

WORKING WITH SMALL DATASETS



A thesis submitted towards the partial fulfillment of
BS-MS dual degree programme
from AUGUST 2020 to MAY 2021

by

JITESH SETH

under the guidance of

VIRAJ KULKARNI

DEEPTeK INC.

INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH
PUNE



Certificate

This is to certify that this dissertation entitled “WORKING WITH SMALL DATASETS” submitted towards the partial fulfilment of the BS-MS degree at the Indian Institute of Science Education and Research, Pune represents original research carried out by **Jitesh Seth** at **DeepTek Inc.**, under the supervision of **Viraj Kulkarni** during academic year August 2020 to May 2021.

Supervisor:
VIRAJ KULKARNI
CHIEF DATA
SCIENTIST
DEEPTeK INC.
DATE:
15/05/2021

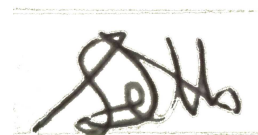
JITESH SETH
20161101
BS-MS
IISER PUNE
DATE:
15/05/2021

Declaration

I, Jitesh Seth, hereby declare that the matter embodied in the report titled “WORKING WITH SMALL DATASETS” is the results of the investigations carried out by me at DEEPTeK INC. from the period 15-08-2020 to 15-05-2021 under the supervision of Viraj Kulkarni and the same has not been submitted elsewhere for any other degree.



Supervisor:
VIRAJ KULKARNI
CHIEF DATA
SCIENTIST
DEEPTeK INC.
DATE:
15/05/2021



JITESH SETH
20161101
BS-MS
IISER PUNE
DATE:
15/05/2021

Dedicated to my father, who would have been proud of me today.

Acknowledgements

The year 2020 and the COVID-19 pandemic brought a lot of uncertainty and anxiety among all the students. Fellowships for PhDs, Master's theses and summer projects got cancelled at the last moment. I feel extremely privileged of having the good fortune to find DeepTek, a place with the most understanding people, with a computational project that could be done remotely.

First and foremost, I thank my mentor Rohit Lokwani and supervisor Viraj Kulkarni. Their suggestions and criticisms helped me to go from a complete outsider in the field of deep learning and medical image analysis to the point where I could conduct original research. Rohit helped me keep pace during the occasional lulls, motivating me to do something every week to have something substantial over time. Dr Aniruddha Pant also guided me in the initial months of the project and recommended excellent resources to enter the field of deep learning.

My friends and family have supported me in more ways than I can count. I thank my mother, Geeta Seth, for leading by example and showing me what a strong work ethic looks like. Thanks to my sister Ruchika Seth and my dear friend Jhelam Deshpande for always being up for discussions on statistics, broader aims, thesis writing and such. Even though they are from different fields, their suggestions proved invaluable. My friend Arya Samanta discussed some ideas with me as well.

I owe my entire life to my institute, IISER Pune. I am sure that any path that I take in my future, I will be able to trace it back to some learnings from here. My classmates and faculty members created such a holistic environment that encourages achievement beyond our perceived capabilities. I would especially like to thank Dr Anindya Goswami and the Data Science department for giving the 2016 BS-MS batch the opportunity for projects in data science and for clarifying our doubts, no matter how small. I am also grateful to Dr Pranay Goel, my TAC member, for his suggestions throughout my project.

I also thank Kishore Vaigyanik Protsahan Yojana (KVPY) for their financial support during my BS-MS.

I believe that I have more than what I would deserve only based on my hard work. Some call it faith in God or stars or fate. I do not know the truth, but I think several events worked out in my favour purely on chance. For being kind to me, thank you, luck.

Abstract

Deep learning semantic segmentation algorithms can localise abnormalities or opacities from chest radiographs. However, collecting and annotating training data is expensive and requires expertise which remains a bottleneck for algorithm performance. We investigate the effect of image augmentations on reducing the requirement of labelled data in the semantic segmentation of chest X-rays for pneumonia detection. We train fully convolutional network models on subsets of different sizes from the total training data. We apply a different image augmentation while training each model and compare it to the baseline trained on the entire dataset without augmentations. We find that rotate and mixup are the best augmentations amongst rotate, mixup, translate, gamma and horizontal flip, wherein they reduce the labelled data requirement by 70% while performing comparably to the baseline in terms of AUC and mean IoU in our experiments. Further, we try a semi-supervised learning approach called pseudo-labelling on the same segmentation model. The approach makes use of unlabelled data and augmentations to enhance the performance of the model. Using the labels of only 8% of the data, we show that it is possible to achieve a similar IoU as the previous experiments.

Contents

1	Introduction	8
1.1	Background	8
1.1.1	Computer-Aided Diagnosis in Radiology	8
1.1.2	Classification and Segmentation	9
1.1.3	Data Availability for Training Neural Networks	10
1.1.4	Learning a Lot from a Little	10
1.2	Related Work	12
1.3	Approach	14
2	Basics of Convolutional Neural Networks	15
2.1	Supervised Learning	15
2.2	The Building Blocks of CNNs	16
2.2.1	Model Architectures	16
2.2.2	Layers	17
2.2.3	Loss Functions and Metrics	20
2.2.4	The Learning Process	23
2.2.5	Training, Validation and Test Sets	23
3	Methods	25
3.1	Data	25
3.1.1	RSNA Pneumonia Dataset	25
3.1.2	Other Datasets	26
3.2	Model Architecture and Details	26
3.2.1	Augmentations	28
3.3	Pseudo-labelling	29
3.4	Implementation	31
3.4.1	Batch Generator	31
3.4.2	Custom Utils Library	31

4 Results and Discussion	33
4.1 The model is able to learn within thirty epochs	33
4.2 Comparison of performances after augmentation	33
4.3 Pseudo-labelling helps in segmentation but not in classification	39
4.4 Discussion and Conclusion	39
References	41

List of Figures

1.1	Classification and Segmentation	9
2.1	A visual description of convolution	19
2.2	The residual connection	19
2.3	Modern Activation Functions: ReLU and Leaky ReLU	21
2.4	A visual representation of intersection over union	22
3.1	Examples from RSNA Pneumonia Dataset	26
3.2	The architecture of the fully convolutional neural network that we used. (a) The encoder part and (b) The decoder part.	27
3.3	Original CXR with opacities labelled(left) and augmentations with updated bounding boxes.	28
3.4	(Top) original CXRs for mixup; (bottom left) result of mixup augmentation and (bottom right) corresponding mask.	29
3.5	Soft labels for pseudo-labelling	30
4.1	Model performance as a function of epochs	34
4.2	Two examples of CXR with the ground truth mask (left) and predicted mask by the baseline model (right)	35
4.3	Mean IoU performance of the models	37
4.4	AUC ROC performance of the models	38

List of Tables

2.1	The confusion matrix.	21
4.1	AUC ROC of different augmentations	35
4.2	p-values of DeLong Test	36
4.3	AUC and mean IoU for semi-supervised learning approaches	39

Abbreviations Used

- **AUC**: Area Under the Curve
- **BCE**: Binary Crossentropy
- **CADx**: Computer-Aided Diagnosis
- **CNN**: Convolutional Neural Network
- **CXR**: Chest X-Ray
- **IoU**: Intersection over Union
- **ReLU**: Rectified Linear Unit
- **ROC**: Receiving Operator Characteristics
- **RSNA**: Radiology Society of North America
- **SSL**: Semi-supervised Learning

Work from this thesis has been used for the research paper titled “Reducing Labelled Data Requirement for Pneumonia Segmentation using Image Augmentations” available on arXiv (<https://arxiv.org/abs/2102.12764>). The same has been accepted for publication at the sixth International Conference on ICT for Sustainable Development (ICT4SD, 2021, <https://ict4sd.org/>).

Chapter 1

Introduction

This chapter will introduce deep learning in medical imaging analysis and cover a few studies and their results. I will discuss the problem of data availability in this area and a few approaches of tackling the same. Although my focus would be on chest radiography wherever possible, I have included several ideas of our interest in other deep learning areas. In the end, I introduce my problem statement and discuss the scope of my project.

1.1 Background

1.1.1 Computer-Aided Diagnosis in Radiology

Computer-aided diagnosis (CADx) for chest radiography started around the 1960s. Diagnostics and other image processing techniques used rule-based reasoning or algorithms such as edge detection, region growing, fitting geometrical models or dynamic programming ([Ginneken *et al.*, 2001](#)). For these processes, experts would convert the chest radiograph into a sequence of numbers that the computer could manipulate - what is now known as feature vectors. Constructing such feature vectors would require in-depth domain knowledge and still have high inter-observer variability. This is where deep learning has transformed and dominated the field ([McBee *et al.*, 2018](#); [van Ginneken, 2017](#)). Deep neural networks not only efficiently map the feature vectors to the labels but also create the most optimal feature extractors on their own. [van Ginneken \(2017\)](#) has discussed the evolution of CADx from rule-based and machine learning approaches to deep learning ones by considering a few examples in the field.

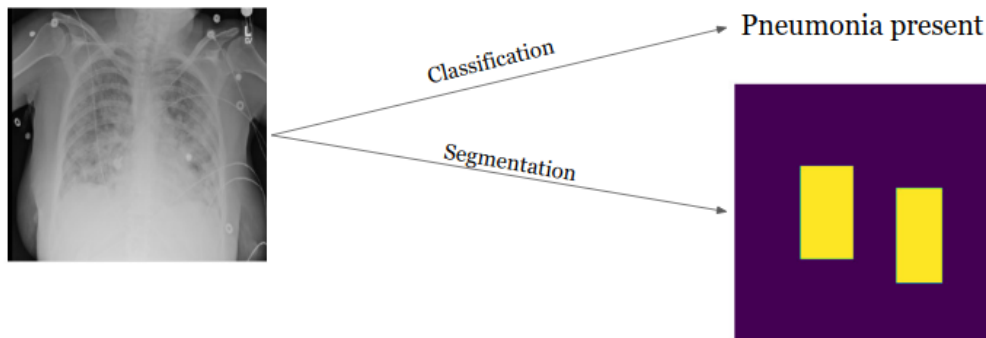


Figure 1.1: Classification produces a single class label for each image (e.g. pneumonia present or absent), whereas segmentation produces a mask of the same size as the image, with each pixel in the mask corresponding to a label. In this image

1.1.2 Classification and Segmentation

Deep learning approaches in CADx use Convolutional Neural Networks (CNNs) for two major purposes - classification and segmentation. In classification, each image is given a label or assigned to a class. In segmentation models, each pixel is assigned to a class. Thus for each image, the model predicts a corresponding mask (see Fig. 1.1). Both the approaches have shown great success in radiographs like chest X-rays and CT scans (Lakhani and Sundaram, 2017; Dunnmon *et al.*, 2018).

Research on pathology identification in chest X-rays (CXRs) has mainly focussed on classification. This is useful when we need models to predict a class label from a broad set of pathologies. For example, the dataset CheXpert (Irvin *et al.*, 2019) has CXRs corresponding to fourteen different pathologies. Annotating data for classification is also a more manageable task now - CheXpert was annotated using a label extraction algorithm. Even for manual annotation, radiologists only need to identify the pathology, as opposed to marking the regions in the CXR in the case of segmentation.

The disadvantage of classification models is their lack of interpretability. If we classify an entire image with a single label, we do not know the regions in the CXRs responsible for the class label. We have to unpack the neural network to understand which pixels it pays the most attention to while making decisions. This is achieved by an additional step of plotting saliency maps or gradient class activation maps and then interpreting how the CNN is making decisions (Pasa *et al.*, 2019). Moreover, we might find that the algorithm focuses on irrelevant regions to make decisions, such as areas out-

side the body. In such a case the CNN would no longer be trustworthy, even if it is accurate.

Segmentation and object detection algorithms provide an advantage over standard CNNs because they are trained directly on the regions of interest, which they can then predict. Semantic segmentation architectures such as U-Net (Ronneberger *et al.*, 2015) construct a free-form area onto the object of interest, whereas object detection algorithms such as YOLO (Redmon *et al.*, 2016) detect the presence of an object and output the coordinates of a rectangular bounding box to mark it. Semantic segmentation models are more useful than classification models in two ways: they require less training data since they have pixel-level labels for every image, and they can assist radiologists in their work by localising the abnormalities (Hurt *et al.*, 2020). We have chosen semantic segmentation as the focus of this study.

1.1.3 Data Availability for Training Neural Networks

The cost and effort to label data for supervised learning algorithms is often a significant constraint on training models that perform well in practice (Prevedello *et al.*, 2019). There have been efforts on two fronts. First, the amount of labelled data available is increasing manifold, as more research groups create massive datasets such as CheXpert (Irvin *et al.*, 2019) and ChestX-ray8 (Wang *et al.*, 2017) into the public domain. As the popularity of artificial intelligence in healthcare increases, companies are also investing in annotating more data. Second, research groups are creating techniques that can learn better from small amounts of labelled data. A few examples are image augmentations, semi-supervised learning, special architectures such as U-Nets (Ronneberger *et al.*, 2015), GANs (Goodfellow *et al.*, 2014) and others.

1.1.4 Learning a Lot from a Little

In this section, we discuss a few techniques to learn from a limited amount of labelled data. This list is in no manner exhaustive but is a review of the various approaches we can take to tackle low data availability.

Image Augmentations

Image augmentations is a vital data processing technique to improve the performance of machine learning models. They can make the CNN indifferent to naturally present variations in the data, such as position, scale, or different radiography equipment. Augmentations can be of varied types. Rotate,

scale and flip and similar augmentations are called geometrical augmentations. Photometric augmentations transform the colour space of the images. More complex transformations include elastic deformation or mixing multiple images. However, inflating the dataset with numerous augmentations would add to the training time and compute requirements without necessarily increasing performance. A drawback of augmentations is that they may cause overfitting by making the CNN invariant to some features but highly tailored to the training data in others ([Shorten and Khoshgoftaar, 2019](#)).

The problem of augmentations on the performance of segmentation models in the medical domain is not sufficiently addressed in research. Knowledge of specific augmentations which reduce the labelled data requirements will help researchers and data scientists fine-tune their models faster and better. Moreover, data augmentation studies investigate the increase in model performance rather than the decrease in the labelled training data requirements. Thus there is an opportunity to fill the gap in this space.

Semi-Supervised Learning

Though there is a paucity of labelled training data in medical images, the effort in obtaining unlabelled data is far lesser. Unsupervised learning methods such as clustering algorithms could discover structure in the data that could enhance the discriminatory ability of supervised algorithms. This is the basis of semi-supervised learning (SSL), which makes use of both labelled and unlabelled data.

There are many approaches for SSL. For example, we could train a supervised model on the available labelled data, then use it to label the unlabelled data. Then, using this larger dataset, we train the model again. This is known as self-training or pseudo-labelling ([Lee, 2013](#)). Another approach could be to use a clustering algorithm to build a graph out of the inputs and propagate the labels from labelled data to their nearest neighbours, then further neighbours and so on till the entire graph is labelled.

Though a lot of statistically robust SSL techniques have been built (see [Zhu and Goldberg \(2009\)](#) for a detailed introduction to SSL), the combination of SSL and Deep Learning in medical images is still in its infancy. We attempted pseudo-labelling to medical images to see if we get an improvement in performance.

1.2 Related Work

This section covers the recent advancements in CADx using deep learning, image augmentations and semantic segmentation in the field of CXRs.

[Kermany *et al.* \(2018\)](#) showed the generalizability of deep neural networks using the same neural network to classify retinal Optical Coherence Tomography images and pediatric pneumonia classification in CXRs. The latter model achieved an accuracy of 92.8%, and the area under the ROC curve was 96.8%.

Recent studies have proven that image augmentations improve the machine learning model performance. [Sirazitdinov *et al.* \(2019\)](#) showed the effect of augmentations on the classification of chest radiographs. They concluded that a combination of increasing brightness, random rotation and horizontal flips led to the best performance on the ChestX-Ray14 dataset, with an area under the curve of the receiving operator characteristics (AUC-ROC) of 0.808 (compared to 0.785 without any augmentations). However, they do not quantify the extent of each augmentation, thus decreasing its reproducibility. After the invention of mixup ([Zhang *et al.*, 2018](#)), [Eaton-Rosen *et al.* \(2018\)](#) applied it to a dataset of MRI images. They provide a graphical overview of mixup compared to other augmentations and a baseline for a large (199 images) and a small (10 images) dataset.

[Souza *et al.* \(2019\)](#) created an automatic method for segmentation and reconstruction of lungs, which can take into account lung opacities from pneumonia or tuberculosis, reconstruct the lung boundaries, and finally, segment the lungs. They used the segmented lungs for a classification model, which achieved an accuracy of 96.97%, an average Dice coefficient of 0.94 on the Montgomery County’s Tuberculosis Control dataset ([Jaeger *et al.*, 2014](#)). [Selvan *et al.* \(2020\)](#) tackled the same problem by treating high opacity regions as missing data and using a variational auto-encoder for data imputation. They achieved an accuracy of 88.15% and a Dice coefficient of 0.8503 on a curated CXR dataset. Thus segmentation was mainly used to demarcate the lungs for use in classification models. However, we can also use semantic segmentation algorithms to demarcate lung opacities. This opportunity became plausible after the publication of the Radiology Society of North America’s (RSNA) pneumonia dataset ([Shih *et al.*, 2019](#)), also used in this study.

[Wu *et al.* \(2020\)](#) took the concept of lung segmentation and opacity detection one step forward. They segmented both the lungs, divided them into three zones each and predicted the presence of pneumonia in each zone using the patient’s radiology report. Thus they created an object detection dataset from radiology reports. Using this dataset, they trained a RetinaNet model

and tested it on the RSNA data set. The model had a mean IoU of 0.29 per pneumonia positive image. [Hurt *et al.* \(2020\)](#) have shown that semantic segmentation of CXRs can be used as a probability map to interpret the radiographs. Their segmentation model on the RSNA dataset showed a dice coefficient of 0.603, and the classification had an AUC of 0.854.

[Lee \(2013\)](#) introduced the concept of pseudo-labelling by showing its effectiveness on the MNIST dataset. They used t-SNE visualisation to show how the images form well-separated clusters, which the CNN can identify. This assumption that classes belong to dense clusters in the representation space and the separation boundaries pass through regions of less density is a common assumption in semi-supervised learning ([Zhu and Goldberg, 2009](#)). [Arazo *et al.* \(2020\)](#) have shown some pitfalls of pseudo-labelling and have identified regularisation techniques to prevent overfitting the labelled data and show state-of-the-art performance. They used the labels of only 500 images from the CIFAR-10 dataset and trained on soft labels along with mixup augmentation. They achieve an error rate of $8.8 \pm 0.45\%$, much lower than other studies with the same starting point. However, both ([Lee, 2013](#); [Arazo *et al.*, 2020](#)) use small images of 32×32 pixels for their studies and what works there need not work in the high-resolution medical image analysis regime. [Zou *et al.* \(2020\)](#) have built a pseudo-labelling platform for segmentation and shown a mean IoU of 73.23 on the VOC12 dataset.

[Peikari *et al.* \(2018\)](#) brought the cluster-then-label approach to breast pathology classification. It is an interesting approach where they map the images to a feature space and use the OPTICS algorithm to identify clusters among them. Then a support vector machine is trained to learn classification. They have not used deep learning approaches for building the feature detector, nor have they used a deep learning classifier. There is a ripe opportunity here to do so.

An essential aspect of deep learning in CADx is the usability of the models - we do not desire clever models that might be clinically irrelevant ([Lundervold and Lundervold, 2019](#)). For example, both in Pan and Cadrin-Chênevert’s model and Cheng’s model of the RSNA dataset ([Pan *et al.*, 2019](#)), they systematically decreased the predicted bounding boxes by 12-17%, which increased the performance for the particular test set, but there is no medically relevant reason to do the same in practical settings. Another practical constraint with Pan’s models was extensive ensembling, which requires the availability of high-end GPUs.

1.3 Approach

In this thesis, we explore how augmentations and SSL methods can help reduce the requirements of labelled data. We implement five different augmentations on the training of CNN models on Chest X-rays. We propose three criteria for identifying augmentations that reduce labelled data requirement. First, the model should perform comparably to the baseline on a subset of the data. Second, the models with augmentation trained on partial data should perform better than models without any augmentations trained on the same data. Third, the model should satisfy the criteria above for multiple test sets. We validated our results with an in-sample and out-of-sample test set. In the second set of experiments, we use a similar model as [Arazo *et al.* \(2020\)](#) to the same dataset and observe if training on unlabelled data improves over a baseline trained on a small amount of labelled data.

The thesis is organised as follows: Chapter 2 discusses the fundamentals of CNNs as they apply to my study. Chapter 3 is about the data and the methodology, with a note on implementation. Finally, Chapter 4 lays down the results of the experiments and the discussion.

Chapter 2

Basics of Convolutional Neural Networks

This chapter covers the theory behind the convolutional neural networks (CNNs) that we used throughout our study. We do not intend it as a primer on CNNs but rather a description of the underlying theory behind the project. The objective is to showcase the various choices available at every step and discuss why we selected one particular method.

Several concepts described in this chapter are from *Deep Learning* (Goodfellow *et al.*, 2016), an advanced text in the field. Andrew Ng’s “Deep Learning Specialisation” course¹ also covers essential concepts discussed in this chapter.

2.1 Supervised Learning

Supervised Learning is a subsection of a larger category of algorithms known as machine learning. The definition of machine learning is given in definition 1. Based on that, we define supervised learning as per definition 2.

Definition 1 (Machine Learning). *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E . (Mitchell, 1997)*

Definition 2 (Supervised Learning). *A quantity x and its corresponding label y are related by an unknown function $y = f(x)$. The task T is to find a function \hat{f} that approximates f and predicts the labels \hat{y} well according to*

¹<https://www.coursera.org/specializations/deep-learning>

performance measure P . The experience E for the algorithm is going through multiple (x, y) pairs to find \hat{f} .

As we can see, definition 2 is ambiguous. We still need to elucidate how we can estimate the function \hat{f} and how going through multiple (x, y) pairs plays a role in that. We still have to clarify what ‘going through an (x, y) pair’ means. The answer lies in convolutional neural networks. A CNN itself is the function \hat{f}_θ , with several parameters θ (CNNs with parameters in the order of 10^6 to 10^9 are commonplace nowadays). These parameters have to be tuned according to the dataset to get accurate predictions. The predictions $\hat{y} = \hat{f}_\theta(x)$, are used to calculate a cost function $\mathcal{C}(y, \hat{y}) \geq 0$ which is designed to be large if the performance measure P is low and 0 if all predictions by \hat{f} are correct. Therefore, the problem of finding the correct parameters is reduced to minimising the cost function.

2.2 The Building Blocks of CNNs

Convolutional Neural Networks have several working parts that come together to become capable of state-of-the-art performance on their tasks. In this section, we begin with the highest level - that of model architecture and zoom in to the layers that build up the architecture. Then we discuss how to calculate the cost function, how to minimise the cost and how to judge model performance. Finally, we end the chapter with a discussion of the significance of the three sets involved in an end to end machine learning task - the training, validation and test sets.

2.2.1 Model Architectures

A CNN is a composition of functions $f_1(f_2(f_3(\dots(f_d(x)))))$. The order of the f_i 's determines how well the model will perform, and so does the *depth* of the CNN d . After a lot of research and trial-and-error, the deep learning community has developed a range of such orderings that perform well in practice. These orderings are called model architecture.

The basic structure of a CNN is relatively standard nowadays. It consists of an **encoder** that builds representations from the input image. In this study, the input $x \in \mathbb{R}^{n \times n}$ refers to a square CXR image of width n pixels. The subsequent transformations and label depend on the downstream task, classification or segmentation.

Classification

In classification, the representations are sent through a **classifier**, which is usually a fully connected network. It classifies the input into one of k classes. Thus the label $y \in \{1, 2, \dots, k\}$.

[Lecun *et al.* \(1998\)](#) built the first CNN classifier, called LeNet-5, which took images of 32×32 pixels containing handwritten digits 0-9 (the MNIST dataset) and classified them correctly. This model had 60,000 trainable parameters. Since then, the number of parameters has grown manifold, and the increase in computational power has supported it. The best performing classification architectures today, for example, ResNet-50([He *et al.*, 2015a](#)) has about 23 million parameters.

Segmentation

In segmentation, the representations are transformed into new information through a **decoder** network. The output label $y \in \mathbb{R}^{n \times n}$, and $y_{i,j} \in \{1, 2, \dots, k\} \forall i, j \in \{1, \dots, n\}$, i.e. the output mask is the same dimensions as the input, with each pixel belonging to one of the k classes.

Some of the well known deep learning architectures for segmentation include Fully Convolutional Networks (FCNs) ([Long *et al.*, 2015](#)) and the U-Net([Ronneberger *et al.*, 2015](#)). U-net became the standard in semantic segmentation after the group won the ISBI cell tracking challenge 2015 in the segmentation of neuronal structures in electron microscopic stacks category. They trained the U-net on tiny dataset of 30 images and made use of extensive data augmentation.

2.2.2 Layers

Layers are the primary building blocks of all neural networks. They take input and apply a function to it. The function is often as simple as a linear transformation or an addition, but the stacking up of these layers on top of each other lets the CNN understand complex data. We discuss the layers used in our study below.

Dense Layer

A dense or fully-connected layer is the simplest and oldest form of layers in artificial neural networks. If the input is a vector $\mathbf{x} \in \mathbb{R}^n$ and the desired output is a vector $\mathbf{y} \in \mathbb{R}^m$, this layer implements the function

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.1}$$

Where σ is an activation function (discussed below) and $W \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ are the parameters of the dense layer. Therefore a dense layer with an n -dimensional input and an m -dimensional output has $(n+1) \times m$ parameters. The number of trainable parameters grows very fast with increasing input size, thus increasing computational costs. It is also not necessary for every input element to be connected to every output element. Thus in modern neural network architectures, dense layers are only found in the final layers for classifying the features extracted by the convolution layers.

The Convolution Layer and Transposed Convolution Layer

As is evident from the name, the convolution operation is the essential operation in a CNN. It utilizes the spatial information in a picture, building on the intuition that a square of side 5 pixels cropped from an image can give more information than 25 randomly selected pixels. A convolution involves a matrix called a kernel or a filter, usually a square with an odd side length.

The kernel behaves as a feature detector. The initial convolutions detect edges, loops and such low-level features. The convolution operations in the deeper layers build upon these low-level features to detect higher-order features such as parts of objects and, further in, objects themselves. The convolution of an image I with a kernel K of size (m, n) , denoted by $(I * K)$ is given by equation 2.2. A visual representation of convolution is shown in Fig. 2.1.

$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K \quad (2.2)$$

Another crucial layer in a segmentation architecture is the transposed convolution, also known as the deconvolution layer. It forms a part of the decoder network and is used to go from a small-sized input to a larger sized output. A transposed convolution can be thought of as the gradient of a convolution. If convolution on image I by kernel K produces the output I' , transposed convolution on I' by kernel K will produce I (Dumoulin and Visin, 2016).

The Residual Connection

This is a special connection that was introduced in ResNets. It adds a previous layer directly to the current layer. This makes it easy for the neural network to learn the identity function, ensuring that the model performance does not deteriorate, even if it does not improve. Mathematically, if a layer x goes through some convolution operations f , the residual connection returns

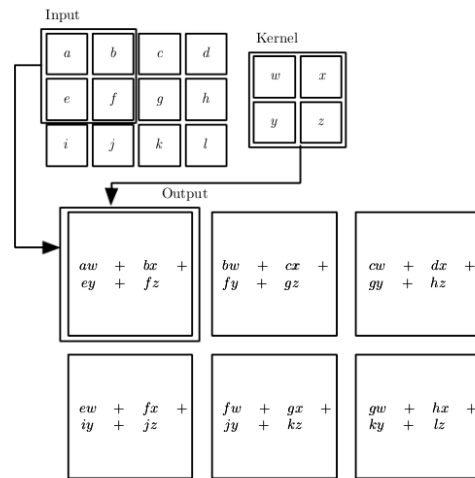


Figure 2.1: **A visual description of convolution:** An image of size 3×4 convolved with a kernel of size 2×2 results in an output of size 2×3 . Image retrieved from [Goodfellow *et al.* \(2016\)](#)

$y = f(x) + x$. If $f(x)$ and x do not have the same dimensions, the residual branch can also have convolutions or pooling operations to ensure it.

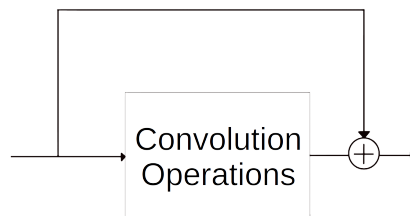


Figure 2.2: **The residual connection.**

Max pooling, Batch Normalisation, and Activation Functions

Max pooling is a method to reduce the dimensions of the layer. This helps to prevent overfitting and cuts computational costs. The algorithm for the max pooling layer is given below. As we can see, it reduces the number of elements in the matrix by a factor of m^2 . Usually, the stride and window size is 2, and thus there are no overlapping windows.

Max Pooling

```
Input: Matrix  $A_{n \times n}$ , stride  $m$ , window size  $w$ 
Initialise matrix  $C_{n/m \times n/m}$ 
For  $i$  in 0 to  $n$ , step size  $m$ :
    For  $j$  in 0 to  $n$ , step size  $m$ :
        Choose submatrix  $B$  from  $i$  to  $i + w$  row and  $j$  to  $j + w$  column
        Assign  $c_{i/m, j/m} = \max(B)$ 
Output  $C$ .
```

Batch Normalisation reparametrizes the layer in order to add multiplicative and additive noise. It is pretty useful for models with several layers and prevents overfitting. Activation functions add a non-linearity between two layers of the CNN. Without activation function, consecutive convolution layers can be combined since a composition of two linear transformations is a linear transformation. In such a case adding more layers would be meaningless.

In the initial days of deep learning, the most common activation functions were sigmoidal, such as $\arctan(x)$ and $\sigma(x) = (1 - e^{-x})^{-1}$. Nowadays, the piecewise linear functions such as Rectified Linear Unit (ReLU) or leaky ReLU (Fig. 2.3) are preferred. These are much faster in usage since their derivative is easy to calculate.

Another non-linear layer used in CNNs is the Softmax layer. In CNNs, the output layer is a vector of k dimensions corresponding to the k labels. Each dimension will have numbers belonging to $(-\infty, \infty)$. In the case of a single-label problem, we need to convert these numbers into probabilities that sum to 1. This is achieved using the softmax function (equation 2.3).

$$\sigma(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}} \quad (2.3)$$

2.2.3 Loss Functions and Metrics

As mentioned at the beginning of section 2.1, we need to calculate the cost function $\mathcal{C}(y, \hat{y})$. The cost function is calculated for a batch of images by adding the per-image cost function, known as loss. Depending on the requirement of the model, we use different loss functions. Some general properties are desirable in every loss function, such as being convex. This leads to better optima and faster training times.

In the case where there are only two labels, the most well-known loss function is binary cross-entropy (BCE). BCE is calculated using equation

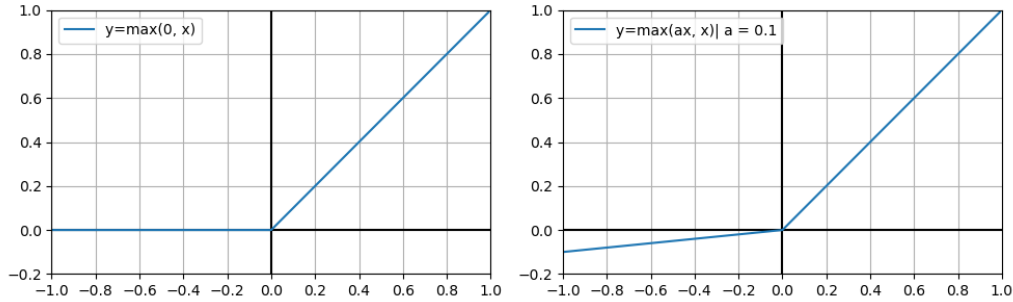


Figure 2.3: The modern activation functions. The Rectified Linear Unit (ReLU) (left) returns only the positive activations. Leaky ReLU (right) is a modification of it and returns some value even for $x < 0$. The formulae are given in the figure.

2.4, where y is the truth value (0 or 1), and \hat{y} is the probability predicted by the CNN.

$$H(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \quad (2.4)$$

BCE can be used as a loss function for every pixel in a segmentation problem with only two classes. Though there are other loss functions for segmentation, such as Dice loss or IoU loss, we found this loss function to be the best for our tasks.

		Ground Truth	
		Positive	Negative
Model Predictions	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Table 2.1: The confusion matrix.

The cost function is one indication of how well our model is performing. We can use several other indicators, which allows our results to be more interpretable and comparable to others. The most common metrics for classification are accuracy, precision, recall and F1 score. All of these can be calculated from the confusion matrix, shown in table 2.1. The definitions of these are given in equations 2.5 - 2.8.

$$\text{Accuracy:} \quad = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.5)$$

$$\text{Precision:} \quad = \frac{TP}{TP + FP} \quad (2.6)$$

$$\text{Recall:} = \frac{TP}{TN + FN} \quad (2.7)$$

$$\text{F1 Score:} = \frac{2TP}{2TP + FP + FN} \quad (2.8)$$

A problem with the above four metrics is that they are sensitive to the threshold chosen for classification. Another important metric is AUC-ROC, which does not rely on a chosen threshold. The ROC curve plots how the true positive rate changes with respect to the false positive rate as the threshold changes. For a random classifier, the curve would be the line $y = x$, and the area under the curve would be 0.5. The maximum possible area under the curve is 1. Thus AUC always lies between 0.5 and 1.

These metrics are helpful in the classification regime. For segmentation, a good measure is the mean intersection over union (IoU)(Fig. 2.4). It gives us an estimate of how good the overlap between predicted and ground truth masks. We chose mean IoU as our evaluation metric. It is calculated using equation 2.9.

$$IoU(A, B) = \frac{A \cap B}{A \cup B} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive} + \text{False Negative}} \quad (2.9)$$

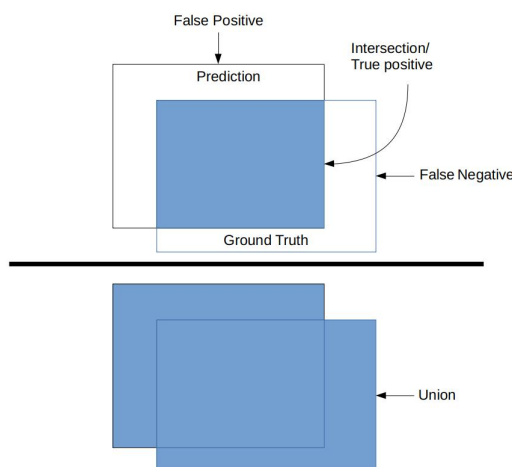


Figure 2.4: **A visual representation of intersection over union.** As the overlap increases, the intersection becomes larger, and the union becomes smaller, till at perfect overlap they are equal, and IoU is 1. At no overlap, the intersection is 0 and so is the IoU. Hence IoU varies from 0 to 1.

2.2.4 The Learning Process

Now that we have all the components in place, we can describe how the CNN learns, i.e. calculates the optimal parameters θ . First, all the parameters are initialised randomly from a specified distribution (such as He initialisation (He *et al.*, 2015b)). Using these parameters, $f_{\theta}(x)$ is calculated for a specified number of images, called a ‘batch’. This is known as forward propagation, as the calculation progresses from the first layer to the last.

Once we have the predictions for a batch, we can calculate the cost function for it. The gradient of this cost function is used to reduce the parameters by a small amount. This updation takes place from the last layer progressing sequentially to the first and is called backpropagation. Each pass through the entire dataset is called an epoch. The model trains for multiple epochs until the cost function converges and the cost stops decreasing. At this stage, the training is halted.

There are several methods of the parameter update step. As the field has evolved, more computationally efficient and faster methods have been developed. These methods are known as optimisers, and some of the most commonly used ones are Stochastic Gradient Descent, RMSProp and Adam. Adam (Kingma and Ba, 2017) is named after ‘adaptive moment estimation’ and is currently the most recommended method. It calculates the first and second moment of the gradient and changes the parameters accordingly.

2.2.5 Training, Validation and Test Sets

Any dataset for a machine learning project must be split into three sets. The **training set** contains about 70% of the total data and is used for the training of the CNN as described in the previous sections. Once the model is trained, we must ascertain its performance on data that it has never seen before. This is done by making a **test set** from a different source. If an external source is unavailable, we can partition about 20% of the data into the test set. Without a test set, we can not claim that the model has generalised the information that it has learnt, which is the hallmark of artificial intelligence.

In addition to the parameters, the model performance also relies heavily upon the choice of hyperparameters. These can be chosen by seeing which set of hyperparameters has the best performance. However, if we choose hyperparameters based on the test set performance, we are tuning our model to that particular test set, and thus we can not claim anymore that the model has generalised. To overcome this issue, we use the remaining 10% of the data as a **validation set**. The primary purpose of this set is to identify the best hyperparameter set. In addition, the validation set performance

is used to stop model training. It is observed in statistical learning that the cost function of the training set and validation set first decreases. That signifies that the model is learning generalisable features. After a point, the validation set loss starts to increase again, while the training error keeps decreasing. This is because the model starts to overfit the training data. Thus using the validation set performance or cost function, we can choose the model with the best features learnt and least overfitting.

Chapter 3

Methods

We illustrate the methods involved in our project in this chapter.

3.1 Data

3.1.1 RSNA Pneumonia Dataset

In order to develop techniques that worked with a small dataset, we needed a large dataset from which we could take different sized subsets. This would test the limits of the techniques.

We chose a publicly available dataset of chest X-Rays annotated for pneumonia-related lung opacities. This dataset, hereafter referred to as the RSNA dataset, was jointly annotated by the Radiology Society of North America (RSNA) and Society of Thoracic Radiology (STR) (Shih *et al.*, 2019). A team of eighteen radiologists labelled CXRs taken from the National Institutes of Health (NIH) CXR8 dataset (Wang *et al.*, 2017). The CXRs they chose either had pneumonia-like labels, such as “pneumonia”, “infiltration” and “consolidation”, or a “no findings” label, or labels other than the ones above. They annotated these again and labelled them as “Lung Opacity”, “Normal”, or “No Lung Opacity / Not Normal”, respectively. For each CXR with the “Lung Opacity” label, they demarcated the regions with opacities using bounding boxes.

The dataset was used for a competition called the RSNA Pneumonia Detection Challenge on the data science competition website Kaggle (<https://www.kaggle.com/c/rsna-pneumonia-detection-challenge>). About 1400 teams participated in the competition and used various types of convolutional neural networks for segmentation and object detection to identify the lung opacities.

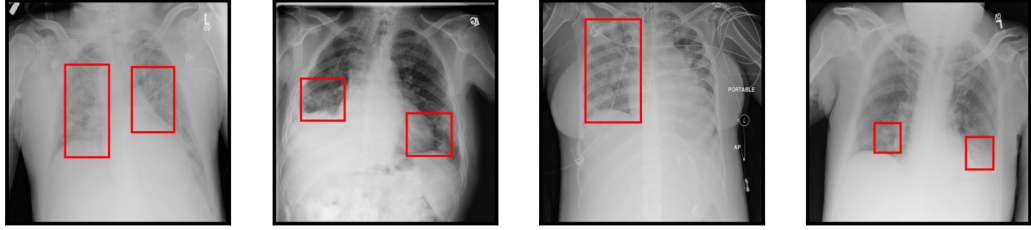


Figure 3.1: A few positive examples from the RSNA Pneumonia Dataset, shown with their bounding boxes.

Each image in the dataset is in the DICOM image format and has dimensions 1024×1024 . The labels are of two kinds - classification labels (as discussed above) and locations. The latter contains the coordinates of the bounding boxes demarcating the opacities. It is a list of 4-dimensional vectors, listing the coordinate of the top-left point, height and width, respectively, i.e. the form $(x_{min}, y_{min}, height, width)$. Examples of images with the bounding boxes are given in Fig. 3.1.

Out of the 26,684 images in the training dataset, we selected all the images with bounding boxes (n=6012) and randomly picked other images (n=8488) and split them into three sets with an equal prevalence of pneumonia (41.4%). Thus we obtained the training, test and validation set with 10,000, 3,000 and 1,500 images, respectively.

3.1.2 Other Datasets

We also curated CXRs from out-of-sample sources for testing our models. We obtained these images from Padchest (Bustos *et al.*, 2020) and four private hospitals and population screening programmes in India and Indonesia. This set also contains 3,000 images, out of which 1125 are pneumonia positive. The annotations are in the form of rectangular or polygonal bounding boxes.

3.2 Model Architecture and Details

We used a U-net-like CNN with depthwise separable convolutions style connections using the Keras library for this study (Chollet, 2019). We resize the CXR to (512,512,3) and use it as the input of the CNN. The network has an encoder and a decoder part, as shown in Fig. 3.2. The CNN uses residual connections, depthwise separable convolution and 2D convolution, max pooling, transpose convolution and 2D upsampling.

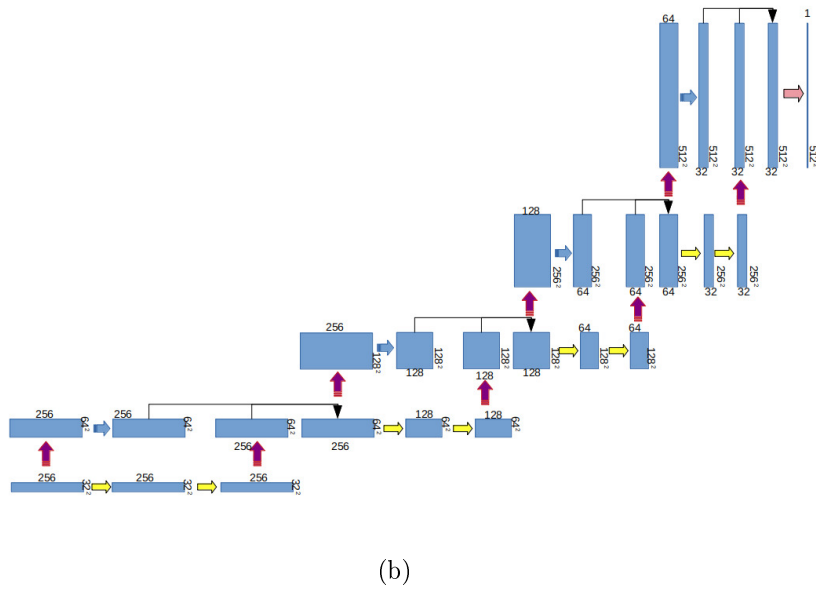
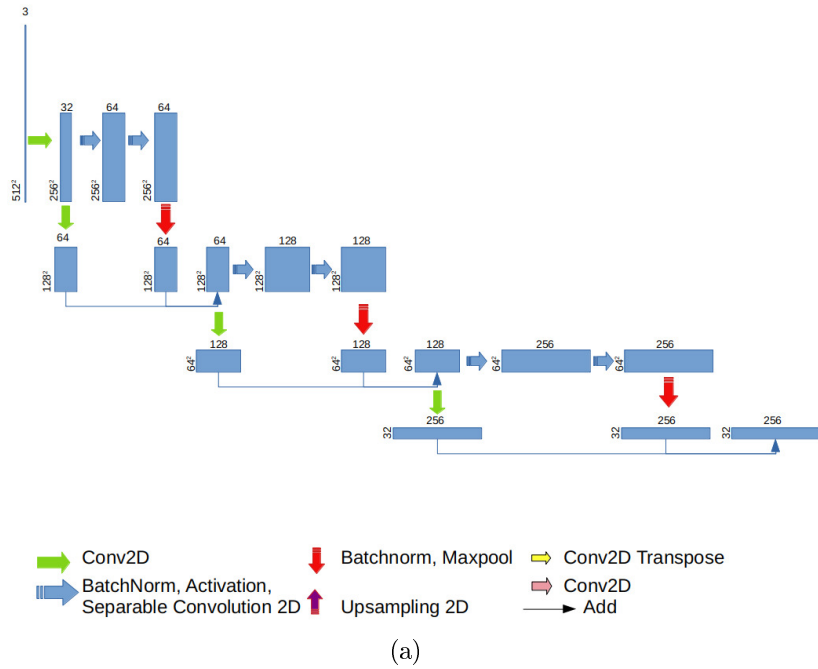


Figure 3.2: The architecture of the fully convolutional neural network that we used. (a) The encoder part and (b) The decoder part.

For training the CNN, we used binary cross-entropy (BCE) $H((y, \hat{y}))$ (equation 2.4) as the loss function,

We used the Adam optimiser with an initial learning rate of 10^{-3} and a learning rate scheduler called “ReduceLROnPlateau”. This scheduler decreased the learning rate by a factor of 10 if there was no improvement on validation loss since five epochs. We trained each model until it had shown no improvement in validation loss for at least five epochs. After training, we saved the weights of the model with the lowest validation loss.

Once the model was trained, we used it to predict the test set masks. These were compared to the ground truth masks to calculate the mean IoU, the Dice coefficient and the loss. If any pixel in the predicted or truth mask had an intensity > 0.5 , the mask as a whole was classified as pneumonia positive. Based on this classification, we calculated other metrics such as precision, recall, F1 score and binary accuracy.

3.2.1 Augmentations

We have chosen five image augmentations for this study. These are random rotation between -10° to 10° , changing the gamma between 0.75 to 1.25, translating the image randomly between 0-5% of its length in x and y direction, horizontal flips, and mixup (Zhang *et al.*, 2018). In mixup, two images and their masks (represented by x_1 and x_2) are combined using the formula:

$$x = x_1\lambda + x_2(1 - \lambda)$$

Where $\lambda \sim \beta(0.2, 0.2)$. These augmentations have negligible computational cost. Mixup results in better-performing segmentation models according to recent studies (Eaton-Rosen *et al.*, 2018).

We train the baseline (i.e. no augmentations) on 100% of the training set. For each of the six conditions - five augmentations and one with no augmentation (hereafter referred to as “NoAug”), models were trained using 30%, 50%, 70% and 90% of the training set.

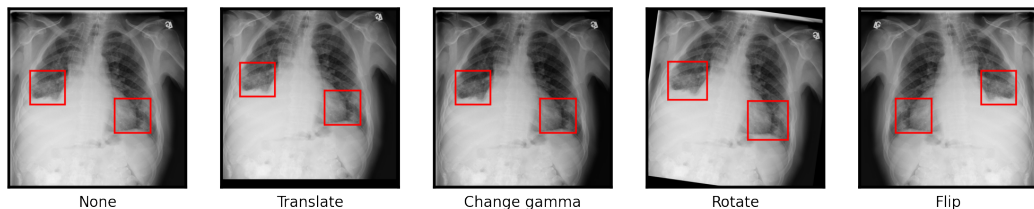


Figure 3.3: Original CXR with opacities labelled(left) and augmentations with updated bounding boxes.

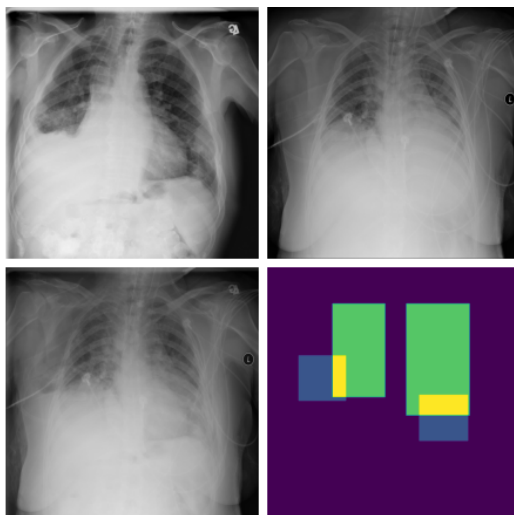


Figure 3.4: (Top) original CXRs for mixup; (bottom left) result of mixup augmentation and (bottom right) corresponding mask.

3.3 Pseudo-labelling

For pseudo-labelling, we adopted the same workflow as [Arazo *et al.* \(2020\)](#) but adapted it to segmentation. The main problem with pseudo-labelling is that it can reinforce wrong labels and go into a feedback loop. To prevent this, [Arazo *et al.* \(2020\)](#) decrease the confidence in the masks by using soft labels i.e. using the softmax predictions directly instead of thresholding them (see Fig. 3.5). Another confidence tempering technique is the mixup augmentation that they use for training. They report that the model learns better with these two additions. Therefore we have trained two models - one with mixup and one without the mixup augmentation.

Since pseudo-labelling is an architecture agnostic technique, we used the same architecture as mentioned in section 3.2. We partitioned the RSNA dataset in a different manner:

- The test set remained the same as the test set for the augmentations experiment (3000 images, 41.4% prevalence).
- We created a new validation set with 1000 images, out of which 200 had bounding boxes.
- We used the remaining 22684 images as the training set.
- Out of these, we chose 2000 images to be the labelled set, with 50% of images having bounding boxes.

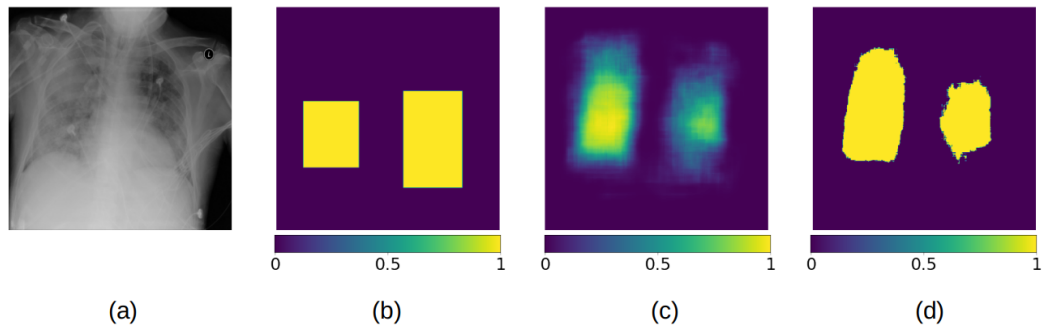


Figure 3.5: **Soft labels for pseudo-labelling:** (a) The chest X-Ray with pneumonia label. (b) The ground truth mask provided in the dataset. (c) The prediction of the model, with the scale below it. (d) The prediction with a threshold - all values > 0.3 have been set to 1, and the rest to 0. Notice that the model is more confident about lung opacity in the same region as the ground truth. Therefore if the model is trained on (c), it would learn better than if it were trained on (d), where it would consider the entire lung as positive.

The workflow for pseudo-labelling was as follows.

Pseudo-labelling algorithm

Warm up Phase

Use only the labelled images to train a supervised learning model.
Train for 15 epochs.

SSL Phase

Use the entire training set.
Load image for training.
If image belongs to the labelled part:
 Load mask from database
Else:
 Use model to predict a mask
Train model on loaded images and masks.

3.4 Implementation

We used the Keras library (version 2.4.3) to execute all the code in Python 3.7 using a Google Colab server with GPUs. This section discusses some of the constraints on training CNNs and how we overcame them.

3.4.1 Batch Generator

Each image input into the CNN has dimensions (512,512,3). If we save each of them `uint8` format, which takes stores each number in 1 byte, each image will occupy 0.75 MB space. Each mask has a dimension (512,512), and the smallest format they can be stored in is `boolean` which stores 0's and 1's, and would thus occupy 0.25 MB space. 10,000 such images and masks will thus require about 10 GB of storage and an equal amount of RAM. Thus it is crucial to have a method to feed the CNN only as many images and masks as it needs. We defined a batch generator class in python using the Keras utility `keras.utils.Sequence` and with the help of Shervine Amidi's blog¹. The box below briefly describes how it works.

All augmentations except mixup were applied using the `imgaug` library, version 0.4.0 (Jung *et al.*, 2020). For mixup, we defined a custom function which takes two images and returns the mixup. The batch generator needed to run only for half as many steps when using mixup, and we updated it to reflect the same.

3.4.2 Custom Utils Library

In order to keep the code clean, we wrote the most common functions into a separate python file and imported all the functions from it while training the models. The following is a brief description of the functions in that library:

- **Mean IoU:** Defines the mean intersection over union (section 2.2.3)
- **Extract files:** Extracts the images from a zip file. This was required as data had to be imported into Google Colab every time.

¹<https://stanford.edu/shervine/blog/keras-how-to-generate-data-on-the-fly>

Batch generator algorithm

Input: `images`: A list of image paths, `pneumonia_locations` a dictionary containing pneumonia locations, `batch_size`, `image_size`, `augmentation`: the augmentation function.

Function `get_item`: Loads the batch and sends to the model

For a subsection of `images` of size `batch_size`,
use `load_image` function for each image

Return images and masks

Function `load_image`:

Take image name and load it.

Create an empty mask with all entries 0

If image is in `pneumonia_locations`:

Update mask with entries inside bounding boxes = 1

Resize image and mask to `image_size`

Apply `augmentation` to both image and mask

Return image and mask

Function `on_epoch_end`:

When epoch ends, shuffle the images

Function `length`:

Divide the total number of images by `batch_size` to know
how many steps there will be in the epoch.

- **Batch Generator**: Generates a batch of images for model training, discussed above.
- **Name Generator**: Generates filenames to save model weights and training history in.
- **Keras Unet**: Generates the U-net architecture used in this study.

Chapter 4

Results and Discussion

4.1 The model is able to learn within thirty epochs

An important part of the machine learning is to train for enough duration such that the model neither underfits nor overfits the training data. In our experiments, we trained the models for thirty epochs and saved the model with the smallest validation loss. Figure 4.1 shows an example of the different model metrics as a function of epochs. As we can see, validation loss first reduces quite fast and then starts to slowly increase around epoch 16. Therefore we came to the conclusion that thirty epochs are adequate to train all the models. In our experiments, validation loss never decreased after the twenty-fifth epoch.

4.2 Comparison of performances after augmentation

For comparison of the models' segmentation performance, we plotted each model's mean IoU in Fig. 4.3. We also compared the different models' classification performances by using AUC, in Fig. 4.4.

It is essential to note the behaviour of the NoAug models. In both Fig. 4.3 and Fig. 4.4, we see that the performance of NoAug models is lower than baseline for 30% and 50% data. However, the NoAug models with 70% and 90% data perform as good as, or even better, than the baseline. Thus it is irrelevant to study augmentations on 70% or more fraction of the data for this study. The p-values for the one-tailed DeLong Test between AUC of NoAug and baseline are $\sim 10^{-3}$ and $\sim 10^{-4}$ respectively for the internal test

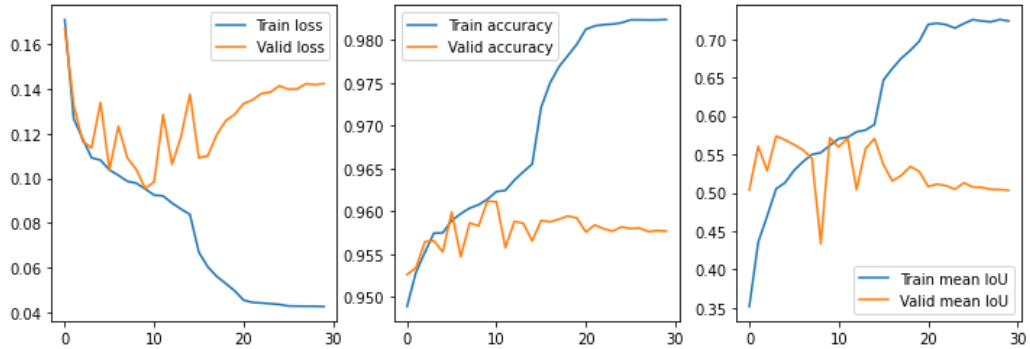


Figure 4.1: **Model performance as a function of epochs:** As we can see, the validation loss reaches its lowest point of 0.0954 at epoch 11, and then only increases. This graph is from the NoAug model with 90% data.

set.

We checked for criteria mentioned in section 1.3. To check the first criterion, we did two DeLong hypothesis tests to see if the augmentation models' performance is comparable to the baseline. The first has the null hypothesis that the AUC of augmentation models is equal to that of the baseline trained on 100% data. In this case, a p-value larger than 0.05 would mean that we fail to reject the null hypothesis, and thus we can say the two AUC are comparable. In the second test, the null hypothesis is that the AUC of the augmentation models is lesser than the baseline. Here a p-value of less than 0.05 would mean that we can reject the null hypothesis, and the AUC of augmentation models is significantly larger than that of the baseline. We find that except for gamma with 30% data and flip with 50% data, all models trained on 30% and 50% data pass either of the two tests on the internal test set. However, for the external test set, only rotate and flip with 30% data, and translate, rotate, gamma and mixup with 50% data pass the hypothesis tests (see table 4.2 for the p-values).

We checked the second criterion by performing a one-tailed DeLong Test on the AUC of the augmentation against NoAug models trained on the same data on both test sets. We find that for the external test set, all augmentations with 30% and 50% data perform significantly better than NoAug ($p < 10^{-2}$). For the internal test set, all augmentations trained on 30% data performed better than NoAug with 30% data. However, on 50% data, only the gamma and mixup performed better than NoAug.

We propose that good augmentations should satisfy both the criteria above for both the test sets. We found that rotate and flip with 30% data and gamma and mixup with 50% data accomplished this.

Test Set	Internal Test set		External Test Set	
Augmentation	30%	50%	30%	50%
Baseline on 100%	0.8569	0.8569	0.9298	0.9298
None	0.8251	0.854	0.859	0.9058
Translate	0.8615	0.8515	0.9106	0.9232
Rotate	0.8626	0.8587	0.9373	0.9251
Gamma	0.8464	0.868	0.8954	0.9428
Mixup	0.8574	0.8638	0.9042	0.9329
Flip	0.8584	0.8483	0.9249	0.9135

Table 4.1: The AUC ROC of the different augmentations calculated on 30% and 50% data for both internal and external test sets.

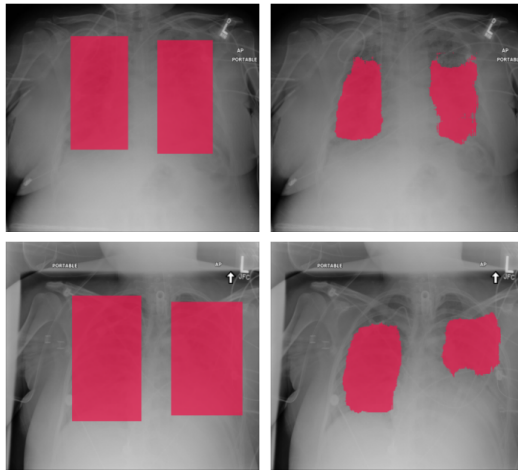


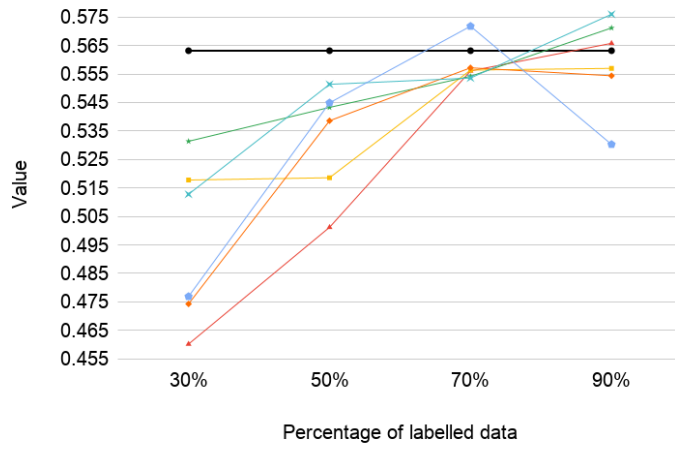
Figure 4.2: Two examples of CXR with the ground truth mask (left) and predicted mask by the baseline model (right)

On the other hand, the mean IoU plot informs us of the best data augmentations at pixel-level. As we can see in the internal test set (Fig. 4.3(a)), rotate and mixup perform pretty well at 30%, whereas mixup and flip perform better at 50%. For the external test set (Fig. 4.3(b)), we see that rotate and mixup at 30% and flip and gamma at 50% performing better than NoAug and almost as good as the baseline.

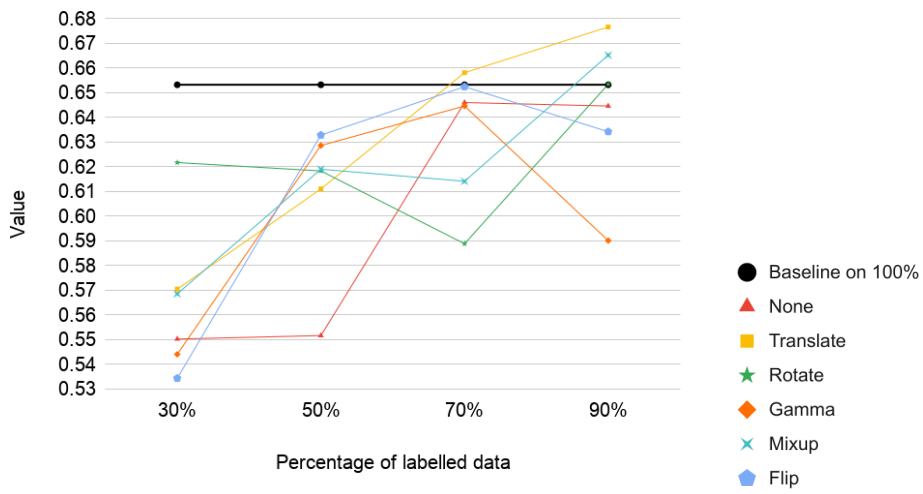
Therefore, we find that rotate and mixup are the best augmentations for semantic segmentation on our dataset. These augmentations are capable of reducing the labelled data requirements by even 70%.

Test Set	Internal Test set		External Test Set	
	30%	50%	30%	50%
Augmentation	7.66×10^{-12}	0.4826 /	2.20×10^{-16}	6.04×10^{-8}
	/ 1	0.7587	/ 1	/ 1
Translate	0.2288 /	0.176 /	2.34×10^{-5}	0.084 /
	0.1144	0.912	/ 1	0.958
Rotate	0.1576 /	0.6309 /	0.0599 /	0.2606 /
	0.0788	0.3154	0.0300	0.8697
Gamma	0.0147 /	1.86 /	1.11×10^{-13}	5.15×10^{-4}
	0.9927	9.29×10^{-4}	/ 1	/ 2.58
				$\times 10^{-4}$
Mixup	0.908 /	0.0546 /	2.70×10^{-8}	0.4118 /
	0.454	0.0273	/ 1	0.2059
Flip	0.7151 /	0.02505 /	0.2316 /	6.984×10^{-4}
	0.3575	0.9875	0.8842	/ 0.9997

Table 4.2: p-values of the DeLong Test compared to the baseline for the external test set. The first value corresponds to two-tailed DeLong test and the second corresponds to the one-tailed DeLong test. Bold indicates that the value satisfied our criteria, as explained in [Results and Discussion](#)



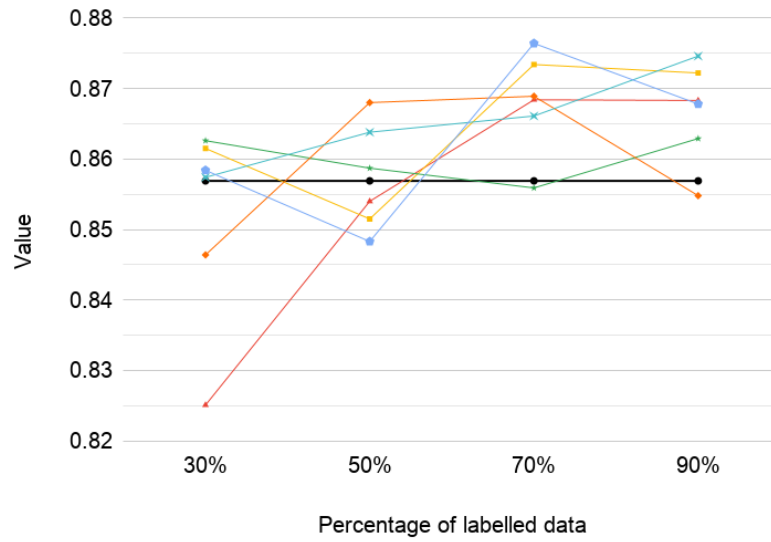
(a)



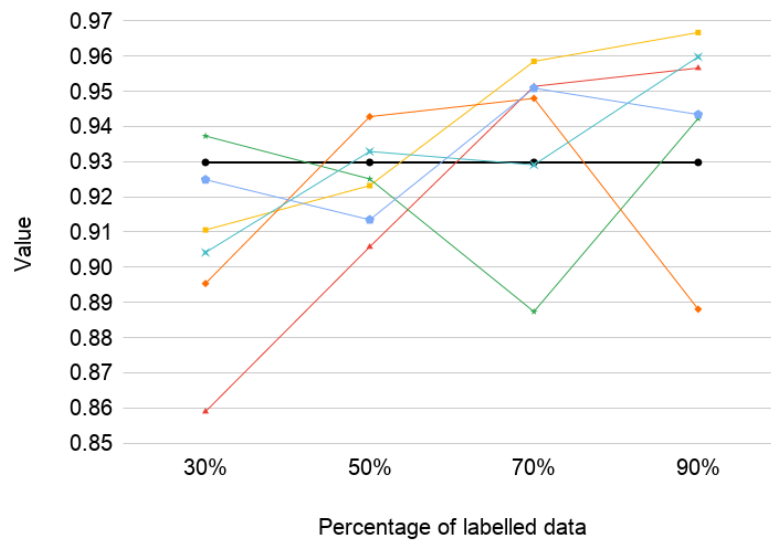
(b)

(c)

Figure 4.3: Mean IoU on the (a) internal test set and (b) external test set for the various models trained. (c) The legend



(a)



(b)

Figure 4.4: AUC ROC of the (a) internal and (b) external test set for the various models.

	Model Name			
	Plain model Warm-up Phase	Plain Model After SSL	Mixup Model Warm-up Phase	Mixup Model With SSL
Mean IoU	0.4197	0.4861	0.3793	0.5453
AUC	0.8316	0.8443	0.7570	0.7333

Table 4.3: AUC and mean IoU for semi-supervised learning approaches

4.3 Pseudo-labelling helps in segmentation but not in classification

For the pseudo-labelling approach as well, we measured the mean IoU and the AUC of the models on the internal test set, as given in table 4.3. As we can see, the mean IoU is the most for pseudo-labelling with Mixup, which is the same result that [Arazo *et al.* \(2020\)](#) report. However, the AUC for those models is not as high as the models without augmentations.

4.4 Discussion and Conclusion

Users of deep learning algorithms often overlook image augmentations as an extra step for a minor boost in performance. Our study has shown that augmentations are capable of reducing the amount of labelled training data required. To the best of our knowledge, this is the first study addressing augmentations in this manner.

There is an assumption that augmentations are most valuable when the training and test data are from the same distribution [Shorten and Khoshgof-taar \(2019\)](#). Using two test sets from vastly different populations (internal test set from the RSNA data set, sourced from the USA and the out of sample test set sourced from India) and achieving good results on both, we have shown that this assumption need not hold. While this study looked at individual augmentations, there is still scope for improving the model performance and reducing the labelled data requirement further by combining multiple augmentations.

Pseudo-labelling approaches are the starting point of semi-supervised learning, yet studies about them in the medical domain have been limited. Segmentation instead of classification adds another layer of complexity. Moreover, our segmentation models have a slightly different goal than other segmentation models - usually, all instances in the training data have a positive class, for example, in datasets such as COCO. In our dataset, the negative class is also present, and more often than not in a higher proportion

than the positive class. Thus our segmentation model is also playing the role of a classifier. Under these considerations, our work brings a new perspective to the field and opens up the avenue to further research.

References

- Arazo E, Ortego D, Albert P, O'Connor NE, McGuinness K (2020). Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning. arXiv:190802983 [cs] ArXiv: 1908.02983.
- Bustos A, Pertusa A, Salinas JM, de la Iglesia-Vayá M (2020). PadChest: A large chest x-ray image dataset with multi-label annotated reports. Medical Image Analysis 66, 101797. ArXiv: 1901.07441.
- Chollet F (2019). Keras documentation: Image segmentation with a U-Net-like architecture. Keras website. https://keras.io/examples/vision/oxford_pets_image_segmentation/. Accessed: 28-Dec-2020.
- Dumoulin V, Visin F (2016). A guide to convolution arithmetic for deep learning. arXiv:160307285 [cs, stat] ArXiv: 1603.07285 version: 1.
- Dunnmon JA, Yi D, Langlotz CP, Ré C, Rubin DL, Lungren MP (2018). Assessment of Convolutional Neural Networks for Automated Classification of Chest Radiographs. Radiology 290(2), 537–544.
- Eaton-Rosen Z, Bragman F, Ourselin S, Cardoso MJ (2018). Improving Data Augmentation for Medical Image Segmentation .
- Ginneken BV, Romeny BMTH, Viergever MA (2001). Computer-aided diagnosis in chest radiography: a survey. IEEE Transactions on Medical Imaging 20(12), 1228–1241.
- Goodfellow I, Bengio Y, Courville A (2016). Deep Learning. MIT Press. <http://www.deeplearningbook.org>.
- Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014). Generative Adversarial Networks. arXiv:14062661 [cs, stat] ArXiv: 1406.2661.
- He K, Zhang X, Ren S, Sun J (2015a). Deep Residual Learning for Image Recognition. arXiv:151203385 [cs] ArXiv: 1512.03385.

- He K, Zhang X, Ren S, Sun J (2015b). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv:1502.01852 [cs] ArXiv: 1502.01852.
- Hurt B, Yen A, Kligerman S, Hsiao A (2020). Augmenting Interpretation of Chest Radiographs With Deep Learning Probability Maps. *Journal of Thoracic Imaging* 35(5), 285–293.
- Irvin J, Rajpurkar P, Ko M, Yu Y, Ciurea-Ilcus S, Chute C, Marklund H, Haghighi B, Ball R, Shpanskaya K, Seekins J, Mong DA, Halabi SS, Sandberg JK, Jones R, Larson DB, Langlotz CP, Patel BN, Lungren MP, Ng AY (2019). CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. arXiv:1901.07031 [cs, eess] ArXiv: 1901.07031.
- Jaeger S, Candemir S, Antani S, Wang YXJ, Lu PX, Thoma G (2014). Two public chest X-ray datasets for computer-aided screening of pulmonary diseases. *Quantitative Imaging in Medicine and Surgery* 4(6), 475–477.
- Jung AB, Wada K, Crall J, Tanaka S, Graving J, Reinders C, Yadav S, Banerjee J, Vecsei G, Kraft A, Rui Z, Borovec J, Vallentin C, Zhydenko S, Pfeiffer K, Cook B, Fernández I, De Rainville FM, Weng CH, Ayala-Acevedo A, Meudec R, Laporte M, *et al.* (2020). *imgaug*. <https://github.com/aleju/imgaug>. Online; accessed 05-Jan-2021.
- Kermany DS, Goldbaum M, Cai W, Valentim CCS, Liang H, Baxter SL, McKeown A, Yang G, Wu X, Yan F, Dong J, Prasadha MK, Pei J, Ting MYL, Zhu J, Li C, Hewett S, Dong J, Ziyar I, Shi A, Zhang R, Zheng L, Hou R, Shi W, Fu X, Duan Y, Huu VAN, Wen C, Zhang ED, Zhang CL, Li O, Wang X, Singer MA, Sun X, Xu J, Tafreshi A, Lewis MA, Xia H, Zhang K (2018). Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell* 172(5), 1122–1131.e9.
- Kingma DP, Ba J (2017). Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs] ArXiv: 1412.6980.
- Lakhani P, Sundaram B (2017). Deep Learning at Chest Radiography: Automated Classification of Pulmonary Tuberculosis by Using Convolutional Neural Networks. *Radiology* 284(2), 574–582.
- Lecun Y, Bottou L, Bengio Y, Haffner P (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11), 2278–2324.

- Lee DH (2013). Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. ICML 2013 Workshop : Challenges in Representation Learning (WREPL) .
- Long J, Shelhamer E, Darrell T (2015). Fully Convolutional Networks for Semantic Segmentation. arXiv:14114038 [cs] ArXiv: 1411.4038.
- Lundervold AS, Lundervold A (2019). An overview of deep learning in medical imaging focusing on MRI. *Zeitschrift für Medizinische Physik* 29(2), 102–127.
- McBee MP, Awan OA, Colucci AT, Ghobadi CW, Kadom N, Kansagra AP, Tridandapani S, Auffermann WF (2018). Deep Learning in Radiology. *Academic Radiology* 25(11), 1472–1480.
- Mitchell TM (1997). *Machine Learning*. McGraw-Hill, New York.
- Pan I, Cadrin-Chênevert A, Cheng PM (2019). Tackling the Radiological Society of North America Pneumonia Detection Challenge. *American Journal of Roentgenology* 213(3), 568–574.
- Pasa F, Golkov V, Pfeiffer F, Cremers D, Pfeiffer D (2019). Efficient deep network architectures for fast chest x-ray tuberculosis screening and visualization. *Scientific Reports* 9(1), 6268.
- Peikari M, Salama S, Nofech-Mozes S, Martel AL (2018). A Cluster-then-label Semi-supervised Learning Approach for Pathology Image Classification. *Scientific Reports* 8(1), 7193.
- Prevedello LM, Halabi SS, Shih G, Wu CC, Kohli MD, Chokshi FH, Erickson BJ, Kalpathy-Cramer J, Andriole KP, Flanders AE (2019). Challenges Related to Artificial Intelligence Research in Medical Imaging and the Importance of Image Analysis Competitions. *Radiology: Artificial Intelligence* 1(1), e180031.
- Redmon J, Divvala S, Girshick R, Farhadi A (2016). You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. ISSN: 1063-6919.
- Ronneberger O, Fischer P, Brox T (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:150504597 [cs] ArXiv: 1505.04597.
- Selvan R, Dam EB, Detlefsen NS, Rischel S, Sheng K, Nielsen M, Pai A (2020). Lung Segmentation from Chest X-rays using Variational Data Imputation. arXiv:200510052 [cs, eess, stat] ArXiv: 2005.10052.

- Shih G, Wu CC, Halabi SS, Kohli MD, Prevedello LM, Cook TS, Sharma A, Amorosa JK, Arteaga V, Galperin-Aizenberg M, Gill RR, Godoy MC, Hobbs S, Jeudy J, Laroia A, Shah PN, Vummidi D, Yaddanapudi K, Stein A (2019). Augmenting the National Institutes of Health Chest Radiograph Dataset with Expert Annotations of Possible Pneumonia. *Radiology: Artificial Intelligence* 1(1), e180041.
- Shorten C, Khoshgoftaar TM (2019). A survey on image data augmentation for deep learning. *Journal of Big Data* 6(60).
- Sirazitdinov I, Kholiavchenko M, Kuleev R, Ibragimov B (2019). Data Augmentation for Chest Pathologies Classification. In 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), 1216–1219. ISSN: 1945-8452.
- Souza JC, Diniz JOB, Ferreira JL, da Silva GLF, Silva AC, de Paiva AC (2019). An automatic method for lung segmentation and reconstruction in chest X-ray using deep neural networks. *Computer Methods and Programs in Biomedicine* 177, 285–296.
- van Ginneken B (2017). Fifty years of computer analysis in chest imaging: rule-based, machine learning, deep learning. *Radiological Physics and Technology* 10(1), 23–32.
- Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM (2017). ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3462–3471. ISSN: 1063-6919.
- Wu J, Gur Y, Karargyris A, Syed AB, Boyko O, Moradi M, Syeda-Mahmood T (2020). Automatic Bounding Box Annotation of Chest X-Ray Data for Localization of Abnormalities. In 2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI), 799–803. ISSN: 1945-8452.
- Zhang H, Cisse M, Dauphin YN, Lopez-Paz D (2018). mixup: Beyond Empirical Risk Minimization. *arXiv:171009412 [cs, stat]* ArXiv: 1710.09412.
- Zhu X, Goldberg AB (2009). Introduction to Semi-Supervised Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3(1), 1–130.
- Zou Y, Zhang Z, Zhang H, Li CL, Bian X, Huang JB, Pfister T (2020). PseudoSeg: Designing pseudo labels for semantic segmentation. *arXiv:201009713v2 [cs]* .