RADBOUD UNIVERSITY

MASTER THESIS

Pneumothorax Detection On Chest Radiographs: A Comparative Analysis Of Public Datasets, Deep Learning Architectures, And Domain Adaptation Via Iterative Self-Training.

Author:
R.M.W. KLUGE
(S4388267)

Supervisors:
E. SOGANCIOGLU
Prof. B. VAN GINNEKEN
Prof. E. MARCHIORI

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in

Computer Science: Data Science

May, 2020

Radboud University

Abstract

Faculty of Science
Computer Science: Data Science

Master of Science

Pneumothorax Detection On Chest Radiographs: A Comparative Analysis Of Public Datasets, Deep Learning Architectures, And Domain Adaptation Via Iterative Self-Training.

by R.M.W. KLUGE

In part I of this thesis, we show an open & transparent deep learning approach to pneumothorax detection. We combine various publicly available datasets that are accessible to anyone, giving us a verifiable multi-center approach. Even though we only train on public datasets, we achieve equal performance (AUC of 0.94) compared to state-of-the-art research that uses additional private datasets (Majkowska et al., 2020). Because all the datasets used in this research are public, we publish our model weights & algorithm[1] and hope that future research can benefit from our work.

Further, in part II we propose an unsupervised domain adaptation method – *iterative self-training* – that improves performance on an unseen dataset without the need for additional labelling (i.e. different hospital data). These results show an increase in performance (AUC 0.82 $\rightarrow$ 0.89) for pneumothorax detection on public datasets *CheXpert* $\rightarrow$ *SIIM*. This method was submitted to *MIDL 2020* as a short paper (Appendix C).

Finally, in part III we evaluate the complete pipeline including our *iterative self-training* method on a local private dataset of 28.207 images (*RadboudCXR*), and verify that *iterative self-training* successfully adapts to an unseen local dataset (AUC 0.87 $\rightarrow$ 0.92). We integrate the final algorithm using the *grand-challenge* platform and is publicly accessible for testing:
https://grand-challenge.org/algorithms/cxr-pneumothorax-detection/

[1] This algorithm is part of *OpenCXR*, and will be released by DIAG accordingly.

Contents

Part I

Pneumothorax detection

Chapter 1

Introduction

1.1 Pneumothorax

A pneumothorax, also known as a collapsed lung, is defined as an acute pathology where there exists air between the lung and the chest wall (chest cavity). Usually, this region is a vacuum, making the lung as large and inflated as possible. When the vacuum is lost, the lung loses its inflation and causes the lung to 'collapse'. Due to the collapse, the lung is unable to expand well during inhalation and, therefore, unable to extract oxygen from the air. Thus, lung collapse results in the inability to breathe.

1.1.1 Types

The main two general causes for a pneumothorax are: traumatic pneumothorax and a spontaneous pneumothorax. A traumatic pneumothorax can occur due to a traumatic (physical) injury to the chest. Possible causes can be stab wounds, car crashes, but a pneumothorax can also occur during surgery. After chest injuries, a traumatic pneumothorax is the second most frequent chest pathology, occurring in up to 50% of chest trauma victims (Yarmus and Feller-Kopman, 2012).

The next category is the spontaneous pneumothorax, which occurs spontaneously without direct attributed cause which is subdivided into two classes:

1. primary spontaneous pneumothorax (PSP)
2. secondary spontaneous pneumothorax (SSP)
3. tension pneumothorax

Primary spontaneous pneumothorax (PSP) A pneumothorax is called a primary spontaneous pneumothorax (PSP) when there are no other causes (e.g., underlying lung diseases) leading to a pneumothorax. According to Sahn and Heffner, 2000 and Noppen and De Keukeleire, 2008, a PSP is seen between 7.4-18 / 100.000 times per year for males, and 1.2-6 / 100.000 times per year for females. Smoking increases this risk by 7 times for females and 22 times for males (Cheng et al., 2009).

Signs of a pneumothorax in these patients are sharp chest pains and a higher resting heart rate (tachycardia). A quick diagnosis of the right disease is critical here, as it is essential to rule out other more life-threatening diseases that show similar symptoms, such as a heart attack.

Secondary spontaneous pneumothorax (SSP) A spontaneous pneumothorax is classified as a *secondary spontaneous pneumothorax (SSP)* when other underlying lung

diseases are present. Some underlying lung diseases that can trigger a pneumothorax are: emphysema, COPD, tuberculosis, and cystic fibrosis. SSPs tend to be more life-threatening than PSPs, as underlying lung diseases already affect the vital functions of the patient, and a pneumothorax degrades the intake of oxygen even further. Although prevalence rates are not reported for SSPs, they have the same prevalence rate as a PSP. When a pneumothorax is reported, the exact type of pneumothorax is often not (Hallifax and Rahman, 2015).

Tension pneumothorax The most critical variant of a pneumothorax is the *tension pneumothorax*. This variant requires immediate intervention, as the size of the pneumothorax increases each breath. It can be seen as a one-way pressure valve, where for each breath, the pressure within the chest is increased. Treatment for this type is imminent, and waiting a few hours for a radiology report is unthinkable. When a tension pneumothorax is not timely treated, this can further cause respiratory and hemodynamic decompensation (MacDuff; Arnold, and Harvey, 2010).

1.1.2 Assessment

When a patient comes in with respiratory issues, often the first imaging method acquired is a chest radiograph (x-ray). Chest x-rays project an image of the inner chest (lungs, vessels, bones) by using a small radiation doses, which is relatively cheap to perform. In order to get the full scope of the lungs of a patient, chest x-rays are generally acquired using two different viewing angles: a posterior-anterior (frontal) view and a lateral (side) view. In optimal conditions, the patient is standing upright and is holding its breath in order to increase lung volume. However, patients brought directly in the emergency room are often too ill to stand up. This way, we cannot acquire a posterior-anterior (back-to-front) x-ray, but a anterior-posterior (front-to-back) x-ray is made. Assessment of a pneumothorax of these two types of patients differs a lot. We explain the differences for the patient that is able to stand up (PA view assessment) versus critically ill patients (AP view assessment) below.

PA view assessment When the patient is standing up, the lung tends to 'fall down' due to the nature of gravity. As a result, the air between the lung and chest wall is either stuck in the bottom or travels to the top. When a pneumothorax is present, the lung puts pressure on this loose pocket of air. Due to compression of the air pocket, the edge of the lung (pleura) is thickened, and thus easier to spot. When the pleura is observed and is not connected to the chest wall, the patient suffers from a pneumothorax.

AP view assessment When the patient comes in through the emergency department or is unconscious in the intensive care department, the patient is not responding to commands. They are bedridden, and we cannot ask the patient to hold its breath or to stand upright. For some patients, we are even unable to incline the bed, as their low blood pressure has trouble keeping blood running to their brains.

Detection of a pneumothorax for patients lying down (supine position) is generally harder. The forces of gravity are now a disadvantage, as the lungs get pushed down to the posterior side of the body. Because the heart is placed more at the frontal side of the body, anterior-posterior (AP) view images tend to show less of the lungs, as the heart masks part of the left and right lung.

In addition to the possible findings that can be detected using PA view, a supine view pneumothorax might be harder to detect. Although detection is harder, it provides several other signs related to a pneumothorax (Yarmus and Feller-Kopman, 2012):

1. Air does not collect in the top or bottom part of the lung due to gravity.
2. Air collects in the front (anterior) side, which makes it unable to find a clear lung edge.
3. Air can track further around the chest, presenting a *deep sulcus sign*.

Tension pneumothorax A tension pneumothorax adds an additional indication of a pneumothorax: the mediastinal shift. Here, the tension on one side of the lung causes the heart (mediastinum) to shift to one side. We show an example of a tension pneumothorax with a mediastinal shift in Figure 1.1.

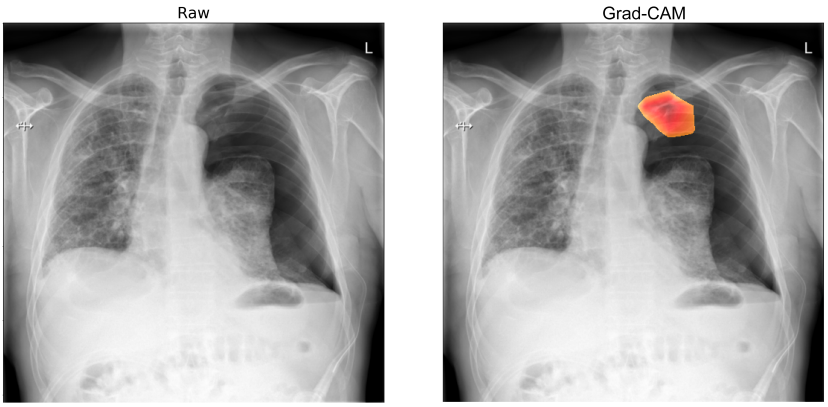


FIGURE 1.1: Tension pneumothorax (left lung) with mediastinal shift to the right. Right image shows the location prediction of our algorithm in red.

Based on only supine AP x-rays, radiologists fail in detecting a pneumothorax 38.8% of the time (Bridges et al., 1993). Raoof et al., 2012 shows that in 15% of the cases, radiologists also need to assess the lateral view x-rays next to the frontal x-rays to be certain of the diagnosis. When a pneumothorax is missed, trauma patients may deteriorate and introduce complications, and require anesthesia or mechanical ventilation, which further decreases the long-term quality of life.

A more precise method to assess the presence of a pneumothorax is by obtaining a CT scan. A CT scan provides a slice-by-slice 3D view of the inside of the chest with a high resolution. The trade-off for acquiring a CT scan is the higher radiation dose, it is more expensive, and the CT itself is large. Because the CT is large, it is not often not feasible for critically ill patients. There is no portable version of the CT, so that means the patient needs to be transported to the device. Transportation is a risky operation, as a critically ill patient is hooked up to additional support devices that keep the patient alive (e.g. mechanical ventilation).

Often, a CT scan is issued when the radiologist is not sure of the diagnosis. Using our algorithm, we can help minimize unnecessary requests for CT scans, and therefore reduce the needed capacity and radiation dose given to patients.

1.1.3 Triage

Some pathologies are more time-sensitive than others, meaning that some illnesses need immediate intervention, whereas with other pathologies treatment can wait. Patient treatment prioritization is also called *triaging*. Triaging is applied at the emergency department, and becomes useful when many patients need to make use of the limited resources. The severity of a patient is determined by looking at abnormalities in the vital signs. As an example, it makes sense to prioritize patients suffering from a heart attack over patients with a broken toe.

Triaging is also applied to the priority list of the radiologist. As a radiologist needs to read many chest radiographs a day, the workload and queue is generally large. If we can determine beforehand which patients have a critical pathology, even before the radiologists assess the image, we can prioritize the chest radiograph above others. It has been shown that applying AI to radiologist workload prioritization reduces the delay for treating critically ill patients from 7.2 hours to 43 minutes (Annarumma et al., 2019). Some pathologies that are marked *critical* are: pneumomediastinum, subcutaneous emphysema, intra-abdominal pathology, but also a pneumothorax. Critically ill patients require immediate attention and could be unnecessarily exposed to the degradation of vital signs when treatment is delayed. As a pneumothorax is considered as a critical disease, our algorithm can help with workload prioritization.

1.2 Chest radiographs

Chest radiographs (Chest X-rays) are the most common film taken in hospitals and contribute to 40% of all the diagnostic images taken worldwide (*Communicating radiation risks in paediatric imaging* 2016). It is often one of the first procedures taken when it comes to lung pathology because it is a relatively inexpensive imaging method compared to CT or MRI scans. The radiation dose of chest x-rays is also significantly less compared to CT scans. One chest x-ray equals three days of exposure to natural radiation. However, a CT scan of the chest is equivalent to 350 times a chest X-ray.

These radiographs can reveal many internal problems: bone fractures, ruption of blood vessels, heart diseases, catheters, cancer nodules, etc. After making a radiograph, a radiologist needs to check for possible pathologies. Only radiologists can form these reports, as interpreting them requires substantial imaging experience. Even after a few months of working on this project, I still am not able to successfully spot the researched pneumothorax pathology myself.

The sole purpose of the radiologist is to interpret medical images (CT, MRI, X-rays, nuclear imaging) and report the findings back to the referring clinicians.

1.2.1 Variance

Not every chest x-ray image is the same. Although there are standardized procedures for taking an x-ray, these are mostly hospital and manufacturer specific. Multiple factors contribute to the variance of chest radiographs, with the most important being:

1. Machine
    - Manufacturer (Philips, Siemens, Toshiba)
    - Type (fixed, portable)

2. Configurable settings
    - Pixel spacing
    - Peak potential voltage
    - Distance patient-machine (Source-to-Image Distance, SID)
    - Angle patient-machine

All these parameters combined cause a difference in image quality and characteristics of chest radiographs. One good example is observed during a day at the radiology department of the Radboudumc. In general, Radboudumc uses two types of machines for their radiographs: fixed x-ray machines and portable x-ray machines.

Fixed x-ray machines tend to have a greater distance towards the patient, with a higher peak potential voltage (125kV) [1]. A greater distance between the machine and the patient results in a more parallel radiation beam, giving clearer and sharper imaging.

However, their portable X-ray imagery is set to a lower peak potential voltage (80kV, depending on obesity levels of the patient), and due to space restrictions, the distance towards the patient is decreased.

The use of the fixed (better quality) x-ray machine is preferred, but not always possible for all patients. Some patients recovering from surgery are still in intensive care. These patients are unconscious and attached to support devices that take over their breathing. Therefore, the portable x-ray machine needs to be brought to the patient. As a result, all critically ill & post-operative patients are imaged using the portable x-ray machine, which introduces a bias. This way, post-operative pneumothoraces are only seen with the portable x-ray machine, giving a higher pneumothorax prevalence to imagery coming from portable x-ray machines.

1.3 Research Question

Radiologists have a busy life, and with multiple imaging modalities to assess each day this work does not become less busy. Also in third-world countries, where x-ray devices are relatively cheap, but radiologists are scarce. Off-loading some of this radiology work increases speed of assessment and eventually increase patient health. Automatic pneumothorax detection can help. A possible use case for automated pneumothorax detection could be to prioritize chest x-rays that show critical abnormalities over other x-rays, so that radiologists takes a look at them first (workload prioritization / triaging). Pan; Agarwal, and Merck, 2019 sees the opportunity to review each image algorithmically and assign a score to this image, before letting the radiologist see this image. Then, using the provided information by the algorithm the efficiency and speed of radiologists finding critical abnormalities will improve.

As developing a solution to pneumothorax detection requires expertise and research costs, most of the existing solutions are commercial. By providing an open source solution to pneumothorax detection, we hope to lay a solid foundation that is available to anyone. In chapter 2, we list an overview of both commercial and open-source efforts to pneumothorax detection. By focusing on pneumothorax detection specifically, we are able to highlight the limitations of current open-source research, and deal with them consecutively. Therefore, our research question is as follows:

[1] According to two laboratory technicians of Radboudumc, 10/02/2020

Can we, based on publicly accessible data, develop a deep learning solution to pneumothorax detection that is able to compete with other related (commercial) pneumothorax detection works?

Chapter 2

Related Work

In this chapter, we briefly summarize related work and compare the experimental validity of this work. Litjens et al., 2017 provides an overview of deep learning implementations in medical imaging. Due to the increase in popularity of deep learning, there is an increase of papers towards computer aided detection of diseases using deep learning. In general, the overall approaches towards pneumothorax detection are similar. Most research uses pre-trained networks and use the same data augmentations. However, there are big differences in terms of model evaluation. In order spot the differences between research, we summarize the papers into a structured overview (Table 2.1).

2.1 Basic CNN approaches

Gooßen et al., 2019 evaluates a comparison of deep learning approaches for pneumothorax detection. A Convolutional Neural Network (CNN) that outputs a binary classification, a Fully Convolutional Network (FCN) delivering a per-pixel segmentation, and a Multiple-Instance Learning (MIL) are considered. Their CNN approach is pre-trained using the *NIH ChestX-ray14* dataset (but trained on a private dataset) and achieves best performance in terms of classification AUC (AUC of 0.96), whereas their FCN and MIL approach provides better localization. Future work could try to merge or cascade this architecture in order to get the best of both worlds. The methods are validated with a private internal dataset, and no external data validation is done.

Park et al., 2019 evaluates a CNN based on a 26-layer YOLO Darknet19 model. The AUC was 0.898. The dataset is constructed from two different hospitals, but no external validation was done. Small pneumothoraces are excluded from the training set.

Baltruschat et al., 2019 shows a ResNet-50 architecture that also takes into account non-image information such as age, gender and view position. This seemed to work better than the image-only networks they have tested. However, we can argue that the non-image features *age* and *gender* should not influence the algorithm's decision depending on the goal of the system. The algorithm should base its finding on image features, and not gain knowledge from a bias like *age* or *gender*. Achieves an AUC of 0.87 on the ChestX-ray14 dataset.

Rajpurkar et al., 2017 claims radiologist-level performance on pneumonia detection. For the pneumothorax task, they achieve an AUC of 0.8887, which is better than other related works (Wang et al., 2017 : 0.806, Yao et al. (2017): 0.841). Their model (*CheXNet*) is a 121-layer Dense Convolutional Network (*DenseNet-121*) (Huang et

al., 2017) trained on the *ChestX-ray14* dataset (98.637 images for training, 6.351 for tuning, 420 balanced labels for testing). They compare the test set performance of 4 radiologists with the performance of the network and concluded that *CheXNet* is *significantly* better, although the difference in performance is very small. It is argued whether the claim of better performance compared to radiologists is valid, as radiologists only had access to 1024x1024 downsampled 8-bit PNG images. However, in real world situations a radiologist is able to use a very high resolution image, and would refer to the patient history and include additional knowledge such as blood tests. For the comparison between radiologists and algorithm performances, the use of additional knowledge is forbidden, which makes radiologist performances debatable.

CheXNet shows the pathology locations using *Class Activation Maps* (CAM). CAMs are generated by feeding an image to the fully trained network, and then extracting the feature maps of the final convolution layer output.

A year later, the follow-up of *CheXnet* was published (Rajpurkar et al., 2018). This paper makes use of an ensemble of 10x *DenseNet-121* networks. Still, this network is trained on *ChestX-ray14*, but now the mean of the ensemble predictions result in the final output. In this version, the ensemble achieves an AUC of 0.944, while radiologists achieves an AUC of 0.940. In contrast to previous work of these authors (Rajpurkar et al., 2017), this time there is no significant difference between radiologist and algorithm performance on pneumothorax detection. There is no external validation done.

Pan; Agarwal, and Merck, 2019 applies an outlier detection approach. Instead of classifying individual pathologies as separate classes, they detect abnormalities in chest radiographs as a class, and then test whether the proposed model generalizes to data from external sources. The data comes from two hospitals: NIH (*ChestX-ray14*) and Rhode Island Hospital (private dataset). Two network architectures (*DenseNet* and *MobileNetV2*) are used, where models trained on the *ChestX-ray14* were also designed to predict the present of the 14 different pathologies. For individual pathology prediction, there is no external data validation done, as the data from Rhode Island Hospital only provided normal / abnormal labels. Achieves an AUC of 0.898 and 0.883 for *DenseNet* and *MobileNetV2* on pneumothorax detection, respectively. This shows that a lighter network like *MobileNetV2* is still able to generate good performances. For normal / abnormal predictions, they conclude that networks trained on one dataset also perform quite well on other datasets, but with decreased performance. That gives confidence that networks learn the generalizable features, while the predictions are still biased towards training data.

Majkowska et al., 2020 (Google Health) makes use of 600.000 images coming from two datasets. The first being data from the Apollo hospital, containing chest X-rays from multiple locations across the Apollo hospital network. 560.000 of these images are labeled using NLP. The second dataset is *ChestX-ray14*, which is also labeled using NLP tools. Eventually, 37.000 training images across the two datasets were visually reviewed by radiologists. The complete test set images were reviewed by four radiologist. Inverse probability weighting was used to see the positive radiograph enrichment and estimate the performance on the general population level.

They use a Xception network architecture pre-trained on 300 million natural images (*JFT-300M*). To compare, *ImageNet* is known to be trained on 16 million images.

Unfortunately, *JFT-300M* belongs to Google and is therefore not accessible to the public. Also, the weights for this pre-trained network are not downloadable. They achieve an AUC of 0.95 on the internal test set, and 0.94 on their own test split of *ChestX-ray14*. However, they publicly release this test split and release new labels that are verified by radiologists.

Rubin et al., 2018 trained their *DualNet* neural network using MIMIC-CXR. The novelty of this paper is that they use both frontals and lateral images as input. However, their baseline approach of DenseNet-121 with only one frontal or lateral input type seems to work just as well as their proposed *DualNet* approach (0.706 AUC). Their algorithm also not make use of data augmentation, but this issue will be addressed in future work.

Tang et al., 2019 does end-to-end deep adversarial one-class learning (DAOL) for semi-supervised normal-abnormal chest radiograph classification, using only normal X-rays. They reason that their DAOL network is able to reconstruct only normal X-ray images, and thus is able to differentiate between abnormal x-rays by using the discriminator of the DCGAN. The ChestX-ray14 is used to and not abnormal X-rays because it has not been trained on them. Uses the NIH Chest X-Ray dataset. All the images are resized to 64x64 and 128x128. It is arguable whether pneumothorax could be detected on such small input sizes.

Guendel et al., 2018 uses a variant of DenseNet-121. It achieves an AUC of 0,846 on the official test split. No external data was used for evaluation.

2.2 Noisy Evaluation

A few (peer-reviewed) articles present their methods, but their evaluation is not reliable. Most of the papers do not use patient-level partitioning between training and test splits, which makes evaluation invalid. This way, the same patient might be present in both the training set and the evaluation set, as there are multiple studies done per patient. In this section, we list the research that does not adhere to the practice of per-patient data splitting.

Cicero et al., 2017 classifies radiographs in multiple categories: normal, cardiomegaly, consolidation, pleural effusion, pulmonary edema, or pneumthorax. They reason that chest radiographs are a great initial application for deep learning, as there are a lot of datasets available and their acquisition technique is standardized. The total normal patient size is 11.702, and the amount of pneumothorax patients is 1.299. Small or mild pneumothoraces are excluded. As network architecture they choose GoogLeNet, which is a 22 layers inception architecture. They consider this network because of its computational efficiency, being capable to learn complex features, all while being less prone to over-fitting. Even with a reduced resolution (256x256 pixels) compared to other research, an AUC of 0,861 is achieved. We have to note that the algorithm is tested with a limited amount of samples (n=167, 78% specificity & 78% sensitivity). They agree with the fact that the current sample size is not large enough to test the quality of predictions for the pneumothorax class. They statistically prove that, at 1% prevalence and 78% sensitivity, a sample size of over 26.000 is needed. They also argue that the adoption of neural networks in clinical practice will depend on the "rationalization of their decision". We have to make sure the predictions are not dependent on external factors, but are only attributed to the x-rays

themselves. This reasoning conflicts with the research of Baltruschat et al., 2019, which uses external factors to improve predictions.

Annarumma et al., 2019 trains an *Inception-v3* network on a private dataset of 470.388 chest xrays from 2007 - 2017. They show that the average assessment time of their hospital is reduced from 11.2 days to 2.7 days for critical images, and from 7.6 days to 4.1 days for urgent images. The network consists of two Inception-v3 networks, with inputs of 299x299 and 1211x1031 respectively. They provide a soft-max output for the four classes: "critical", "urgent", "non-urgent" and "normal". The occurrence of a pneumothorax is classified in the class "critical". The results report an F1 score of 0.63, sensitivity of 65%, specificity of 94% on the critical class. Unfortunately, images are split on study level.

Guan et al., 2018 uses the *ChestX-ray14* dataset to train an *attention-guided convolutional neural network* (AG-CNN). This network consists of a global backbone (DenseNet-121 or ResNet50), and using the saliency map to generate a local image that goes into the classification pipeline of a separate local branch. This method achieves an AUC of 0.921 on pneumothorax detection. However, the test set is self-formed and the images are split at study level, not patient level. This means that the same patient can occur both in the training and test set (data leakage).

Taylor; Mielke, and Mongan, 2018 researches multiple network architectures (VGG16 / VGG19, Xception, Inception, ResNet pretrained on ImageNet) where two models are chosen: a 'high sensitivity' model (*VGG16*) with an AUC of 0.94, and a 'high specificity' model (*Inception*) with an AUC of 0.96. Evaluation happens on a private internal test set (human annotated). Small pneumothoraces are excluded from their training set. Also, during the generation of train & test splits, images are separated on study level instead of patient level. They externally validate their methods on NIH ChestX-ray14 (Chapter 3.2.1), however the performance drops significantly. Evaluation on the external dataset results in a low sensitivity (0.28-0.49) and achieves an AUC of 0.75.

2.3 Commercial solutions

Commercial-grade solutions towards pneumothorax detection also publish their research. Evaluation methods of this type of research is extensive. However, they mostly train with private datasets and do not publish their models online. Below, we list two of the research papers.

Putha et al., 2018 shows that an AI system (*Qure.ai*) can reliably separate normal chest x-ray images from abnormal ones. For this, they used a 2.3 million well-labeled X-ray dataset for training (NLP from radiology reports), and 100k for validation (labels from radiology reports) from 45 centers. Also, 6 radiologists looked at a separate test set of 2000 images (coming from 3 other hospitals). However, this separate test set from the different hospitals were filtered out in a way that they do not contain supine position images (patient lying down on a bed facing upwards). Also, all the bedside or portable X-ray machine images were left out, giving us no images of tubes, cathethers, ECG leads or pacemakers. Putha mentions that the public datasets (*CheXpert*, *MIMIC-CXR* and *PadChest*) have a great size, but also contain a lot of follow-up images which reduces the variability. In our research, we made sure

to address these problems (Chapter 3.2). Their advantage of having images originating from multiple (45!) centers is that the chest x-ray is invariant to chest x-ray hospital procedures. Also, different chest x-ray manufacturerrs and settings are robustly trained for.

Hwang et al., 2019 shows that their AI system (*Lunit*) is able to separate normal chest x-ray images from abnormal ones. Their dataset consists of 100k images from a private dataset of one hospital, where each image is reviewed by at least 1 of their 15 radiologists (with a minimum of 7 years of experience). These radiologists also provide highly localized per-pixel segmentation for each pathology. External validation is done on 1000 images from 5 institutions. In their validation set, they exclude pneumothorax cases of:

- Images with thorax drain cathethers
- Images containing subcutaneous emphysema
- Images with other clinically relevant abnormalities
- Images taken immediately after thoracic surgery

They also show that their algorithm has a significantly better performance (AUC of 0.983) compared to radiologists (AUC of 0.932) in terms of abnormal / normal image classification.

Paper	Method	Datasets	# Total	# Pneumothorax	Radiologist verified	External Validation	Split	Performance (AUC)
Gooßen et al., 2019	CNN (ResNet-50), FCN, MIL	Washington Medical Center (Private)	1.000	437	NLP labels	No	Patient	0,96
Park et al., 2019	YOLO Darknet19	Private	11.137	1.596	Yes	No	Patient	0,898
Cicero et al., 2017	GoogLeNet CNN	'tertiary care center' (Private)	13.001	1.299	NLP labels	No	Study	0,861
Blumenfeld; Konen, and Greenspan, 2018	CNN pixel segmentation	Private	117	Unknown	Unknown	No	Unknown	0,95
Annarumma et al., 2019	InceptionV3	Private	470.388	Unknown	Yes (4.551/47.849 in test set)	No	Study	F1: 0.63 (sensitivity 65%, specificity 94%)
Wang et al., 2017	ResNet-50 (AlexNet, GoogLeNet, VGGNet-16 also considered)	ChestX-ray8	108.948	Unknown; 80.000 normal images	NLP labels	No	Patient	0,7891
Guan et al., 2018	DenseNet-121 & ResNet-50 (AG-CNN)	Chest X-ray14	112.120	5.302	NLP labels	No	Study	0,921

Rajpurkar et al., 2018	10x DenseNet-121 (Ensemble)	Chest X-ray14	112.120	5.302	NLP labels	No	Patient	0,944. Radiologist performance: 0,940
Guendel et al., 2018	DenseNet-121	Chest X-ray14	112.120	5.302	NLP labels	No	Patient	0,846
Baltruschat et al., 2019	ResNet-50	Chest X-ray14	112.120	5.302	NLP labels	No	Patient	0,840
Pan; Agarwal, and Merck, 2019	DenseNet-121 & MobileNetV2	Chest X-ray14	112.120	5.302	NLP labels	No	Patient	0,898 & 0,883
Rubin et al., 2018	*DualNet* (2x DenseNet-121 for PA & Lateral)	MIMIC-CXR	106.875	∼3.847 (prevalence)	NLP labels	No	Patient	0,706
Majkowska et al., 2020	Xception (private pre-training on Google JFT-300M)	Private & Chest X-ray14	759.611	283 in test set	Yes (complete validation & test set)	Yes, 5 centers	Patient	0,94
Putha et al., 2018	SENet	Private	2.300.000	Unknown	Yes (100k validation set, 2k test set)	Yes, 45 centers	Patient	0,9227, but 0,8558 during external validation
Hwang et al., 2019	DenseNet-121 (Ensemble)	Unknown	98.621	113 in test set (41.140 abnormal)	Yes, 15 physicians	Yes, 5 centers	Patient	0,95

Taylor; Mielke, and Mongan, 2018	VGG16/19, Xception, Inception, ResNet	Private	13.292	3.107	Yes	Yes, on Chest X-ray14	Study	0,86, but 0,75 on external NIH validation set.

TABLE 2.1: Structured summary of related work on automated pneumothorax detection.

2.4 Comparison

When comparing the related work as seen in Table 2.1, we see a lot of different reported performances throughout the years for pneumothorax detection. There are three important findings that we can conclude from the table:

1. Most research does not test their algorithm with external validation data
2. Some research creates validation splits on study level, not patient level
3. Some research uses private datasets to train and/or verify their models
4. Various performance metrics are reported

External validation One important finding is that most papers do not validate their algorithms on an external dataset; data coming from a different hospital than that there has been trained on. Kim et al., 2019 found that of 516 published peer-reviewed medical imaging studies, only 31 (6%) include external data validation. External validation is important, as algorithms need to show robustness by generalizing well towards data from other hospitals. Ideally, this external validation is a public dataset where everybody can test their algorithm. In related work we see popularity towards the *ChestX-ray14* (Wang et al., 2017) dataset.

Data splits Guendel et al., 2018 found out that in *ChestX-ray14*, around 3.6 studies per patient are present. When creating data splits on study level, a patient may be present in the training, validation, and test split! Following Table 2.1, we see that 3 research papers do not split on patient level, and are thus prone to this error.

Private datasets In literature where only private datasets are used, it is often not mentioned what data exactly has been selected or left out for the experiment. We should be cautious of trusting this research, as these papers might be fallible to 'cherry-picking'; conveniently leaving out test samples that decrease model performance. We can only confirm the research by evaluating on a public dataset.

Performance metrics Most papers report various metrics, where the most popular metric is the area under the ROC curve (AUC). However, some papers only mention the positive predicted value (PPV), sensitivity, specificity, and F1 score. There is no consensus about which metrics to report. This makes it harder to compare the results of different approaches. In Chapter 3.4, we further evaluate which metrics are important and why.

Chapter 3

Methods

In this chapter, we address the methodological approaches to pneumothorax detection. We cover the feasibility of (public) datasets, consider deep learning architectures, and decide on performance metrics. Finally, we try to do external data validation by keeping a keen eye on patient-level data splitting.

3.1 Domain Knowledge

Before covering the dataset in itself, we need to know how this data is acquired and what its limitations are. By closely inspecting the type of images in the datasets, we are aware of possible biases. This increases our understanding of the problem, increases model performance, and sets us apart from other related work.

We find a couple of problems in terms of discovered bias. As mentioned in Chapter 1.1.2, bed-side pictures are always taken with a portable x-ray machine of patients in supine position (AP, lying down). Patients that just have had surgery and are in the intensive care department often get a chest x-ray taken. This is done to exclude possible complications that were missed during surgery. For example a pneumothorax caused by the surgery, which occurs quite often (Choi et al., 1998). The algorithm could be biased towards images coming from the portable x-ray machine, giving it a (incorrect) higher probability of containing a pneumothorax. Mahajan et al., 2020 shows that by having a good collaboration between data scientists and radiologists, one is able to interpret these algorithmic failures. Once these failures (e.g. chest tubes in images) are detected, dealing with these problems during data preprocessing tend to pay off in an increase in model performance.

3.2 Datasets

In this section, we explore the datasets that we use for pneumothorax detection. After giving an overview, we state the methods of deriving our preprocessed datasets. In Table 3.1 we show a summary of the final number of images used with the corresponding pneumothorax prevalence.

In general, we are only interested in datasets providing images labeled for a pneumothorax. Ideally, we would want to train on healthy lungs versus lungs with a pneumothorax in order to learn the absolute difference. However, as mentioned in Chapter 1.1, a pneumothorax can occur as a side effect of underlying diseases. This means that for many images containing a pneumothorax, there will be other comorbidities present. Generally speaking, images containing a pneumothorax have a high probability of containing other disease categories such as infiltration, effusion, emphysema, or nodules. Next to comorbidities, it is important to know what the

conditions are when a chest x-ray is taken, as the timing can introduce a possible bias (see Chapter 1.2.1).

3.2.1 NIH: ChestX-ray14

Note: not in use for model development.

ChestX-ray14 consists of data from the National Institutes of Health Clinical Center (NIH) (Wang et al., 2017), containing 60% of all the frontal chest x-ray images of the hospital. The dataset consists of 112.120 frontal chest x-ray images of 30.805 patients, where 7.134 images contain a pneumothorax. NIH annotates the *ChestX-ray14* dataset by applying machine learning-based natural language processing (instead of expert systems), which comes with a lot of badly labeled images. In general, we observe a 30-90% error rate depending on the class. The documentation of this dataset leaves out important information regarding the label construction, and in which cases we can or cannot apply this dataset. Also, the resolution has been downsampled from 3000x2000 to 1024x1024 (Oakden-Rayner, 2019). A point of critique regarding pneumothorax labels is that this dataset does not differentiate between untreated & treated pneumothoraces. It means that there are many thorax drains present, which is a sign of a treated pneumothorax (i.e., already detected, treated, and currently in the healing phase). Baltruschat et al., 2019 shows that a pneumothorax detection algorithm trained on NIH *ChestX-ray14* would treat images containing a chest tube as a pneumothorax. Therefore, the algorithm learned the chest tube instead of the pneumothorax. Because the pneumothorax labels contain structured noise, we do not use this dataset to train our models.

Improved labels Majkowska et al., 2020 (research by Google) provides 1962 multi-radiologist annotations and performance results for the NIH *ChestX-ray14* test set. Although the problem concerning images with treated pneumothoraces remains, these annotations drastically improve label quality. We use these labels to evaluate our model performances as well.

3.2.2 SIIM-ACR (Kaggle)

The SIIM-ACR dataset is generated by the Society of Imaging Informatics in Medicine (SIIM) as part of the pneumothorax detection challenge on Kaggle. This dataset seems to be an annotated subset of the NIH ChestXray14 dataset. SIIM is not transparent in the assembly of this dataset, nor do they provide patient or series information for the collected images (only image filenames). By not being transparent, other researchers are unable to define their patient-based data splits and verify the correctness & usability of the dataset. SIIM provides a training and test split but does not provide any details on how it is formed. As it turns out, some participants of the Kaggle challenge found that there appears to be a data leakage between the training and test set. Apparently, images of the same patient are present in both the training and test set. Therefore, we can conclude that there might be multiple series per patient present, but we are unsure of how many series are used per patient. For other datasets where patient and series information is included, we only obtain the first series to make sure each included image consists of unique lungs.

The images that are present, however, contain detailed segmentation masks. This is the only dataset that contains annotations on pixel-level and is therefore very useful to compare segmentations against visualizations of a trained neural network. Of all the x-ray images, 8.296 are non-pneumothorax, and 2.379 contain a pneumothorax with a corresponding segmentation mask. Thirty-seven of these images seem to have no label at all, and therefore we discard them.

3.2.3 CheXpert

With the aim of having a robust penumothorax detection system, we have used several public datasets coming from different hospitals. One of our training dataset is the *CheXpert* dataset, a commonly used dataset that comes from the Stanford ML group Irvin et al., 2019. The Stanford ML group was also in need of a high-quality dataset other than *ChestX-ray14*, therefore releasing a dataset containing a lot of expert annotations. The complete dataset contains 224.316 entries of chest X-rays of a total of 65.240 patients. From these X-rays, they extract the observations of 14 different diseases using NLP. Compared to other datasets, where the test set is developed with NLP (e.g., NIH *ChestX-ray14*), the Chexpert validation & test set is labeled manually by expert radiologists. The dataset distribution of CheXpert in terms of pneumothoraces are as follows:

- 19.448 pneumothorax images
- 56.341 non-pneumothorax images
- 3.145 images where there is uncertainty about the presence of a pneumothorax

Just like *ChestX-ray14*, *CheXpert* does not differentiate between untreated and treated pneumothoraces. What then happens is that nearly half of the pneumothorax images contain a chest tube, which is an indication that the pneumothorax has already been treated. The presence of a chest tube means that the images of these patients are taken after treatment occurred, and do not indicate a first occurrence of a pneumothorax as is used for diagnosis. In Figure 3.1, we show the difference between a treated and untreated patient.

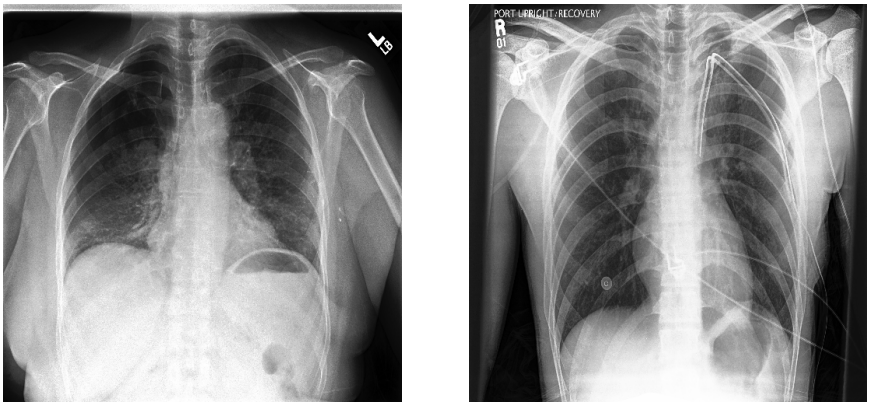
By visually inspecting some patient images, we see that there are multiple series per patient, sometimes even up to 60 series. We know that in follow-up chest x-ray series, the patient is often already treated and have a thorax drain in place. Therefore, we choose to select only the first chest x-ray series, as these are most likely to have untreated pneumothoraces (and thus no thorax drains). Also, this methodology provides us with images containing unique lungs, without duplicates. Because the CheXpert dataset contains both frontal and lateral images, we further reduce the image selection by selecting only the frontals. After applying the steps, we end up with the following number of images:

- 3.698 patients with an observed pneumothorax
- 21.590 patients without an observed pneumothorax
- 794 patients where it is unclear whether there is a pneumothorax observed

After looking at the training split of Chexpert, we apply the same logic to the pre-defined validation split. Then, we can add 7 more patients with a confirmed pneumothorax, and 195 without a confirmed pneumothorax.

The downside of CheXpert is the format of the data in which it is provided. The standard format for X-ray imaging is DICOM (.mha/.mhd). The format of CheXpert images is in 8-bit JPEG. There are two problems with their conversion from DICOM to JPEG:

1. DICOM contains pixel spacing information, so exact measurements can be made to determine, e.g., the size of the heart.
2. DICOM has a higher 16-bit representation (65.536 possible values per pixel) compared to 8-bit JPEG (255 values per pixel), which means that JPEG downsamples significantly, losing much information.



(A) Untreated patient #482 (B) Treated patient #78

FIGURE 3.1: Untreated patient (A) vs treated patient (B). The treated patient shows the presence of a chest tube in the left lung. The untreated patient does not have this tube.

Dataset	# Total Samples	# Pneumothorax	% Pneumothorax
CheXpert	10.836	3.603	33.25%
SIIM-ACR (Kaggle)	10.648	2.379	22.34%
MIMIC-CXR	5.124	1.281	25%
*RadboudCXR**	*28.207*	*6.500*	*23.04%*

TABLE 3.1: Overview of the used preprocessed datasets, including the number of positive pneumothorax / true positive (TP) cases.
* We use this internal dataset only for domain adaptation, and is further explained in Chapter 12.1.

3.2.4 MIMIC-CXR-JPG

Johnson et al., 2019 consists of 370k images. Of the complete dataset, 65% were PA & lateral (fixed machine), whereas 33% were made using a portable device (AP images). 46.5k of the images are confirmed to not be a pneumothorax with frontal images, belonging to 7.933 unique patients. In the dataset, the pneumothorax prevalence is 3.2%, which means that we have 11.6k images of 1.281 patients containing a pneumothorax. In follow-up studies of the patients, these images often contained thorax drains, which we want to exclude. We only include the first study of a patient, which gives us a total of 1.281 usable (unique) pneumothorax images and 7.933

images of patients not having a pneumothorax. To avoid having a too unbalanced dataset, we acquire 3x the rate of pneumothorax images, which means that we use 3.843 non-pneumothorax patients randomly.

All of the images contain additional text reports and are parsed with the same tools as used for the *CheXpert* dataset. Access to this dataset is heavily controlled. Getting access requires users to sign a user agreement stating that algorithms resulting from using this dataset should be public. Also, one needs to complete the "Data or Specimens Only Research" course from MIT before obtaining credentials to download.

3.3 Network Architectures

Deep neural network architectures (fully connected networks, convolutional neural networks, etc.) have been widely researched as a fundamental part of deep learning. The downside of applying deep learning on medical imaging is the limited size of (labeled) datasets. Therefore, training from scratch is often not ideal, because it will take the network longer to converge. To overcome this issue, we make use of existing network architecture research and use pre-existing networks with a significant track record. We re-train the pre-existing networks towards our own problem using transfer learning. Transfer learning allows us to make use of learned features such as the detection of lines and edges that we can adapt to fit our problem space. For all selected models, we pre-load their trained *ImageNet* weights and retrain the complete architecture. No layers are frozen. The input & output layers are excluded, as we need to define our own input sizes and class labels. We depict our architecture in Figure 3.2. A detailed pipeline and the steps to (hyper)parameter training can be found in Appendix A.

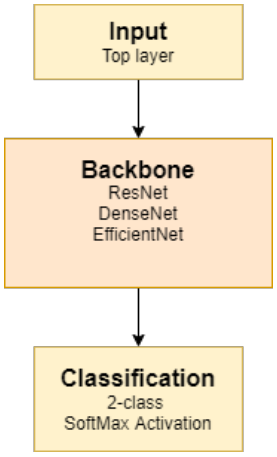


FIGURE 3.2: General network architecture. We customize the top-layer in order to use our own specified input dimensions, and append a 2-class dense layer for final predictions.

We select three network architectures according to the state-of-the-art on the ImageNet classification task and usage in relevant related work. It is shown that models performing higher on ImageNet also tend to perform high on other tasks through transfer learning, and that *ResNet* (Chapter 3.3.1) & *DenseNet* (Chapter 3.3.2) architectures achieves a higher accuracy compared to other models (Kornblith; Shlens, and Le, 2019). The most recent state-of-the-art architecture that is released after the paper of Kornblith; Shlens, and Le, 2019 is *EfficientNet* (published mid 2019, Tan

and Le, 2019). For our backbone, we implement the following network architecture concepts:

- Residual Networks (*ResNet*) (He et al., 2016).
- Densely connected networks (*DenseNet*) (Huang et al., 2017). Particularly *DenseNet-121*.
- *EfficientNet* (Tan and Le, 2019).

3.3.1 Residual Networks

The residual network (*ResNet*) by He et al., 2016 is a relatively old network configuration, but still able to compete with the state-of-the-art ImageNet and recent medical imaging X-ray tasks (Gooßen et al., 2019) (Wang et al., 2017) (Taylor; Mielke, and Mongan, 2018). For robustness we add *batch normalization* layers, and no *dropout* is used. Using both *batch normalization* and *dropout* simultaneously tend to increase training time and decreases regularization performance (Ioffe and Szegedy, 2015).

3.3.2 Densely Connected Networks

Densely connected convolutional networks (*DenseNet)* (Huang et al., 2017). The difference between residual networks and densely connected networks is that each layer of a densely connected network is connected with each of the individual layers where it is propagating through. It has been shown that this approach reduces the vanishing gradient problem (where the gradient becomes so small that it has no significant effect on weight updates), support re-use of features but simultaneously decreasing the number of parameters, and decreasing training time.

In related chest x-ray classification tasks, *DenseNets* achieve good results (Rajpurkar et al., 2018) (Hwang et al., 2019). This architecture, specifically *DenseNet-121*, is known to be the backbone of the popular *ChexNet* architecture, which claims radiologist-level performance on pneumonia detection (Rajpurkar et al., 2017). The authors prefer this network, because it "improves flow of information and gradients through the network, making the optimization of very deep networks tractable".

3.3.3 EfficientNet

Currently, no papers argue the reasoning of a chosen depth or width of a neural network. Tan and Le, 2019 rethinks the current approaches towards new architectures by proposing a scaling method for the depth, width & resolution parameter. On top of this, they present a new baseline network architecture called *EfficientNet* and use their scaling method increase capacity. The results on *ImageNet* are state-of-the-art while having relatively lower parameters compared to its competitors. This means that *EfficientNet* is able to converge fast while not sacrificing performance.

From this model family, we include models ranging from *EfficientNet-B3* (small) to *EfficientNet-B6* (large). We choose *EfficientNet-B3* as low-end, as this network capacity corresponds with competing network architectures. Due to computing capacities we are unable to consider even larger networks such as *EfficientNet-B7*, *EfficientNet-L1*, and *EfficientNet-L2*.

3.4 Metrics

Most papers report various metrics, where the most popular metric is the area under the ROC curve index (AUC). However, the AUC score does not capture complete model performances. To complement the AUC score, a metric considering precision & recall such as the F1 score or AUCPRC is additionally mentioned.

AUC / AUCROC The AUC score is calculated by using the *true positive rate (TPR)* (= recall) (true positives / (true positives + false negatives)) and the *false positive rate (FPR)* (false positives / (false positives + true negatives)). AUC measures the volume that the ROC curve is generating by computing the sensitivity and 1-specificity by evaluating all possible threshold values. The greater this area, the better the algorithm tends to be. The axis of a ROC plot consists of the false positive rate (1-specificity, FPR) against the true positive rate (sensitivity, TPR). These numbers are generated by varying the algorithmic threshold. Then, for each threshold, there exists a combination of false positives & true positives, transforming the point cloud into a curve. To choose the optimal model, we can derive the threshold by looking up the desired FPR / TPR.

AUCPRC Next to the AUCROC score, there exists the Area Under the Precision / Recall Curve (AUCPRC). The apparent difference between AUCROC and AUCPRC is in the calculation of the curve. On one side we have AUCPRC, which is calculated by using the precision (true positives / (true positives + false positives)) and recall (true positives / (true positives + false negatives)). The combination of precision and recall is also known as the F1 score.
As we see in Table 2.1, most research only reports one metric, in particular the AUCROC score. We can argue that when there is a class imbalance (in our case, more non-pneumothorax than pneumothorax samples), one should use a different metric that can deal with class imbalance. For class imbalance datasets, the AUCROC curve can be misleading, and reporting the AUCPRC tends to provide better visual cues (Saito and Rehmsmeier, 2015). The AUCPRC does not include true negatives in its calculation, which for disease classification tend to come for 'free'. Some studies mention the precision (i.e. PPV) together with AUC scores in order to deal with class imbalance. Ranjan et al., 2018 reports only the AUCPRC score for their research and suggests that the AUCPRC is a better metric for *ChestX-ray14* due to sparseness of some diseases. Reporting AUC metrics are indeed useful, but it can fail to mirror expected real-world performance because of issues in prevalence and possible under-representation of critical findings (Majkowska et al., 2020)

As an illustrated example, if we have one million healthy x-rays, and 10 x-rays contain a pneumothorax, then we are not interested in how many times the algorithm has found a true negative (healthy x-ray), but we want to maximize the fact that every pneumothorax image will be found (precision and recall)! It is debatable whether we do/do not care about true negatives, as we will use this algorithm in clinical routine. If the true negatives are too low, this will give the impression that the algorithm is not reliable.

To conclude this chapter, for our experiments we report the AUC score together with the class-weighted F1 score. When related work reports different metrics, we also compute the same metrics in order to compare results.

Chapter 4

Results

In this chapter, we show the results of our algorithms (*DenseNet*, *EfficientNet*, *ResNet*) trained on three different publicly available datasets. The chosen hyperparameters and training pipeline is further explained in Appendix A. We first evaluate on the internal test set, which consists of predefined test splits of all three datasets. Then, we visualize a few examples of our network output using Grad-CAM, which might potentially be presented to the radiologists in order to provide an insight into what the algorithm has learned. Finally, we evaluate our algorithm externally on the original test split of *ChestX-ray14* with the labels of Majkowska et al., 2020. This way, we try to compare our performance against the performance of related work.

4.1 Internal validation

For each dataset, we exclude 20% of the total training files for validation and testing (10% and 10%, respectively). During preprocessing, we already make sure to include only one image per patient (first study of the patient, Chapter 3.2), so patient-level splitting has already been dealt with. We depict a complete overview of data splits for each dataset in Table 4.1.

Dataset	# Train (80%)	# Validation (10%)	# Test (10%)
CheXpert	8.669	1.084	1.083
SIIM	8.518	1.065	1.064
MIMIC-CXR	4.099	512	512

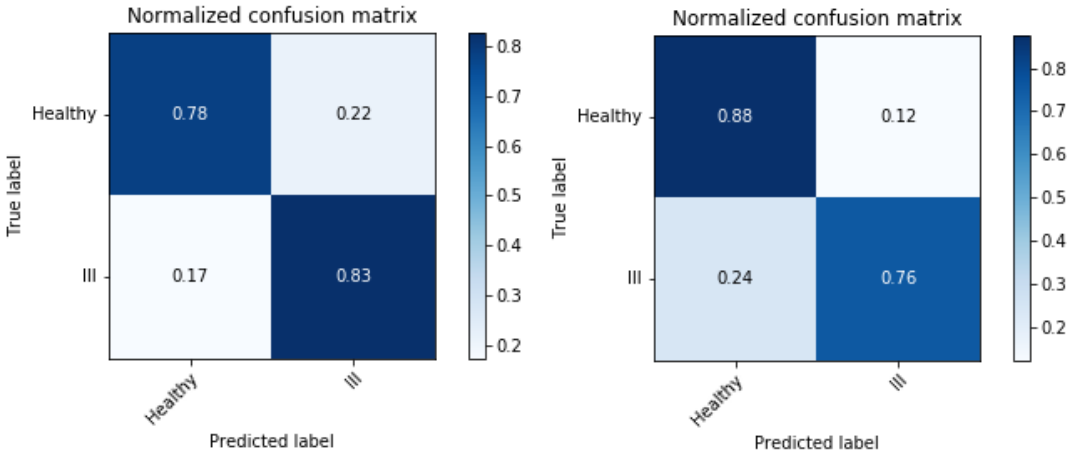
TABLE 4.1: Data splits per dataset that is used to train & internally test our algorithm.

Internal validation means validating on data that has the same origin as the training data. The data samples in the datasets might be different, but the distributions and biases that exist in the dataset are still present. By evaluating on an internal validation set, we might not be able to compare results with other research unless there is a pre-defined test split available. However, not for all datasets there is an official test split available. For our internal evaluation, we depict the results on the test splits in Table 4.2.

In Table 4.2, we see that in general, results for higher image dimensions are better. One outlier that does not fit this logic is *ResNet-152*. During this experiment, we saw that the loss kept increasing already after the first epoch. Thus, we can argue that the capacity of *ResNet-152* is too limiting for higher image dimensions. We achieve best results with EfficientNet-B3 in terms of AUC, and best results with DenseNet-121 in terms of (class-weighted) F1 score. For these two results, we show their corresponding ROC curve and confusion matrix in Figures 4.2 and 4.1.

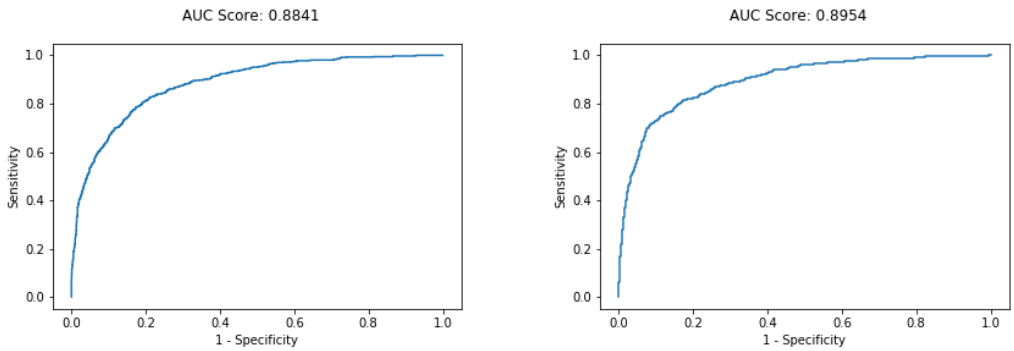
Model	Dimensions	F1 Score	AUC Score
ResNet-152	512x512	0.83	0.8720
EfficientNet-B3	512x512	0.76	0.8622
DenseNet-121	512x512	0.83	0.8856
ResNet-152	1024x1024	0.79	0.8339
EfficientNet-B3	1024x1024	0.83	**0.9148**
DenseNet-121	1024x1024	**0.85**	0.8954

TABLE 4.2: Results on the test splits of the used datasets. Best performance is achieved at higher image dimensions.



(A) EfficientNet-B3 (1024x1024) (B) DenseNet-121 (1024x1024)

FIGURE 4.1: Normalized confusion matrices of the best performing networks. EfficientNet-B3 shows less false-negatives and thus is more sensitive, while DenseNet-121 shows less false-positives, and thus is more specific.



(A) EfficientNet-B3 (1024x1024) (B) DenseNet-121 (1024x1024)

FIGURE 4.2: ROC curves of the best performing networks.

4.2 Visualization

In order to try to understand what a neural network has learned, we can visualize the learned gradients based on network input. A popular method that is proven to present the propagating class gradient reliably is Grad-CAM (Selvaraju et al., 2017) (Pasa et al., 2019). We calculate the Grad-CAM by computing the derivative of the predicted class against the activation of a feature map (convolution layer in the network). If the computed derivative is small, then the feature map does not contribute much towards the final output. However, when the derivative is large, the activation is important for the final output. By computing the derivative for each spatial convolution filter in the layer, we can reconstruct a complete image. As we are interested in parts of an image that contribute to a positive prediction of a pneumothorax, we compute Grad-CAM visualizations for only the positive class and not vice versa.

By presenting the Grad-CAM visualization next to the predicted pneumothorax probability, we can show radiologists which part of the image is the most important. One remark that we have to make is that the most important part of the image might not always be the location of a pneumothorax. There are several symptoms that can indicate the presence of a pneumothorax, including the *deep sulcus sign* and *mediastinal shift* (Chapter 1.1.2). Before we present the computed Grad-CAM to the radiologist, we threshold the image to make sure only the most important regions are shown. For our experiments, we set the threshold to 0.7.

Next to Grad-CAMs, we provide occlusion sensitivity maps, where we measure the difference in scores by occluding parts of the image using small patches (Zeiler and Fergus, 2014). The greater this difference in score, the more 'sensitive' this part of the image is to the pneumothorax class. However, it has been shown that Grad-CAMs are more interpretable and more 'faithful' compared to occlusion sensitivity maps (Selvaraju et al., 2017).

4.2.1 Examples

We show a correctly classified example (true positive) of each dataset (*SIIM*, *Chexpert*, *MIMIC-CXR*). For the *SIIM* dataset, we add the segmentation mask as a reference image, as these examples have pixel-level annotations from radiologists. In this way, we can investigate at what image features the algorithm looks at. There are multiple possibilities:

1. The activation is concentrated around the pneumothorax area.
2. The activation is concentrated around areas showing symptoms of a pneumothorax (e.g. *deep sulcus sign*). These are scenarios where
3. The activation is concentrated around areas not related to a pneumothorax, but on bias (e.g. chest tubes, AP / Portable image types, pacemakers, etc.). This type of activation map is something we want to avoid. If this type is observed, it shows that the algorithm selects its classes based on tertiary effects other than the pneumothorax.

We show visualizations for the *EfficientNet-B3* network architecture, as these activations tend to be the better compared to other networks.

Figure 4.3 shows an untreated patient from PA view while standing upright. This is a normal procedure for patients coming in that are not (yet) critically ill. These

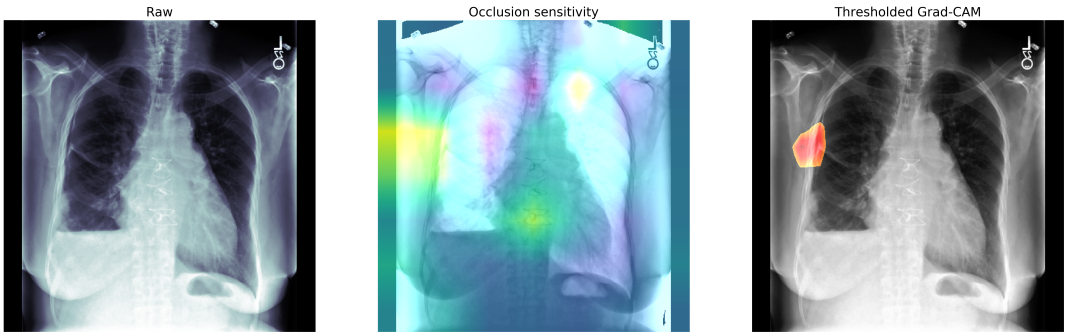


FIGURE 4.3: PA chest-xray Grad-CAM of patient p15794797 from MIMIC-CXR. Pneumothorax present in the right lung laterally. EfficientNet-B3 predicted a pneumothorax probability of 90.12%.

patients are able to walk to the radiology department and stand in front of a fixed x-ray device. This generally assures better image quality as well. In this case, we see a pneumothorax on the right lung at the outer lateral side.

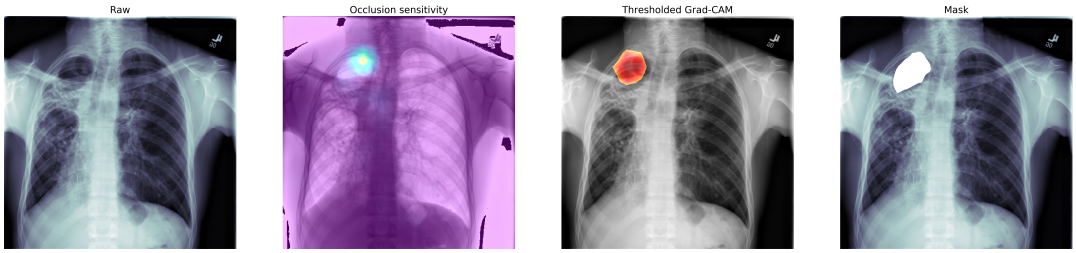


FIGURE 4.4: AP chest x-ray from the *SIIM* dataset. The Grad-CAM activation seems to be completely covering the segmented area of the pneumothorax (top of the right lung). *EfficientNet-B3* predicted a pneumothorax probability of 99.92%.

For the *SIIM* dataset, We show the activations of *EfficientNet-B3* together with the corresponding radiologist pixel-level annotations. As a sidenote, we do not train our algorithms with pixel-level annotations, but provide an image-level label for each image. As we see in Figure 4.4, the thresholded Grad-CAM exactly marks the location of the pneumothorax.

In Figure 4.5, we see an example of an AP view image, where the patient is in lying down on the bed (supine position). The patient is critically ill, as it is connected to all sorts of wires. A thorax drain is also present, which shows that the patient has been treated and is now in the recovery phase. Chest x-rays are made during the recovery phase to ensure the lung is re-inflating and the patient gets better. The Grad-CAM shows activations on the lower left part of the lung. A situation like this is typical for patients with a pneumothorax lying down. As found in Yarmus and Feller-Kopman, 2012, the air bubble of a pneumothorax cannot travel to the top part of the lung because the patient is in a supine position. Thus, air collects in the anterior (frontal side of the body) side without clear marks for the lung edge. Air will travel around the chest, which results in the *deep sulcus sign*, as we see highlighted by our algorithm.

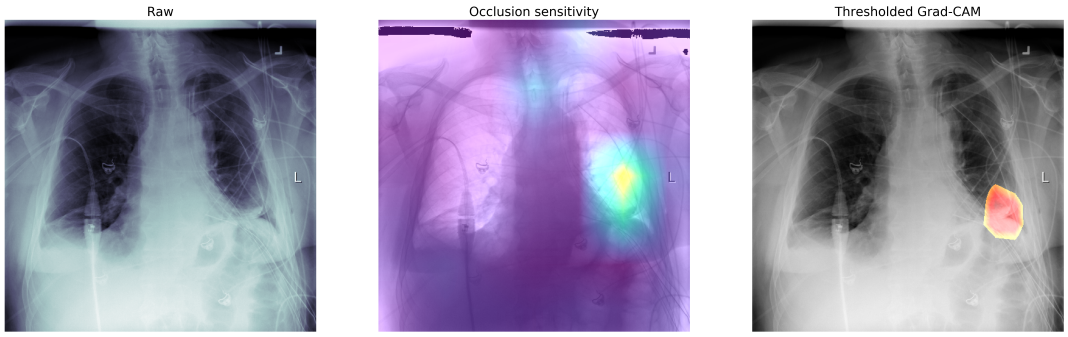


FIGURE 4.5: Grad-CAM of a pneumothorax patient in supine position (AP view) showing a *deep sulcus sign* in the left lung. Thorax drains are present in the same lung. EfficientNet-B3 predicted a pneumothorax probability of 99.21%.

4.2.1.1 False positives

Some images are incorrectly classified as the pneumothorax class. This means that the images do not contain a pneumothorax, but the algorithm predicts that there is. Sometimes this is a genuine mistake of the classifier, other times this is debatable. One example is Figure 4.6, where the algorithm incorrectly classifies this image as having a pneumothorax. This false positive can be explained by the fact that there is a chest tube in place, which is a sign that the patient is currently recovering from a pneumothorax.

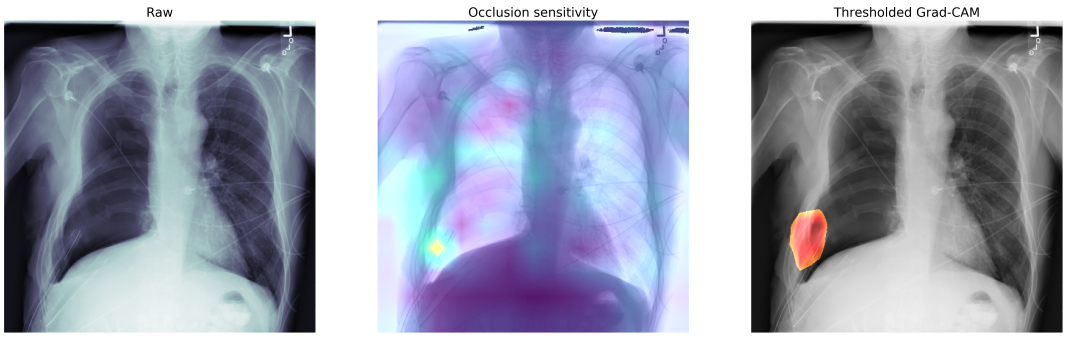


FIGURE 4.6: False-positive classification of EfficientNet-B3 on an image from the SIIM dataset. However, the patient was treated for a pneumothorax in the same lung, as there are still chest tubes present.

4.2.1.2 False Negatives

In the cases where there is a pneumothorax present but not been picked up by the algorithm is called a *false negative*. In these cases, the algorithm was not sensitive enough to pickup the pneumothorax. Often, we can explain this by the fact that there are other comorbidities present which can confuse the algorithm, or that the pneumothorax is small. In Figure 4.7 we show an example of a false negative found in the *SIIM* test set. The segmentation maps show that the pneumothorax is relatively small compared to others. The thresholded Grad-CAM shows activations at a part of the pneumothorax area, but seemed it was not significant enough to give a positive prediction (48.53% probability was predicted).

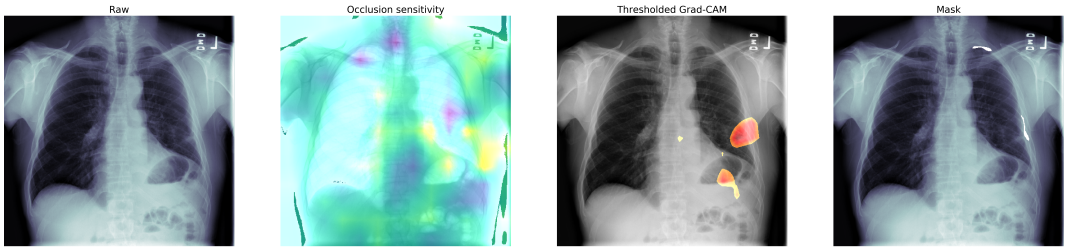


FIGURE 4.7: False-negative classification of *EfficientNet-B3* on an image of the *SIIM* dataset. In this case, the pneumothorax is missed because it is really small. The predicted pneumothorax probability is 48.53%, therefore classifying it as non-pneumothorax.

4.3 Performance vs related work (*ChestX-ray14*)

In related research using the *ChestX-ray14* dataset, Rajpurkar et al., 2018 found that radiologists achieved an AUC of 0.940 based on 45 pneumothorax images of a total of 420 images. The *CheXneXt* algorithm reached an AUC of 0.944 on this small test set for pneumothorax detection. Statistically speaking, the authors found no significant difference in performance. The assessment of radiologists took 240 minutes to complete, which means around 34 seconds to assess an image. Our algorithm needs just 34ms for an image to propagate through the neural network.

Majkowska et al., 2020 publishes new and improved labels for the *ChestX-ray14* validation and test set. The final evaluation set consists of 1.962 images where 195 images contain a pneumothorax. Instead of reporting F1 measures or an AUC score, Majkowska et al., 2020 reports specificity, sensitivity, and PPV. Radiologists achieve an average specificity of 92.8, sensitivity of 79.2, and a PPV of 54.8. In order to compare the results against our networks, we choose our positive prediction threshold such that the algorithm's sensitivity matches the sensitivity as achieved by the radiologist. The eventual threshold for *EfficientNet-B3* (1024x1024) is 0.8539, *EfficientNet-B4* (512x512) is 0.4363, and the threshold for *DenseNet-121* (1024x1024) is 0.6721. In Table 4.4 we compare our results with the algorithm of Majkowska et al., 2020 and their radiologist performance. As we can see, we generally achieve better specificity and sensitivity, but not in terms of PPV.

Research	**# Pneumothorax**	**# Total**	**Model AUC**
ChestXray14 (Wang et al., 2017)	5.302	112.120	0.7993
ChestXray14 (Taylor; Mielke, and Mongan, 2018)	5.302	112.120	0.75
ChestXray14 (Guendel et al., 2018)	5.302	112.120	0.846
ChestXray14 (Rajpurkar et al., 2018)	45	420	0.944
ChestXray14 (Majkowska et al., 2020)	195	1.962	0.94
EfficientNet-B3 (1024x1024)	195	1.962	0.939
EfficientNet-B3 – UDA (Chapter 13) (1024x1024)	195	1.962	0.944
DenseNet-121 (1024x1024)	195	1.962	0.942

TABLE 4.3: Test set sizes of ChestXray14 models with their corresponding pneumothorax model scores (AUC)

In terms of algorithmic comparison, where we compare all thresholds instead of evaluating a fixed number, Majkowska et al., 2020 reaches an AUC of 0.94 on

Model	Specificity (%)	Sensitivity (%)	PPV (%)
Majkowska et al., 2020	90.8	72.8	**48.7**
EfficientNet-B4* (512x512)	86.8	79.4	32.7
EfficientNet-B3* (1024x1024)	91.5	**80.1**	44.8
DenseNet-121* (1024x1024)	**91.69**	79.43	45.12
Radiologists	***92.8***	***79.2***	***54.8***

TABLE 4.4: Performance of the models and radiologists on the ChexRay14 test set of Majkowska et al., 2020.
* Algorithm thresholds are determined by average radiologist sensitivity.

the *ChestX-ray14* dataset. Our *EfficientNet-B3* (1024x1024) algorithm achieves equal performance (0.9391 AUC), even though we only train on public data.

As we mentioned in Chapter 3.2.1 we use the *SIIM* dataset to train our networks. We have observed that this is a subset of *ChestX-ray14*. We are aware that this can cause data leakage and we explain how we deal with this in Appendix B.

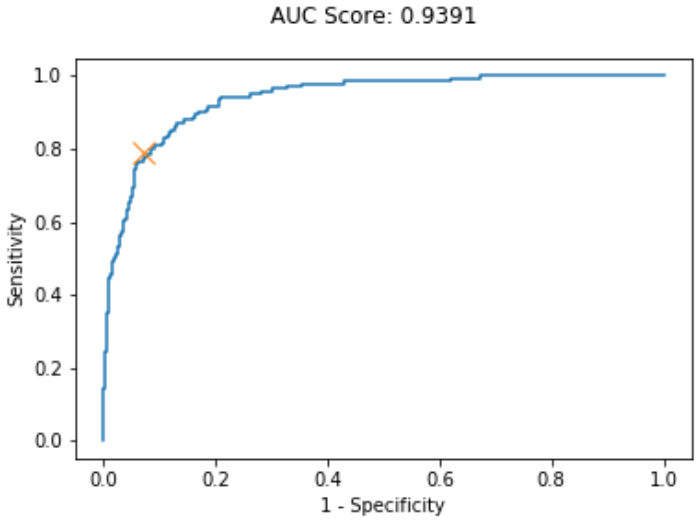


FIGURE 4.8: AUC curve of *EfficientNet-B3* (1024x1024) evaluated on the test set of Google NIH labels. The orange marker represents the average radiologist performance.

4.3.1 Visualization

Unfortunately, Majkowska et al., 2020 only depicts two images of a *SmoothGrad* visualization on the pneumothorax class. It is unknown to which patients these results belong to, otherwise we selected the same patients for visualization.

In Figure 4.10 we depict the Grad-CAMs of *DenseNet-121*, which has been trained at a resolution of 1024x1024. We choose the last convolution concatenation layer (*conv5 block16 concate*) to calculate our Grad-CAM derivations.

Subfigure 4.10a shows an example of a true-positive prediction. Here, a chest tube is present just below the localized Grad-CAM. Baltruschat et al., 2019 argues that the algorithm will learn the presence of a chest tube instead of looking at the pneumothorax. Luckily, the algorithm chooses the exact location of the pneumothorax over the tertiary signs of a pneumothorax such as the chest tube.

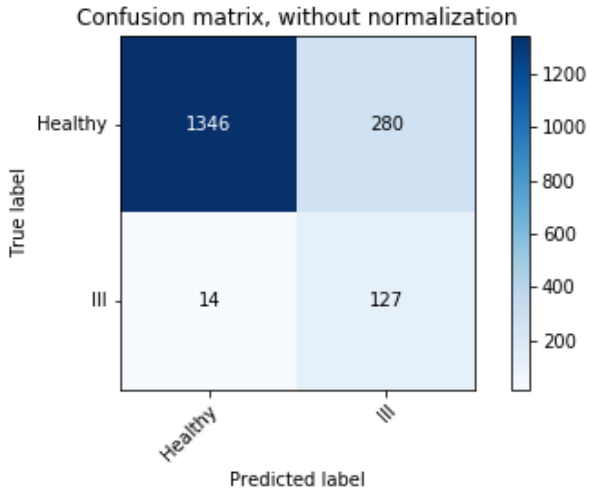
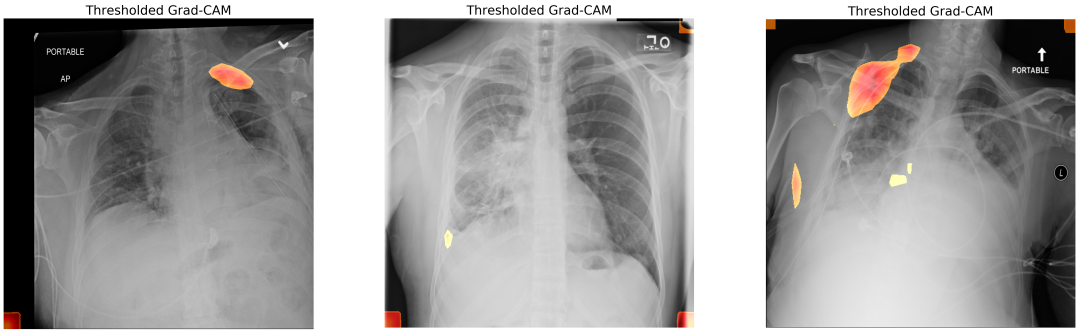


FIGURE 4.9: Confusion matrix of EfficientNet-B3 (1024x1024) on the test set of Google NIH.

We depict a false negative prediction in subfigure 4.10c. *DenseNet-121* (falsely) predicts the absence of a pneumothorax, with a pneumothorax probability of 7.61%. Though, the Grad-CAM shows activation around the area of the pneumothorax.



(A) True positive (99.7% pneumothorax predicted)

(B) False positive (63.44% pneumothorax predicted).

(C) False negative (7.61% pneumothorax predicted).

FIGURE 4.10: *DenseNet-121* (1024x1024) thresholded Grad-CAM examples of some evaluation images of the test set of Majkowska et al., 2020.

4.4 External data validation – A simulation

Because we want to achieve the best possible performance for the pneumothorax detection system, we train on all pneumothorax datasets that are publicly released. This way, we cannot do external data validation, which is an important part in verifying algorithm robustness. Therefore, we simulate external data validation by changing the datasets we train on. To do this, we first train on the *CheXpert* dataset, and evaluate results on the *SIIM* test split as a means of external data validation. We show the results in Table 4.5, and conclude that performance decreases when validating on data coming from other hospitals. We are able to restore performance by training on the distribution of the *SIIM* dataset. This generalization issue is also seen in related research of Taylor; Mielke, and Mongan, 2018 and Pan; Agarwal, and

Merck, 2019. As we want to implement a pneumothorax detection algorithm to our own hospital data, it is important that we explore this issue in depth.

Model	**Training data**	**F1**	**AUC**
DenseNet-121	CheXpert	*0,57*	*0,8602*
EfficientNet-B3	CheXpert	0,60	0,8169
ResNet-152	CheXpert	0,78	0,7875
DenseNet-121	CheXpert & SIIM	0,78	0,8894
EfficientNet-B3	CheXpert & SIIM	0,81	0,8987
ResNet-152	CheXpert & SIIM	0,81	0,9011

TABLE 4.5: External data validation experiment. Scores represent performance on the *SIIM* test split. Performance of all architectures increase when the training split of *SIIM* is used during training. This shows that the problem of domain adaptation is present in pneumothorax detection.

Chapter 5

Discussion

We present an approach to automated pneumothorax detection on chest radiographs using three publicly available datasets. Considering the problem setting, the datasets for pneumothorax detection contain biases such as chest tubes. However, those are the patients who had already been treated for pneumothorax, and therefore, detection of such cases are clinically not interesting. In order to reduce this bias in the datasets, we only make use of the first study of a patient assuming those patients are likely not to have chest tubes since it is their first study. In this way, we can ensure that images are split on patient level consistently, so no data leakage between the training and validation split takes place. In related work (Chapter 2), we see that not all research adheres to patient-level data splitting, or evaluating on public datasets. Using these practices, we achieve equal performance on the *ChestX-ray14* test set (0.936 AUC) compared to related work that additionally use private datasets (Majkowska et al., 2020). This proves that deciphering algorithm failures and dealing with biases improve performance by a great degree (Mahajan et al., 2020).

Next to presenting probability scores, we show thresholded Grad-CAMs in order to provide explainable results, which might assist radiologists with their decision making. Further, we demonstrate the importance of external data validation, and show that also our algorithm suffers from a decreased performance on external data due to the domain shift problem.

In terms of limitations, there are a couple of points of discussion which we need to address for this research. In particular:

1. NLP labels. Labels as acquired by NLP tools are not 100% correct.
2. Comparing against related work. Related research might not be transparent in their ways of comparing against related work.
3. Comparing against radiologists. The conditions in where an algorithm is compared against the performance of a radiologist might not represent real-world scenarios.
4. Presence of bias. Different types of bias is present in our datasets.
5. Additional pre-processing. Investigate the possibility of lung masking as an additional pre-processing step.
6. Robustness of results. Due to time limitations, we were not able to perform K-Fold Cross-Validation.

5.1 NLP labels

Datasets that use natural language processing (NLP) to analyze radiology reports are not 100% accurate. For *ChestX-ray14*, the labels are derived automatically from radiology reports using NLP. Their NLP tool achieves an accuracy of >90% on report parsing, so it can occur that the wrong labels are assigned. This phenomenon is a shortcoming of the public datasets, as making use of NLP labels gives a trade-off between quality and quantity of labels.

5.2 Comparing performance with related work

Most related work that evaluates its algorithms on the *ChestX-ray14* dataset use different partitions of the test split. The difference in test splits makes the comparison of different algorithms both difficult and unfair.

As an example, Majkowska et al., 2020 evaluates on 1.962 test samples (extra radiologist annotations, 195 pneumothorax samples, 0.94 AUC), while Baltruschat et al., 2019 evaluates on ∼22.400 test samples (∼1060 pneumothorax samples, 0.84 AUC). Finally, Rajpurkar et al., 2018 evaluates on a self-defined test split of *ChestX-ray14* consisting of 420 test images (45 containing a pneumothorax, 0.944 AUC). The main point here is that these three research papers all test on *ChestX-ray14*, but the number of images are all different, even though there exists an official test split.

For Rajpurkar et al., 2018, the self-defined test split is understandable. As this research compares results against the results of a radiologist, it does not make sense to let radiologists evaluate 1.962 images. It would cost a radiologist too much time to evaluate the official test split images, although this would strengthen the contribution towards *ChestX-ray14* even further.

5.3 Comparing against radiologists

Some research papers claim radiologist-level performance with their algorithm on some diseases. However, the methodology to compare with radiologists is different from real-world scenarios. In these cases, a radiologist is given only a frontal chest x-ray, without access to tertiary information such as patient history, visual assessment, or multiple imaging modalities. In real-world scenarios, a radiologist would consider all tertiary information in its assessment.

Rajpurkar et al., 2018 tries to compare performance against radiologists anyway, and found that radiologists achieve an AUC of 0.940 based on 45 pneumothorax images of a total of 420 images. The assessment takes radiologists an average of 240 minutes to complete, which is around 34 seconds to assess one image. A trained neural networks need just 34ms for an image to propagate through the neural network and achieves equal performance.

5.4 Bias

5.4.1 Prevalence

We do not make use of the original pneumothorax prevalence ratios during the evaluation of our algorithm. Therefore, the number of false-positives might be lower than in real-world scenarios. Park et al., 2019 argues that because the data has been

collected from multiple sources, there is a spectrum bias present. Here, specificity and sensitivity might differ because of different population/prevalence outcomes.

Images containing chest tubes, fractures, respiratory devices can be tertiary signs of the presence of a pneumothorax. The prevalence of pneumothorax in these patients is higher, as the patients have more complications in general.

5.4.2 Chest tubes

Baltruschat et al., 2019 shows that when training on *ChestX-ray14*, the active regions of Grad-CAM visualizations are located around the chest tube. This means that the network mainly used the chest tube to distinguish between pneumothorax / non-pneumothorax. In our case we do not experience this issue. Already during the preprocessing of the datasets, we make sure to minimize the number of chest tubes present by selecting only the first study of a patient. This might drastically reduce the number of images, but we get a cleaner dataset for it in return.

5.4.3 AP / PA ratio

This bias can also be seen in terms of the number of AP versus PA images. For the *CheXpert* dataset, 89% of the frontal pneumothorax images are AP. For other datasets (*SIIM*, *MIMIC-CXR v1.0*) the ratio AP / PA remains unknown.

5.5 Additional pre-processing

As an additional preprocessing step, we tried to include lung masking (Appendix A.1.2). This step supposedly increases signal-to-noise ratios. However, we observed that lung masking did not seem to improve performance for the pneumothorax detection task. This can be due to the fact that the lung segmentation algorithm sometimes removes the *deep sulcus sign*, which is still part of the lung. The heart is also removed from the segmentation, making another symptom (mediastinal shift) invisible to the algorithm.

5.6 Future work

5.6.1 Robustness of results

To strengthen our internal results, we can compute an average of trained networks on all the datasets using K-Fold cross-validation. However, training a network on all available training data takes a week to train. Due to time limitations, we were not able to wait an additional five weeks for 5-fold cross-validation to finish. It is interesting to see whether there will be a performance difference across network architectures, which due to variance is something we did not yet find viable to explore.

5.6.2 Leveraging segmentation labels

The Grad-CAM localizations are not useful for pneumothorax segmentation, as we see that these can be located around other signs related to a pneumothorax (e.g. *deep sulcus sign*). Making use of the segmentation labels of the *SIIM* dataset might improve the localization output of the network. We can incorporate the segmentation labels by adding a dedicated segmentation (U-Net) pipeline after the classification phase.

Part II

Domain Adaptation

Chapter 6

Introduction

In research, we often assume that our test data exactly matches the data distribution and biases of our training data. We validate the models on internal data splits, which are a direct derivation of the training sets. As a result, the test set contains the same biases and prevalence rates as present in the training sets. Most research assess the performance of their algorithm on datasets which it is trained on. However, in practice we see that models sometimes fail to perform within the performance boundaries as seen during model development. This can be attributed to the fact that data from a different hospital has different internal biases and methods of acquisition. This phenomenon is called *domain shift*, or *covariate shift* (Shimodaira, 2000). The introduced external dataset might contain a different representation of the same type of data, which makes internal features not compliant with the dataset of interest.

In clinical algorithms, we see this phenomenon occurring with data coming from different hospitals. It is often the case that algorithms are developed with data from *hospital 1*, but is also put into practice in *hospital 2*. Suddenly, the algorithm does not give confident predictions anymore and generally performs worse. We also observe domain adaptation issues in our preliminary results for simulated external data validation, as seen in Chapter 4.4.

The change in domains might not always be attributed to the difference in data, but also due to the difference in label determination. Cohen et al., 2020 shows evidence that the generalization of x-ray images is possibly not caused by the domain shift of the images itself, but by the introduced uncertainties in the abnormality labels. This label shift can be caused by errors in the NLP tools or uncertainty among radiologists. The research confirms this is also the case for pneumothorax labels in multiple datasets: *Chexpert*, *ChestXray-14* (Wang et al., 2017), *ChestXray-14* (Majkowska et al., 2020), and *Open-I*. There is a small inter-rater agreement of the algorithms between datasets, indicating that there is a probable label shift.

6.1 Research Question

In order to overcome the *domain shift* phenomenon, we develop a method that can learn the characteristics of a new dataset without the need of having image labels beforehand. Thus, an unsupervised method to achieve domain adaptation. The different approaches towards domain adaptation are described in Chapter 7. Formally speaking, we propose a method that belongs to the family of inference-based methods (Chapter 7.4). To test our new method, we simulate a domain adaptation on public datasets, so that our results are reproducible. To achieve domain adaptation, we will adapt domains from the *CheXpert* dataset towards the *SIIM* dataset using

the *pneumothorax detection* task.

We extend the research in inference-methods by combining methods of *mean-teacher* and *self-training*. Xie et al., 2020 proposes a similar method, and shows that an iterative mean-teacher approach can be used to improve source domain performance in a semi-supervised setting. In our case, we are interested in improving the target domain performance in unsupervised setting. This leads to the following research question:

> Can we adapt the *iterative self-training* concept to achieve unsupervised domain adaptation?

Although our current experiment setup consists of a labeled target dataset (*SIIM*), we simulate this dataset to be unlabeled, and use the existing labels as a check to examine how certain decisions (e.g., thresholds) lead to a particular set of soft-labels. Furthermore, we explore potential improvements to *iterative self-training* such as *soft-label multiplication* and *model ensembling*. These methods are needed, as the size of our target dataset is limited compared to original research (10.000 images vs 300.000.000 images).

Soft-label multiplication Our target dataset is much smaller compared to the dataset of Xie et al., 2020 (10.000 images vs 300.000.000 images). Xie et al., 2020 has a lot more soft-labels to consider during his self-training approach. In order to still leverage our limited dataset size, we propose a multiplication factor to the samples acquired by *self-training*. This factor determines the oversampling rate for soft-labels.

Model ensembling Dupre et al., 2019 shows that model ensembling techniques can help to provide more confident predictions during *iterative self-training* to add to the training set. Therefore, we propose an ensemble consisting of *ResNets*, *EfficientNets*, and *DenseNets*.

To show that this method works by walking through the following steps:

1. State the max achievable performance of pneumothorax detection by training on both *CheXpert* and *SIIM* dataset (complete supervised learning).
2. State the baseline performance of a trained model on *CheXpert* and test external data validation by evaluating on the unseen *SIIM* dataset.
3. We perform our domain adaptation method: *iterative self-training*, and show intermediate results.

Chapter 7

Related work

During model development, there are several ways to overcome domain adaptation issues. We can look at the way the input is presented, how the model is handling the inputs from another domain, or try to improve the robustness of the model itself. In general, there are variety of techniques that deal with domain shift (i.e. domain transformation). Wang and Deng, 2018 and Kouw and Loog, 2019 provides a survey of deep domain adaptation methods, which we use as a template to overview the most important methods:

1. Sample-based methods
2. Input image transformations
3. Feature space transformations
4. Inference-based methods

7.1 Sample-based methods

Sample-based domain adaptation methods tackle the problem of data bias. While sampling from a dataset distribution, it is important that each data split resulting from the dataset consists of the same biases. A good example is by having data from multiple hospitals during training (Kouw and Loog, 2019). If the data splits are biased, the internal distribution of bias is also skewed, leading to *covariate shift* or *concept (label) shift* (Cohen et al., 2020). Two generally used methods:

1. Data importance-weighting – Weight samples by the attributes of the data (e.g. age or gender distribution).
2. Class importance-weighting – Weight samples by looking at label distribution.

7.2 Input image transformations

We can pre-process/modify our input images in such a way, that it represents the target domain. Popular examples are *Autoencoders* and *Generative adversarial networks*.

Generative Adversarial Networks (GANs) GANs can generate images by transposing images from one domain to another. It is also applied to the field of medical imaging, as differences in data distributions could be caused by differences in imaging standards (Liu and Tuzel, 2016) (Ganin et al., 2016) (Kamnitsas et al., 2017)

(Hoffman et al., 2018) (Bousmalis et al., 2017). One example of a GAN is the *Domain-Adversarial Neural Network* (DANN) (Ganin et al., 2016). DANN produces a representation such that a generated image for that domain cannot be distinguished from the rest of corresponding data distributions.

7.3 Feature space transformation

Feature space transformations are methods that reshape the feature space so that a classifier trained on transformed source data will generalize to target data (Bousmalis et al., 2016). Some methods, as described in Kouw and Loog, 2019:

1. Subspace mappings. Provide derivations of how data is mapped to other domains.
2. Optimal transportation techniques.
3. Learning domain-invariant representations. This method helps to get rid of domains in general, by learning only the core representations.

d-SNE d-SNE is a latent-space transformation algorithm requiring few labels for training towards a new domain. For each class, they calculate a distance measure between domains. The task of the loss function is then to minimize pair-wise distance within classes while maximizing the pair-wise distance between classes (Xu et al., 2019). This work is extended to semi-supervised learning by applying a mean-teacher approach (Tarvainen and Valpola, 2017) as well.

CyCADA Hoffman et al., 2018 is a method which uses both input image transformations and feature space transformations. Feature space transformations are difficult to interpret and seem to misjudge pixel-level translations. On the other hand, input image transformations "fail to incorporate high-level semantic knowledge relevant for the end task".

7.4 Inference-based methods

Inference-based methods overcome the domain adaptation problem by implementing adaptation methods during the inference procedure. This can be achieved by reformulating the optimization procedure or apply constraints based on the target dataset (Kouw and Loog, 2019). There are different inference-based methods that use unsupervised or weakly supervised labels to capture additional knowledge of the target domain. Two methods are *co-training* (Blum and Mitchell, 1998) (Nigam and Ghani, 2000) and *self-training* (Rosenberg; Hebert, and Schneiderman, 2005).

7.4.1 Co-training

During *co-training*, we train multiple algorithms on different features. When combining the algorithms, we get more robust results as we have trained our algorithm on multiple features for one dataset. The *co-training* method of Chen; Weinberger, and Blitzer, 2011 does not train multiple algorithms on different features but formulates a single optimization problem for all algorithms instead. *Self-training*, on the other hand, does not use a split of features. Compared to *co-training*, *self-training*

uses just one model. This method adds the most confident samples to its training set by assuming those are the correct labels.

7.4.2 Self-training

Self-training is a relatively old method (Yarowsky, 1995) that is still relevant and applied in modern techniques like neural networks. Specifically, *self-training* is a method where an existing algorithm trained on all labelled data makes predictions on unseen (unlabeled) data of a different distribution. If the algorithm is confident enough of its predictions (e.g., soft-max prediction exceeds a defined threshold), we assume this is the correct label for that particular sample and add this sample to our training set. Pérez and Sánchez-Montañés, 2007 extends the Expectation-Maximization algorithm to cope with *concept drift*, meaning that statistical distributions for each domain are different. They prove that it works for data coming from different hospitals.

7.4.3 Mean-Teacher

Tarvainen and Valpola, 2017 shows a method that penalizes predictions that are inconsistent with the target. *Mean-Teacher* is a method that averages model weights instead of changing label soft-max predictions as during *self-training*.

Mean-Teacher is a teacher-student model that provides a training method on a derivation of the data. The teacher learns on *hard* labels (e.g. binary labels) as input, and gives class probabilities as output. The consecutive student network then trains its network based on the output of the teacher model. Because these labels are probabilities instead of hard labels, these predictions provide a continuous label regarding the information of the image.

7.4.4 Self-training mean-teacher

Xie et al., 2020 uses a student-teacher model to create a method that achieves state-of-the-art on the *ImageNet* classification task. In this paper, a teacher model is trained on labelled images and then used to generate pseudo labels on unlabeled data. Then, a new student model is trained on the combination of labelled and pseudo-labelled images. This loop is iterated over by making the trained student the teacher and repeating the steps until convergence. This method makes use of unlabeled external data to strengthen performance on the labelled (internal) dataset. Therefore, this is a semi-supervised method (makes use of both labelled and unlabelled images). What is novel about this method is that they add several techniques to improve knowledge extraction from the external unlabeled dataset:

1. Input to the student network is noised using dropout, data augmentation, and stochastic depth.

2. Student model should be larger than the teacher model.

3. Use soft labels instead of binary (hard) labels, as these tend to converge faster and lead to better stability.

As most student-teacher models are used for model compression, this approach is precisely the opposite. They scale up network architectures beyond *EfficientNet-B7*, generating their own EfficientNet-L0, L1, L2. The training time of EfficientNet-L2 is five times the training time of EfficientNet-B7. Complete training of this model

takes 3.5 days on a Cloud TPU of 2048 cores. We can argue that this limits verifiability, as only a few have access to these type of resources. The fact that they made their model larger only contributed to a 0.5% increase in performance, while the *Noisy Student* approach reported a gain of 1.9%.

Sun et al., 2019 proposes methods that try to learn on self-generated labels on a target dataset, assuming that this dataset will generalize better. Related work on this is that they try to transform the source dataset into the target dataset by using generative models (Taigman et al., 2016), or another method, where pseudo-labels are generated based on the trained model of the source dataset. When the prediction of the pseudo-labels reaches a certain threshold, we add the pseud-label to the training set (bootstrapping with unlabeled target data). This method is also called self-ensembling French; Mackiewicz, and Fisher, 2017 or co-training.

7.4.5 Consistency training

Laine and Aila, 2016 introduces *consistency training*, which provides methods such that models robust and insensitive towards input noise (i.e. adversarial attacks). These methods show that using unlabeled data to address input noise issues improves adversarial robustness. However, we do not use *consistency training* methods during self-training, as using these methods to achieve regularization prevents having good results on the target dataset (Xie et al., 2020).

Chapter 8

Methods – Iterative self-training

The iterative self-training pipeline by Xie et al., 2020 is originally formed to improve performance on the source dataset. However, we want to perform well on the target domain, and therefore define a slightly different task structure. We state the definition our our pipeline below.

Require:
Labeled images – $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$
Unlabeled images – $\{\tilde{x_1}, \tilde{x_2}, ..., \tilde{x_n}\}$
Cross entropy loss function – $\mathcal{L}$
Confidence threshold – τ

1. Train the *teacher* model ensemble θ^t on the source dataset n by minimizing the cross-entropy loss.

$$\frac{1}{n}\sum_{i=1}^{n} l(y_i, f^{noised}(x_i, \theta^t))$$

2. Using the trained *teacher* model ensemble θ^t, generate soft-labels on the target dataset m.

$$\forall \theta^i \in \theta^t, \forall i \in m, \quad y\tilde{}_{\theta_i} = f(\tilde{x_i}, \theta^i)$$

3. Calculate the mean prediction scores and add the most confident predictions using threshold τ to the training dataset. Optionally: oversample the soft-labels with a multiplication factor.

$$[\frac{1}{\theta^t}\sum_{i=\theta_i}^{\theta^t} \tilde{y}_{\theta_i}] > \tau$$

4. Create a new model ensemble with increased architectural capacity (so the model ensemble has increased capacity to capture nuances), which we call the *student* model ensemble.

5. Train the *student* model ensemble on the source dataset + (oversampled) target soft-labels and apply heavy data augmentation (noise). Since we do not have any labels from the target domain in unsupervised domain adaptation setting, our validation set only consists of the source dataset and no soft-labels.

$$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(y_i, f^{noised}(x_i, \theta^t)) + \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\tilde{y_i}, f^{noised}(\tilde{x_i}, \theta^t))$$

6. The trained *student* model ensemble θ^s becomes the *teacher* model ensemble θ^t.

7. Go to step 2. Iterate until convergence.

In the following sections, we will explain how to achieve domain adaptation for our *CheXpert* $\rightarrow$ *SIIM* experiment.

8.1 Teacher: Generating soft-labels

To compute soft-labels, we use an ensemble of three neural network teachers: *ResNet-152* (Chapter 3.3.1), *DenseNet-121* (Chapter 3.3.2) and *EfficientNet-B3* (Chapter 3.3.3). Pérez and Sánchez-Montañés, 2007 shows that an ensemble can lead to better confidence scores. We re-train these networks on the *CheXpert* dataset after we have initialized them with *ImageNet* weights (pre-training). Then, we let the networks generate predictions for the training set labels of *SIIM*. For each image in the *SIIM* training set, we compute a mean score of all three networks in order to aggregate the scores into a single prediction estimate. We call this score our soft-label. Each image of the *SIIM* training set now has a soft-label prediction. Generally speaking, when the soft-label prediction is above 0.5, the network classifies that particular image as containing a pneumothorax (positive class). However, a prediction score of around 0.5 is not a confident prediction value. The higher this score, the more confident the networks are in their prediction of the positive class (pneumothorax). We need to determine a threshold value in such a way that we get as many confident true-positive pneumothorax predictions as possible while minimizing the number of false-positive pneumothorax predictions. In related semi-supervised work (Dupre et al., 2019), a threshold value is chosen such that 99% of the included data is *correctly* labeled. Unfortunately, due to our method being unsupervised, we cannot choose a threshold this way. We do not have the labels of our target dataset to determine how many soft-labels are correctly labeled. However, the assumption that the threshold value is a trade-off between correctness of the labels (quality) and amount of labels (quantity) still holds.

For our experiment, we consider the labels of *SIIM* as a verification. When we use a threshold of 99% true-positive predictions, we have very few samples for each corresponding class: less than 50! Compared to the entire *CheXpert* training set (10.836), the number of added soft-labels given this threshold will probably not affect the network during training. We can thus say that the threshold value is a trade-off between the quality & quantity of soft-labels.

For the non-pneumothorax soft-labels, the criteria for choosing a threshold is different. We need to choose a threshold in such a way that we achieve the same number of non-pneumothorax cases as pneumothorax cases, in order to maintain an equal class-balance. Because the *SIIM* training set has a relatively low prevalence for the pneumothorax class, the non-pneumothorax class gives us a lot more images when using the same threshold as the pneumothorax class.

Label presence (simulated) Generally speaking, during unsupervised learning, we would not know the amount of incorrectly classified soft-labels, because we do

not have labels of this dataset. However, in this research setting, we do have the actual labels of the dataset, but only use these labels to assess the quality of results. Because these labels are present, we can inspect what percentage of predicted soft-labels are correct. To illustrate the original class-imbalance, we plot the ensemble (mean) thresholds against the number of obtained predictions in Figure 8.1. In this figure, we see that a higher threshold value results in a more confident prediction due to the number of false-positive predictions decreasing, and fewer soft-labels are returned. By increasing the threshold, we basically "filter out" images that do not represent a pneumothorax. At a base of the x-axis (threshold = 0.0) we clearly see the class imbalance: 6.000 *non-pneumothorax*, 1.725 *pneumothorax*[1].

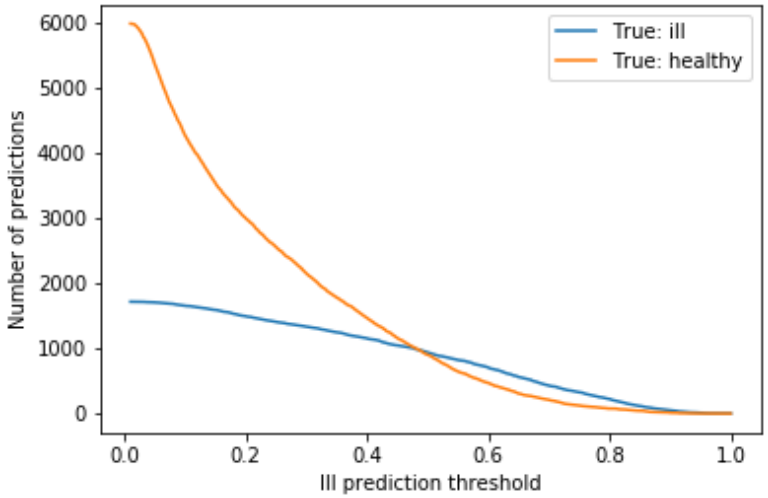


FIGURE 8.1: Positive class soft-labels as generated by taking the mean value of the ensemble predictions on the SIIM dataset. True positives in blue, false positives in orange. We need to choose a (lower-bound) threshold value for our soft-label inclusion.

8.1.1 Soft-labels: *pneumothorax*

When we look at Figure 8.2, which is a magnified version of Figure 8.1, we see the number of *non-pneumothorax* (i.e. 'healthy') images dropping below the number of *pneumothorax* (i.e. 'ill') images starting at a threshold of 0.5. Ideally, we want to maximize the gap between *non-pneumothorax* and *pneumothorax* depending on the threshold, but by still taking into account the false-positive rate.

For the first iteration we choose a threshold value of 0.85, which gives us 715 *pneumothorax* soft-labels (155 false-positives, FPR of 21.67%) [2]. This way, we have a significant quantity of soft-labels while keeping the false-positives to a minimum.

8.1.2 Soft-labels: *non-pneumothorax*

For the *non-pneumothorax* class, we also select a threshold value. This class returns less false predictions compared to the *pneumothorax* class, as the prevalence rate for

[1] The numbers are different from Table 3.1, as these numbers represent the (stratified) *SIIM* training split

[2] In true unsupervised setting we would not know the amount of false-positives, and therefore choose a higher threshold value such as 0.90.

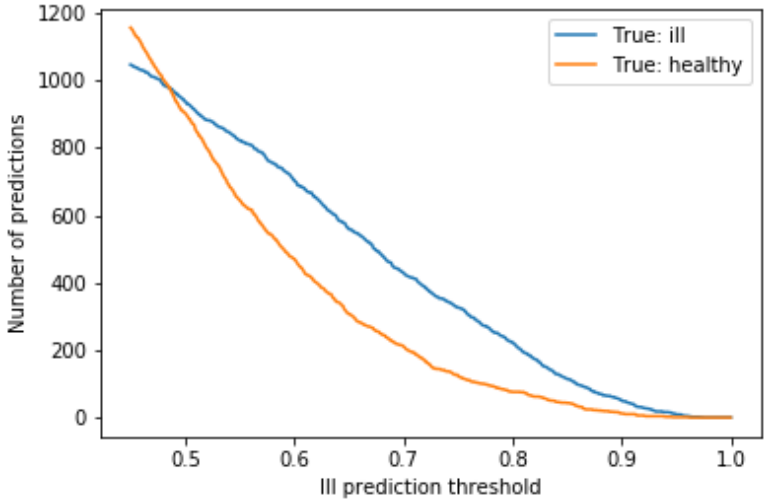


FIGURE 8.2: Zoomed in version of Figure 8.1. We choose a threshold such that we have a maximum of *True: ill*, but a minimum of *True: healthy*.

non-pneumothorax images is high. Therefore, we choose a really confident threshold of 0.9 (translated to 0.1 in terms of upper-bound threshold). This results in 713 *non-pneumothorax* soft-labels (4 false-negatives, FNR of 0.5%).

8.2 Training the *Student* model

Now that we have soft-labels as generated by the *teacher* model ensemble, we can start training with our additional soft-labels from our target domain. Following the steps, we need to create a *student* model ensemble and start training on the soft-labels. Xie et al., 2020 argues that a larger model is needed for the *student* model in order to learn a more powerful model. Because we run a model ensemble, we run these steps for each of our network architectures:

EfficientNets For *EfficientNet*, we start with *EfficientNet-B3* with a starting dropout rate of 0.3 and stochastic depth with a survival rate of 0.8. For each iteration, we increase the network size (*EfficientNet-B4*, *EfficientNet-B5*, *EfficientNet-B6*) and the dropout rate.

DenseNets We start our iteration with *DenseNet-121*, and scale up to *DenseNet-169*. Due to GPU limitations, we are not able to scale up to *DenseNet-201*. This gives us effectively two *DenseNet* architectures to iterate with.

ResNets During experimentation, we had issues with the *ResNet* networks, as GPU errors kept arising. Increasing network capacity was an issue, and batch sizes were minimal. Although we could not scale our *ResNets*, we still provide preliminary results and use this network in our ensemble.

We re-initialize each network of the ensemble with *ImageNet* weights and retrain the models. We use the same approach as mentioned in Appendix A. The only difference

is that during the creation of the training & validation split (80% train, 10% validation, 10% test), we only add the generated soft-labels to our training set. By leaving the validation set intact, we guarantee equal performance on our source dataset. We apply heavy data augmentation during training to make sure each epoch contain different images.

Soft-label multiplication / oversampling Because we use such heavy data augmentation, we can duplicate our soft-labels to increase the presence of the target dataset even further (oversampling). Even when multiplying the soft-labels, data augmentation makes sure each epoch contains a unique derivation of an image. Depending on the network architecture, we found multiplication rates between x6 - x15 to work best.

Chapter 9

Results

In this chapter, we present the results of our *iterative self-training* approach on domain adaptation by adapting domains from the *CheXpert* dataset towards the *SIIM* dataset. We generate soft-labels by making use of the model ensemble but still track performances of each network architecture individually.

For each network architecture, we present the max achievable performance and the results of *iterative self-training*. The max achievable performance is the performance when the labels of the target dataset are known (regular supervised learning). To attain the max achievable performance, we train on both *CheXpert* and *SIIM* dataset (complete supervised learning). After training on both the source and target dataset, we can assume that the performance of *iterative self-training* will never transcend the max achievable performance. The results of *iterative self-training* are displayed per iteration so we can gauge when the network has fully converged.

9.1 ResNet

The max achievable performance (by training on both *CheXpert* and *SIIM*) is 0,81 weighted F1 score, and an AUC score of 0,9011. In table 9.1 we show the performance of *iterative self-training* on a *ResNet* backbone [1]. The network seems to be robust, as the baseline scores come close to the max achievable performance. The best model in terms of AUC score is seen during iteration 1, where the *teacher* model equals the *student* model, but additional soft-labels are added during training of the network. This model seems to overfit after iteration 2, as the F1 and AUC score on the test set drops significantly. We can conclude that *iterative* self-training might not work for this architecture.

#	Model	Soft-labels	F1	AUC
0	*ResNet-152*	*None (baseline)*	*0,78*	*0,7875*
1	ResNet-152	x6 (500/class)	0,76	**0,9038**
2	ResNet-152	x10 (800/class)	**0,78**	0,8867
3	ResNet-152	x10 (960/class)	0,67*	0,632

TABLE 9.1: ResNet-152 performance on SIIM iterative self-training.
* best model on epoch 1..

[1] For this network we cannot increase network capacity to bigger networks due to GPU memory limitations

9.2 EfficientNet

The best model performance of *EfficientNet-B3* by training on both *CheXpert* and *SIIM* results in a weighted F1 score of 0,81 and an AUC score of 0,8987. We show the results for the *iterative self-training* of the *EfficientNet* backbone in Table 9.2. At iteration 3 we achieve best AUC score, where as iteration 2 shows the best F1 score.

#	Model	Soft-labels	F1	AUC
0	*EfficientNet-B3*	*None (baseline)*	*0,60*	*0,8168*
1	EfficientNet-B4	10x (500/class)	0,72	0,8570
2	EfficientNet-B5	15x (800/class)	**0,79**	0,8873
3	EfficientNet-B6	15x (960/class)	0,67	**0,9008**

TABLE 9.2: EfficientNet performance on SIIM iterative self-training

After *iterative self-training*, the true positives changed from 214 to 226, having a difference of 19 more compared to the baseline. For the number of false positives, this resulted in a decreased from 436 to 367. 161 cases are now correctly classified as negative.

9.2.1 Visualization

9.2.1.1 True positives

We show an image (Figure 9.1 & 9.2 where the network predicts a false-negative (40.82% pneumothorax probability), which eventually turns into a true positive (89.53% pneumothorax probability). The localization of the pneumothorax is also improved.

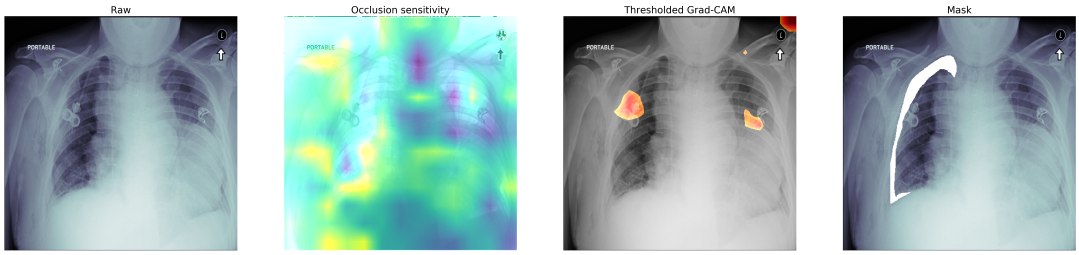


FIGURE 9.1: *EfficientNet-B3* baseline. A pneumothorax is present, but the network predicts otherwise (predicted pneumothorax of 40.82%).

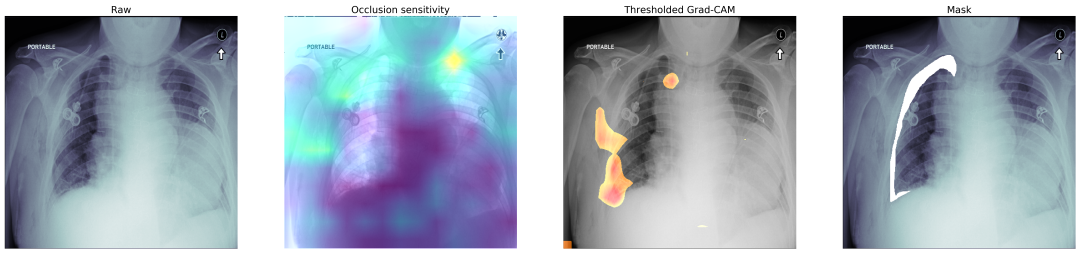


FIGURE 9.2: *EfficientNet-B6* final iteration. After iterative self-training the pneumothorax probability increased to 89.53%.

9.2.1.2 False positives

Figure 9.3 depicts a false positive classification where the location of the Grad-CAM show activation in the left lung. The pneumothorax probability is 51.05% before *iterative self-training*. Afterwards, the pneumothorax probability decreases to 7.13%, and the Grad-CAM activation in the left lung completely disappears (Figure 9.4).

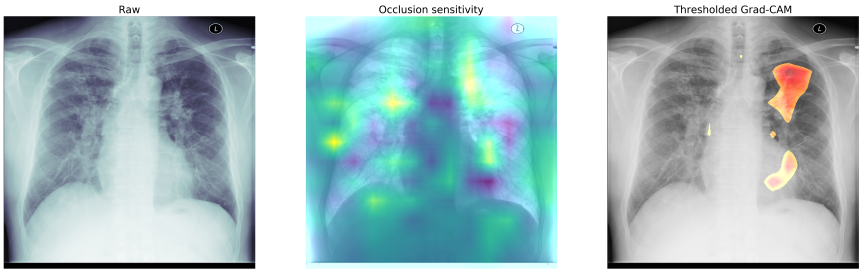


FIGURE 9.3: *EfficientNet-B3* baseline. The network incorrectly predicts this image as containing a pneumothorax. The pneumothorax probability is 51.08%., which indicates low confidence of this prediction.

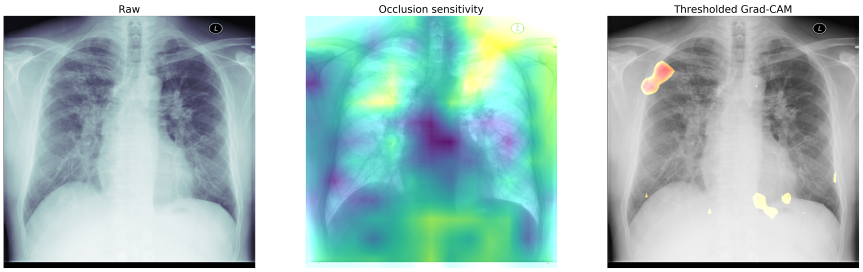


FIGURE 9.4: *EfficientNet-B6* final iteration. For this patient without a pneumothorax, the pneumothorax probability successfully decreased to 7.13%.

9.3 DenseNet

By training on both *CheXpert* and *SIIM*, *DenseNet-121* achieves a maximum weighted F1 score of 0,78 and an AUC of 0,8894. After *iterative self-training*, we manage to decrease the number of false positives from 485 to 291. The true positives increased from 219 to 226. The results of *iterative self-training* For *DenseNets* can be seen in Table 9.3.

#	Model	Soft-labels	F1	AUC
0	*DenseNet-121*	*None (baseline)*	*0,57*	*0,8602*
1	DenseNet-121	6x (500/class)	0,72	0,8885
2	DenseNet-169	10x (800/class)	0,73	0,8928
3	DenseNet-169	10x (960/class)	**0,75**	**0,8982**

TABLE 9.3: DenseNet performance on SIIM iterative self-training

9.3.1 Visualization

9.3.1.1 True positives

Figures 9.5 & 9.6 show the before and after of *iterative self-training* for *DenseNet-121* towards *DenseNet-169*. The pneumothorax probability increases from 38.17% to 93.01%. After the iterations, the Grad-CAM covers a larger area of the masked pneumothorax compared to the baseline.

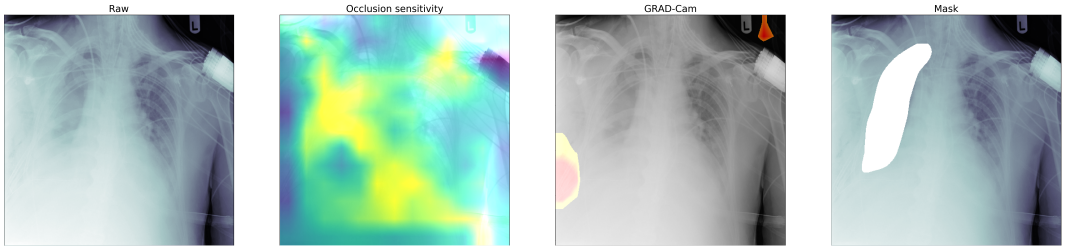


FIGURE 9.5: *DenseNet-121* baseline. Predicted pneumothorax of 38.17%.

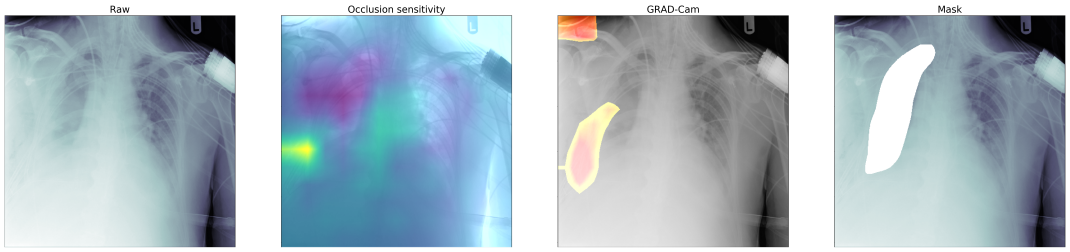


FIGURE 9.6: *DenseNet-169* final iteration. Predicted pneumothorax of 93.01%.

9.3.1.2 False positives

The number of false positives decreased by 40% (485 to 291). In Figure 9.7 we see an image falsely classified of containing a pneumothorax (77.80% pneumothorax probability) in the top-right lung. After *iterative self-training*, the pneumothorax probability decreases to 24.08%. The final Grad-CAM as depicted in Figure 9.8 no longer shows activation in the top right lung only.

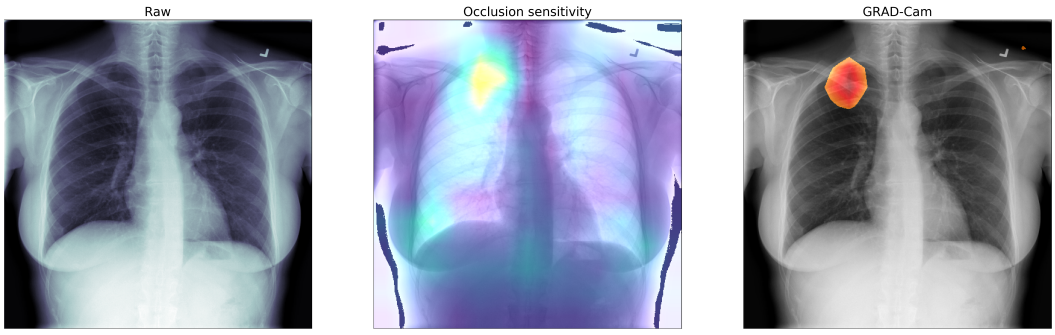


FIGURE 9.7: *DenseNet-121* baseline false-positive. Predicted pneumothorax of 77.8%.

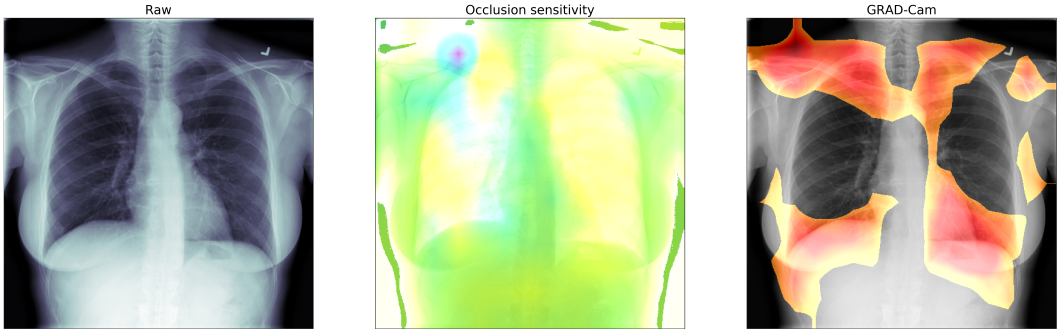


FIGURE 9.8: *DenseNet-169* final iteration false-positive. Pneumothorax prediction decreased to 24.08%.

Chapter 10

Discussion

Our adapted *iterative self-training* method shows promising results. For the *EfficientNet* architecture, were all the prerequisite experiment criteria are met, we are able to improve the performance on the target dataset (0,60 → 0,79 weighted F1; 0,8602 → 0,8982 AUC).

- *EfficientNets* seem to overfit at iteration 3. Improvement until iteration 2: 0,60 → 0,79 weighted F1; 0,8168 → 0,8873 AUC.
- *DenseNets* keep improving even at iteration 3. Improvement until last iteration: 0,57 → 0,74 weighted F1; 0,8602 → 0,8982 AUC.
- *ResNet-152* does not have the capacity to learn the target domain soft-labels during iterations. The best performance is achieved after the first iteration, when the initial soft-labels are added to the training set: 0,78 → 0,76 weighted F1; 0,7875 → 0,9038 AUC. Consequent iterations seem to decrease this performance.

As we see, *EfficientNets* converge faster using *iterative self-training* compared to *DenseNets*. This can be attributed to the fact that the research setup is not precisely the same. We are not able to directly compare iterations as the different architectures do not scale up their size equally. We also see that we achieve the largest improvements in terms of performance during the first iteration of *iterative self-training*. From this, we can conclude that adding soft-labels of the target domain to the training of our source domain without any iterations already helps. Yet, we run into some limitations during our experiments:

1. Determine soft-label thresholds. For now, we explore a single threshold value per iteration.
2. Minibatch label distribution. We do not ensure a correct distribution of training labels & soft-labels in each minibatch during training.
3. Network scaling. Not every architecture scales as much as we want to. We should consider re-running the experiments by keeping in mind network architectures with their GPU limitations.

Model ensemble During experimentation, we see that the calculated mean of three network predictions provide better soft-labels in terms of the number of false-positives & false-negatives compared to individual network soft-labels. Although we do not verify these results by comparing performances on the complete pipeline, this would be a good ablation study for future work.

Soft-label thresholds We currently determine soft-label thresholds by looking at the false-positive rate. In true unsupervised setting, this number is not available (therefore we only use it for verification!). The general thought is that when a higher threshold is chosen, the more confident the predictions are. For future work, we might experiment with this threshold value in order to determine the quality – quantity trade-off between soft-labels. A lower soft-label threshold might result in a larger quantity of soft-labels, but comes with a higher false-positive rate.

Minibatch label distribution For this approach to work effectively, we have to make sure that each batch consists of an equal contribution of both the source data & target soft-labels. However, due to our small batch sizes, we cannot guarantee this assumption.

Network architectures We are not able to scale up the *ResNet-152* architecture due to GPU limitations. One could argue that *ResNet-152* as a baseline is too high to start with. However, *ResNets* started to give good results on the original training set, starting with a network capacity of *ResNet-152*. Other research might achieve great results using this architecture, but they use lower input resolutions.

Robustness of results Just as in part I (Chapter 1.1), we might apply cross-validation to verify results and possibly highlight differences in network architectures.

Part III

Clinical validation

Chapter 11

Introduction

In this part of the thesis we verify our findings against a local private dataset (*RadboudCXR*, Chapter 12.1). We combine the knowledge and methods as seen in previous chapters and evaluate their use case in a true clinical setting. We hope that by using our pneumothorax detection knowledge & *iterative self-training* method, we can successfully push the performance of our pneumothorax detection algorithm even further. This leads to the following research question:

> Are we able to verify our *iterative self-training* domain adaptation method on a local (private) dataset with different biases?

To test this question, we start our *iterative self-training* method using our trained models of Part I (Chapter 4). These models are trained on the public datasets *SIIM*, *CheXpert*, and *MIMIC-CXR v1.0*. The target dataset for unsupervised domain adaptation is *RadboudCXR*. We carefully sample & verify a held-out test set in order to compare our results.

Chapter 12

Methods

We first dive into the clinical dataset and investigate the limitations of this dataset. Because this dataset is very noisy, We manually define a held-out test set that is of high-quality standards, and that does not include chest tubes or invalid images.

After we explore the clinical dataset, we run the *iterative self-training* pipeline and investigate the effects of choosing a certain *multiplication factor*.

12.1 RadboudCXR

The dataset *RadboudCXR* is an internal private dataset from the Radboudumc hospital. The chest x-rays come with attached (unstructured) radiology reports and are written in Dutch. At the time of writing there are 281.061 chest x-ray reports available. Unfortunately, the NLP tools used by other datasets such as *CheXpert labeller* cannot be used, as these tools do not support the Dutch language. Therefore we develop our own rule-based NLP tool to extract *pneumothorax* and *non-pneumothorax* images from the dataset. Because the radiology reports do not have a standardized format, our NLP tool is prone to errors.

Using our NLP tool, we are able to extract 20.212 reports mentioning a positive pneumothorax. When we exclude thorax drains (filter for "drain", "tube"), we are left with 6.500 frontal pneumothorax images. 95% of these images have an indicated AP view position, meaning that most of the images are obtained via inpatient clinic.

For non-pneumothorax images, we randomly sample until we reach the same prevalence rate ($\sim$25%) as our other preprocessed datasets. We completely filter out thorax drains and the word "pneumothorax". Eventually, 21.707 non-pneumothorax images are present.

RadboudCXR contains a lot of mislabeled chest x-rays. A good example is the selection of frontal (PA/AP) images. When we filter for frontal chest x-rays, upon inspection, some images show lateral (side view) images. The trained networks have never seen these types of images before and have a hard time classifying these samples. *RadboudCXR* also contain many x-rays of babies/children. Because these children are small, their "chest" x-ray shows their complete body. We want to improve the quality of the internal dataset and do this by setting up criteria that excludes these types of cases. We cannot verify all these images by hand. Therefore we randomly select 500 chest x-rays for our verified test set.

12.1.1 Verified Test set

To be sure we have a good standard to measure our results, we randomly sample a test set of around 500 images from *RadboudCXR*. Then, we inspect each file and belonging text report to see whether they meet quality standards (see Chapter 12.1.2). Because not all sampled images are suitable to include in our test set,

we keep drawing samples until a total of 504 images is achieved (259 pneumothorax, 245 non-pneumothorax). Also, we make sure to have a balanced distribution of view positions; 47% of all test images are AP view.

During our experiments, we treat these images as a held-out test set, so we do not use these images for training or validation.

12.1.2 Sampling criteria

The test set should be of high quality, and therefore should meet quality standards. To determine the difference between a good sample and a bad sample, we look at both the image and the text report.

12.1.2.1 Text report criteria

For the pneumothorax class, the text report should clearly mention that there is a pneumothorax found, and there should be no mention of a thorax drain. The pneumothorax class therefore also contains patients with other comorbidities other than a pneumothorax. Sample of a report that meets the guidelines:

> X-thorax op IC AP: vergelijk onderzoek van eerder XXX.
>
> Bekende pneumothorax rechts, schildikte maximaal ongeveer 24mm. Bekende versterkte intrapulmonale tekening. Longvaattekening lijkt binnen de norm. Bronchopathie. Corgrootte gering vergroot.
>
> ...
>
> Conclusie. Pneumothorax rechts. Versterkte tekening intrapulmonaal; lijkt niet geheel te verklaren door overvulling;
>
> ...
>
> Overweeg XXX.

For the non-pneumothorax class, the only criteria we have that there is no mention of a pneumothorax and no mention of a thorax drain. This means that the non-pneumothorax class does not only consists of healthy patients, but also other non-normal findings such as consolidations, emphysema, nodules, etc. Example report of the non-pneumothorax class:

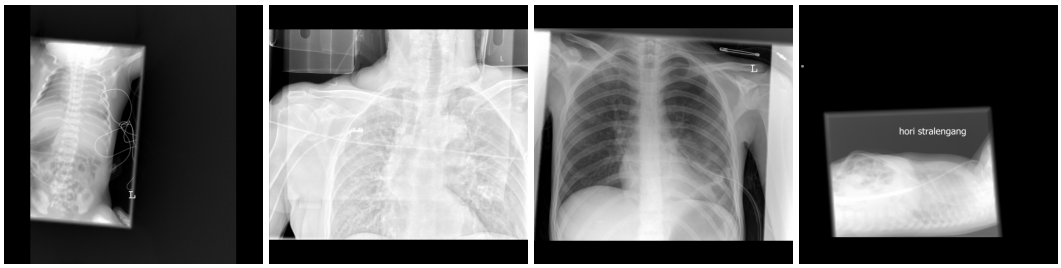
> X-thorax 2 richtingen. Wordt vergeleken met opnamen van XXX. In XXX. met de vorige opname is een in essentie ongewijzigd beeld van de XXX. Ongewijzigd aspect van de restafwijkingen rechts boven en in mindere mate linker bovenveld en links basaal. Open sinus pleurae beiderzijds. Enigszins afgeplatte diafragmakoepels passend bij COPD. Verder slank mediastinum en corfiguur. Conclusie: in essentie ongewijzigd beeld.

12.1.2.2 Image criteria

We randomly sample images from the main (noisy) *RadboudCXR* dataset and verify by hand whether the class corresponds with the class found in the text reports. Furthermore, we inspect the individual images belonging to the labels and adhere them to the following criteria:

1. Exclude images where a thorax drain is present. These are patients that have been treated for a pneumothorax, and therefore the pneumothorax has already been detected (Figure 12.1c).

2. Exclude images of babies. These patients have underdeveloped ribs, and most of these images show the complete body (Figure 12.1a).
3. Exclude lateral images. We include only frontal images, but these can be mislabeled (Figure 12.1d).
4. Exclude incomplete lung imagery. Make sure that both lobes are visible (Figure 12.1b).
5. Exclude small-scaled crops. In some cases a mere 50% of the complete image is usable (Figure 12.1a & 12.1d).
6. Exclude different organs. Sometimes there are other organs present, which incorrectly represent the lung.



(A) Small-scaled crop & imagery of a baby (B) Incomplete left lung (C) Thorax drain present in left lung (D) Small-scaled crop & lateral image

FIGURE 12.1: Examples of excluded test images based on the exclusion criteria. All the images originally belong to the pneumothorax class.

For the remaining images, we can assume that they are suitable for our test set and that they represent the corresponding class correctly. However, it could still be the case that in the non-pneumothorax case, there is still a pneumothorax present even though there is no mention of it in the text report.

12.2 Soft-label multiplication (RadboudCXR)

Xie et al., 2020 states that it is beneficent to add a lot of soft-label samples from the target domain into the training set. However, there is a trade-off between exploring the target dataset and exploiting the target dataset. On the former, the network can capture additional nuances that are not present in the source dataset. The latter causes overfitting on the target dataset, causing a decrease in generalizable model performance.

To investigate this trade-off, we experiment with the multiplication rate of our *iterative self-training* approach. We depict the performance of various multiplication rates on the test set and show the ratio soft-labels - original training data in Table 12.1. In this table, we see a significant performance difference between 0x multiplication (baseline) and 1x multiplication.

We take a look at the network architecture *EfficientNet-B4* while being in the first iteration of iterative self-training. As seen in Table 12.1, we see a significant difference between the baseline and 1x addition of soft-labels. When we experiment with the multiplication rate, we see that *10x* gives the best F1 score, while *1x* gives the best AUC score.

Multiplication Rate	% Soft-labels	# Total training data	**F1**	**AUC**
0x (baseline)	*0%*	*32.719*	*0.67*	*0.8168*
1x	6.37%	34.945	0.74	**0.8607**
2x	11.98%	37.171	0.77	0.8515
10x	40.48%	54.979	**0.78**	0.8587
20x	57.64%	77.239	0.76	0.8521
30x	67.12%	99.499	0.74	0.8577

TABLE 12.1: Multiplication factor influences the exploration – exploitation trade-off. Shown are the results of *EfficientNet-B4* on the *RadboudCXR* held-out test set.

Visualization further proves that such a trade-off exists. Figure 12.2 shows a diffused localization pattern for a high multiplication factor. Supported by the decrease in F and AUC score, we conclude that the usage of large multiplication factors during *iterative self-training* is discouraged.

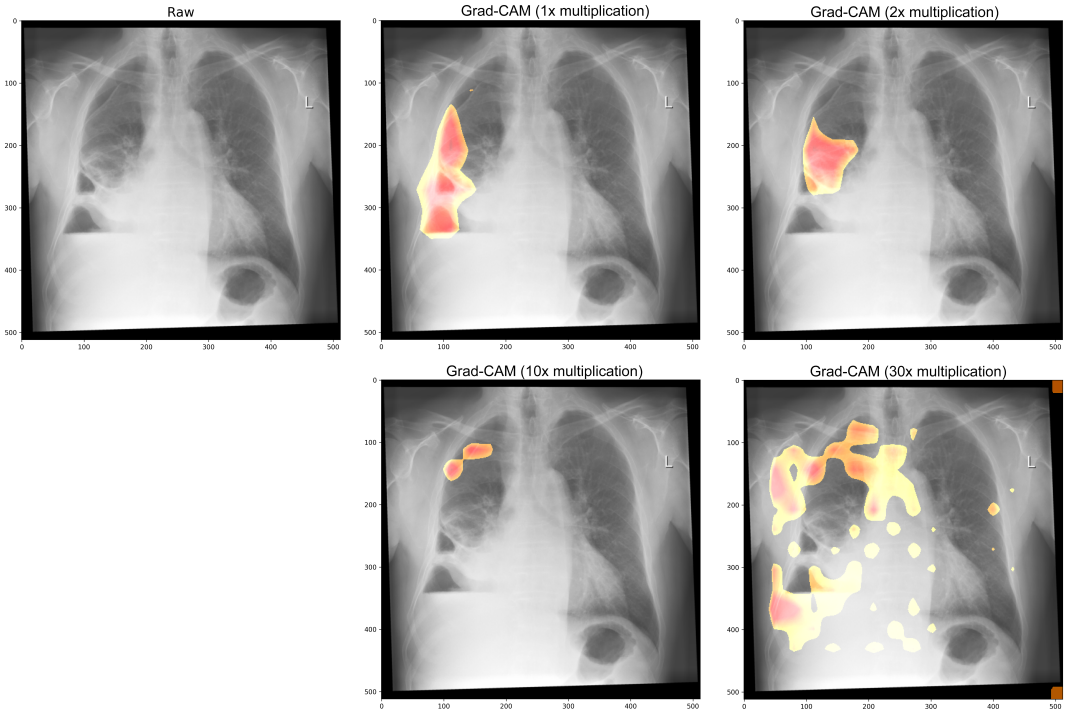


FIGURE 12.2: Grad-CAMs of *EfficientNet-B4* on a RadboudCXR test case. A pneumothorax is observed in the right lung. Prediction scores are 96.27%, 89.54%, 97.2%, and 90.4% for multiplication rates of 1x, 2x, 10x and 30x, respectively. The Grad-CAM localization is best for a multiplication factor of 1x, the worst for factor x30. Therefore, a higher multiplication factor might not result in better performance, but is prone to overfit.

Chapter 13

Results

We evaluate our best-performing networks that are trained on all public datasets directly to the held-out test set of *RadboudCXR*. The baseline scores for class-weighted F1 and AUC are 0,69 and 0,8259, respectively. We question whether *iterative self-training* can improve this score. If the scores are significantly higher than the baseline, we consider *iterative self-training* successful.

13.1 Iterative Self-training

During the experimentation, we apply *iterative self-training* to the complete dataset of *RadboudCXR*, and evaluate each iteration against the held-out test set. The results for the 512x512 input dimensions can be seen in Table 13.1, whereas the results for greater input dimensions (1024x1024) are shown in Table 13.2.

For 512x512, we see that all networks at iteration 2 show a decrease in AUC score, but increased F1 score compared to iteration 1. The trade-off between precision – recall is improved, but sensitivity – specificity is decreased. Further explanations regarding the metrics are explained in Chapter 3.4.

#	Model	Soft-labels	F1	AUC
0	*EfficientNet-B3*	*None (baseline)*	*0,67*	*0,8168*
0	*DenseNet-121*	*None (baseline)*	***0,70***	***0,8638***
0	*ResNet-152*	*None (baseline)*	*0,68*	*0,7970*
1	EfficientNet-B4	1x (1104/class)	0,74	0,8607
1	DenseNet-121	1x (1104/class)	0,74	**0,8769**
1	ResNet-152	1x (1104/class)	0,74	0,8626
2	EfficientNet-B5	2x (1054/class)	0,76	0,8394
2	DenseNet-169	2x (1054/class)	**0,77**	**0,8561**
2	ResNet-152	2x (1054/class)	0,76	0,8313

TABLE 13.1: Iterative self-training results for input size 512x512.

GPU problems The asterisk (*) as displayed in Table 13.2 states that there are GPU problems while training the networks using greater input image resolution. There are various causes for this. For iteration 0 of the *ResNet-152* network, iteration 2 of the *EfficientNet-B4* network, and iteration 2 of *DenseNet-169*, the maximum batch size is set to 1. If we increase the batch size further, the GPU runs out of memory. The downside of such a limited batch size is that the functioning of batch normalization is not achieved. On top of that, the network takes longer to train as well. As an example, *ResNet-152*, in particular, took over a week to train, which is not doable for the time that was left. *EfficientNet-B3* took around 5 days to train with an input

#	Model	Soft-labels	F1	AUC
0	*EfficientNet-B3*	*None (baseline)*	***0,77***	***0,8750***
0	*DenseNet-121*	*None (baseline)*	*0,73*	*0,8814*
0	*ResNet-152**	*None (baseline)*	*0,57*	*0,7595*
1	EfficientNet-B3	1x (1139/class)	**0,82**	**0,9242**
1	DenseNet-121	1x (1139/class)	0,78	0,9055
1	ResNet-152*	–	–	–
2	EfficientNet-B4*	2x (1415/class)	0,77	0,8629
2	DenseNet-169*	2x (1415/class)	0,12	0,2283
2	ResNet-152*	–	–	–

TABLE 13.2: Combined table 1024x1024.
* GPU problems during training of the networks.

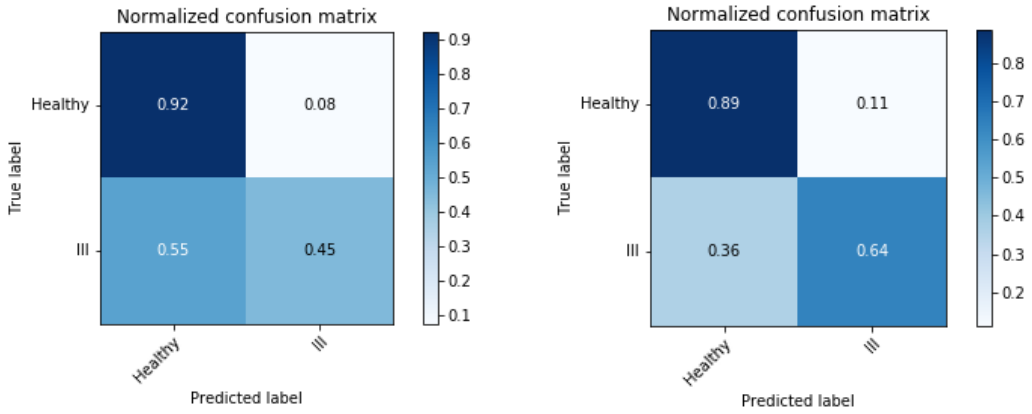
resolution of 1024x1024. In this table, insufficient results are seen for *DenseNet-169*. Apparently, the network is not able to converge due to a batch size of 1.

We can conclude that the results for the 1024x1024 input resolution experiments are not reliable, although these look promising. For the remaining of the chapter, we only take results of *iterative self-training* for a resolution of 512x512 into account.

13.1.1 EfficientNet

The confusion matrices for the *EfficientNet* (512x512) domain adaptation experiments are depicted in Figure 13.1. Before domain adaptation, *EfficientNet* has a high specificity. Of all the non-pneumothorax images, *EfficientNet* is right 92% of the time. However, when a pneumothorax is present, *EfficientNet* successfully classifies this image as a pneumothorax just 45% of the time (low sensitivity).

After *iterative self-training*, the sensitivity increases to 64%. The trade-off here is that specificity slightly decreases to 89%.



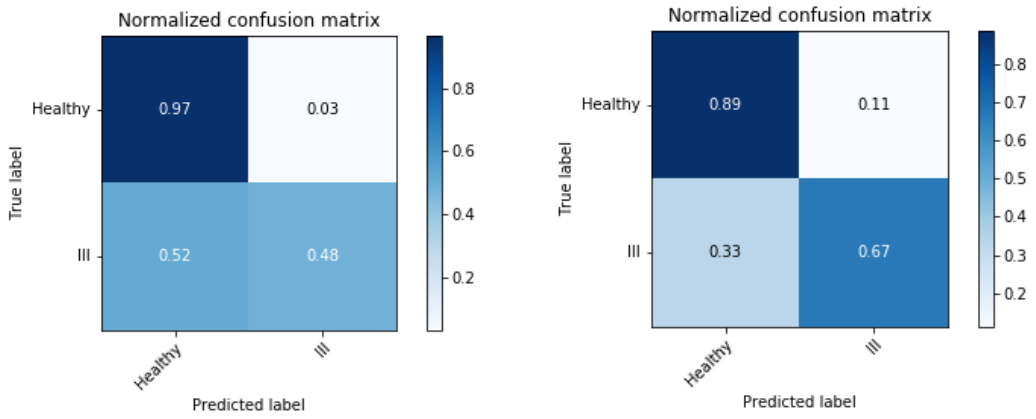
(A) Baseline (B) Iteration 2 (domain adapted)

FIGURE 13.1: Confusion matrices of *EfficientNet* experiments. After *iterative self-training*, the sensitivity increased with a slight decrease in specificity.

13.1.2 DenseNet

The normalized confusion matrices for DenseNet (512x512) (Figure 13.2 show better results for both the baseline and *iterative self-training*. The baseline results also have

the tendency to have a very high specificity (97%) and show a slight decrease in specificity during *iterative self-training*. The sensitivity significantly increases from 48% to 67%.



(A) Baseline (B) Iteration 2 (domain adapted)

FIGURE 13.2: Normalized confusion matrices of *DenseNet* experiments. After *iterative self-training*, the sensitivity increased with a slight decrease in specificity.

13.2 Visualizations

Although the performance of pneumothorax detection on larger resolutions (1024x1024) look promising, we have not managed to run a complete pipeline of *iterative self-training* for this resolution. Therefore, we choose to validate our findings on the 512x512 resolution that went through all the iterations successfully. We visualize the activations of *EfficientNet*, as this is the only network where we are able to scale up efficiently. In particular, we show the differences in activation between *EfficientNet-B3* and the final iteration trained on *EfficientNet-B5*.

We show that our *iterative self-training* method can infer knowledge based on various circumstances. In all cases, we show examples where the discovery of a pneumothorax is new and therefore showing untreated pneumothoraces.

Figure 13.3 shows a chest radiograph and radiology report of a patient directly after surgery. The patient is recovering in the intensive care unit, and by routine, a chest radiograph is taken after an invasive procedure to rule out possible abnormalities that the surgery has caused. In this case, the patient has had a *sternotomy*, meaning that the middle of the chest has been surgically opened (often done for open-heart surgery). In this case, the chest radiograph shows that the *sternotomy* caused the right lung to collapse (traumatic pneumothorax). The algorithm seems to improve its localization towards the right apical lung correctly.

Figure 13.4 shows a PA-view chest radiograph of a patient not on the intensive care, but obtained during the outpatient clinic. This patient shows a large pneumothorax on the lateral side of the right lung. After *iterative self-training*, the Grad-CAM correctly shows a new blob downward laterally.

Figure 13.5 shows a chest radiograph from the portable device, on a patient in a supine position lying in bed. Due to this patient being in supine position, the pneumothorax does not have to be located around the edges of the chest. Iteration 2 shows a pneumothorax on the upper left lung, but not around the chest wall. This finding of the algorithm corresponds to the finding of the radiologist.

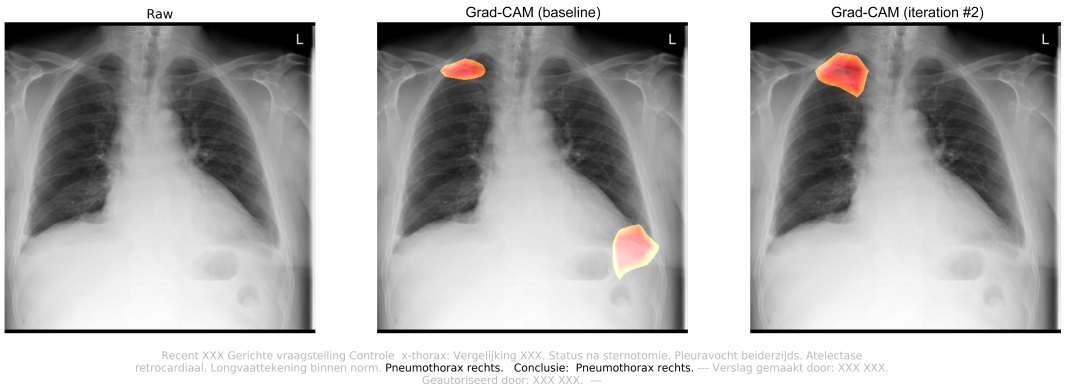


FIGURE 13.3: Patient with a traumatic pneumothorax in the right lung, which happened after heart surgery (sternotomy). Iterative self-training improved the pneumothorax probability from 28.58% (FN) to 79.91% (TP)

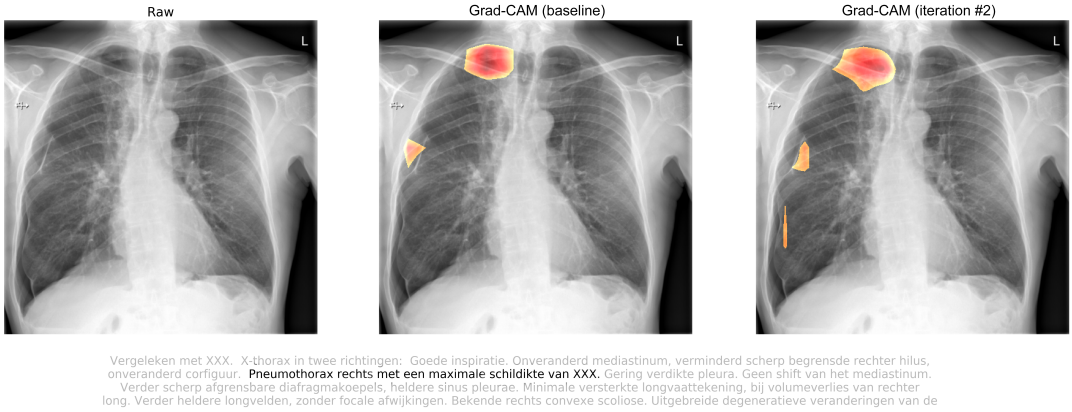


FIGURE 13.4: Patient with a large lateral pneumothorax in the right lung. After iterative self-training, the network is able to locate a more substantial part of the pneumothorax. Pneumothorax probability increased from 40.92% to 79.91%.

Figure 13.6 shows a false-positive prediction of a pneumothorax. At least that is what the label says. The corresponding radiology report states: "the lung is probably not yet fully attached on the base". We do not know which lung this entails, but there seems uncertainty amongst the presence of a pneumothorax. The baseline of *iterative self-training* shows activation at the right lung apically and basally. Iteration 2 of *iterative self-training* shows activation only at the right lung basally. Iteration 2, therefore, nails the localization but decreases the pneumothorax probability from 62.71% to 17.92%. In this case, the algorithm could support the radiologist in his decision making, as by comparing the localization areas with the suggested area of the radiologist.

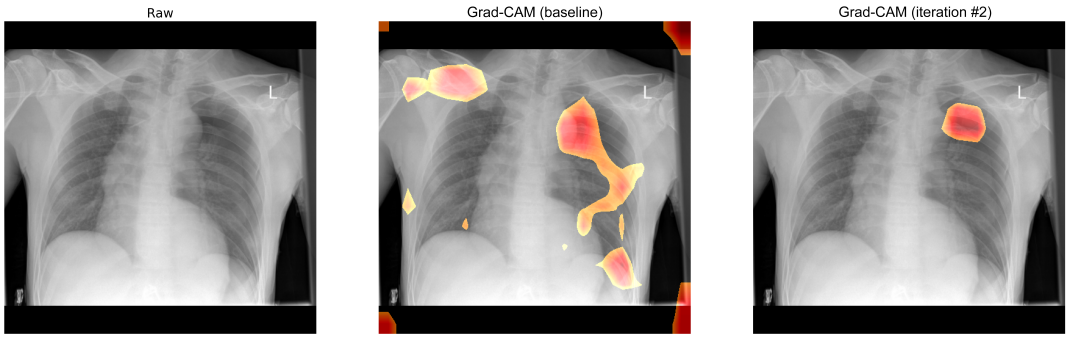


FIGURE 13.5: Patient in a supine position with a pneumothorax. A supine position causes the pneumothorax to appear in the middle of the chest due to gravity. Iterative self-training drastically improved pneumothorax probability (22.07% to 90.69%) and localization up to a point where the Grad-CAM accurately marks the location of the pneumothorax.

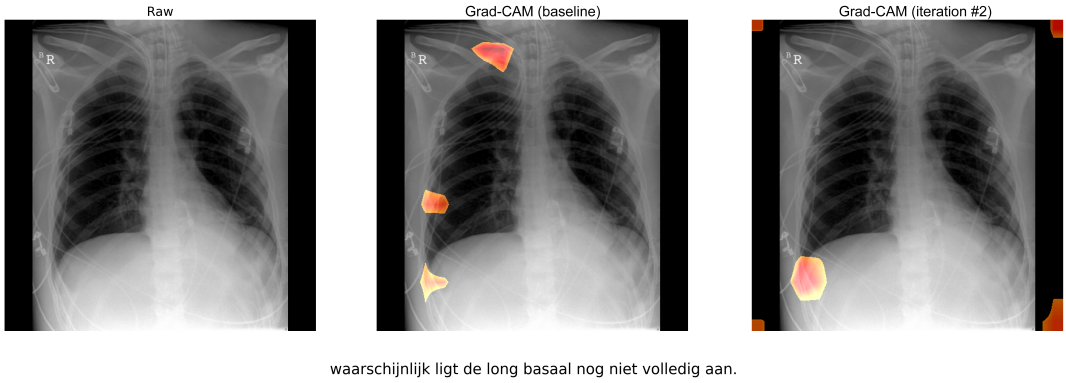


FIGURE 13.6: Image of a patient that probably does not have a pneumothorax, although the radiologist is uncertain as well: "the lung is probably not yet fully attached on the base". Initially, the pneumothorax probability is 62.71% (false-positive prediction), but after iterative self-training, this decreases to 17.92%.

13.3 Prevalence

As we have seen in Chapter 3.4, reporting merely the sensitivity and specificity for this task is not enough. The pneumothorax prevalence rates tend to differ between datasets, as all these datasets originally suffer from some sort of selection bias (e.g., opt-in method of acquiring data, unknown data source). Depending on the prevalence rate, a positive prediction might yield many false-positive predictions. In *RadboudCXR*, the pneumothorax prevalence rate is 8%.

To illustrate how the prevalence rate affects the number of false-positives, we depict an example calculation. Assume that the best possible pneumothorax model achieves a sensitivity and specificity of 0.95 and 0.85, respectively [1]. With a prevalence rate of 8%, this gives a positive predicted value (PPV) of 0.355. From this PPV we can conclude that for every 100 positively-predicted x-ray images, there will be 65 false positives! With an increased prevalence rate, the number of false positives

[1]Commercial product Qure – qXR v2.0 (Putha et al., 2018). This is the highest self-reported model performance for a commercial product currently on the market, although these scores can be criticized.

will be lower. This shows that even for the most accurate models, the trust in the algorithm cannot be easily established. It makes sense that these algorithms are therefore deployed in triage situations, where recall is more important.

Chapter 14

Discussion

We apply the *iterative self-training* scheme to our local clinical dataset. As baseline performance on the local clinical dataset, we take our best performing models as seen in Part I (Chapter 4). After the experiments, generally all networks seem to benefit from the *iterative self-training* method. The mean F1 scores increase from 0,68 to 0,77, and the mean AUC scores improve from 0,8259 to 0,8422 for low input resolution (512x512). Higher resolution images (1024x1024) tend to push performance even further: an F1 score of 0,82 and an AUC score of 0,9242 for the *EfficientNet* architecture is achieved. Unfortunately, training these high-resolution images cost a significant amount of computing power and pushes the boundaries of the GPU we have available. Ideally, we would want to compute more iterations (3 or 4), to see at which iteration the networks converge and show the best results.

To answer our research question: the proposed *iterative self-training* method increases performance on the target dataset, meaning domain adaptation is successful. The performance on the source dataset by following this approach seem to be unaffected, except for performance on the external test set *ChestX-ray14* (Chapter 4.4). These results show that the performance might not increase on the source dataset and achieve *domain generalization*. However, results show that *iterative self-training* is robust enough to improve performance on the target dataset while maintaining at least equal performance on the source dataset.

14.1 Multiplication rates

There is a difference in multiplication rates between our clinical experiments and the original *SIIM* experiment (Chapter 8.2). For clinical validation, we use multiplication rates of x1, x2, while for domain adaptation towards *SIIM* we use multiplication rates of x10, x30. We found that for clinical validation, a lower multiplication rate is needed to achieve comparable results. We are unsure what causes this, but we assume this has to do with the fact that we already have a quite robust model trained on all the public datasets, and we need only a small 'nudge' in the right direction, and no new concepts need to be learned.

14.2 Robustness

In order to make the results more robust, we would need to apply cross-validation in our experiments. This way, we have a better overview of which models would be significantly better, if at all. Due to time limitations, verifying these results using cross-validations was not an option, and thus leave cross-validation for future work.

14.3 *RadboudCXR*

The local clinical dataset *RadboudCXR* is noisy. We spent limited time investigating this dataset, meaning that better labeling tools and filtering methods might significantly improve performance on this dataset.

Prevalence As we mention in Chapter 13.3, the pneumothorax prevalence rate of *RadboudCXR* is low (8%). It makes sense to integrate this algorithm for triaging only, as this algorithm will present relatively many false-positives.

Part IV

Concluding remarks

Conclusion

In this thesis, we investigate three parts that all focus on improving automated pneumothorax detection. First, we develop an open-source approach towards automated pneumothorax detection using deep learning. Our advantage over related work is that:

1. We avoid chest tube bias using study selection.
2. We verify our method on public data.
3. We verify our method using external data validation.
4. We split our data into training/test split at patient-level, which makes sure a patient is not present in both the train and test split.
5. We report multiple performance metrics and calculate different metrics when needed for comparison (e.g., on *ChestX-ray14*).

Few related work studies (6 / 15 papers) use a public dataset (such as *ChestX-ray14*) to verify its results. Even when the a certain public dataset is chosen for evaluation, we observe a difference in the assembly of test splits (Table 4.3). Using the most recent results on the official test split of Majkowska et al., 2020, we achieve equal performance as state-of-the-art: 0.9391 AUC. As all the datasets are public, we publish our algorithm and model weights online so other people can benefit from our work [1].

After achieving a solid open-source foundation to pneumothorax detection, we knew that we had a noisy local dataset at hand (*RadboudCXR*), but without any labelling available. As most hospitals rely on unstructured radiology reporting, we need a solution that is not that much dependent on report labels. We adapt the *iterative self-training* method of Xie et al., 2020 in order to achieve unsupervised domain adaptation using an inference-based method. This allows us to leverage the local dataset without any additional labeling. Apart from changing the goal of the method, we further propose the following adaptations:

1. Soft-label multiplication (i.e. oversampling rate).
2. Model ensembling.

We test this method by simulating domain adaptation on the public datasets *CheXpert* $\rightarrow$ *SIIM*. Both *EfficientNets* and *DenseNets* seem to benefit from *iterative self-training*: 0.60 $\rightarrow$ 0.79 weighted F1; 0.8602 $\rightarrow$ 0.8982 AUC.

Finally, we evaluate the *iterative self-training* domain adaptation method using a local dataset *RadboudCXR*. The results show that the performance on the source datasets (*CheXpert*, *SIIM*, *MIMIC-CXR*) remain equal, but significantly achieves domain adaptation performance on the target dataset *RadboudCXR*: 0.68 $\rightarrow$ 0.77 mean F1; 0.8259 $\rightarrow$ 0.8422 mean AUC. This proves that our method is able to perform in a real-world noisy dataset. The final algorithm is deployed and testable at the *Grand-Challenge* platform: https://grand-challenge.org/algorithms/cxr-pneumothorax-detectio

[1] DIAG will publish this algorithm as a package in *OpenCXR*.

Future work

Cross-validation For every part of the thesis, adding cross-validation will strengthen the results, and possibly highlight different performances for various network architectures. As each network takes around five days to train, it is not feasible to achieve cross-validation results during the limited timespan.

Leveraging *SIIM* segmentation labels For the *SIIM* dataset, we have pixel-level pneumothorax annotations. We currently only use image-level annotations for all our data, and therefore not make use of these detailed annotations. For future work, we could leverage these segmentation labels to strengthen the localization results, such as combining Grad-CAM localization with a segmentation pipeline.

Iterative self-training experiments We have scratched the surface of iterative self-training. Although these results look promising, it might be good to experiment with the following parameters:

- Multiplication rates.
- Soft-label thresholds. Determine the optimal soft-label quality – quantity trade-off.
- Number of iterations. Continue running experiments until the maximum performance is achieved. In the current experiments, the network capacities did not allow for such large iterations.
- Larger batch size. Due to GPU constraints, we could not adhere to the prerequisite of an equal minibatch distribution of source data and target soft-labels.
- Ablation study: Solo versus ensemble models.

RadboudCXR The local dataset *RadboudCXR* is in need for reliable labels. We spent limited time in creating a simple NLP tool to extract pneumothorax labels, but improving the quality of the labels might help to get a better insight into this dataset.

Integration in the clinic We have implemented a state-of-the-art pneumothorax detection algorithm using only publicly available datasets. We have made our code and model publicly available and provide an online demo with the aim of accelerating future research. Further, we have investigated an unsupervised domain adaptation approach to adapt to new domains without additional labeling, which could be useful for integration of the model into new hospitals. Since the proposed algorithm is available online, we could directly integrate the algorithm into the clinical workflow of a radiologist using the *Grand-Challenge* API. The results of this algorithm could trigger specific priority flags for triaging or append a line in the radiology report stating the suspected presence of a pneumothorax. This could potentially be useful to prevent delays in reporting the critical time-sensitive abnormality: pneumothorax.

Appendix A

Training Pipeline

The pneumothorax classification pipeline is written in Python 3.7 using Tensorflow 2.0.

For our experiments, we set a randomness seed to the integer *1337*, so that for each experiments we have controlled (pseudo-random) conditions. By using a pseudo random number, we ensure the same stratified data splits & data augmentations.

We optimize our networks using the Adam optimizer (Kingma and Ba, 2014) and only save our model when validation loss compared to previous epochs decreased.

A.1 Hyperparameter optimization

We try to optimize the following parameters in order to get the most robust results:

- Normalizations
  - ImageNet
  - ClaHe
- Lung masking
- Data augmentation
- Learning rate
  - 1e-3
  - 1e-4
  - 1e-5
  - 1e-6
- Learning rate scheduling
  - Factor
  - Patience
- Loss function
  - Binary focal loss
  - Binary cross-entropy
- Image Resolution
  - 512x512

  - 1024x1024

- Bottom convolution pooling
  - Max pooling
  - Average pooling

There are many parameters to be selected, but the pre-trained networks are too big to brute-force all of the parameters. Training time for one configuration takes around 10 hours! Therefore, we optimized the parameters *normalization*, *learning rate* and *lung masking* using the python package *HParams* (Petrochuk, 2019), and explored the other parameters by trial and error.

Learning rate & scheduling Many papers argue that the learning rate is affected by the used batch size of the network. Because for different network architectures and image resolutions we would need a different batch size it is even more costly to determine the optimal learning rate. In our case, we found that a learning rate of 1e-5 works well. When the performance on the validation set does not improve after 5 epochs (plateau), we decrease the learning rate by a factor 0.3.

Loss function We experimented with two loss functions: binary focal loss and binary cross-entropy. Focal loss (Lin et al., 2017) is used as a loss function for highly imbalanced datasets, where there exists a sparse class. It is basically a weighted cross-entropy loss, where the classes that are underrepresented get a higher weight that contributes to the computation of the loss.

Binary cross-entropy is a measure that computes the distance between the predicted outcome and the expected value. This method does not account for class imbalance and therefore other measures should be taken to address class imbalance.

We found focal loss not effective enough to replace binary cross-entropy loss. In order to still tackle the class imbalance problem, we use oversampling as a class balancer, where we oversample the underrepresented class *pneumothorax*.

Image Resolution For *EfficientNet-B3*, using higher resolution images (in our case we tested 512x512 and 1024x1024) seems to improve improve F1 & AUC scores from 0.76 & 0.8622 to 0.83 & 0.9148, respectively. There are two problems with using higher image resolution. The first problem is that in order to train on the GPU, we need to decrease the batch size, which reduces the effect of batch normalization. The second problem is that when applying the iterative self-training scheme, the memory usage becomes too big with increased network sizes. We cannot increase network capacity when upscaling from *EfficientNet-B5* to *EfficientNet-B6*.

Bottom convolution pooling When connecting the backbone with the last classification layer, we have to deal with the outgoing pooling layer coming from the backbone. Two types of pooling that we select are *max pooling* and *average pooling*. *Max pooling* will get the maximum value of a convolution area as defined by the filter size, while *average pooling* averages the convolution area to end up with a value. *Max pooling* performs worse on localization and ignores a large part of the data. It has been found that in our case, *average pooling* seems to give best results.

A.1.1 Normalizations

ImageNet normalization Many deep learning papers describe that input normalizations help to keep proper gradient propagation throughout the network. Because our networks are initialized with pre-trained ImageNet weights, it is a common practice to standardize network input by subtracting the ImageNet mean and dividing by the standard deviation. For each input image, we subtract the ImageNet means (0.485, 0.456, and 0.406 for each channel, respectively). Then, we divide by the standard deviation (0.229, 0.224, 0.225 for each channel, respectively). It is important to note that ImageNet normalizations may cause the network to converge faster, but not necessarily lead to better results.

However, every network consists of *batch normalization* layers, which means that after each convolution layer, the activations get normalized. Batch normalizations also help us to retain gradient propagation, making ImageNet normalization to have less of an impact compared to networks without batch normalizations. During our experiments, we confirmed that there exists no significant difference between using *ImageNet* normalization and not applying these normalizations.

Contrast Limited Adaptive Histogram Equalization (CLAHE) Images straight out the X-ray machine are flat and generally lack contrast. Contrast is needed to observe edges and possible pathologies in chest radiographs. CLAHE normalization is often used for pre-processing and helps to increase local details and contrast. It improves the local details by dividing the images into blocks and then applying histogram equalization (Koonsanit et al., 2017). In Figure A.1, we see a normal and a CLAHE normalized image.

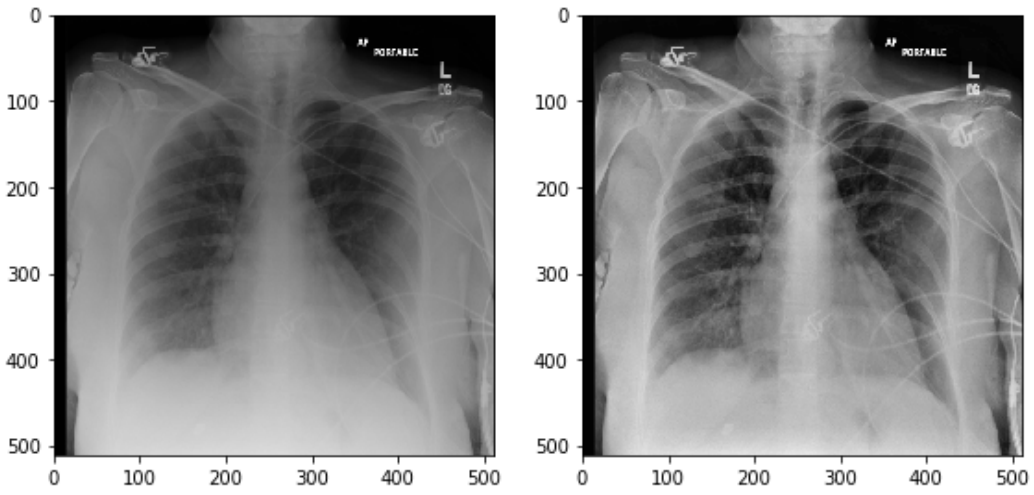


FIGURE A.1: Normal image (left) vs CLAHE normalized image (right)

A.1.2 Lung masking

We introduce lung masking as a preprocessing step, as this decreases the amount of noise the network will get as input. We first run a separate algorithm that segments lung tissue, followed by a dilation kernel. Dilation is applied in order to make up for lung segmentation errors and to provide an overview of the bone structure. In

Figure A.2, we display the difference between input images before and after lung masking.

(A) Normal (B) Lung mask (C) Dilated lung mask

FIGURE A.2: Example of a lung segmented chest x-ray

We ran different experiments to determine whether lung masking helps to improve results. Unfortunately, we see that lung masking does not contribute to a better performance as the scores of using lung masking (dilated lung masks or regular lung masks) equals the score of not using lung masking at all. Therefore, the best setting is to use a regular input image without lung masking.

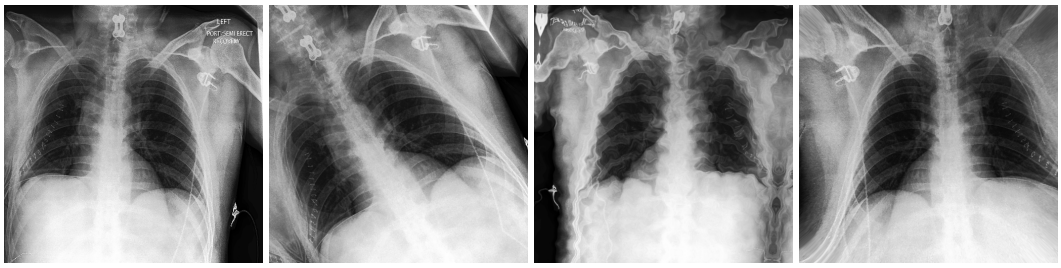
A.1.3 Data Augmentation

Because we have limited samples of a pneumothorax, and we want to make the network invariant to scaling, shifting, and rotations, we introduce data augmentations. For this we use the *albumentation* python framework (Buslaev et al., 2020). We found that in related work, the best performing data augmentations are as follows:

- Horizontal flipping (probability: 0.5)
- One of: random contrast, random gamma or random brightness
- One of: Elastic transformation (Simard; Steinkraus; Platt, et al., 2003), grid distortion or optical distortion
- Random scaling and rotation (max 45 degrees)

In Figure A.3 we see some augmentations that are performed. By making sure the network never sees the same example of an image, we know it learns something new every epoch.

Figure A.3b shows a zoomed in (scaled) and rotated variation. By using this we make the network invariant to rotations & scaling. In Figure A.3c the image is horizontally flipped, shifted, and grid distorted.



(A) Normal (B) Random augment (C) Random augment (D) Random augment

FIGURE A.3: Example data augmentations

Appendix B

Data Leakage

In this chapter, we explore the possible data leakage between the *Google NIH* labels (Chapter 3.2.1) and the *SIIM-ACR* dataset (Chapter 3.2.2), which we use for training. The *SIIM-ACR* dataset does not provide any description of where the dataset is coming from.

A naive approach would be to assume that the *SIIM-ACR* training set does not incorporate the test set from *ChestX-ray14*, but just the training set. As the Society for Imaging Informatics in Medicine (*SIIM*) has radiologists at their disposal, it would be a good research practice for them to select a subset of the pneumothorax training set of *ChestX-ray14* and let the radiologists annotate a segmentation map for those images. In turn, this improves the quality of the *ChestX-ray14* as the training split only consists of NLP extracted labels, and the researchers that use the *SIIM* dataset for training can test on the *ChestX-ray14* split without worrying about data leakage.

In research, we should not be naive. Therefore, we actively test whether the current training set of *SIIM* consists of images that exist in NIH test set (as labeled by Google in 2019). To do this, we calculate the *average hash* and the *perceptual hash* (Zauner, 2010) for each image. Hashing each image gives us a unique fingerprint of the content of the images, which is robust against scaling and watermarking (e.g., AP/PA labels or lung side markings on chest x-rays). To our surprise, we do find 213/1964 images (11%) colliding with the same hash. Of these images, 64 are pneumothorax and 149 are non-pneumothorax. After inspecting the images with a hash collision, we verify that indeed the images are the same. Thus, we can conclude that the *SIIM* dataset contains images coming from the **test** set of *ChestX-ray14*!

We would have wanted to remove these files from our training set, but due to time constraints, there was no option to re-train our network on a new training set. Therefore, although not ideal, we removed these images from the test set. The difference in performance between the test sets with and without data leakage remains minimal (0.940 AUC vs. 0.9391 AUC).

Appendix C

Paper

Note: The paper contains preliminary results for the pneumothorax detection task.

Unsupervised Domain Adaptation via Iterative Self-Training for Chest Radiograph Analysis

Author(s) names withheld EMAIL(S) WITHHELD
Address withheld

Editors: Under Review for MIDL 2020

Abstract

Many supervised systems suffer a drop in performance in domains distinct from their training data. In this work, we propose an unsupervised domain adaptation approach through iterative self-training in order to improve performance on an unseen dataset without additional labelling. We investigate the impact of this approach in two different chest radiography applications: heart segmentation and pneumothorax classification, using four publicly available datasets. Our results show that this can improve IoU for heart segmentation from 0.70 to 0.81 and AUC for pneumothorax classification from 0.86 to 0.90.

Keywords: unsupervised domain adaptation, self training, pseudo labeling

1. Introduction

It is well known that supervised deep-learning algorithms often do not perform optimally on data outside their training domain. This is a major obstacle to successful implementation of medical image analysis algorithms, because in clinical practice there exists a wide variation in scanner hardware, acquisition protocols and reconstruction methods. Such variations occur even within single hospitals. Collection of new annotations from many datasets is time-consuming and costly. Unlabelled medical images are usually available in abundance, however, which provides motivation for unsupervised domain adaptation (UDA). These techniques focus on scenarios where labeled training data is available in a source domain but the goal is to achieve a good performance on a different (target) domain without additional labelling.

There is a large body of literature for unsupervised domain adaptation (Sun et al., 2019). One line of research focuses on image transformation using generative models (Hoffman et al., 2018; Bousmalis et al., 2017). Another option is to induce alignment between the source and target domains in some feature space (Bousmalis et al., 2016; Long et al., 2017). Self-training is an alternative approach where the network is trained on source data, used to generate pseudo-labels on the target dataset, and then re-trained using information from these generated pseudo-labels. Recently (Xie et al., 2019) demonstrated that an iterative self-training approach achieved a state-of-the-art performance on natural image classification in a semi-supervised setting.

In this work, we adapt the iterative self-training concept for unsupervised domain adaptation and investigate its performance for two different medical image analysis tasks: heart segmentation and pneumothorax classification on chest radiographs. In addition, we demonstrate the effect of the iterative process and of increasing numbers of unlabeled images.

2. Data

The public source and target datasets used for each task are detailed in Table 1. The unlabelled target data for heart-segmentation is randomly selected. The heart is manually segmented in 600 radiographs for use as a held-out test set. For pneumothorax classification the SIIM provided validation split is used as the held-out test set. All images underwent per sample mean-standard deviation normalization.

	Source Data			**Target Data**		
	Name	#images	#annotations	Name	#images	#annotations
Segmentation	JSRT	247	247	CXR14	10.600	600
Classification	CHEXPERT	10.836	10.836	SIIM	10.648	1.602

Table 1: Overview of the used datasets, JSRT (Shiraishi et al., 2000; van Ginneken et al., 2006), CHEXPERT (Irvin et al., 2019), CXR14 (Wang et al., 2017), SIIM (SIIM, 2019).

3. Methods

3.1. Baseline Method

A standard U-Net architecture (Ronneberger et al., 2015) was used for segmentation experiments, and a DenseNet-121 architecture (Huang et al., 2017) pretrained on ImageNet was used for pneumothorax classification. Data augmentations (rotation, brightness, contrast, horizontal flipping) were applied during training. As a baseline, the models were trained on the source domain and tested on the target domain.

3.2. Iterative Self-Training

We adapt the iterative self-training approach (Xie et al., 2019) for UDA as follows:

1. Train the network on the source domain and use it to generate pseudo-labels (SoftMax output) for the target domain.
2. Pseudo-labels with SoftMax value >0.9 are selected and included in the training set for the next iteration.
3. We train the network on the increased training set, and generate new pseudo-labels as before. Iterate from step 2.

In the first iteration, we expand the capacity of U-Net by adding 2 convolutional layers followed by a max pooling operation in both the expanding and contracting path. For the classification architecture, we switch from Densenet-121 to Densenet-169 on the first iteration. The additional capacity is beneficial for the networks to be able to adapt to the new domain. We decrease the batch size to compensate for the larger architectures.

We experiment with different numbers of unlabeled images for segmentation experiments to see its impact on performance. Performance is measured by area under the receiver operating curve (AUC) for classification and by intersection over union (IoU) for segmentation.

4. Results

As can be seen in Table 2, the classification baseline method achieved an AUC of 0.860, which increased to 0.904 by self-training on the unlabeled set. For the segmentation task, the baseline method achieved an IoU of 0.702 in the target domain, which increased to a maximum of 0.818 by self-training on 5000 unlabeled images. Experiments with 100 and 10000 unlabelled images showed similar but slightly lower improvements. There appears to be a sweet spot for the number of unlabeled images to use in the self-training process. We will further investigate this in future work. Figure 1 shows sample results for each task.

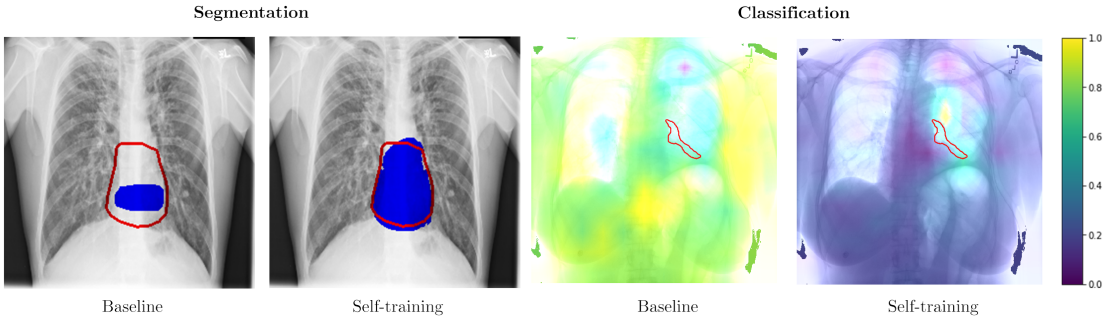


Figure 1: Two examples with a large difference between baseline and self-training. **Left**: CXR14 segmentation results with reference in red and network prediction in blue. **Right**: Classification results showing occlusion sensitivity (Selvaraju et al., 2017) overlay before and after self-training on the SIIM validation set. The pneumothorax location from the reference is shown in red. SoftMax was 0.40 in baseline classifying this as a false negative, and 0.90 after self-training, making this a true positive.

Task	**Dataset Source/Target**	**Number of unlabeled images**	**Performance 1^{st} | 2^{nd} | 3^{rd} iter**
Classification	CheXpert/SIIM	0	0.860
		10000	0.888 | 0.892 | 0.904
Segmentation	JSRT/CXR14	0	0.702
		100	0.768 | 0.795 | 0.801
		5000	0.802 | 0.815 | 0.818
		10000	0.786 | 0.803 | 0.812

Table 2: Performance with no self-training, and after 1,2 and 3 iterations of self-training. Classification performance is measured by AUC and segmentation performance by IOU.

5. Conclusion

In this work we show that self-training based UDA can improve performance in both segmentation and classification tasks on chest radiographs. The iterative process of pseudo-labeling consistently improved performance for both tasks. In the segmentation task we observed that using only 100 unlabeled images was sufficient to improve performance substantially.

Acknowledgments

Acknowledgments withheld.

References

K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 343–351. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6254-domain-separation-networks.pdf.

K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 95–104, 2017. doi: 10.1109/CVPR.2017.18. URL https://doi.org/10.1109/CVPR.2017.18.

J. Hoffman, E. Tzeng, T. Park, J. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1989–1998, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/hoffman18a.html.

G. Huang, Z. Liu, L. van der Maaten, and K. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 07 2017. doi: 10.1109/CVPR.2017.243.

J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. L. Ball, K. S. Shpanskaya, J. Seekins, D. A. Mong, S. S. Halabi, J. K. Sandberg, R. Jones, D. B. Larson, C. P. Langlotz, B. N. Patel, M. P. Lungren, and A. Y. Ng. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *AAAI*, pages 590–597, 2019. URL https://doi.org/10.1609/aaai.v33i01.3301590.

M. Long, H. Zhu, J. Wang, and M. I. Jordan. pages 2208–2217, 2017.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

J. Shiraishi, S. Katsuragawa, J. Ikezoe, T. Matsumoto, T. Kobayashi, K. Komatsu, M. Matsui, H. Fujita, Y. Kodera, and K. Doi. 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. *AJR. American journal of roentgenology*, 174:71–4, 02 2000.

SIIM. Siim-acr pneumothorax segmentation challenge. https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/overview, 2019.

Y. Sun, E. Tzeng, T. Darrell, and A. A. Efros. Unsupervised domain adaptation through self-supervision, 2019.

B. van Ginneken, M.B. Stegmann, and M. Loog. Segmentation of anatomical structures in chest radiographs using supervised methods: a comparative study on a public database. *Medical Image Analysis*, 10(1):19–40, 2006.

X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *arXiv:1705.02315*, 05 2017.

Q. Xie, M.T. Luong, E. Hovy, and Q. V. Le. Self-training with noisy student improves imagenet classification, 2019.

Bibliography

Annarumma, Mauro et al. (2019). "Automated triaging of adult chest radiographs with deep artificial neural networks". *Radiology* 291.1, pp. 196–202.

Baltruschat, Ivo M et al. (2019). "Comparison of deep learning approaches for multi-label chest X-ray classification". *Scientific reports* 9.1, pp. 1–10.

Blum, Avrim and Mitchell, Tom (1998). "Combining labeled and unlabeled data with co-training". *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100.

Blumenfeld, Aviel; Konen, Eli, and Greenspan, Hayit (2018). "Pneumothorax detection in chest radiographs using convolutional neural networks". *Medical Imaging 2018: Computer-Aided Diagnosis*. Vol. 10575. International Society for Optics and Photonics, p. 1057504.

Bousmalis, Konstantinos et al. (2016). "Domain separation networks". *Advances in neural information processing systems*, pp. 343–351.

Bousmalis, Konstantinos et al. (2017). "Unsupervised pixel-level domain adaptation with generative adversarial networks". *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3722–3731.

Bridges, Kenneth G et al. (1993). "CT detection of occult pneumothorax in multiple trauma patients". *The Journal of emergency medicine* 11.2, pp. 179–186.

Buslaev, Alexander et al. (2020). "Albumentations: Fast and Flexible Image Augmentations". *Information* 11.2. ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https://www.mdpi.com/2078-2489/11/2/125.

Chen, Minmin; Weinberger, Kilian Q, and Blitzer, John (2011). "Co-training for domain adaptation". *Advances in neural information processing systems*, pp. 2456–2464.

Cheng, Yeung-Leung et al. (2009). "The impact of smoking in primary spontaneous pneumothorax". *The Journal of thoracic and cardiovascular surgery* 138.1, pp. 192–195.

Choi, Young Soon et al. (1998). "Delayed tension pneumothorax complicating subclavian vein catheterization and positive pressure ventilation: a case report". *Korean J Anesthesiol* 34.3, p. 660.

Cicero, Mark et al. (2017). "Training and validating a deep convolutional neural network for computer-aided detection and classification of abnormalities on frontal chest radiographs". *Investigative radiology* 52.5, pp. 281–287.

Cohen, Joseph Paul et al. (2020). "On the limits of cross-domain generalization in automated X-ray prediction". *arXiv preprint arXiv:2002.02497*.

Communicating radiation risks in paediatric imaging (Jan. 2016). URL: https://www.who.int/ionizing_radiation/pub_meet/chapter1.pdf.

Dupre, Robert et al. (2019). "Iterative self-learning: Semi-supervised improvement to dataset volumes and model accuracy". *Proc. CVPR Workshop*, pp. 79–82.

French, Geoffrey; Mackiewicz, Michal, and Fisher, Mark (2017). "Self-ensembling for visual domain adaptation". *arXiv preprint arXiv:1706.05208*.

Ganin, Yaroslav et al. (2016). "Domain-adversarial training of neural networks". *The Journal of Machine Learning Research* 17.1, pp. 2096–2030.

Gooßen, André et al. (2019). "Pneumothorax Detection and Localization in Chest Radiographs: A Comparison of Deep Learning Approaches".

Guan, Qingji et al. (2018). "Diagnose like a radiologist: Attention guided convolutional neural network for thorax disease classification". *arXiv preprint arXiv:1801.09927*.

Guendel, Sebastian et al. (2018). "Learning to recognize abnormalities in chest x-rays with location-aware dense networks". *Iberoamerican Congress on Pattern Recognition*. Springer, pp. 757–765.

Hallifax, RJ and Rahman, NM (2015). *Epidemiology of pneumothorax–finally something solid out of thin air*.

He, Kaiming et al. (2016). "Deep residual learning for image recognition". *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hoffman, Judy et al. (July 2018). "CyCADA: Cycle-Consistent Adversarial Domain Adaptation". *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, pp. 1989–1998. URL: http://proceedings.mlr.press/v80/hoffman18a.html.

Huang, Gao et al. (2017). "Densely connected convolutional networks". *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

Hwang, Eui Jin et al. (2019). "Development and validation of a deep learning–based automated detection algorithm for major thoracic diseases on chest radiographs". *JAMA network open* 2.3, e191095–e191095.

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

Irvin, Jeremy et al. (2019). "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison". *arXiv preprint arXiv:1901.07031*.

Johnson, Alistair EW et al. (2019). "MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports". *Scientific Data* 6.

Kamnitsas, Konstantinos et al. (2017). "Unsupervised domain adaptation in brain lesion segmentation with adversarial networks". *International conference on information processing in medical imaging*. Springer, pp. 597–609.

Kim, Dong Wook et al. (2019). "Design characteristics of studies reporting the performance of artificial intelligence algorithms for diagnostic analysis of medical images: results from recently published papers". *Korean journal of radiology* 20.3, pp. 405–410.

Kingma, Diederik P and Ba, Jimmy (2014). "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*.

Koonsanit, Kitti et al. (2017). "Image enhancement on digital x-ray images using N-CLAHE". *2017 10th Biomedical Engineering International Conference (BMEiCON)*. IEEE, pp. 1–4.

Kornblith, Simon; Shlens, Jonathon, and Le, Quoc V (2019). "Do better imagenet models transfer better?" *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2661–2671.

Kouw, Wouter Marco and Loog, Marco (2019). "A review of domain adaptation without target labels". *IEEE transactions on pattern analysis and machine intelligence*.

Laine, Samuli and Aila, Timo (2016). "Temporal ensembling for semi-supervised learning". *arXiv preprint arXiv:1610.02242*.

Lin, Tsung-Yi et al. (2017). "Focal loss for dense object detection". *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988.

Litjens, Geert et al. (2017). "A survey on deep learning in medical image analysis". *Medical image analysis* 42, pp. 60–88.

Liu, Ming-Yu and Tuzel, Oncel (2016). "Coupled generative adversarial networks". *Advances in neural information processing systems*, pp. 469–477.

MacDuff, Andrew; Arnold, Anthony, and Harvey, John (2010). "Management of spontaneous pneumothorax: British Thoracic Society pleural disease guideline 2010". *Thorax* 65.Suppl 2, pp. ii18–ii31.

Mahajan, Vidur et al. (2020). "The Algorithmic Audit: Working with Vendors to Validate Radiology-AI Algorithms—How We Do It". *Academic Radiology* 27.1, pp. 132–135.

Majkowska, Anna et al. (2020). "Chest radiograph interpretation with deep learning models: Assessment with radiologist-adjudicated reference standards and population-adjusted evaluation". *Radiology* 294.2, pp. 421–431.

Nigam, Kamal and Ghani, Rayid (2000). "Analyzing the effectiveness and applicability of co-training". *Proceedings of the ninth international conference on Information and knowledge management*, pp. 86–93.

Noppen, Marc and De Keukeleire, Tom (2008). "Pneumothorax". *Respiration* 76.2, pp. 121–127.

Oakden-Rayner, Luke (Feb. 2019). *Half a million x-rays! First impressions of the Stanford and MIT chest x-ray datasets*. URL: https://lukeoakdenrayner.wordpress.com/2019/02/25/half-a-million-x-rays-first-impressions-of-the-stanford-and-mit-chest-x-ray-datasets/.

Pan, Ian; Agarwal, Saurabh, and Merck, Derek (2019). "Generalizable inter-institutional classification of abnormal chest radiographs using efficient convolutional neural networks". *Journal of digital imaging*, pp. 1–9.

Park, Sohee et al. (2019). "Application of deep learning–based computer-aided detection system: detecting pneumothorax on chest radiograph after biopsy". *European radiology*, pp. 1–8.

Pasa, F et al. (2019). "Efficient deep network architectures for fast chest x-ray tuberculosis screening and visualization". *Scientific reports* 9.1, pp. 1–9.

Pérez, Óscar and Sánchez-Montañés, Manuel (2007). "A new learning strategy for classification problems with different training and test distributions". *International Work-Conference on Artificial Neural Networks*. Springer, pp. 178–185.

Petrochuk, Michael (2019). *HParams: Hyperparameter management solution*. https://github.com/PetrochukM/HParams.

Putha, Preetham et al. (2018). "Can Artificial Intelligence Reliably Report Chest X-Rays?: Radiologist Validation of an Algorithm trained on 2.3 Million X-Rays". *arXiv preprint arXiv:1807.07455*.

Rajpurkar, Pranav et al. (2017). "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning". *arXiv preprint arXiv:1711.05225*.

Rajpurkar, Pranav et al. (2018). "Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists". *PLoS medicine* 15.11, e1002686.

Ranjan, Ekagra et al. (2018). "Jointly Learning Convolutional Representations to Compress Radiological Images and Classify Thoracic Diseases in the Compressed Domain".

Raoof, Suhail et al. (2012). "Interpretation of plain chest roentgenogram". *Chest* 141.2, pp. 545–558.

Rosenberg, Chuck; Hebert, Martial, and Schneiderman, Henry (2005). "Semi-supervised self-training of object detection models." *WACV/MOTION* 2.

Rubin, Jonathan et al. (2018). "Large scale automated reading of frontal and lateral chest x-rays using dual convolutional neural networks". *arXiv preprint arXiv:1804.07839*.

Sahn, Steven A and Heffner, John E (2000). "Spontaneous pneumothorax". *New England Journal of Medicine* 342.12, pp. 868–874.

Saito, Takaya and Rehmsmeier, Marc (2015). "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". *PloS one* 10.3.

Selvaraju, Ramprasaath R et al. (2017). "Grad-cam: Visual explanations from deep networks via gradient-based localization". *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.

Shimodaira, Hidetoshi (2000). "Improving predictive inference under covariate shift by weighting the log-likelihood function". *Journal of statistical planning and inference* 90.2, pp. 227–244.

Simard, Patrice Y; Steinkraus, David; Platt, John C, et al. (2003). "Best practices for convolutional neural networks applied to visual document analysis." *Icdar*. Vol. 3. 2003.

Sun, Yu et al. (2019). "Unsupervised Domain Adaptation through Self-Supervision". *arXiv preprint arXiv:1909.11825*.

Tan, Mingxing and Le, Quoc (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". *International Conference on Machine Learning*, pp. 6105–6114.

Tang, Yu-Xing et al. (2019). "Deep adversarial one-class learning for normal and abnormal chest radiograph classification". *Medical Imaging 2019: Computer-Aided Diagnosis*. Vol. 10950. International Society for Optics and Photonics, p. 1095018.

Tarvainen, Antti and Valpola, Harri (2017). "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". *Advances in neural information processing systems*, pp. 1195–1204.

Taylor, Andrew G; Mielke, Clinton, and Mongan, John (2018). "Automated detection of moderate and large pneumothorax on frontal chest X-rays using deep convolutional neural networks: A retrospective study". *PLoS medicine* 15.11, e1002697.

Wang, Mei and Deng, Weihong (2018). "Deep visual domain adaptation: A survey". *Neurocomputing* 312, pp. 135–153.

Wang, Xiaosong et al. (2017). "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases". *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2097–2106.

Xie, Qizhe et al. (2020). "Self-training with Noisy Student improves ImageNet classification". *arXiv preprint arXiv:1911.04252*.

Xu, Xiang et al. (2019). "d-SNE: Domain adaptation using stochastic neighborhood embedding". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2497–2506.

Yarmus, Lonny and Feller-Kopman, David (2012). "Pneumothorax in the critically ill patient". *Chest* 141.4, pp. 1098–1105.

Yarowsky, David (1995). "Unsupervised word sense disambiguation rivaling supervised methods". *33rd annual meeting of the association for computational linguistics*, pp. 189–196.

Zauner, Christoph (2010). "Implementation and benchmarking of perceptual image hash functions".

Zeiler, Matthew D and Fergus, Rob (2014). "Visualizing and understanding convolutional networks". *European conference on computer vision*. Springer, pp. 818–833.