Detection of Motion Artifacts in Thoracic CT Scans

Beri, Puneet S.

Award date:
2020

Awarding institution:
Université Côte d'Azur
Polytech Nice Sophia

Polytech Nice Sophia / Université Côte d'Azur

Detection of Motion Artifacts in Thoracic CT Scans

Master Thesis

PUNEET BERI

Carried out at:

Thirona B.V.

Supervisors:
Jean-Paul Charbonnier, Thirona B.V.
Jean Martinet, Université Côte D'Azur

Final Report

Nijmegen, September 2020

Abstract

The analysis of a lung CT scan can be a complicated task due to the presence of certain image artifacts such as cardiac motion, respiratory motion, beam hardening artefacts, and so on. In this project, we have built a deep learning based model for the detection of these motion artifacts in the image. Using biomedical image segmentation models we have trained the model on lung CT scans from the LIDC dataset. The developed model is able to identify the regions in the scan which are affected by motion by segmenting the image. Further it is also able to separate normal (or easy to analyze) CT scans from CT scans that may provide incorrect quantitative analysis, even when the examples of image artifacts or low quality scans are scarce. In addition, the model is able to evaluate a quality score for the scan based on the amount of artifacts detected which could hamper its authenticity for the further diagnosis of disease or disease progression. We used two main approaches during the experimentation process - 2D slice based approaches and 2D patch based approaches of which the patch based approaches yielded the final model. The final model gave an AUC of 0.814 in the ROC analysis of the evaluation study conducted. Discussions on the approaches and findings of the final model are provided and future directions are proposed.

Acknowledgements

First and foremost, I would like to thank my project supervisors at Thirona, Jean-Paul Charbonnier and Rūdolfs Latišenko for giving me this opportunity and guiding me throughout the course of this project. Your constant and insightful feedback about the strategy, the direction of the project, as well as the technical guidance regarding the implementation have shaped up this project to what it is today. I would also like to acknowledge my colleagues at Thirona, Iva Pertot, Gerwin Salentijn, and the analysts for their wonderful collaboration in providing non-stop support with the data requirements as well as evaluations.

I would also like to express my gratitude towards my university supervisor Jean Martinet for providing additional guidance as well as reviewing and offering feedback to the thesis. I also want to thank my university program coordinator Françoise Baude for providing valuable support and guidance whenever needed.

Last but not the least, I would like to thank my family back home in India for always being there for me. I would particularly like to thank my dad, Dr. Shirish Beri, for guiding me with the fundamentals of the medical domain as well as providing original material for writing the thesis. I would like to also thank my mom, Smita Beri, and my brother, Anupam Beri, for providing wise counsel and a sympathetic ear along with strong motivation. And finally, I would like to thank my dog Rocket for his virtual companionship and happy distractions to rest my mind outside of the project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

1.1 Introduction

Analysis of medical images can be a complicated task due to the presence of artifacts which impede further diagnosis. An artifact[1] can be defined as something that is visible in an image but it is artificial, and is often detrimental to diagnosis. Artifacts are commonly encountered in Computed Tomography (CT) and may obscure or simulate pathology. There are many different types of CT artifacts, including noise, beam hardening, scatter, pseudo-enhancement, motion, cone beam, helical, ring, and metal artifacts[2]. They can occur due to multiple types of reasons which could be related to faults in the image acquisition machine or on the side of the patient.

There are several probable phases in the pipeline of the medical imaging machine where a potential flaw could cause the creation of an unwanted artifact in the final image. For example, a ring artifact is caused by a miscalibrated or defective detector element which results in rings centered on the center of rotation. Poisson noise occurs due to the statistical error of low photon counts, and results in random thin bright and dark streaks that appear preferentially along the direction of greatest attenuation. Patient based artifacts could occur due to voluntary or involuntary actions of the patient. For example, constant heartbeats could generate blurred or ghost lines along the periphery of the heart in the final image. Few examples of artifacts occuring in CT images are shown in Figure 1.1.

Figure 1.1: Examples of CT image artifacts: (a) Ring Artifact (b) Poisson Noise (c) Beam Hardening and Scatter Artifacts (Source: CT artifacts: Causes and reduction techniques)

Artificial intelligence (AI) is a branch of computer science that encompasses machine learning, representation learning, and deep learning. More recently due to the optimization

of algorithms, improved computational hardware, and access to large amount of imaging data, deep learning is constantly demonstrating a technical superiority over classical machine learning paradigms. Deep learning is a class of machine learning that uses artificial neural networks (ANNs) which structurally emulate the human brain. Some of the popular choices of artificial neural networks are Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). AI can be used for a wide range of tasks to solve problems within the domain of medical imaging. Using computer vision techniques we can perform tasks including detection and segmentation of anatomical structures and the detection of lesions, such as hemorrhage, stroke, lacunes, microbleeds, metastases, aneurysms, primary brain tumors, and white matter hyperintensities[3]. A growing number of clinical applications based on deep learning using computer vision to detect and segment image artifacts as well as diseases are being implemented constantly in the medical world.

For the scope of this project, we focus on one particular kind of artifacts i.e. motion artifacts within the domain of lung CT scans. Motion artifacts can occur due to a variety of causes which are further described in Section 1.3.3. The main goal of this project was to build a deep learning pipeline using biomedical image segmentation which is able to highlight the presence of motion artifacts by segmenting the regions of a lung CT scan. The developed model is able to distinguish between regions of motion versus similar regions in the scan not containing motion. During the course of this project, we performed a series of experiments focused on training the model in different ways such that it learnt the properties of motion artifacts as accurately as possible. Using the final trained model, we also performed a series of evaluations to judge the performance of the model. The final model gave an AUC of 0.815 in the ROC analysis, more on that in chapter 4. Let's now take a look at the similar work and already existing research related to the project.

1.2 Related Work

Let's take a look at the already existing research related to "Motion Artifacts Detection in CT Scans". A lot of research involves the techniques of performing motion detection at a lower sinogram level as well as the hardware level. As these are irrelevant to the context of the project, we are not going to discuss them, and instead focus on the research in the artificial intelligence domain. Lorch et al[4] in their paper illustrate the use and evaluation of a decision forests based method for the detection of synthetic motion artifacts which they've introduced in the reconstructed slices. Another paper by Kustner et al[5] elaborates about an automated reference-free method for specifically MRI scans. MRI scans have a slightly higher level of detail and information as compared to CT scans but the techniques used are insightful. A report by Tamada et al[6] explores motion artifact reduction using the U-Net network. U-Nets are convolutional networks for biomedical image segmentation[7], and have been chosen as the architectural framework for constructing the deep learning models used for this project. U-Nets are explored further in the next chapter.

The paper by Oksuz et al[8] directly involves cardiac motion artefacts. This paper cites the papers by Lorch et al[4] as well as Kustner et al[5] and proves that the techniques described there perform better than the ones cited. The authors first perform a k-space data augmentation to the UKBB data (because of imbalances), and then apply and compare the performance of two deep learning techniques: 1) 3D spatio-temporal CNNs (3D-CNN) and 2) Long-term Recurrent CNN (LRCN). In addition, they also use a model-training technique

known as curriculum learning which involves training the deep learning model in a specific order of easier samples first and then the difficult ones progressively. Based on their experiments, they've concluded that LRCN gives a better performance than 3D-CNN.

One of the main challenges for the image segmentation task at hand was class imbalance. H. Small et al[9] approach the problem of training CNNs with unbalanced data through alternative sampling strategies intended to increase the accuracy of the learned model and neutralize misclassifications, and examine their efficacy in comparison to random sampling. We used similar sampling strategies which are more tailored towards our problem for training our models. Another big challenge for our motion detection pipeline has been overestimation i.e. presence of a high density of false-positives. Park et al[10] propose a semi-supervised learning method where pseudo-negative labels from unlabeled data are used to further refine the performance of a pulmonary nodule detection network in chest radiographs. Similarly we also used a false-positive mining strategy using a smart technique of false-positive patches in the final stages of the project.

1.3 Background

Now that we've been introduced to the project, we will get acquainted with the necessary fundamentals in the subsequent sections. Starting with the pulmonary anatomy in Section 1.3.1, we take a look at the higher level structure of the respiratory system where we concentrate more on the focal point of this thesis, the lungs. We then dive further into the topic of Computed Tomography (CT) Scans in Section 1.3.2 and explain its fundamentals including the process of image acquisition, what are Hounsfield units, and what are the different views through which a CT image can be observed. And finally in Section 1.3.3, we shall examine motion artifacts in detail and explore their causes as effects on a CT scan.

1.3.1 Pulmonary Anatomy

The human respiratory system consists of various structures starting lying between the nose and the lungs. These structures are responsible for providing the supply of oxygen which is essential for each and every cell of the body. We breathe air through the nose which reaches the lungs through a rigid conducting zone. Inside the lungs, there is an exchange of gases between blood and alveoli. Oxygen is taken in by the blood and supplied to the whole body by the cardiovascular system (circulatory system). Air travels from the nose to the pharynx, then to the larynx (voice box), further on to the trachea, and then finally diverges into two bronchi.

The thorax or chest has a cavity called the thoracic cavity which contains two lungs, the right and left lungs. In between the two lungs there lies a space called as mediastinum. The mediastinum mainly contains the heart, big blood vessels, lymphatics and the esophagus. For the extent of this project, we explore only the anatomy of the lungs along with their structure and functions in brief.

A major organ of the respiratory system, each lung houses structures of both the conducting and respiratory zones. The main function of the lungs is to perform the exchange of oxygen and carbon dioxide with air from the atmosphere. The right lung is larger and shorter than the left lung. It is divided by 2 oblique and horizontal fissures into 3 lobes - upper, middle and lower lobes. The left lung is divided by one oblique fissure into 2 lobes - upper and lower lobes. There is no horizontal fissure in the left lung. It also has a cardiac

notch at lower part of its anterior border. The right lung is responsible for 55% of the lung activity while the left lung is responsible for 45%. The structure of the lungs is depicted in Figure 1.2 from the lateral view.

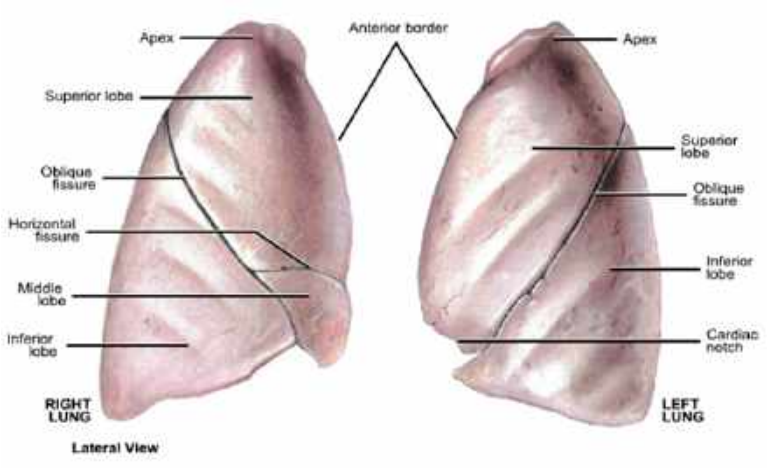

Figure 1.2: Lung Anatomy (Source: Dr. Shirish Beri)

Bronchopulmonary segments are the anatomical, functional, and surgical units of the lungs. Each lobar (secondary) bronchus gives off segmental (tertiary) bronchi which supplies each bronchopulmonary segment. A bronchopulmonary segment is a subdivision of a lung lobe. It is pyramidal in shape, its apex lies toward the root, while its base lies on the lung surface. The right and left lungs have 10 bronchopulmonary segments each. Each segment is surrounded by connective tissue septa and has a segmental bronchus, a segmental artery, lymph vessels, and autonomic nerves. The segmental vein lies in the inter-segmental C.T. septa between the segments. A diseased segment can be removed surgically, since it is a structural unit.

1.3.2 Computed Tomography Scans

A computed tomography (CT or CAT) scan allows doctors to see inside your body. It uses a combination of X-rays and a computer to create pictures of your organs, bones, and other tissues. It shows more detail than a regular X-ray. CT scanners use a narrow X-ray beam that circles around one part of your body. This provides a series of images from many different angles. A computer uses this information to create a cross-sectional picture. Like one piece in a loaf of bread, this two-dimensional (2D) scan shows a "slice" of the inside of your body. This process is repeated to produce a number of slices. The computer stacks these slices one on top of the other to create a detailed image of your organs, bones, or blood vessels. For the scope of our project, we focus specifically on lung CT scans.

Low dose chest CT is routinely used for evaluation of acquired and congenital lung abnormalities, such as pneumonia, interstitial lung disease or tumor evaluation. The radiologist decides the proper settings to be used for the scan depending on the patient's medical problems and what information is needed from the CT scan. The reconstructed image from the scanner is 3-dimensional in nature. Each voxel in this image has a different intensity level based on the object present at that position. These intensities are recorded in the form of

Hounsfield Units (HU) which make up the grayscale in medical CT imaging. It is a scale from black to white of 4096 values (12 bit) and ranges from -1024 HU to 3071 HU (zero is also a value). It is defined by the following:

-1024 HU is black and represents air (in the lungs). 0 HU represents water (since we consist mostly out of water, there is a large peak here). 3071 HU is white and represents the densest tissue in a human body, tooth enamel. All other tissues are somewhere within this scale; fat is around -100 HU, muscle around 100 HU and bone spans from 200 HU (trabecular/spongeous bone) to about 2000 HU (cortical bone). (Source: materialise.com)

Table 1.1 gives some of the useful HU values relevant to chest CT imaging.

Table 1.1: Useful Hounsfield Unit Scales (Source: Wikipedia)

Substance	**HU**
Air	-100
Lung	-500
Fat	-100 to -50
Water	30
Blood	+30 to +70
Muscle	+10 to +40
Liver	+40 to +60
Bone	+700 (cancellous bone) to +3000 (cortical bone)

The reconstructed image from the scanner can be viewed from three perspectives as shown in Figure 1.3. The axial plane produces slices from the bottom-up perspective. Looking at a slice in the axial plane is like looking at the lungs from the bottom of the body. The coronal plane gives you slices from the front to the back and the saggital plane gives you a sideways perspective. Looking at a single scan from three different perspectives gives us different kinds of insight into the task of looking for the elements that we require. Figure 1.4 shows the axial, coronal and saggital views of a CT scan.

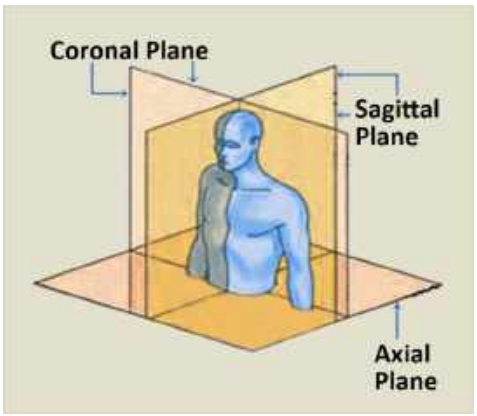

Figure 1.3: Image Reconstruction Planes (Source: ipfradiologyrounds.com)

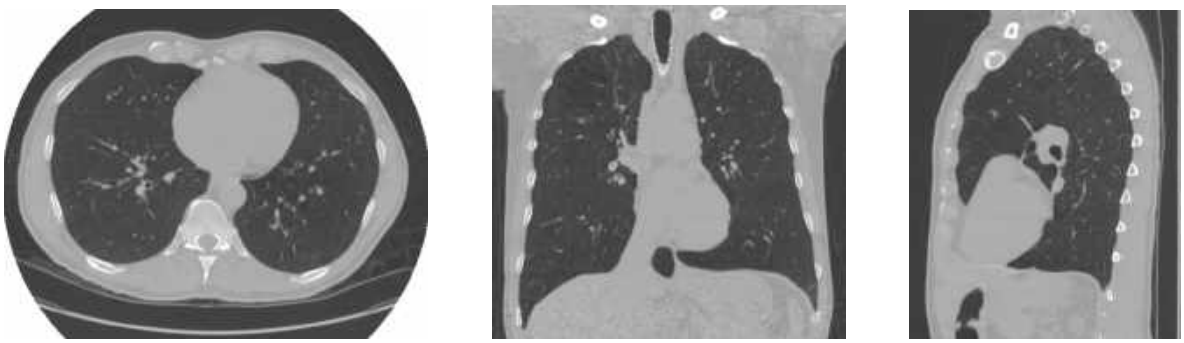

Figure 1.4: Axial, Coronal, Saggital Views of a Chest CT Scan

1.3.3 Motion Artifacts

Motion artifacts are patient-based artifacts that occur with voluntary or involuntary patient movement during image acquisition. Steps can be taken to prevent voluntary motion, but some involuntary motion may be unavoidable during body scanning which produces these artifacts in the final image. This involuntary motion is usually caused by cardiac or respiratory activity. Which leads to the existence of two main kinds of motion artifacts present in CT scans, cardiac motion and respiratory motion. Cardiac motion is caused by heartbeats during the process of image acquisition. This type of motion is involuntary and hence is unavoidable to eliminate. Most scans have cardiac motion artifacts which are usually spotted by blurring, streaking or shading along the heart borders as shown in Figure 1.5.

Figure 1.5: Cardiac motion artifact indicated by the heartbeat (marked by the red arrows)

Breathing artifacts are caused by respiratory activities. They are usually spotted by blurring of the airways. Most of the times, the image acquisition is done by conducting a breath hold technique hence breathing artifacts don't often occur. In most cases, you can spot some artifacts in the parenchyma of the lungs close to a nearby heartbeat. The cardiac motion depending upon its degree affects the nearby airways to have some blurring or streaking. Figure 1.6 shows an example of blurred airways due to respiratory motion.

Majority of the motion artifacts are often spotted in the left lung due to the presence of the heart. Cardiac motion affects the nearby region in the parenchyma of the left lung. The right lung has far less motion artifacts, most of which are caused due to small heartbeats in the upper lobes.

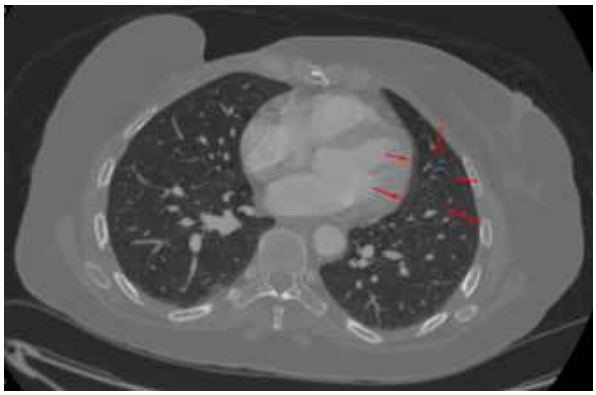

Figure 1.6: Respiratory motion artifact indicated by blurred airways (marked by red arrows)

1.4 Context of this Work

This project was carried out at Thirona B.V. in Nijmegen in the Netherlands. Thirona is a firm which provides automated medical image analysis using artificial intelligence as a service. Thirona has a variety of solutions for image analysis which aids in various tasks like the diagnosis of a disease and disease progression. The work of the motion artifacts detection system falls within the scope of one of the solutions, Chest CT. For developing this project, I was constantly guided by the data scientists in the Chest CT team who supported me throughout different stages during the project. Along with that, I also worked together with the head of the image analysis service in procuring the data as well as the analysts team for generating the annotations and conducting the evaluation studies.

1.5 Structure of the Thesis

In this chapter we have successfully introduced the project as well as covered the necessary fundamentals for understanding the work. In the subsequent chapters, we shall look at the work done into more detail. Chapter 2 sets the stage for the experimental process where we explore the aspects of the model development process. It explores the dataset, data formats, preprocessing, experimental design along with the metrics, and the tools used. Further, chapter 3 describes the experimental process with some of the noteworthy experiments along with their observations. After describing the experiments and models, chapter 4 goes through the evaluation studies conducted with the models and elaborates on their results and findings. And finally, chapter 5 presents a discussion on the findings and concludes the thesis while also talking about the future improvements.

Chapter 2

Description of the Proposed Work

The last chapter gave a brief introduction to the project along with the necessary fundamentals. In this chapter we go a step further by introducing the different aspects of the work. This chapter will focus on detailing the setup of the experimental process by talking about the description of the dataset, data preprocessing procedure, experimental design and metrics, and the model evaluation framework in the next sections.

2.1 Dataset Description

In this section we talk about the different aspects of the dataset used for the project, by first introducing the LIDC-IDRI database. The Lung Image Database Consortium image collection (LIDC-IDRI) consists of diagnostic and lung cancer screening thoracic CT scans with marked-up annotated lesions [11]. LIDC-IDRI is publicly accessible database and hence the dicoms can be easily downloaded from their online portal. Most of the data is available to be used for scientific or educational purposes. The database contains annotations for different lesions but we are not going to be using them for the project. Instead, we have manually gone through the scans, identified regions with motion artifacts, and annotated them by hand. The annotation process has been explained further in subsection 2.1.5. The detailed description of the data[1] is summarized in Table 2.1.

Table 2.1: Detailed Description of the LIDC Database

Collection Statistics	**updated 3/21/2012**
Modalities	CT (Computed Tomography) DX (Digital Radiography) CR (Computed Radiography) SEG (Dicom Segmentations)
Number of Participants	1010
Number of Studies	1308
Number of Series	1398
Number of Images	244,617
Image Size (GB)	125

[1]LIDC-IDRI: https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI

The database contains CT scans of a total 1010 patients performed across 1308 studies and 1398 studies as shown in the table. Each dicom in the set has an anonymized *patient_id*, *study_id*, and a *series_id* associated with it. All dicoms are 3-dimensional in shape with 512x512 slices and a varying dimension along the z-axis.

2.1.1 Sampling

For our project we first carefully selected and sampled 50 lung CT scans from the LIDC database. These scans contain varying levels of motion from subtle to high as well as different kinds of motion including cardiac and respiratory. I started by annotating 12 scans and splitting them into train, validation, and test sets for training the segmentation models. Each set was made sure to be representative of scans containing varying levels of motion artifacts. Going by the results of the models, I progressively specifically selected more scans to annotate and further add to the required sets. In the initial phase, the analysts at Thirona were occupied with COVID-19 work due to which I annotated the scans myself. Gradually, the next sets of scans were annotated by the analysts and by the final stage, all of the scans was analyst annotated.

2.1.2 Data format

The CT scans used in the project are stored in two types of formats.

Dicoms

Dicoms are usually stored as *.dcm* files. A DCM file is an image file saved in the Digital Imaging and Communications in Medicine (DICOM) image format. It stores a medical image, such as a CT scan or ultrasound. DCM files may also include patient information to pair the image with the patient. In our case, the dicoms we use store lung CT scans of the patients.

Chunks

Another way of storing the scans are as chunks. A chunk is a cropped part of the dicom which only contains the lungs. The extra part of the image at the top, bottom, front, back, and sides which does not contain the lung voxels is chopped off to make a chunk. Chunks are stored in the form of *.mhd* data files. MHD data files are related to ITK (Insight Segmentation and Registration Toolkit). The headers of the image store the spacing, direction and origin of the image. For training our models, we start by first reading the chunks of the scans as well as the lung segments, lobe segments, and annotations, explained further in sections 2.1.3, 2.1.4 and 2.1.5.

2.1.3 Lung Segments

For each scan, we have the segmentation masks which highlight both the lungs. The right lung and the left lung have different labels in the lung segment. We used these segments to calculate the percentage of motion voxels inside the lungs. These percentages are used to make quantitative evaluation metrics like the Bland-Altman plot as well as the paired plots for the model as explained in Section 2.4. Figure 2.1 shows the visualization of a lung segment overlaid on top of the original CT slice as seen from the axial view.

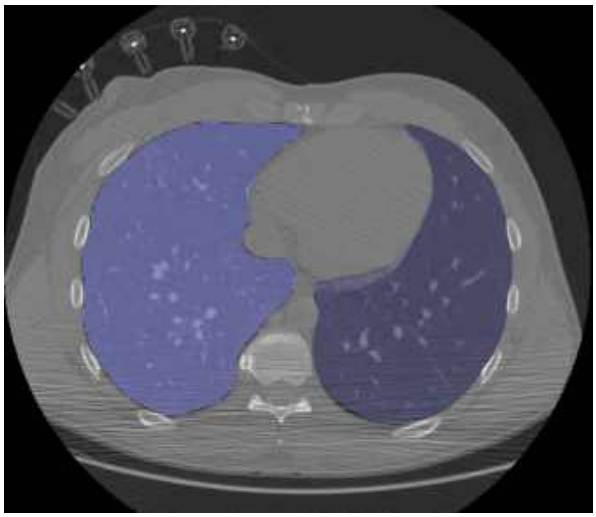

Figure 2.1: Lung Segmentation Map over a CT Slice

2.1.4 Lobe Segments

Similar to the lung segments, we also have lobe segmentation maps for each scan. These segments contain different labels for each of the five lobes inside the lung. Like the lung segment, the lobe segment is also used for generating quantitative evaluation metrics like the paired plot as explained in Section 2.4. Figure 2.2 shows a visualization of a lobe segment.

Figure 2.2: Lobe Segmentation Map over a CT Slice

2.1.5 Annotations

The annotations are simply the segmentation masks which denote the regions of motion artifacts. We have produced these annotations ourselves by individually going through each slice in each scan and highlighting the regions with motion artifacts by hand. For this purpose, we have used a custom pattern annotation tool made by Thirona which helps us easily draw

over the regions of interest in multiple different ways. The interactive drawing tool allows you to draw blobs over two or more slices and interpolates the slices in between. The free hand drawing tool lets you paint over the slice directly as you wish. The annotations serve the purpose of being the ground truth or the true labels. For each voxel in the annotation, the label 0 corresponds to a clean voxel whereas the label 1 corresponds to a motion voxel. Figure 2.3 shows the screenshot of the pattern annotation tool with the region of interest highlighted in dark blue.

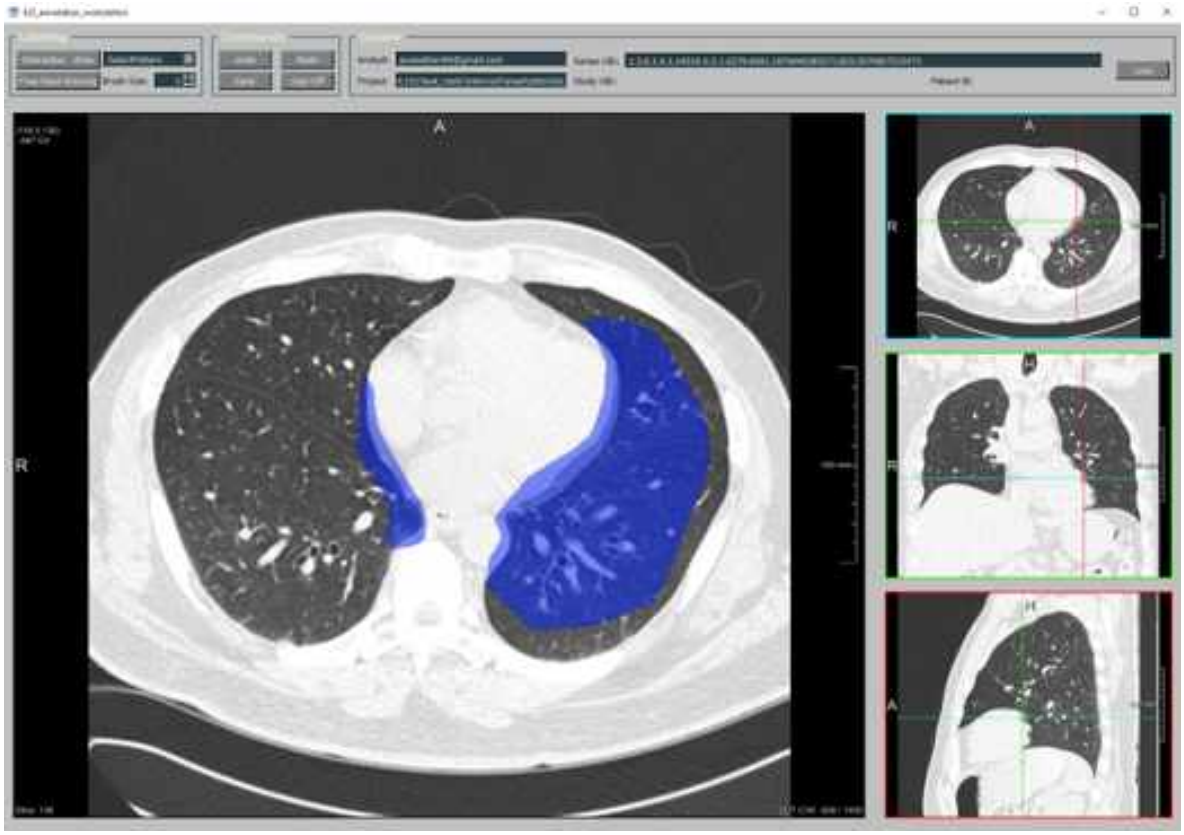

Figure 2.3: Pattern Annotation Tool with a Sample Annotation

2.2 Data Preprocessing

Before we feed our input data to the neural networks, we need to perform some preprocessing steps in order to get it in the right format. We first start by reading the chunks as 3D numpy arrays.

1. Each value in these arrays denotes a Hounsfield units value for that particular voxel position. The HUs are integers in the range from -1024 to 3071 (zero is also a value), and hence these numpy arrays are stored in the datatype of *int16*. Each scan has a different HU range based on the intensities at which it was taken. That is why it is necessary to get all of the input scans in a fixed common range. To do this, we first perform clipping of the intensity values between a fixed range of -1000 to 400.

This means that all voxel intensities below -1000 become -1000 and the ones above 400 become 400.

2. After clipping, we perform a zero-one normalization of the clipped intensities. This way each voxel intensity transforms into a floating point value between 0 and 1.

3. Since each scan has slices of a fixed 512x512 size, the only aspect that varies is its z-dimension making the shape as (512, 512, *z-shape*). And since the images are grayscale, the number of channels is 1.

4. Our deep neural networks are designed to work with 2D images. Hence we reshape the 2D slices as (1, 512, 512, 1) before feeding them to the neural network. When using smaller patches, they are shaped as (1, *patch_size*, *patch_size*, 1).

2.3 Experimental Design

Now that we have explored what the data looks like, let's talk about the experimental setup. We first explore the deep learning models that we use along with the metrics applied for training and evaluating the model. We also talk about the hardware and software tools which were used to build and evaluate our deep learning experiments. And then finally, we take a look at the various evaluation methods and techniques we built to fully evaluate the performance and validity of our models.

2.3.1 Deep Learning Models

The deep learning models are the backbone of the project. They are based on artificial neural networks. The main function of a deep neural network is to learn the specific features of the dataset which we want them to learn and to use that knowledge to make predictions. For the project, I have used the U-Net which is a classic implementation of the encoder-decoder architecture, making it a popular choice for biomedical image segmentation problems.

U-Net

Since the main problem is that of semantic segmentation and not classification, we dive deeper into the concept of "Fully Convolutional Networks". FCNs unlike traditional CNNs do not use fully connected layers and hence retain all spatial information. One of the most prominent examples of an FCN is the U-Net [7]. The U-Net uses what is called as the encoder-decoder architecture. In the encoder part, the convolutional layers coupled with the downsampling layers produce a low-resolution tensor containing all the high level information. The goal is to produce high resolution segmentation maps. Hence the decoder part which follows the encoder contains convolutional layers coupled with up-sampling layers which increase the size of the spatial tensor. These up-sampling layers bring in information from the corresponding layers of the encoder in such a way that they help reconstruct the higher resolution semantic segmentation maps. During model training, the encoder outputs a tensor containing information about the objects, shape, and size. The decoder takes this information and produces the segmentation maps.

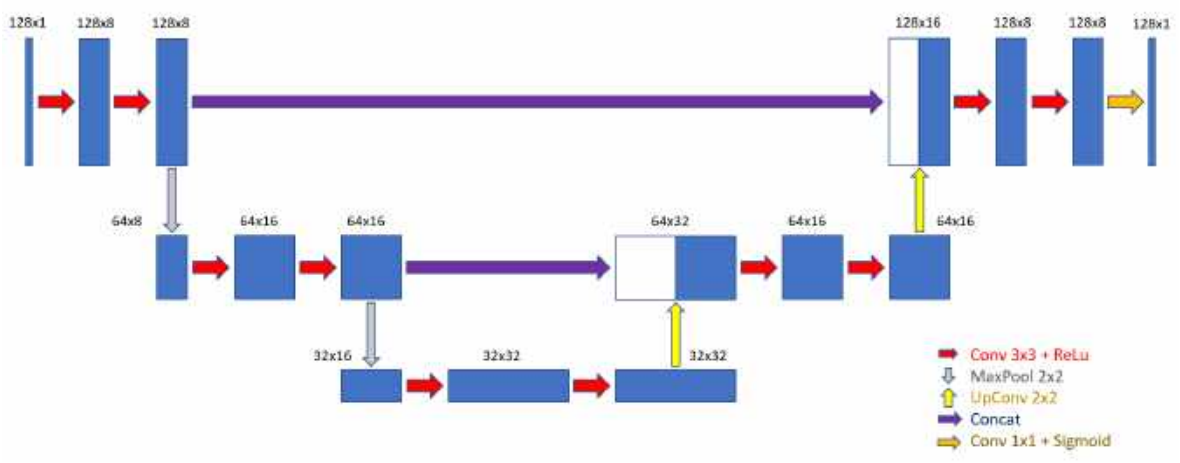

Figure 2.4: Structure of the U-Net Model

The U-Net which we have used in our project is a compact low power U-Net which has 3 encoder levels and 3 decoding levels. Each encoding level has two convolutional layers with a ReLU activation and a max-pooling layer. At the end of the decoding levels, the network has a sigmoid activation function which outputs a probability between 0 and 1 for each voxel in the segmentation map. I have used two different U-Nets for development, the slice based U-Net and the patch-based. Both have the exact same architecture, the only difference being the size of the input and output. The structure of the patch-based U-Net is shown in Figure 2.4 where we can appropriately see the convolutions as well as the up-sampling layers of the network. Figure A.1 in the appendix depicts the U-Net summary which provides even deeper details into each layer of the model.

2.3.2 Metrics for Model Training

Dice Coefficient

The Dice similarity coefficient, also known as the Sørensen–Dice index or simply Dice coefficient, is a statistical tool which measures the similarity between two sets of data. For the image segmentation problem, the dice coefficient is an efficient metric for the U-Net which measures the overlap between the predicted mask and the ground truth. Given two sets X and Y, the equation for the dice coefficient is:

$$DSC = \frac{2|X \cap Y| + \text{smooth}}{|X| + |Y| + \text{smooth}}$$

where the 'smooth' factor is added to avoid a divide-by-zero error.

Dice Loss

The dice loss is a more efficient and accurate loss metric for computer vision problems over the conventional cross entropy loss used for most deep learning problems. The dice loss originates from the dice coefficient and like the former, it calculates the loss for the network based on the overlap between the predicted mask and the ground truth. The dice loss is calculated as the negative of the dice coefficient i.e. $-DSC$.

Tversky Index

The Tversky index [12] is used for the false-positive mining phase of the experiments (explained later). The Tversky Index (TI) is a asymmetric similarity measure that is a generalisation of the dice coefficient and the Jaccard index. In simple terms, it is a modified version of the dice coefficient which allows you to set different weights for false-positives as well as false negatives as per your requirement. Setting equal weights for both yields the dice coefficient. The equation for the Tversky index is given as:

$$TI = \frac{TP + \text{smooth}}{TP + \alpha FN + \beta FP + \text{smooth}}$$

where TP is true-positives, FN is false-negatives, FP is false-positives, and $\beta = 1 - \alpha$. Again, the 'smooth' factor is added to avoid a divide-by-zero error.

Focal Tversky Loss

The Focal Tversky Loss (FTL) is a generalisation of the Tversky loss. The non-linear nature of the loss gives you control over how the loss behaves at different values of the Tversky index obtained.

$$FTL = (1 - TI)^{\gamma}$$

γ is a parameter that controls the non-linearity of the loss. As γ tends to positive ∞ the gradient of the loss tends to ∞ as the Tversky Index (TI) tends to 1. As γ tends to 0, the gradient of the loss tends to 0 as TI tends to 1.

2.3.3 Software and Hardware Used

MeVisLab

MeVisLab is a powerful medical image processing and visualization tool which includes advanced software modules for segmentation, registration, volumetry, as well as quantitative morphological and functional analysis[2]. Using different modules, you can build your own custom network to perform specific image processing or visualization tasks.

In this project, we have used MeVisLab for quite a lot of purposes. First and foremost, the itkImageFileReader module is used to read the scans. It can read files stored as both dicoms and chunks. When connected to a View2D or View3D module, we can visualize the scan in 2D and 3D respectively. Figure 2.5 shows the output of a View2D module visualizing a lung dicom in axial view. This is the primary visual perspective used for the visual analysis of the results and performance evaluation of the model. The OrthoView2D module can also be used which produces the axial, coronal, and saggital views of the same scan as shown in Figure 2.6. The orthogonal views haven't been fully used during the course of the project, but they provide additional insight from a different perspective.

[2]MeVisLab: `https://www.mevislab.de/mevislab`

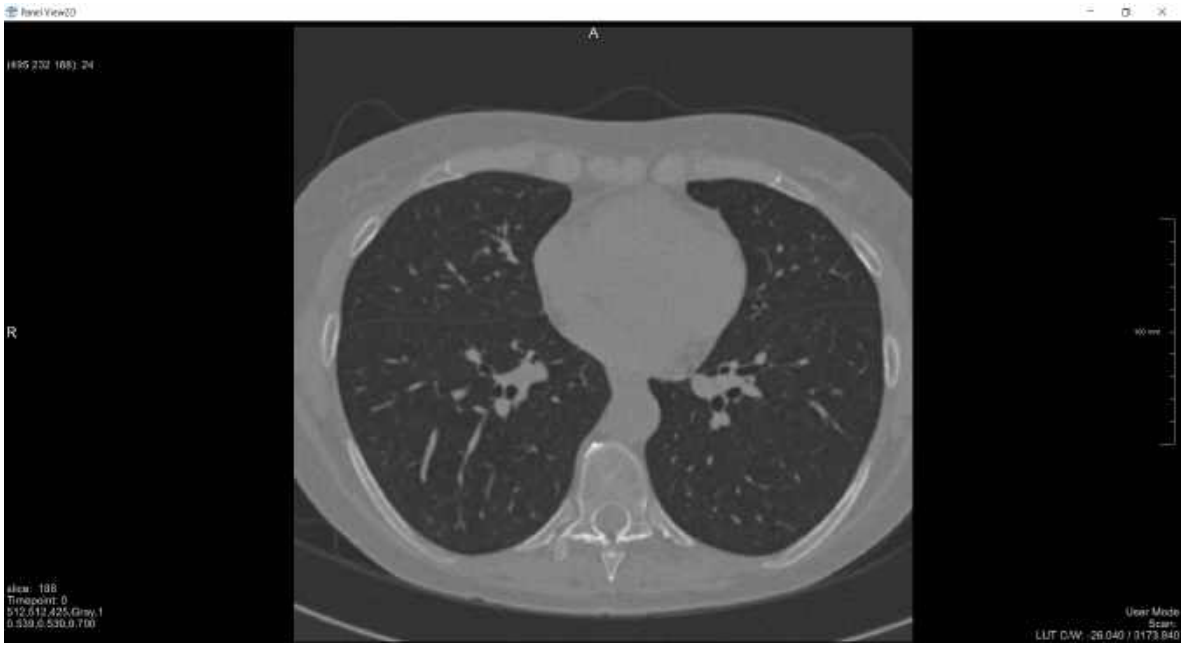

Figure 2.5: 2D Slice Visualization from the Axial View in MeVisLab

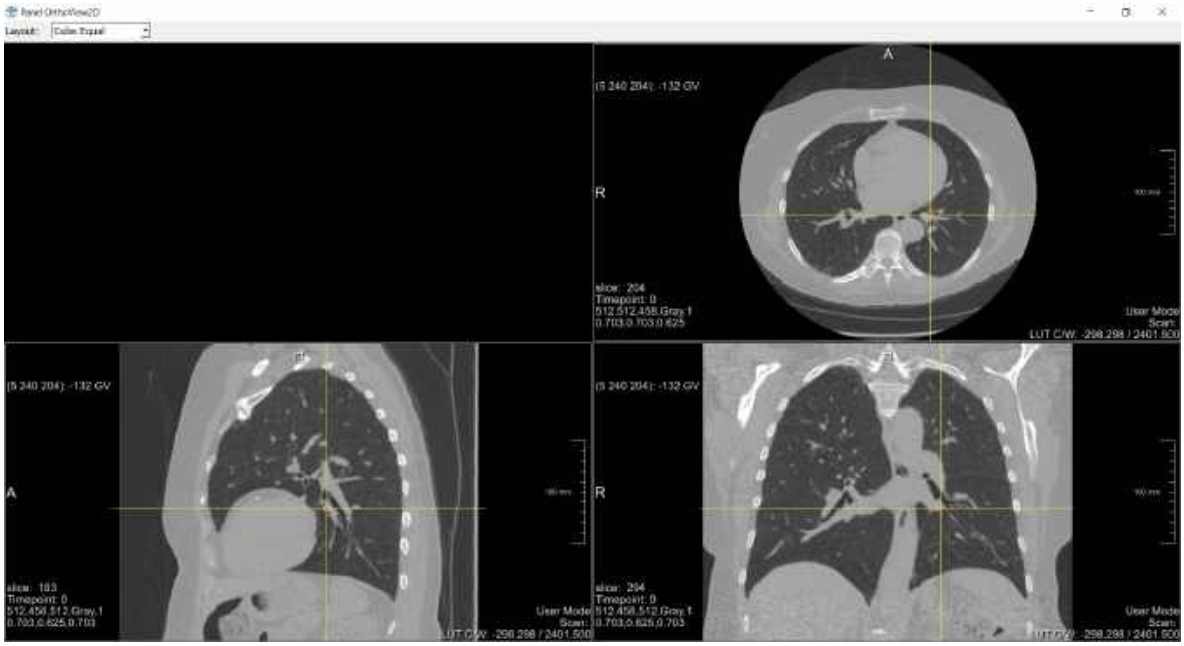

Figure 2.6: 2D Orthogonal Visualization in MeVisLab

A simple visualization network was constructed to inspect the results of each experiment. The visualization consists of the ground truth and the predicted motion segmentations overlaid on the original lung dicom. The MeVisLab network is shown in figure 2.7.

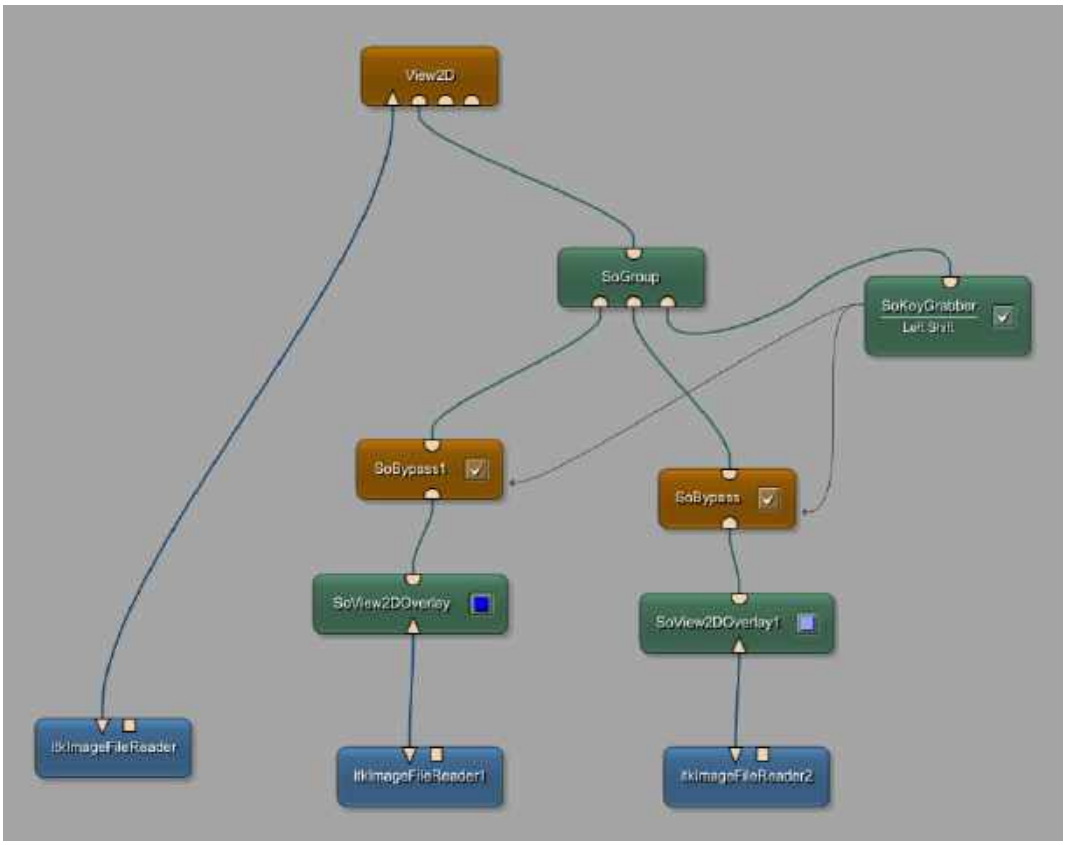

Figure 2.7: Visualization Network in MeVisLab

Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. I used the Jupyter notebook installed in Thirona's NAS to build and train the models using Python3 and its associated libraries.

Keras

Keras is an open-source library written in Python which is used for constructing deep neural networks. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. For the project, I used keras on top of Tensorflow to develop and train the deep learning models.

Hardware

Since each scan is in the order of a few hundred MBs (GBs too in some cases), building and training models has a substantially high hardware requirement. Hence, the project was entirely developed and run on Thirona's servers with high end GPUs.

2.4 Model Evaluation Framework

To evaluate the quality of our results, we use a number of different methods and techniques of evaluation. Every single method gives an insight into the different aspects of the model's performance. While most of our methods are quantitative in nature and guide the development process further, sometimes we find out that the visual results tell a different story. This is why during the course of the project, we have come up with novel and innovative approaches to evaluate our model based on the insights gained from each experiment. The following subsections showcase the arsenal of the methods of evaluation used during the project.

2.4.1 Learning Curves

Model learning curves are the fundamental standard set of graphs used in almost all deep learning projects. These two sets of graphs demonstrate the behaviour of the model during the training/learning phase. One of the graphs plots the training v/s the validation loss over the course of the training epochs. In our case, the value of the loss function is given by the dice loss. The x-axis denotes the epochs while the y-axis denotes the value of the loss function as depicted in Figure 2.8. The second graph plots the dice coefficient, for the training as well as the validation on the y-axis and the epochs on the x-axis.

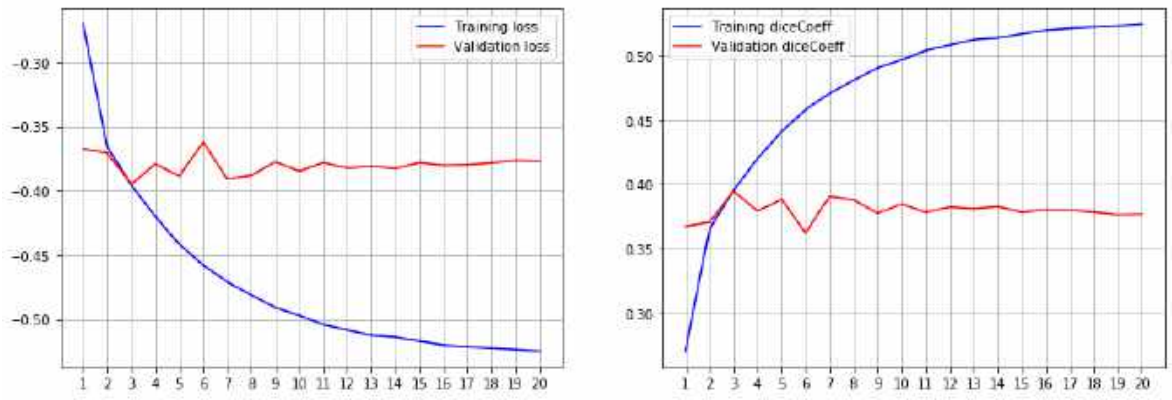

Figure 2.8: Sample Learning Curves

2.4.2 Bland-Altman Plots

The Bland-Altman analysis focuses on the assessment of agreement between two quantitative methods of measurement. The Bland-Altman plots perfectly showcase the quantification of the agreement between the two measures. This is done by studying the mean difference and constructing the limits of agreement. In our case, we have used the Bland-Altman plots to showcase the quantitative agreement between the percentage of motion in the lungs in the ground truth v/s that in the model predictions. The ultimate goal of our model is to get the predicted motion percentages as close as possible to the ground truth. We measure the percentage as the number of motion voxels divided by the total number of voxels inside the lung multiplied by hundred. The calculation of motion percentages is done on different levels like per scan or per 2D slice of a scan. The percentages can also be calculated even one level further, inside the right and left lungs per scan or per slice depending on how deep the analysis

needs to go. Figure 2.9 shows a sample Bland-Altman plot per case and per slice. The x-axis of the plot depicts the average between the two quantitative measurements whereas the y-axis depicts the mean differences between them. The region highlighted by the red dotted lines represents the region within the limits of agreement. The goal of the model is to get all the points to fall close to the mean difference, and to get the mean difference as close to zero as possible. This also makes the limits of agreement shrink to as small as possible around the mean difference.

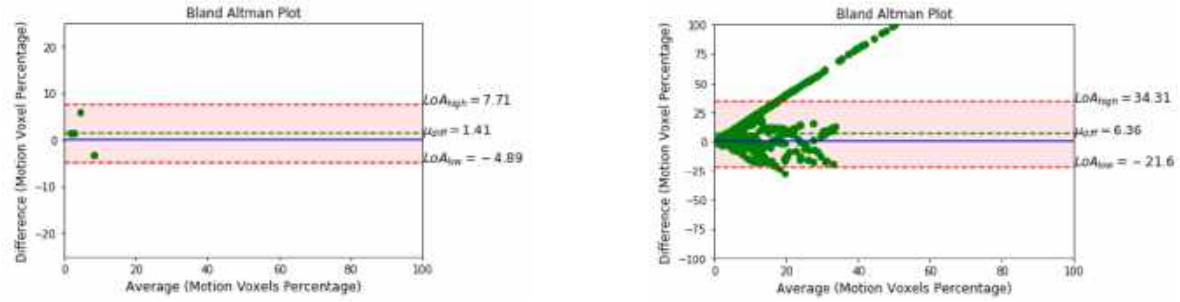

Figure 2.9: Bland-Altman Plots: (a) Per Case (b) Per Slice

2.4.3 Paired Box-Plots

These plots are another way of showcasing the quantitative agreement between the original motion percentages v/s the predicted. These plots are plotted specifically per lobe. Each graph contains two box-plots, one for the motion percentages in the ground truth and the other for predictions. As represented in Figure 2.10, each box plot inside a single sub plot represents the distribution of percentages of motion. The points are also highlighted by the scatter in each box. The corresponding points representing the motion percentages in test scans are connected by a line to show the quantitative difference. The goal of the model is to get these lines to be as horizontal as possible implying the least percentage wise difference in the lobes. This plot helps pinpoint the regions where the model goes close to the ground truth as well as the regions where the model is the most likely to commit mistakes. Using this visualization helped us identify the next direction to take for each consecutive experiment.

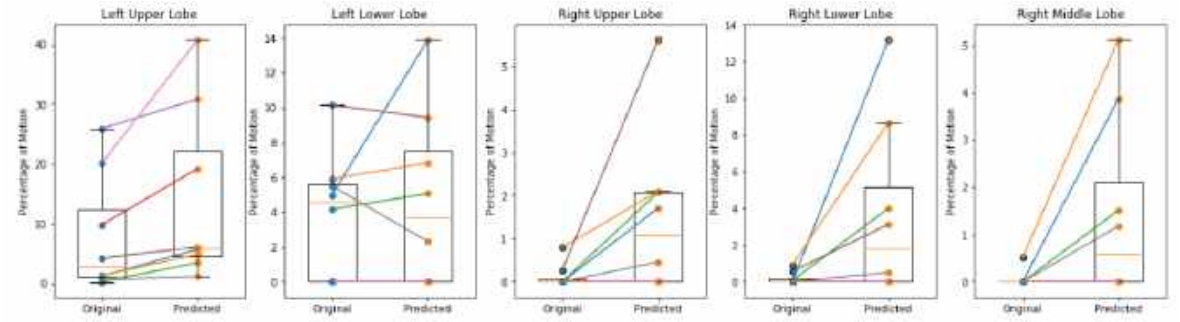

Figure 2.10: A Sample Paired Box Plot

2.4.4 Visual Inspection

Although the quantitative evaluation metrics and graphs help identify a lot of areas for improvement, they don't tell the full story about the model's performance. Even if they might seem tedious to inspect, looking at the visual results gives us the full picture. You can see with your own eyes how the model performs on real test cases which it hasn't seen before. Visual results not only helped identify areas of improvement, but also gave insights to come up with newer quantitative measures and plots. Figure 2.11 depicts a sample visual result. It shows a slice of a CT scan from the axial view. This slice contains a cardiac motion artifact from a heart-beat. The highlighted blob in dark blue represents the original motion artifact annotation while the light blue blob represents the model prediction.

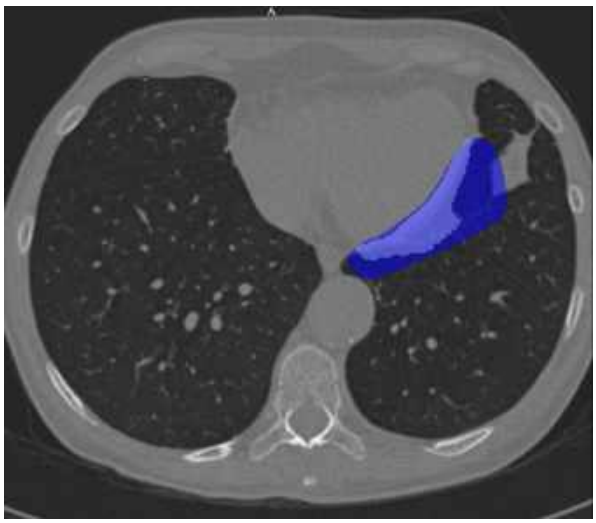

Figure 2.11: Sample Visual Results

2.5 Summary

To summarize this chapter, first section 2.1 introduced the dataset used for the model training and development. Along with that, we also learnt more about the data sampling, format, and the annotation process used. Further, section 2.2 explored the preprocessing steps applied on the data before training. Then, section 2.3 went into detail about the different aspects of the experimental design including the deep learning model architecture, metrics, and the tools used during experimentation. And finally, section 2.4 explained the different metrics, plots, and graphs used for evaluating the resulting model after each experiment. Next, in chapter 3, we see the detailed design and approaches used for the experiments.

Chapter 3

Deep Learning Experiments

In the previous chapters, we got to know the necessary fundamentals as well as the different aspects of the experimental process. This chapter directly deals with the approaches and strategies employed for the deep learning experiments including the setup, execution, and evaluation. I used two fundamental approaches for the experimentation process. In the initial phase, I first started with 2D slice-based U-Nets and training strategies. After running a few experiments and studying their findings, we decided to switch to the 2D patch-based models and approaches due to reasons explained further. More than 20 main experiments were run during the development process of which 16 were documented on Thirona's Phabricator platform. The following sections highlight only a small set of the experiments which were deemed noteworthy, with their strategies and findings. Section 3.1 includes the experiments with the 2D slice-based approaches while 3.2 highlights the experiments with the 2D patch-based approaches which ultimately yielded the final model. The detailed plots and visual results can be found in the appendix.

3.1 Experiments with 2D Slices

The following subsections describe the noteworthy experiments from the slice based approach. All experiments under this section used 2D slices of 512x512 voxels as input. All models were trained using the dice coefficient and dice loss with a smooth factor of 1. For further reference, the reader is advised to take note of the following terms. A *positive slice* is a slice with at least one motion voxel in the ground truth, whereas a *clean slice* implies the absence of motion voxels in the ground truth of that particular slice. All experiments used a training, validation, and a test set. For training the models, the Adam optimizer with a decaying learning rate was used. Before training, each scan went through the preprocessing procedure as described in section 2.2 yielding a *clipped and normalized scan*.

3.1.1 Model 1.01: U-Net with Positive Slices

This is the first successful experiment which yielded a sensible output. The dataset consisted of 8 training scans, 3 validation scans, and 3 test scans. The class distribution of positive slices vs clean slices was heavily imbalanced ($< 1\%$ voxels contained motion). Hence, I trained the network with positive slices only. Even after eliminating the clean slices, only $\approx 1.42\%$ voxels were positive. The slice wise distribution of the data after eliminating clean slices had

4147 slices in the training set and 1695 in the validation set.

The network used same padding in the convolutions. The model used a batch size of 4 and was trained for 10 epochs. The dataset was entirely loaded in memory before training. For evaluation, we only used the dice score and visual inspections. After training and predicting on the test set, I observed that the predicted output had a lot of false-positives while there was some level of overlap in the positive regions. Since the model trained only with positive slices, the network predicted a lot of false-positives even for slices which did not contain any motion artifacts. The test set evaluation produced a dice coefficient of 0.1233.

Model 1.02 used the same parameters except for valid padding instead of same padding to the convolutions. Based on the learning curves, it was found that valid padding makes the model learn slower which is why it wasn't used for future experiments. For model 1.03, an additional scan was added in the training set and all sets were reshuffled such that they were representative of different severities of motion. The model was trained for 100 epochs and the main finding that the learning and loss curves flatten out after around 20 epochs. This is why all future experiments were trained for 20 epochs. Another finding was that the model produced a lot of false-positives in scans with subtle motion whereas lots of false-negatives in the scans with large motion artifacts.

3.1.2 Model 1.04: U-Net with Custom Training Mini Batches

Instead of training the batches with only positive slices, we slightly modified the strategy for this experiment. The training mini-batches were customized to contain:

- 2 positive slices. These slices were sequentially sampled from the set of positive slices. Each epoch ran as long as the network had been trained with every single positive slice.
- 2 clean slices. These slices were randomly sampled from the set of clean slices. During the run of an epoch, not all clean slices were used to train the network.

An additional scan was added to the training set. In addition to visual inspection, the test set performance evaluation was done with the interpretation of Bland-Altman plots for the first time here. After training and predicting, each of the 3 test scans were inspected individually. The BA plots were made per slice for each of the lungs in all cases. In the BA plots, quite interestingly both the lungs showed a linear scatter on the positive side of the difference-axis. This implied that the model produced false positives in the regions where there is no motion in the ground truth, for many clean slices as well. The BA plots for one of the test cases is shown in figure 3.1 where a linear scatter is clearly visible.

The Bland-Altman plots provided an insightful evaluation metric to examine how the model performs on test data. Training the model with clean slices along with positive slices decreased the false-positive regions in all slices in spite of the existence of a linear scatter in the BA plots. Model 1.04 was also used for the evaluation studies to compare against the final model 2.08, as described in chapter 4.

Model 1.05 used shuffling for the positive slices before every epoch and resulted in not having a significant improvement in performance. Model 1.06 used 3 positive slices and 1 clean slice in each mini-batch which also did not affect the performance significantly. Model 1.07 used 2 positive slices, 1 closest clean slice, and 1 randomly sampled clean slice. The closest clean slice is the slice which has the least distance on the z-axis from one of the positive slices sampled, literally the slice which is *closest*. The purpose of this strategy was to make the

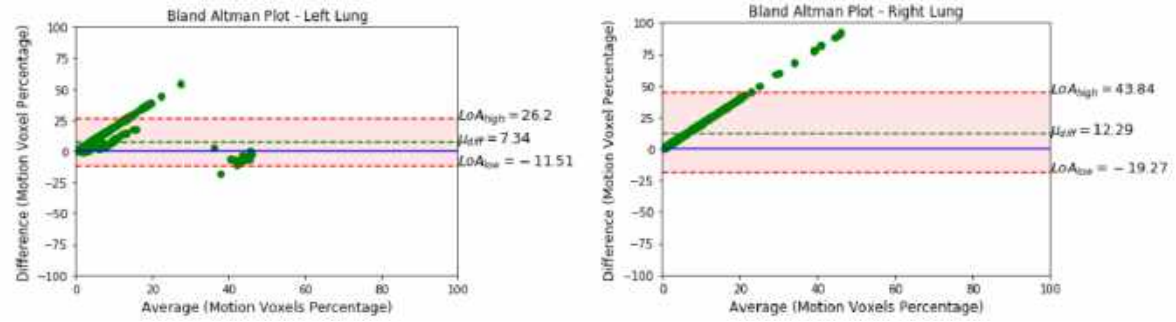

Figure 3.1: Bland-Altman plots per slice (left and right lung)

model learn the difference between motion v/s no motion in two slices which look almost the same. This strategy showed a reduction in false-positives as indicated by the BA plots and the visual results.

3.1.3 Model 1.08: U-Net with Batch Size 8

This model used a batch size of 8 and a modified mini-batch sampling strategy. Each scan was divided into 10 equally spaced regions along the z-axis. Each mini-batch contained:

- 4 positive slices.
- 3 clean slices which fall within the same region of the positive slices. These slices could come from the same scan or other scans. The goal was to make the model learn how to distinguish between a positive slice v/s a clean slice in the same region of the lung. One clean slice was not sampled twice per epoch.
- 1 randomly sampled clean slice.

Additional data was also added which made the *train:validation:test* split as 12:4:4. From the predictions, it was found that false-positive regions inside the parenchyma away from the lung periphery were greatly minimized while the false-negatives in the parenchyma with small airways and arteries, were significant. On the other hand, the true positive regions around the periphery of the heart and in the lower lobes were of great quality, some even better than the ground truth.

One common observation from all the experiments was that while the numerical metrics did show much difference, the visual results quite often told a different story. Another main observation was that the slice-based models tended to predict motion in the most likely locations of the slice for instance, the heart periphery even if there was no motion present. Due to this reason, we decided to switch to the patch based approach which was location independent. The experiments with the patch-based approach are described in the next section.

3.2 Experiments with 2D Patches

This section describes the experiments using 2D patches of size 128x128 as the input to the U-Net. Each experiment implemented a training generator which sampled small patches from

512x512 slices using the centre-voxel sampling strategy. The centre-voxel sampling strategy samples one random voxel from a slice based on the condition, and constructs a patch of 128x128 voxels using it as the centre of the patch. If the sampled voxel is close to the edges, it is re-adjusted accordingly. Figure 3.2 shows an example where a patch is centre-sampled from a positive voxel position in the ground truth. The square patch in green denotes the patch sampled by the generator. By using a patch sampling approach, we artificially generate much more samples of data for the model training than we actually have.

The batch size for each of the experiments was 32. As before, we have used the terms *positive slice*, *closest clean slice*, and *random clean slice*. The patch size of 128 was chosen as suitable based on experiments analysing the performance by varying the patch size. For each experiment, I built a custom smart patch generator which sampled positive or clean patches as per the required balance. A validation set was constructed with the same generator, which effectively consisted of the same balance of patches. This validation set is constructed one-time before training the model and is kept as static for proper validation. For constructing the test segmentations, we use a sliding window generator to predict over sequential patches. For each patch, we only use the central 32x32 segmentation, and proceed to the next patch with a stride of 32. This size of 32 is called the *valid region dimension* and is usually kept much smaller than the patch size to minimize border effects.

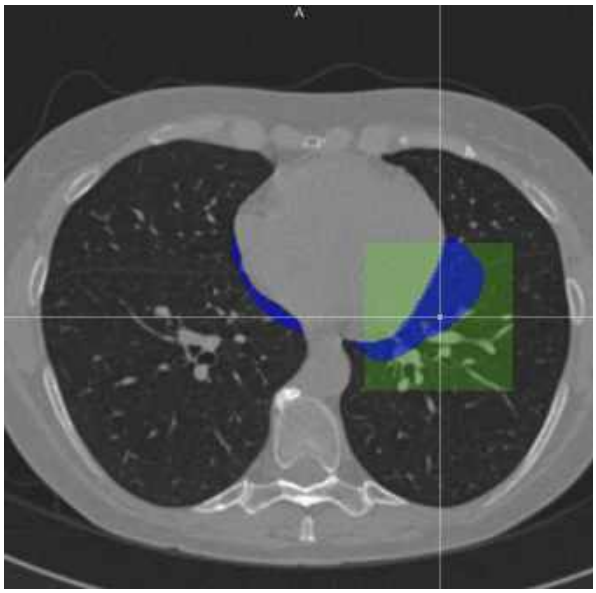

Figure 3.2: Example of Centre Voxel Sampling

3.2.1 Model 2.01: U-Net with 2D Patches

This is the first experiment with the patch-based approach. The steps for patch extraction for each training mini-batch are as follows:

- For each step, the generator selects a positive slice. Before each epoch, the set of positive slices is re-constructed with a repeat factor (3 in this case) which makes sure that each positive slice is sampled 3 times per epoch. This approach pseudo-augments the data.

- The generator additionally chooses the closest clean slice and a random clean slice. Hence the set of selected slices per mini-batch looks like *(positive slice, closest clean slice, random clean slice).*
- 12 patches centre-sampled around positive voxels are chosen from the positive slice.
- 12 patches from the same co-ordinate positions are chosen from the closest clean slice.
- 8 patches are randomly sampled from a random clean slice. This step ensures that the model also sees patches from random regions of the scan.

After training, it was found that the BA plots indicated an expansion in the limits of agreement as compared to 1.08. This was confirmed in the visual inspections from the fact that there were a lot of false positives inside the parenchyma. The main finding was that predicting on patches instead of slices helped the model learn more about identifying artifacts without having the knowledge of the location where they are most likely to occur. A lot of false positives and false negatives were predicted in the parenchyma which needed to addressed by a different strategy. Model 2.02 changed the balance of the mini-batches and used an arbitrary false-positive mining strategy where the trained model predicts on one scan and retrains with the patches where mistakes were committed. While this model showed a decent improvement, the strategy was too arbitrary and was not strong enough to be replicated in a fool-proof way.

3.2.2 Model 2.03: U-Net with Augmentations, False-Positive Mining

Model 2.03 added more scans making the train:val:test split to be 16:7:8. Half of the train mini-batch consisted of 8 positive and 8 closest clean patches. The other half either added augmentations of the first half, based on random 90° rotations and flips, or random patches based on a random binary decision variable. In addition, it added an extra false-positive mining phase. To elaborate further, the trained model first predicted on the training set. False-positive maps were constructed from the predictions and the model was retrained with patches centre-sampled around FP voxels from these slices. Each mini-batch contained 16 FP patches, 8 positive patches, and 8 closest clean patches. To examine whether the false-positive mining strategy indeed had an effect in reducing the FP regions in the scans, we decided to build the paired box-plot visualization per lobe. They showed that the model over-estimated heavily especially in the lobes of the right lung. Taking this into account, I adjusted the balance of FP patches during the training to sample more patches from the right lung. The paired box-plots from the FP mining phase showed a significant decrease in over-estimations as compared to the plots from the pre-training phase. This is demonstrated by figures B.4 and B.5 in the appendix. The FP mining strategy had an effect on minimizing the FP regions but not as much as expected.

In model 2.04, I switched to the Tversky index and focal tversky loss for the FP mining phase. The parameters were $\alpha = 0.3, \beta = 0.7, \gamma = 0.5$. The new metric did in fact have the desired effect and reducing FP regions greatly, with some scope of improvement. Model 2.05 added more scans to make the dataset ratio as 35:7:8 which was the final ratio. In addition, all annotations which I had made myself previously were replaced by annotations from analysts. Due to the heavy changes in data, model 2.05 performed worse with increase in FP as well as FN regions.

3.2.3 Model 2.06: U-Net with Updated FP Maps

For this experiment, the main focus was make the model narrow down further on the areas of mistakes. Hence for the FP mining phase, the model was first trained for 4 epochs, and then made to recompute the FP maps, post which it was again trained for 4 more epochs. Throughout these 8 epochs, the decay in the learning rate was maintained. The paired box-plots showed that the model reduced false-positives while slightly tending towards underestimation implying false-negatives in some cases. This was also confirmed by the BA plots. On inspecting the visual results, a number of observations were found. A considerable number of false-negative regions in some scans had quite subtle occurrences of blurred airways which weren't picked up by the model. Some of the FNs were actually annotated regions resulting from interpolation and which did not contain any motion. Some false-positives actually turned out to be real motion artifacts missed by the annotations which were picked up by the model. We observed many such cases where the model performed better than the annotations. The screenshots of these can be found in the appendix. By recomputing the FP maps every few epochs, the false-positives were actually reduced to a great extent in the test results but also introduced underestimation and false-negatives. Hence it also became essential to also keep track of the false-negatives and ensure a good balance for an "ideal" model.

To account for this, we replaced the 8 random patches with false-negative patches in the mini-batches for the FP mining phase in model 2.07. Along with that the model trained for 10 epochs in FP mining, recomputed the FP maps, and trained for another 10 epochs. The improvement wasn't significant over 2.06.

3.2.4 Model 2.08: U-Net with Corrected Annotations

The TI and FTL values were changed to $\alpha = 0.25, \beta = 0.75, \gamma = 2$. For this experiment all the annotations in the ground truth were corrected in such a way that only the segmentation within the lung was maintained while that falling outside the lung borders was removed. This was done by intersecting the lung segment with the ground truth and using the result to train the model. As shown in figure 3.3, the image on the left shows the original annotation, whereas the image on the right shows the corrected annotation.

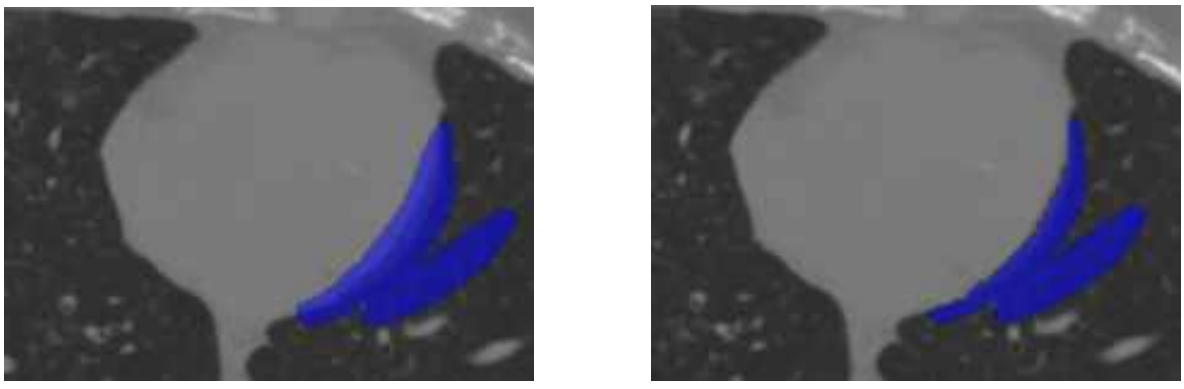

Figure 3.3: Annotation Correction: Intersection with Lung Segment

For the FP mining phase, I first calculated the FP and FN maps and then intersected them with the eroded lung segments. The erosion of the lungs was done with a 7x7x7 kernel.

The purpose of doing this was to make the training generator sample points lying further inside the lung parenchyma, away from the peripheries. In figure 3.4 the first picture shows the original FP map of the pre-trained model. We noticed that a lot of voxels lie close to the lung periphery (and even outside) which implied that the sampled patches were likely to contain a huge proportion of region outside the lung, which wasn't relevant. By removing the regions closer to the lung borders as shown in the second figure, we forced the model to pick centres of patches lying in the inner regions of the lung.

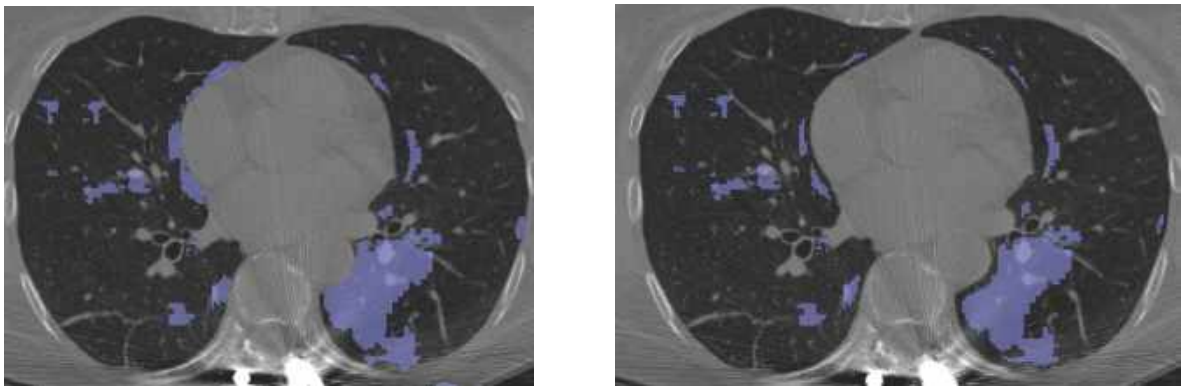

Figure 3.4: Intersection of FP Map with Eroded Lung Segment

The model fit the training data very well which implied that it learnt exactly what we wanted it to learn from the data. Although the model overestimated or underestimated in most scans, it was observed that these so called regions of mistakes were in fact correct predictions implying that the model outperformed the ground truth. Most false-positives around the heart periphery were generated as a result of interpolation in the training data. False-negatives usually were a result of interpolation or contained very subtle motion artifacts which weren't picked up by the model. One of the biggest causes of these false-negatives could be the under-representation of subtle artifacts in the training data. For some cases the model predicted FP regions in the lower lobes of both lungs. These regions were falsely highlighted because they resemble regions with motion artifacts and also because the a lot of the training data contains motion in those regions. Practically it is almost impossible to remove these FPs unless we had highly precise data to train on. Figures B.18 and B.19 show the paired box-plots and the BA plots for the model. Based on the results as well as the time constraints for the internship, model 2.08 was chosen as the final model. The next chapter details the evaluation studies conducted on the final model and compares the results to a baseline model.

Chapter 4

Evaluation

In this chapter, we focus on evaluating the performance of our final model through a couple of evaluation studies. In each study, we compare the performance of a baseline model from one of our earlier experiments to the latest model developed. For the baseline model, I chose experiment 1.04 which focused on having a 50-50 balance in positive v/s clean slices in each training mini-batch. For the final model I chose experiment 2.08. For both the studies, we decided to involve the analysts at Thirona who helped us with the tasks of visual scoring and annotations, which are explained further on. Each study has a different objective. In the higher level evaluation, we take a bird's eye view at the model's performance evaluating its ability to accept or reject a scan for further analysis based on the amount of motion it detects. Further, in the lower level evaluation, we dive deeper into the analysis by looking at the model's predictions in each lobe. During the course of this project, the CT scans from which data was used to train the models were all sampled from the LIDC-IDRI dataset. For the evaluation studies as well, I used additional scans from the LIDC dataset.

In addition to LIDC scans, I also wanted to see how the model would perform in scans with disease. The COVID-19 pandemic has introduced a new rapidly spreadable disease into the world which directly affects a human being's respiratory system. COVID-19 attacks the lungs and causes damage to the lung tissue, which is permanent in some cases. This is clearly reflected in the lung CT scans of the patients. In most COVID-19 cases which are moderate to severe in nature, the scans end up containing heavy motion artifacts. This could be attributed to a variety of factors like for example the patient cannot stop coughing even while taking the scan. I wanted to satisfy my curiosity to see how the model performs in detecting motion artifacts in these scans. Which is why I added 10 scans from COVID-19 patients sourced from Kaggle, to the evaluation studies.

4.1 Higher Level Evaluation

The objective of the higher level evaluation is to study the model's capability in categorizing a scan based on the amount of motion detected. In addition, we want to quantify the model's ability in accepting or rejecting a scan for further diagnosis for disease or disease progression. To perform this evaluation, we needed a large set of scans which would be scored visually into 3 different categories - **MILD, MODERATE, SEVERE**, depending upon the amount and severity of the motion artifacts contained. I sampled a set of 95 scans from the LIDC set based on an arbitrary scan selection strategy which is described in Section 4.1.1. To achieve

the visual scoring task, we involved 2 analysts from Thirona and I designed a specific protocol for them which is explained further in Section 4.1.2.

4.1.1 Scan Selection

For this evaluation, I selected scans from the LIDC and Kaggle COVID sets in a pseudo-random fashion in such a way that the probability of having a uniform distribution (in terms of motion density) is high. The strategy followed for selecting the scans is described below:

- The model was used to predict on all the LIDC scans, except the ones which were used in the train, validation, and test sets.
- After prediction, I calculated the percentage of positive voxels in each scan. Calculating motion percentage based on total voxels in the scan instead of total voxels in the lung segment increases the possibility of having a varied distribution of scans.
- After calculating the percentages, I looked at the distribution of percentages of motion found. Based on the distribution and visual examination of a few scans, I set arbitrary thresholds on the 3 categories.
- For each threshold interval, I randomly selected 30 scans per category such that they added up to 90 scans. Adding the 10 COVID cases I had a total of 100. Later, 5 scans from the LIDC set were removed because they were in the prone position instead of supine. Hence the total number of scans used for the higher level evaluation was 95.
- The assumption is that there are very few scans from the selected LIDC set which actually are severe in nature. Also, since it is observed that most COVID-19 scans contain severe amounts of motion, we assume that the COVID scans will make up the distribution of the severe scans in the set.

4.1.2 Protocol

Once the scan selection was done, the entire set of 95 scans was handed over to the analysts for visual scoring. The analysts had the task of going through it each scan, and assigning a category to it from either of **MILD, MODERATE** or **SEVERE** based on the specific criteria in the protocol. Each analyst scored the scans independently so that they would not be influenced by each other's scores. The criteria for scoring a scan into one of the three categories are described below:

- If the scan contains only subtle artifacts depicting cardiac motion (heartbeats), with very little to no parts of the parenchyma containing motion, it is be classified as MILD. Mild scans mostly have motion only along the heart periphery and the upper lobes in a few slices, that too quite subtle as shown in Figure 4.1(a).
- When a significant portion of the parenchyma shows motion i.e. lots of visible blurred airways as shown in Figure 4.1(b), the scan is scored as MODERATE. A moderately affected scan shows regions with very obvious artifacts in the upper as well as lower lobes in some cases. Although a significant portion of the lung is affected, it isn't affected in its entirety in a moderate scan.

- In the cases where it is observed that both the lungs in their entirety are affected by heavily blurred airways and heartbeats in a significant number of slices, the scan is scored as SEVERE. The difference between moderate and severe scans would be the proportion of the lungs affected by motion as well as the severity of the artifacts spotted. Usually there is more motion in the left lung than the right due to the presence of the heart. Hence if the right lung also has huge regions affected by motion, it becomes severe. An example is shown in Figure 4.1(c).

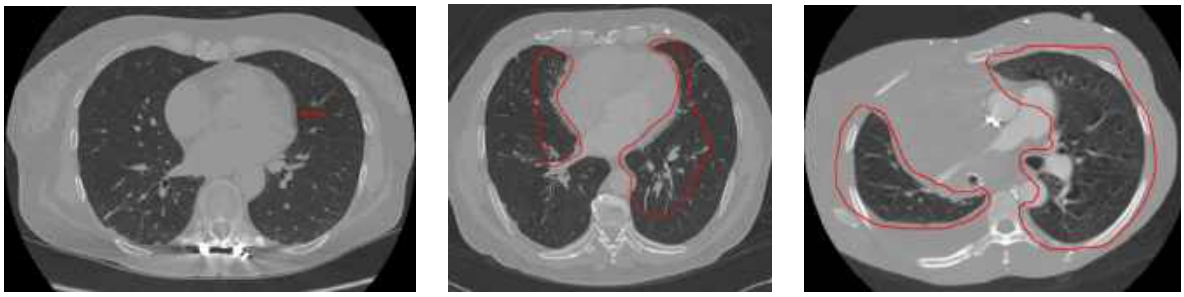

Figure 4.1: Classes of Motion Severity: (a) Mild (b) Moderate (c) Severe

After the visual scoring was done, I measured the agreement between different parties. Since the data is categorical, I used Cohen's Kappa score to measure agreements between two raters. The Kappa score is used to measure the interrater reliability between two raters and is calculated as:

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)}$$

where $\Pr(a)$ represents the actual observed agreement, and $\Pr(e)$ represents chance agreement. Since our categories of mild, moderate, and severe happen to be ordinal we will also be measuring the weighted kappa score using linear weights. A kappa score of ≥ 0.8 signifies a strong and substantial agreement.

Section 4.1.3 explains the initial hypothesis of this study along with the expectations of the model performances. Further, section 4.1.4 describes the actual results of the analysis and then finally in section 4.1.3 we make the ROC analysis of the results and report the findings with the conclusions of the evaluation study.

4.1.3 Initial Hypothesis

Before conducting this study, the initial hypothesis made was that the observer agreement i.e. the Kappa score measured between the two analysts would be substantially high. Which would imply that both the analysts agree strongly on the outcome of the visual scoring i.e. the categories of motion severity assigned to each scan. When we compare both models to the analysts, we expect model 2.08 to have a strong agreement with both the analysts while model 1.04 to have a lower agreement. Which would imply that model 2.08 is much better at determining the category of motion detected in a scan and also determining whether a scan is suitable for further diagnosis.

4.1.4 Results

We first made a confusion matrix of the analyst classifications as shown in Table 4.1. Based on it, we can tell that both analysts agreed on 21 scans as mild, 22 scans as moderate, and 8 scans as severe.

Table 4.1: Confusion Matrix of Analyst Classifications

		Analyst 2			
		Mild	**Moderate**	**Severe**	**Total**
Analyst 1	**Mild**	21	10	1	**32**
	Moderate	15	22	13	**50**
	Severe	0	5	8	**13**
	Total	**36**	**37**	**22**	**95**

The Kappa calculation on this confusion matrix yields the following results:

Observed agreement, $\Pr(a)$	$= 51/95 = 53.68\%$
Expected agreement, $\Pr(e)$	$= 34.6/95 = 36.42\%$
Kappa Coefficient, κ	**$= 0.271$**
Weighted Kappa, $w\kappa$	**$= 0.381$**
Standard Error, SE_κ	$= 0.081$
95% Confidence Intervals, CI	$= 0.114$ to 0.429

As we can clearly see, the weighted kappa score of 0.381 between the two analysts signifies a minimal agreement which is contradictory to our initial hypothesis. This implies that the analysts do not agree on the category of motion severity on a large number of scans leading to the result. This outcome could be due to various factors. One of them could be that the amount and severity of motion contained in each scan is quite subjective leading to different opinions amongst multiple human observers. Another factor could be that the scans where the analysts disagree are difficult to analyse or confusing enough to lead to a non-consensus between both parties. Either way, it implies that we don't have a strong reference set to compare our models to and that our initial hypothesis is rejected. We hence proceed with our analysis in an alternative approach.

Clinically the purpose of the model is to separate scans which are not analysable for further diagnosis from the scans which are. The criteria in the initial protocol dictates that the scans which are absolutely not suitable for further analysis are classified as SEVERE. Hence we group the MILD and MODERATE categories into one category 'Non-Severe' and leave the 'Severe' category as is. The new confusion matrix therefore becomes:

Table 4.2: Binary Confusion Matrix of Analyst Classifications

		Analyst 2		
		Non-Severe	**Severe**	**Total**
Analyst 1	**Non-Severe**	68	14	**82**
	Severe	5	8	**13**
	Total	**73**	**22**	**95**

From the new matrix, we can say that the analysts agree in 68+14=76 scans of the 95, while disagreeing on 5+14=19 scans. From the scans classified as 'Severe', both analysts agree on 8 scans, while either one of the analysts thinks that 19 scans are severe in nature. Using this new confusion matrix, we now perform an ROC analysis where we compare both our models 2.08 and 1.04 to the analyst classifications in section 4.1.5.

4.1.5 ROC Analysis

Now that we have our data into 2 distinct classes, we can perform the Receiver Operator Characteristic (ROC) analysis on the data. An ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. ROC is a probability curve and the area under the curve (AUC) represents degree or measure of separability. It demonstrates how much a model is capable of distinguishing between two classes. The higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. In our case, higher the AUC, the better a model is at separating non-diagnosable scans from diagnosable ones. The ROC curve is plotted with the true-positive rate (TPR) against the false-positive rate (FPR) where TPR is on the y-axis and FPR is on the x-axis.

Before making the ROC plots, let's go back to table 4.2. We need a strong base of reference or a standard ground truth for performing a reliable comparison between the two models in the ROC plots. From the table, we can see that the analysts agreed upon 76 scans but disagreed for 19 scans. The main question was what could we do to resolve the labels of these 19 scans. There were a few approaches we could take:

1. We could have had the two analysts sit down together and discuss on which of these 19 scans should be rejected or accepted for further analysis. A drawback of this approach is that the analyst who has the bigger mouth could have dominated the discussion and produced the outcome in their favour, creating a huge bias.

2. Another approach was that we could have had a third expert analyst who would act like a super-analyst and decide the labels of these 19 scans themselves.

3. We could have removed these ambiguous 19 scans from the analysis and make the ROC plot only from the 76 scans where there is a clear consensus.

4. We could have assigned a clear label of 'Severe' or 'Non-severe' to these 19 scans ourselves and included them in the analysis.

Approaches 1 and 2 were impossible to carry out due to the unavailability of the analysts and due to time constraints. Hence we chose to go ahead with approaches 3 and 4. For the first ROC plot, we include only 76 scans where there is a clear consensus between the analysts on whether the scan is acceptable for further diagnosis or not. For the second ROC plot, we make a decision for the ambiguous 19 scans that if either one of the analysts has marked it as 'Severe' then the scan should be rejected for further analysis altogether giving it the 'Severe' label. Therefore, we also include the ambiguous 19 scans in the second plot while giving them the label of 'Severe'. Hence ROC plot 1 has 68 scans marked 'Non-severe' and 8 scans marked 'Severe'. ROC plot 2 has 68 scans marked 'Non-Severe' and 27 scans marked 'Severe'. For making both plots, we calculated the percentage of motion in the lungs for each scan from both models' predictions and compared them against the analyst labels. Let's now look at both ROC plots shown in Figures 4.2 and 4.3.

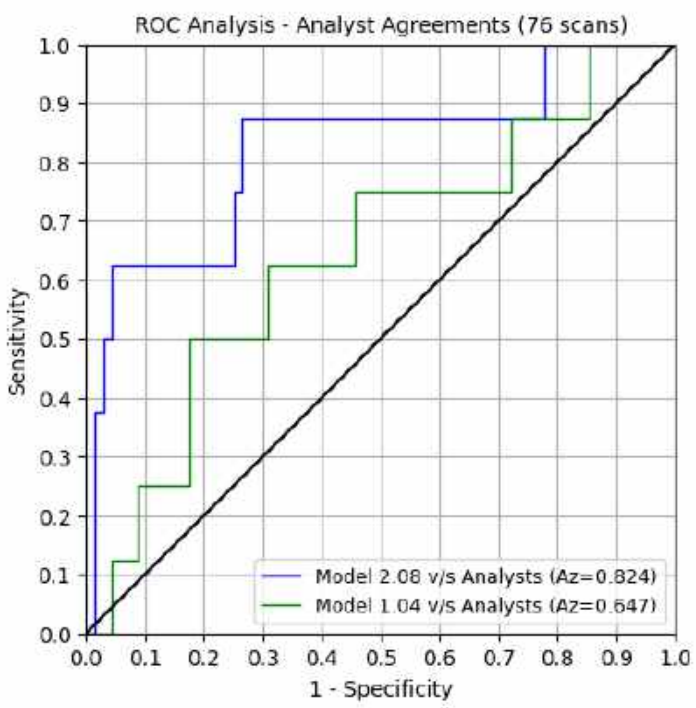

Figure 4.2: ROC Plots of Analyst Agreements

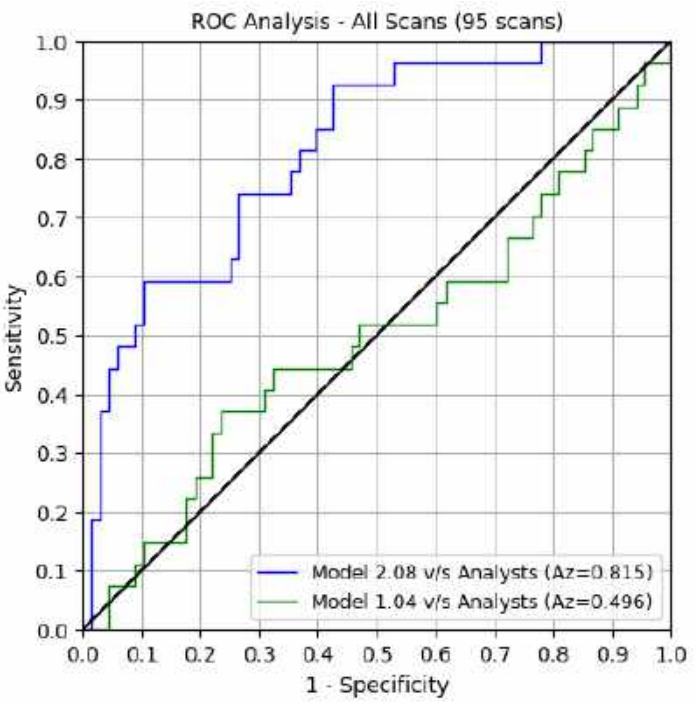

Figure 4.3: ROC Plots of Analyst Agreements + Disagreements

From both figures, the first thing we can clearly conclude is that model 2.08 (blue curves) performs better than model 1.04 (green curves). Model 2.08 has an AUC of 0.824 in figure 4.2 and 0.815 in figure 4.3 which is higher than 0.647 and 0.496 respectively in both cases. The main thing to note is that model 1.04 performs worse in the plot in 4.3 as compared to 4.2 as shown by the green curves. This has to do with the fact that there is disagreement in plot 4.3. Plot 4.2 has 76 scans which are unanimously accepted or rejected for further diagnosis. Plot 4.3 adds 19 more scans which have some motion but they won't be obviously rejected or accepted. The fact that there is disagreement shows that model 1.04 performs reasonably well with easy cases which either have no motion or lots of motion. The AUC reported in this case is quite low (0.647) but still better than chance (0.5). But the moment we add more insecurities to the data implying disagreement, model 1.04 is not as distinctive as 2.08. At this point, 1.04 starts breaking down while 2.08 stays unaffected. Model 2.08 not only performs well on the easier scans, but it is probably also better at estimating the overall percentage of motion. Hence it finds motion even in difficult scans and the ROC plots prove this very well. Earlier we discussed some approaches for resolving dispute for the 19 scans by involving analysts. This won't be very effective on the analysis, it could probably only slightly affect the green curve yielding a different AUC in plot 4.3. It still won't be significant enough to outperform the blue curve either way.

The higher level evaluation study and the ROC analysis was entirely conducted on a dataset which wasn't used for the training or testing of the models during development. Therefore, we can confidently say that it offers a fair judgement of performance of the models. In clinical and practical applications, we can do nothing with these curves since all that's required is for the system is to tell the user whether a particular scan is suitable for further analysis or not. The system would get a scan as input and would predict a figure of percentage of motion in the lungs, and that's it. This is where we can use these curves to decide how we are going to implement a model with this performance by fixing an operation point on the curve. For example in plot 4.2, if we are interested in developing a highly sensitive model which rejects many scans, while being less interested in the specificity, we could choose the point on the blue curve which has a 100% sensitivity and an FPR ≈ 0.77 implying a specificity ≈ 0.23. This would translate to the model thresholds which find all sorts of heavy motion along with false positives to be rejected for further analysis. On the other hand, if we still want to be highly sensitive without rejecting that many scans, we could choose an operation point on the blue curve which has a sensitivity ≈ 0.88 and an FPR ≈ 0.27 implying a higher specificity of ≈ 0.73. This operation point would set the model thresholds which are highly specific and reject much lesser number of scans.

In summary, this is the practical performance we can expect from the model. The larger the evaluation dataset, the better it translates to the real world performance. The point that we pick on the curve translates to the cutoffs for the motion percentages which decide whether to accept or reject a scan for further analysis. So for example if the cutoff from a high sensitivity high specificity operation point translates to 10% then that means that the scans having above 10% motion would be rejected for further analysis. We then know for sure that the model would have a sensitivity close to 90% and a specificity of 73%. A different operation point would evaluate to a different threshold depending on the requirements of the clinical application. Hence, these are some of the examples of how the ROC curves could be used in practical scenarios.

4.2 Lower Level Evaluation

In the previous section, we saw how the model performs on a categorical level in either accepting or rejecting a scan for further diagnosis. Now, we dive deeper into the evaluation and examine both models' performance at the lobe and lung level. For this evaluation study, I chose a subset of 14 scans from the previous set of 95. These scans were chosen such that they contain an equal distribution of the three previously identified categories of motion severity. Similar to the higher level study, we had 2 analysts perform the task of annotating the scans by drawing contours around the motion areas, exactly as the annotation process employed for the training data. Ideally to perform such an evaluation study, a much larger set of scans would be needed because the findings would translate better to the real world. Due to time constraints from the analysts however, we had to go ahead with a much smaller set than intended. Let's now measure the agreement between analyst percentages with the help of the Bland-Altman plot. Figure 4.4 shows the BA plot of both analysts' percentages on a per case basis.

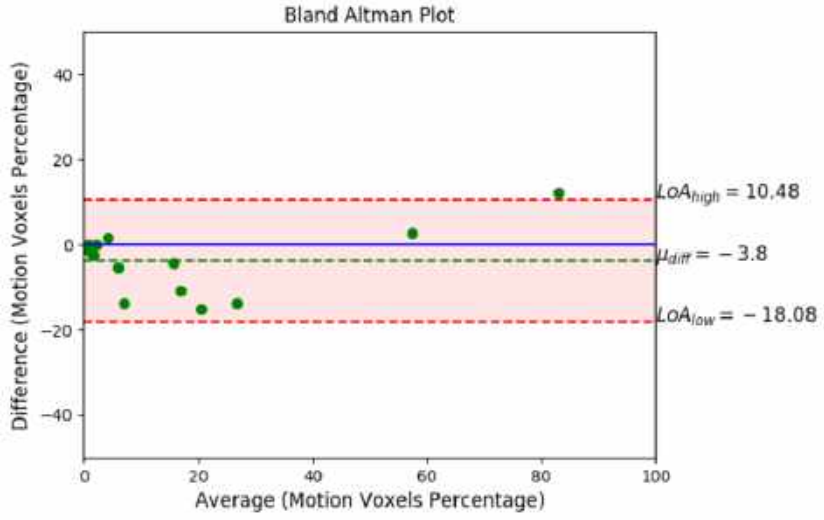

Figure 4.4: Bland-Altman Plot of Analyst Percentages (Per Case)

The BA plot shows a fair to moderate level agreement within the analysts with a mean percentage difference of -3.8 and confidence intervals of -18.08 to 10.48. While 9 out of the 14 cases still lie closer to the mean difference and the zero line, the other 5 cases have slightly worse results lying closer to the confidence intervals. By now we knew that the analysts are not entirely precise when it comes to identifying all regions of motion in the scans. While going through the annotations case by case, I recorded the following observations:

- Both analysts were accurate in identifying motion artifacts i.e. all regions annotated do contain artifacts. False-positives were quite scarce and in many cases don't exist.
- Each analyst missed certain regions with artifacts in the scans. More often than not, these tended to be subtle artifacts.

From these observations we can say that one analyst's annotations do not serve as a strong base of reference for evaluation. Hence, we took a union of both analysts' annotations and

compared the motion segmentations from our models against them in this study. We wanted to see how the models have performed on a lobar level. Hence we made use of the paired box-plots to evaluate the lobar performance. In the case of the lobes though, each lobe has a different size, especially the right middle lobe is much smaller as compared to the other lobes. Which is why comparing relative quantities like percentages of motion within lobes is not intuitive. Instead I calculated an absolute quantity, the volume of motion regions in millilitres within each lobe. To give an example for the reason why we do this, let's consider one of the quantities from the right middle lobe. When percentages are calculated, the original percentage for motion inside the right middle lobe could evaluate to almost 100% while the predicted percentage would be somewhere around 40% which signifies a huge margin of error. Whereas if we make the same calculation in mL, it would evaluate $\approx$ 200 mL in the original and $\approx$ 110 mL in the prediction. This margin of error isn't huge since the volume of motion within the other lobes can go higher than 500 mL. Hence, we used the volume in millilitres instead of the motion percentages to make our plots and analysis. To calculate the volume, we first got the voxel size and spacing (in millimetres) from the headers of the *.mhd* file. The volume of one voxel is calculated from these quantities and multiplied by the number of motion voxels within a particular lobe. Let's first look at the paired box-plot for model 2.08 which is shown in figure 4.5.

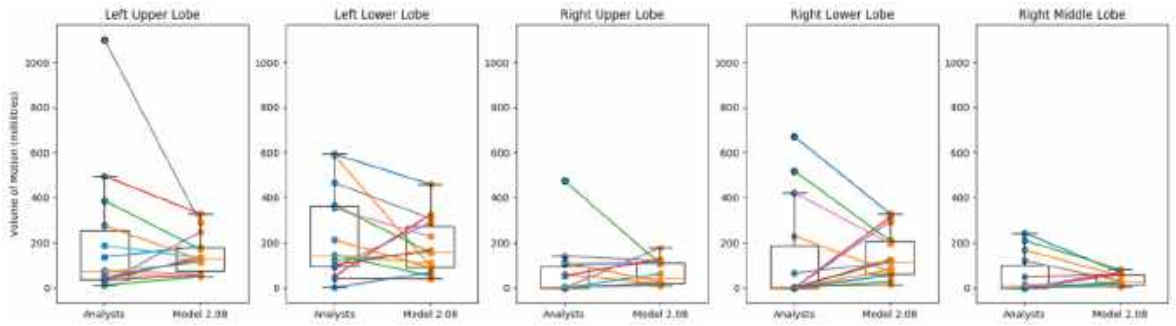

Figure 4.5: Model 2.08 - Paired box-plots per lobe

The first observation we can make from the above plot is that the left lung lobes have more motion than the right lung. This can be attributed to the fact that the left lung contains the heart and most motion artifacts end up occurring due to cardiac activity. The model performs well in the left upper and lower lobes, with the exception of the case indicated by the gray line in the upper lobe. One common aspect we can observe here is that the model ends up predicting within a specific smaller range, around 50 to 350 mL for the left upper lobe and around 50 to 500 mL for the left lower lobe. Cases which have much less motion are overestimated while cases which have high motion are underestimated. As for the lobes of the right lung, the model tends to perform quite well in the right upper lobe with an exception of the case indicated by the green line. For the right lower lobe, we see some heavy differences with both under and overestimation which suggests that some improvement is needed. As for the right middle lobe which is much smaller in size, we see much lesser quantities in the ground truth as well as the predictions. A general trend in this lobe points towards underestimation which would still require some improvements. Let's now see the paired box-plots for model 1.04 and compare them to model 2.08. Figure 4.6 shows this paired box-plot.

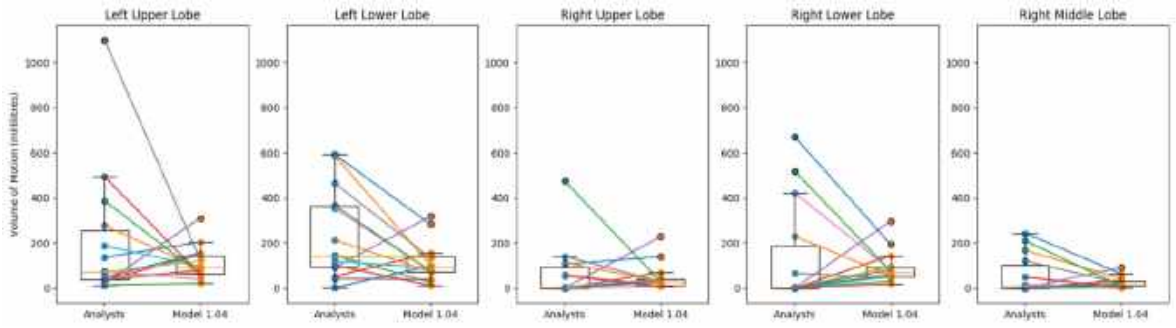

Figure 4.6: Model 1.04 - Paired box-plots per lobe

The first thing which we can strikingly notice is that the ranges of predictions for each lobe are much smaller than model 2.08. Most notably in the left lower, right upper and right lower lobes, we can see heavy differences in the sizes of the boxes. This implies that no matter how much motion is contained in the lobes, the model will only predict in a fixed small range. The results from this model will see slight over-estimations from scans with low motion and heavy underestimations from scans with severe motion. This is also confirmed by the ROC analysis we did previously. The ROC curves showed that model 1.04 is worse at rejecting scans for further analysis which is additionally confirmed by lobar evaluations. They also showed that model 1.04 falters and drops heavily in performance when it sees ambiguous scans i.e. scans with disagreement between human observers on the severity of motion.

4.3 Evaluation of COVID-19 Scans

The novel coronavirus disease has created a global pandemic which has affected society worldwide. COVID-19 is a disease that directly affects the lungs. Since this project began at a time when the pandemic had just started, and because of its relevance to the problem statement, we also decided to see how our model performs on lung scans from COVID-19 patients. Note that only LIDC scans were used for model training and testing, hence the model hasn't seen any COVID-19 scans during its development. I got a set of 10 scans from Kaggle for the analysis. These scans contain a varying level of COVID-19 severity. Most of these scans also contain severe motion artifacts. Heavy proportions of the lung are affected by motion. A probable cause could be that severe COVID-19 patients end up coughing heavily even while taking the CT scan. This aspect makes the scans interesting to analyse for our project.

COVID-19 lung scans slightly resemble scans from pneumonia patients. A scan from an early stage COVID-19 patient would show bilateral (unilateral in some cases) and rounded ground glass opacities (GGO). A further stage scan (> 10 days) would also start showing consolidations indicating the accumulation of liquid in the lung tissue. Usually, most of the lung with COVID-19 ends up affected by motion which is why for model training, the entire lung segment could be used as the annotation. To be more precise, the analysts made the annotations carefully excluding the regions with GGOs and consolidations. Let's see if the model can also be as precise in segmenting motion around the GGOs and consolidations instead of over them. Figure 4.7 shows a slice from a COVID-19 scan affected almost entirely by motion.

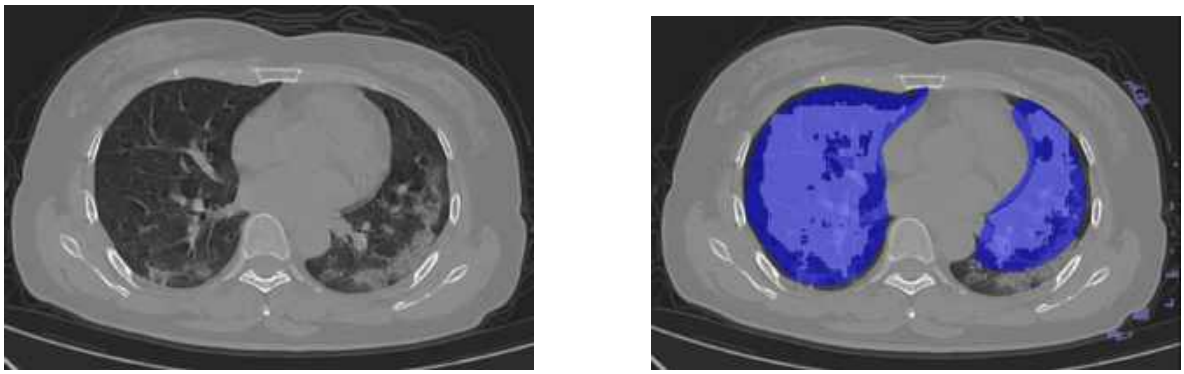

Figure 4.7: COVID-19 Scan Evaluation: (a) Ground Truth (b) Model Predictions

It is quite interesting to see that the model has been precise in identifying the regions with motion while carefully excluding the regions with GGOs and consolidations like the annotations. Note that the model's segmentations are highlighted in light blue while the analyst annotations in dark blue. Figure 4.8 shows a slice from another scan where the model accurately segments the motion regions while avoiding the consolidations and committing minor mistakes in the right lung.

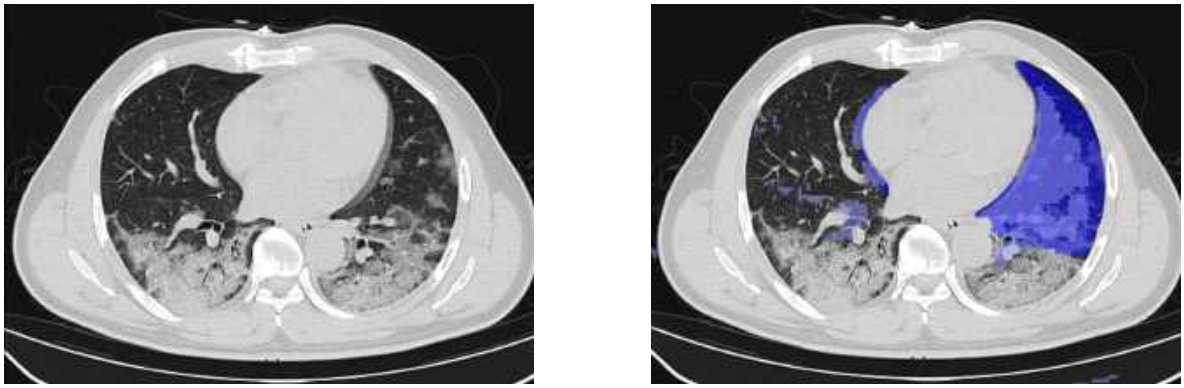

Figure 4.8: COVID-19 Scan Evaluation: (a) Ground Truth (b) Model Predictions

To summarize, we performed two evaluation studies for our model. One of them being the higher level evaluation with a large set of 95 scans. It evaluated the model's capability to accept or reject scans for further diagnosis based on motion severity. And the other being the lower level evaluation with a smaller set of 14 scans. It dived deeper into the lobar performance of the model. Both our evaluations conclusively prove that model 2.08 performs better in detecting motion artifacts in the scans and in accepting or rejecting a scan based on the severity of motion. For both evaluations, we also incorporated COVID-19 scans and discussed model 2.08's performance in them briefly.

Chapter 5

Conclusion and Discussion

5.1 Conclusion

In this thesis, we have demonstrated the potential of deep learning models to predict motion artifacts in CT scans. The final model has a great performance in detecting motion artifacts which are quite obvious and unambiguous. While at the same time, the model has a difficulty in finding artifacts which are quite subtle, like small vibrating airways.

One of the main observations encountered during visual inspections of the experimental results was that the analysts are not entirely precise when it comes to annotating scans for motion. Quite often it is observed that they have missed huge regions with motion which were pretty obvious to spot. In many cases, the model outperforms the analysts by finding these exact regions precisely in the scan. In the ground truth, many clean slices and clean regions were annotated as a result of interpolation from the pattern annotation tool. In these cases as well, the model goes one step further and does not classify these areas as motion. Both these factors together introduce false-positives and false-negatives in our evaluation metrics and plots. This is why for most of our experimental processes, the visual results have told a different story as compared to the quantitative metrics.

The evaluation studies showed that there is a heavy disagreement between two or more different human observers when it boil down to the severity of motion in a scan. It implies the existence of a prominent subjectivity factor along with a difference of opinion between analysts. The disagreement coupled with the subjectivity factor translates to the fact that "detection of motion artifacts" is indeed a difficult problem to solve. In spite of these obstacles, we see the model perform decently, and in many cases even better than the analysts at detecting motion areas as well as accepting or rejecting a scan for further diagnosis. Eventually after addressing the issues and making improvements, we can conclusively say that the model can replace a human analyst in accepting or rejecting scans.

During our experiments, we found common mistakes in the slice based experiments which directed the switch to a patch based approach. We got the best results out of a patch based model coupled with false-positive mining. In all honesty, we do not claim the patch based approaches to be better than slice based. With more time and research, the best model could probably have been obtained from the slice based approach by addressing its issues and building upon them. Or maybe a combination of the two could have yielded a stronger model combining the strengths of both. Ultimately, this problem can be solved by many different approaches and we chose our path by making decisions based on all intermediate findings.

The mini-batch balancing and false-positive mining strategies have proven to be effective in making the model learn what we want it to learn. With a few modifications to the data pipeline as per the clinical application, this solution as whole can be generalized to solve more problems beyond the scope of motion detection.

5.2 Future Directions

In this section, we identify several aspects of the thesis project that can be expanded upon and improved.

Additional Data

One of the first things we can do to improve the model performance is the incorporation of scans with disease. Based on the clinical application, an appropriate dataset could be chosen for training. For example, a system detecting motion artifacts in lung CT scans with pneumonia would yield the best performance on training with pneumonia scans. Along with that, training with scans which have an abundant representation of subtle motion artifacts could greatly improve the performance of the model.

3D Models and Approaches

Training with 3D data and models instead of 2D could solve the problem of visual discontinuity in the motion segmentations even though it isn't important for the clinical application. This would mean using 3D U-Nets as well as training with 3D patches of a fixed size, since it is practically impossible to use an entire scan as one input to the model.

Additional Segmentation Classes

Currently we use only two classes - motion or no motion for our segmentations. To go one step further, we could have different classes for the different types of motion. In our case, it would imply different annotation and segmentation classes for cardiac and respiratory motion.

More Precise Annotations

One of the main problems is the fact that analyst annotations aren't of great quality. In the future, having highly trained experts perform a more precise and accurate annotation process would guarantee more reliable quantitative evaluation metrics and an overall better performance of the model.

Bibliography

[1] Pedro F. Ferreira, Peter D. Gatehouse, Raad H. Mohiaddin, and David N. Firmin. Cardiovascular magnetic resonance artefacts, dec 2013. 1

[2] F Edward Boas and Dominik Fleischmann. CT artifacts: Causes and reduction techniques. Technical Report 2, 2012. 1

[3] Guangming Zhu, Bin Jiang, Liz Tong, Yuan Xie, Greg Zaharchuk, and Max Wintermark. Applications of deep learning to neuro-imaging techniques. *Frontiers in Neurology*, 10(AUG):1–13, 2019. 2

[4] Benedikt Lorch, Ghislain Vaillant, Christian Baumgartner, Wenjia Bai, Daniel Rueckert, and Andreas Maier. Automated Detection of Motion Artefacts in MR Imaging Using Decision Forests. 2017. 2

[5] Thomas Küstner, Annika Liebgott, Lukas Mauch, Petros Martirosian, Fabian Bamberg, Konstantin Nikolaou, Bin Yang, Fritz Schick, and Sergios Gatidis. Automated reference-free detection of motion artifacts in magnetic resonance images. *Magnetic Resonance Materials in Physics, Biology and Medicine*, 31(2):243–256, apr 2018. 2

[6] Daiki Tamada and Hiroshi Onishi. Motion artifact reduction in abdominal MR imaging using the U-NET network. Technical report, 2018. 2

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9351, pages 234–241. Springer Verlag, may 2015. 2, 13

[8] Ilkay Oksuz, Bram Ruijsink, Esther Puyol-Antón, James R. Clough, Gastao Cruz, Aurelien Bustin, Claudia Prieto, Rene Botnar, Daniel Rueckert, Julia A. Schnabel, and Andrew P. King. Automatic CNN-based detection of cardiac MR motion artefacts using k-space data augmentation and curriculum learning. *Medical Image Analysis*, 2019. 2

[9] Harriet Small and Jonathan Ventura. Handling Unbalanced Data in Deep Image Segmentation. Technical report. 3

[10] Sejin Park, Woochan Hwang, Kyu Hwan Jung, Joon Beom Seo, and Namkug Kim. False Positive Reduction by Actively Mining Negative Samples for Pulmonary Nodule Detection in Chest Radiographs. jul 2018. 3

[11] Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, Lawrence Tarbox, and Fred Prior. The cancer imaging archive (TCIA): Maintaining and operating a public information repository. *Journal of Digital Imaging*, 26(6):1045–1057, dec 2013. 9

[12] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. Tversky loss function for image segmentation using 3D fully convolutional deep networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10541 LNCS:379–387, jun 2017. 15

Appendix A

Model Schematics

A.1 Compact U-Net for experiments with 2D Slices

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 512, 512, 1)	0	
conv2d_1 (Conv2D)	(None, 512, 512, 8)	80	input_1[0][0]
conv2d_2 (Conv2D)	(None, 512, 512, 8)	584	conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 256, 256, 8)	0	conv2d_2[0][0]
conv2d_3 (Conv2D)	(None, 256, 256, 16)	1168	max_pooling2d_1[0][0]
conv2d_4 (Conv2D)	(None, 256, 256, 16)	2320	conv2d_3[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 128, 128, 16)	0	conv2d_4[0][0]
conv2d_5 (Conv2D)	(None, 128, 128, 32)	4640	max_pooling2d_2[0][0]
conv2d_6 (Conv2D)	(None, 128, 128, 32)	9248	conv2d_5[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 256, 256, 32)	0	conv2d_6[0][0]
cropping2d_1 (Cropping2D)	(None, 256, 256, 16)	0	conv2d_4[0][0]
concatenate_1 (Concatenate)	(None, 256, 256, 48)	0	up_sampling2d_1[0][0] cropping2d_1[0][0]
conv2d_7 (Conv2D)	(None, 256, 256, 16)	6928	concatenate_1[0][0]
conv2d_8 (Conv2D)	(None, 256, 256, 16)	2320	conv2d_7[0][0]
up_sampling2d_2 (UpSampling2D)	(None, 512, 512, 16)	0	conv2d_8[0][0]
cropping2d_2 (Cropping2D)	(None, 512, 512, 8)	0	conv2d_2[0][0]
concatenate_2 (Concatenate)	(None, 512, 512, 24)	0	up_sampling2d_2[0][0] cropping2d_2[0][0]
conv2d_9 (Conv2D)	(None, 512, 512, 8)	1736	concatenate_2[0][0]
conv2d_10 (Conv2D)	(None, 512, 512, 8)	584	conv2d_9[0][0]
zero_padding2d_1 (ZeroPadding2D	(None, 512, 512, 8)	0	conv2d_10[0][0]
conv2d_11 (Conv2D)	(None, 512, 512, 1)	9	zero_padding2d_1[0][0]

Total params: 29,617
Trainable params: 29,617
Non-trainable params: 0

Figure A.1: Summary of the U-Net Model

Appendix B

Experimental Results

B.1 Model 1.04

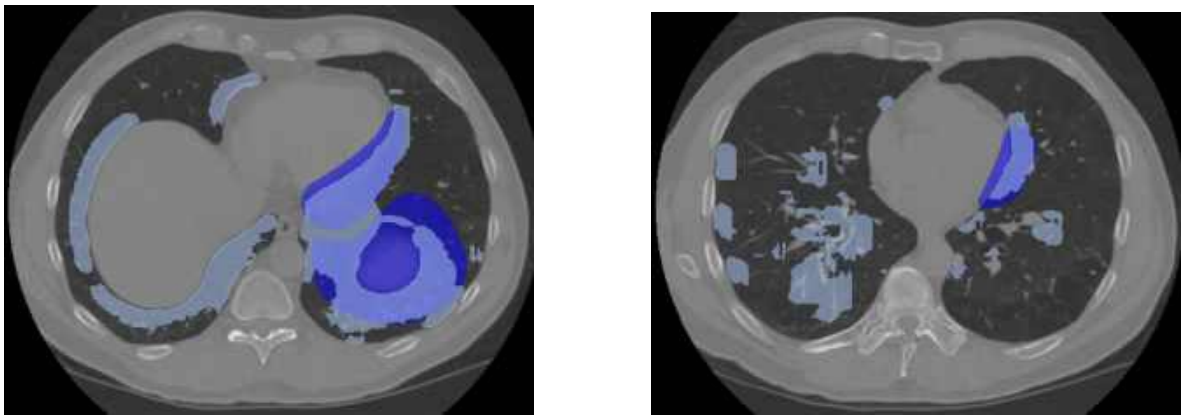

Figure B.1: Visual Results: (a) False positives in lower lobes (b) False positives in parenchyma

B.2 Model 2.01

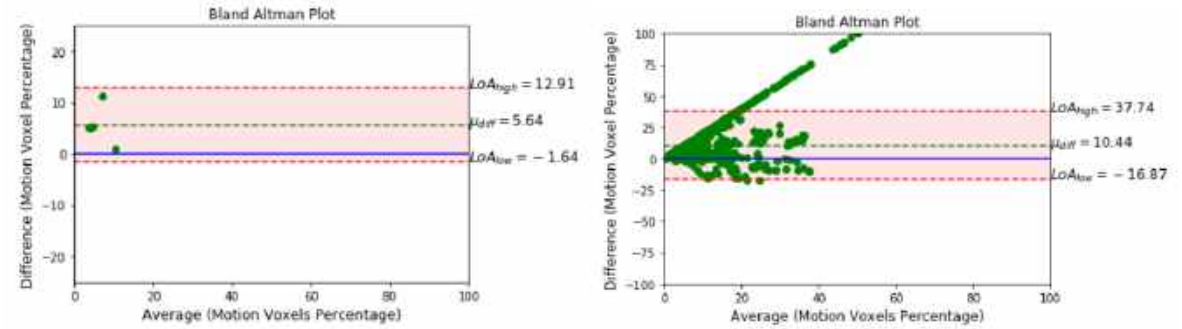

Figure B.2: Bland-Altman plots

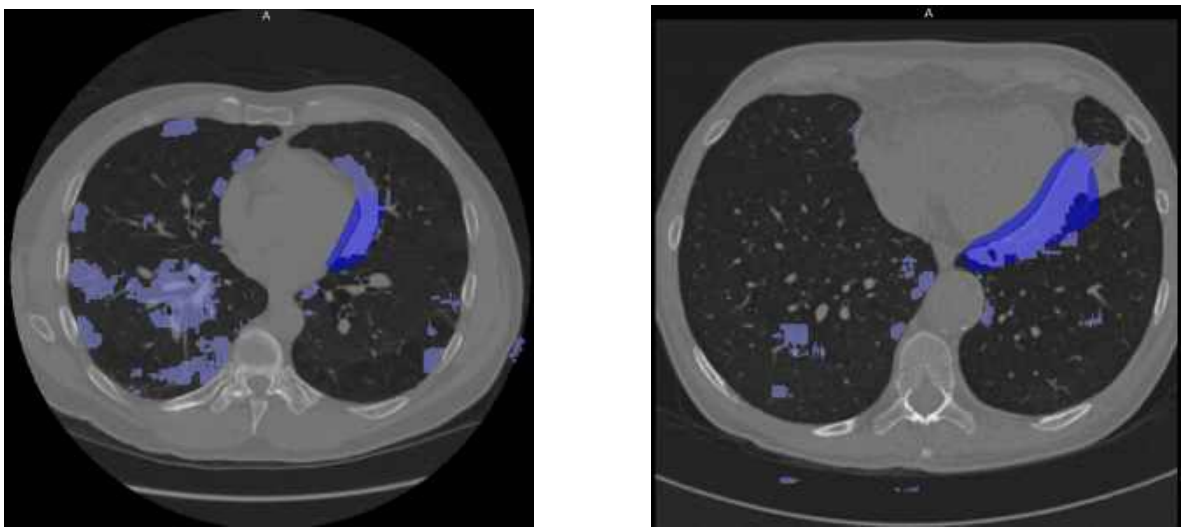

Figure B.3: Visual Results: (a) False positives in parenchyma (b) True positives around heart periphery

B.3 Model 2.03

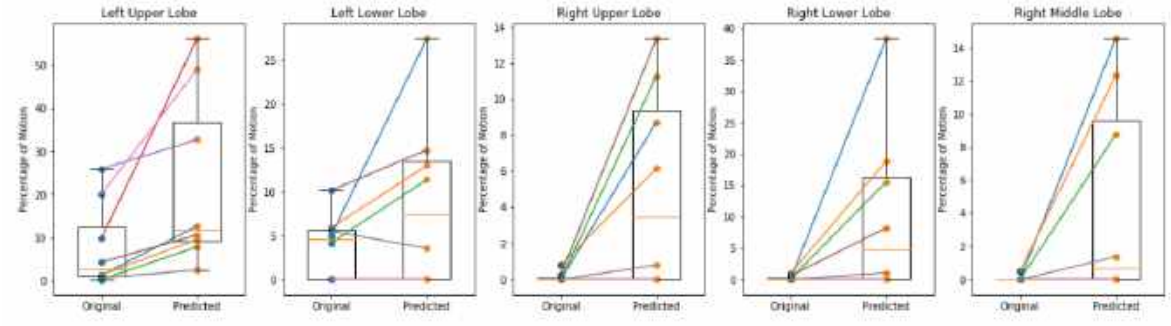

Figure B.4: Paired box-plots of the pre-trained model

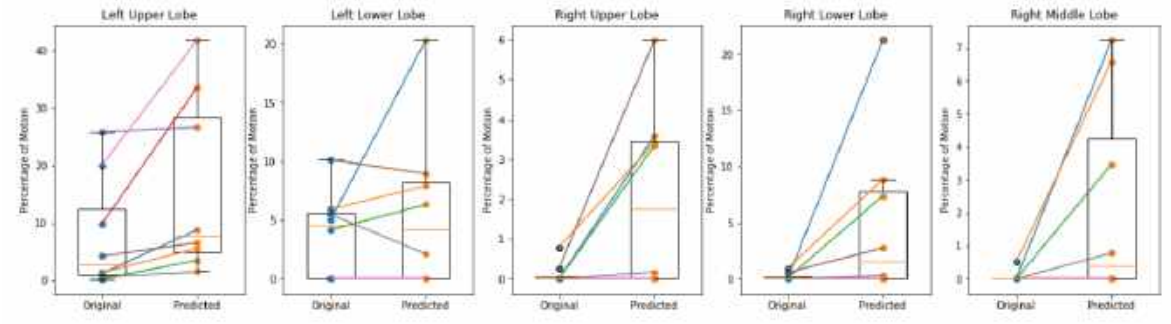

Figure B.5: Paired box-plots of the FP mined model

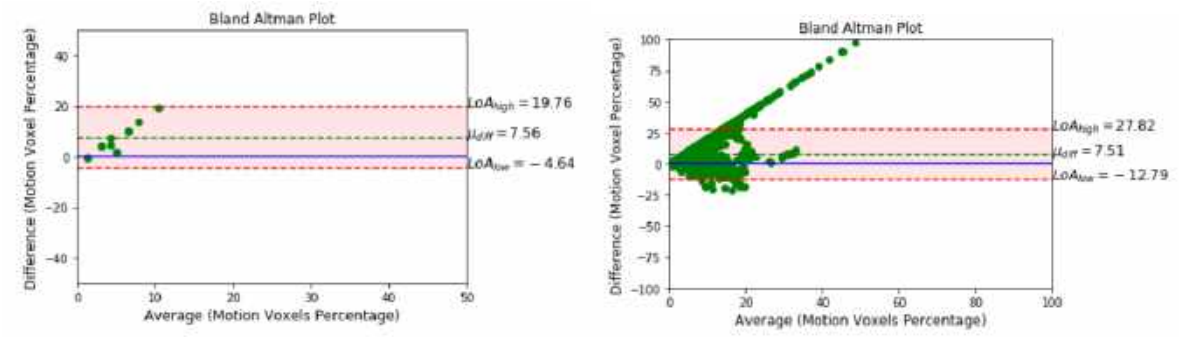

Figure B.6: Bland-Altman plots

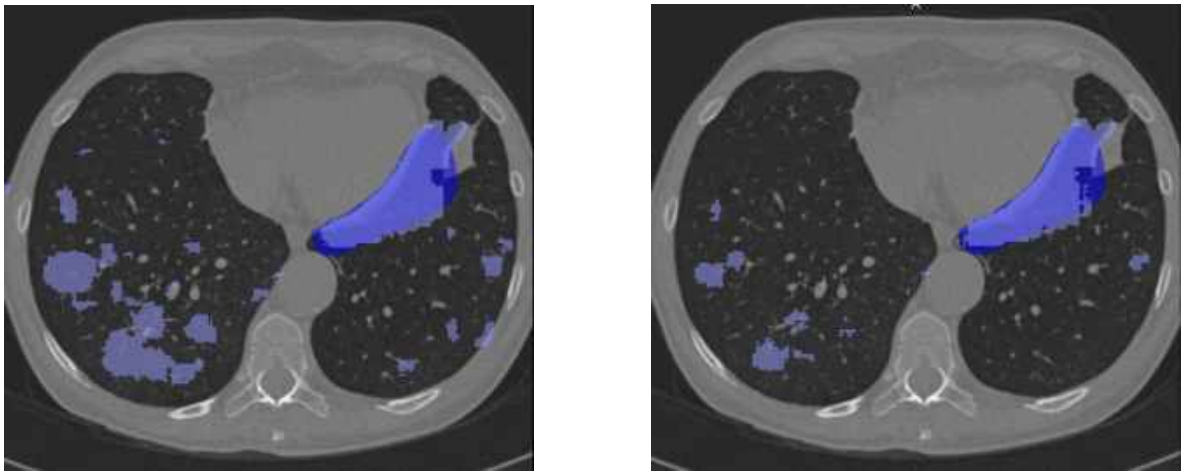

Figure B.7: Model Predictions: (a) Before FP Mining (b) After FP Mining

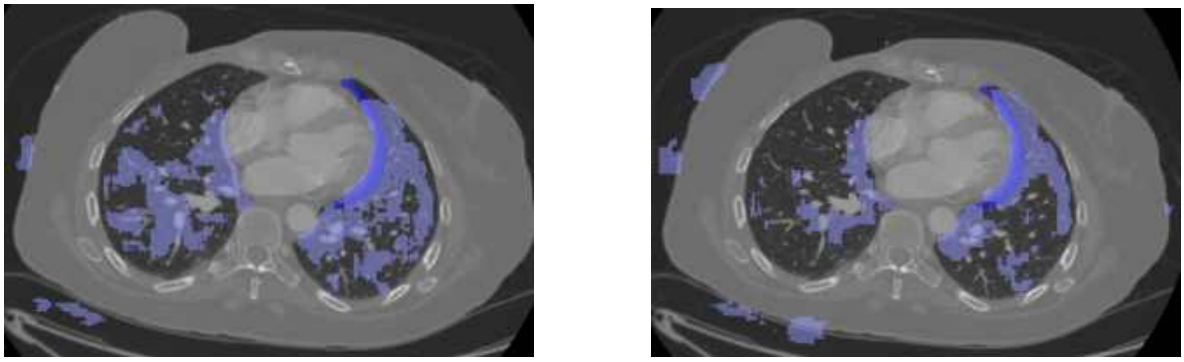

Figure B.8: Model Predictions: (a) Before FP Mining (b) After FP Mining

B.4 Model 2.04

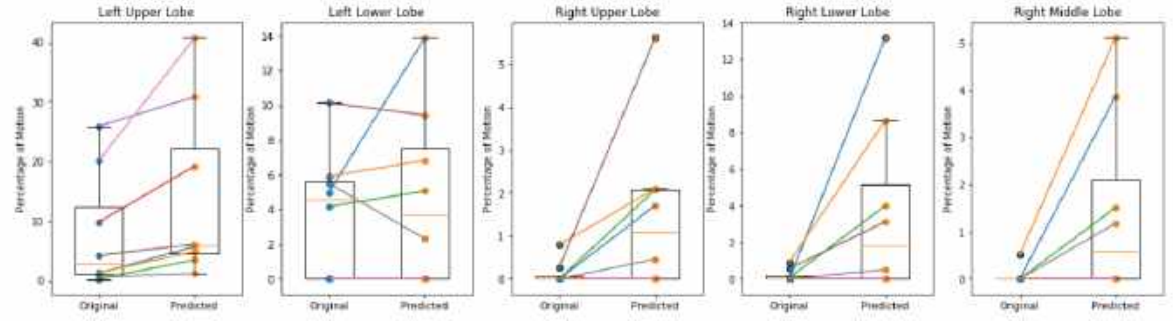

Figure B.9: Paired box-plots

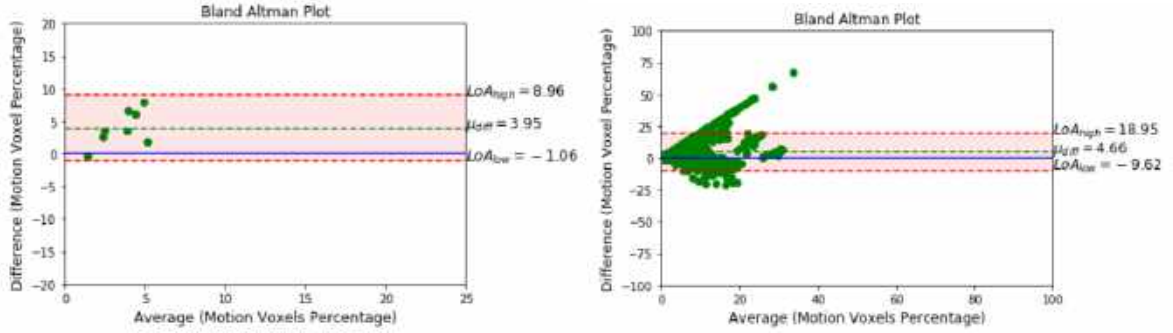

Figure B.10: Bland-Altman plots

B.5 Model 2.05

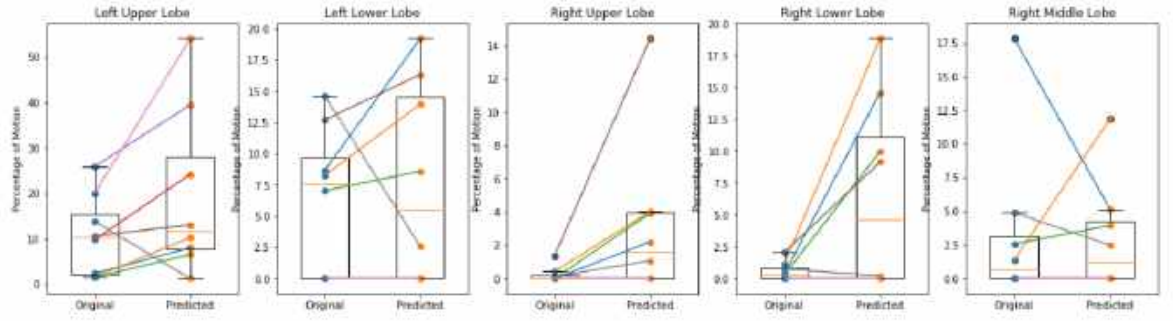

Figure B.11: Paired box-plots

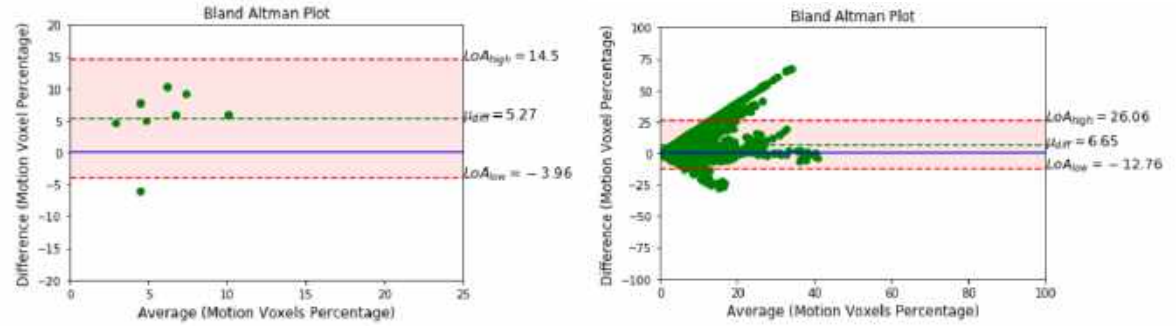

Figure B.12: Bland-Altman plots

B.6 Model 2.06

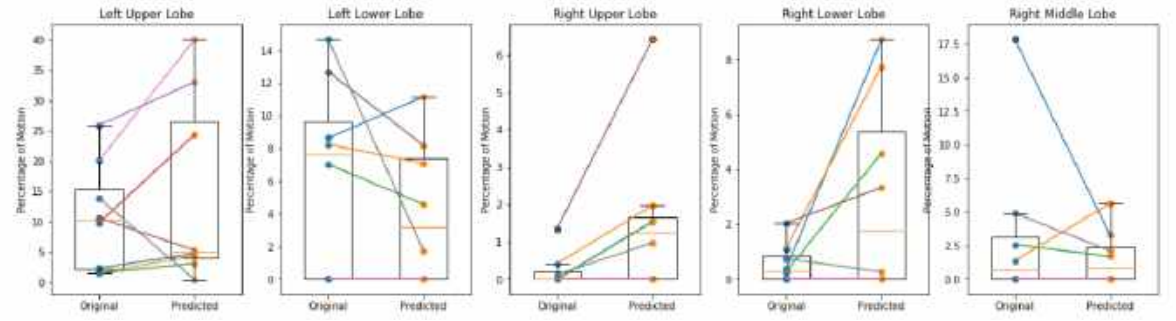

Figure B.13: Paired box-plots

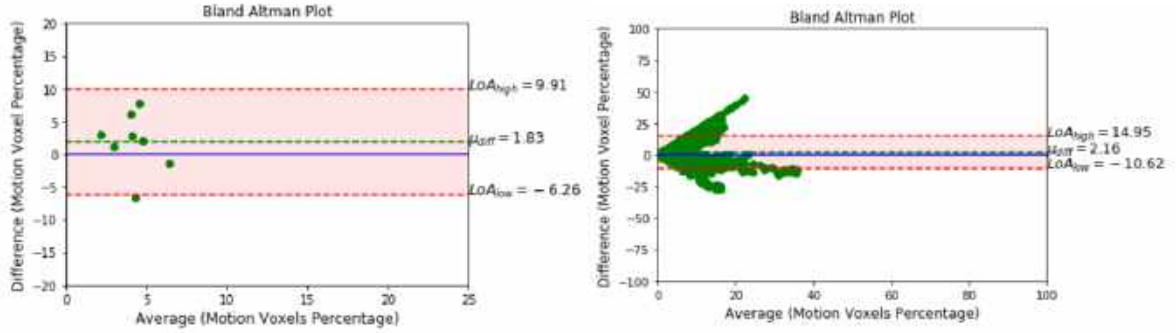

Figure B.14: Bland-Altman plots

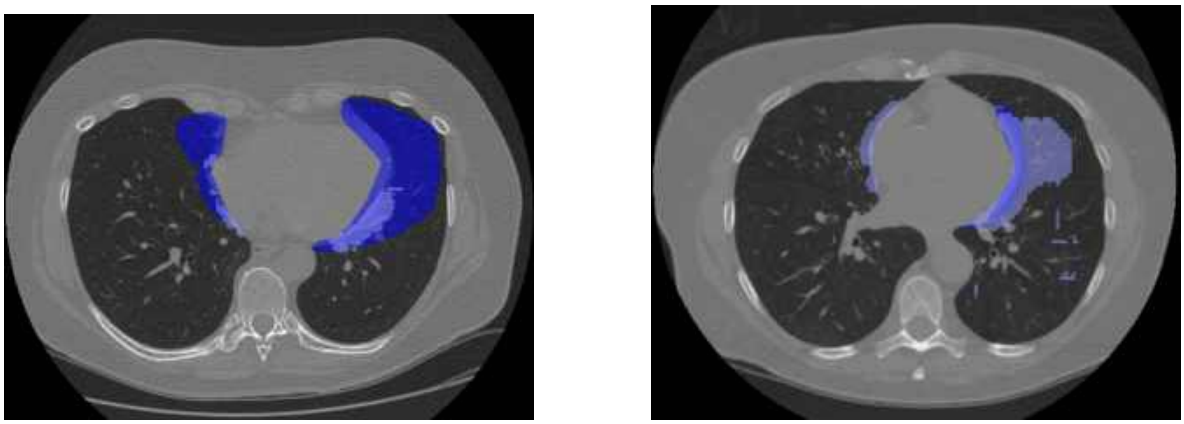

Figure B.15: Visual Results: (a) False negatives which contain subtle artifacts (b) False positives which contain real artifacts missed in the ground truth

B.7 Model 2.07

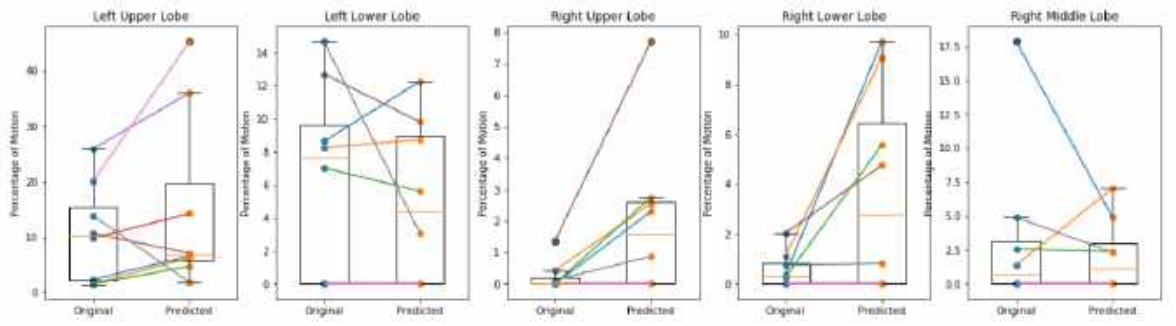

Figure B.16: Paired box-plots

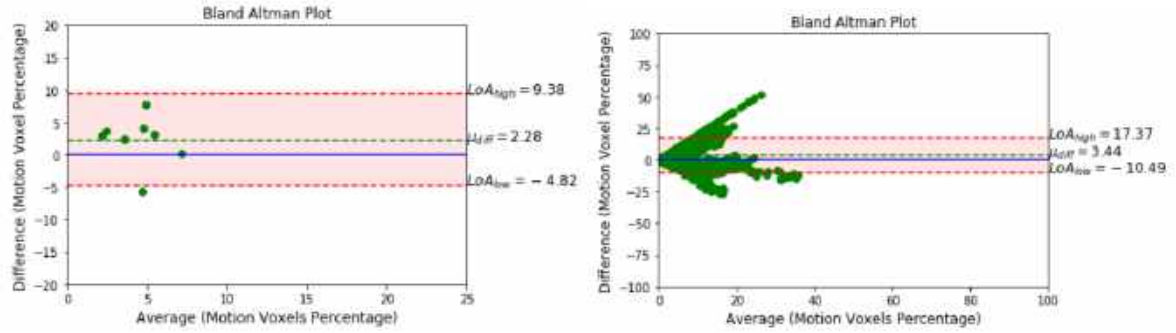

Figure B.17: Bland-Altman plots

B.8 Model 2.08

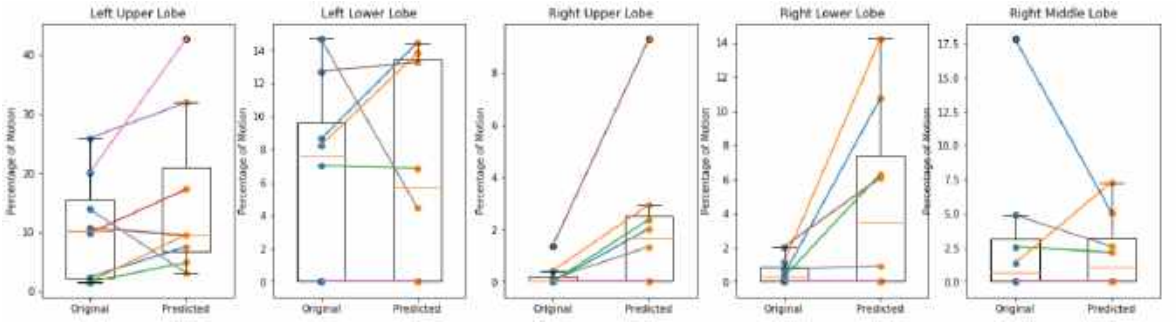

Figure B.18: Paired Box-plots

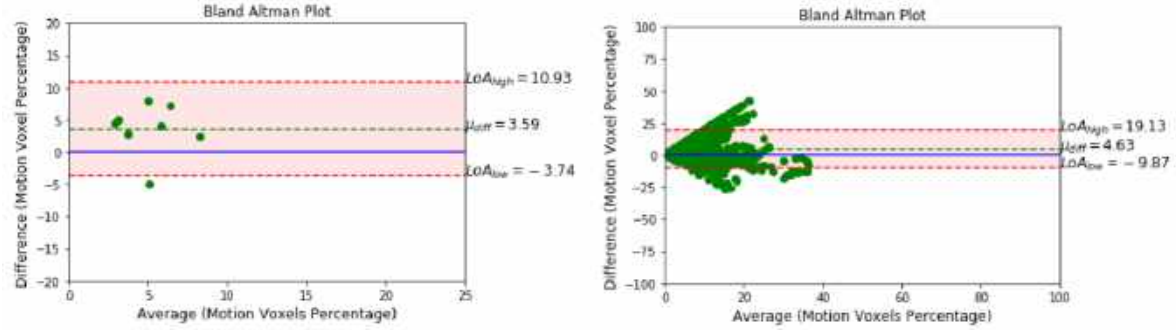

Figure B.19: Bland-Altman plots

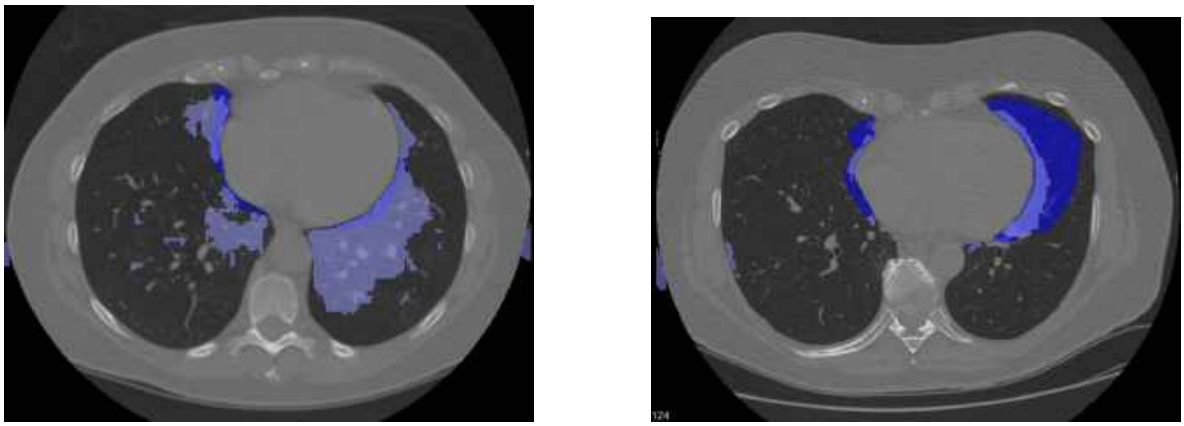

Figure B.20: Visual Results: (a) False positives which contain real artifacts missed in the ground truth (b) False negatives which contain subtle artifacts

To summarize the work, I built a motion artifacts detection system using AI. I had the option to use already working models from Thirona as a starting point. Instead for a better learning process, I decided to build the models and the training process entirely from scratch. I learnt many things during this process. First of all, I learnt how to handle real world medical imaging data. Secondly, I learnt how to solve class imbalance problems with neural networks. Throughout the experimentation of deep learning pipelines, I learnt about different types of real world issues which could arise and how to tackle them. During the project, I was systematically documenting the experiments along with their setup, parameters, observations, and findings. Through this documentation process, I learnt how to create and maintain a comprehensive record to track the experiments. And finally, I learnt how to work and stay motivated during a global crisis which is the COVID-19 pandemic.