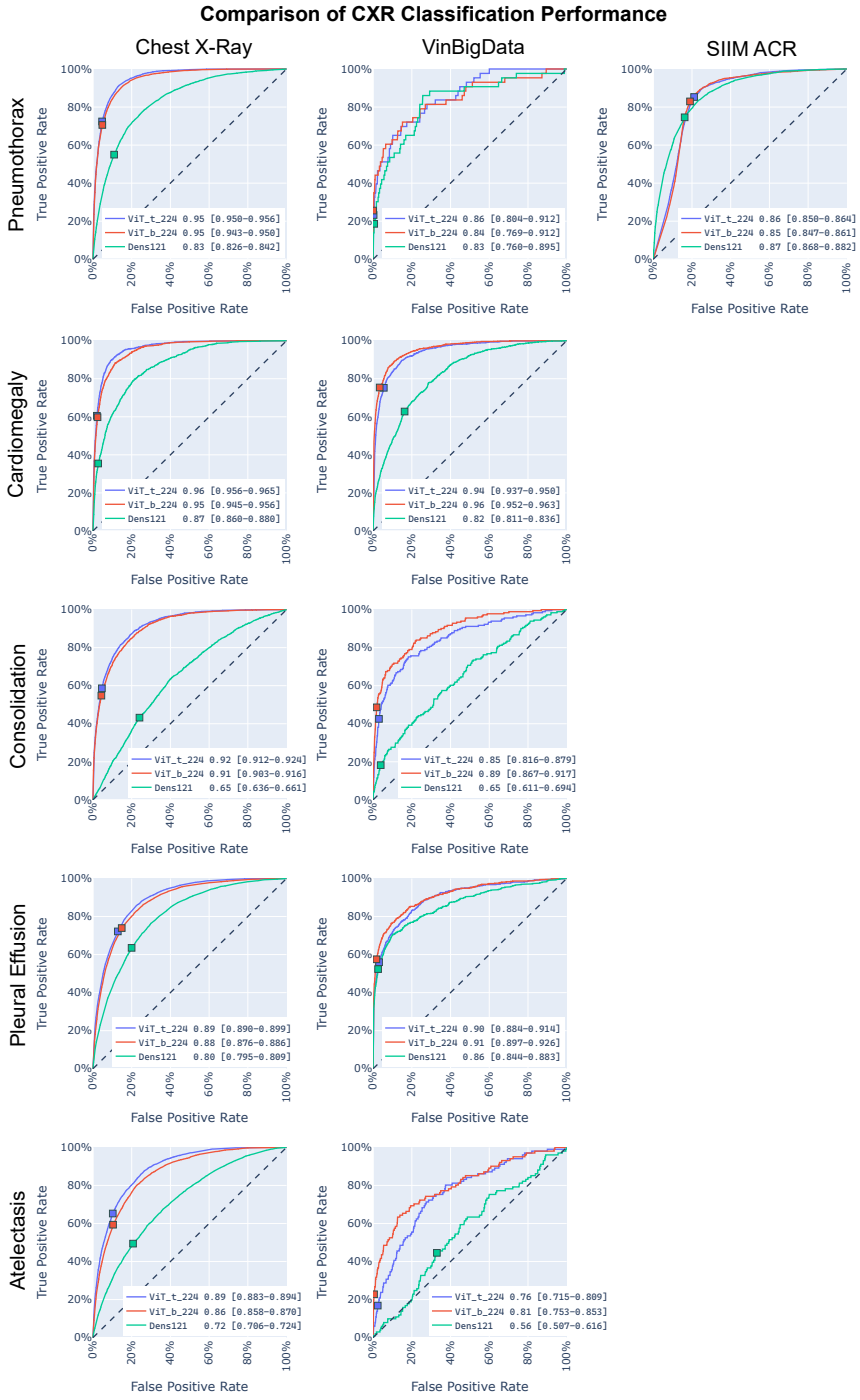

Figure 7.8: Receiver operating characteristic curves (ROC) for all five classes: pneumothorax, cardiomegaly, consolidation, pleural effusion, and atelectasis. The SIIM-ACR data set contains only pneumothorax labels. ViT_b_244 (red) is the ViT reported in the main text. During prototyping we additionally tested the smaller version, ViT_t_244 (blue). The area under the ROC curve and 95 % confidence intervals are reported in the legend. SIIM-ACR = Society for Imaging Informatics in Medicine-American College of Radiology.

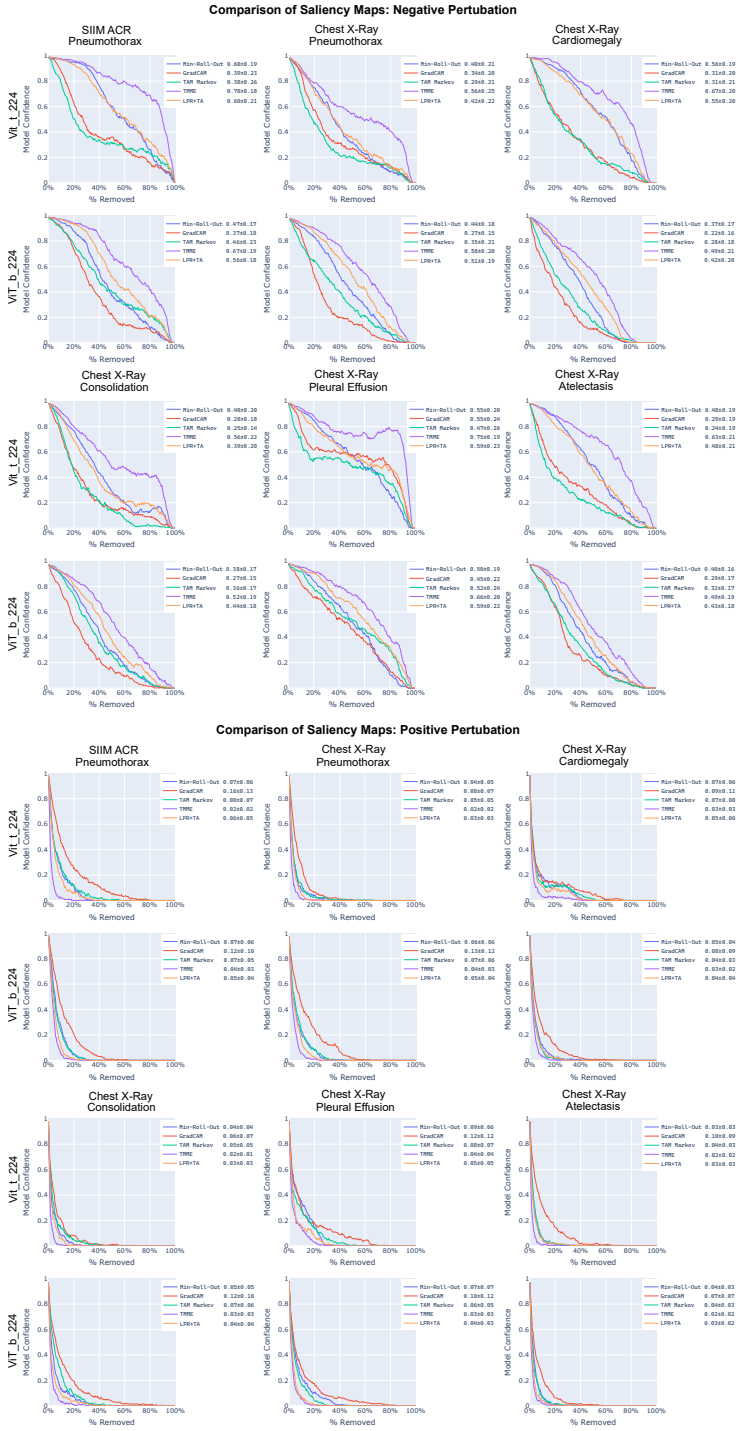

Figure 7.9: Positive and negative perturbation test on all 5 classes: pneumothorax, cardiomegaly, consolidation, pleural effusion, and atelectasis. ViT_b_244 is the same ViT as reported in the text, ViT_t_244 is the smaller version used during prototyping. During prototyping we investigated other, less performing, attention-based saliency map methods: min-roll-out, transition attention maps (TAM) Markov, and layer-wise relevance propagation transformer-attribution (LRP+TA). The area under the receiver operating characteristic curves and SD are reported in the legend. The results are averaged over 250 true positive examples.

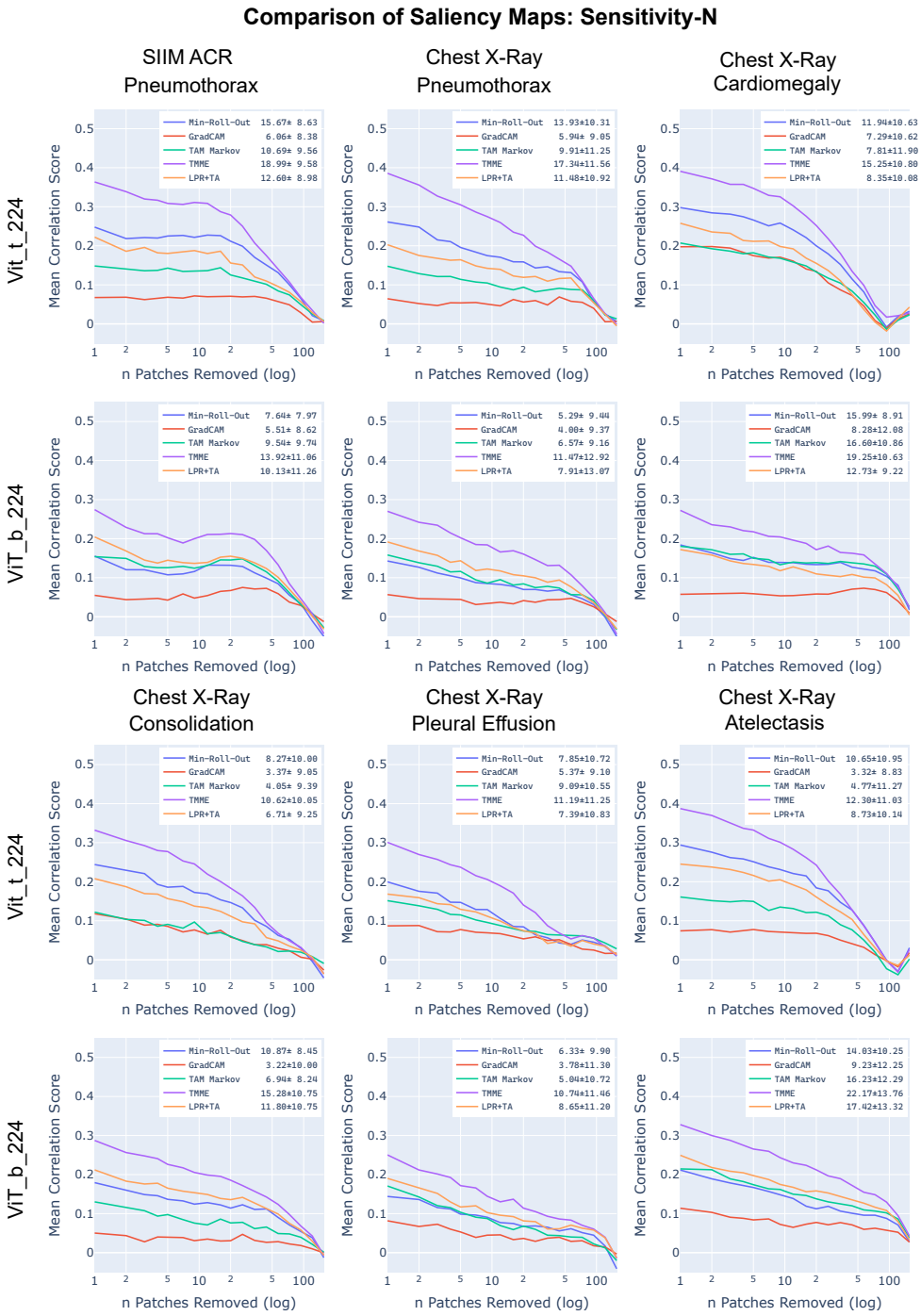

Figure 7.10: Sensitivity-n test on all 5 classes: pneumothorax, cardiomegaly, consolidation, pleural effusion, and atelectasis. ViT_b_244 is same ViT as reported in the text, ViT_t_244 is the smaller version used during prototyping. During prototyping we investigated other, less performing, attention-based saliency map methods: min-roll-out, transition attention maps (TAM) Markov, and layer-wise relevance propagation transformer-attribution (LRP+TA). The area under the receiver operating characteristic curves and SD are reported in the legend.

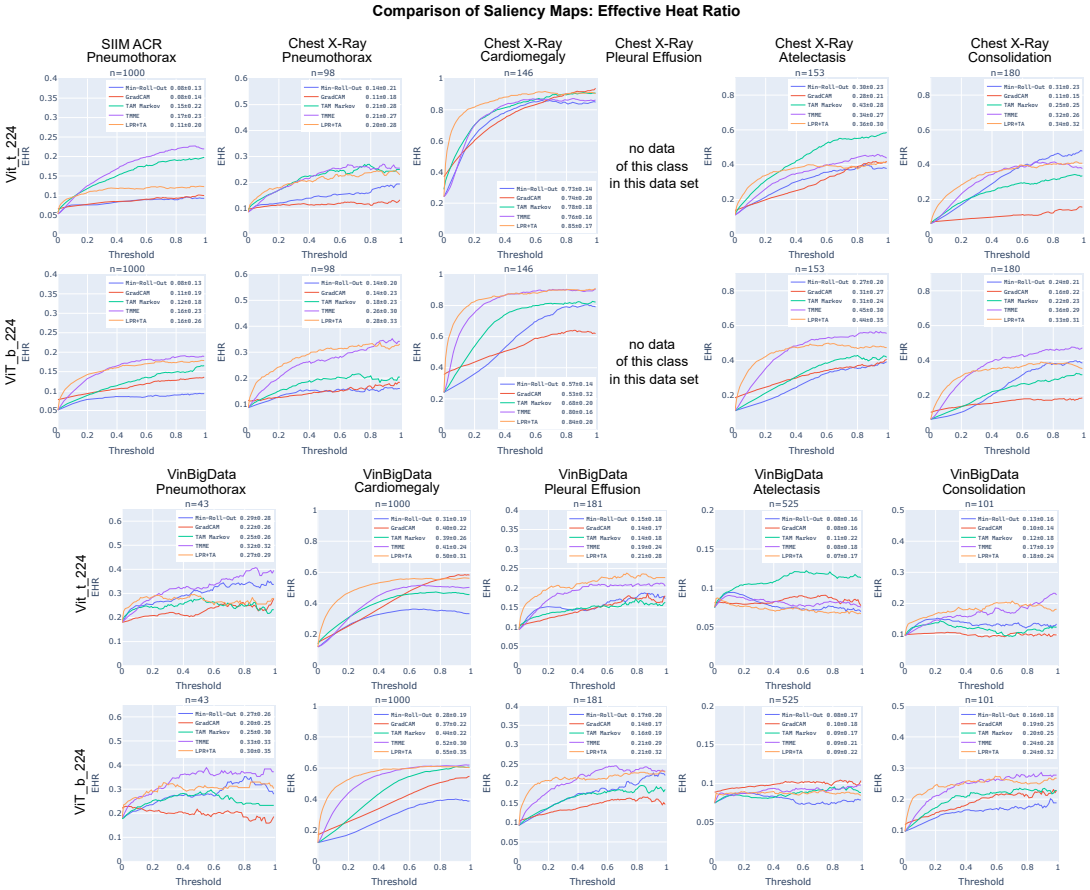

Figure 7.11: EHR test on all 5 classes: pneumothorax, cardiomegaly, consolidation, pleural effusion, and atelectasis. ViT_b_244 is same ViT as reported in the text. While ViT_t_244 is the smaller version used during prototyping. During prototyping we investigated other, less performing, attention-based saliency map methods: min-roll-out, transition attention maps (TAM) Markov, and layer-wise relevance propagation transformer-attribution (LRP+TA). The area under the receiver operating characteristic curves and SD are reported in the legend. The results are averaged over n segmented examples, where n is reported in the diagram.

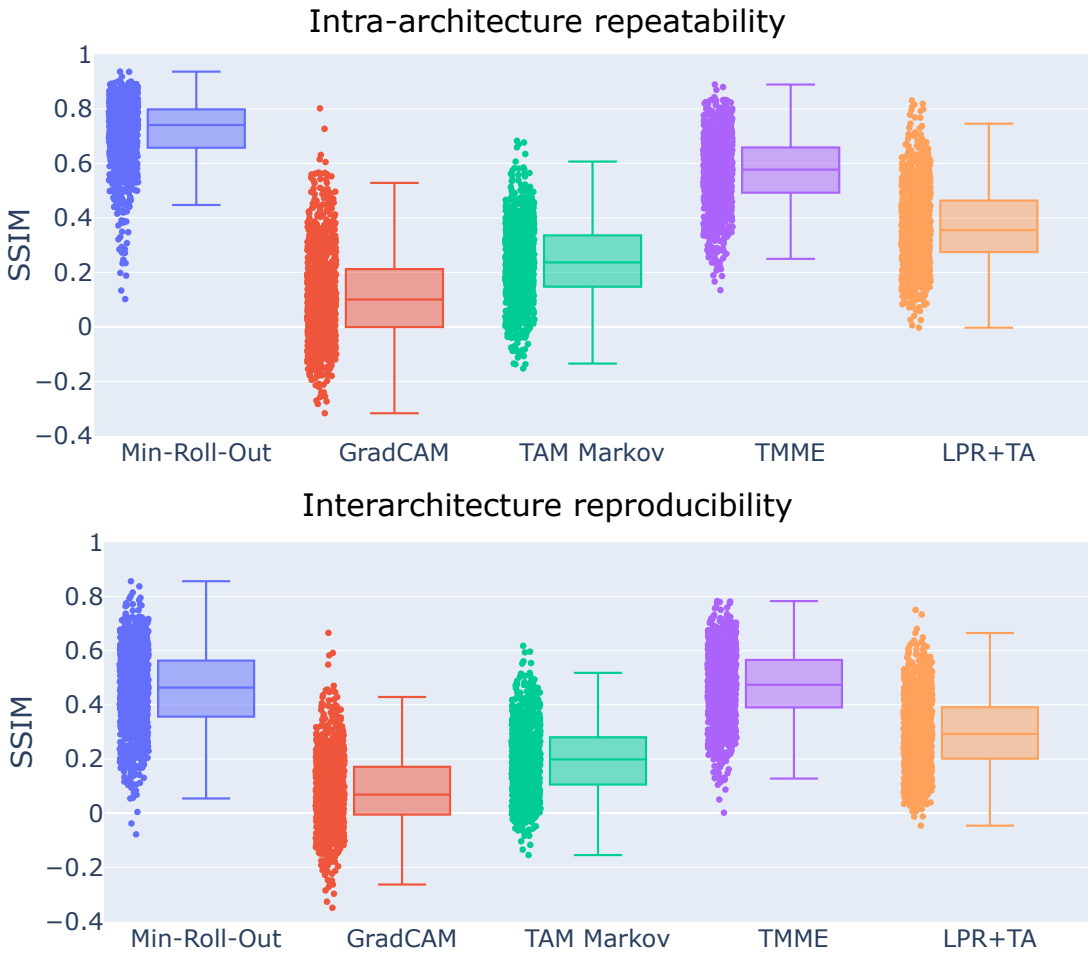

Figure 7.12: Intra-architecture repeatability and interarchitecture reproducibility. Intra-architecture repeatability is the SSIM score of two saliency maps of the same ViT (ViT_b_244) from different training sessions. Min-Roll-Out 0.71 (95 % CI: 0.71, 0.72); GradCAM 0.116 (95 % CI: 0.11, 0.13); TAM Markov 0.24 (95 % CI: 0.23, 0.25); TMME 0.57 (95 % CI: 0.56, 0.58); LPR+TA 0.37 (95 % CI: 0.36, 0.38); interarchitecture reproducibility is the SSIM score of two saliency maps of two different architectures: the reported ViT (ViT_b_244) and the smaller ViT (ViT_t_244). Min-roll-out 0.46 (95 % CI: 0.45, 0.47); GradCAM 0.082 (95 % CI: 0.074, 0.091); TAM Markov 0.20 (95 % CI: 0.19, 0.21); TMME 0.47 (95 % CI: 0.47, 0.48); LPR+TA 0.30- (95 % CI: 0.29, 0.31). During prototyping we investigated other, less performing, attention-based saliency map methods: min-roll-out, transition attention maps (TAM) Markov, and layer-wise relevance propagation transformer- attribution (LRP+TA). SSIM = structured similarity index measure

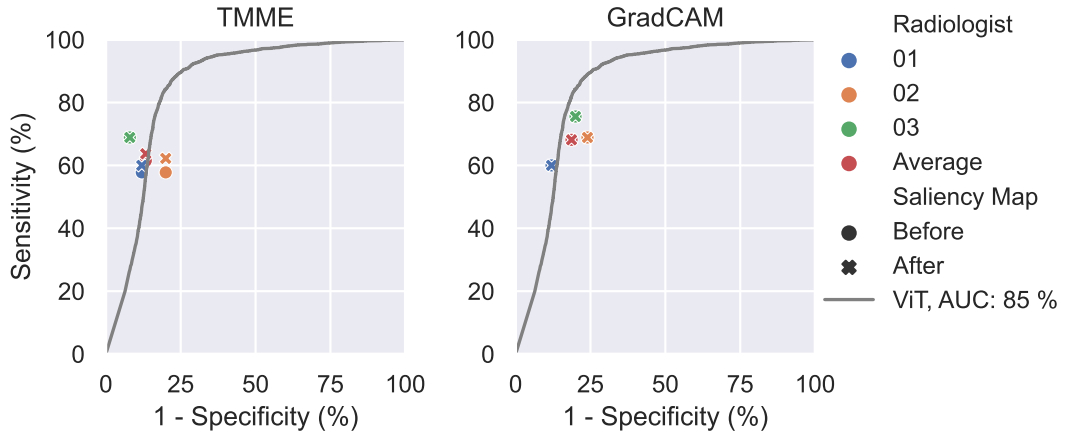

Figure 7.13: Receiver operating characteristic curves (ROC) and model AUC for the user study. Performance of the ViT model (grey) and the radiologists without (circles) and with (crosses) additional saliency map for pneumothorax classification. The average reader performance is shown by the red circle/cross. AUC = area under the ROC curve.

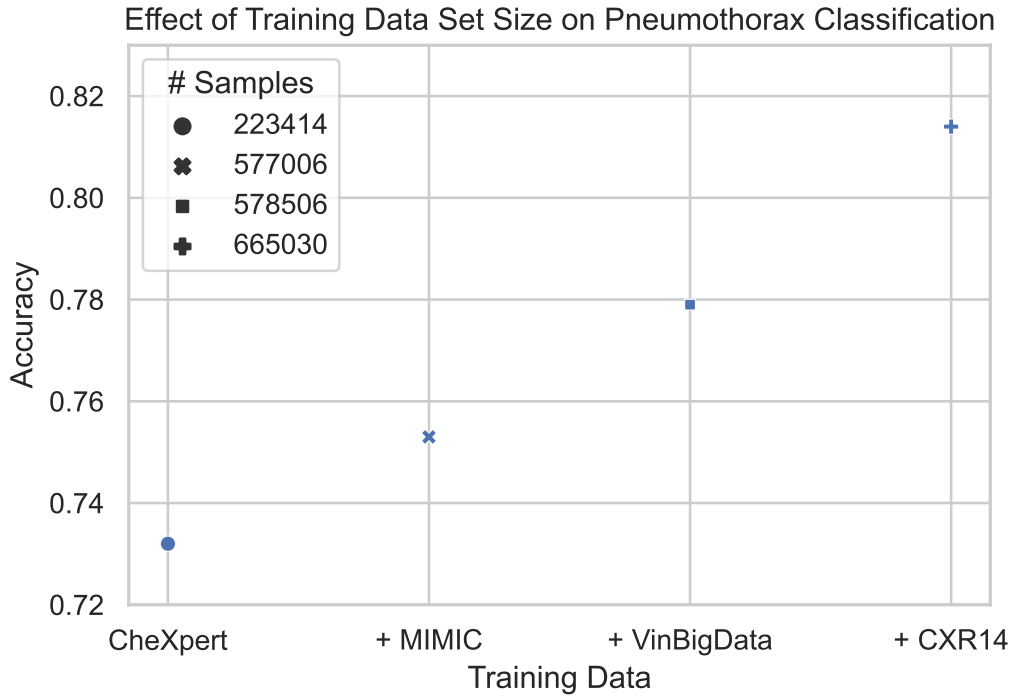

Figure 7.14: Effect of increased training data on pneumothorax classification on the SIIM ACR test data set. The classification thresholds were computed according to the best respective F1-score on the validation data. All networks were trained for the same number of iterations. CXR14 = Chest X-Ray 14, SIIM-ACR = Society for Imaging Informatics in Medicine-American College of Radiology.

8 A knee cannot have lung disease: out-of-distribution detection with in-distribution voting using the medical example of chest X-ray classification

Most of the work presented in this chapter has been submitted as part of an article to the Medical Physics journal on May 8th, 2023. The paper is currently under review (as of June 2023) [148].

8.1 Introduction

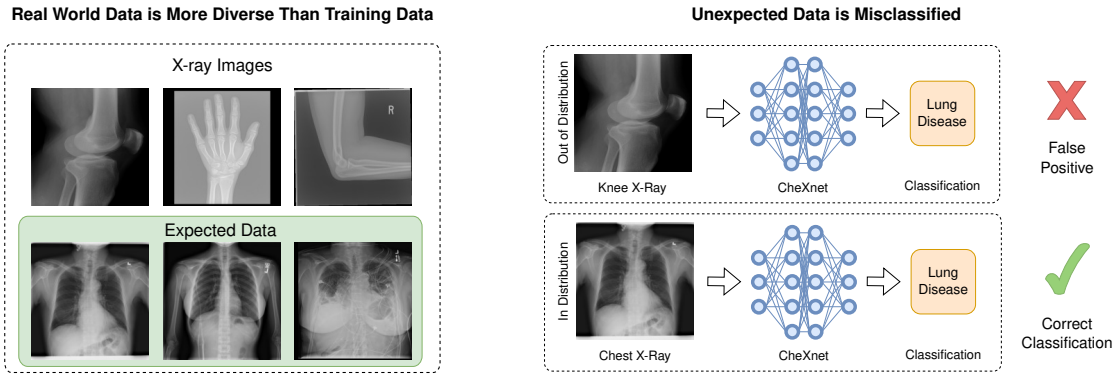

Figure 8.1: Deep learning models in the real world must be able to handle OOD data. (**left**) A chest X-ray classifier (CheXnet) is trained on chest X-rays and tested with expected production data: chest X-rays. In a clinic, the model has to handle non chest X-ray images confidently, as the data cannot be manually cleaned beforehand. (**right**) A model trained and tested only on chest X-ray images will incorrectly classify OOD images (here: an X-ray of a knee) as having lung disease.

Modern machine learning models are achieving great successes in real world medical applications, such as diabetic retinopathy diagnosis [149], skin cancer classification [150], or lung disease assessment [13, 89, 151]. Due to the early and profound digitization of imaging techniques, machine learning in radiology can already show convincing successes, such as the detection of certain critical pathologies of the lung on X-ray images with performance non-inferior to radiologists [13]. Considering the increasing demand for imaging, while the number of radiologists remains insufficient, the aforementioned and similar models can help improve medical patient care, for example, by screening acquired

radiographs for critical findings prior to radiologist interpretation [29, 102, 28, 24, 26, 94]. Then, patients with time sensitive illnesses will receive treatment earlier, potentially saving their lives.

What all of these chest X-ray classifiers have seen, once trained, validated and tested, are chest X-rays of a certain type, the in-distribution (ID) images. Consequently, the features learned depend on the assumption that the input is ID. But despite the advanced level of digitization, individual workflows for creating and archiving radiological images and linking them to other patient data are subject to manual intervention by staff and are consequently prone to human error, breaking this assumption. Just one example would be that mixed-up labeling can arise of patients for whom X-ray images of several body parts have been taken. Consequently, images of a knee joint, for example, would be fed to a model for detecting pulmonary pathologies. Hence, in the aforementioned scenario, the presentation of out-of-distribution (OOD) images, erroneous and potentially patient-harming events are possible.

A major problem of current deep learning models is that they make high confidence predictions when facing unexpected (OOD) data, like a knee X-ray [152, 153, 154]. In our scenario, prioritization based on false, high-confidence, OOD X-rays can lead to longer waiting times for other patients with time critical conditions, like a pneumothorax, potentially risking their life until the error is discovered and resolved. Moreover, repeated instances of such misreporting will - if not balanced with transparency measures sufficiently, e.g., using saliency maps [94] - quickly lead physicians to distrust the model, eventually leading them to refrain from using it [155, 156, 157].

Therefore, in recent years, several methods, have been proposed to detect OOD samples [158, 159, 160, 161, 162, 163]. Commonly, the OOD detector converts the output of a model to an ID score. For example, one of the earliest approaches, Max. Probability [159] uses the highest class probability as ID probability. In their experiments, the authors noticed lower confidence scores for the highest class probability for OOD inputs compared to ID inputs. Another approach, proposed by Lee et al. [164], also applied to chest X-rays [163], models OOD data based on the smallest Mahalanobis distance between the input and a class conditional Gaussian distribution in the latent space. Furthermore, Hendrycks et al. propose to use a self-supervised training scheme to improve OOD detection performance [162].

So far, the problem caused by OOD data has been investigated mostly on toy data sets, for example, a model trained on the CIFAR-10 data set [165], learning to classify automobiles and trucks, is tested on the SVHN data set [166] containing house numbers, or in-house data sets [163]. This raises the question if the test performance of proposed OOD detectors translate to an existing model trained on chest X-rays. Figure 8.1 motivates this problem: as the real world data consists of more than frontal chest X-rays, a classifier like CheXnet [13] must handle OOD images safely. Çallı et al. investigated the effect of an in-house collected OOD X-ray data set on the task of nodule classification, localization and lung segmentation [163]. In contrast, we focus on the more general multi-label chest X-ray classification problem.

In this work, we are addressing the practical consequences of OOD data by examining the impact of non chest radiographs on the chest X-ray classifier CheXnet. We

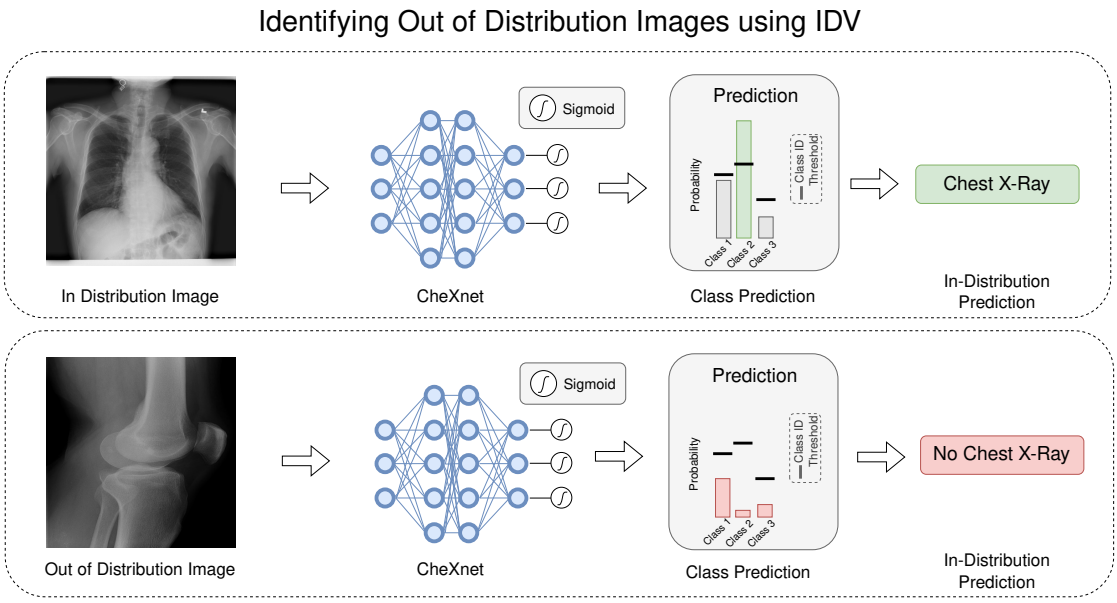

Figure 8.2: Out-of-distribution (OOD) detection using our proposed method in-distribution voting (IDV). The model is trained with in-distribution (ID) and OOD images. Before inference, class-wise ID threshold are used to classify the input images as either ID (chest X-ray, **top**) or OOD (knee, **bottom**) In this multi-label setting, a sample is classified as OOD only if all classes unanimously vote against ID.

selected this model as it performs similarly to radiologists [13, 89] and is widely used as a benchmark. The major contributions of our work are: we systematically explore the OOD detection performance of the CheXnet chest X-ray classifier on three realistic OOD data sets; we show that the benchmark performance of current OOD detection methods mostly do not translate to this domain; and we demonstrate that our proposed method in-distribution voting (IDV) improves OOD detection and generalizes to other data sets.

8.2 Methods

8.2.1 Chest X-Ray Classification: CheXnet

Following [13], we fine-tuned a DenseNet-121 [167] on the CXR14 data set. The model was pre-trained on ImageNet and the weights are available on `pytorch.org`. For fine-tuning, we replaced the last layer with a fully-connected layer with 15 outputs, matching the 14 classes of the CXR14 data set plus the "no finding" class which represents the absence of any of the 14 pathologies. We expanded the original CheXnet architecture from 14 to 15 outputs to differentiate between ID "no finding" CXRs and OOD samples. Since an image may exhibit signs of multiple pathologies, we modeled the classification task as a multi-label classification task, where each class is predicted independently. The output scores were converted to a probability for each class by applying the sigmoid

function:

$$\sigma(x) = \frac{1}{1 + e^{-x}.}$$

We used binary cross entropy as loss function and trained the model using ADAM [53] optimization with default parameters ($\beta_1 = 0.9, \quad \beta_2 = 0.999$) and an initial learning rate of 0.0003. We divided the learning rate by a factor of ten if the validation loss did not improve over the last two epochs. We applied weight decay with a value of 0.0001. We trained the model for eight epochs and selected the best model based on the mean area under the receiver operating characteristic curve (AUC) for classifying the ID validation data set. The input images were resized to 256 × 256 pixels and normalized according to the ImageNet mean and standard deviation. Then, we applied 224 × 224 ten crop, i.e., we took crops from each corner and the center of the image and repeated the process for the horizontally flipped image: producing ten 224 × 224 pixel images per sample. The model predictions of the ten images were averaged before calculating the loss.

When including OOD images into the training data, the model must predict the absence of any pathology. This is in contrast to healthy CXR images, where the "no finding" class must be predicted. In the default CheXnet setup this would result in predicting the same for both OOD and CXR with no finding. Like the ID images, the OOD images are normalized according to the ImageNet mean and standard deviation and passed to the model in the same fashion as the ID images.

8.2.2 Proposed Method: In-Distribution Voting

To improve the robustness of CheXnet's predictions and OOD detection performance we propose in-distribution voting (**IDV**). We classified a sample as ID if at least one class-wise prediction exceeds the class-wise ID threshold, as illustrated in Figure 8.2.

Unlike other multi-label OOD detection techniques in the literature, this approach leverages OOD data to separate actual image classification from OOD detection. We adapted approaches proposed in the literature [161, 168] and included OOD data in the training data set, known as outlier exposure [161] or negative data [169]. We motivate the use of OOD data during training to break the "closed world" assumption. In other words, we forced the model not to condition the predictions on the chest X-ray input assumption. In our case, the model was required to predict the absence of any class for OOD samples, resulting in a zero vector.

It is noteworthy that although "no finding" samples do not have any labeled classes, we consider them as ID, as they are chest X-rays. For such CXRs that exhibit no indications of the 14 classes, the model had to predict the "no finding" class. In our experiments, we also used unrealistic OOD samples such as photos from ImageNet during training, since they are expected to exist when employing a pre-trained model.

In a production setting, the ID thresholds would be set independently for each class utilizing the validation set that contains both ID and OOD images instead of calculating the AUC to report the general performance. When training with OOD images, both

training and validation splits are expanded to include the OOD training/validation splits.

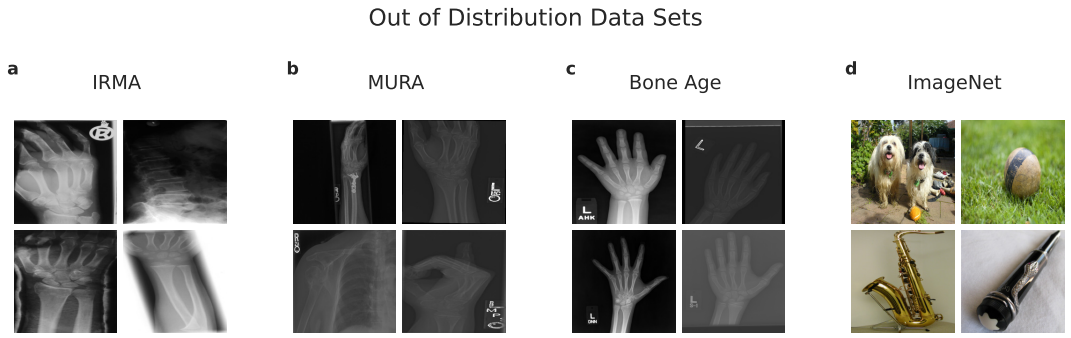

Figure 8.3: In this work, we utilized four out-of-distribution datasets: the Image Retrieval in Medical Applications (IRMA) data set [170] **(a)**, the Musculoskeletal Radiographs (MURA) data set [171] **(b)**, the Bone Age data set [172] **(c)**, and the ImageNet data set [103] **(d)**. The IRMA data set comprises a diverse collection of radiographic images, while the MURA data set contains solely upper extremity radiographs, and the Bone Age data set consists of hand radiographs. Lastly, the ImageNet data set is a collection of web-scraped photographs. All four data sets are publicly available.

8.2.3 Other Out-of-Distribution Detection Methods

In the literature, several methods for OOD detection have been proposed to explicitly filter OOD samples. Hendrycks and Gimpel were among the first to tackle this problem [158]. They utilized the maximum value of the softmax prediction as the ID probability. In their work, they justify this choice by observing that the highest prediction for OOD samples is lower than that of ID samples. Since softmax is commonly used for single-label classification problems, they extended the approach to multi-label classification tasks by using the maximum logit of the classification layer (**Max. Logit**) [159]. In contrast, Wang et al. used the label-wise energy function [173] instead of the sigmoid to transform the model output to an ID score (**Max. Energy**) [160].

Instead of converting the model's output to an ID score, several approaches use the activations of the model to generate class-conditional Gaussian distributions (**Mahalanobis**) [164, 163]. This method models OOD images as unlikely points in the class distribution, i.e., having a large Mahalanobis distance to the modeled class means in the latent space. Lee et al. [164] motivated the use of the Mahalanobis distance between the mean representation of a class and the input in the feature space, instead of performing OOD detection in the label space due to "label overfitting", i.e., that the model predictions are conditioned on the training labels. For Mahalanobis-based OOD detection, we use the output of the penultimate layer to determine the Mahalanobis scores similar to the work by Çallı et al [163].

Hendrycks et al. propose training the classification model with additional self-supervised heads to improve OOD robustness (**SS OOD**) [162]. In this method, the model has to additionally predict image rotation and translation. OOD detection is performed by taking the highest class prediction probability as ID score.

	In-	Out-of-Distribution				
Data Set	CXR14	IRMA	MURA	Bone Age	ImageNet	Subset
Pre-processing	-	Remove CXR	-	-	Sample	Sample
Training	78,468	3,088	35,366	8,179	217,818	3,088
Validation	11,219	772	772	772	54,455	772
Testing	22,433	3,860	3,860	3,860	3,860	3,860
Total	112,120	7,720	39,998	12,811	276,133	7,720

Table 8.1: Data Sets used in our experiments. The smallest out-of-distribution data set (IRMA) is split into 40/10/50 % training/validation/testing. To compare different scenarios we used the same number of images for validation and testing of the other OOD data sets (MURA [171], BoneAge [172], ImageNet [103]). The remaining images were used for training. Because the ImageNet data set is an order of magnitude larger than the ID CXR14 data set [78] we took a random sample first. To examine the different data set sizes we also fixed the size of the OOD training data splits to have the same amount of images (Subset). CXR = chest radiograph.

8.2.3.1 Data Sets

Not every OOD sample is equally likely in a real-world scenario. In a production setting, the CheXnet model can encounter OOD X-ray images, as the distinction between ID and OOD X-ray images is based on manual, error-prone tagging. Photographs on the other hand are not part of the image processing pipeline in a radiology department and can thus be assumed not to be found in a real-world scenario.

For our OOD detection experiments, we selected three publicly available radiographic data sets, IRMA [170], MURA [171], and BoneAge [172], containing X-ray images of various body parts as realistic OOD test data sets to test cross-data set generalization [169]. We specifically chose publicly available data sets to ensure reproducibility of our findings and encourage future work. Further data set details are listed in Table 8.1.

8.2.3.2 In-Distribution Chest X-ray 14

We use the train-test split provided by the authors of the Chest X-ray 14 (CXR14) data set, having non-overlapping patients. We further randomly split the provided training data set into training and validation sets, again with non-overlapping patients resulting in 78,468 training, 11,219 validation, and 22,433 test images (see also Table 8.1). All three splits have a similar prevalence of class labels. In summary, the original data set is split into 70 % training, 10 % validation, and 20 % test data. In contrast to the original CheXnet model, we expand the target classes and include “no finding” to differentiate between healthy CXR and other images. We also use the images labeled as “no finding” for training, as 46 % of the images are labeled as such. For these images, the model must predict the absence of all 14 pathologies in the original CheXnet setup.

8.2.3.3 Out-of-Distribution Data Sets

For our OOD detection experiments we use the following data sets:

- **IRMA**: the image retrieval in medical applications data set [170] consists of 14,410 diverse radiographic images; 12,677 are annotated according to the anatomical category, 1733 are test images without annotation. The original task was to predict the correct anatomical category.

- **MURA**: the musculoskeletal radiographs data set [171] consists of 40,561 radiographic images, displaying different upper extremity bones. The original task was to predict if the X-ray study is normal or abnormal.

- **BoneAge**: the Bone Age data set [172] consists of 12,811 hand radiographs of children. The original task was to predict the age of the patient.

- **ImageNet**: the ImageNet data set [103] contains over one million web scraped photographs. The data set is often used for pre-training computer vision models. There are several tasks for this data set, including image classification and object detection.

While the CheXnet model has been pre-trained on predicting the ImageNet classes, they are OOD regarding the target task of chest X-ray classification, as the data set does not include chest X-rays. Therefore, we use it as additional non chest X-ray OOD data set, allowing us to investigate the performance of our proposed method "In-Distribution Voting" (IDV). This is relevant for use cases where no or only few realistic OOD images are available. The OOD data sets are illustrated in Figure 8.3.

Because the IRMA data set is the smallest data set, we sample every OOD data set so that their test and validation split sizes match the IRMA splits. Furthermore, we create a subset of every OOD data set to account for training data size.

IRMA We only use the provided training images, as we require the IRMA labels to exclude ID chest radiographs from the data set. We remove all chest X-rays from the data set according to their anatomical code and exclude images with an anatomical code starting with 57, 75, 05, or 150, resulting in 7,720 images. We split the remaining images randomly into training, validation, and testing using a 30 % / 20 % / 50 % split to ensure enough images in the test split.

Bone Age We randomly sample the test and validation images according to the data split sizes of the IRMA data set (772 validation images, 3,860 test images, see Table 8.1). The remaining 8,179 images are either used all or sampled according to the IRMA training set size (3,088 images) for the training split.

MURA We split the MURA data set, containing 40,561 images, similar to the Bone Age data set: the validation and test partitions are randomly sampled, matching the size of the IRMA validation/test splits, listed in Table 8.1. Either all remaining images or ones sampled according to the IRMA training set size (3,088 images) are used for training.

ImageNet Due to the size of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) data set compared to the Chest X-ray 14 data set we use only 50 % of the 544,546 images provided in the "LOC_train_solution.csv" file for training and another 20 % for validation, see Table 8.1. When accounting for OOD data set sizes, we sample the training and validation sets according to the size of the IRMA splits. For both cases we sample the test set according to the IRMA test split size. All train/validation/test splits were created with non-overlapping images.

8.3 Results

8.3.1 Chest X-ray Classification

We trained the CheXnet model successfully on the Chest X-ray 14 (**CXR14**) data set and evaluated the impact of training with OOD data on chest disease classification performance by measuring the performance on the ID test data set, without any OOD samples. Furthermore, we report the CXR classification results when training with SS OOD heads.

We report the mean AUC over all 15 classes, as well as the AUC for each individual class. The model achieved a mean AUC of 83 % when trained and tested on the CXR14 data set without any OOD images, as shown in Table 8.2. Figure 8.4 displays the corresponding receiver operating characteristic (ROC) curves for all 15 classes in the CXR14 data set.

To further evaluate the performance of the model, we trained it with additional self-supervised heads, as it modifies the training procedure, which resulted in a reduction in the classification mean AUC to 81.1 %. We also tested the model's performance when trained with OOD samples from the IRMA and ImageNet data sets. In contrast to the self-supervised training scheme, the mean AUC improved to 83.3 % when including IRMA and ImageNet OOD data (3088 samples). Trained with only IRMA or ImageNet data the mean AUC resulted in 82.7 % and 82.4 %, respectively. Table 8.2 shows the AUCs obtained when training with all data set combinations, including MURA and Bone Age, which resulted in similar AUCs of 83.2 % and 83.0 %, respectively. However, using the full ImageNet OOD set resulted in a lower AUC of 81.1 %.

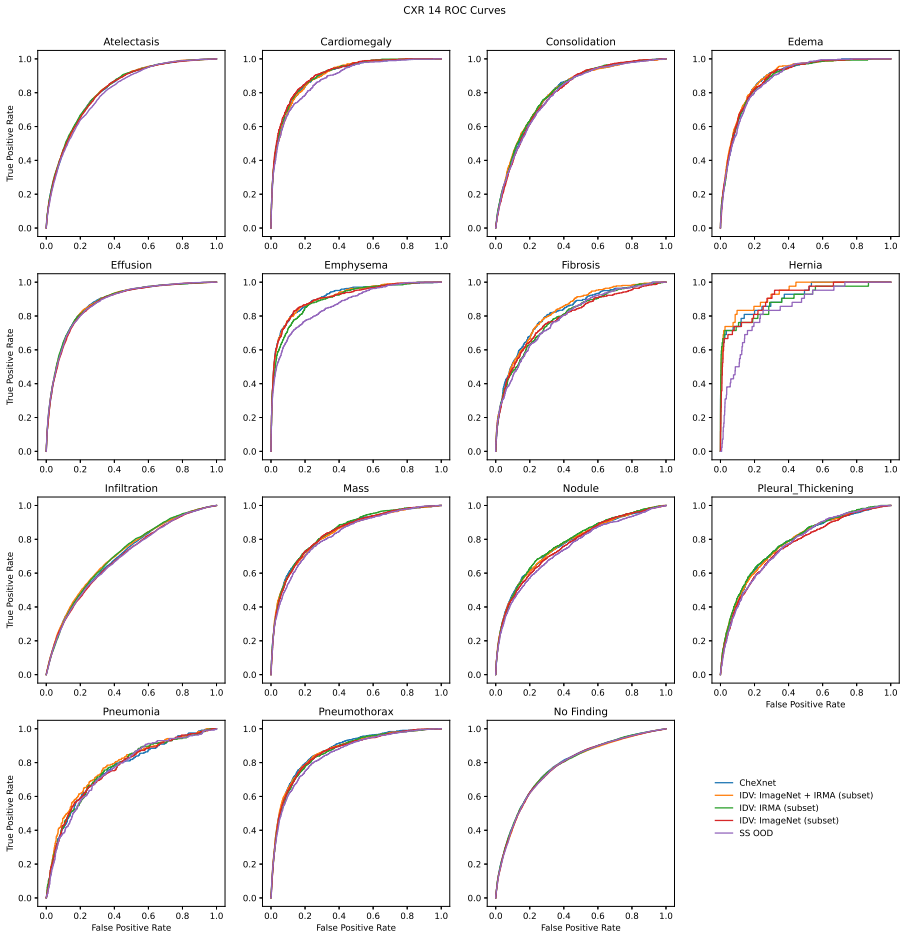

Figure 8.4: ROC curves of all CXR14 classes for the main experiment settings. Training with OOD data, as proposed by our method, IDV, had no clear negative effect on CXR classification. In contrast, training with self-supervised heads (SS OOD) affected the classification negatively. The depicted IDV runs were trained with 3088 OOD images (subset).
ROC = Receiver Operating Characteristic, OOD = out-of-distribution, IDV = in-distribution voting, CXR = Chest X-ray.

	IDV	IDV	IDV	IDV	CheXnet	IDV	IDV	IDV	IDV	SS OOD	IDV	IDV	IDV
Training Data													
CXR14	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
IRMA	✓	-	-	-	-	-	-	✓	-	-	✓	-	-
MURA	-	-	-	✓	-	-	✓	-	-	-	-	-	-
Bone Age	-	✓	✓	-	-	✓	-	-	-	-	-	✓	-
ImageNet	✓	-	-	-	-	✓	-	-	✓	-	✓	✓	✓
OOD Train Samples	3088	8179	3088	3088	0	3088	35366	3088	3088	0	220906	225997	217818
Atelectasis	81.9	81.7	81.9	81.8	81.8	81.9	81.2	**82.0**	81.6	80.6	81.1	80.4	79.8
Cardiomegaly	90.3	90.3	90.6	89.4	90.1	90.1	90.7	90.8	**91.0**	88.7	**91.0**	89.9	89.2
Consolidation	80.5	80.1	**81.0**	80.8	80.3	80.7	79.4	80.6	79.8	79.5	79.7	80.3	78.9
Edema	**89.3**	88.9	88.5	89.1	88.8	88.5	88.0	88.3	88.7	88.1	86.3	87.0	86.5
Effusion	88.0	87.9	**88.1**	87.7	88.0	88.1	87.7	88.0	87.5	87.8	87.1	87.5	87.3
Emphysema	91.4	**92.2**	91.4	92.0	91.6	91.3	91.1	90.2	91.2	87.0	87.8	87.3	87.4
Fibrosis	**82.6**	82.1	79.9	82.1	82.1	80.4	82.3	79.6	79.3	79.0	79.4	77.9	78.3
Hernia	93.6	**94.3**	91.5	92.8	91.3	90.9	92.9	89.7	91.1	84.2	82.6	83.5	82.1
Infiltration	71.0	**71.3**	71.1	70.8	70.2	71.1	70.3	70.8	69.2	69.1	70.1	69.3	69.0
Mass	83.7	84.7	**84.8**	84.4	84.2	83.9	84.5	84.5	84.0	82.6	82.7	81.7	80.6
Nodule	77.4	77.9	**78.5**	75.7	77.3	77.6	76.9	77.9	76.6	75.0	74.7	73.8	72.8
Pleural Thickening	77.6	77.1	77.7	77.6	77.3	77.5	78.1	**78.2**	75.7	76.4	75.6	75.0	74.8
Pneumonia	**77.7**	75.9	77.6	75.5	76.0	76.8	75.0	76.4	76.1	75.6	74.8	74.3	74.2
Pneumothorax	86.9	86.6	86.2	87.3	**87.4**	86.7	85.6	86.2	86.4	85.1	84.7	85.3	84.5
No Finding	77.6	78.2	**78.3**	77.9	78.1	78.0	77.9	78.0	77.7	77.7	77.8	77.4	77.3
Mean AUC	**83.3**	**83.3**	83.2	83.0	83.0	82.9	82.8	82.7	82.4	81.1	81.0	80.7	80.2

Table 8.2: CXR14 classification performance evaluated by the AUC. Experiments are sorted by mean AUC, with the best AUC highlighted in **bold**. The CheXnet baseline method achieved a mean AUC of 83 %. Our proposed in-distribution voting (IDV) method, which trained with few (3088) OOD images, had no clear negative effect on CXR classification, with mean AUCs ranging from 82.4 % to 83.3 %. However, training with an OOD data set larger than the in-distribution data set reduced the mean classification AUC by up to three percentage points. Additionally, incorporating self-supervised heads (SS OOD) had a negative effect on the classification AUC by two percentage points. AUC = area under the ROC curve, OOD = out-of-distribution, CXR = Chest X-ray.

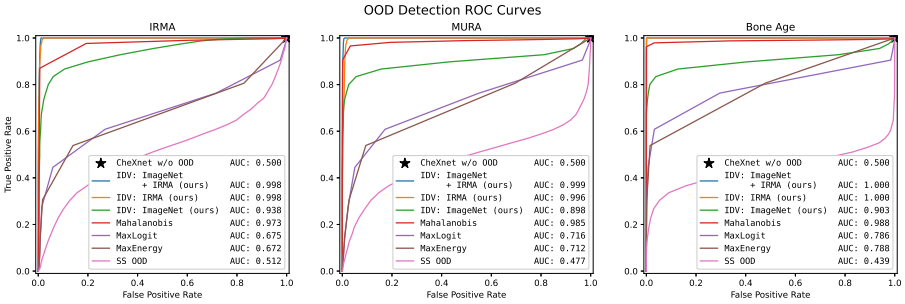

Figure 8.5: ROC curves for OOD detection on the test datasets of CXR14 + IRMA, MURA, Bone Age with their respective AUC. The CXR classifier, CheXnet, cannot handle OOD data itself, resulting in a false positive rate of 100 % on all test datasets. This means that all OOD images were classified as having lung disease by the base model. Training the model with self-supervised heads (SS OOD) improved the OOD detection AUC only on the IRMA dataset, not on MURA and Bone Age. Converting the model's output to an OOD detection score (MaxLogit, MaxEnergy) improved the OOD AUC on all three datasets. Using the Mahalanobis distance to the class means in the feature space as an OOD signal resulted in an AUC greater than 97 % on all three datasets. Our proposed method, IDV, performed best with an average OOD detection AUC of 99.9 % across all three datasets when trained with ImageNet and IRMA data. Training with a domain-specific OOD dataset (IRMA) performed better than using only a general dataset (ImageNet), and training with a diverse OOD dataset containing domain-specific OOD data as well (ImageNet + IRMA) performed best. All IDV runs were trained with a subset (3088 images) of the available OOD training data, using 1044 ImageNet and 1044 IRMA images in the case of ImageNet + IRMA. ROC = Receiver Operating Characteristic, OOD = out-of-distribution, AUC = area under the ROC curve, CXR = Chest X-ray, IDV = in-distribution voting, CXR14 = Chest X-Ray 14, IRMA = image retrieval in medical applications data set, MURA = musculoskeletal radiographs data set.

8.3.2 Out-of-Distribution Detection

The objective of OOD detection is to classify each image as either ID or OOD. For each of the three OOD data sets (IRMA, MURA, and BoneAge), we evaluate the performance of the OOD detection methods by measuring how many ID and OOD samples from the test set are correctly classified as such. As a baseline, we employed the default CheXnet model with no extra OOD detection mechanism, which represents the current CXR classification models. We report the AUC as our evaluation metric.

Figure 8.5 shows the ROC plots for the three different OOD data sets with their corresponding AUCs. CheXnet, without any OOD detection method, failed to filter any OOD image in all data sets, with a false positive rate of 100 % and an AUC of 50 %.

Training CheXnet with self-supervised heads (SS OOD) increased the OOD AUC to 51.2 % on the IRMA data set but resulted in a worse OOD detection performance on the MURA and Bone Age data sets with 47.7 % AUC and 43.9 % AUC, respectively.

The conversion of the logits to an OOD score using MaxLogit and MaxEnergy improved the OOD performance considerably compared to the CheXnet model with MaxLogit achieving AUCs of 67.5 %, 71.6 %, and 78.6 %, respectively on the IRMA, MURA, and

Bone Age data sets. MaxEnergy performed similarly, with 67.2 %, 71.2 %, and 78.8 %, respectively.

Using the Mahalanobis distance increased the OOD detection performance significantly compared to MaxLogit and MaxEnergy with an AUC of 97.3 % on the IRMA data set, 98.5 % on MURA, and 98.8 % on Bone Age.

Our method, IDV, trained with 1544 ImageNet and 1544 IRMA images surpassed the Mahalanobis performance on all three data sets with an AUC of 99.8 % on the IRMA data set, 99.9 % on the MURA data set, and 100 % on the Bone Age data set. Training with IRMA images resulted in 99.8 %, 99.6 %, and 100 % AUC on IRMA, MURA, and Bone Age, respectively. IDV with ImageNet in 93.8 %, 89.8 %, and 90.3 %, respectively.

8.3.3 Effect of Out-of-Distribution Training Data

To investigate the impact of out-of-distribution training data selection, we conducted a series of experiments to measure the performance of OOD detection, using the area under the receiver operating characteristic curves. We trained our model using all available OOD data sets, including IRMA, MURA, Bone Age, and ImageNet. To account for the smaller sample size of IRMA and Bone Age data sets, we combined them with the ImageNet data set. Specifically, we chose ImageNet training data larger than the in-distribution CXR14 training data to measure the effect of using more OOD than ID data. To ensure a fair comparison among different OOD training data sets, we randomly sampled a subset of 3088 images from each data set, matching the smallest data set size (IRMA). When training using two data sets (ImageNet + IRMA, ImageNet + Bone Age), we selected 50 % from each, resulting in 1544 images from ImageNet and 1544 radiographs from either IRMA or Bone Age. In all experiments, we used a test set consisting of 3860 samples (see Table 8.1).

Figure 8.6 shows the ROC curves and AUC values for the different IDV runs evaluated on the thee test data sets: IRMA, MURA, and Bone Age. Generally, our proposed method outperformed the CheXnet baseline (AUC 50 %) when trained on any OOD data. Also, the models performed best on the Bone Age data set, containing only hand X-rays, and worst on the IRMA data set, containing a wide variety of radiographs. Training with ImageNet data generalized to X-ray OOD data sets with AUCs from 97 % to 100 % when trained on the whole data set and from 90 % to 94 % when trained on the subset. Training on the Bone Age data set performed worst on the IRMA data set (AUC 71 %) but achieved an AUC of 100 % on the Bone Age test set; using only a subset improved the performance and combining the data with ImageNet even further. Training on the MURA and IRMA data set individually performed best, but was exceeded only by the combination of ImageNet and IRMA data. Overall, our results suggest that incorporating diverse data sets, such as ImageNet and IRMA, is a promising approach to improve OOD generalization for X-ray classification tasks.

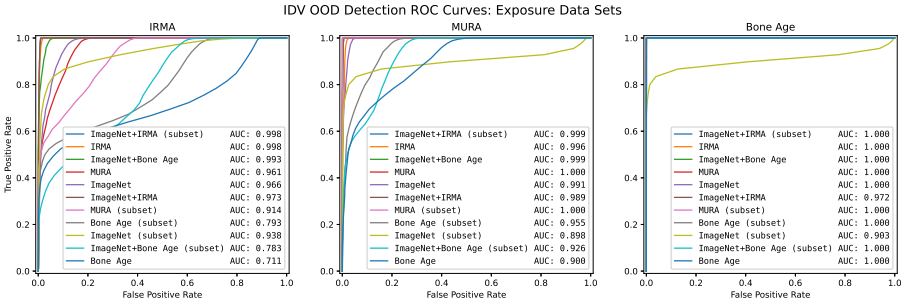

Figure 8.6: ROC curves and AUCs of all OOD detection runs using IDV on three OOD test data sets: IRMA, MURA, and Bone Age with CXR14 in-distribution data. IDV OOD detection with any OOD data improved OOD detection performance. Generally, all models performed best on the Bone Age data set, which includes only hand X-rays, and the worst on IRMA, which comprises a variety of X-rays. Consequently, using only the specific Bone Age data during training improved OOD detection performance less than using the diverse ImageNet data set, expect on the Bone Age test data. Training with ImageNet OOD images provided strong OOD detection performance, with an AUC greater than 96 % on all data sets. Additionally, training with the most diverse data set, ImageNet + IRMA, provided the overall best performance, using only 3088 training images (subset). ROC = Receiver Operating Characteristic, OOD = out-of-distribution, AUC = area under the ROC curve, CXR = Chest X-ray, IDV = in-distribution voting, IRMA = image retrieval in medical applications data set, MURA = musculoskeletal radiographs data set.

8.4 Discussion

Assessing whether the tested model performance in a benchmark translates to an intended production setting, including potential OOD data is a necessary step before deploying a machine learning model. This is particularly important in safety critical applications, e.g. when classifying chest X-rays to assist radiologists in diagnosing patients. Our results show that the CheXnet model cannot handle OOD samples out-of-the-box. However, combining it with our proposed method, IDV, and trained with any OOD data, even the photographs of ImageNet, improved the OOD detection performance compared to the baseline CheXnet model and most OOD detection methods considerably with OOD detection AUCs up to 100 %.

In their paper, Rajpurkar et al. conclude that their CheXnet model exceeds practicing radiologists in detecting pneumonia [13] and note as limitation that only frontal CXR were used, giving a potential low estimate of the model's performance. Our experiments showed that a further limitation was not considered: out-of-distribution images. A model that cannot handle OOD images, making confident predictions based on wrong evidence, will lead to worse quality of care, eroding the trust of physicians into the model's predictions when facing ID images, and impede the potential benefits of computer assisted diagnosis. Having robust models trusted by radiologists is necessary to leverage such classifiers to assist radiologists in clinical practice. Out-of-the-box, the CheXnet model failed to provide this robustness against realistic OOD images.

In our experiments, we noted that other OOD detection methods based on a model's output (Maxlogit, MaxEnergy, and SS OOD) performed considerably worse than in their original works. This suggests that the presented OOD data sets are more challenging, highlighting the importance of considering OOD data in the medical domain. Our experiments showed that even limited OOD data leveraged by IDV had a large effect on the OOD detection performance without negatively affecting the intended classification task.

We interpret the large OOD detection difference between the output based methods, MaxLogit and MaxEnergy, and Mahalanobis as evidence for the "label overfitting" hypothesis. Our approach, including an "no finding" / OOD label into the training procedure, breaks this overfitting problem and improves the OOD detection performance without a complex clustering component like the Mahalanobis distance. We interpret the IDV results as indicating that the model incorporates the fact that OOD images exist into its output.

While training with OOD data improves OOD detection performance, it is important to consider the intended use-case: CXR classification. We can conclude that training with OOD data, as proposed in our method IDV, does not affect chest disease classification performance negatively. This means that diversifying the training and validation data set with OOD samples improves the model performance in real-world scenarios, as potential OOD images are filtered.

Regarding the choice and availability of OOD training data our results showed that only few thousand samples are sufficient and even unrelated OOD data, such as ImageNet, is immensely useful. When comparing the OOD performance trained on the very specific Bone Age data set and the general ImageNet for cross-data set generalization we note that training with unrelated ImageNet data generalized better. We therefore conclude that using a generic OOD data set alone could improve a model's OOD detection performance. Including domain specific, diverse OOD images improves the OOD detection AUC even further (cf. ImageNet vs. ImageNet + IRMA in Figure 8.6).

In this work, we investigated the effect of OOD images on a chest X-ray classifier. We showed that the model, reportedly performing as good as radiologists [13, 89], was not able to filter OOD images, leading to obvious false positives to the human observer. We assume its predictions are conditioned on chest X-rays, because the model was only trained on chest X-rays, leading to overconfident predictions given OOD images. As hypothesized by Lee et al. [164], this leads to an ID-overfitted output space. This interpretation explains why established output-based OOD detection methods failed in our experiments, when compared to detecting OOD samples in the feature space. Our solution, ID voting and training with OOD images, regularizes the output space and expands the model's knowledge horizon, leading up to a 100 % ID OOD detection AUC.

One reason why OOD data are rarely considered is their dependency on the intended application. We showed that including a small OOD training data set from the same data set as the OOD test data resulted in a better OOD detection performance than a general OOD data set. While this suggests that there is no ideal application independent OOD data set, we found that training with any OOD data improved the baseline performance considerably. Furthermore, we showed that even a few thousand OOD samples from the

intended application boosted the specificity considerably. Therefore, when creating a data set to train and evaluate a model in a production setting, we recommend to remove anomalies, outliers and other OOD with caution. Instead, including this "real-world" data not only in the training process, but also into the model validation, will lead to more robust ML models and ultimately improve clinical acceptance

One limitation of this work is that we use the CheXnet model as a representative for other chest X-ray classification models. While we argue that this architecture is a strong baseline, further research is necessary to determine if our findings translate to other architectures. Furthermore, we only tested our model on chest X-ray images, even though our approach remains relevant to all multi-label OOD detection data sets. Finally, this retrospective work was performed only on public data and further work is necessary to evaluate our findings on real-world clinical data.

8.5 Conclusion

In conclusion, our study demonstrated that training solely on ID data leads to incorrect classification of OOD images as ID, resulting in increased false positive rates. We also showed, that our proposed method, IDV, substantially improves the model's ID classification performance, even when trained with data that will not occur in the intended use case or test set. Consequently, our approach makes the final model more robust and considerably improves its predictive performance in a real-world setting.

9 Discussion

The purpose of this dissertation was to develop and improve the chest X-ray classification pipeline for supporting radiologists in their diagnoses. The pipeline includes data anonymization and annotation, pre-processing, classification, interpretability, and robustness during deployment in the clinic. Throughout the dissertation, models and methods were proposed for significant improvements of each part.

Despite the progress made, there are areas for further development. For example, simply comparing radiologist routine and current advances in deep learning-based diagnoses highlights opportunities for further advancements. Currently, automated chest X-ray diagnoses are based on only the frontal or frontal and lateral views, while radiologists consider patient history and refer to previous studies for their diagnoses. For clinicians, highlighting changes between two imaging studies could be even more important than the diagnosis of an already known pathology. Limited by available data, current models are not addressing these multi-image settings.

Moreover, most models are only image-based, as patient meta data and radiology reports were not released alongside the images. This is likely due to data privacy concerns and time constraints. However, the data anonymization algorithm and automatic report labeler presented in Chapter 4 could address these problems. Besides multi-modal modeling, releasing the underlying reports would also provide a form of forward-compatibililty for future labeling methods.

Another challenge is the lack of a common labeling standard in large-scale chest X-ray datasets. While an exhaustive list of labels may not necessarily address this issue, common entities in reports could be analyzed to improve labeling. Additionally, predicting bounding boxes could increase usability considerably, but missing ground truth labels are a bottleneck for such predictions. To address this, semi-automatic methods, similarly to our proposed report labeler, could reduce annotation time significantly.

Another aspect was highlighted by the recent COVID-19 pandemic. For some important chest X-ray pathologies, there will only be a handful of example images available. Here, zero or few-shot approaches could improve classification models for cases with limited data.

Finally, downstream applications based on chest X-ray classification such as prioritization of incoming images and report generation could greatly benefit from publicly available, state-of-the-art models, similarly to pre-trained ImageNet classifiers or large language models.

Overall, data gathering is still the most important reason that is holding the development of advanced decision support systems back. Working closely with the radiologists can help in the development of efficient data annotation tools and uncover gaps between research and problems in the clinical practice.

10 Conclusion

This dissertation investigated several crucial components of chest radiography processing pipelines, from data processing to image-based pathology classification. The work of this doctoral project was conducted in close collaboration with radiologists. The objective was to address their challenges in clinical practice and to propose and evaluate corresponding solutions to support them.

After elucidating the benefits of applying deep learning for radiologist decision support systems (Chapter 1), deep learning concepts, tools, and resources were introduced (Chapter 2). Then, chest X-ray data sets and models were presented (Chapter 3), followed by processing radiology reports (Chapter 4), and assessing the effects of image resolution (Chapter 6) and windowing (Chapter 5) on model performance. To improve interpretability of decision support systems, the use of attention-based saliency maps for chest X-ray classifications with vision transformers was proposed (Chapter 7). Finally, to ensure safe use of such systems in clinical practice, an out-of-distribution detection method was suggested (Chapter 8).

As discussed in the previous chapter, current deep learning models already have the potential to improve patient care in everyday clinical practice. The results of this dissertation suggest that the necessary key components for clinical image processing software are now available, and I am looking forward to see these components being used in clinical applications.

A Appendix

A.1 Journal Publications

A. Wollek, R. Graf, S. Čečatka, N. Fink, T. Willem, B. Sabel, and T. Lasser, "Attention-Based Saliency Maps Improve Interpretability of Pneumothorax Classification", *Radiology: Artificial Intelligence*, 5.2 (2023).

A.2 Submitted Journal Publications

- **A. Wollek**, T. Willem, M. Ingrisch, B. Sabel, and T. Lasser, "A knee cannot have lung disease: out-of-distribution detection with in-distribution voting using the medical example of chest X-ray classification". Submitted to *Medical physics*, in May 2023.
- **A. Wollek**, S. Hyska, T. Sedlmeyr, P. Haitzer, J. Rueckel, B. Sabel, M. Ingrisch, and T. Lasser, "German CheXpert Chest X-ray Radiology Report Labeler". Submitted to *RöFo-Fortschritte auf dem Gebiet der Röntgenstrahlen und der bildgebenden Verfahren*, in June 2023.

A.3 ArXiv Pre-Prints

The following pre-prints are being prepared for publication (as of June 2023).

- **A. Wollek**, S. Hyska, T. Sedlmeyr, P. Haitzer, J. Rueckel, B. Sabel, M. Ingrisch, and T. Lasser, "Automated Labeling of German Chest X-Ray Radiology Reports using Deep Learning ", ArXiv (2023).
- **A. Wollek**, S. Hyska, B. Sabel, M. Ingrisch, and T. Lasser, "WindowNet: Learnable Windows for Chest X-ray Classification ", ArXiv (2023).
- **A. Wollek**, S. Hyska, B. Sabel, M. Ingrisch, and T. Lasser, "Exploring the Impact of Image Resolution on Chest X-ray Classification Performance ", ArXiv (2023).

Bibliography

[1] W. K. Röntgen. "Über Eine Neue Art von Strahlen: Vorläufige Mitteilung". In: *Sitzungsber. Phys. Med. Gesell.* (1895).

[2] William G. Bradley. "History of Medical Imaging". In: *Proceedings of the American Philosophical Society* 152.3 (2008), pp. 349–361.

[3] Darlene Berger. "A Brief History of Medical Diagnosis and the Birth of the Clinical Laboratory. Part 1–Ancient Times through the 19th Century". In: *MLO Med Lab Obs* 31.7 (1999), pp. 28–30.

[4] Andreas Maier et al., eds. *Medical Imaging Systems: An Introductory Guide*. Vol. 11111. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-96519-2 978-3-319-96520-8. DOI: 10.1007/978-3-319-96520-8. (Visited on 04/12/2023).

[5] Directorate-General for Energy (European Commission). *Medical Radiation Exposure of the European Population*. LU: Publications Office of the European Union, 2015. ISBN: 978-92-79-45374-8. (Visited on 09/06/2022).

[6] V. Gershan, S. Nestoroska Madjunarova, and E. Stikova. "Survey on the Frequency of Typical X-Ray Examinations and Estimation of Associated Population Doses in the Republic of Macedonia". In: *CONFERENCE ON MEDICAL PHYSICS AND BIOMEDICAL ENGINEERING*. 2013, p. 14.

[7] American College of Radiology. "ACR–SPR–STR Practice Parameter for the Performance of Chest Radiography". In: *Reston, Va: American College of Radiology* (2017).

[8] Elizabeth Puddy and Catherine Hill. "Interpretation of the Chest Radiograph". In: *Continuing Education in Anaesthesia Critical Care & Pain* 7.3 (June 2007), pp. 71–75. ISSN: 1743-1816. DOI: 10.1093/bjaceaccp/mkm014. (Visited on 05/11/2023).

[9] Thomas H. Payne. "Computer Decision Support Systems". In: *Chest* 118.2, Supplement (Aug. 2000), 47S–52S. ISSN: 0012-3692. DOI: 10.1378/chest.118.2_suppl.47S. (Visited on 09/09/2022).

[10] Reed T. Sutton et al. "An Overview of Clinical Decision Support Systems: Benefits, Risks, and Strategies for Success". In: *npj Digital Medicine* 3.1 (Feb. 2020), pp. 1–10. ISSN: 2398-6352. DOI: 10.1038/s41746-020-0221-y. (Visited on 02/08/2022).

[11] Ali Syed and Adam Zoga. "Artificial Intelligence in Radiology: Current Technology and Future Directions". In: *Seminars in Musculoskeletal Radiology* 22.05 (Nov. 2018), pp. 540–545. ISSN: 1089-7860, 1098-898X. DOI: 10.1055/s-0038-1673383. (Visited on 01/04/2021).

[12] Catherine Curtis et al. "Machine Learning for Predicting Patient Wait Times and Appointment Delays". In: *Journal of the American College of Radiology: JACR* 15.9 (Sept. 2018), pp. 1310–1316. ISSN: 1558-349X. DOI: 10.1016/j.jacr.2017.08.021.

[13] Pranav Rajpurkar et al. "Chexnet: Radiologist-level Pneumonia Detection on Chest x-Rays with Deep Learning". In: *arXiv preprint arXiv:1711.05225* (2017). DOI: 10.48550/arXiv.1711.05225. arXiv: 1711.05225.

[14] Anna Majkowska et al. "Chest Radiograph Interpretation with Deep Learning Models: Assessment with Radiologist-Adjudicated Reference Standards and Population-Adjusted Evaluation". In: *Radiology* 294.2 (2020), pp. 421–431.

[15] Ivo Baltruschat et al. "Smart Chest X-ray Worklist Prioritization Using Artificial Intelligence: A Clinical Workflow Simulation". In: *European Radiology* 31.6 (June 2021), pp. 3837–3845. ISSN: 1432-1084. DOI: 10.1007/s00330-020-07480-7.

[16] Jarrel C Y Seah et al. "Effect of a Comprehensive Deep-Learning Model on the Accuracy of Chest x-Ray Interpretation by Radiologists: A Retrospective, Multireader Multicase Study". In: *The Lancet Digital Health* (July 2021). ISSN: 2589-7500. DOI: 10.1016/S2589-7500(21)00106-0. (Visited on 07/05/2021).

[17] Alejandro Rodríguez-Ruiz et al. "Detection of Breast Cancer with Mammography: Effect of an Artificial Intelligence Support System". In: *Radiology* 290.2 (Feb. 2019), pp. 305–314. ISSN: 0033-8419. DOI: 10.1148/radiol.2018181371. (Visited on 09/09/2022).

[18] Axel Wismüller and Larry Stockmaster. "A Prospective Randomized Clinical Trial for Measuring Radiology Study Reporting Time on Artificial Intelligence-based Detection of Intracranial Hemorrhage in Emergent Care Head CT". In: *Medical Imaging 2020: Biomedical Applications in Molecular, Structural, and Functional Imaging*. Vol. 11317. SPIE, Feb. 2020, pp. 144–150. DOI: 10.1117/12.2552400. (Visited on 02/04/2022).

[19] Mohammad R. Arbabshirani et al. "Advanced Machine Learning in Action: Identification of Intracranial Hemorrhage on Computed Tomography Scans of the Head with Clinical Workflow Integration". In: *npj Digital Medicine* 1.1 (Apr. 2018), pp. 1–7. ISSN: 2398-6352. DOI: 10.1038/s41746-017-0015-z. (Visited on 02/04/2022).

[20] Koichiro Yasaka and Osamu Abe. "Deep Learning and Artificial Intelligence in Radiology: Current Applications and Future Directions". In: *PLOS Medicine* 15.11 (Nov. 2018), e1002707. ISSN: 1549-1676. DOI: 10.1371/journal.pmed.1002707. (Visited on 06/01/2023).

[21] Center for Devices and Radiological Health. "Artificial Intelligence and Machine Learning (AI/ML)-Enabled Medical Devices". In: *FDA* (Wed, 09/22/2021 - 12:25). (Visited on 09/15/2022).

[22] Geeta Joshi et al. *FDA Approved Artificial Intelligence and Machine Learning (AI/ML)-Enabled Medical Devices: An Updated 2022 Landscape*. Jan. 2023. DOI: 10.1101/2022.12.07.22283216. (Visited on 06/01/2023).

[23] Curtis P. Langlotz. "Will Artificial Intelligence Replace Radiologists?" In: *Radiology: Artificial Intelligence* 1.3 (May 2019), e190058. DOI: 10.1148/ryai.2019190058. (Visited on 06/01/2023).

[24] Andrew B. Rosenkrantz, Danny R. Hughes, and Richard Duszak Jr. "The US Radiologist Workforce: An Analysis of Temporal and Geographic Variation by Using Large National Datasets". In: *Radiology* 279.1 (2016), pp. 175–184. DOI: 10.1148/radiol.2015150921.

[25] Andrew B. Rosenkrantz et al. "A County-Level Analysis of the US Radiologist Workforce: Physician Supply and Subspecialty Characteristics". In: *Journal of the American College of Radiology* 15.4 (2018), pp. 601–606.

[26] Abi Rimmer. "Radiologist Shortage Leaves Patient Care at Risk, Warns Royal College". In: *BMJ: British Medical Journal (Online)* 359 (2017). DOI: 10.1136/bmj.j4683.

[27] Sarah Bastawrous and Benjamin Carney. "Improving Patient Safety: Avoiding Unread Imaging Exams in the National VA Enterprise Electronic Health Record". In: *Journal of digital imaging* 30.3 (2017), pp. 309–313. DOI: 10.1007/s10278-016-9937-2.

[28] David A. Rosman et al. "Imaging in the Land of 1000 Hills: Rwanda Radiology Country Report". In: *Journal of Global Radiology* 1.1 (2015), p. 5. DOI: 10.7191/jgr.2015.1004.

[29] Farah S. Ali et al. "Diagnostic Radiology in Liberia: A Country Report". In: *Journal of Global Radiology* 1.2 (2015), p. 6.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet Classification with Deep Convolutional Neural Networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[31] Luca Saba et al. "The Present and Future of Deep Learning in Radiology". In: *European Journal of Radiology* 114 (May 2019), pp. 14–24. ISSN: 0720-048X, 1872-7727. DOI: 10.1016/j.ejrad.2019.02.038. (Visited on 12/18/2020).

[32] Yann LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[33] Seymour A. Papert. "The Summer Vision Project". In: (1966).

[34] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer Nature, 2022.

[35] Md Zahangir Alom et al. "A State-of-the-Art Survey on Deep Learning Theory and Architectures". In: *Electronics* 8.3 (Mar. 2019), p. 292. ISSN: 2079-9292.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

[37] Simon J.D. Prince. *Understanding Deep Learning*. MIT Press, 2023.

[38] Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. MIT press, 2022.

[39] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[40] Christopher M. Bishop and Nasser M. Nasrabadi. *Pattern Recognition and Machine Learning*. Vol. 4. Springer, 2006.

[41] Yury Gorishniy et al. "Revisiting Deep Learning Models for Tabular Data". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18932–18943.

[42] Rishi Bommasani et al. *On the Opportunities and Risks of Foundation Models*. Comment: Authored by the Center for Research on Foundation Models (CRFM) at the Stanford Institute for Human-Centered Artificial Intelligence (HAI). Report page with citation guidelines: https://crfm.stanford.edu/report.html. July 2022. DOI: 10.48550/arXiv.2108.07258. arXiv: 2108.07258 [cs]. (Visited on 05/15/2023).

[43] Xavier Glorot and Yoshua Bengio. "Understanding the Difficulty of Training Deep Feedforward Neural Networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256.

[44] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile: IEEE, Dec. 2015, pp. 1026–1034. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.123. (Visited on 04/11/2023).

[45] Marvin Minsky and Seymour A. Papert. *Perceptrons, Reissue of the 1988 Expanded Edition with a New Foreword by Léon Bottou: An Introduction to Computational Geometry*. MIT press, 2017.

[46] Chuan Guo et al. "On Calibration of Modern Neural Networks". In: *International Conference on Machine Learning*. Comment: ICML 2017. PMLR, July 2017, pp. 1321–1330. arXiv: 1706.04599. (Visited on 07/02/2021).

[47] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 807–814.

[48] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[49] Dan Hendrycks and Kevin Gimpel. "Gaussian Error Linear Units (Gelus)". In: *arXiv preprint arXiv:1606.08415* (2016). arXiv: 1606.08415.

[50] Augustin Cauchy. "Méthode Générale Pour La Résolution Des Systemes d'équations Simultanées". In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538.

[51] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors". In: *nature* 323.6088 (1986), pp. 533–536.

[52] Yurii Evgen'evich Nesterov. "A Method of Solving a Convex Programming Problem with Convergence Rate O\backslashbigl(K$\hat{2}$\backslashbigr)". In: *Doklady Akademii Nauk*. Vol. 269. Russian Academy of Sciences, 1983, pp. 543–547.

[53] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR (Poster)*. Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 2015. arXiv: 1412.6980.

[54] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. Comment: Published as a conference paper at ICLR 2019. Jan. 2019. DOI: 10.48550/arXiv.1711.05101. arXiv: 1711.05101 [cs, math]. (Visited on 05/18/2023).

[55] Ilya Tolstikhin et al. "MLP-Mixer: An All-MLP Architecture for Vision". In: *arXiv:2105.01601 [cs]* (May 2021). arXiv: 2105.01601 [cs]. (Visited on 05/06/2021).

[56] Yann LeCun. "Generalization and Network Design Strategies". In: *Connectionism in perspective* 19.143-155 (1989), p. 18.

[57] Chelsea Voss et al. "Visualizing Weights". In: *Distill* 6.2 (Feb. 2021), e00024.007. ISSN: 2476-0757. DOI: 10.23915/distill.00024.007. (Visited on 06/02/2023).

[58] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[59] Gao Huang et al. "Densely Connected Convolutional Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Comment: CVPR 2017. 2017, pp. 4700–4708. arXiv: 1608.06993.

[60] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *International Conference on Machine Learning*. PMLR, 2015, pp. 448–456.

[61] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *arXiv:1905.11946 [cs, stat]* (Sept. 2020). Comment: ICML 2019. arXiv: 1905.11946 [cs, stat]. (Visited on 02/25/2021).

[62] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.

[63] Alec Radford et al. "Improving Language Understanding by Generative Pre-Training". In: *OpenAI blog* (2018).

[64] Alec Radford et al. "Language Models Are Unsupervised Multitask Learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[65] Tom Brown et al. "Language Models Are Few-Shot Learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[66] OpenAI. *GPT-4 Technical Report*. Comment: 100 pages. Mar. 2023. DOI: 10.48550/arXiv.2303.08774. arXiv: 2303.08774 [cs]. (Visited on 05/19/2023).

[67] Alexey Dosovitskiy et al. "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *arXiv:2010.11929 [cs]* (Oct. 2020). Comment: Fine-tuning code and pre-trained models are available at https://github.com/google-research/vision_transformer. DOI: 10.48550/arXiv.2010.11929. arXiv: 2010.11929 [cs]. (Visited on 02/02/2021).

[68] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. July 2016. DOI: 10.48550/arXiv.1607.06450. arXiv: 1607.06450 [cs, stat]. (Visited on 05/19/2023).

[69] Maithra Raghu et al. "Do Vision Transformers Act like Convolutional Neural Networks?" In: 2021. (Visited on 10/14/2022).

[70] Connor Shorten and Taghi M. Khoshgoftaar. "A Survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (Dec. 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. (Visited on 05/21/2023).

[71] Connor Shorten, Taghi M. Khoshgoftaar, and Borko Furht. "Text Data Augmentation for Deep Learning". In: *Journal of Big Data* 8.1 (July 2021), p. 101. ISSN: 2196-1115. DOI: 10.1186/s40537-021-00492-0. (Visited on 05/21/2023).

[72] Jia Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[73] Alexey Dosovitskiy et al. "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *arXiv:2010.11929 [cs]* (June 2021). Comment: Fine-tuning code and pre-trained models are available at https://github.com/google-research/vision_transformer. ICLR camera-ready version with 2 small modifications: 1) Added a discussion of CLS vs GAP classifier in the appendix, 2) Fixed an error in exaFLOPs computation in Figure 5 and Table 6 (relative performance of models is basically not affected). DOI: 10.48550/arXiv.2010.11929. arXiv: 2010.11929 [cs]. (Visited on 09/21/2021).

[74] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *arXiv preprint arXiv:180106146* (May 2018). Comment: ACL 2018, fixed denominator in Equation 3, line 3. DOI: 10.48550/arXiv.1801.06146. arXiv: 1801.06146 [cs, stat]. (Visited on 03/22/2023).

[75] Junji Shiraishi et al. "Development of a Digital Image Database for Chest Radiographs with and without a Lung Nodule: Receiver Operating Characteristic Analysis of Radiologists' Detection of Pulmonary Nodules". In: *American Journal of Roentgenology* 174.1 (2000), pp. 71–74.

[76] PLCO Project Team et al. "The Prostate, Lung, Colorectal and Ovarian (PLCO) Cancer Screening Trial of the National Cancer Institute: History, Organization, and Status". In: *Controlled clinical trials* 21.6 (2000), 251S–272S.

[77] Dina Demner-Fushman et al. "Preparing a Collection of Radiology Examinations for Distribution and Retrieval". In: *Journal of the American Medical Informatics Association* 23.2 (2016). Open-i data set, pp. 304–310.

[78] Xiaosong Wang et al. "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Comment: CVPR 2017 spotlight;V1: CVPR submission+supplementary; V2: Statistics and benchmark results on published ChestX-ray14 dataset are updated in Appendix B V3: Minor correction V4: new data download link upated: https://nihcc.app.box.com/v/ChestXray-NIHCC V5: Update benchmark results on the published data split in the appendix. July 2017, pp. 3462–3471. DOI: 10.1109/CVPR.2017.369. arXiv: 1705.02315. (Visited on 11/26/2020).

[79] Luke Oakden-Rayner. "Exploring Large Scale Public Medical Image Datasets". In: *arXiv preprint arXiv:1907.12720* (July 2019). Comment: 9 pages, 5 tables. DOI: 10.48550/arXiv.1907.12720. arXiv: 1907.12720 [cs, eess]. (Visited on 09/12/2022).

[80] Jeremy Irvin et al. "Chexpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 590–597.

[81] Alistair E. W. Johnson et al. "MIMIC-CXR-JPG, a Large Publicly Available Database of Labeled Chest Radiographs". In: *arXiv:1901.07042 [cs, eess]* (Nov. 2019). arXiv: 1901.07042 [cs, eess]. (Visited on 12/14/2020).

[82] Ross W Filice et al. "Crowdsourcing Pneumothorax Annotations Using Machine Learning Annotations on the NIH Chest X-ray Dataset". In: *Journal of digital imaging* 33.2 (2020), pp. 490–496. DOI: 10.1007/s10278-019-00299-9.

[83] Aurelia Bustos et al. "Padchest: A Large Chest x-Ray Image Dataset with Multi-Label Annotated Reports". In: *Medical image analysis* 66 (2020), p. 101797. arXiv: 1901.07441.

[84] Olivier Bodenreider. "The Unified Medical Language System (UMLS): Integrating Biomedical Terminology". In: *Nucleic acids research* 32.suppl_1 (2004), pp. D267–D270.

[85] *VinBigData Chest Xray Abnormalities Detection Kaggle*. Dec. 2020.

[86] Sijing Feng et al. "Curation of the CANDID-PTX (Chest x Ray Anonymised New Zealand Dataset in Dunedin–Pneumothorax) Dataset with Free-Text Reports". In: *Radiology: Artificial Intelligence* (Oct. 2021), e210136. DOI: 10.1148/ryai.2021210136. (Visited on 10/28/2021).

[87] Eduardo P. Reis et al. "BRAX, Brazilian Labeled Chest x-Ray Dataset". In: *Scientific Data* 9.1 (Aug. 2022), p. 487. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01608-8. (Visited on 09/12/2022).

[88] Erdi Çallı et al. "Deep Learning for Chest X-ray Analysis: A Survey". In: *Medical Image Analysis* 72 (2021), p. 102125.

[89] Pranav Rajpurkar et al. "Deep Learning for Chest Radiograph Diagnosis: A Retrospective Comparison of the CheXNeXt Algorithm to Practicing Radiologists". In: *PLOS Medicine* 15.11 (Nov. 2018), e1002686. ISSN: 1549-1676. DOI: 10.1371/journal.pmed.1002686. (Visited on 01/18/2021).

[90] Alexander Ke et al. "CheXtransfer: Performance and Parameter Efficiency of ImageNet Models for Chest X-Ray Interpretation". In: *arXiv:2101.06871 [cs]* (Jan. 2021). arXiv: 2101.06871 [cs]. (Visited on 02/15/2021).

[91] Ivo-Matteo Baltruschat. "Deep learning for automatic lung disease analysis in chest x-rays". PhD thesis. TUHH Universitätsbibliothek, 2021. DOI: 10.15480/882.3511. (Visited on 06/02/2021).

[92] Ken C. L. Wong et al. "Identifying Disease-Free Chest X-ray Images with Deep Transfer Learning". In: *Medical Imaging 2019: Computer-Aided Diagnosis* (Mar. 2019). Comment: SPIE Medical Imaging, 2019 (oral presentation), p. 24. DOI: 10.1117/12.2513164. arXiv: 1904.01654. (Visited on 02/15/2021).

[93] Hieu H. Pham et al. "Interpreting Chest X-rays via CNNs That Exploit Hierarchical Disease Dependencies and Uncertainty Labels". In: *Neurocomputing* 437 (2021), pp. 186–194.

[94] Alessandro Wollek et al. "Attention-Based Saliency Maps Improve Interpretability of Pneumothorax Classification". In: *Radiology: Artificial Intelligence* (Mar. 2023), e220187. DOI: 10.1148/ryai.220187. (Visited on 03/06/2023).

[95] Junfei Xiao et al. "Delving into Masked Autoencoders for Multi-Label Thorax Disease Classification". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 3588–3600.

[96] Zhe Li et al. "Thoracic Disease Identification and Localization with Limited Supervision". In: *arXiv:1711.06373 [cs, stat]* (June 2018). Comment: Conference on Computer Vision and Pattern Recognition 2018 (CVPR 2018). V1: CVPR submission; V2: +supplementary; V3: CVPR camera-ready; V4: correction, update reference baseline results according to their latest post; V5: minor correction; V6: Identification results using NIH data splits and various image models. arXiv: 1711.06373 [cs, stat]. (Visited on 02/15/2021).

[97] Soham Uday Gadgil et al. "CheXseg: Combining Expert Annotations with DNN-generated Saliency Maps for X-ray Segmentation". In: *Proceedings of the Fourth Conference on Medical Imaging with Deep Learning*. PMLR, Aug. 2021, pp. 190–204. (Visited on 06/04/2023).

[98] *EUR-Lex - 32016R0679 - EN - EUR-Lex*. https://eur-lex.europa.eu/eli/reg/2016/679/oj. (Visited on 06/04/2023).

[99] Alistair EW Johnson et al. "MIMIC-CXR, a de-Identified Publicly Available Database of Chest Radiographs with Free-Text Reports". In: *Scientific Data* 6 (2019).

[100] Stefan Schweter and Alan Akbik. "Flert: Document-level Features for Named Entity Recognition". In: *arXiv preprint arXiv:2011.06993* (2020). DOI: 10.48550/arXiv.2011.06993. arXiv: 2011.06993.

[101] Alessandro Wollek et al. "German CheXpert Chest X-ray Radiology Report Labeler". In: *arXiv preprint arXiv:2306.02777* (June 2023). DOI: 10.48550/arXiv.2306.02777. arXiv: 2306.02777 [cs]. (Visited on 06/06/2023).

[102] Bukunmi Idowu and Tolulope Okedere. "Diagnostic Radiology in Nigeria: A Country Report". In: *Journal of Global Radiology* 6.1 (June 2020). ISSN: 2372-8418. DOI: 10.7191/jgr.2020.1072.

[103] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015). Comment: 43 pages, 16 figures. v3 includes additional comparisons with PASCAL VOC (per-category comparisons in Table 3, distribution of localization difficulty in Fig 16), a list of queries used for obtaining object detection images (Appendix C), and some additional references, pp. 211–252. (Visited on 02/07/2022).

[104] Alexey Dosovitskiy et al. "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2020. DOI: 10.48550/arXiv.2010.11929.

[105] Thao Nguyen et al. "Learning to Diagnose Common Thorax Diseases on Chest Radiographs from Radiology Reports in Vietnamese". In: *PLOS ONE* 17.10 (Oct. 2022). Ed. by Tarik A. Rashid. [TLDR] This work proposes a data collecting and annotation pipeline that extracts information from Vietnamese radiology reports to provide accurate labels for chest X-ray (CXR) images and employs a variety of loss functions to overcome the curse of imbalanced multi-label datasets., e0276545. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0276545. (Visited on 03/30/2023).

[106] S. Nowak et al. "Transformer-Based Structuring of Free-Text Radiology Report Databases". In: *European Radiology* (Mar. 2023). ISSN: 1432-1084. DOI: 10.1007/s00330-023-09526-y. (Visited on 05/02/2023).

[107] Johannes Rueckel et al. "Impact of Confounding Thoracic Tubes and Pleural Dehiscence Extent on Artificial Intelligence Pneumothorax Detection in Chest Radiographs". In: *Investigative Radiology* 55.12 (Dec. 2020), pp. 792–798. ISSN: 1536-0210, 0020-9996. DOI: 10.1097/RLI.0000000000000707. (Visited on 03/09/2021).

[108] Viviana Cotik et al. "Negation Detection in Clinical Reports Written in German". In: *Proceedings of the Fifth Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM2016)*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 115–124. (Visited on 01/09/2023).

[109] David M. Hansell et al. "Fleischner Society: Glossary of Terms for Thoracic Imaging". In: *Radiology* 246.3 (2008), pp. 697–722.

[110] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv:1301.3781 [cs]* (Sept. 2013). DOI: 10.48550/arXiv.1301.3781. arXiv: 1301.3781 [cs]. (Visited on 04/08/2020).

[111] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5 (Dec. 2017), pp. 135–146. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00051. (Visited on 04/08/2020).

[112] Jacob Devlin et al. "Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018). arXiv: 1810.04805.

[113] Akshay Smit et al. "CheXbert: Combining Automatic Labelers and Expert Annotations for Accurate Radiology Report Labeling Using BERT". In: *arXiv preprint arXiv:2004.09167* (2020). DOI: 10.48550/arXiv.2004.09167. arXiv: 2004.09167.

[114] *German BERT — State of the Art Language Model for German NLP*. https://www.deepset.ai/german-bert. (Visited on 06/06/2023).

[115] Tom Kimpe and Tom Tuytschaever. "Increasing the Number of Gray Shades in Medical Display Systems—How Much Is Enough?" In: *Journal of Digital Imaging* 20.4 (Dec. 2007), pp. 422–432. ISSN: 0897-1889. DOI: 10.1007/s10278-006-1052-3. (Visited on 05/23/2023).

[116] Ella A. Kazerooni and Barry H. Gross. *Cardiopulmonary Imaging*. Lippincott Williams & Wilkins, 2004. ISBN: 978-0-7817-3655-8.

[117] Manohar Karki et al. "CT Window Trainable Neural Network for Improving Intracranial Hemorrhage Detection by Combining Multiple Settings". In: *Artificial Intelligence in Medicine* 106 (June 2020), p. 101850. ISSN: 0933-3657. DOI: 10.1016/j.artmed.2020.101850. (Visited on 04/20/2023).

[118] Yuankai Huo et al. "Stochastic Tissue Window Normalization of Deep Learning on Computed Tomography". In: *Journal of Medical Imaging* 6.4 (Oct. 2019), p. 044005. ISSN: 2329-4302. DOI: 10.1117/1.JMI.6.4.044005. (Visited on 04/20/2023).

[119] Hyunkwang Lee, Myeongchan Kim, and Synho Do. *Practical Window Setting Optimization for Medical Image Deep Learning*. Comment: Machine Learning for Health (ML4H) Workshop at NeurIPS 2018 arXiv:cs/0101200. Dec. 2018. DOI: 10.48550/arXiv.1812.00572. arXiv: 1812.00572 [cs]. (Visited on 04/20/2023).

[120] Jangho Kwon and Kihwan Choi. "Trainable Multi-contrast Windowing for Liver CT Segmentation". In: *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*. Feb. 2020, pp. 169–172. DOI: 10.1109/BigComp48618.2020.00-80.

[121] Hyunkwang Lee, Myeongchan Kim, and Synho Do. "Practical Window Setting Optimization for Medical Image Deep Learning". In: *arXiv:1812.00572 [cs]* (Dec. 2018). Comment: Machine Learning for Health (ML4H) Workshop at NeurIPS 2018 arXiv:cs/0101200. arXiv: 1812.00572 [cs]. (Visited on 11/12/2021).

[122] Carl F. Sabottke and Bradley M. Spieler. "The Effect of Image Resolution on Deep Learning in Radiography". In: *Radiology: Artificial Intelligence* 2.1 (Jan. 2020), e190015. DOI: 10.1148/ryai.2019190015. (Visited on 05/26/2023).

[123] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*. 2018.

[124] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 618–626. (Visited on 01/27/2021).

[125] Rafael Padilla, Sergio L. Netto, and Eduardo AB da Silva. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2020, pp. 237–242.

[126] Cynthia Rudin. "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead". In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215. DOI: 10.1038/s42256-019-0048-x.

[127] Mauricio Reyes et al. "On the Interpretability of Artificial Intelligence in Radiology: Challenges and Opportunities". In: *Radiology: artificial intelligence* 2.3 (2020), e190043. DOI: 10.1148/ryai.2020190043.

[128] Finale Doshi-Velez and Been Kim. "Towards a Rigorous Science of Interpretable Machine Learning". In: *arXiv preprint arXiv:1702.08608* (2017). DOI: 10.48550/arXiv.1702.08608. arXiv: 1702.08608.

[129] Ramón Alvarado. "Should We Replace Radiologists with Deep Learning? Pigeons, Error and Trust in Medical AI". In: *Bioethics* 36.2 (Feb. 2022), pp. 121–133. ISSN: 1467-8519. DOI: 10.1111/bioe.12959.

[130] Mantaj S. Brar et al. "Occult Pneumothoraces Truly Occult or Simply Missed: Redux". In: *The Journal of Trauma* 69.6 (Dec. 2010), pp. 1335–1337. ISSN: 1529-8809. DOI: 10.1097/TA.0b013e3181f6f525.

[131] Hugo Touvron et al. "Training Data-Efficient Image Transformers & Distillation through Attention". In: *arXiv:2012.12877 [cs]* (Jan. 2021). arXiv: 2012.12877 [cs]. (Visited on 02/02/2021).

[132] Hila Chefer, Shir Gur, and Lior Wolf. "Generic Attention-Model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 397–406. (Visited on 10/07/2022).

[133] Nishanth Arun et al. "Assessing the Trustworthiness of Saliency Maps for Localizing Abnormalities in Medical Imaging". In: *Radiology: Artificial Intelligence* 3.6 (2021).

[134] Weina Jin et al. "Artificial Intelligence in Glioma Imaging: Challenges and Advances". In: *Journal of neural engineering* 17.2 (2020), p. 021002. DOI: 10.1088/1741-2552/ab8131.

[135] Weina Jin, Xiaoxiao Li, and Ghassan Hamarneh. "One Map Does Not Fit All: Evaluating Saliency Map Explanation on Multi-Modal Medical Images". In: *arXiv preprint arXiv:2107.05047* (2021). DOI: 10.48550/arXiv.2107.05047. arXiv: 2107.05047.

[136] Fabian Eitel et al. "Uncovering Convolutional Neural Network Decisions for Diagnosing Multiple Sclerosis on Conventional MRI Using Layer-Wise Relevance Propagation". In: *NeuroImage: Clinical* 24 (2019), p. 102003. DOI: 10.1016/j.nicl.2019.102003.

[137] Murat Seçkin Ayhan et al. "Clinical Validation of Saliency Maps for Understanding Deep Neural Networks in Ophthalmology". In: *Medical Image Analysis* (2022), p. 102364. DOI: 10.1016/j.media.2022.102364.

[138] Jing Zhang et al. "Explainability for Regression CNN in Fetal Head Circumference Estimation from Ultrasound Images". In: *Interpretable and Annotation-Efficient Learning for Medical Image Computing*. Springer, 2020, pp. 73–82.

[139] Kyle Young et al. "Deep Neural Network or Dermatologist?" In: *Interpretability of Machine Intelligence in Medical Image Computing and Multimodal Learning for Clinical Decision Support*. Springer, 2019, pp. 48–55.

[140] Ha Q Nguyen et al. "VinDr-CXR: An Open Dataset of Chest X-rays with Radiologist's Annotations". In: *arXiv:2012.15029 [eess]* (Dec. 2020), p. 10. arXiv: 2012.15029 [eess].

[141] Patrick Schober, Christa Boer, and Lothar A Schwarte. "Correlation Coefficients: Appropriate Use and Interpretation". In: *Anesthesia & Analgesia* 126.5 (2018), pp. 1763–1768. DOI: 10.1213/ANE.0000000000002864.

[142] Yang Zhang et al. "Fine-Grained Neural Network Explanation by Identifying Input Features with Predictive Information". In: *Advances in Neural Information Processing Systems* 34 (2021).

[143] Bianca Zadrozny and Charles Elkan. "Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers". In: *Icml*. Vol. 1. Citeseer, 2001, pp. 609–616.

[144] Xu Sun and Weichao Xu. "Fast Implementation of DeLong's Algorithm for Comparing the Areas Under Correlated Receiver Operating Characteristic Curves". In: *IEEE Signal Processing Letters* 21.11 (Nov. 2014), pp. 1389–1393. ISSN: 1558-2361. DOI: 10.1109/LSP.2014.2337313.

[145] Elizabeth R. DeLong, David M. DeLong, and Daniel L. Clarke-Pearson. "Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach". In: *Biometrics* (1988), pp. 837–845.

[146] Julius Adebayo et al. "Sanity Checks for Saliency Maps". In: *Advances in neural information processing systems* 31 (2018).

[147] Cindy S Lee et al. "Cognitive and System Factors Contributing to Diagnostic Errors in Radiology". In: *American Journal of Roentgenology* 201.3 (2013), pp. 611–617. DOI: 10.2214/AJR.12.10375.

[148] Alessandro Wollek et al. *A Knee Cannot Have Lung Disease: Out-of-Distribution Detection with in-Distribution Voting Using the Medical Example of Chest X-ray Classification*. Comment: Code available at https://gitlab.lrz.de/IP/a-knee-cannot-have-lung-disease. Aug. 2022. DOI: 10.48550/arXiv.2208.01077. arXiv: 2208.01077 [cs, eess]. (Visited on 08/03/2022).

[149] Varun Gulshan et al. "Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs". In: *JAMA* 316.22 (Dec. 2016), pp. 2402–2410. ISSN: 0098-7484. DOI: 10.1001/jama.2016.17216. (Visited on 02/10/2022).

[150] Andre Esteva et al. "Dermatologist-Level Classification of Skin Cancer with Deep Neural Networks". In: *Nature* 542.7639 (Feb. 2017), pp. 115–118. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature21056. (Visited on 03/04/2021).

[151] Anna Majkowska et al. "Chest Radiograph Interpretation with Deep Learning Models: Assessment with Radiologist-adjudicated Reference Standards and Population-adjusted Evaluation". In: *Radiology* 294.2 (Dec. 2019), pp. 421–431. ISSN: 0033-8419. DOI: 10.1148/radiol.2019191293. (Visited on 03/09/2021).

[152] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images". In: *CVPR*. 2015, pp. 427–436. (Visited on 02/17/2022).

[153] Eric Nalisnick et al. "Do Deep Generative Models Know What They Don't Know?" In: *arXiv preprint arXiv:1810.09136* (2018). arXiv: 1810.09136.

[154] Dan Hendrycks et al. "Natural Adversarial Examples". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15262–15271.

[155] Paul Robinette, Ayanna M. Howard, and Alan R. Wagner. "Effect of Robot Performance on Human–Robot Trust in Time-Critical Situations". In: *IEEE Transactions on Human-Machine Systems* 47.4 (2017), pp. 425–436.

[156] Effy Vayena, Alessandro Blasimme, and I. Glenn Cohen. "Machine Learning in Medicine: Addressing Ethical Challenges". In: *PLoS medicine* 15.11 (2018), e1002689.

[157] Oded Nov et al. "The Transformation of Patient-Clinician Relationships with Ai-Based Medical Advice". In: *Communications of the ACM* 64.3 (2021), pp. 46–48.

[158] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *ICLR*. 2017.

[159] Dan Hendrycks et al. "Scaling Out-of-Distribution Detection for Real-World Settings". In: *arXiv:1911.11132 [cs]* (Dec. 2020). Comment: StreetHazards dataset and code are available at https://github.com/hendrycks/anomaly-seg
Comment: StreetHazards dataset and code are available at https://github.com/hendrycks/anomaly-seg. arXiv: 1911.11132 [cs]. (Visited on 12/16/2021).

[160] Haoran Wang et al. "Can Multi-Label Classification Networks Know What They Don't Know?" In: *Advances in Neural Information Processing Systems* 34 (2021).

[161] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. "Deep Anomaly Detection with Outlier Exposure". In: *ICLR* (2019).

[162] Dan Hendrycks et al. "Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 15663–15674.

[163] Erdi Çallı et al. "FRODO: An in-Depth Analysis of a System to Reject Outlier Samples from a Trained Neural Network". In: *IEEE Transactions on Medical Imaging* (2022), pp. 1–1. ISSN: 1558-254X. DOI: 10.1109/TMI.2022.3221898.

[164] Kimin Lee et al. "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks". In: *Advances in neural information processing systems* 31 (2018).

[165] Alex Krizhevsky and Geoffrey Hinton. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto, 2009.

[166] Yuval Netzer et al. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*. 2011.

[167] Gao Huang et al. "Densely Connected Convolutional Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4700–4708.

[168] Petra Bevandić et al. "Simultaneous Semantic Segmentation and Outlier Detection in Presence of Domain Shift". In: *German Conference on Pattern Recognition*. Springer, 2019, pp. 33–47.

[169] Antonio Torralba and Alexei A. Efros. "Unbiased Look at Dataset Bias". In: *CVPR 2011*. June 2011, pp. 1521–1528. DOI: 10.1109/CVPR.2011.5995347.

[170] Thomas Deserno and B. Ott. *15.363 IRMA Bilder in 193 Kategorien für ImageCLEFmed 2009*.
Vorbemerkung
Auf dieser Web-Seite werden Daten der Allgemeinheit zur Verfügung gestellt, die in verschiedenen wissenschaftlichen Untersuchungen von mir verwendet wurden. Ziel hierbei ist es, den Vergleich verschiedener Methodiken zu ermöglichen, in dem andere Forschergruppen mit gleichen Daten arbeiten können. Da die Zusammenstellung bzw. Aufnahme der Daten teilweise sehr aufwändig war, werden diese ausschließlich einer nicht-kommerziellen Nutzung zugänglich gemacht, wenn in allen Publikationen auf die Quelle der Daten hingewiesen wird.
Beschreibung
Dieser Datensatz enthält anonymisierte Röntgenbilder der Klinik für Radiologische Diagnostik der RWTH Aachen. Die Bilder wurden willkürlich aus der Routine entnommen und reflektieren unterschiedliche Altersklassen, Geschlechter, Aufnahmepositionen und Pathologien und unterscheiden sich daher auch in der Bildqualität. Alle Aufnahmen wurden unter Beibehaltung des Seitenverhältnisses auf das maximale Format von 512 x 512 Pixel verkleinert. Alle Bilder wurden gemäß des IRMA Codes von erfahrenen Radiologen klassifiziert und dann in 193 zu unterscheidende Gruppen eingeteilt. Für 12.677 Bilder ist diese Unterscheidung bekannt, für weitere 1.733 nicht. Diese Bilder dienen als Test für den Wettbewerb ImageCLEFmed 2009.

- Training: 12.677 Trainingsbilder mit bekannten Kategorien.
- Kategorie-Verteilung: Zuordnung aller Trainingsdaten.
- Training: 12.677 Trainingsbilder mit bekannten Kategorien.
- Test: 1.733 Bilder ohne Kategorisierung.

Remark
On this page, you can access data that has been used in several scientific studies. In order to make the results comparable, these datasets are available for other non-commercial evaluations.
Description
This collection compiles anonymous radiographs, which have been arbitrarly selected from routine at the Department of Diagnostic Radiology, Aachen University of Technology (RWTH), Aachen, Germany. The imagery represents different ages, genders, view positions and pathologies. Therefore, image quality varies significantly. All images were downscaled to fit into a 512 x 512 bounding box maintaining the original aspect ratio. All images were classified according to the IRMA code. Based on this code, 193 categories were defined. For 12,677 images, these categories are provided. The remaining 1,733 images without code are used as test data for the ImageCLEFmed 2009 competition.

- Training data: 12,677 radiographs with known categories
- Class distribution: Distribution of classes in the training data.
- Test data: 1,733 radiographs without classification.

. 2009. DOI: 10.18154/RWTH-2016-06143.

[171] Pranav Rajpurkar et al. "MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs". In: *Medical Imaging with Deep Learning*. Comment: 1st Conference on Medical Imaging with Deep Learning (MIDL 2018). Amsterdam, 2018. arXiv: 1712.06957. (Visited on 02/15/2022).

[172] Safwan S. Halabi et al. "The RSNA Pediatric Bone Age Machine Learning Challenge". In: *Radiology* 290.2 (Feb. 2019), pp. 498–503. ISSN: 0033-8419. DOI: 10.1148/radiol.2018180736. (Visited on 02/07/2022).

[173] Shiyu Liang, Yixuan Li, and R. Srikant. "Enhancing the Reliability of Out-of-Distribution Image Detection in Neural Networks". In: *6th International Conference on Learning Representations, ICLR 2018*. Comment: ICLR 2018. 2018. arXiv: 1706.02690.