Pneumothorax Segmentation

Submitted May 2020, in partial fulfillment of
the conditions for the award of the degree **BSc Computer Science with Artificial Intelligence.**

18025713

Supervised by Dr Mercedes Torres

School of Computer Science

University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text

Signature ___________________

Date ______ / ______ / ______

Acknowledgment

First and foremost, I would like to thank God Almighty for giving me the strength, knowledge, ability and opportunity to undertake this project and to persevere and complete it satisfactorily. Without his blessings, this achievement would not have been possible.

I would like to express my gratitude to my supervisor Dr. Mercedes Torres for her quick feedback and constructive comments. I’m extremely grateful to Dr. Mohammed Abdelsamea for his invaluable guidance, unwavering support, and insightful suggestions. I would like to thank my colleagues and friends for their discussions, suggestions and criticism.

This acknowledgment wouldn’t be complete without thanking my family. My parents whom I owe everything to for encouraging me and helping me at every stage of my personal and academic life. My siblings whose utter innocence and joyfulness has kept motivated and encouraged along the years and longed to see this achievement come true.

—Alhamdulillah

Usama Zidan
May, 2020

Abstract

Neural Networks have revolutionised the healthcare industry by introducing Deep Learning models that reach human-level accuracy. In this report, a pipeline implementation is explored that tries to enhance the accuracy of Deep Learning models in the detection and segmentation of pneumothorax from a set of chest radiographs. This is done in the hopes of facilitating future applications in the medical field.

By cascading a CNN and a U-Net, we ensure that the CNN will filter out all the cases with no pathology, i.e.'Normal' cases, leaving the U-Net (trained on positive cases only) to focus on arranging a set of filters for the separation of pathology from the infected lungs. The models are cross validated on the same data to ensure consistency. The proposed method demonstrates higher accuracy in both segmenting and detecting the pathology. By implementing it with more complex architectures and integrating in the domain knowledge of radiologists, this methods can be applied in conjunction with other applications to rapidly triage and prioritise cases for the presence of anomalies.

Keywords: ***Fully Convolutional Networks***$_1$; ***pneumothorax***$_2$; ***medical image analysis***$_3$; ***deep learning***$_4$

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pneumothorax is a life-threatening disorder caused by the abnormal accumulation of free air in the pleural space. It is a critical condition that often requires urgent coordination of imaging results and immediate intervention. [3–6]. As a pathology, it is usually diagnosed by a radiologist based on a Chest X-ray which is used as a general method for Pneumothorax screening and detection. However, a diagnosis from an X-ray image can be difficult to confirm due to the complex appearance of some pathologies. As it is a life-threatening event, timely interpretation of radiographs can be challenging, especially when not reviewed immediately due to queues of cases waiting to be examined by radiologists.

A rapid Computer Aided Diagnosis (CAD) system for detecting Pneumothorax can be used to interpret radiographs as soon as they are obtained to send alerts to radiologists, hence improving the efficiency of the diagnosis process. Such a system could then be adapted to triage chest radiographs for priority interpretation or to provide non-radiologists with a confident diagnosis.

Recent Deep Learning applications have gained attention for using sophisticated algorithms to achieve high performance in computer vision tasks for natural image processing [7]. Several methods have also been developed in medical image analysis for classifying thoracic diseases in chest radiographs [8]. With the availability of large datasets and advanced computing resource, more complex architectures have been developed promising improved representation and generalizability of data than other methods, maximizing performance robustness in real-world applications. These breakthroughs paved the way for opportunities in many aspects of radiology, such as the detection of pulmonary tuberculosis [9], segmentation and diagnosis of cardiovascular imaging [10], diabetic retinopathy detection [11], and detection of mass effects, hydrocephalus and haemorrhage throughout the brain [12]

With the increasing demand on hospitals, they need to be efficient in diagnosis and treatment of incoming patients. Typically, in any given hospital workflow, diagnosis begins by some form of medical imaging, with workload of clinical radiology increasing annually. In turn, this exerts increasing pressure on radiology services to increase their efficiency while maintaining quality. In such institutions, the ability to prioritise some imaging exams is possible by the use of emergent labeling [13]. However, because of their overuse and misuse, radiologists frequently find it difficult to prioritise those exams with more significant results. This increasing pressure has warranted numerous recent scientific breakthroughs in the field. Some applications have gained attention for using sophisticated algorithms to 'learn' features from large volumes of data, using the gained insight to assist clinical practices.

Given such advantages, these systems can help reduce diagnostic and therapeutic errors that are inevitable in human clinical practices. One of the many ways AI-based technologies can be implemented in hospitals is as s triage or screening tool. It can analyze medical images and use the probability of disease as a basis to decide which images should be interpreted first by a human radiologist, thus prioritizing life threatening cases for eminent treatment. Rajpurkar et al. [14] trained a network on the larges public available chest X-ray dataset to detect 14 diseases and achieved state-of-the-art results on all 14 pathologies. In order to bridge the gap between Deep learning researchers and medical professionals Cohen et al. [15] developed a very accessible free prototype system that can be used by medical professionals to grasp the nature of Deep Learning techniques for X-ray diagnosis. The system is intended to act as a second opinion in which a person may view an image to validate or assist with their diagnosis. Code and network weights are transmitted to a web server (including mobile phones) through a URL, but the patient data stays on the user computer, and all processing takes place locally.

The project aims to determine the effects or bias of negative cases on the effectiveness of FCNs, specifically U-Nets, in the segmentation of pneumothorax from a set of radiographs. The results will help in future implementations of CAD systems in hospitals.

The goals of this project are as follows:

- Investigate an approach that might prove helpful in future implementations of real-life applications of diagnosis systems
- Develop a pipeline that can be used interchangeably with other more complex architectures in the future
- Provide an insight into the inner-workings of Neural networks by investigating the bias that they might suffer from given large amounts of negative samples in datasets

In this dissertation, the following chapters will explore relevant approaches in the literature. Followed by a detailed explanation of the methodologies used for the Pneumothorax segmentation. Implementation and model configuration are in chapter 4, Finally chapters 6 and 7 conclude the paper outcomes with future work.

Chapter 2

Related Work

2.1 Medical Image analysis

Over the past several decades, medical imaging methods such as magnetic resonance imaging (MRI), computed tomography (CT), ultrasound, and X-ray have been utilised for the early detection and diagnosis of diseases. In a clinic, the processing of medical images was done mainly by human experts such as radiologists and physicians. Nonetheless, due to the wide variation of some pathologies and the potential human fatigue, specialists have shifted from systems that are completely designed by humans to ones that are trained by computers. These computer-assisted interventions have seen huge advancements, with the advent of Machine Learning solutions.

Machine Learning plays a pivotal role in discovering or learning features that are highly informative about the patterns inherent in data. Conventionally, practical or domain-specific features were developed mainly by human experts based on their experience of the subject areas, making it difficult for non-experts to use machine learning methods on their own research. Meanwhile, attempts have been made to learn some sparse representations from training samples, based on predefined dictionaries. While these dictionary learning methods could still find informative patterns inherent in the data, their representational power is limited with shallow architectures. To overcome this barrier, Deep Learning requires that the feature engineering phase be turned into a learning phase. Thus, instead of manually extracting the features, a process involving a set of data with some preprocessing and a representation is, iteratively, discovered in a self-taught manner. During this learning process, Deep Learning Networks can enable Higher level features to be extracted from low-level features, through an exploration of the hierarchical representations of features [16].

Due to these advancements, Machine learning has seen a dramatic amount of attention in recent years. Deep Learning Networks have started to outperform various other models on a number of benchmarks, becoming the state-of-the-art model across a variety of fields. These developments have encouraged the use of Deep Learning in the field of medical imaging and diagnostics, such as image segmentation [17, 18], image fusion [19], image annotation [20], computer-aided diagnosis (CADx) and prognosis [21], lesion/landmark detection [22], and microscopic image recognition [23].

2.2 Feature Representation

Many current methods of processing any medical image rely on features attained using morphological operations to identify local anatomical features. Yet, such representations of the data tend to specifically designed for a certain problem and aren't guaranteed to be applicable to other types of images.

While some supervised learning approaches can identify important and necessary features for the target task, they do require huge amounts of manually labeled data. Furthermore, the features extracted can be too simplistic and underestimate the anatomical structures of the pathology. Most importantly, the learning process is often limited to a single template context, with a number of features pre-designed. Therefore, if the template or image features shift, the whole training cycle must restart. To counteract these issues, Wu et al. [24] developed a sparse auto-encoder (SAE) that hierarchically learned feature representations in a layer-by-layer manner. Figure 2.1 demonstrates how Deep Learning methods offer a more accurate and informative feature representation while allowing for more flexibility of application to different types of medical images. These applications demonstrate how Deep Learning systems can be utilised by researchers to rapidly improve imaging methods by finding inherent characteristics that can be used to extend many other medical imaging applications. They also offer an understanding of how latent character representation inferred by Deep Learning can effectively explain local image features.

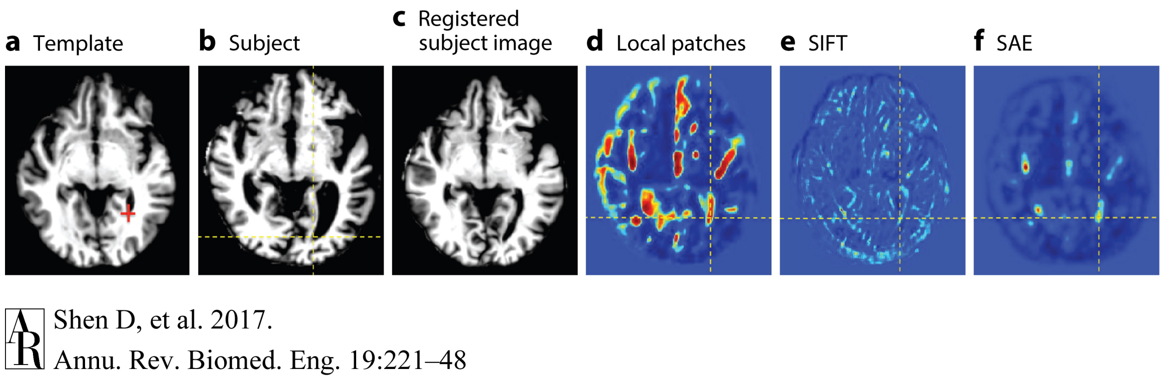


Figure 2.1: Similarity maps for identifying the correspondence from different approaches [1]

2.3 Deep Learning for Classification

Medical image classification is a sub-domain of image classification. Many of the techniques used for image classification may also be applied to medical images. Convolutional neural networks (CNNs) have been becoming the state of the art in many classification and image recognition competitions. They consist of a series of layers of neurons each undergoing a particular operation such as pooling, loss calculation, etc. These layers receive the output of the previous layer as their input. The starting layer contains an equal number of neurons to the number of pixels in the input image. The next set of layers operate on tuning their filters, commonly referred to as Kernels, to extract as many features from the input data by applying various transformations on the input. This optimisation process of the kernel parameters is possible through strategies such as backpropagation that constantly optimises the weights of the network for better accurate results. [25].

To demonstrate how systems utilising CNNs can be generalised to many Medical Imaging datasets, Li et al. [26] demonstrated a modified CNN that can classify image patches of lung disease. Their work showed that these systems can be applied to large Chest-Xray datasets with over 100 thousand images, like (ChestX-ray8), to achieve high accuracy. Yet, due to the limited data availability, training models effectively from scratch is unattainable.

An approach that proved quite effective in training deep models is Transfer Learning, in which models trained on other tasks, such as ImageNet, are *ported* over and trained on another dataset that is related to the task at hand. The intuition here is that the knowledge gained from one problem can be applied to another, thus improving training efficiency as the model doesn't need to re-learn the same basic weights again. Vianna [27] demonstrated the effectiveness of transfer learning by using a pretrained InceptionV3 model trained on a dataset consisting of 108,312 OCT images. They achieved an average 96% accuracy, with 97.8% sensitivity, and a specificity of 97.4%. To show how these numbers translate in comparison to real human experts, they tested the system against 6 experts. They report that most of the experts got high sensitivity but low specificity, while the CNN achieved

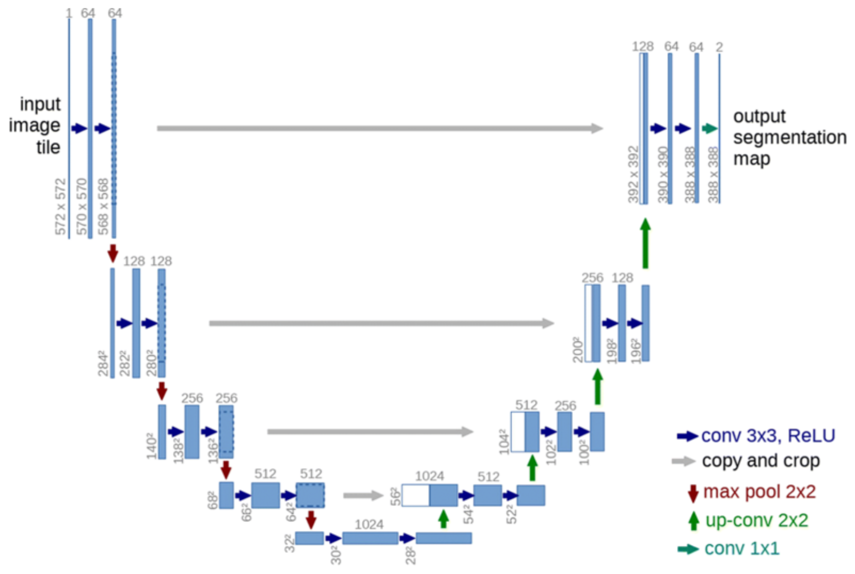


Figure 2.2: The structure of a U-Net

high values in both [28].

In Vianna [27] work, a study was carried out to assess the utility of transfer learning in building X-ray image classification systems that are a vital component of (CAD) systems. They found that a fine-tuned pretrained model trained with data augmentation can essentially alleviate the over-fitting problem and deliver a better result.

2.4 Deep Learning for Segmentation

Long et al. [29] improved on CNNs by introducing Fully Convolutional Network (FCN) where the last fully connected layer is replaced with a fully convolutional layer, enabling the FCN to have pixel-level predictions. This allows the FCN to get predictions for the full-sized image instead of patch-wise predictions. The achievement here is that FCNs are capable of carrying out predictions of the entire image in just one forward pass.

Another well know structure based on the elegant design of FCNs, that this project is also utilities, is a U-Net proposed by Ronneberger et al. [30]. U-Net benefits from the superior design of skip connections between different network stages. It uses some adjustments to overcome the trade-off between localization and context utilisation. This trade-off increases as the large patches require more pooling layers, thus reducing the accuracy of the localization. While small patches, on the other hand, can only observe a small input context. The structure suggested follows CNN's structure (Figure 2.2)

In medical image segmentation, U-Nets have attracted a lot of attention based on which many variants have been developed [31–33]. For example, with a U-Net structure-based network, Gordienko et al. [31] explored lung segmentation in X-ray scans. The results showed that U-Nets are capable of rapid and precise segmentation of the medical images.

Cascading fully connected networks were used in Christ et al. [34] for segmenting the liver and its lesions. A U-Net is trained to segment the liver as ROI for the next FCN that focuses on finding the lesions inside that segmented liver. The novel ensemble achieved a Dice scores over 94% for the liver, with computation times below 100s per volume. Their experiments show the potential that U-Nets possess for ensemble techniques.

Gooßen et al. [13] carried out performance measurements to evaluate the different approaches there are to tackle the problem of image classification and segmentation of medical images. The techniques used were: Convolutional Neural Networks, fully connected convolutional networks, and Multiple Instance learning. Their analysis of the results detailed an average score of AUC 0.94 for the three methods. Although the three techniques presented provided promising options for the detection and localization of Pneumothorax in chest X-ray images, U-Nets were shown to possess better pixel-wise predictions, making them a more precise option for segmentation tasks.

Çiçek et al. [32] created a 3D U-Net model in an effort to empower the U-Net architecture with more spatial details. The model used the input data, 2D annotated slices, to generate volumetric segmentation, demonstrating how the architecture can find whole 3D volume data from a few annotated slices successfully. The authors utilised a softmax loss function that enabled the densification of sparse annotated samples and achieved an average IoU of 0.863.

2.5 Computer-Aided Detection

A Computer-Aided Detection system's goal is to find or localize a potential abnormal region in a medical image, and alert specialists of the finding. These tools aim to have a high detection rate of the pathological regions while maintaining a low false negative rate that might occur due to an error or fatigue on the observers' part. While CADe systems are well known in the field of medical imaging, Deep Learning approaches have enhanced their performance in various clinical applications.

Kooi et al. [35] show how a CNN trained on a large data set of mammographic lesions can outperform a traditional CAD system at low sensitivity, performing comparably at high sensitivity. The authors subsequently analyse to what degree do features such as patient data, location, and manually extracted features can add to the network's generalization performance. They also analyse the performance of the CNN against human performance on a patch level. Their results show that human readers and CNNs have similar performances.

2.6 Computer-Aided Diagnosis

CADx systems can offer a second unbiased opinion on the evaluation of the disease from image-based knowledge. The uses of CADx require the detection, classification, and segmentation in one framework to assist radiologists efficiently in an accurate diagnosis and recognition of such diseases from one or more images. Conventionally, most CADx applications have been built to use human-designed features produced by domain experts. Luckily, Deep learning approaches have, recently, been widely extended to CADx allowing for more flexibility in applications.

In a study by Al-antari et al. [36], the authors utilised a new deep network model that can be is used to detect and classify the masses as either benign or malignant. They demonstrated that due to the segmentation capability of the proposed deep model using a FrCN, the CAD system can achieve much better classification results. The pixel-to-pixel mass segmentation may then be a key to reducing false positive/negative pixel rates, thereby increasing the overall efficiency of the proposed CAD system. Furthermore, they elaborated that a CNN's classification results indicate the reliability and effectiveness of the new CAD system relative to other approaches in the field.

Deep Learning for medical image processing has had a major impact on both clinical and research applications. Recent developments in Deep Learning have shed a new light on medical image processing by allowing the detection of morphological and/or textural variations in images solely from image observations. Deep learning approaches have attained state-of-the-art efficiency through numerous medical applications; however, there is still scope for development. Deep learning systems have recently proved capable by achieving impressive results in a variety of Image processing and computer vision problems [13, 31, 37–41], initiating interest on its applications on domains like medical image analysis. The feasibility of accessing the data and development tools allowed the development of more effective algorithms. That being said, it remains to be the problem of which approach is suitable for a real-life implementation in hospitals, taking into account the hardware requirements, training time, and efficiency/effectiveness of results.

Chapter 3

Methodology

This section is going to address chosen methodologies, techniques, and tools used in the project, including theories and algorithms that are relevant to the project scope and have been researched and experimented with to determine the best approach that meets the objectives specified.

3.1 Convolutional Neural Networks

A CNN is a type of Neural Network that is mostly used for classifying images. They consist of multiple layers, each with a specific operation, such as convolution, pooling, estimation of loss, etc [40]. In a typical CNN architecture, the first layer contains the same amount of neurons as the number of pixels in the input image. The next set of layers are convolutional layers that act as a feature extractor by convolving the input data with a number of filters. Filters, generally referred to as kernels, are of arbitrary size, determined by developers. The kernel size also affects how receptive the layer is to the input features as each neuron only responds to a specific area of the previous layer called receptive field. The output of these layers acts as an activation map to highlight the effect of applying a specific filter on the input. The following layer is usually an activation layer that applies a non-linearity to the activation maps. To reduce the dimensionality of the convolution's output, sometimes a pooling layer is used to downsample the feature maps. Finally, fully connected layers are used to extract high-level features. Figure 3.1 gives a general illustration of the inner workings of a CNN layer.

While Neural Networks architectures are becoming more used, designing an effective architecture is becoming increasingly difficult with the growing number of hyper-parameters, especially when there are many layers. The VGG-nets[42] illustrate a basic but efficient method of constructing very deep networks: stacking the same type of building blocks. ResNets[43] inherits this approach and by stacking modules of the same topology. These two architectures have proven their effectiveness in a variety of classification tasks [29, 44]. Xie et al. [45] expands on this method by adopting the same strategy that VGG/ResNets use of repeating blocks of layers while exploiting the

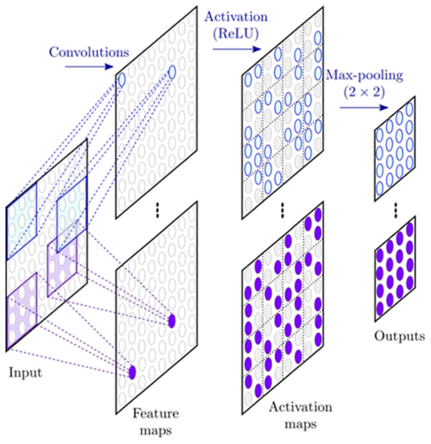


Figure 3.1: Structure of a CNN layer [2]

split-transform-merge strategy in an easy, extensible way. Their ResNeXt architecture performs a set of transformations, each on a low-dimensional embedding, whose outputs are aggregated by summation. This is possible through cardinality defined as the number of independent paths that can be adjusted through the models' Capacity hyperparameter. These independent paths are achieved by dividing the input into groups of feature maps and performing convolutions respectively. The outputs are then depth-concatenated and fed to a 1x1 convolutional layer (Figure 3.2). Experiments show that by increasing the cardinality, accuracy can be achieved more effectively than by going deeper or wider. The authors conclude that this new architecture is easier to adapt to different datasets/tasks relative to similar Inception architectures, since it has a basic framework and only one hyperparameter to modify. During their testing, they showed a significant performance improvement over Resnets as shown in Figure 3.3

3.2 Segmentation architectures

Segmentation as a task posses a challenge in both natural and medical image processing. Earlier approaches involved using a CNN to classify each pixel in an image separately by presenting it to the network along with patches extracted from around that particular pixel. The issue with this approach is that these input patches have a huge overlap causing convolutions to be computed many times. To overcome this, fully connected layers are rewritten as convolutions. This resulted in a model that can infer on image sizes that it wasn't trained on and produce a likelihood map, rather than an output for a single pixel.

One of the most well-known structures for image segmentation is a U-Net. The model is based on FCN employing some modifications to overcome the trade-off between localization and the context of use. This architecture has attracted a lot of attention demonstrating its capabilities of fast and precise image segmentation as seen in [13, 34, 40].

The proposed architecture in this project was inspired by the strategy of one class

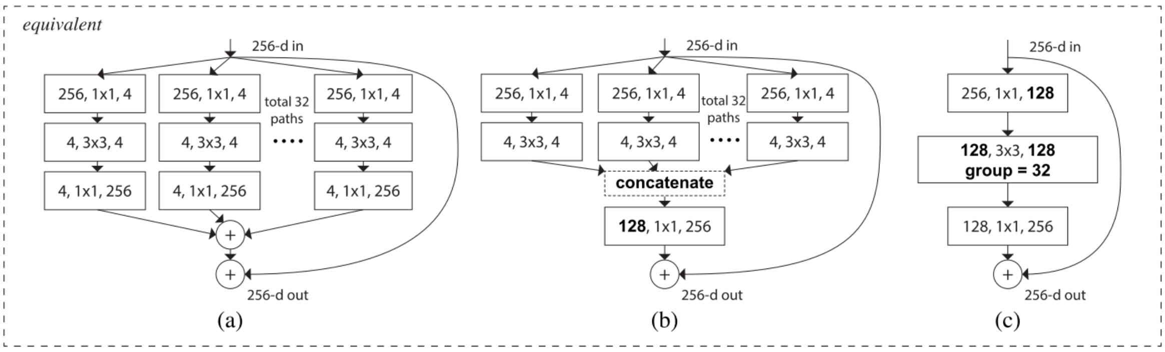


Figure 3.2: Building block of ResNeXt

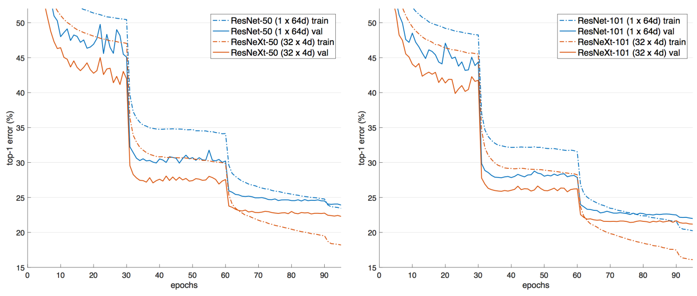


Figure 3.3: Performance of Resnet VS ResNeXt

training. The intuition behind one-class segmentation is to recognize instances of a class by only using examples of the same class in training. The idea is that, normally, the trained U-Net may predict masks even when there is no pneumothorax. Therefore, a CNN will be used to zero out wrongly predicted masks to help get better performance. Perera and Patel [46] demonstrated how one-class classification, where only training samples of a single class are available during training facilitated the learning process. They trained a CNN using two novel loss functions, descriptiveness loss and compactness loss, on a publicly available dataset for abnormal image detection, achieving state-of-the-art results in each test case.

To apply this in segmentation tasks some adjustments are needed whereby instead of focusing on *one class*, we target *one case*. Furthermore, by cascading a CNN and a U-Net, we ensure that the CNN will filter out all the cases with no pathology, i.e.'Normal' cases, leaving the U-Net to focus on arranging a set of filters for the separation of pathology from the infected lungs. Thereby, achieving *One Case* training. The filtering out as step 1 helps reduce false positives for the pathology detection in the lungs. We train a network on chest radiographs to classify them as 'Normal' and 'Abnormal' (step 1). This network solely concentrates on learning discriminate features for the health vs unhealthy lungs. The U-Net can then segment the area where the pressure on lungs exists given an image of affected lungs (step 2). Therefore, the U-Net concentrates on learning discriminate features for pneumothorax area vs background segmentation.

3.3 Tackling Data Imbalance

Most of the recent research around Medical Images usually has to tackle one prominent problem in Medical Image datasets. The ratio of ”normal” samples to ”abnormal” ones is usually biased towards the former, with the latter making a very small percentage of the set. This Imbalance problem has a significant impact on the training process. In such cases, a classification model will focus on the more dominant classes, leading to a poor classification performance for the minority classes. Furthermore, in medical diagnosis, misclassifications can be costly as the minority classes are often the ones containing a disease and thus a false negative can imply serious consequences. Coupled with the fact that the data distribution of the test set can differ from the training set, it makes the true misclassification rate unknown at the time of training. Sampling techniques have been used to overcome the imbalance problem by either eliminating some data points from the majority class (under-sampling) or artificially generating more data points from the minority class (over-sampling) [47].

3.4 Pre-processing techniques: Data Augmentation

One important step that has been shown to be effective in training deep learning models is ’Data augmentation’ is used. In medical imaging, this is normally achieved through transformations that are applied equally to both images and labels, producing varied representations of the training data. Augmentation techniques typically use transformations such as rotations, reflections, and elastic deformations, which generates training images that closely resemble one specific training example. ‘Data augmentation’ helps avoid the memorization of training data and enhances the efficiency of the network on data from outside the training set. Thus it is essential in building robust deep learning pipelines.

Although different strategies and their combinations have been researched heavily for natural images, little has been done for optimal augmentation techniques for medical tasks. The scarcity of training data in medical imaging datasets makes it critical to explore methods that serve as a regularizer and prevent overfitting the data. Hussain et al. [48] demonstrated different techniques for classifying mass and non-mass mammographic images by binary features. They showed that some augmentation techniques capture medical image statistics more efficiently than others, contributing to higher accuracy in training and validation. They also suggested that using a combination of augmentations might have higher mutual information, leading to a holistic capture of image statistics.

[28] experimented with different augmentation strategies to analyse the effects of augmentation on a VGG16 model. Their results show that Data Augmentation does improve performance regardless of the model, which is attributed to the fact that these augmentations facilitate the learning process by allowing the model to learning underground features without the effects of rotation and scale. However, the authors argued that some complicated

transformations can introduce some noise to the training set, disturbing the learning process.

3.5 Platforms for Deep Learning

Since the huge interest in deep learning, a lot of platforms emerged to help facilitate the development of AI applications. The chosen development platform for this project is PyTorch. The following section discusses its advantages.

PyTorch

Pytorch is an open-source machine learning library based on Torch which is a scientific computing framework that focuses primarily on GPU accelerated computation [49]. One of PyTorch's key advantages is its support for the creation of dynamic computation graphs.

The main utility of a dynamic computation graph is that it allows you to process complex inputs and outputs, without worrying to convert every batch of input into a tensor. Its also worth noting that PyTorch is one of the easier platforms to learn as its front-end is in python and given the time to implement this project, makes it a more compelling choice.

Fastai

Fastai is a library built on top of PyTorch with the goal of making deep learning more accessible. The library breaks down a lot of barriers to getting started with complex deep learning as it is open-source, easy to customise and replaces elements of your architecture to suit your prediction tasks, making it even easier to use PyTorch by providing functions for data pre-processing and transformation as well as Data Loaders and Pre-trained Models. Furthermore, There is a lot of starter code for loading the datasets with example models, allowing more focus to go into experimentation and fine-tuning approaches. The various 'ready-made' functions and Pre-trained Models provided for experimentation, makes using this library an effective choice in the this projects' scope.

3.6 Training Cycle

3.6.1 Loss Functions

Loss functions define how Neural network models calculate the overall error from their residuals for each epoch. This, in turn, affects how they adjust their coefficients when performing optimisation. The choice of loss function has a direct influence on model performance. The following are some loss functions that have been encountered during the research process.

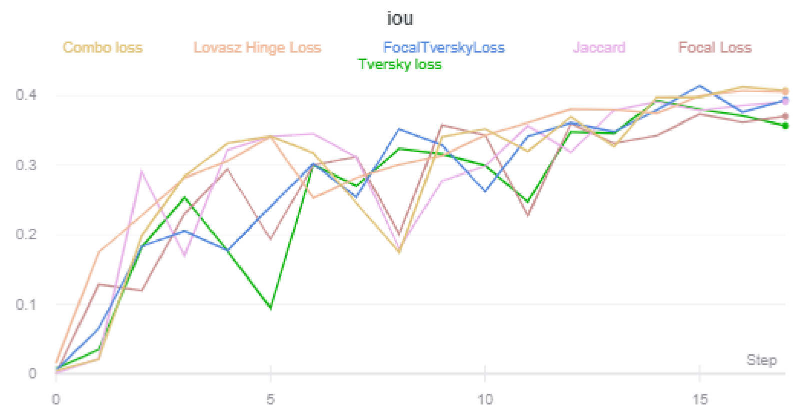


Figure 3.4: Loss functions experimented with

Binary Cross Entropy

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p(y_i)\right) + (1 - y_i) \cdot \log\left(1 - p(y_i)\right)$$

Cross-entropy loss measures the performance of a classification model whose output is a probability between 0 and 1. As the predicted probability (*p(y)*) diverges from the True label (*y*), the CE loss increases exponentially. Therefore, predicting a really low probability for a data point that has a True label of 1, would result in a high loss value. This loss function is the default for many classification tasks as it is equivalent to fitting a model using maximum likelihood estimation which is known to be statistically efficient. This loss function works quite well with gradient descent algorithm because it can undo any exponential behaviour that can result from some activation functions.

Focal Loss

$$FL(p_t) = -\alpha_t(1 - p_t)^{\tau} log(p_t)$$

To address the class imbalance problem Lin et al. [50] developed a loss function to address scenarios where extreme imbalance occurs between foreground and background classes during training. The function is designed to dynamically scale cross entropy loss, where the scaling factor decays to zeros as confidence increases in the correct class label. This application of a scale factor can automatically down-weight the contribution of easy classes during training and rapidly focus the model on hard examples. Since easy negatives usually overwhelm the model during the training cycle, the function proves to be ultimately useful in tackling class imbalance.

Tversky Loss

Another loss function developed for FCNs to try and tackle class imbalance problems is Tversky loss function [51]. It's based on the Tversky index to achieve a better trade-off between precision and recall in training 3D fully convolutional deep Neural networks. The author designed it to weigh more on False Negatives than on False Positives as it is crucial in detecting small lesions in an image. To realise this, they used the Tversky index that contains Alpha and Beta parameters which can be used to penalise for FPs and FNs, respectively, as seen below. They formulated that this removes the need to balance the weights during training and allows for more control over the trade-off between FN and FP. They also hypothesised that using higher Betas in their function will lead to higher generalisation and improved performance for imbalanced data. This function, alongside a variant that uses the gamma modifier from Focal Loss, was experimented with during the experimentation phase.

$$T(X, Y, \alpha, \beta) = \frac{|X \cap Y|}{|X \cap Y| + \alpha|X - Y| + \beta|Y - X|}$$

Lovasz Hinge Loss

Berman et al. [52] developed a function to directly optimise for the mean of IoU loss in Neural networks. It is designed to optimise the Intersection over Union score for semantic segmentation, particularly for multi-class instances. Specifically, it sorts predictions by their error before calculating cumulatively how each error affects the IoU score. This gradient vector is then multiplied with the initial error vector to penalise most strongly the predictions that decreased the IoU score the most. Their results showed better performance with regards to traditional methods like Cross Entropy loss.

Combo Loss

This loss was introduced by] as a combination of Dice Loss and a modified Cross-Entropy function that, like Tversky loss, has additional constants which penalise either false positives or false negatives more respectively. Their function tries to leverage the *Dice coefficient* to escape local minima and at the same time learn better weights by penalising for false positives/negatives using a cross entropy term.

$$Combo = CE\_ratio * Weighted\_ce) - ((1 - CE\_ratio) * Dice)$$

Metric derived Loss Function

In situations where a particular metric is used to to judge a model's performance, loss functions that derive from these metrics can be used to optimise the model towards a better classification performance. These functions typically take the form of *1 - f(x)* where *f(x)* is the metric to be optimised.

The **Intersection over Union (IoU)** metric, also referred to as the Jaccard index is a method to essentially quantify the percent overlap between the target mask and our prediction output. This calculated as the ratio between the overlap of the positive instances between two sets, and their mutual combined values. van Beers et al. [53] demonstrated how training directly for IoU can significantly increase performance for both models compared to training on conventional Binary Cross Entropy loss. Milletari et al. [54] showed promising work on how DICE loss does not need sample re-weighting when the amount of background and foreground pixels is strongly unbalanced and is indicated for binary segmentation tasks.

This approach inspired the development of a loss function to directly optimise for F1 score metric for classification models. One issue encountered with this approach is that F1 as a metric is non-differentiable, which means it cannot be used as a loss function as is. It has to be modified by accepting probabilities instead of 0/1 binaries. For instance, if the ground-truth value is 1 and the model predicts 0.4, it's calculated as 0.4 true positive and 0.6 false negative and vice-versa. Thus, this function can be minimised by means of ***1 - f(X)***. Guided by the implementation of Park [55], an implementation of that loss function was carried out and successfully presented better resulted than previously used loss functions.

3.6.2 Optimisation

During training, what controls weight updates to reduce losses is handled by the optimizer. An optimization algorithm tries to navigate the loss space by optimising towards a solution that leads to the best result possible.

Stochastic Gradient Descent

A variant of gradient Descent that updates more frequently. The parameters are updated after loss computation for each training example. This can lead to faster convergence, but can cause a high variance in model parameters which doesn't always lead to better results. To stabilise the optimisation process, the addition of mini-batches can help make the frequent updates less variant and thus leading to a more stable convergence.

Adam: Adaptive moment Estimation

A more state-of-the-art approach is Adam. The intuition behind Adam is that navigating the loss space towards a faster convergence doesn't always mean a higher learning rate. The algorithm stores an exponentially decaying average of the past gradients and squared gradients. Its various use in the literature proves its favorability to other stochastic optimizations methods which can be attributed to its computational efficiency and little memory requirements. However, in some areas, some research, showed that despite superior training time, Adam is not converging to an optimal solution in some areas, so for some tasks (such as image classification on popular CIFAR datasets) state-of-the-art results are still achieved only by applying SGD with momentum [56].

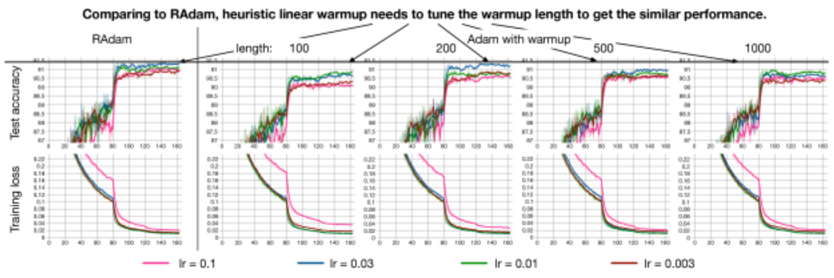


Figure 3.5: RAdam automatically provides variance reduction, outperforming manual warm ups under a variety of warmup length's and various learning rates

RAdam: Rectified Adam

A recent Adam optimizer variant which delivers an automated dynamic improvement to the adaptive learning rate based on a thorough study of the effects of variance and momentum during training. Since adaptive learning rate optimizers like Adam, RMSProp, etc. suffer from a risk of converging into a poor local optima, if a warm up method is not implemented, Liu et al. [57] found that the root issue is that adaptive learning rate optimizers have too large of a variance, especially in the early stages of training, and make excessive jumps based on limited training data, thus falling into a poor local optima. The authors then tested running Adam with no warmup, but avoided any use of momentum for the first 2000 iterations (adam-2k). They found that similar results as Adam plus warmup were achieved, thus verifying that warm-up functions as a 'variance reduction' during initial training and avoids Adam jumping into bad optima at the start when it does not have enough data to work with.

RAdam dynamically turns on or off the adaptive learning rate depending on the underlying divergence of the variance. In fact, it provides a dynamic warmup with no tunable parameters needed. The authors confirm that RAdam outperforms the traditional manual warmup tuning where the number of steps of warmup required, have to be surmised or guessed at, see Figure 3.5.

3.6.3 Cyclic learning Rates

Learning rate is another hyperparameter that requires a number of experimentation to reach an optimal result. If the rate is too slow, the neural net is going to take forever to learn, but if its too high, each step the model takes will go over the minimum and will never get to an acceptable loss. Effects of different learning rates can be seen in Figure 3.6. Smith [58] suggested an approach know as the 'One Cycle Policy' where we start training with a low learning rate and slowly climb to a higher rate. Then, taking as much time, we decrease the learning rate back down again to a lower bound than the one started with (Figure 3.7). The motivation behind this is that during the middle of learning when learning rate is higher, the learning rate works as regularisation method and keeps the network from overfitting. This

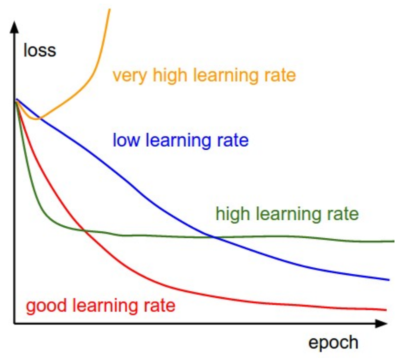


Figure 3.6: Effect of various learning rates on convergence

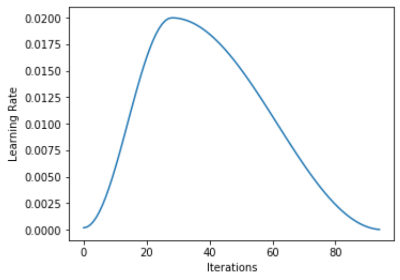


Figure 3.7: One Cycle for learning rate = 0.08–0.8

helps the network to avoid steep areas of loss and land better flatter minima. It is also worth noting that the One Cycle Policy updates the learning rate before any other adjustment to the learning rate is made (e.g.by Adam, which calculates the distinct learning rate values for each parameter using a single learning rate value and the distinct gradients of the respective parameters). This technique resulted in a phenomena known as 'Super-Convergence' where a model converges in a fewer number of epochs while using CLR, leading to a boost in performance relative to the standard training and fewer training times.

Cyclic Momentum

Momentum and learning rate are closely related, The weight update equation emphasises more on this relationship as seen below:

$$v_{iter+1} = \alpha * v_{iter} - \varepsilon * \partial L * (F(x, \theta), \theta)$$

$$\theta_{iter+1} = \theta_{iter} + v$$

Authors in [58] found in their studies that better results are obtained by reducing the momentum when the learning rate increases. This leads to faster initial convergence and greater convergence stability over a large range of learning rates.

This reinforces the idea that while the optimiser is moving quickly in new directions to find a better minima, because of the increasing learning rate, the decreasing momentum will stabilise the network not to diverge or overshoot a minima.

In practice, 2 values are chosen for momentum In step 1 we reduce momentum from higher to lower bound, and in step 2 we increase momentum from lower to higher bound, like in one cycle, we do a 2 step cycle of momentum. This cyclic momentum gives the same final results as using the best constant momentum value but stabilizes the training to allow larger learning rates. Furthermore, this approach saves time and effort to run multiple complete cycles with different momentum values.

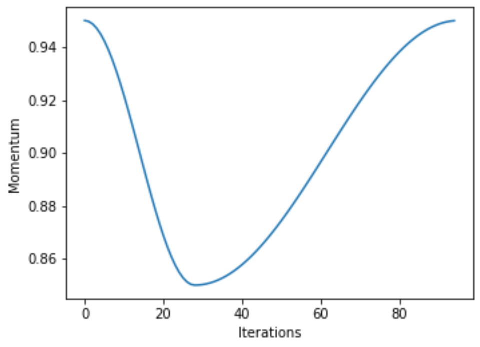


Figure 3.8: Momentum during 2 step cycle

3.7 Metric logging: WandB

In order to realise the aims of project, a lot of experimentation had to be carried out. To keep up with all the results and have a good visualisation ready, **Weights and Biases** framework was used to visualise the results of the experiments. Their cloud-hosting of the model outputs meant less load on the server that was running the training. The framework integrated really well with the project workflow with as low as 2 lines to log all the results. A lot of the graphs presented in this report are derived from using this framework.

Chapter 4

Implementation

This chapter goes over the implementation details behind every major component of the pipeline developed and details of the tools and methods chosen, some of which are mentioned in section 3 and discussed a bit more in detail here. Subsequent to the explanations of some sections are some technical specifications that clarify the languages and technologies used for the methods followed.

4.1 Process Pipeline

This section showcases a more general overview of the data flow through the components shown in figure 4.1. The process is divided into four parts, where the data flows in different sequences depending on the operation at hand; training/validation, and inference. In-depth descriptions of the particular parts will be properly discussed in later sections.

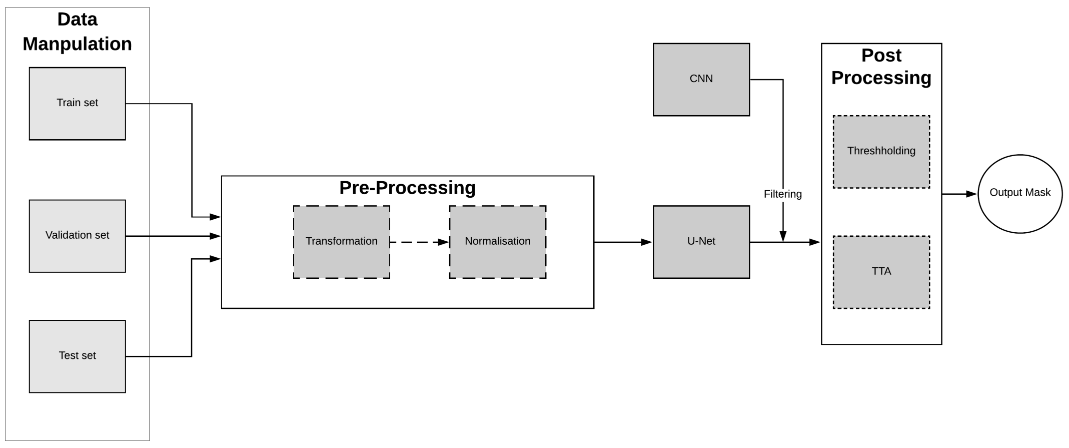


Figure 4.1: Data flow through components

4.1.1 Training/Validation sequence

The first stage of any Deep Learning system is model training. The data is resized to the desired image size and then split to training and validation sets each of which contained a batch of images. From the data manipulation phase, the data flows into preprocessing where the images are modified with various transformations, specified in section 4.2.1, that add diversity to the data input. Normalisation follows where the images are normalised using the ImageNet statistics by subtracting the mean (of ImageNet dataset) from the pixel intensity values and dividing the result by standard deviation of the dataset. So far the data has been processed so fit the input layer of the models that where the next phase starts, model training. During this phase, the data is traversed through the network multiple times, adjusting its weights and parameters. After each epoch, the predictions are run through the evaluation metrics by calculating the error rate that the model yields on the validations set. This helps asses how accuracy and consistency of the models. This process goes on for however many epochs where specified in the Model configuration section 4.3. Both models, CNN and U-Net, were trained separately and used together as specified in the Inference sequence. The post-processing stage is used at the end of the Training/Validation phase to document the performance of the resulting models. A detailed description of the post-processing techniques is discussed in section 5.

4.1.2 Inference sequence

Inference usually is done after the models are trained to benchmark their performance. The data flows in a similar manner to the flow in the Training/validation sequence, except that this time the model parameters are not tweaked at all. The models infer on the designated test image and their predictions are combined together in the following manner:

.

Algorithm 1: CNN and U-Net integration

```
1 Function Predict($inputs$):
      /* Pre-process input                                         */
2     $input =$ pre_process($inputs$)
      /* Get prediction from segmentation model on all the
         inputs                                                    */
3     $unet\_pred =$ UNET.predict($input$)
      /* Get prediction from CNN model                             */
4     $cnn\_pred =$ CNN.predict($input$)
      /* find images where the CNN predicates as a negative
         class                                                     */
5     $idx\_empty = cnn\_pred == 0$
      /* Filter segmentation model predictions                     */
6     $unet\_pred[idx\_empty] = empty\_mask\_template$
```

In essence, the CNN acts as a filter for the U-Net results. Since the U-Net is trained only to predict localise Pneumothorax for Chest-Xray images, it will probably predict masks for many of the inputs; that's where the CNN role comes in as it filters out the negative cases with an empty mask. Therefore, the final output from this ensemble is an array of masks where only positive cases have pixel-level predictions locating the disease. The last phase is post-processing, chapter 5, where the different techniques are used to optimise the models prediction for the best result.

4.2 Dataset

Medical datasets are hard to come by because of the time it takes to annotate and collect them. Fortunately, Society for Imaging Informatics in Medicine (SIIM) provided a dataset on Kaggle, compromising of 12,047 images in DICOM[1] format. Since the Neural Network used in this project can't process DICOM data, they had to be converted into PNGs and resized to fit the input layer. The images consist of Chest X-rays taken from AP(anterior of the body to posterior) or PA(posterior of the body to anterior) views. The training data is split 22% positive to 78% negative cases. As the masks are created from pixels, we can easily count them to calculate the area. As we can see, figure 4.2, the majority of ill patients suffer from a smaller Pneumothorax, and some patients have a severely collapsed lung.

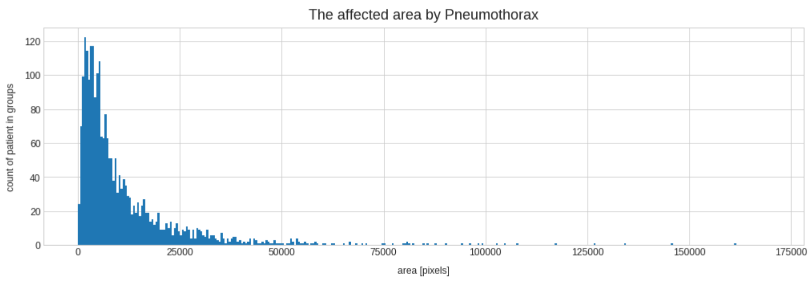


Figure 4.2: Affected area by Pneumothorax

Usually, AP view is used for x-rays, but in this case, the chest x-rays are rather taken from the PA view. If the health level of the patient does not allow to do PA, AP can also help. This is reinforced when analysing the area of the masks in relation to view position, figure 4.3. The ”AP” positioned x-ray images seem to have more serious Pneumothorax with the biggest masks made in AP position, however, this view position occurs fewer times than PA. To get a general insight into the location of Pneumothorax, ”lung heatmaps” area created for AP and PA views. Of course, every patient is different, thus the size of the lungs and their place on the x-ray image can be different. However, a heatmap may give an overview. Figure 4.4 shows that in both cases the top left corner of the lung is the most common location. In the PA cases, the x-ray images create a cleaner view, the lungs are outlined, while the general lungs of AP cases are noisier and have orange tone. The color-bar indicates the maximum number of overlapped pixels.

[1](Digital Imaging and Communications in Medicine) an international standard to transmit and store medical imaging information

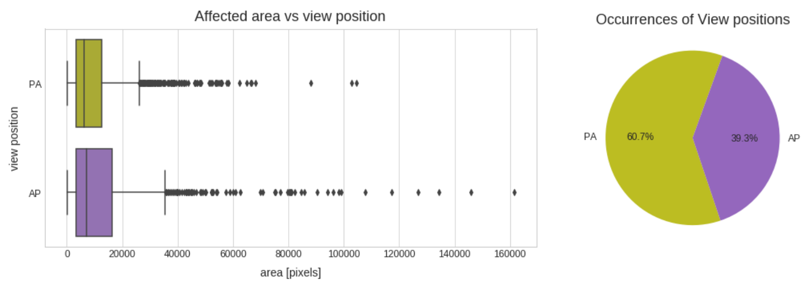


Figure 4.3: View distribution

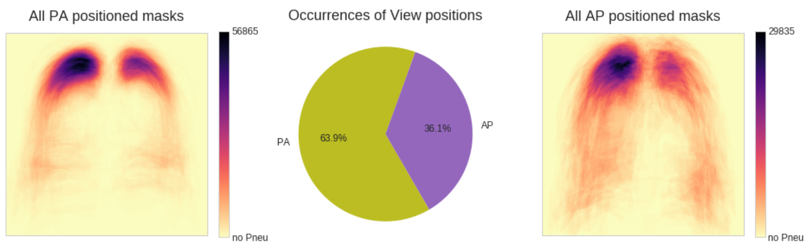


Figure 4.4: Lung Heat maps

4.2.1 Data Preprocessing

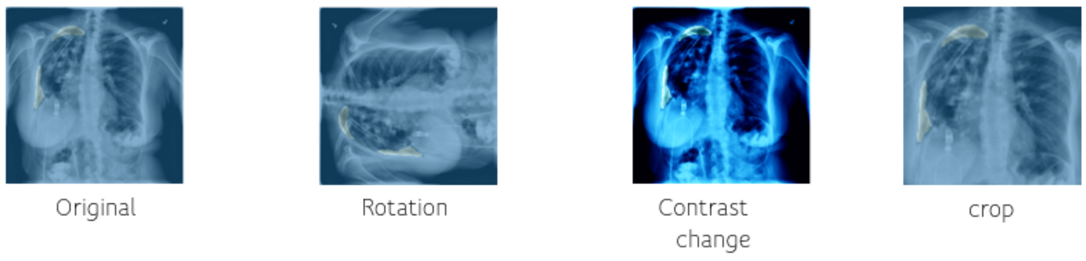


Figure 4.5: Example of transformations applied

Any pattern recognition model should be able predictions that are invariant to various inputs of a given label. The straightforward way is to gather a large collection of training samples with enough variation. The scarcity of training examples predominant in medical datasets makes using such straightforward techniques unrealistic. To solve this, data augmentation is used as means to increase data in the training set. Various affine transformations have been shown to improve model performance as discussed in section 3.4 an example of the ones implemented in the training of the models is seen in figure 4.5. To that end the following augmentations were carried out before the training of any model implemented in this project.

Table 4.1: Transformations applied to models

Transformation	**Probability**	**Magnitude**
Flip	0.5	-
symmetric_warp	0.75	(-0.2, 0.2)
rotate	0.75	(-10.0, 10.0)
zoom	0.75	(1.0, 1.1)
brightness	0.75	(0.4, 0.6)
contrast	0.75	(0.8, 1.25)

4.3 Baseline

Before testing the proposed architecture, a baseline is needed as a benchmark for the results. For this project, a U-Net was chosen with the following configuration: This baseline was

Table 4.2: Baseline configuration

Architecture of Encoder	ResNet
Number of layers	34
Optimizer	Adam
Loss Function	Jaccard/Iou Loss
Input image size	256
Number of epochs	6(frozen) + 12(un-frozen)

chosen as it represents a typical solution used for segmentation problems in this subject area. As discussed in section 2.4, this architecture was utilised successfully in many medical image segmentation problems as it provides a joint classification and localisation capabilities [13]. Therefore, a similar architecture was chosen with a 34-layer residual encoder. The number of layers was chosen as 34 because it provides a fair middle point between 18 and 50 layer architectures in terms of performance computational cost. *JaccardLoss()* function was chosen to optimise for better intersection results. This model is gonna be used to demonstrate and benchmark the performance of the proposed architecture in section 5.

4.4 Proposed Model

The proposed architecture, figure 4.6, utilities an approach inspired by one class training. Training on one class in segmentation problems is unattainable as pixels on their own don't hold any informative value. Instead, we train on ”One Case”, namely the anomaly or disease. In Binary segmentation problems, the data is split between data points that have empty masks and ones that have masks with localisation information. The intuition behind this project is that if we completely ignore the empty mask data-points, and focus the model on just learning the images with the anomaly in them, ie. non-empty masks. Therefore, the network

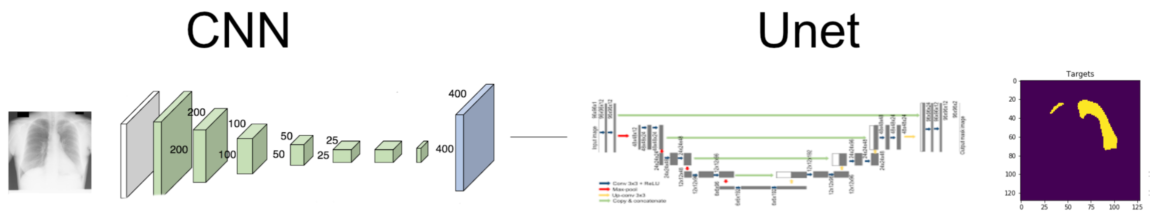


Figure 4.6: Proposed pipeline

concentrates on learning discriminate features for the anomaly. This will naturally mean that the network may predict masks with positive pixels even when there is no disease in the image. To counteract that, a CNN is used to zero-out wrongly predicted masks, thus fulfilling the classification and segmentation task. The results in section 5.3 demonstrate the effectiveness of this approach. The following sections discuss the implementation that came about as a result of the above proposal.

4.4.1 The CNN

The first step to bring this proposal to reality is to see to heف training of the network that is gonna handle the first step, filtering out cases with no pathology, ie. 'Normal' cases. This will leave the U-Net to focus on arranging a set of filters for the separation of pathology from the infected lungs.

To train network to discriminate features for the health vs unhealthy lungs, the following architecture configuration was chosen: This implementation utilities the ResNeXt

Table 4.3: CNN configuration

Architecture of Encoder	ResNeXt
Number of layers	50 with capacity = 32
Optimizer	RAdam
Loss Function	F1 loss
Input image size	224
Number of epochs	127

architecture discussed in section 3.1. The versatility and adaptability of this architecture make a good fit for this dataset. The ability to stack multiple building blocks of the same shape is a strategy that allows for a reduction of the free choices of hyperparameters [45]. The ResNeXt blocks inherit the bottleneck design of ResNets with the addition of separate paths or dimensions that add depth and "Cardinality" to each neuron, as seen in figure 4.7.

The model was trained on the full SIIM dataset, 2379 (disease) and 8296 (healthy) images. Multiple augmentations were used on the data, including flipping, zooming, and

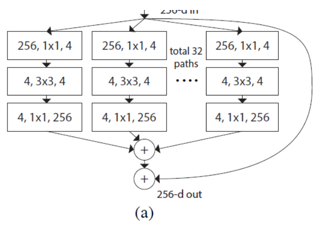


Figure 4.7: Building block of ResNeXt architecture

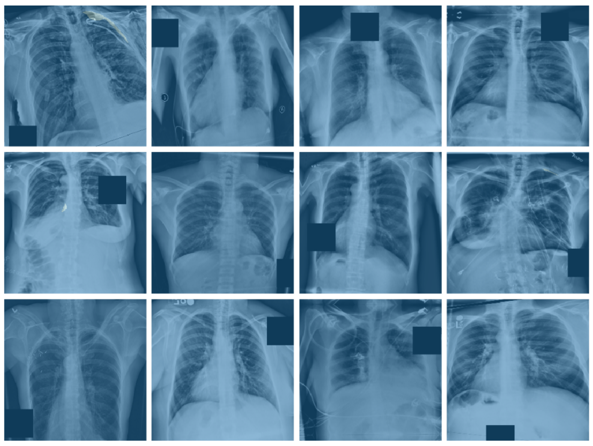


Figure 4.8: Visualisation of the Cutout + other augmentations applied before training

symmetric wrap. After around 70 epochs, a new augmentation was introduced, Cutout which randomly removes small patches from the images as seen in figure 4.8. This technique has been shown to prevent models from overfitting to the data and lead to a better generalisation performance.

4.4.2 The U-Net

Moving to the main part of the project, the ”One Case” U-Net. As seen in the pipeline overview section 4.1, the U-Net is used to predict masks for the input images and then have its results filtered-out using a CNN. Since the network only has to predict correctly for positive cases, it only makes sense to train it on only positive instances. This meant to trim the dataset to 2379 images, 20% of which was used for the validation set leaving only 1903 images for the training cycle. The U-Net utilised a residual network with 34 layers in the following configuration: The intuition here is to train the network with as close of a configuration to the baseline U-Net as possible so that the bench-marking is as fair as possible and to demonstrate the effectiveness of the approach. This meant using the same cyclic learning rate ranges, number of epochs, loss function, and optimiser. Different configurations were experimented with and the results are outlined in section 5.3.

Table 4.4: Proposed U-Net, trained only on positive cases

Architecture of Encoder	ResNet
Number of layers	34
Optimizer	Adam
Loss Function	Jaccard Loss
Input image size	256
Number of epochs	18

4.5 Technical Specifications

All the models were implemented in Python 3.7 using Pytorch Deep Learning library. The encoders were imported from the Fastai library as their training loop functions offered multiple pre-trained models along with various data augmentation functions. For the CNN however, the ResNeXt model was imported from a GitHub repository that had multiple pre-trained Pytorch models [59]. Some other libraries were used in some aspects of pre-loading the data and post-processing it such as NumPy, Pandas, and sklearn. Almost all the training was done on university servers, specifically using a GTX 2080 graphics card that helped in facilitating the training time and processing the models.

Chapter 5

Evaluation and Results

This chapter discusses the evaluation methods used to determine the effectiveness of the models used. It also includes overall results obtained from experiments and a justification for the best results.

5.1 Evaluation Metrics

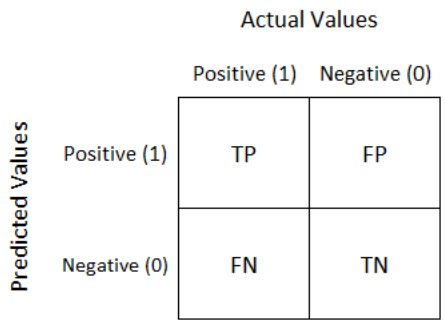


Figure 5.1: Confusion matrix

In tasks like Binary Classification, there are many ways to measure the performance of a classifier. Yet, not all of them accurately measure the generalisation ability of said model. For instance, **Accuracy** is usually an obvious method in evaluating performance, but in the Neural Networks case, it can be greatly misleading. While counting the number of correctly predicted samples and dividing by the total number of samples sounds intuitive enough, it doesn't offer a good representation of the overall performance of the model. Given a dataset with a class ratio of 1000:1, if a model predicts towards the majority class on all the samples, it would have an *Accuracy* of 90%. This leads to a misleading belief that the model is performing quite well, while, in fact, it's practically unusable in any application. Therefore, given that the dataset in this project does have a similar class imbalance and the fact that in healthcare applications misclassifications can have fatal outcomes, Accuracy wasn't used as a metric for evaluation.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

A more useful metric is **Precision** that measures the number of True Positives (TP) over the total amount of Positives (TP + FP). It answers the question about which proportion of the predicted positives is truly Positive.

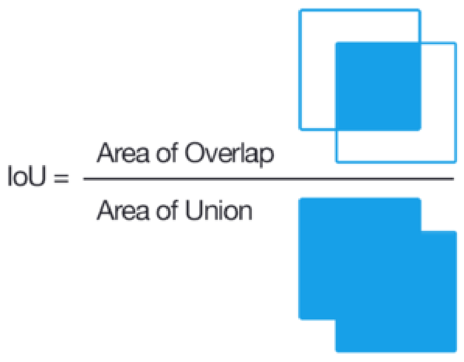
$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Recall, on the other hand, measures the amount of TP over the predicted (TP+FN). It concerns itself with the proportion of True positives and how much of those were correctly predicted or recalled. Recall is a valid choice to use when trying to develop something that can capture as many positives as possible, for example, a medical algorithm that predicts a certain pathology. Precision, however, is more suitable in the development when the goal is to be very sure of whatever is predicted.

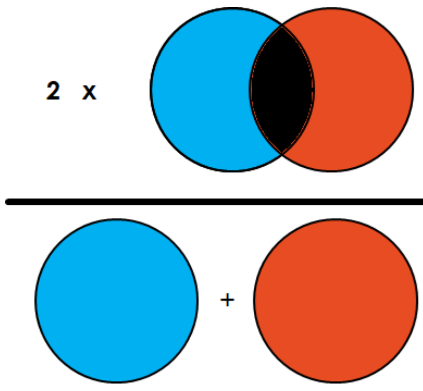
A really important metric that bridges the gap between Precision and recall is **F1 Score**, otherwise knows as the harmonic mean between the two. The main advantage of using F1 is that it maintains a balance between precision and recall for any classifier, i.e. if Precision is low, the F1 is low and if recall is low F1 is low.

$$F_1 = 2 * \frac{precision \cdot recall}{precision + recall}$$

For Semantic segmentation there are mainly two metrics used for evaluating the U-Net Jaccard/IOU and Dice score [60]:



(a) Jaccard



(b) Dice

Figure 5.2: Jaccard and Dice calculation visualized

Jaccard Index or IoU (Intersection over Union) is one of the most commonly used metrics in segmentation as it is very simple to implement and understand. As seen in Figure 5.2a, it is the area of overlap between the prediction and the ground truth divided by the area of union between the prediction segmentation and ground truth.

Dice Score is pretty similar to Jaccard, it is (2x) the area of overlap divided by the total number of pixels in both predicted and target images (Figure 5.2b). Both metrics are positively correlated, meaning if one says model A is better than model B at segmenting an image, then the other will say the same. The difference becomes clearer when taking the average over a set of inferences. As willem [61] mentions, the IoU metric tends to penalize single instances of bad classification more than the Dice score quantitatively even when they can both agree that this one instance is bad. The *Dice score* appears to measure something similar to average performance, whereas the IoU score measures something closer to the performance of the worst case. Another variation of **Dice Score** where the Dice coefficient is defined as 1 when both predicted and True masks are empty and the final score is the mean if Dice Score for each image in the set. So instead of treating all image pixels as one set, every image gets a score and then the average is taken as the model's final score. This **Dice_adjusted** metric was ported from the Dataset organizers page where they used this metric to benchmark models submitted to them for evaluation.

5.2 post-processing

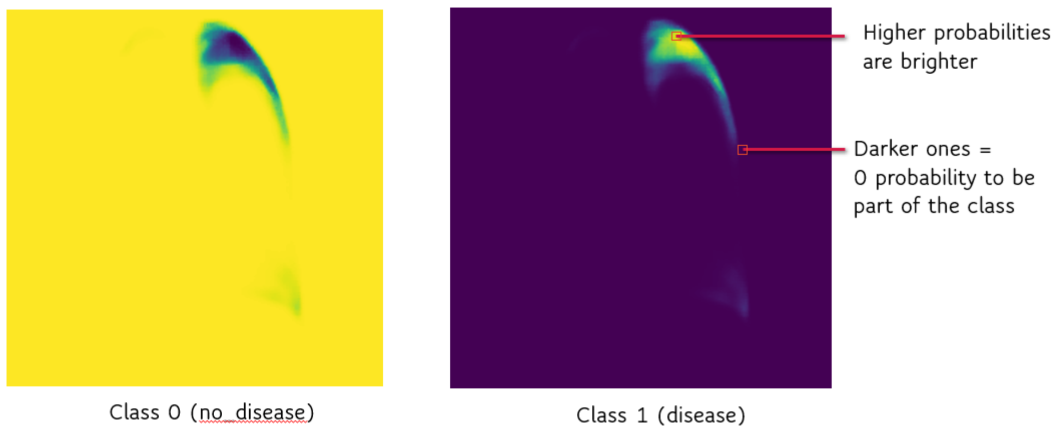
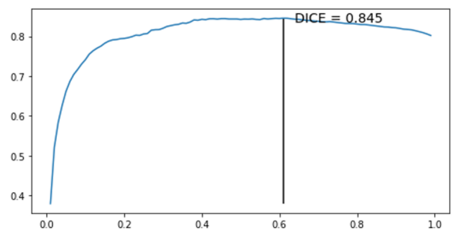

Higher probabilities are brighter
Darker ones = 0 probability to be part of the class
Class 0 (no_disease)
Class 1 (disease)

Figure 5.3: Segmentation model outputs

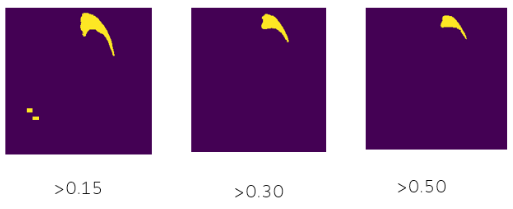
There are three main methods used to post-process the predictions from the segmentation models. A typical output is visualised in figure 5.3. Since this binary classification problem, each pixel is classified as either negative (No Pnumeothorax) or positive (with Pnumeothorax). The results are two output masks for each input image, each mask represents a probability for each pixel belonging to said class. A higher probability translates to a higher intensity value which translates into brighter pixels and vice versa. The outputs at that stage cannot be used for evaluation with the ground truth masks as they are binary masks and the models' outputs are 2 probability masks. Therefore, these masks need to be processed into a binary representation.

The first conventional way is to assign the pixel to the class with the highest probability. This is done using the **ArgMax()** function that returns the index of the highest number in an array, in this case the index is the class number. This approach usually results in smaller masks as it aggressively cuts down any pixel with a low probability, demanding high confidence in predictions. While it sounds intuitive to only keep higher confidence results, in datasets with high class-imbalance, higher confidence probabilities are usually biased the majority class as it dominates the distribution. In this application on healthcare, it might also misrepresent the shape of the affected area as, usually, confidence is higher at the center of the anomaly and gets lowers at the boundaries.

A different method is then introduced that is **Thresholding**, in which a cut-off threshold is explored in one of the masks to allow pixels above a certain probability to be classified as class 1 (disease) and anything below that threshold is classified as class 0 (No disease). Figure 5.4a illustrates this search process in which a series of thresholds between (0.01, 1) are tested to determine the best in terms of accuracy. A threshold acts as a parameter of how leniently the models' outputs are treated, higher thresholds can lead to a lower number of positive pixels and lower ones can lead to some noise leaking in, as seen in figure 5.4b. Fundamentally, this method allows for more control of the model predictions and allows for a much better interpretation of the results. During evaluation, each model's results undergo a search process for a threshold that best maximises the models' performance.



(a) Threshold testing



(b) Effects of different thresholds

Figure 5.4: Illustration of thresholding model outputs

Another commonly used method in post-processing models' outputs to reduce generalization error and improve performance is Test-time augmentation, or TTA [62]. It is a process that involves applying data augmentation on inference input images and creating multiple copies of the image, having the model predict on each image and return an average of the results. This, essentially, gives the model a better chance to correctly classify a given image by showing it a slightly modified version of the input. This method consistently resulted in higher segmentation accuracy compared to predicting based on original images only.

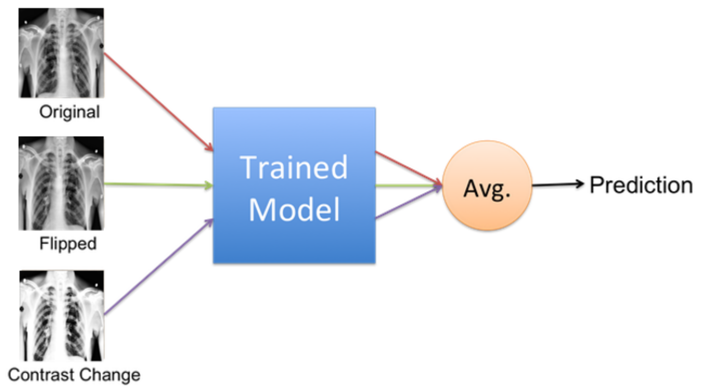


Figure 5.5: Test-time augmentation

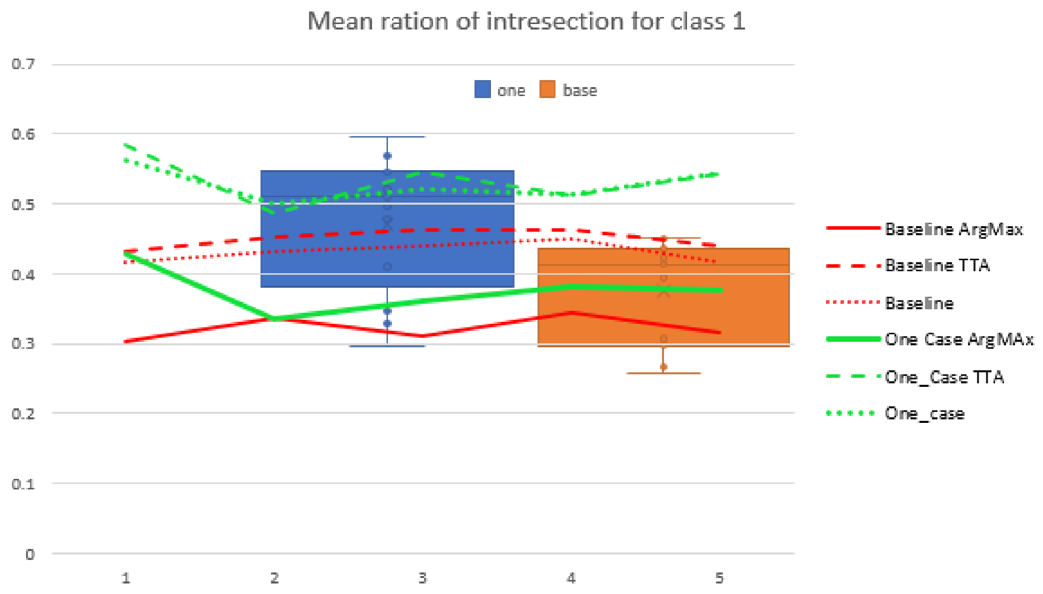
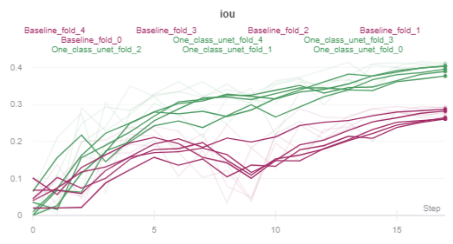


Figure 5.6: Mean ratio of intersection results

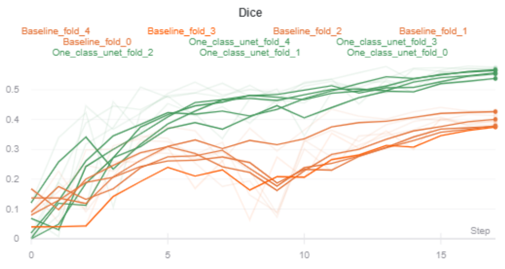
5.3 Results and Discussion

To demonstrate the effectiveness of the proposed architecture, multiple experiments carried out, some were mentioned in section 3 and the rest are discussed and illustrated in this section. To ensure the evaluation is consistent, robust, and complete, the testing has been carried out on all the samples with the same testing (same threshold) method and achieved the same result regardless of the fold. This also includes a report on the metrics per cross validation folds, giving average metrics and standard deviations. Throughout this section, the proposed architecture is referred to as ”One Case” model and results displayed are after the processing step specified in algorithm 1.

Figure 5.6 illustrates the results of the mean ratio of intersection of class 1 (disease), demonstrating how well a model’s prediction intersects, the positive pixels, with the ground truths. As seen in the graph, the One Case model outperforms the baseline in every fold.



(a)



(b)

Figure 5.7: Results during training

	Dice per image		Dice		IoU		CM	
	One Case	*Baseline*	*One Case*	*Baseline*	*One Case*	*Baseline*	*One Case*	*Baseline*
ArgMax	**0.86876(±0.005)**	0.83382(±0.005)	0.61824(±0.02)	**0.62312(±0.01)**	0.44778(±0.03)	**0.45266(±0.01)**	**0.377018(±0.03)**	0.321465(±0.02)
Raw	**0.88212(±0.005)**	0.815(±0.007)	**0.64766(±0.02)**	0.63718(±0.004)	**0.4792(±0.02)**	0.47216(±0.01)	**0.54681(±0.04)**	0.489224(±0.02)
TTA	**0.88482(±0.004)**	0.82726(±0.007)	**0.66632(±0.016)**	0.65664(±0.006)	**0.4998(±0.02)**	0.4842(±0.01)	**0.562933(±0.04)**	0.49073(±0.02)

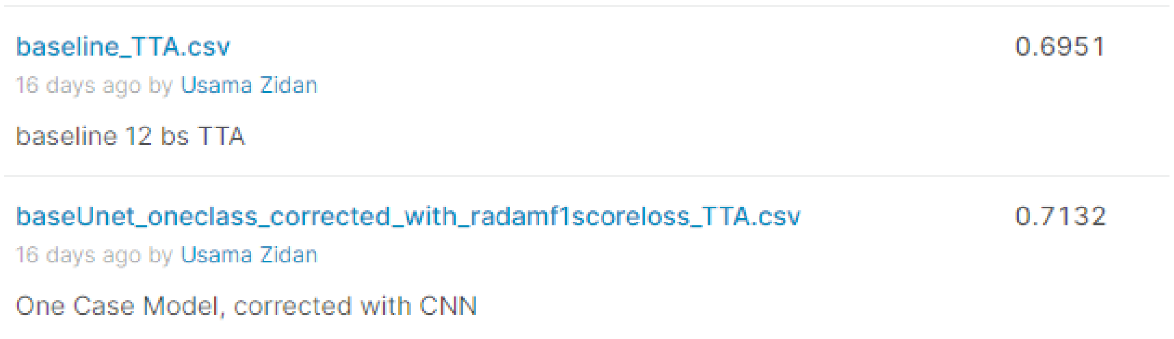
Table 5.1: Averaged folds results

This means that, generally, the One Case predicts masks that fit the actual true masks better. This reinforces the idea that training in positives masks only can lead to better performance, intersection-wise.

Table 5.1 shows the results obtained from 5-fold cross validation in which each model was validated 5 times against different validation sets and then tested on a different test set. Due to space constraints, the table displays average results of the 5 folds after applying each of the different post-processing techniques mentioned in section 5.2. Looking at the average *Dice per image*, One Case model does offer a significant improvement in single image cases. Yet, when it comes to IoU and Dice metrics, the Baseline gets a slight advantage. Since these metrics asses all the samples as a whole not in an individualistic manner, it can be argued that in real-life applications where images are served individually for assessment, a model that performs well 'per image' is preferred. To get a more qualitative look at the results, figure 5.8b shows the difference between the outputs of the models and demonstrates the better segmentation of One Case model. The addition of a CNN proves useful when looking at times when the baseline misses an image completely, passing it as a healthy image while it is an unhealthy one, figure 5.8a. This supports the proposed pipeline as it offers better classification and segmentation results.

During the training process, figure 5.7 shows how IoU and Dice metrics are dominated by the One Case model, this can be attributed to the fact that the One Case model was trained on just positive cases, thus allowing the model to focus more on learning patterns for the disease. On the other hand, the baseline model is overwhelmed by the significantly more prominent negative pixels in the training set.

To get a more *'real-world'* idea of how the models performed, both models were submitted to the Dataset site to be evaluated on their test set. From the results, seen below, we can observe the performance bump that One Case has over the baseline. Even though the results don't rank high in the leader-boards, coupled with more complex architectures, the same pipeline can offer a performance boost to other methods, more on that in chapter 6.



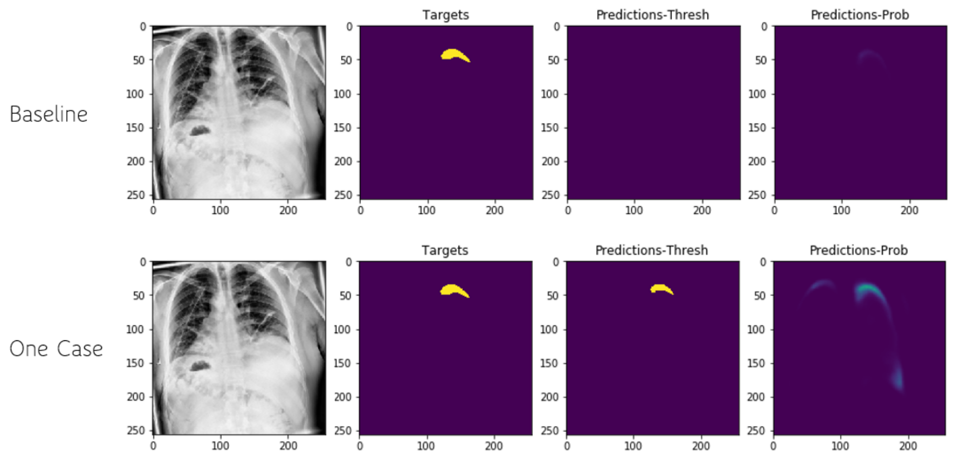
baseline_TTA.csv 0.6951
16 days ago by Usama Zidan
baseline 12 bs TTA

baseUnet_oneclass_corrected_with_radamf1scoreloss_TTA.csv 0.7132
16 days ago by Usama Zidan
One Case Model, corrected with CNN

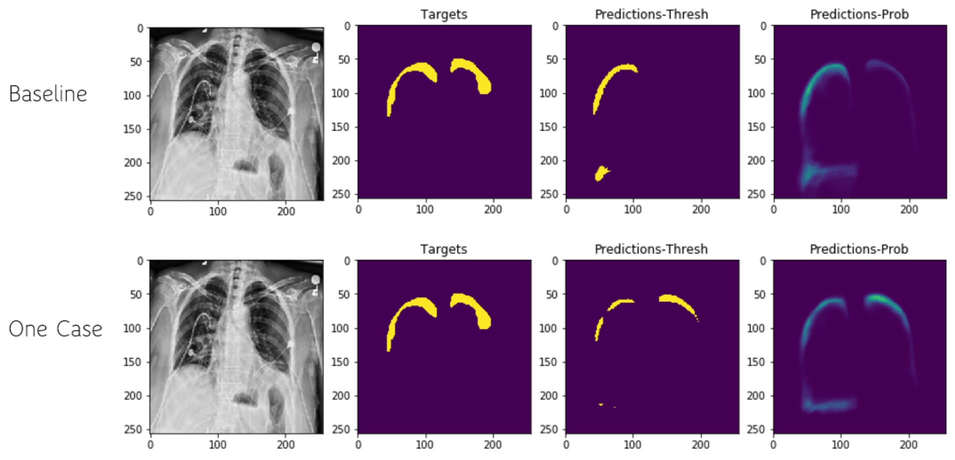
Finally, the improvements demonstrated by One Case model supports the hypothesise of this report and can be credited to the following attributes:

- Focusing the U-Net on positive cases allowed for more pattern recognition of the unique structures to the disease
- Having a CNN as a filter for predictions provides better classification of the input
- Task-specific training can offer better results, and may be used as means to counteract the over-fitting caused by class imbalance

These experiments highlight how Neural Networks can become biased towards majority classes in datasets. In segmentation datasets, the problem becomes much more prevalent as the ratio of positive to negative pixels is huge, causing networks to recognise the background more than the foreground. While this can lead to a network that can confidently recognise the background patterns, and thereby isolating the foreground elements, it does cause the network to be overwhelmed and have a huge false negative rate which in some applications can be critical.



(a)



(b)

Figure 5.8: (a) Output probabilities of both models on a test image, baseline completely misses the disease. (b) Shows the difference between both model's output when they both predict positively.

Chapter 6

Reflections and Future Work

The trick with any lengthy research project is knowing where to stop. The strife for better results can sometimes be a mirage leading us to believe that the curve can go higher. Yet since we can't have all the time in the world to dive into various projects till we see their groves. Given the time spent on this project, there are some modifications that would've probably made a difference in the results and implementation of the project components. These ”technical Reflections” are broken down in this chapter emphasising reflections in the research phase, the implementation phase, and some personal reflections on the project as a whole alongside some approaches that are expected to fix the drawbacks that the project suffers from.

6.1 Data

Although the dataset used here is usually the same size as other datasets used in related applications, testing the models on different datasets that have pixel-level annotations for pneumothorax images would have given more insight on the generalisability of the models and offer a comparison to other approaches.

As it's known in the scientific community, experimentation is a major aspect in proving/disproving a hypothesis. Therefore, multiple experiments can show the impact of using different augmentation techniques and whether that is of impact on the performance of predicting on positive pixels. Even though lots of trials have been done on the impact of different parameters in this project, the impact of different image and batch sizes wasn't assessed mostly because of computational limitations. Nonetheless, some related work has suggested that training on higher resolutions leads to better results. Some other techniques can be used to train on higher resolutions images using hard-positive mining by portioning a big image into patches and feeding the network more instances of patches that contain positive pixels. various other techniques can be used to pre-process the data to focus the network more towards learning the ’disease’ aspect of the images such as cropping, etc.

6.2 Model hyperparameter

Neural network training depends heavily on the hyperparameters chosen. From loss functions to learning rates, these parameters determine how the weights get tuned w.r.t models error on the training set. A lot of loss functions were experimented within this project, yet the extent of these trials wasn't as in-depth as done in the literature. Some functions had their own hyperparameters that needed to be tuned for each use-case. Future work can benefit from trying to fine-tune one of the many segmentation loss function to the requirements of medical images.

Optimisation relies heavily on the choice of learning rate as it is used to determine how much change is done to the network weights. Since cyclic learning rates is has proved to, evidently, useful, evaluating the effect of different ranges of learning rates in conjunction with different optimisers can benefit the overall accuracy of the model. These hyperparameters are responsible for how quickly the model converges. Therefore, it is recommended that further research is spent in tuning these parameters to observe which pairs lead to better results and maybe even a reason for why is that so.

6.3 Evaluation methods

The realm of machine learning and medical informative look at neural network models with different perspectives. One values accuracy overall, while the other looks more onto the effects of that accuracy in a real-world application. The models in this paper were evaluates based on how well they match the ground truth and while that is an important first step in determining the effectiveness of any model, what remains is how valuable is the information that the model provides in a clinical environment. To assess these attributes, further study is needed to determine how to better evaluate ANN models for medical images and workflows.

Circling back to the proposed model, the results section outlines the effect of focusing a network on positive cases and one on filtering-out negative ones. What is yet to be determined is the extent to which the classification model helped the segmentation one and which had more of an influence on the performance and why. These questions need further study to get a better understanding of the effects that negative cases might have on the outputs of a network and how that plays a role given the huge class imbalance problem that medical image datasets suffer from.

6.4 Models used

Even though this project didn't set out to transcend the stat-of-the-art, it provided evidence that can help the stat-of-the-art transcend in itself. If training on positive cases can help a simple model, like the one set as the baseline, it can hypothesised that these findings can be propagated through to other models, resulting in a better performance in general. Furthermore, models like FrCNs, RNNs, etc. had shown great performance in segmentation tasks in the literature and further work in developing pipelines that take these models into account can result in better, applicable, performance.

6.5 Future clinical applications

This project focused on the segmentation of a disease, namely pneumothorax, and developing a pipeline to better enhance that process. While it still remains to be answered how effective are these models in clinical applications, the process implemented can carry forward to other areas in the medical field. Other diseases, lesions in the lungs, and even 3D reconstruction from CTs are all other areas that training on a positive case pipeline can be explored as a component of a CAD system that can deployed in a hospital someday.

Another application of can see the deployment of these models, after training, in a mobile application that can directly and efficiently provide a diagnosis on an uploaded X-Ray image. Some variation of that was explored for this project but in a web-based approach but due to the time-consuming manner of developing a web-based application that incorporates a neural network model this is proposed as future work.

6.6 Personal Reflections

This project deems to show the huge potential that neural networks can offer in the medical field. The research is overwhelmingly loaded with many ideas and implementations from novel architectures to ensembles of everything out there. Like any junior scientist/researcher, the excitement of implementing the knowledge gained from all that reading, lead to a ”Not-thought through” intermediate implementation that wasn't included in the final submission but took up a substantial amount of time between doing the research and the final implementation. For example, some experimentation with the loss functions, as seen in section 3, that didn't make it in the final process because of a compatibility issue with the libraries used. Yet, these hiccups along the way were extremely beneficial in facilitating my knowledge by experimenting with the frameworks and libraries available.

Throughout my academic life, this has been the biggest endeavor I faced, yet it is the most exciting. From not knowing what exactly is the next step but knowing that the answer is somewhere out there in some article or research paper is what makes machine learning projects always interesting to tackle. They operate on the edge of the new technologies and have a great sense of excitement when it comes to working on them. At times, the research would be overwhelming to understand but I, honestly, am glad to have taken this project as my dissertation as it never failed to introduce me to interesting topics in the industry and its applications.

Chapter 7

Conclusion

Deep learning has revolutionised the field of computer vision detection with many Deep models becoming the state-of-the-art in benchmarking datasets. This begs the question of well can these models perform in medical image datasets. Multiple studies have dived into these aspects but what this project tries to offer is a new process/pipeline in training and evaluating Deep models in medical imaging datasets. By isolating positive masks and allowing a network to focus on segmenting the disease, this project aims to explore the effects of negative masks in training Fully Connected Convolutional Networks (FCNs) and introduces a new pipeline that achieves better results than traditional methods. As a result, it was demonstrated that the proposed pipeline can offer performance improvements when it comes to segmenting a pathology form images. The results support the hypothesise of the project and highlight how Neural Networks can become biased towards majority classes in datasets.

References

[1] G. Wu, M. Kim, Q. Wang, Y. Gao, S. Liao, and D. Shen, "Unsupervised deep feature learning for deformable registration of mr brain images," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2013, pp. 649–656.

[2] F. Commandeur, M. Goeller, J. Betancur, S. Cadet, M. Doris, X. Chen, D. S. Berman, P. J. Slomka, B. K. Tamarappoo, and D. Dey, "Deep Learning for Quantification of Epicardial and Thoracic Adipose Tissue from Non-Contrast CT," *IEEE Transactions on Medical Imaging*, vol. 37, no. 8, pp. 1835–1846, 8 2018.

[3] L. Yarmus and D. Feller-Kopman, "Pneumothorax in the critically ill patient," *Chest*, vol. 141, no. 4, pp. 1098–1105, 4 2012.

[4] H. B. Harvey, T. K. Alkasab, P. V. Pandharipande, J. Zhao, E. F. Halpern, G. M. Salazar, H. H. Abujudeh, D. I. Rosenthal, and G. S. Gazelle, "Radiologist compliance with institutional guidelines for use of nonroutine communication of diagnostic imaging results," *Journal of the American College of Radiology*, vol. 12, no. 4, pp. 376–384, 4 2015.

[5] T. Montaque, E. K. Fishman, and S. P. Rowe, "The Future of Digital Communication: Improved Messaging Context, Artificial Intelligence, and Your Privacy," *Journal of the American College of Radiology*, 1 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1546144019314784

[6] S. Hussain, "Communicating Critical Results in Radiology," *Journal of the American College of Radiology*, vol. 7, no. 2, pp. 148–151, 2 2010.

[7] Y. Taigman, M. Y. Marc', A. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," Tech. Rep., 2014.

[8] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January. Institute of Electrical and Electronics Engineers Inc., 11 2017, pp. 3462–3471.

[9] L. M. Prevedello, B. S. Erdal, J. L. Ryu, K. J. Little, M. Demirer, S. Qian, and R. D. White, "Automated Critical Test Findings Identification and Online Notification System Using Artificial Intelligence in Imaging," *Radiology*, vol. 285, no. 3, pp. 923–931, 12 2017. [Online]. Available: http://pubs.rsna.org/doi/10.1148/radiol.2017162664

[10] T. A. Retson, A. H. Besser, S. Sall, D. Golden, and A. Hsiao, "Machine learning and deep neural networks in thoracic and cardiovascular imaging," pp. 192–201, 5 2019.

[11] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, R. Kim, R. Raman, P. C. Nelson, J. L. Mega, and D. R. Webster, “Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs,” *JAMA - Journal of the American Medical Association*, vol. 316, no. 22, pp. 2402–2410, 12 2016.

[12] B. Lassen, E. M. Van Rikxoort, M. Schmidt, S. Kerkstra, B. Van Ginneken, and J. M. Kuhnigk, “Automatic segmentation of the pulmonary lobes from chest CT scans based on Fissures, Vessels, and Bronchi,” *IEEE Transactions on Medical Imaging*, vol. 32, no. 2, pp. 210–222, 2013.

[13] A. Gooßen, H. Deshpande, T. Harder, E. Schwab, I. Baltruschat, T. Mabotuwana, N. Cross, and A. Saalbach, “Deep Learning for Pneumothorax Detection and Localization in Chest Radiographs,” Tech. Rep., 2019.

[14] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng, “CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning,” *arXiv preprint arXiv:1711.05225*, 11 2017. [Online]. Available: http://arxiv.org/abs/1711.05225

[15] J. P. Cohen, P. Bertin, and V. Frappier, “Chester: A Web Delivered Locally Computed Chest X-Ray Disease Prediction System,” *arXiv preprint arXiv:1901.11210*, 1 2019. [Online]. Available: http://arxiv.org/abs/1901.11210

[16] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” pp. 436–444, 5 2015.

[17] W. Zhang, R. Li, H. Deng, L. Wang, W. Lin, S. Ji, and D. Shen, “Deep convolutional neural networks for multi-modality isointense infant brain image segmentation,” *NeuroImage*, vol. 108, pp. 214–224, 3 2015.

[18] J. Kleesiek, G. Urban, A. Hubert, D. Schwarz, K. Maier-Hein, M. Bendszus, and A. Biller, “Deep MRI brain extraction: A 3D convolutional neural network for skull stripping,” *NeuroImage*, vol. 129, pp. 460–469, 4 2016.

[19] H. I. Suk, S. W. Lee, and D. Shen, “Hierarchical feature representation and multimodal fusion with deep learning for AD/MCI diagnosis,” *NeuroImage*, vol. 101, pp. 569–582, 11 2014.

[20] H.-C. Shin, K. Roberts, L. Lu, D. Demner-Fushman, J. Yao, and R. M. Summers, “Learning to Read Chest X-Rays: Recurrent Neural Cascade Model for Automated Image Annotation,” Tech. Rep., 2016.

[21] H. I. Suk, S. W. Lee, and D. Shen, “Latent feature representation with stacked auto-encoder for AD/MCI diagnosis,” *Brain Structure and Function*, vol. 220, no. 2, pp. 841–859, 12 2015.

[22] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, “Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1240–1251, 5 2016.

[23] L. Roux, D. Racoceanu, N. Loménie, M. Kulikova, H. Irshad, J. Klossa, F. Capron, C. Genestie, G. Naour, and M. Gurcan, “Mitosis detection in breast cancer histological images An ICPR 2012 contest,” *Journal of Pathology Informatics*, vol. 4, no. 1, p. 8, 2013.

[24] G. Wu, M. Kim, Q. Wang, B. C. Munsell, and D. Shen, “Scalable high-performance image registration framework by unsupervised deep feature representations learning,” *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 7, pp. 1505–1516, 2015.

[25] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” pp. 611–629, 8 2018.

[26] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, “Medical image classification with convolutional neural network,” in *2014 13th International Conference on Control Automation Robotics and Vision, ICARCV 2014*. Institute of Electrical and Electronics Engineers Inc., 2014, pp. 844–848.

[27] V. P. Vianna, “Study and development of a Computer-Aided Diagnosis system for classification of chest x-ray images using convolutional neural networks pre-trained for ImageNet and data augmentation,” *arXiv preprint arXiv:1806.00839*, 6 2018. [Online]. Available: http://arxiv.org/abs/1806.00839

[28] S. S. Yadav and S. M. Jadhav, “Deep convolutional neural network based medical image classification for disease diagnosis,” *Journal of Big Data*, vol. 6, no. 1, pp. 1–18, 12 2019.

[29] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[30] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[31] Y. Gordienko, P. Gang, J. Hui, W. Zeng, Y. Kochura, O. Alienin, O. Rokovyi, and S. Stirenko, “Deep learning with lung segmentation and bone shadow exclusion techniques for chest X-ray analysis of lung cancer,” in *Advances in Intelligent Systems and Computing*, vol. 754. Springer Verlag, 2019, pp. 638–647.

[32] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3d u-net: learning dense volumetric segmentation from sparse annotation,” in *International conference on medical image computing and computer-assisted intervention*. Springer, 2016, pp. 424–432.

[33] G. Zeng, X. Yang, J. Li, L. Yu, P.-A. Heng, and G. Zheng, “3d u-net with multi-level deep supervision: fully automatic segmentation of proximal femur in 3d mr images,” in *International workshop on machine learning in medical imaging*. Springer, 2017, pp. 274–282.

[34] P. F. Christ, F. Ettlinger, F. Grün, M. E. A. Elshaera, J. Lipkova, S. Schlecht, F. Ahmaddy, S. Tatavarty, M. Bickel, P. Bilic, M. Rempfler, F. Hofmann, M. D. Anastasi, S.-A. Ahmadi, G. Kaissis, J. Holch, W. Sommer, R. Braren, V. Heinemann, and B. Menze, “Automatic Liver and Tumor Segmentation of CT and MRI Volumes using Cascaded Fully Convolutional Neural Networks,” *arXiv preprint arXiv:1702.05970*, 2 2017. [Online]. Available: http://arxiv.org/abs/1702.05970

[35] T. Kooi, G. Litjens, B. van Ginneken, A. Gubern-Mérida, C. I. Sánchez, R. Mann, A. den Heeten, and N. Karssemeijer, “Large scale deep learning for computer aided detection of mammographic lesions,” *Medical Image Analysis*, vol. 35, pp. 303–312, 1 2017.

[36] M. A. Al-antari, M. A. Al-masni, M. T. Choi, S. M. Han, and T. S. Kim, “A fully integrated computer-aided diagnosis system for digital X-ray mammograms via deep learning detection, segmentation, and classification,” *International Journal of Medical Informatics*, vol. 117, pp. 44–54, 9 2018.

[37] M. T. Islam, M. A. Aowal, A. T. Minhaz, and K. Ashraf, “Abnormality Detection and Localization in Chest X-Rays using Deep Convolutional Neural Networks,” *arXiv preprint arXiv:1705.09850*, 5 2017. [Online]. Available: http://arxiv.org/abs/1705.09850

[38] J. Cornebise, P. A. Keane, and O. Ronneberger, “Clinically applicable deep learning for diagnosis and referral in retinal disease,” *Nature Medicine*, vol. 24, no. 9, pp. 1342–1350, 2018. [Online]. Available: https://doi.org/10.1038/s41591-018-0107-6

[39] M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, and S. Mougiakakou, “Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1207–1216, 5 2016.

[40] M. H. Hesamian, W. Jia, X. He, and P. Kennedy, “Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges,” *Journal of Digital Imaging*, 8 2019.

[41] S. Guendel, F. C. Ghesu, S. Grbic, E. Gibson, B. Georgescu, A. Maier, and D. Comaniciu, “Multi-task Learning for Chest X-ray Abnormality Classification on Noisy Labels,” *arXiv preprint arXiv:1905.06362*, 5 2019. [Online]. Available: http://arxiv.org/abs/1905.06362

[42] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[43] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[44] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition,” in *International conference on machine learning*, 2014, pp. 647–655.

[45] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” *CoRR*, vol. abs/1611.05431, 2016. [Online]. Available: http://arxiv.org/abs/1611.05431

[46] P. Perera and V. M. Patel, “Learning deep features for one-class classification,” *IEEE Transactions on Image Processing*, 2019.

[47] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[48] Z. Hussain, F. Gimenez, D. Yi, and D. Rubin, “Differential Data Augmentation Techniques for Medical Imaging Classification Tasks,” *AMIA ... Annual Symposium proceedings. AMIA Symposium*, vol. 2017, pp. 979–984, 2017.

[49] W. G. Hatcher and W. Yu, “A survey of deep learning: Platforms, applications and emerging research trends,” *IEEE Access*, vol. 6, pp. 24 411–24 432, 2018.

[50] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 8 2017. [Online]. Available: http://arxiv.org/abs/1708.02002

[51] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, “Tversky loss function for image segmentation using 3d fully convolutional deep networks,” in *International Workshop on Machine Learning in Medical Imaging*. Springer, 2017, pp. 379–387.

[52] M. Berman, A. Rannen Triki, and M. B. Blaschko, “The lovász-softmax loss: a tractable surrogate for the optimization of the intersection-over-union measure in neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4413–4421.

[53] F. van Beers, A. Lindström, E. Okafor, and M. Wiering, “Deep neural networks with intersection over union loss for binary image segmentation,” in *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, 02 2019.

[54] F. Milletari, N. Navab, and S. Ahmadi, “V-net: Fully convolutional neural networks for volumetric medical image segmentation,” in *2016 Fourth International Conference on 3D Vision (3DV)*, Oct 2016, pp. 565–571.

[55] S. Park, “F1 score in PyTorch.” [Online]. Available: https://gist.github.com/SuperShinyEyes/dcc68a08ff8b615442e3bc6a9b55a354

[56] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, “The marginal value of adaptive gradient methods in machine learning,” in *Advances in Neural Information Processing Systems*, 2017, pp. 4148–4158.

[57] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, “On the variance of the adaptive learning rate and beyond,” *arXiv preprint arXiv:1908.03265*, 2019.

[58] L. N. Smith, “A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay,” *arXiv preprint arXiv:1803.09820*, 2018.

[59] R. Cadene, “Cadene/pretrained-models.pytorch: Pretrained ConvNets for pytorch: NASNet, ResNeXt, ResNet, InceptionV4, InceptionResnetV2, Xception, DPN, etc.” [Online]. Available: https://github.com/Cadene/pretrained-models.pytorch

[60] “Metrics to Evaluate your Semantic Segmentation Model.” [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

[61] willem (https://stats.stackexchange.com/users/159052/willem), “F1/dice-score vs iou,” Cross Validated, uRL:https://stats.stackexchange.com/q/276144 (version: 2017-11-13). [Online]. Available: https://stats.stackexchange.com/q/276144

[62] N. Moshkov, B. Mathe, A. Kertesz-Farkas, R. Hollandi, and P. Horvath, “Test-time augmentation for deep learning-based cell segmentation on microscopy images,” *Scientific Reports*, vol. 10, no. 1, pp. 1–7, 12 2020.