Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Vemund Fredriksen & Svein Ole Matheson Sevle

Pulmonary Tumor Segmentation Utilizing Mixed-Supervision in a Teacher-Student Framework

Master's thesis in Computer Science

Supervisor: André Pedersen, Frank Lindseth, Thomas Langø & Gabriel Kiss

June 2021

NTNU
Norwegian University of Science and Technology

SINTEF

Vemund Fredriksen & Svein Ole Matheson Sevle

Pulmonary Tumor Segmentation Utilizing Mixed-Supervision in a Teacher-Student Framework

Master's thesis in Computer Science
Supervisor: André Pedersen, Frank Lindseth, Thomas Langø & Gabriel Kiss
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

Cancer is a leading cause of death in the developed world, and lung cancer is the most lethal. Clinical experts rely on advanced medical imaging techniques and visual inspection to detect lung tumors. Prognosis is highly dependent on the stage of cancer, and mortality can be reduced by early detection. Recent development in machine learning techniques shows promise for automating time-consuming tasks that are currently performed manually by trained experts.

Lack of annotated data is one of the primary constraints in developing automatic methods for medical image segmentation tasks. Public datasets often contain different degree of annotations, if any. In an attempt to address some of these problems, we have investigated the potential of training deep neural networks that can learn from a larger set of less accurately annotated data, to improve performance on lung tumor segmentation. We implemented a framework, based on the recently emerging Teacher-Student design, to utilize bounding box annotated datasets to facilitated lung tumor segmentation learning.

Our research shows that using a sufficiently large, less accurately annotated dataset, the Teacher-Student framework can improve segmentation and detection performance. We also demonstrate that our produced teacher may be used as a semi-automatic method to facilitate the labeling of medical images.

Our best model, trained from only 48 human-annotated images and 991 teacher-annotated images (given bounding box supervision), reached a Dice Coefficient Score of 0.7156 on the MSD dataset, tested on nine images, which is on the state-of-the-art level. Another model trained under the same conditions, reached a perfect tumor-level F1 Score of 1.0 on the MSD dataset.

Keywords Lung Cancer, Deep Learning, Medical Image Segmentation, Mixed Supervision, Teacher-Student Framework

Sammendrag

Kreft er en av de fremste dødsåraskende i den utviklede verden, og den mest dødelige typen er lungekreft. Kliniske eksperter er avhengig av avanserte medisinske bildebehandlingsteknikker og visuell analyse for å oppdage kreftsvulster. Hvor lenge kreften har utviklet seg påvirker i stor grad prognosen, og dødeligheten kan reduseres ved tidlig påvisning. Utvikling innen maskinlæringsteknikker den siste tiden har åpnet for optimisme knyttet til å kunne automatisere tidkrevende oppgaver som i dag utføres manuelt av trente eksperter.

Begrenset tilgang til annotert data er en av de største utfordringene knyttet utviklingen av automatiserte metoder for medisinsk bildesegmentering. Offentlig tilgjengelige datasett inneholder ofte ulike typer annoteringer. I et forsøk på å løse noen av disse problemene, har vi undersøkt potensialet i å trene dype nevrale nettverk som kan lære av et større datasett, med mindre nøyaktig annoterte data, for å forbedre ytelsen på lungesvulstsegmentering. Vi implementerte et rammeverk basert på et nytt konsept kalt *Teacher-Student Design* for å utnytte datasett annotert med avgrensningsbokser for å lære lungesvulstsegmentering.

Vår forskning viser at ved bruk av et tilstrekkelig stort, mindre nøyaktig annotert datasett, kan Teacher-Student-rammeverket forbedre segmenterings- og deteksjonsytelsen til helautomatiske metoder. Vi demonstrerer også at vår implementasjon av *the teacher* kan brukes som en halvautomatisk metode for å bistå i prosessen med å annotere medisinske bilder.

Vår beste modell, trent på kun 48 bilder annotert av eksperter, og 991 bilder annotert av *the teacher*, oppnådde en *Dice Coefficient Score* på 0.7156 på MSD-datasettet, testet på ni bilder. En annen modell trent på samme måte oppnådde en perfekt tumornivådeteksjon målt i F1 på 1.0 på det samme datasettet.

Preface

Ever since our first encounter with deep learning, we knew that this was a field that excites us both, and we continued to pursue this path. We truly believe that artificial intelligence will play a significant role in solving our society's biggest challenges. If we are ever fortunate enough to play just a small part in that, we would be most grateful.

We recognize how fortunate we have been to be able to work within a field that is in the middle of intense advance. We also recognize how privileged we have been to get access to state-of-the-art hardware through NTNU and that we had highly competent supervisors at our disposal throughout the whole project.

A special thanks to Frank Lindseth and Gabriel Kiss from NTNU and Thomas Langø and André Pedersen from SINTEF. With their years of experience with artificial intelligence within the medical domain, they guided us through the semester.

Vemund Fredriksen Svein Ole M. Seule

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

ADAM ADAptive Momentum Estimation. 15, 38, 42

AI Artificial Intelligence. 8, 23, 24, 26, 45

CE Cross Entropy. 12, 13, 26

CNN Convolutional Neural Network. i, 16, 17

CPU Central processing unit. 45

CT Computed Tomography. ii, iv, 2, 3, 5–7, 23, 25, 28, 30, 33–37, 39, 40, 43, 45, 47, 57, 59, 60, 72, 73, 81, 84, 86

DICOM Digital Imaging and Communications in Medicine. 30, 32, 33

DSC Dice Similarity Coefficient. v, 21, 23, 25, 26, 33, 39, 42, 45–49, 51–54, 57, 58, 61, 62, 64, 65, 67–69, 73, 86

GPU Graphics Processing Unit. 23–25, 33, 45

HU Hounsfield Unit. 7, 74

IoU Intersection over Union. 20, 21, 27

MONAI Medical Open Network for AI. 3, 37, 38, 40, 42, 44, 45, 70, 80, 81, 85, 86

MRI Magnetic Resonance Imaging. 6, 26, 59

MSE Mean Squared Error. 12, 13, 22

NIfTI Neuroimaging Informatics Technology Initiative. 28–33, 81, 84

NTNU Norwegian University of Science and Technology. 3, 44, 45

PET Positron Emission Tomography. 6, 26, 30, 33, 57, 59, 60

PReLU Parametric Rectified Linear Unit. i, 22, 37

RAM Random access memory. 45

ReLU Rectified Linear Unit. 12, 22

SGD Stochastic Gradient Descent. 15

SINTEF Stiftelsen for industriell og teknisk forskning. 3

SOTA state-of-the-art. 2, 26, 44, 62, 68, 69

VRAM Video Random Access Memory. 23, 24, 37, 39, 42, 45, 60

CHAPTER 1

INTRODUCTION

1.1 Motivation & Project Description

Cancer is one of the most common causes of death in developed countries. In the US, approximately 30% of the patients with cancer are diagnosed with lung cancer [1]. It is crucial to detect cancer at an early stage, especially while it is still only located within the lungs. The prognosis is drastically worse when cancer spreads. Early detection of cancer relies on multiple factors, one of which is precise medical imaging and the capacity to perform and interpret medical scans. Most of the tasks associated with creating and processing computer-generated images are performed automatically by computers, though experts still perform some of these tasks. Tasks like configuring presets and selecting imaging protocols are still performed by experts. Radiologists today spend quite some time performing segmentation and analysis of the aforementioned medical images. With the recent improvements in the field of deep learning, some of these tasks seems within reach for automation. If tasks like segmentation of medical images could be automated, this would free valuable time from the experts' schedule, and even improve the quality of medical diagnostication.

As briefly mentioned, the field of artificial intelligence, and especially deep learning, has emerged rapidly in recent years. The development of hardware and more accessible data in digital form, is some of the reasons behind this rapid development. It is not a novel idea to automate tasks in the medical sector, as doctors have been relying on computers and algorithms for the last 50 years. However, many tasks have been too difficult to automate with traditional computer science. One of the fields where deep learning has proved to outperform all traditional algorithms is the field of computer vision. Humans are excellent at quickly glance at an image and gather tons of information from this image. Traditionally, computers have never mastered this until recent years. Now, especially after the development of deep convolutional networks, computer vision seems to be ready to tackle image tasks of medical nature.

One of the inconveniences of deep learning is the demand for available data to train the algorithms. Medical data that can be used is limited, its complicated relationship with privacy is among the reasons. On top of that, segmentation annotations are arguably the most expensive type of annotation to create. Availability of annotated data is often sparse and lung tumor data is no exception.

This thesis investigates the possibility of combining multiple types of annotations to further increase the state-of-the-art performance of automatic end-to-end lung tumor segmentation. To make use of these mixed-supervision datasets, we develop a method based on the recently emerging Teacher-Student design. Our method makes use of multiple datasets, some of which contain segmentation annotations, other bounding box annotations. Annotating data with bounding boxes is a much easier job than to annotate with masks. Being able to utilize bounding box datasets to learn segmentation is therefore benificial.

1.2 Thesis Goal

The goal of this thesis is to investigate the use of the Teacher-Student design for lung tumor segmentation on CT images. We also want to explore how well semi-supervised methods can perform lung tumor segmentation given CT image and weaker forms of supervision, as they are less expensive to create and can aid experts, and facilitate dataset annotation.

1.3 Research Method & Research Questions

To research whether the Teacher-Student design can improve the performance of fully automatic methods on lung tumor segmentation three datasets with different supervision, were acquired. Two of which was annotated with masks, the last with bounding boxes. We call the mask annotated data *strongly labeled*, and the bounding box annotated data *weakly labeled*. We designed three research questions as a foundation to discuss our results and to conclude the project. The first research questions is related to the semi-automatic method, and how its performance can be improved by giving it additional information. The two other research questions are designed to enlighten the effect of the Teacher-Student design both in general, and when the strongly labeled dataset is limited.

Research Questions

All research questions are related to the task of segmenting lung tumors given CT-scans.

RQ1: How does semi-automatic methods, that utilizes either pre-calculated bounding boxes or the center of the tumor as additional input, compare to a fully automatic method that only uses the image as input?

RQ2: By expanding the dataset using a Teacher-Student Framework, could the performance of a fully automatic model be increased compared to a model trained purely on strongly annotated data?

RQ3: When shifting the balance of the dataset towards less strongly annotated data and more weakly annotated data, does this cause the Teacher-Student approach to yield higher performance than the standard fully-supervised approach that only uses the strongly annotated data?

To answer these three research questions, we designed three separate experiments. These three experiments are presented in their respective sections in chapter 4. To address the research questions, observations across the three experiments are discussed.

1.4 Contributions

The primary contribution of this project is the research on the effect of the Teacher-Student Framework used on lung tumor segmentation from CT images. To the best of our knowledge, there exists no published research on the usage of a Teacher-Student Framework on the task of lung tumor segmentation on CT images. We released an open source repository with pre-trained models for anyone to use. We discussed the usage of a semi-automatic method that can aid the experts performing segmentations or even make the process of generating large datasets easier by introducing a teacher.

Contributions

- Researched usage of a Teacher-Student Framework to perform automatic lung tumor segmentation on CT images
- Released an open source repository for automatic lung tumor segmentation, given a CT image, with pre-trained models
- Explored the usage of a semi-supervised method that can aid experts in day-to-day work, or facilitate creation of segmentation annotated datasets in a cheaper way than the current process involving manual segmentation performed by experts
- We intend to publish an article based on the findings in this project in collaboration with NTNU and SINTEF.

1.5 Thesis Outline

Chapter **2 Background & Related Work** contains descriptions of the relevant medical domain and the fundamental theory required to understand the implementation of the method of this project. Brief summaries of published papers related to lung tumor segmentation and the Teacher-Student Framework is covered.

Chapter **3 Methodology** describes, in-depth, the method of the project. This chapter contains information about the datasets used, hardware settings, and a detailed implementation description. Everything necessary to reproduce the results stated in this thesis is available in this chapter.

Chapter **4 Results** contains all the results from the experiments performed.

Chapter **5 Discussion** contains our reflections regarding the results achieved and discussions of observations of the dataset and our method.

Chapter **6 Conclusion & Further Work** concludes this project by answering the research questions and concluding on the overall goal of the project. This chapter also contains some of our thoughts of how to improve performance further and how to investigate the potential of the Teacher-Student Framework further.

Appendix A contains a brief description of MONAI, a framework used frequently in our code.

Appendix B contains an overview of the way we worked during this project. It contains a timeline that illustrates when we conducted the different parts of the thesis and exemplifies some situations where we struggled with challenges or made interesting observations.

Appendix C contains a brief summary of an open source repository we published on Github. The repository is a plug-and-play method for automatic lung tumor segmentation given CT input with pre-trained weights.

CHAPTER 2

BACKGROUND & RELATED WORK

2.1 Medical Background

2.1.1 Lung Anatomy & Physiology

The lungs supply the cells in the human body with oxygen. They are placed behind the ribs and usually weigh between 800 and 1100 grams on an adult human being[2]. On average, a healthy person repeats the breathing cycle 12-16 times per minute. During one inhalation, approximately 500ml of air is inhaled. This air is transferred through the bronchial tree until it reaches small pouches called the alveoli. Figure 2.1 shows how the bronchial tree is structured in a tree-like fashion. In the alveoli, the oxygen is separated from the air and absorbed into the blood vessels. When oxygen is being absorbed, the alveoli dispose of carbon dioxide, which the lungs, in turn, get rid of during exhalation. The process of absorbing oxygen from the air and disposing of carbon dioxide from the blood is called gas exchange. When the oxygen is absorbed into the lungs' blood vessels, it is then transported through the lung artery to the heart. The oxygen is then transported to the rest of the body through the circulatory system. Carbon dioxide is transported from the whole body to the heart, and then through the lung veins into the alveoli to be disposed of through exhalation.

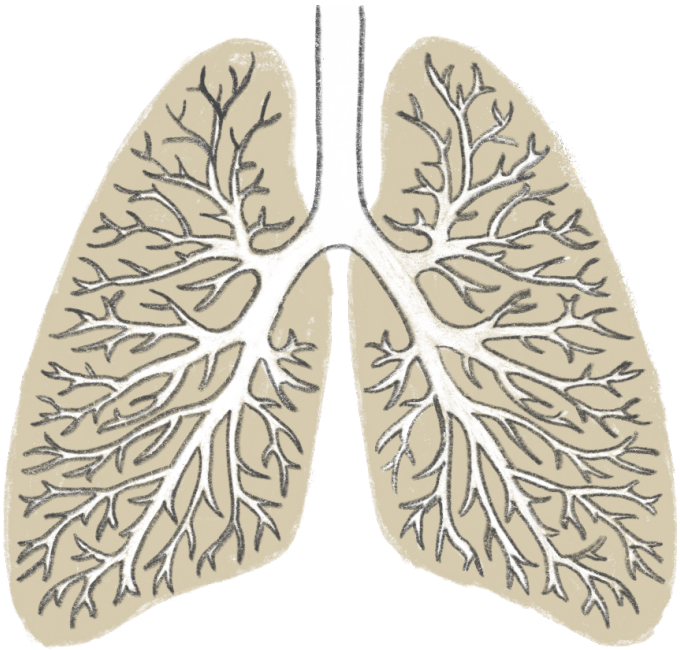


Figure 2.1: Illustration of the Bronchial Tree.

2.1.2 Lung Cancer

Lung cancer is the type of cancer that causes the most deaths in many developed countries, like Norway and the USA [3, 4]. As of 2019, the five-year survival rate for patients diagnosed with lung cancer in Norway is as low as 29 percent for women and 23 percent for men. Late detection is one of the crucial reasons why the prognosis is usually bad. Early detection is vital to improve lung cancer prognostication. Patients that discover cancer at an early stage have higher survival rates [5, 6].

Lung cancer is defined as cancer that *originates* in the lungs [7]. Cancer starts when cells mutate and grow uncontrollably until a tumor forms. When the tumor grows further, it might destroy healthy tissue, resulting in organs failing to function correctly. Particularly in the lungs, the tumor might block parts of the bronchial tree, leaving parts of the lung unable to function, effectively crippling the patient's respiratory system. Tumors that grow uncontrollably are called *malignant tumors*. Parts of the tumor might shed off and be transported to other parts of the body. The breakout cells might continue to grow new cancerous cells, effectively forming a new tumor, called metastases, in another organ.

With modern medical imaging technologies, it is possible to detect tumors visually. Section 2.1.3 will cover some of these methods in more depth. Figure 2.2 show a Computed Tomography (CT) scan of a patient with a lung tumor. As can be seen in the figure, the tumor is visible because it is denser than the rest of the lung and therefore casts a brighter shadow in the CT scan.

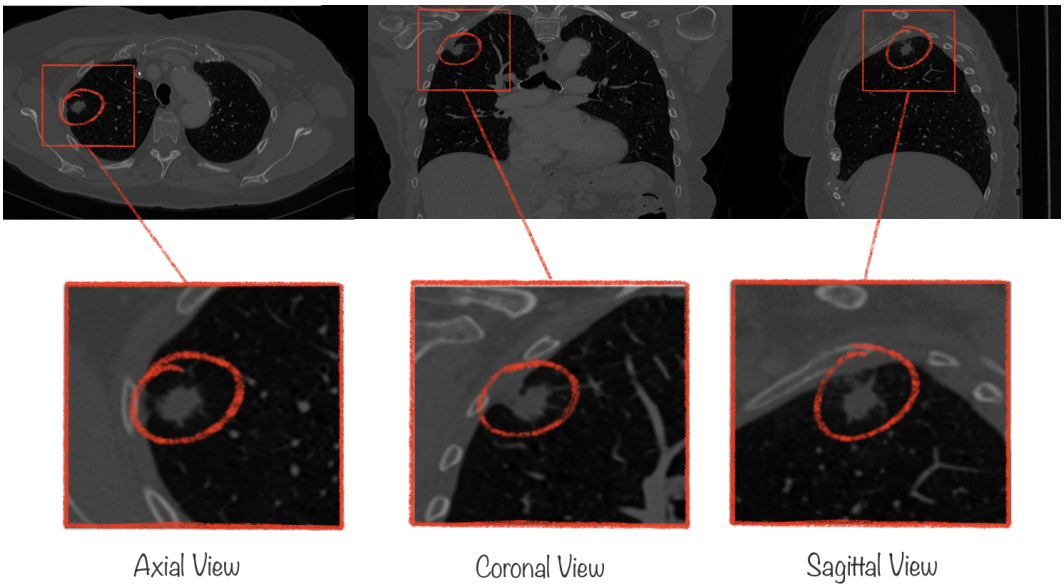


Figure 2.2: CT Scan of a Patient with a Tumor in the Top Right Lung. The image shows three planes extracted from the CT scan in three orthogonal planes. The CT is part of the MSD Lung. dataset [8]. The image is extracted using the ITK-Snap software [9]

Tumors are not the only anomalies in the lung that casts shadows like figure 2.2 show. Infections can create scar tissue inside the lungs that form *nodules*. It can be challenging to distinguish between cancerous nodules (malignant nodules), called tumors, and noncancerous nodules (benign nodules) from image inspection. Diagnosis of cancerous nodules is often made by performing a second scan later in time to observe a change in size or form. The concept of tumor doubling time is often used to diagnose. Cancerous nodules grow uncontrollably while noncancerous often do not. Sometimes noncancerous nodules can grow quicker than a cancerous nodule can. This can also help experts uncover that the relevant nodule was, in fact, not cancerous. It is also possible to perform a *needle biopsy*. A needle is used to extract a small part of the nodule to be later analyzed in the lab. Some medical imaging usually aids this procedure to guide the needle. Needle biopsy is an invasive procedure. If possible, it would be beneficial to diagnose without the need to perform such an invasive procedure. Ideally, it would be possible to perform diagnosis from medical images such as CT images, perhaps with the help of artificial intelligence.

2.1.3 Medical Imaging

Medical imaging is the art of creating visually interpretable images of the inside of the human body. The ability to visually inspect tissue and organs inside the body has been an advantage for medical experts during diagnostics and surgery planning. There exist various imaging techniques, each relying on different physical phenomena. Ultrasound, X-Ray, Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), and CT are among the most commonly used.

X-Ray

X-ray imaging was the state-of-the-art tool to inspect bone structures and tissues inside the patients visually for a long time. X-ray images are created by generating an X-ray beam and aiming it at the part of the body of interest. On the opposite side of the body part being imaged, an absorbing mechanism can absorb the X-rays. Since different tissues inside the body

have different abilities to block X-ray beams, the absorbing module will detect different intensities of X-rays behind each respective part of the body. Bones, for instance, have a higher ability to block the rays than softer tissues like fat or muscle tissue, and therefore the bones will appear as a *shadow* on the other side of the body part. The result is a 2D image that shows the different tissues with different intensities. After some experience, it is possible to recognize different tissues based on the images and even irregular artifacts like fractures or even tumors, popularly used in mammography for detecting breast cancer [10].

Computed Tomography (CT)

Although X-ray imaging is useful for certain tasks, it lacks the precision and granularity for many applications. In 1971, the first CT scan was conducted [11]. CT scans use the same concept as X-ray imaging. X-ray beams are aimed at the body part of interest, and depending on what kind of tissue the beams hit, a certain amount of the beam will be blocked. Unlike traditional X-ray imaging, the X-ray source is not stationary but is rotated around the body part that is being scanned. The receiving module is also rotated around the body, always to be the opposite of the X-ray source. A computer algorithm is used to analyze the beams received by the absorbing module. After a full rotation, a single slice of CT-scan is created. Like a regular 2D image is made up of pixels, 3D images generated from CT scans are made up of voxels. A single slice looks much like a traditional X-ray scan, though a lot sharper and more detailed. By creating many of these slices by slowly moving the whole machine vertically along the patient's length, it is possible to stitch together a 3D image based on all the 2D slices.

Modern CT scans do not usually perform this stop-and-go method for creating 3D scans, however. They perform a helical CT scan. Instead of doing a complete rotation around the patient around the same slice, the CT machine moves along the patient's length while rotating in a spiraling way. One advantage of this is that it is possible to scan larger parts of the body in a shorter amount of time which is of interest in, for instance, lung scans because one can scan the whole lung within one breath. CT machines might have different configurations concerning speed or slice spacing, for example. To be able to reconstruct CT images from different machines and keep the spatial properties within the real world intact, a measure of *voxel spacing* needs to be stored with the CT scan. The voxel spacing holds information about how far apart each voxel is in the real world. There might be different voxel spacings in each axis. Software that can view CT scans or even construct 3D models from CT scans utilize the voxel spacing value to accurately recreate the CT scan captured by the CT scanner.

As mentioned, different tissues have different abilities to block X-ray beams. The intensity that is captured by the absorbing module is quantified. Different tissues have different values associated with them. The common unit to define these values in is the Hounsfield Unit (HU) [12]. Equation (2.1) shows how the Hounsfield Unit is calculated for an arbitrary tissue with an attenuation coefficient μ_x.

$$HU = 1000 \times \frac{\mu_x - \mu_{H_2O}}{\mu_{H_2O}} \tag{2.1}$$

Since water is the prominent substance in the human body, the Hounsfield Unit used water as reference; in other words, a Hounsfield Unit of zero means that the particular part of the CT scan contains water. Tissues that are denser than water will have a positive Hounsfield Unit score. For CT images, HU values is often set to range between -1024 and 1024.

2.2 Artificial Intelligence Background

There exists no clear definition of what Artificial Intelligence (AI) is, and the interpretations are many. AI can be seen as the field that is concerned with algorithms that make intelligent decisions, often in interaction with their environment. The public interest in AI and the media coverage has exploded in recent years, making it one of the hottest topics in computer science for the time being [13]. Given the lack of a clear definition of what AI is, it should come as no surprise that the field is divided into many subfields with different ideologies and concerns in mind. Given the aforementioned broad definition, one can accept that intelligent, adaptive, hand-crafted algorithms will fit into the definition of AI. The sub-field within AI that has emerged lately is the field of machine learning and especially deep learning.

2.2.1 Machine Learning

While hand-crafted algorithms may be part of the AI branch, machine learning is about writing algorithms where the machine learns to solve the task by experience. Traditionally, a programmer would write the rules of an algorithm and let the computer perform the calculations. The programmer would need to learn the rules of the domain before defining the algorithm. However, in machine learning, the idea is that the algorithm should learn the rules itself. In some applications, even humans are not able to precisely define the rules, but machine learning algorithms might be able to find the underlying, complicated patterns. There are many fields within machine learning, but supervised learning is of particular interest for the remainder of this thesis.

Supervised & Unsupervised Learning

In certain machine learning tasks, the correct output is known beforehand. In other tasks, the correct output is not certain. For instance, when classifying (classification is discussed in section 2.2.2) images of cats and dogs, humans can label the expected output associated with each image in advance. On the other hand, if an algorithm is learning to play chess, it is hard to accurately tell how good a given chess move is at any given time. Only the image is necessary to solve an image classification task. However, for solving the chess move problem, knowledge about previous states and potential future moves is required to find the best move. The chess move problem might have multiple solutions, and it is hard to rank them in advance quantitatively.

For tasks where the expected output is known in advance, it is possible to use a *supervised learning* procedure. When the correct output is not known in advance, the problem is within the field of *unsupervised learning*. Supervised learning requires what is called labeled data, which is data where the correct output is attached to it, often called *ground truth*. An example of an unsupervised learning method is clustering which can be applied to data without any ground truth. The clustering algorithm can find similarities or relationships between the elements in the dataset and cluster them accordingly.

Variants of supervised and unsupervised learning also exist. Semi-supervised learning is about combining both partly labeled and unlabeled data. Often labeled datasets are limited in size, as annotating data is tedious and might require an expert in the field. Hence, unlabelled data are often larger and easier to acquire. Supervised learning can be divided into several sub categories. One of these categories is mixed supervised learning, which aims to utilize data with different forms of annotations. Mixed supervision is of particular interest in this thesis.

Figure 2.3 shows where deep supervised learning fits within the field of AI. Deep learning refers to

the idea of using artificial neural networks as part of the solution. Neural networks are explained in more detail in section 2.2.3

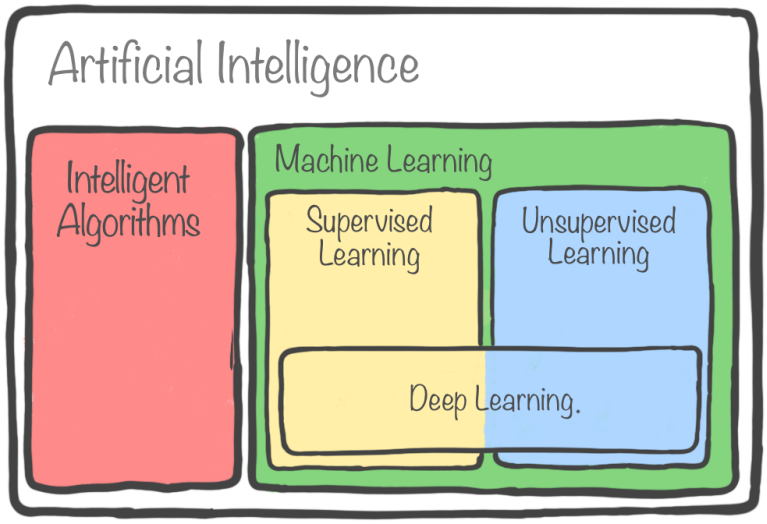


Figure 2.3: The Field of Artificial Intelligence

2.2.2 Computer Vision Tasks

Classification

Classification is the task of correctly assigning an element to a predefined class. Deep learning approaches have proved suitable for many classification tasks in recent years, even outperforming humans [14, 15]. Typically a classification algorithm will take some features associated with the element as input. These features can be predefined parameters or even images, sounds, or time series plots. The remainder of this chapter will focus on image analysis. Image classification commonly revolves around taking an image as input and classifying the content of the given image into predefined classes. For instance, one task might be to feed a 256×256 RGB image of a dog to an algorithm, and make it predict which dog breed was present in a given image. Image classification has its limits. For instance, image classification algorithms will not differentiate between multiple dogs in the same image. If the same algorithm tries to classify an image which contains multiple dog breeds, the algorithm is forced to output a single class, which neglects at least one or more dogs in the image.

Object Detection

While image classification is about correctly assigning an image to a class, in object detection/recognition the goal is to classify and locate the object(s) of interest in an image. Using object detection, one is therefore able to solve the aforementioned task where multiple dog breeds were present in an image. Modern object detection algorithms can locate tens of objects within the same image and classify them into tens or 100s of different classes. Object detections are indicated by placing bounding boxes around the detected objects. Figure 2.4 shows how the output of an algorithm that detects cars and persons might look.

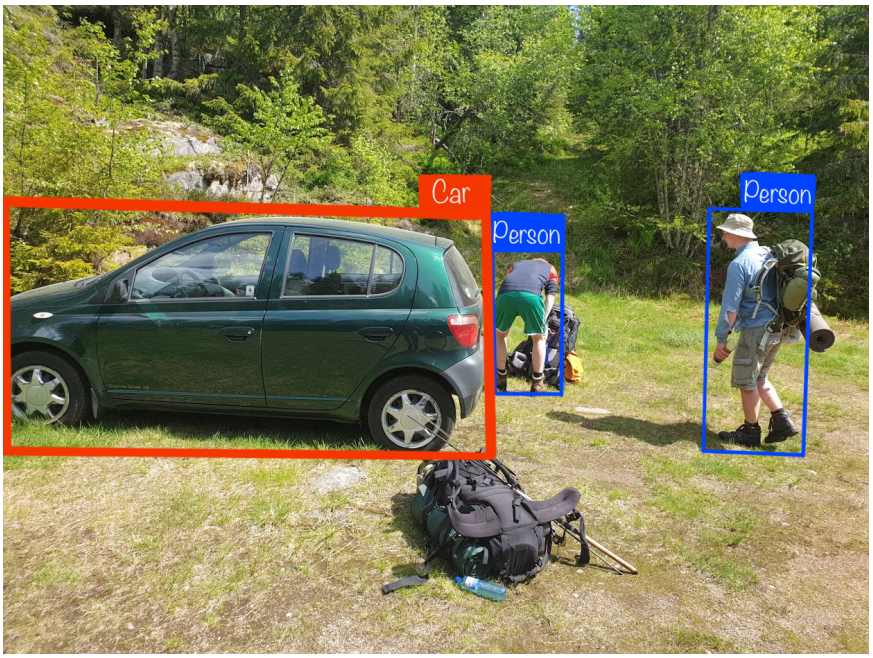


Figure 2.4: Output of an Object Detection Task

Object detection models are evaluated and trained according to how well they place their bounding boxes relative to the ground truth bounding boxes. Different evaluation metrics are used to quantitatively measure how different models perform compared to one another. This is further described in section 2.2.5.

Segmentation

Locating and classifying objects with bounding boxes is sufficient for many applications, however there are applications where a bounding box is not accurate enough. Some applications need to detect all pixels that belong to an object or class. When pixels are being assigned to the relevant classes, it is sometimes referred to as making a *mask* or a *segmentation*.

Image segmentation can be further divided into the two types: *semantic segmentation* and *instance segmentation*. Semantic segmentation aims to assign pixels to certain classes, dependent on whether a class is present. For instance, given an image containing multiple cars and people. The algorithm would assign the same value to all pixels containing cars, and a different value to pixels containing people. However, it does not distinguish between different cars or different persons. Instance segmentation, on the other hand, differentiates between objects within the same class, it produces a set of individual masks for every detection and classify these simultaneously. Figure 2.5 shows the difference between object detection and the two segmentation types.

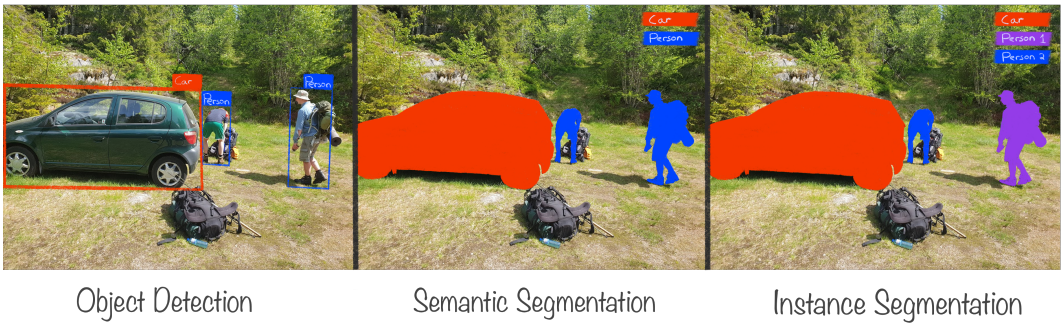


Figure 2.5: Object Detection and two Types of Segmentation

All of the aforementioned tasks have their own applications and domains where they fit in. Instance segmentation is interesting but is unfortunately quite expensive, especially for 3D applications. In this project, like many other medical applications, the semantic segmentation type is the most relevant.

2.2.3 Artificial Neural Networks

The understanding of the human brain heavily inspires artificial neural networks. It takes input from senses, past experiences, and other factors, runs them through biological neural networks, and concludes based on the output of these networks. The basic building block of biological neural networks is the biological neuron, the brain cell. The basic building block of artificial neural networks is the artificial neuron, which is an artificial version of the biological neuron.

The artificial neuron

The simplest neural network is made up of only one neuron. This is commonly referred to as a perceptron. The perceptron takes the weighted sum of the inputs x and bias, resulting in the output z. The perceptron requires as many weights as it has inputs. In practice, this is done by taking the dot product between the transposed weights and the inputs, then adding the bias to this dot product. The bias is a value that allows the neuron to shift its output value up or down. This is useful when the data is not centered around zero. The mathematical expression is shown in equation (2.2).

$$z = \omega^T x + b = \sum_i w_i x_i + b \tag{2.2}$$

One of the biggest limitations of the perceptron, is that it can not classify datasets that are not linearly separable. A linearly separable dataset means that one could separate the classes of the dataset by using a linear function, such as a line in 2D or a plane in 3D.

Activation functions

As the operation of the perceptron is a linear operation, it is possible to use an activation function to introduce non-linearity. In general, an activation function takes input z, which is the weighted sum of the perceptrons input and bias, and then returns another value f(z); the output value of the perceptron. The expression is shown in equation (2.3).

$$a = f(z) \tag{2.3}$$

There are several different activation functions, each with different advantages and disadvantages. Two of the most common activation functions are the rectified linear unit, often called ReLU, and the sigmoid function. These are examples of activation functions that take linear inputs and produce non-linear outputs. Their respective expressions are shown in equation (2.4) and equation (2.5).

$$ReLU(z) = max(0, z) \tag{2.4}$$

$$Sigmoid(z) = \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{e^z + 1} \tag{2.5}$$

A network of neurons

The perceptron is able to solve simple problems, however it struggles with complex problems, thus it offers limited practical utility. However, if one were to combine several of these perceptrons into layers, add activation functions to these layers, then combine these layers into networks, then the resulting network would be able to solve problems that are fairly complex and not linearly separable. These kind of networks are often called *Feed Forward Fully Connected Neural Networks (FF-FCNN)*.

The forward pass

The network produces an output by forwarding the input through all layers. The output of the first layer, a_1, is calculated by taking the dot product between the input matrix x and the transposed weight matrix ω_1, and then adding the bias matrix b_1, to this sum. Then the chosen activation function is applied to each of the elements resulting in the output matrix of the first layer. The expression is shown in equation (2.6), where x is the input to the network, $f_1(z)$ is the activation function applied in the first layer, and b_1 is the bias of the first layer.

$$a^1 = f^1(z^1) = f^1((\omega^1)^T x + b^1) \tag{2.6}$$

The output of the first layer is then used as input for the second layer, the output of the second layer is used as input to the third layer and so on. This is done until the output of the last layer is produced. The generalized expression of layer n is shown in equation (2.7), where $a_0 = x$.

$$a^n = f^n(z^n) = f^n((\omega^n)^T a^{n-1} + b^n) \tag{2.7}$$

Loss functions

The loss is sometimes referred to as *the cost*. The loss of a network is an estimate of how well the network, or model, fits the ground truth of the data [16]. The loss of a network is calculated by a *loss function*.

There exists many loss functions for neural networks. Depending on the problem domain, different loss functions should be used [16]. Two of the most common are the Mean Squared Error (MSE) and Cross Entropy (CE). The expression for MSE is shown in equation (2.8), where Y_i is

the ground truth and $\hat{Y}_i$ is the value produced by the network. The expression for CE is shown in equation (2.9), where $p(x)$ is the ground truth for the class x and $q(x)$ is the predicted value for the class x, by the network.

$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.8}$$

$$\mathcal{L}_{CE} = -\sum_{x \in \mathcal{X}} p(x)\ \log q(x) \tag{2.9}$$

MSE is typically used for regression tasks. It calculates the distance between the ground truth and the prediction and squares it, so the negative and positive errors do not cancel each other out. CE is often used for classification tasks. It can handle both continuous and discrete ground truths. equation (2.9) shows the discrete CE function.

Backpropagation

When an artificial neural network is initialised, all its weights are set to be a pseudo-random value [17]. The initial weights are often distributed according to a mathematical distribution, for instance, normal distribution or Xavier initialization [18]. At first, the output of the artificial neural network will be highly random, as the weights have not been adjusted much. As the weights are being adjusted, the network's output should, in time, approximate the ground truth of the data.

How much the respective weights should change is determined by the gradient of the network. The gradient of a network is a measure of each of the weights' contribution to the total loss. This is determined by calculating the partial derivative of the loss with regard to the weights. As the error of a network is transferred from the first layer through the last layer, the error in layer n+1 could be used to calculate the error in layer n. Because of this, one could say the error *backpropagates* through the network [19]. The first step is to find the value δ^N, which expression is shown in equation (2.10).

$$\delta^N = \frac{\partial C}{\partial a^N} f^{N\prime}(z^N) \tag{2.10}$$

δ^N denotes the delta of the last layer N. The delta of each respective layer n is denoted δ^n and can be calculated by equation (2.11).

$$\delta^n = ((\omega^{n+1})^T \delta^{n+1}) f^{n\prime}(z^n) \tag{2.11}$$

Each delta is used to determine the contribution of the weights and biases of the respective layers' contribution to the total loss. The contribution of weight ω^n_{jk}, which is the weight between the neuron k in layer n-1 and neuron j in layer n, to the loss, is given by equation (2.12). The contribution of the biases can also be calculated by equation (2.13). When the contributions of every weight and bias to the total loss are calculated, they can be adjusted accordingly.

$$\frac{\partial C}{\partial w^n_{kj}} = a^{n-1}_k \delta^n_j \tag{2.12}$$

$$\frac{\partial C}{\partial b_j^n} = \delta_j^n \tag{2.13}$$

Gradient descent

The weights' contributions to the loss are used to adjust the weights, so the loss of the network is lower the next time it receives the same or similar inputs. One way of doing this is to use the gradient descent method. It is an optimization method which attempts to minimize the loss of the network by changing its weights in iterations. As the loss of the network decreases, the adjustment of the weights becomes smaller. The delta rule is an example of a method that uses gradient descent. It requires the output of the previous layer a^{n-1}, a learning rate α, and the delta calculated during backpropagation δ^n. The weights are adjusted according to equation (2.14).

$$\omega_{new}^n = \omega_{old}^n - \alpha \delta^n a^{n-1} \tag{2.14}$$

There are some pitfalls when using the delta rule to update the weights. If the learning rate is too small, it will take a lot of time before seeing any improvement in the network. If the learning rate is too high, the weights are adjusted too much, and the network might not improve.

Another problem with using gradient descent is that the solution might get stuck in a *local minima*. Although there exists an optimal solution, a *global minima*, the model may fail to escape the valley it is in, as it costs too much to escape. This is an example of a greedy optimization. This problem is illustrated in figure 2.6.

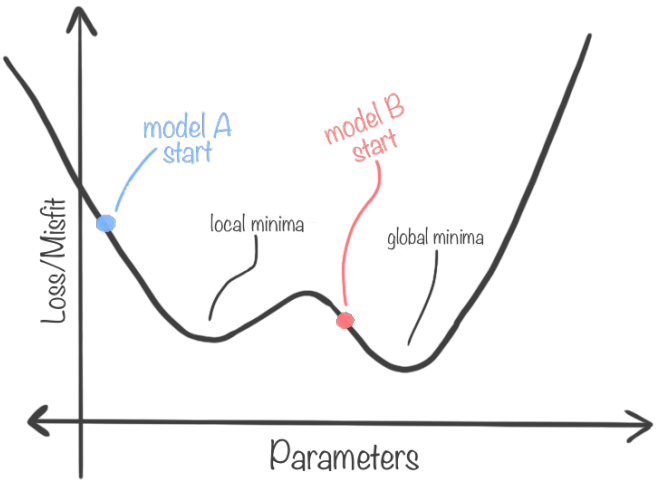


Figure 2.6: Illustration of Local vs Global Minima. In this example, model A may get stuck in the local minima, while model B will reach the global minima. The only difference between the two is the initial weights (starting point).

Optimizers

One way to escape a local minima is to use momentum, denoted ρ, which pushes optimization out of areas where the gradients are low. This is also useful for boosting convergence in low-gradient areas such as *saddle points*. Using momentum, one allows optimization to explore more valleys and thus has a higher probability of finding the optimal solution.

In practice, there are different optimizer that are used to avoid these local minimas and achieve faster convergence towards the global minima. Some optimizers do not use momentum, for example Stochastic Gradient Descent (SGD) [20]. SGD is often quite slow [21], and thus other optimizer that uses some form of momentum, such as ADAptive Momentum Estimation (ADAM) [22], is often used.

Overfitting

Since training data is of limited size, the network's training could, in theory, continue until the total loss is zero, given that the model is sufficiently complex. However, the produced model may generalize poorly when evaluated on unseen data [23]. That is because the network might have learned features that are only relevant for the training dataset, or even memorized the training set. This problem is known as the *overfitting problem*.

One way of solving the overfitting problem is to divide the training set into two parts: a training set and a validation set. The validation set is used to determine when the model starts overfitting. This is done by calculating the loss of the validation set periodically during training. This loss is then compared to the previously calculated validation losses. If the new loss is lower than the previous ones, then the training continues. If the most recent loss is higher than the previous, this can be interpreted as the start of overfitting, and weight adjustments should stop. In practice, the model is not stopped until several validation steps have shown higher loss than the validation loss at a given point. This method is known as *early stopping*. Figure 2.7 illustrates when to apply early stopping.

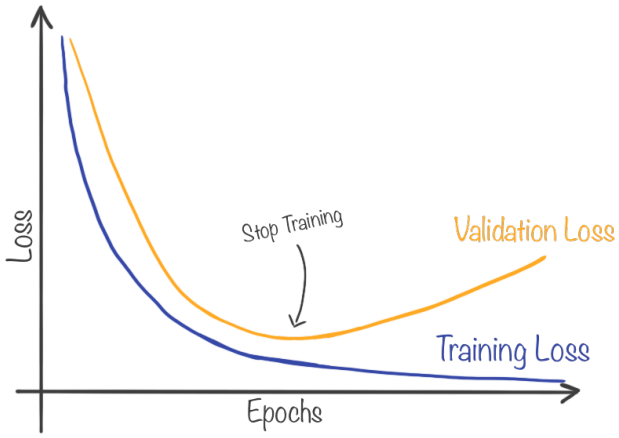


Figure 2.7: Illustration of When to Apply Early Stopping.

Another way of avoiding the overfitting problem is to use regularization. Regularization can be used in addition to having a training set and a validation set with early stopping. The basic idea of regularization is to simplify the model, thus making the model less likely to overfit the training data. Two common regularization techniques are L1 regularization and L2 regularization. L1 regularization works by pushing the weights of the network towards zero by a constant factor, λ, also called the regularization parameter [24]. The process of pushing the weights toward zero can be thought of as an effort to reduce the number of features in a network and thus make the network focus more on the essential features. L2 regularization works similarly. The difference is that while L1 regularization reduces the network's weights toward zero by a constant factor λ, L2 regularization reduces the network's weights towards zero based on the value of the weight

multiplied by a constant factor, λ. The effect of L2 regularization is much the same as the effect of L1 regularization, and the difference is that L1 pushed weights towards zero to get rid of worthless features while L2 strives to keep weights small in general.

A third option to avoid overfitting is *dropout*. Dropout is based on the idea of randomly dropping specific nodes in the network, setting their output to zero[25]. This way, new routes in the network are created to solve the task, rather than reinforcing the existing paths. This makes it harder for the network to memorize the training data and thus reduce overfitting.

2.2.4 Convolutional Neural Networks

Although traditional neural networks are great at certain tasks, they also have their limits. When working with images, one may use the pixels as input to a neural network. However, when high-resolution images, such as 4k images, are used, the size of the fully connected layers increase rapidly. Attempts to encode the image and guide the neural network in image analysis, were therefore commonly performed. This was done using algorithms that extracted a selection of predefined or hand-crafted features from images. This reduced the complexity of the network, but required a lot of effort designing suitable feature extractors.

Although this strategy yields transparent and interpretable results, it is often challenging for humans to find the most suitable features for a specific task. The main idea behind Convolutional Neural Networks (CNNs) is to automatically learn the feature filters relevant to solve the task. All of the operations described in this section can be performed on images or arrays of any dimension. All examples are for 2D images, but any hyperdimensional array can be convolved, transposed, or pooled.

The convolution operation

The key operation in a CNN is the convolutional operation performed by the feature filters. A convolutional layer takes an image with resolution $W \times H$ and depth N as input. In each convolutional layer, several different kernels are convoluted over the input image. The kernel can have arbitrary width and height but has the same depth as the input image. Each kernel is matrix multiplied over the image in a sliding window fashion according to the specified stride. Figure 2.8 illustrates how a filter is applied to an input image in a sliding window fashion. During training, the filters evolve to learn different patterns associated with the different classes. Hence the machine learns the features rather than programmers handcrafting them.

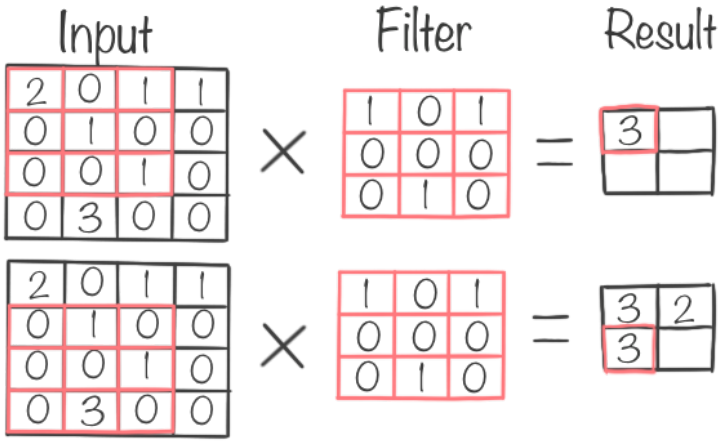


Figure 2.8: Illustration of a 2D Convolution. Input image is of size 4×4 with a single channel. The filter has a kernel size of 3×3. The operation is carried out with stride of one which means that the kernel is moved with one step for each convolutional operation. There is no padding, therefor the filter applied 4 operations to the image resulting in a 2×2 output image.

The pooling operation

Another regular operation performed in CNNs is pooling. These are non-learnable filters that aim to reduce the image size and at the same time keeping some essential information. There are different types of pooling. Max-pooling and average-pooling are among the most common.

A max-pooling operation is simply an action that selects the maximum value within the pooling filter. The max-pooling operation is visualized in figure 2.9. Average-pooling, as the name suggests, computes the average of the values within it as output. The result of an example average-pooling can be seen in figure 2.10.

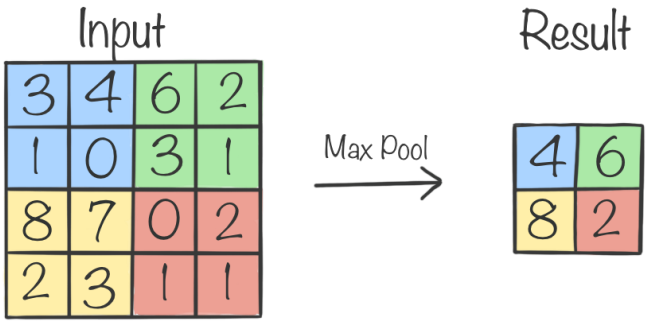


Figure 2.9: Max Pooling Example

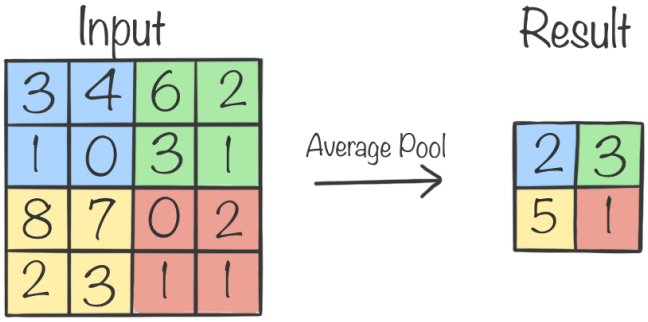


Figure 2.10: Average Pooling Example

The transpose convolution operation

As pooling operations scale images or features maps down, it makes sense to have operations that can scale images up. There are several applications where this is useful, and it's not limited to tasks of enhancing image resolution. Traditionally up-scaling has been done with algorithms such as nearest neighbor interpolation or bilinear interpolation and similar algorithms. However, it makes sense to train the up-scaling filters the same way one trains the convolutional filters. Transposed convolutional filters are trainable filters that scale images up. Like the other convolutional operations, matrix multiplication is at the heart of the transposed convolution. Figure 2.11 shows how the transposed convolution is performed conceptually. In practice, the operation is performed with efficient matrix multiplication.

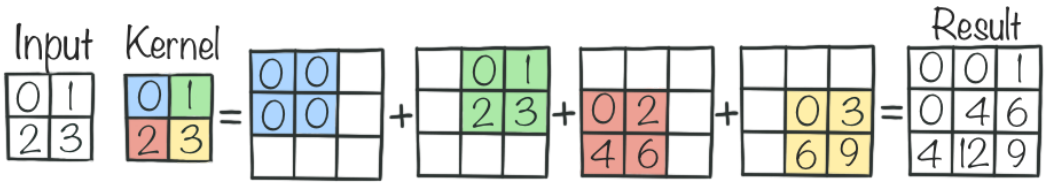


Figure 2.11: Transposed Convolution Example. The figure shows conceptually how a transposed convolutional kernel can scale an image from a resolution of 2×2 to 3×3.

2.2.5 Evaluation Metrics

To be able to compare the performance of different algorithms and architectures to each other and human performance, quantitative metrics are necessary. Several metrics are commonly used, and each metric emphasizes different aspects of performance. For instance, in some applications, it is more important that the algorithms detect all positives, while in others, it is more important that the false positives are reduced.

Positives and Negatives

Many common metrics use the notion of positives and negatives in the calculation. One can divide all classifications performed by an algorithm into four categories. *True positives* are the occurrences where the algorithm correctly predicted that an element belongs to the class. *True negatives* are the occurrences where the algorithm correctly predicted that an element did not belong to the class. When the algorithm predicts that an element belongs to the class and it does not, it is a false positive. Likewise, when the algorithm predicts that an element does not belong

to the class, but it does, this is a false negative. Figure 2.12 illustrates the concepts of positives and negatives. The positives and negatives are usually abbreviated as stated in table 2.1.

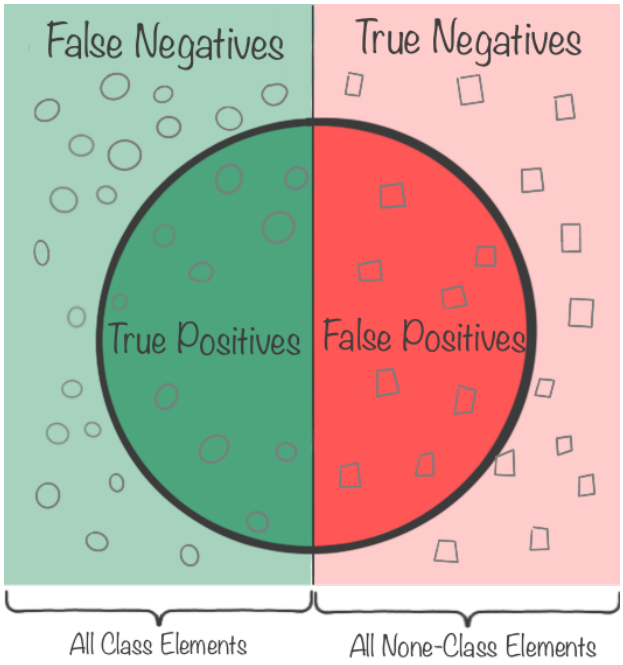


Figure 2.12: Negatives and Positives. The figure shows all elements within a domain where roughly half of the elements is part of the class and the others are not. Elements within the circle are the elements that an algorithm has predicted is part of the class.

Table 2.1: Overview of Positives and Negatives Abbreviations

Term	**Abbreviation**	**Description**
True Positive	TP	Elements that are correctly classified to be part of the given class.
True Negative	TN	Elements that are correctly classified to not be part of the given class.
False Positive	FP	Elements that are predicted to be part of given class, but is not.
False Negative	FN	Elements that are not predicted to be part of given class, but is.

Precision and Recall

As mentioned previously, different applications will emphasize different aspects concerning performance measures. Two of the most fundamental metrics used when dealing with classification, detection or even segmentation, are *precision* and *recall*.

The precision metric measures how likely it is that an element, such as a voxel, that the algorithm predicted was in the class is actually in the class. In other words, how many of the elements that the algorithm predicted were in the class is actually in the class. Equation (2.15) shows how precision is calculated.

Recall measures how many of the elements belonging to the class the algorithm is capable of predicting.

$$Precision = \frac{TP}{TP + FP} \tag{2.15}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.16}$$

It is beneficial to have a high score of both precision and recall, but to a certain degree, one has to choose which is more important. It is common to plot a precision-recall graph showing how one metric's score influences the other. For instance, if one lowers the threshold for how confident an algorithm should be before an element is considered part of the class, this will likely increase the recall score but lower the precision score. This is exactly what the precision-recall graph in figure 2.13 shows, and depending on the application, one has to choose the best trade-off between the two.

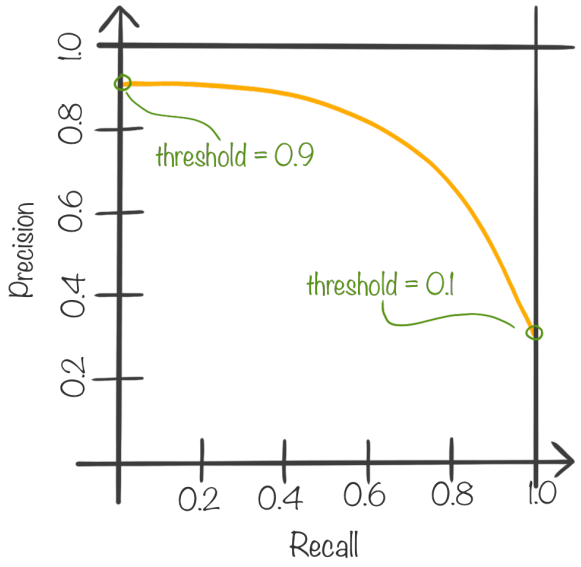


Figure 2.13: Illustration of a Precision-Recall Graph. The illustration show how the precision and recall is affected when adjusting the confidence threshold of an algorithm.

Intersection over Union

Especially in object detection, the Intersection over Union (IoU) metric is commonly used. Since object detection is about correctly placing a bounding box around an object and predict the class, the notion of intersections and unions is a good fit. A high score of IoU means that the predicted bounding boxes closely resemble the ground truth bounding boxes and that the algorithm accurately classifies the objects. IoU is calculated by taking all elements that are present in both the prediction and ground truth divided by all elements present in either prediction or ground truth. Equation (2.17) shows the calculation, where P is the predicted bounding box, and T is the ground truth bounding box. Equation (2.18) shows the calculation based on positives and negatives. If the algorithm is too greedy and predicts too many elements to be part of the class, then the union will be bigger, and the IoU score will suffer. Likewise, if the algorithm is too discriminating, then the intersection will be smaller, and the IoU score will suffer.

$$IoU = \frac{|P \cap T|}{|P \cup T|} \tag{2.17}$$

$$IoU = \frac{TP}{TP + FP + FN} \tag{2.18}$$

Dice Similarity Coefficient

While object detection deals with bounding boxes, object segmentation is more detailed. IoU could be used as a metric for segmentation tasks as well, but a more commonly used metric is the Dice Similarity Coefficient (DSC). DSC is rewarding true positives more than what the IoU metric does. Though DSC weights true positives more than IoU, they are positively correlated. An algorithm that scores better than another according to IoU will also do this according to DSC, unlike the precision vs. recall metrics, for instance. The set notation for calculating the DSC is shown in equation (2.19). DSC can be calculated as a function of negatives and positives as shown in equation (2.20).

$$DSC = \frac{2 \cdot |P \cap T|}{|P| + |T|} \tag{2.19}$$

$$DSC = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.20}$$

2.3 Advanced Techniques

In the previous sections, the basics of neural networks and how they work were presented. To achieve good results on advanced medical segmenting tasks, additional features on top of these basic theories are necessary. This section aims to describe concepts and methods that can be considered a bit more advanced and used in implementing this project.

2.3.1 Mixed Supervision

As previously mentioned, the labels needed during the training of segmentation tasks are expensive to make. This applies to all types of segmentation tasks. However, this effect is possibly more substantial for medical segmentation tasks. Only trained experts can label them accurately. In addition, strict privacy rules often apply to medical data.

The principal idea behind a mixed supervision strategy is to use data with different types of annotations. In deep learning, more data often leads to better models. By utilizing datasets with different types of annotations, one is not restricted to the data with the annotations that fit the exact output type the model will produce.

Often one type of annotation is more fine granular than another. The more detailed annotation type is called *strongly labeled*, and the less detailed annotation type is called *weakly labeled*. For instance, there might exist two datasets where dogs are annotated in images. One of the datasets might have segmentation annotations, while the other has bounding box annotations. In this example, the dataset with segmentation annotations would be the strongly labeled data, and the bounding box dataset would be the weakly labeled data.

There are multiple ways to take advantage of datasets with multiple annotation types. The one method relevant for this thesis is to use the weakly annotated data to make more strong

pseudo-labels. A model is trained to create strong labels from the weak, then the expanded dataset is used to train a new model. This concept is called a Teacher-Student design.

2.3.2 Teacher-Student Framework

When the main task is to produce output that corresponds to strongly annotated data, it is tempting to produce strong pseudo labels from weakly annotated data. In the Teacher-Student Framework, a *teacher* is trained using the strongly annotated data to infer strong labels on the weakly labeled data. The clever part is that the teacher can cheat by using the annotations of the weakly labeled data as an additional input when it performs inference.

For instance, if two datasets are available, one with segmentations and one with bounding boxes, it is trivial to calculate the bounding boxes from the segmentations. A bounding box is simply the closest fitting rectangle around the segmentation mask. A teacher can then take the bounding box as input in addition to the image and learn to create segmentation outputs. Since the teacher has the additional information from the bounding boxes, the idea is that the task of creating a good mask is more effortless than if the teacher was only given the image. The complexity of the task has been reduced. The teacher can then learn from the strongly annotated dataset how to create masks given image and bounding boxes as input. When the teacher has learned to utilize this extra information, it can hopefully create decent masks on the weakly labeled dataset by taking the bounding boxes as additional input.

By using the teacher to infer masks on the bounding box annotated dataset, the available segmentation annotated data has been increased. This expanded dataset can then be used to train a fully automatic model. The idea is that even though the strong pseudo-labels are probably not as good as human-labeled data, the effect of a larger dataset makes the model trained on it better. The model that is trained on the strongly labeled dataset, and the expanded dataset, is called the *student*.

2.3.3 LeakyReLU & PReLU Activation

Leaky ReLU is an activation function closely related to ReLU. Leaky ReLU outputs the input value directly if the value is greater than zero, like ordinary ReLU. Unlike ordinary ReLU, it allows for values less than zero, but they are multiplied with a predefined value to dampen the amplitude. Equation (2.21) shows the Leaky ReLU function. The equation takes an input parameter x and has a predefined slope constant s. s is usually a small number, below 1.0.

$$LeakyReLU(x) = \begin{cases} x, x \geq 0 \\ s \cdot x, x < 0 \end{cases} \tag{2.21}$$

A second related activation function is named PReLU, short for Parametric Rectified Linear Unit. This function behaves exactly like Leaky ReLU, but the key difference is that the slope value that is a constant, predefined value in Leaky ReLU, is a learnable parameter in PReLU.

2.3.4 Dice Loss

Loss functions are an essential part of neural network frameworks. MSE and Cross-Entropy Loss that were discussed previously has proved to have poor performance when it comes to tasks where there is great class imbalance. Class imbalance means that some classes in a dataset

are over-represented or under-represented. This problem often arises in medical tasks where the input can be vast, but only a tiny fraction of it is relevant to the task. For instance, if a model takes CT images of the body as input and produces a mask of the liver as output, then the number of voxels that are part of the liver class is tiny compared to all the voxels that are not. In this case, it is possible that the network would quickly learn that it would receive a low loss from concluding that *no voxels are part of the desired object class* no matter the input.

Multiple strategies have been tried to combat the class imbalance problem, including the development of specialized loss functions. Dice loss is among the most popular loss functions commonly used in medical computer vision tasks. Dice loss is based on DSC. DSC can output values between 0.0 (bad prediction) and 1.0 (perfect prediction). Therefore it makes sense to subtract this sum from 1.0 to form a loss function. This way, when the model is trying to optimize the loss function towards zero, it is, in fact, optimizing DSC towards 1.0. DSC equation shown in equation (2.20) is based on a binary GT mask and a binary output. It is possible to generalize the method to accept continuous values for both the mask and the output of the network. Equation (2.22) shows this generalized dice loss equation. In the equation, P is the prediction matrix, T is the ground truth matrix, and the multiplication between them is element-wise multiplication. The absolute symbols around each matrix respectively denote the sum of all elements within the matrix.

$$\mathcal{L}_{DL} \approx 1 - DSC \approx 1 - \frac{2 \cdot (P \cdot T) + \epsilon}{|P| + |T| + \epsilon} \tag{2.22}$$

An epsilon is added to both the denominator and numerator. This is done to avoid zero division. The derivable dice loss shown in equation (2.22) is sometimes called *soft dice loss* because of this smoothing. Since the only way the model can move away from a 1.0 loss is to hit true positives, it is forced to find the class elements, even though the dataset is highly imbalanced. The intersection is valued two times the amount of the sum in the denominator; in other words, dice loss punishes false negatives harder than false positives, which is often desirable in the field of medical AI.

2.3.5 Batching & Accumulating Gradients

Quite often, it is not possible to hold the entire dataset in memory during training. Hence, dividing the dataset into *batches* is often performed. Luckily, this also works as a natural regularizer, as the network is only able to see a subset of the data for each update, and therefore can not overfit as easily. Using larger batches has the benefit of improved stability and convergence, as the network is able to base its updates on a larger set of samples. Much research has been done to determine the importance of batch sizes and estimate the best batch sizes for different tasks. Some work concludes that small batch sizes offer more stability, while others conclude the opposite [26], but in general a batch size between 2-32 is often suitable. Most research concludes that some batching is advantageous.

The improvement in training speed comes from the fact that one can run a whole batch through the network at the same time, given that the GPU has enough VRAM to keep the whole batch in memory. For instance, if a network accepts inputs with a dimension of 10×10, then one could batch 100 of these inputs into a matrix of dimension $100 \times 10 \times 10$. The network would, in this case, produce 100 outputs, for instance resulting in a 100×1 output vector if the hypothetical network was a binary classifier. Less frequently calculating loss and updating weights also speeds up the training since it is only done once per batch.

VRAM capacity is a constraint that can limit the ability to use larger batch sizes. If the GPU does not have the capacity to hold a batch within memory, it is still possible to achieve the effect of stability through *accumulating gradients*. As the name implies, this technique does not adjust the weights before a batch of gradients has been accumulated. While most of the speed advantage of batching is lost, the stability of the weight adjustments is still utilized with this technique. One can send a single input through the network at a time, then aggregate the gradients, and finally, when a batch of gradients has been accumulated, the model's weights are adjusted. Doing so, it is possible to use an infinite batch size, as one simply iteratively sum the scaled gradients for each mini-batch, and thus avoid storing these in memory.

2.3.6 Batch Normalization & Instance Normalization

Normalization in AI is the process of standardizing the input/output of a layer in a neural network [27]. One form of normalization is the process of subtracting the mean, μ, and dividing by the standard deviation, σ, like equation (2.23) describes [28]. This ensure that the values of $\hat{x}$ is centered around zero, with a standard deviation of one.

$$\hat{x} = \frac{x - \mu}{\sigma} \tag{2.23}$$

In batch normalization, each element is normalized with respect to the elements in the whole batch. One form of batch normalization is the standardization shown in equation (2.23), where the mean and standard deviation is calculated from the whole batch. Instance normalization on the other hand, normalizes the element with respect to the values within that element. For instance, if an image is to be instance normalized, the pixel values within that image is used to normalize the image, rather than all the pixel values in the entire batch.

2.3.7 U-Net

U-Net is a commonly used architecture based on the idea to use fully convolutional layers and to have an encoder block, followed by a decoder block, with a connection between them, often called a bridge [29]. There is no strict border between the encoder and the decoder block; however, the encoder in a normal U-Net is where the number of filters are increased and the resolution of the image is decreased. In the decoder block, the number of filters is decreased, while the resolution is increased, back to the resolution of the original image. An example of a 3D U-Net can be seen in figure 2.14.

Between the encoder and the decoder block, there are several connections, often called *skip connections*. The primary purpose of these connections is to preserve the spatial information of the image. The bottom part of the network is often referred to as the bridge. This is where the image resolution is at its lowest, and the number of filters is at its highest. The encoder encodes the spatial information into a dense feature space that is interpreted by the filters. This dense feature space is then decoded back to the spatial dimensions in the decoder.

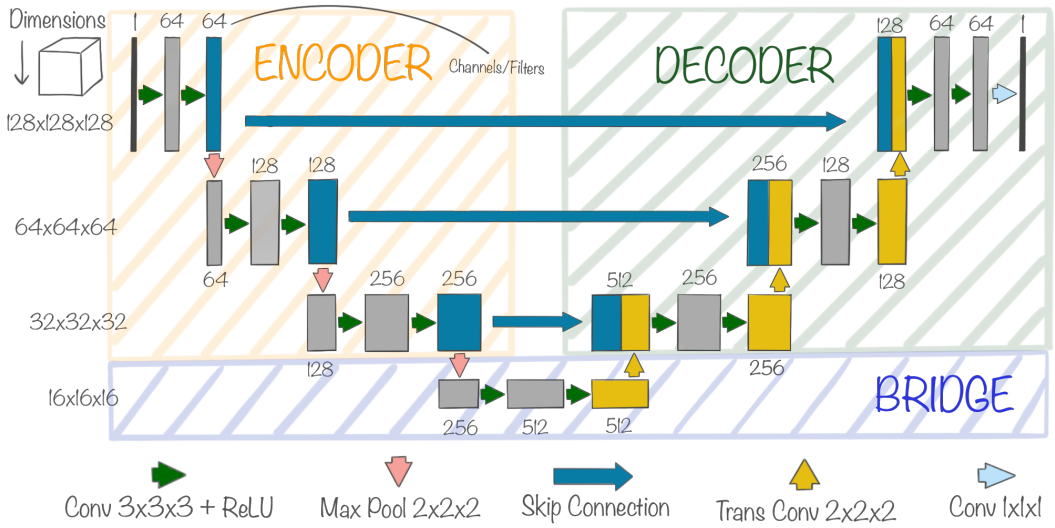


Figure 2.14: U-Net, an Encoder-Decoder Architecture. Example of a 3D U-Net architecture with depth of four. This example shows how an image with resolution $128 \times 128 \times 128$ is incrementally scaled down through the encoder and then scaled up again through the decoder.

U-Net is often used as a backbone for other architectures. It is possible to extend the architecture with branches or various forms of skip connections to be better suited for the specific task. It can be used as a backbone for tasks ranging from classification to detection or segmentation.

2.4 Related Work

2.4.1 Lung Tumor Segmentation

Ray proposed a 2D fully convolutional neural network for semi-automatic segmentation of lung tumors in 2016 [30]. The method assumed predefined bounding boxes of 100×100 pixels in each slice, with goal to reduce annotation workload for clinicians. The architecture achieved a DSC of 0.83 on an in-house dataset consisting of CT scans of 107 patients.

Isensee et al. [31] proposed a two-step method for semantic segmentation of medical volumetric data in 2018. The design involved two separate 3D U-Net models that both performed semantic segmentation. In the first step, a downsampled version of the full volume was used as input, scaled down to fit into GPU memory. The predicted segmentation was then used along with the image as input to the second model. The second model is applied on full resolution in a sliding window fashion across the volume using a 3D patch. They achieved a DSC of 0.69 on the MSD dataset, and also came second on the 2020 BraTS challenge for glioma segmentation [32, 33, 34].

Kamal et al. proposed the novel architecture *Recurrent 3D-DenseUNet* in 2018 that achieved an average DSC of 0.7228 on the NSCLC-Radiomics dataset [35]. They used the popular 3D U-Net as the backbone of the architecture, modified to contain convolutional LSTM-blocks in the bridge for interslice context. The CT-images were scaled from 512×512 to 256×256 in the axial plane. Each input to their network consisted of eight slices, resulting in a $256 \times 256 \times 8$ input tensor. To solve the problem of class imbalance in 3D patches, they proposed to neglect non-tumor patches during training.

Carvalho et al. achieved a DSC of 0.709 on the MSD dataset [36]. They proposed a two-step

method in 2019. First, the tumor is located by a 2.5D object detection algorithm. Then the region proposed by the detector is segmented by a 2D U-Net model. The detector and the segmentation model were trained individually.

Pang et al. achieved an average DSC of 0.7767 on the NSCLC-Radiomics dataset in 2019 [37]. They leveraged the strength of the encoder-decoder pattern in 3D U-Net as part of their segmentation pipeline. To overcome the issue of class imbalance, they proposed a weighting mechanism in the loss function that adaptively adjusted the loss function according to how imbalanced a given batch was (tumor vs. non-tumor elements). The loss function was based on the popular CE loss function. The same batch-weighted CE has been used when segmenting other structures as well, as Bouget et al. showed [38].

Hansen et al. implemented a supervoxel algorithm to segment lung tumors in PET/MRI images in 2020 [39]. The method utilized traditional machine learning techniques, such as clustering. Multiple individual voxels are clustered into supervoxels, and supervoxels can be clustered across multiple patients. They achieve a DSC of 0.47 on a non-public dataset containing 18 PET/MRI scans.

2.4.2 Mixed-Supervision & Teacher-Student Framework

Mlynarski et al. trained a convolutional neural network, based on the U-Net architecture, jointly on image segmentation labels and image class labels in a multi-task learning fashion to segment brain tumors in 2019 [40]. Their proposed architecture includes an additional branch in the U-Net architecture that performs image-level classification. Image class labels are less expensive than costly segmentation labels. They demonstrated the performance on the BatTS 2018 challenge dataset, proving that their model that utilized mixed-supervision labeling outperformed the fully supervised model [32, 33, 34].

Sun et al. used a Teacher-Student Framework to utilize multiple datasets with different types of annotations to perform liver tumor segmentation in 2020 [41]. They divided their data into two subsets: a strong subset and a weak subset. The strong subset contained detailed semantic annotations, while the weak subset contained bounding boxes. By creating bounding boxes from the strong labels, they trained a teacher network that used the bounding boxes as input in addition to the image. The goal of the teacher-network was to create accurate segmentations. A second model, called the student, was then trained using the strong dataset and the pseudo annotated dataset created by the teacher-network of the weak dataset. They demonstrated the performance of the network on the LiTS-challenge dataset, outperforming the current SOTA results [42].

Xie et al. demonstrated the use of self-training in a Teacher-Student design for image classification in 2020 [43]. They called the method *NoisyStudent* because they introduce noise to the input using dropout and various data augmentation during training of the student. They trained the teacher on annotated data and then applied it to generate pseudo-annotations on the unlabeled data to train the student. When the student was trained, it was used as the teacher for a new student. This was done in an iterative process, generating increasingly better teachers and students. They demonstrated the performance on the ImageNet 2012 ILSVRC dataset, which contains more than 1000 different classes [44]. They used images from the JFT dataset containing over 300 million unlabeled images. They reported a significantly better top-1 result than the current SOTA results on the 2D ImageNet classification task.

Gadgil et al. demonstrated the use of expert annotations and AI generated labels in a semi-

supervised fashion to perform medical semantic segmentation of chest X-ray images in 2021 [45]. The task was to segment parts of the X-ray into ten categories of pathologies. They used 200 expert labeled images and over 220000 image-level labels generated by algorithms to train their model. They showed that by combining the expert labels with the vast generated dataset, the mean IoU was improved by over 13 percent compared with a fully supervised method.

CHAPTER 3

METHODOLOGY

Our method is described in detail throughout this chapter, but it can be summarized in six steps. Each step is indicated with its corresponding number in figure 3.1.

1. **Format the Datasets**
   Format all images in the datasets to the NIfTI format and place them in a streamlined folder structure. Create weak labels from the strong ones; in our case, make bounding boxes from segmentations. This step is described in section 3.1.

2. **Data Preprocessing**
   Preprocess the data for training. In this project, we used two different preprocessing pipelines; one specifically for the teacher model, and one for the student model. Both pipelines are described in detail in section 3.2.

3. **Train a Teacher Annotator**
   Train a model that can utilize weak labels to generate segmentations from a dataset of bounding boxes. Section 3.3 presents the teacher, its architecture, and how it was trained.

4. **Expand the Dataset**
   Use the teacher from step three to expand the strong dataset by creating segmentations on the dataset containing bounding boxes. This part is presented in section 3.4.

5. **Train a Student Model**
   Utilize the expanded dataset to train a fully automatic student model that can perform end-to-end lung tumor segmentation from CT-image input. The student architecture and training procedure are documented in section 3.5

6. **Perform Inference**
   The final model has been trained, and the pipeline for fully automatic lung tumor segmentation from CT-images is now ready for use. Appendix C presents our publicly available GitHub repository[1], containing source code, trained models, and a simple way to deploy the pipeline on new data.

Section 3.6 presents an experiment conducted to answer research question three. All the steps in the overall method are performed in this experiment. Hence, to fully understand its process,

[1] https://github.com/VemundFredriksen/LungTumorMask

it is advised to read the preceding sections first. Section 3.7 briefly presents the post-processing method that can be used on the output of the models. For training the models used in this project, we had access to multiple powerful machines. All available hardware is documented in section 3.8

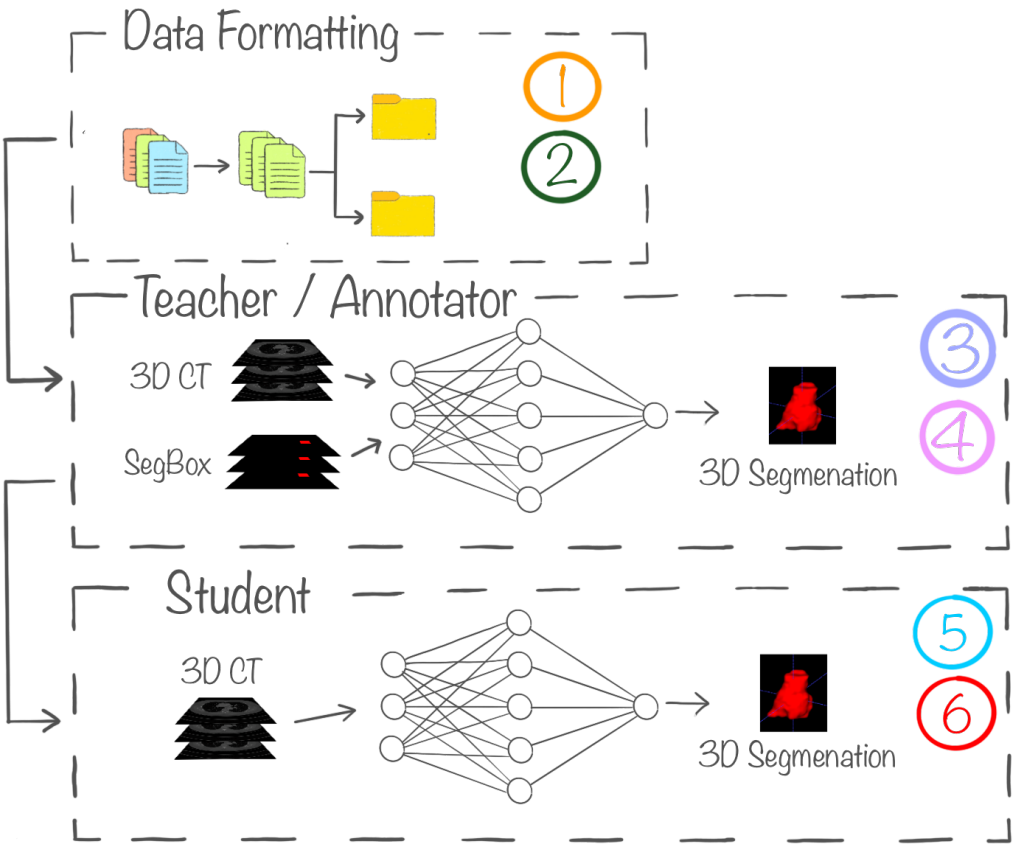


Figure 3.1: Method Overview

3.1 Datasets & Data Formatting

This section covers essential information about the three datasets used in the project and a brief comparison between them. It also covers step one in the overall method; formatting the datasets to a streamlined structure and creating weak labels from strong labels.

The available datasets for mixed-supervision tasks are often stored in various formats. To streamline the training process, all datasets were formatted to fit a predefined structure. All images and segmentation labels were stored as NIfTI files, as NIfTI files are easy to handle, and stores image information and spatial information, such as orientiation and voxel spacing, in a single file. Each stored image was compressed using the gzip algorithm[2]. *Gzip* is a lossless compression algorithm, of which all data loaders in this project accept natively. In addition to file types, folder structures vary between datasets. The datasets were stored in a common folder structure. All input images, segmentations, and bounding boxes were stored with the same name in their respective directories */Images*, */Labels*, and */Boxes*.

[2]https://www.gzip.org/

MSD Lung Dataset

The Medical Segmentation Decathlon (MSD) dataset was released as part of a crowd-sourced online challenge [8]. The goal of the challenge was to facilitate the development of semantic segmentation algorithms. The dataset contains ten subsets, each with annotations of different organs. In this thesis, subset six, *Task06_Lung*, was used. This subset contains 63 CT scans with annotated lung tumors. The tumors were annotated by one experienced radiologist using the software OsiriX [46]. Both input images and annotations were stored in the NIfTI format.

NSCLC-Radiomics Dataset

NSCLC-Radiomics is an open dataset containing 422 CT-scans of patients diagnosed with non-small cell lung cancer [47, 48]. It is acquired through The Cancer Imaging Archive (TCIA) [49]. All images were stored in the DICOM format. Each image in the dataset contains annotations of multiple organs, including lungs, spine, and the primary lung tumor of the patient. Annotations were created manually by a radiation oncologist and stored as DICOM-RTSTRUCT files. The RSTRUCT-files contain extracted surfaces of the tumor annotations, stored as a polygonal mesh, including sets of vertices and a set of triangular faces.

Lung-PET-CT-Dx Dataset

A Large-Scale CT and PET/CT Dataset for Lung Cancer Diagnosis (Lung-PET-CT-Dx) is a collection of 1295 PET, CT, and fused PET/CT scans, distributed over 355 unique patients [50]. All patients were diagnosed with one of the four lung cancer types: Adenocarcinoma, Small Cell Carcinoma, Large Cell Carcinoma, or Squamous Cell Carcinoma. The dataset also contains annotations that mark the location of the tumors as a bounding box in each axial slice of the volume. The annotations were performed by five experienced radiologists. Each annotation was performed by one of the radiologists and verified by the four others. The annotations were stored as separate XML files, and the images were stored as DICOM series.

Dataset Comparison

All three datasets contain annotations indicating the location of lung tumor in CT images. NSCLC-Radiomics and MSD-Lung contain semantic annotations, and LungDx contains bounding boxes in the axial planes. The three datasets differ in size. MSD-Lung includes 63 annotated tumors, NSCLC-Radiomics contains 422 annotated tumors, and LungDx comprises of a total of 1295 CT, PET and/or fused PET/CT scans. After the formatting and preprocessing of the LungDx dataset, the remaining images from this dataset were reduced to 650 images, as only CT images were considered.

The tumors vary in size and shape across the three datasets. Table 3.1 shows the average tumor size and standard deviation of each dataset, respectively. The sizes were computed from the segmentation annotations. The LungDx dataset did not contain segmentation annotations, and the volumes were therefore computed from the segmentations produced by the teacher. The diameter was estimated by computing the average minor and major axis length of the respective tumors, using the python library scikit-image [51].

Table 3.1: Tumor Sizes in the Datasets

Dataset	**Average Tumor Volume**	**Average Tumor Diameter***
MSD-Lung	$(21.98 \pm 51.66)\ cm^3$	$(37.63 \pm 20.08) mm$
NSCLC-Radiomics	$(75.37 \pm 96.30)\ cm^3$	$(63.63 \pm 29.62) mm$
LungDx	$(63.67 \pm 86.26)\ cm^3$	$(48.66 \pm 19.85) mm$

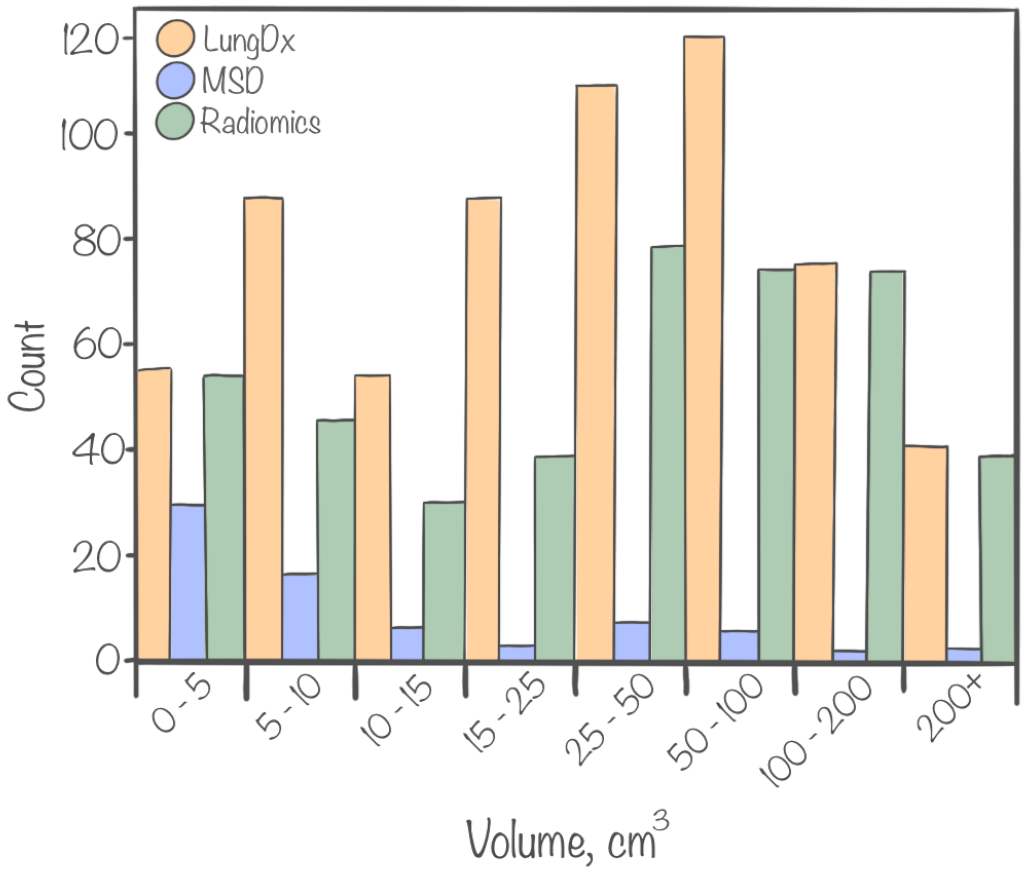


Figure 3.2: Tumor Volume Histogram

Formatting MSD Lung

All images and masks in the MSD dataset were already stored as NIfTI files. The images and labels were already stored in the correct folder structure. To train the teacher, segmentation boxes (segboxes) were created from the original segmentation masks. The segmentation boxes were stored as NIfTI files that inherited the orientation and voxel spacing from the original NIfTI file. A segbox is simply the closest fitting rectangle around the mask, in a axial slice. Listing 3.1 shows how a two-dimensional segbox is created from a two-dimensional segmentation. Each segbox is created in the axial plane. Figure 3.3 shows how the segboxes differ from the original annotations.

Listing 3.1: Segmentation to Segbox

```
# layer is a two-dimensional pixel-array
y = np.any(layer, axis = 1)
x = np.any(layer, axis = 0)
y_min, y_max = np.argmax(y) + 1, len(y) - np.argmax(np.flipud(y))
x_min, x_max = np.argmax(x) + 1, len(x) - np.argmax(np.flipud(x))
layer[y_min - 1:y_max, x_min - 1:x_max] = 1
```

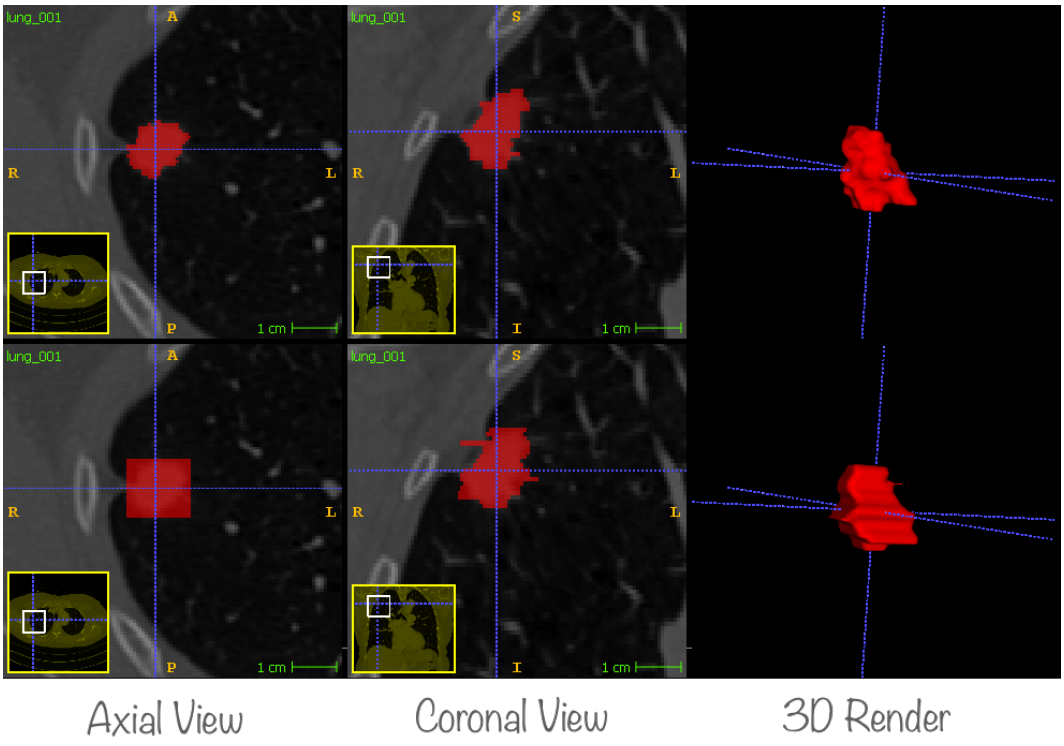


Figure 3.3: Visualization of Segmentations and Segboxes. The top row shows the ground truth annotations, whereas the bottom row shows the generated segboxes. As every segbox is created in the axial plane, the segboxes will always be rectangular in this plane.

Formatting NSCLC-Radiomics

All images were converted from DICOM series to NIfTI images. Voxel intensities, voxel spacing, and orientation were copied from the DICOM files into the NIfTI files. The segmentation annotations were stored in the RTSTRUCT format. This format essentially forms polygons surrounding the tumor in 3D, that can be rendered by an appropriate ITK-software like 3D Slicer [**?**, 52]. This format is not easily manageable for neural networks. To utilize the annotations, we converted the DICOM-RTSTRUT files into NIfTI files, to produce annotations in the same format as the MSD annotations. This was performed using a third-party python package that generates a voxel array and computes which voxels would fall inside the 3D object formed by the polygons [53]. All voxels that would fall inside the 3D tumor object, are considered to be part of the tumor. This process forms a mask similar to the segmentation masks available in the MSD dataset.

Segboxes were then created from the segmentation annotations. This was performed similarly as for MSD. The resulting dataset contained 421 images, as formatting failed for one image.

The images were stored in */Images*, segboxes stored in */Boxes*, and segmentation annotations stored in */Labels*.

Formatting LungDx

This dataset contained PET and CT scans with bounding boxes annotating the location of the tumors. A selection of the images in the dataset are PET/CT-Fused, which means that the PET scan has been merged with the CT scan. During formatting of this dataset, all PET and PET/CT-fused scans were discarded, as our model only targeted CT scans.

Segboxes were created from the bounding boxes where the min and max values were predefined in the annotations, like described in listing 3.1. The annotations were originally stored as XML files, and the generated segboxes were stored as NIfTI files, like MSD. All the images were originally stored as DICOM-series, and these were therefore converted to NIfTI-files.

Originally, the folder structure consisted of one folder per patient, with one subfolder for each examination registered for the given patient. For each examination there was also one subfolder for each scan, conducted during that particular examination. The input image was placed under the folder named */Images* and renamed to match the predefined naming convention. A patient named *A01* that had three separate examinations, with two scans during each examination, would then result in six separate NIfTI files named {*A01_scan1.nii*, ..., *A01_scan6.nii*}. The corresponding segboxes were given the same name, stored in another folder named /Boxes.

The original dataset contained multiple images, which only comprised of a few slices, whereas other images consisted of multiple scans stacked on top of each other. We discarded images where only a fraction of the lung was visible, or if the image consisted of either multiple stacked scans or full-body scans. This was done by calculating the real-world length of each image, by multiplying the voxel spacing in the Z-axis with the number of slices in the image, and then filter out images shorter than $16\ cm$ and longer than $60\ cm$. This interval was selected by inspecting the dataset. No CT image contained lungs that measured more than $60\ cm$ or less than $16\ cm$ in the Z-axis.

3.2 Data Preprocessing

This section corresponds to step two in the overall method, where two different preprocessing pipelines are described. One for preprocessing the images fed to the teacher, named *Teacher-Pipeline*. The other for preprocessing the images fed to the students, named *Student Pipeline*.

3.2.1 Teacher-Pipeline: Tumor Cropped Images

Cropping around the Tumor

Given the existing hardware constraints regarding GPU memory, it was not possible to feed the entire CT scan to teacher network used in this project. During the development of the teacher network, it became clear that downsampling the images significantly degraded performance in terms of DSC. As the tumors can be small, it is compelling to keep as much of the details available to the network as possible. There are two obvious ways of dealing with the resolution issue. One is to downsample the image to a size that fits into the GPU memory. The other is to crop around the tumor. The latter strategy often yields false positives when the algorithm

is applied to new data, but since the teacher network always has the segboxes available, we assumed it would learn to discard tumor-like objects outside the segbox. Based on experiments, it was decided to use the cropping technique when training the teacher.

As briefly mentioned in section 2.1.3, the CT scans have different voxel spacings. This means that two tumors that are both $25 \times 25 \times 25$ voxels on two individual images, might have different dimensions in the real world. When the teacher model was trained and applied, it was a desire that all images appeared as they were in the real world to the network. This was achieved by interpolating the images by trilinear interpolation. They were interpolated such that that each image had a voxel spacing of $(0.7 \times 0.7 \times 0.7)\ mm^3$. This means that it was $0.7\ mm$ distance to all orthogonal neighboring voxels between the center of each voxel. The voxel spacing was chosen by inspecting the dataset and observing that the images with the smallest voxel spacing were around $0.7\ mm$. The finer details of high-resolution images were preserved by choosing a low voxel spacing, whereas images with higher voxel spacing were simply scaled up with no loss of detail.

After the images were interpolated to the correct voxel spacing, they were cropped around the tumor. The center of each tumor was calculated using the segboxes and finding the center of the segbox in each axis. This was done by finding the minimum and maximum value of the segbox in each axis and then finding the center. It was not the center of mass that was calculated, but rather a suitable center for cropping the image. The image was then cropped around this center, such that it measured 128 voxels in each dimension, resulting in a $128 \times 128 \times 128$ image. Figure 3.4 shows the whole procedure.

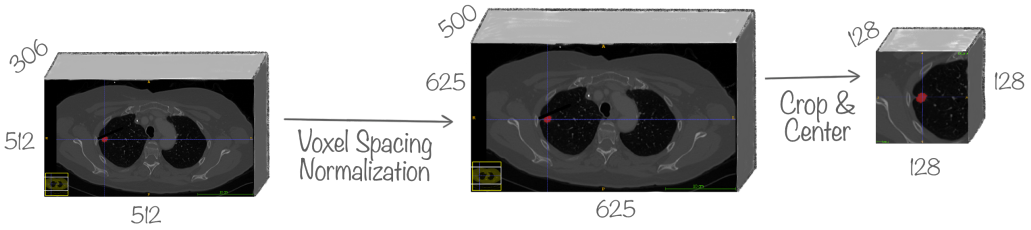


Figure 3.4: Voxel Normalize, Crop and Center Procedure. Conceptual illustration of how a CT image with arbitrary voxel spacings and dimensions are first normalized with respect to voxel spacing, then cropped around the tumor. The resulting 3D image is a $128 \times 128 \times 128$ voxel image with voxel spacing 0.7mm in all dimensions. The figure might give the impression that the image is cropped through the center of the tumor in the axial plane, this is not the case. The figure is cropped this way to show the tumor and illustrate the procedure.

When the formatted images were fed to the network, the full resolution of the tumor was available to the network, and because of the assisting supervision of the bounding boxes, the hypothesis is that false positives should be reduced.

In addition to cropping the tumor, other transforms were applied before the image was fed to the teacher. All voxels that had an intensity value lower than -1024 were changed to the value -1024. In theory, -1024 should be the lowest value, indicating the presence of air in that particular area. This step was performed to remove voxels that might have been artificially added during scanning. We also applied an upper threshold for the voxel values. We set this upper limit to 1000, such that any voxel value with an intensity higher than 1000 was set to 1000. It is common to perform thresholding like this to filter artifacts that are not relevant to

finding tumors. These artifacts might be noise from the scan, metal structures from medical equipment, or metal structures in the body as a result of previous operations on the patient. After the image was adjusted, according to upper and lower intensity thresholds, it was intensity standardized. It was standardized with the *Z-score Normalization* method. Each voxel had its intensity value subtracted by the mean intensity in the image, before the resulting value was divided by the standard deviation of voxel intensities in the image.

Relocate and Paste Tumor Mask

When the network is fed the cropped image, it also produces an output that correlates to the cropped input. If this output should be used as a GT for the original image, it must be placed in the correct location of an image with the same voxel spacing and dimensions as the original image.

During the crop and center procedure, three parameters were stored along with each cropped image. The first parameter was the dimensions of the original image before it is voxel spacing normalized. The second was the dimensions of the image after it is voxel spacing normalized. The last parameter was the center of the cropped image, relative to the voxel spacing normalized image. With these three parameters, it was possible to create an image with the exact same resolution as the original image, and locate where the center of the cropped image aligns with the center of the cropped region in the original image.

The procedure is the reverse of the crop and center procedure. First, an empty image with a resolution equal to the voxel normalized original image was created. Then, the center in this image, where the cropped output should be aligned, was calculated. The cropped output was then pasted into this location in the voxel normalized full resolution image. Remember that the cropped output was already voxel normalized from the crop and center procedure, and thus the voxel spacing of the cropped output and the full resolution placeholder image were the same. Finally, the complete image was interpolated back into the original resolution using trilinear interpolation; the same interpolation that was performed when voxel spacing normalizing was performed in the crop and center procedure. This resulted in the cropped output from the network being placed in the correct location in the full resolution image. Hence, the output could be used as GT for the original image.

3.2.2 Student Pipeline: Separate Lung Cropped Images

The cropping around the tumor, is an appealing strategy for a semi-automatic method, where the tumor's location is known in advance. For a fully automatic method, however, the tumor location is unknown beforehand. Therefore, it is not possible to crop around it. In addition to this fundamental problem, there is another concept that needs to be considered. When inspecting a CT scan, it is hard to separate tumors from other tissues with similar voxel intensities, like bronchial branches, unless global information is taken into consideration. Therefore, if a fully automatic model were trained on cropped images, and applied in a sliding window fashion across the whole image, it would likely run into problems regarding false positives. The model depends on global information, the context around the tumor, to recognize that it is, in fact, a tumor and not part of a healthy tissue. With these two problems in mind, the conclusion is that cropping high-resolution images around the tumor is practically impossible.

The issue with high memory footprint when feeding the whole image remains. Even though cropping around the tumor is not a possibility, cropping the lungs is a viable strategy. Given

that it is possible to accurately locate the borders of the lungs and crop around them, the image size can be drastically reduced by removing air around the patient and even parts of the body that is not within the volume where the lungs are located. It is also possible to take this a step further. Finding a tumor in the left lung is arguably not dependent on information in the right lung. Given this assumption, an image can be cropped around each lung, respectively, resulting in two images that can be processed individually and almost reducing the image size by 50%.

Computing the border of the lungs is not a straightforward task, however. Luckily it is a task that can be said to be solved by other developers, at least within the precision required for our purpose. To locate the lung locations, a third-party GitHub repository developed by Hofmanninger et al. [54] was used. Their inference pipeline is open source and comes with pre-trained models available for anyone to use[3].

The overall procedure is similar to the one used when cropping around the tumor. The fundamental difference is that the lung mask generated using Hofmanninger et al.'s algorithm is used to crop the image, rather than the center of the segboxes. The procedure is illustrated in figure 3.5.

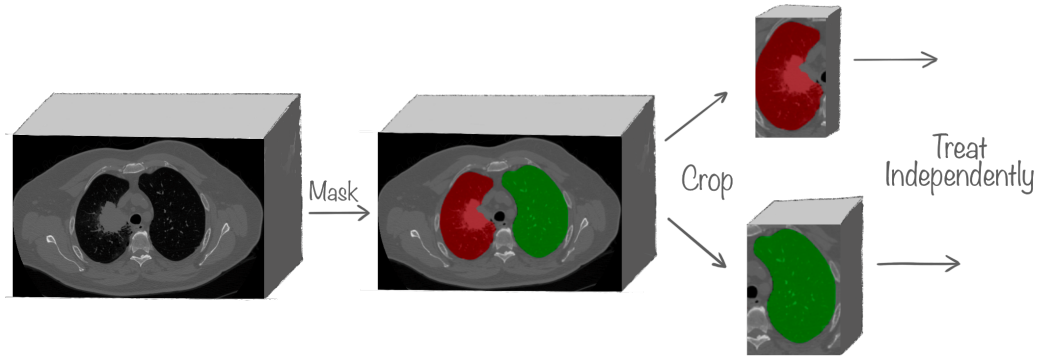


Figure 3.5: The Crop-Around-Lungs Procedure

The process starts by generating a lung mask for each lung given the original CT image (see figure 3.5). The produced masks were then used to find the minimum and maximum intensity values in each axis, to determine a 3D box that tightly encapsulates each lung. These boxes were then used to create a new image, resulting from cropping around each lung, given the extreme values generated from the lung mask. At this point, the original image had been cropped around the lungs and split into two individual images, each containing one of the lungs.

The images were then voxel normalized. Segmentation masks and segboxes were naturally cropped and normalized the same way as the image, such that they were aligned. The final step was to pad the image such that our models could accept it. In the tumor cropping procedure in the **Teacher-Pipeline**, the final resolution was hardcoded to $128 \times 128 \times 128$ voxels. Different patients may have different lung sizes, resulting in different image resolutions after performing voxel spacing. If the resulting images were interpolated into a fixed resolution, this would effectively eliminate the voxel normalization, and then the models would not receive the lungs as they appear in the real world. However, the images cannot have an arbitrary resolution, because the models perform convolutions and poolings requiring the input to be divisible by a certain number. It was decided to pad all the images such that they were divisible by 16 in every dimension, which is equivalent to saying that our models can perform up to four poolings, where

[3]https://github.com/JoHof/lungmask

the resolution is halved each time. The padding was performed by simply adding voxels in each axis until the image was divisible by 16. The voxels added were padded with a zero constant around the original edge. This was done using the python package numpy.

The same intensity transforms, as applied to the teacher data, were applied to the lung-cropped images. The intensity thresholding was performed **before** the padding, such that the padded values were set to zero **after** the intensity normalization was performed.

3.3 The Teacher - A Semi-automatic Annotator

The teacher's task is to produce accurate semantic segmentation annotations from the dataset that has bounding box annotations. These segmentations will, in turn, be used to train the students. The teacher helps expand the available data used to train the students. It is able to create accurate annotations by utilizing the extra input from the bounding box dataset. The tumor location is information that a fully automatic model would not have access to. Due to this additional information, the teacher is able to achieve accuracies that no state-of-the-art, fully automatic model can achieve. One can say that the teacher cheats to help the students later on. This section corresponds to step three in the overall method shown in figure 3.1. The teacher is also interesting as a semi-automatic method. It takes less effort to annotate the tumor with bounding boxes than with masks, potentially saving time.

3.3.1 Architecture

The teacher architecture is based on a residual U-Net variant, developed by Kerfoot et al. [55]. MONAI offers the basic building blocks of this architecture. Slight adjustments concerning depth and the number of input filters were made to fit our needs. The architecture expects a two channeled input; the normalized CT image and the segboxes that indicate the location of the tumor. Figure 3.6 show an overview of the architecture. Unlike the original U-Net, it performs downsampling by using convolutional layers with a stride, rather than pooling operations. Apart from the final part of the network, PReLU is used as the activation function after every convolutional layer. In the final part of the network, the sigmoid activation is used. The number of filters in each layer was decided through experiments and by using the standard method of doubling the number of filters in each layer.

Although this network can accept inputs of any size divisible by eight in all dimensions, it is designed with a resolution of $128 \times 128 \times 128$ in mind. The network is designed to work with data that has been preprocessed by the **Teacher-Pipeline**. The model contains 9.54M trainable parameters. It requires approximately ~ 1.64 GB VRAM during inference, on an image with resolution of $128 \times 128 \times 128$ voxels.

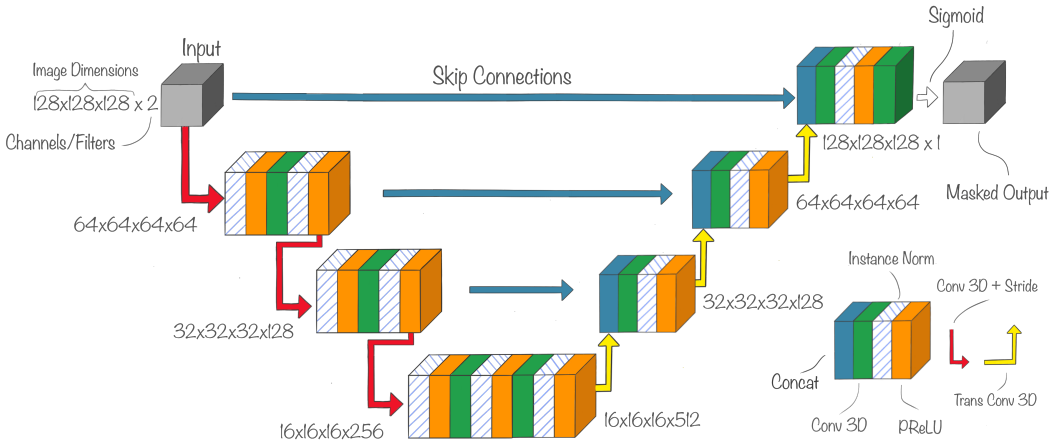


Figure 3.6: Teacher Architectural Design

3.3.2 Training the Teacher

Data & Preprocessing

The teacher was trained on two datasets: the MSD dataset and the Radiomics dataset. Table 3.2 shows how the datasets were split into training and validation. The dataset is not ordered in any particular way. The images selected for each split were selected at random. The dataset splits were balanced with respect to tumor volume. Some images occurred multiple times within one epoch to balance the train and validation sets. The three sets were non-overlapping. The datasets were balanced such that that the network would learn to recognize a wide range of tumor sizes. All images used during the training of the teacher were preprocessed through the **Teacher-Pipeline**.

Table 3.2: Dataset Splits during training of the Teacher. Numbers in parentheses indicates how many unique images there is in each set. Numbers outside parentheses shows the image count after balancing.

Dataset	Train Set	Validation Set	Test Set
MSD	61 (45)	9 (9)	9 (9)
Radiomics	445 (296)	62 (62)	63 (63)
Total	506 (341)	71 (71)	72 (72)

Peripheral Configurations

Dice loss was chosen as the loss function for computing the gradients during training. The PyTorch framework MONAI contains an implementation of dice loss, which is compatible with the tensors in PyTorch. Dice loss was chosen because of its ability to withstand highly imbalanced data, which is common in medical tasks.

To adjust the weights of the network, the **ADAM** optimizer was selected. We used ADAM as it is robust and because experimenting with optimizers is outside the scope of this thesis. The learning rate was selected by trial and error. We ended up using a **learning rate** of $\alpha = \mathbf{1} \times \mathbf{10}^{-\mathbf{4}}$. The batch size was set to one because it was a desire to be able to use the same training loop for both the teacher and the students. Since the students can handle images with varying resolution it was not possible to batch the images. Since the batch size was set to one, we did not get the

speed advantage of feeding multiple images to the network simultaneously. However, to ensure more stable progress during training, we implemented an accumulating gradients system. The accumulating **batch size** was set to **eight**.

Training Phase

The model was trained on Idun, a powerful machine whose hardware is documented in section 3.8. During this particular training, approximately ~ 2.17 GB VRAM was used. During training, the model was validated on the validation set after every epoch. During the validation, the dice loss was calculated in addition to DSC. Before DSC was calculated, a threshold was applied to the output so that the output was a binary mask with zero and one value. The threshold value was set to 0.5, such that outputs above this threshold were interpreted as predicted tumor voxels. The dice loss was calculated on the model's raw output; *before* thresholding is applied to the intensity values. After every validation, the current model was compared to the previously best performing model, which means the model with the lowest validation loss. The best model was always stored on disk as model_best.pth, whereas the last model was always stored as model_last.pth.

After every weight adjustment, the training and validation loss were logged to a file. This enabled monitoring of the progress during training. If we observed that the validation loss was reaching a level where it did not improve after 20 epochs, the training was stopped.

3.4 Expanding the Dataset - Applying the Teacher

After the teacher model was trained, it was ready to expand the strongly annotated dataset given weak labels. This section corresponds to step four in the overall method. The Lung-Dx dataset had been formatted through the **Teacher-Pipeline**. This formatted dataset contained CT images with corresponding segboxes. In addition, the images and segboxes had been cropped around the tumor as explained in section 3.2.

The teacher was then applied to perform inference on the images. The images and the segboxes, were fed as input to the model. The model produced a 3D segmentation with resolution $128 \times 128 \times 128$. Normally, it would be applied a threshold to the network's output, forcing the output to be either one or zero, forming a 3D mask of the tumor. However, it was desired to keep the soft output of the network as the GT for the students. The idea behind this is to embed the uncertainty of the teacher into the annotations. If the teacher has a lower confidence in whether a voxel was part of the tumor or not, this uncertainty should be reflected when training the students as well. If the teacher is used as a standalone semi-automatic method, the network's output should be thresholded to one or zero. The original images in the dataset have resolution $512 \times 512 \times Z$, and the output of the network has a resolution $128 \times 128 \times 128$. The output was rearranged to fit into the original resolution, with the segmentation placed at the correct location.

Initially, we had 484 segmentation annotated lung tumors available through the MSD and Radiomics datasets. By applying the teacher model to the formatted Lung-Dx dataset, the available data was expanded with an additional 628 images, resulting in a total amount of 1112 images. Even though only 484 of them are labeled by human experts, the idea is that the 628 others are labeled good enough that the increased training data will be more important than the lack of precision.

In the Lung-Dx dataset, there are several images belonging to each patient in the study. It is reasonable to believe that the tumor has changed slightly between each examination, given that the tumors in the dataset are malignant. Therefore all images were kept in the dataset, meaning some tumors are present more than once. Images were stratified at patient-level for the Test, Train, or Validation set. No patient was present in two or more categories to ensure that the results are valid.

3.5 The Student - A fully Automatic Method

The student is the fully automatic model that takes single channeled CT images as input, and produces a single channeled output masking the eventual lung tumor. Implementing and training of the student, corresponds to step five in the overall method shown in figure 3.1.

The student was trained on both the human-made segmentation labels and the soft labels generated by the teacher. As opposed to the teacher, the student only receives one channeled input, which is the image. The images the student receive were cropped around the lungs, preprocessed by the **Student Pipeline**.

The idea is that the student will achieve better performance, than a comparable U-Net trained solely on human-labeled data. The hypothesis is that since the student has more available data, because of the soft labeled data annotated by the teacher, the performance will increase compared to a model trained only on the available expert-annotated data.

3.5.1 Architecture

There are two different architectures implemented for the student in this project. One is called **Single-Channel Student** (SC Student) and is identical to the MONAI implemented U-Net with one input channel and one output channel. The other is called **Dual-Channel Student** (DC Student) and is different from the standard MONAI U-Net.

Both these students and the teacher share the same building blocks like activation functions, normalization layers, and convolutional layers. When comparing the teacher architecture in figure 3.6 and the architecture of the two students in figure 3.7 and figure 3.8, these similarities can be observed. One of the differences is that the teacher architecture has four layers with three down-samplings of the resolution, whereas both student architectures have five layers with four down-samplings of resolution. From experiments we observed that four layers were sufficient for the teacher to process $128 \times 128 \times 128$ images. The students are designed to process larger images. Through experiments we discovered that adding a layer was beneficial for the student models.

The difference between the students are related to the input and output channels of the networks. Since the students does not get any bounding boxes to help locate the tumors, both student architectures only have a single input channel. However, the two student architectures differ in the decoder part of the network, as can be seen in the figures. The SC Student has one output channel, as the teacher had, whereas the DC Student has two output channels. One of the channels of the DC Student is predicting the tumor segmentation, whereas the other is predicting the segbox of the tumor. The DC Student architecture contains two separate decoder branches, as can be seen in the figure 3.8. The first branch learns to decode the tumor segmentation, whereas the second branch decodes the tumor segbox. The idea behind this separation is that

the second branch can facilitate tumor detection and location learning, because both branches share the same encoder. During training, GT masks were used to backpropagate the first branch, whereas segbox annotations were used to backpropagate the second. However, both types of labels were used for backpropagating the encoder part of the network. The branch creating segboxes can focus more on finding the tumors' whereabouts, and during this process, the encoder is trained to encode the relevant properties better as well. The first branch will focus on making a segmentation as good as possible, which is the primary task that shares many features with the auxiliary task, meaning it can benefit from an encoder that can encode the relevant features. When the DC Student performs inference, only the output from the segmentation branch is relevant.

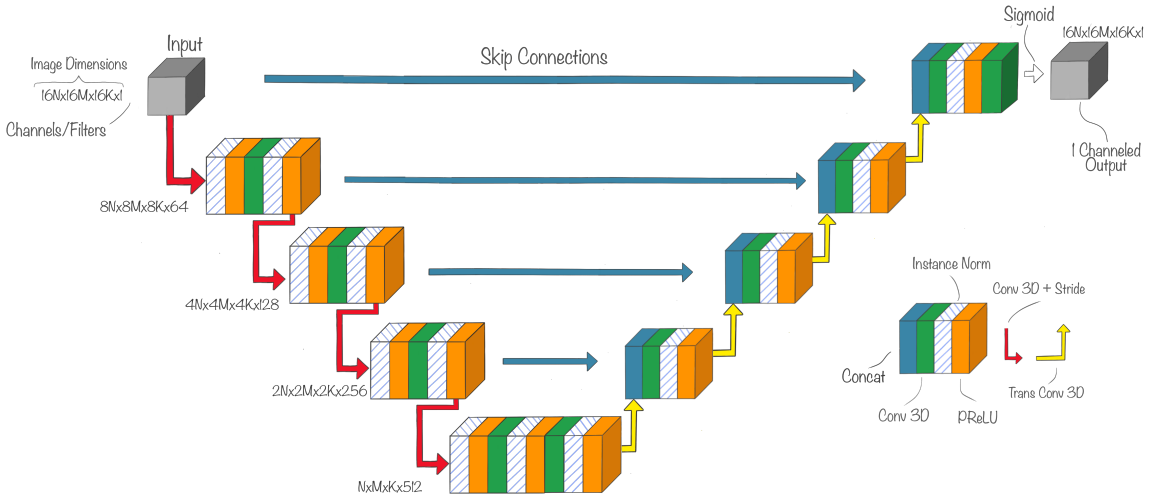


Figure 3.7: SC Student Architectural Design

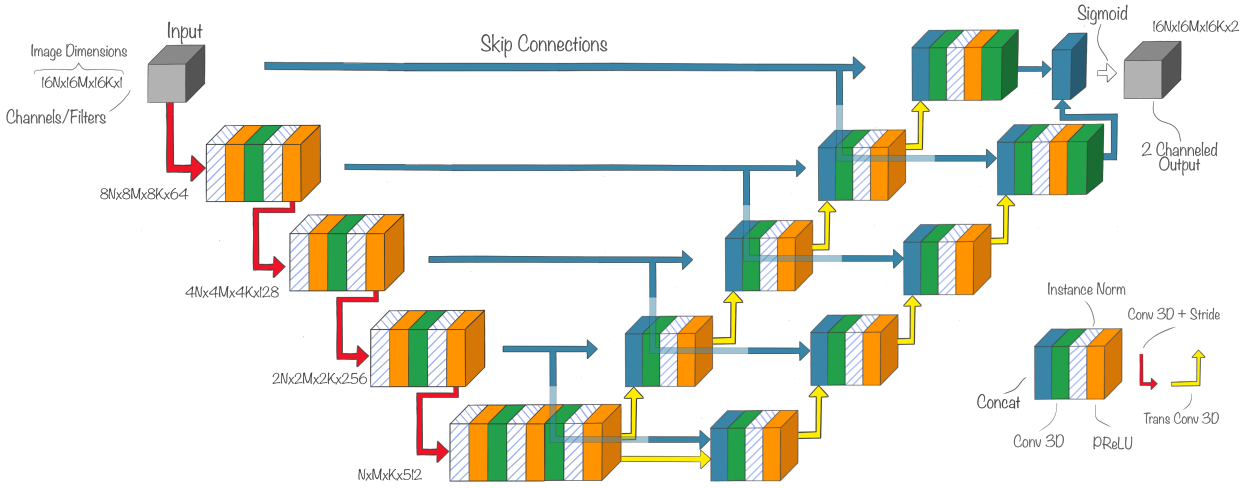


Figure 3.8: DC Student Architectural Design

The implementation of DC Student is inspired by the student architecture found in the paper published by Sun et al. [41]. However, there are some differences between our implementation and theirs. The biggest difference being that our DC Student architecture has two completely separate decoders, whereas Sun et al. proposed an architecture where most of the decoder is

shared between the segmentation task and bounding box task. Their architecture shares more similarities with the original U-Net network, where they use pooling operation, whereas our networks uses convolutional layers with stride for downsampling.

3.5.2 Training the Student

Data & Preprocessing

The teacher was trained on two of the available datasets, and the students was trained on all three datasets. Table 3.3 shows the dataset split. The datasets were balanced with regards to tumor volume. Before the images were passed to the students, they were preprocessed through the **Student Pipeline**. The images in the MSD and Radiomics datasets had expert-labeled annotations used as ground truth during the student's training. The soft labels generated by the teacher were used as ground truth labels for the images in the LungDx dataset.

The test images selected for the student were the same images selected for the teacher's test set, but preprocessed through the **Student Pipeline**. The validation images were not the same as the corresponding validation images used during the training of the teacher. They were randomly sampled and balanced with regards to tumor volume. They were sampled the same way as the teacher's train set was sampled. Because of the additional LungDx images, the sample resulted in a different training set than what was used to train the teacher.

Table 3.3: Dataset Splits during training of the Student. Numbers in parentheses indicates how many unique images there is in each set. Numbers outside parentheses shows the image count after balancing.

Dataset	Train Set	Validation Set	Test Set
MSD	61 (45)	9 (9)	9 (9)
Radiomics	445 (296)	62 (62)	63 (63)
LungDx	760 (513)	115 (115)	0 (0)
Total	1266 (854)	186 (186)	72 (72)

Peripheral Configurations

The MONAI **dice loss** implementation was chosen as the loss function. The **ADAM optimizer** was selected as the optimizer for the students. A learning rate of $\alpha = \mathbf{3} \times \mathbf{10^{-4}}$ was initially used for the students. This learning rate is higher than the learning rate used by the teacher. When the validation DSC stopped increasing, the learning rate was lowered to $\alpha = \mathbf{1} \times \mathbf{10^{-4}}$.

Training Phase

The training of the student used more VRAM than the training of the teacher, up to ~ 22.54 GB VRAM. However, this is dependent on the size of the images in the dataset and which of the student architecture used, where the DC Student was the most memory intensive.

The training of the students shares many of the characteristics as to how the teacher was trained. A threshold value of 0.5 was used when calculating DSC. After each validation, the current model was compared with the previous best model with respect to DSC; if the current was better, it was stored on disk as the currently best. The model was seen fully trained when the validation DSC converged after already lowering the learning rate once. The student models were trained on the Bohaga machine, which is specified in section 3.8.

3.6 Training on Sparse Data

To investigate research question three, we constructed an experiment where we artificially decreased the size of strongly labeled data. Initially, we had 484 strongly annotated images and an additional 628 weakly annotated images. To check how the Teacher-Student Framework performs in extreme cases, we artificially shrunk the strongly labeled dataset to 48 images. These 48 images were picked semi-random from the MSD and Radiomics datasets. They were picked semi-random in the sense that they were picked randomly but balanced according to tumor volume. We picked six images from each *bucket* seen in figure 3.2, this way, the training data is balanced with respect to tumor size. The 72 images used for testing the teacher and the students mentioned previously were kept in the test set during this experiment too. The remaining 992 images were treated like weakly annotated images, that is, images where only the CT and segbox images are present. From the 48 images, one image from each bucket was appointed to the validation set; the remaining 40 were placed in the training set.

The overall method shown in figure 3.1 was used during this experiment. First, a teacher was trained, then it was used to expand the dataset by performing inference on the weakly annotated data, and finally, students were trained on the expanded dataset.

Training the Teacher

The teacher was trained on the 40 strongly labeled data. Table 3.4 shows how the dataset was distributed between train, validation, and test sets. It was trained the same way as other teacher that used all the available strongly labeled data. The results of training the teacher on sparse data are available in section 4.3.

Table 3.4: Dataset Splits during training of Teacher on Sparse Data. *Originally* refers to the data used to train the teacher in section 3.3.2. Numbers in parentheses indicates how many unique images there is in each set. Numbers outside parentheses shows the image count after balancing.

	Train Set	**Validation Set**	**Test Set**
Originally	506 (341)	71 (71)	72 (72)
Sparse (this)	40 (40)	8 (8)	72 (72)

Expanding the Dataset

When the teacher was fully trained, we used it to expand the dataset. The teacher performed inference on all the remaining 992 images as if they were weakly labeled. The inferred annotations were formatted to fit the original image resolution, and ready to be used during the training of the students.

Training the Student

The students were trained on lung-cropped images generated by the **Student Pipeline**. The students were trained on all the images generated from the teacher and the strongly annotated images. It was trained using the same configurations as the students previously described. The dataset splits used during the student's training are shown in table 3.5.

Table 3.5: Dataset Splits during training of Student on Sparse Data. Numbers in parentheses indicates how many unique images there is in each set. Numbers outside parentheses shows the image count after balancing.

	Train Set	**Validation Set**	**Test Set**
Strong Labels	40 (40)	8 (8)	72 (72)
Teacher Soft Labels	1002 (812)	180 (180)	0 (0)
Total	1042 (852)	188 (188)	72 (72)

The architecture of the teacher and the students, and the way they were trained, are the same as in the previous sections. The principle in this experiment is to shift the balance between strongly and weakly labeled images to answer research question three. The results from this experiment is available in section 4.3.

3.7 Post-Processing

A simple post-processing method was implemented to process the output of the models. MONAI offers a function that removes all but the largest connected component in a 3D array. The post-processing method was used to remove noise in the models' output. It is only used during the testing of the models. The models are evaluated both before and after post-processing. Figure 3.9 shows an example of how the output is post-processed.

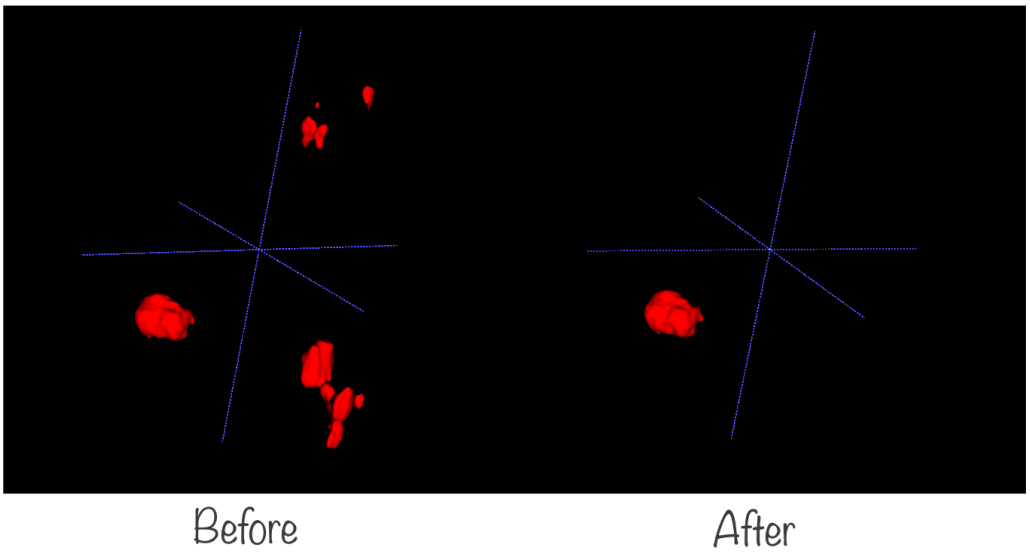


Figure 3.9: The Post-Processing Method. The image shows the output of a network before and after the post-processing is applied.

3.8 Hardware & Software

The development was performed on conventional desktop machines. Debugging, data formatting, and training were performed on powerful computers. We had access to three different machines during this thesis. Most of the training was conducted on either Idun or Bohaga. Idun is a state-of-the-art computer cluster maintained by NTNU's High Performance Computing

Group [56]. The cluster is equipped with multiple nodes with different hardware. In this project, nodes with a Nvidia Tesla V100 GPU with 32 GB VRAM were used. Even though 768 GB RAM is available, only the necessary memory is allocated during the reservation of the node. The allocation was performed manually using a job batching system.

Bohaga is a conventional computer running the Linux Mint operating system, equipped with two powerful GPUs. NTNU AI Lab allowed us to use one of their machines for debugging and data formatting purposes[4]. This machine is named Malvik and is also equipped with powerful hardware. A summary of the specifications of all three machines can be seen in table 3.6.

Table 3.6: Hardware Specifications

Machine	**Device**	**Details**
Idun	GPU	Nvidia Tesla V100, 32GB
	CPU	Intel Xeon Gold 6148, 20 core 2.4GHz
	RAM	768GB
Bohaga	GPU	Nvidia RTX 8000, 48GB
	CPU	Intel Core i9 10940X, 14 core 3.3GHz
	RAM	128GB
Malvik	GPU	Nvidia Tesla V100, 32GB
	CPU	Intel Xeon Gold 6132, 14 core 2.6GHz
	RAM	768GB

The project was implemented using Python 3.9.1. The most central libraries used is specified in table 3.7. The complete list of python requirements is available in the Github repository[5]. To inspect the data and to qualitatively evaluate the results we used ITK-Snap [9], a software capable of viewing medical images and render the masks in 3D.

Table 3.7: The most Important Python Libraries used

Library	**Version**
Torch	1.7.0
Numpy	1.17.2
MONAI	0.4.0
Hofmanninger et al.'s Lungmask[6]	0.2.9

3.9 Model Evaluation

Evaluation Metrics

The models were evaluated using five different metrics; DSC, DSC-TP, F1 Score, and Precision and Recall. DSC-TP is simply the DSC calculated only on the objects regarded as *true positives*. If a model perfectly segments a tumor, but also segments an other part of the CT, then the DSC would take a hit, but the DSC-TP would still be 1.0. This is an effective measure of how well the model performs segmentation when it first detects the tumor. F1 Score is a metric that balances precision and recall, which are already discussed in the background chapter. The F1 Score metric is calculated from positives and negatives as shown in equation (3.1). In some

[4]https://www.ntnu.edu/ailab
[5]https://github.com/VemundFredriksen/lungtumormask

literature, the metrics are presented in the range 0.0 to 1.0, like the equations suggests. We present the metrics in the range 0.0 to 100.0, effectively removing the insignificant, leading zero.

$$F1 = \frac{2TP}{2TP + (FP + FN)} \tag{3.1}$$

We evaluated DSC and DSC-TP on *voxel level* and Precision, Recall and F1 Score on *patient-level*. A recall of 100.0 does not mean that our model found every single voxel, but that it found every single tumor in every single patient. A mask is regarded as a true positive if the mask overlaps the ground truth by more than 25%. Before positives and negatives were calculated, small masks were discarded. Small grains of singular voxels would have had a big impact on the FP count and will never be the tumor. We discarded all predictions smaller than $100\ mm^3$.

When the teacher is evaluated, only voxel-level metrics are considered. Because the tumor is already located, tumor-level metrics are not interesting to measure for these models.

Test Data

The models were evaluated on the same test set, regardless of experiment. The test set comprised of 72 images in total; 9 images from the MSD dataset and 63 from the Radiomics dataset. It was balanced with respect to tumor size.

When evaluating the models, they were evaluated and discussed on each dataset isolated. The Radiomics dataset is ~ 7 times larger than the MSD dataset. Therefore, the combined result of both dataset would be biased towards the Radiomics dataset.

CHAPTER 4

RESULTS

This chapter is divided into three parts. These parts can be regarded as three separate experiments conducted to evaluate the models, and to answer the research questions. In each experiment, the relevant quantitative metrics are presented in addition to sample images that are relevant to qualitatively evaluate the models.

In section 4.1, the baseline model is called U-Net, which is a U-Net model trained on cropped images, but only recieved the CT image as input. *Teacher* is the model that recieves both the CT image and segboxes as input. In section 4.2 and section 4.3, the model named *U-Net* is a baseline U-Net model, which is only trained on strongly annotated data from MSD and Radiomics. The *SC Student* and the *DC Student* are previously covered.

4.1 The Teacher as a Semi-Automatic Method

As can be seen in table 4.1, the U-Net model achieves a DSC of 74.78, whereas the teacher achieved a DSC of 84.91 on the MSD testset. The difference between the two models were more evident on the Radiomics dataset, where the U-Net achieved a DSC of 59.57, wheras the teacher achieved 86.65. Post-processing improved the DSC for both models on both datasets.

Table 4.1: Teacher Results. The best performing model with respect to mean DSC is highlighted in bold. The model is evaluate without and with post-processing (w/post)

Dataset	**Model**	**DSC**
MSD	U-Net	74.78 ± 11.83
	U-Net w/post	75.54 ± 11.54
	Teacher	84.91 ± 06.09
	Teacher w/post	$\mathbf{84.92 \pm 06.08}$
Radiomics	U-Net	59.57 ± 23.90
	U-Net w/post	61.70 ± 26.03
	Teacher	86.65 ± 08.77
	Teacher w/post	$\mathbf{87.17 \pm 05.56}$
Both	U-Net	61.48 ± 23.29
	U-Net w/post	63.43 ± 25.11
	Teacher	86.44 ± 08.50
	Teacher w/post	$\mathbf{86.89 \pm 05.42}$

Figure 4.1 shows examples from the test set. All figures show predictions before post-processing. The results on some of the images, denoted MSD 27, Radiomics 89, and Radiomics 223 in the figure, are somewhat representative for the teacher, but there are cases where the teacher struggles, such as MSD 42 and Radiomics 200. MSD 27 is image number 27 in the MSD dataset, the other samples in the figures are named accordingly.

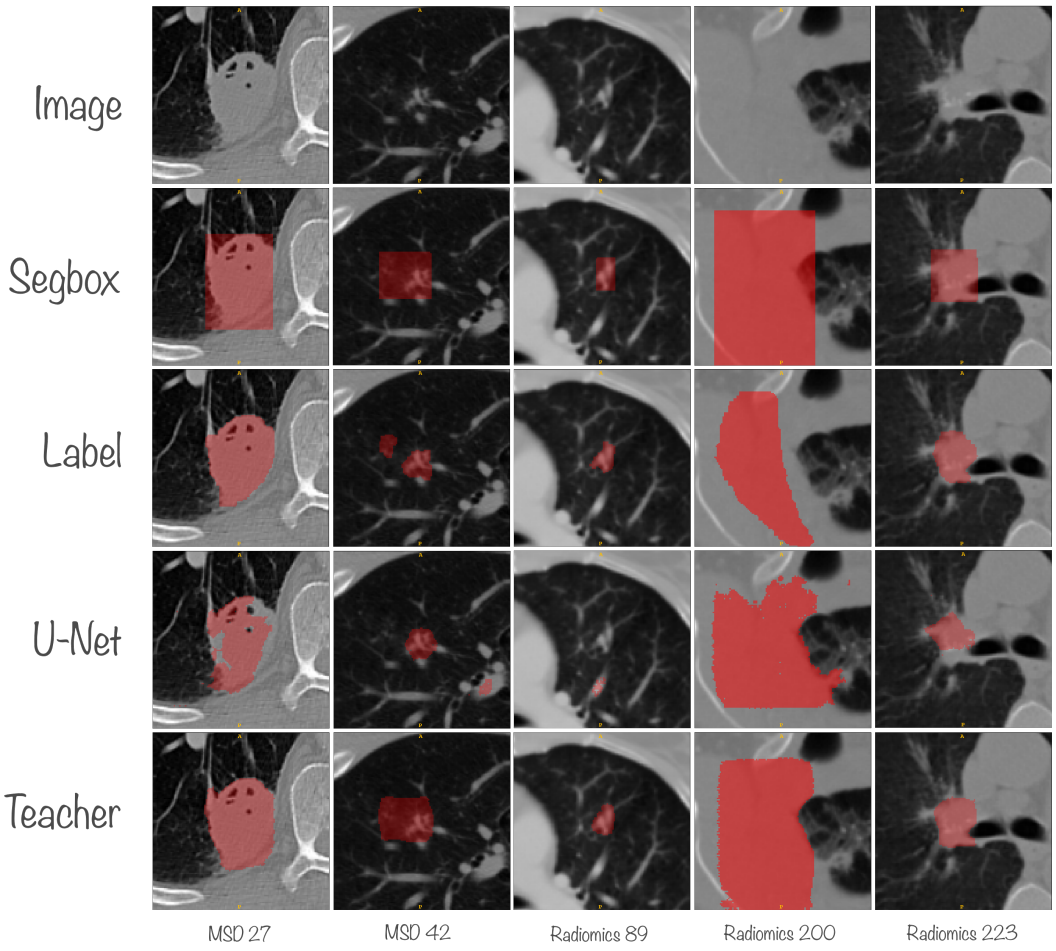


Figure 4.1: Teacher Result Samples in the Axial Plane. The top row shows the image, followed by the segbox and the label. The baseline model output and teacher output are shown in the last two rows.

4.2 The Student as a Fully Automatic Method

The essence of the results of this experiment is that the three models perform on a similar level in terms of the metrics used to evaluate them. All three models consistently performed better on the MSD dataset, than the Radiomics dataset.

In more detail, it can be observed from the table that the U-Net model performed best on the MSD dataset and achieved a DSC of 67.31. All models performed poorer in terms of DSC on the Radiomics dataset. The SC Student was the best student network and achieved a DSC of 52.92 on the Radiomics dataset. Post-processing degraded the DSC for all models on both datasets.

The U-Net achieved the highest DSC-TP on the MSD dataset, with a DSC-TP of 73.12. For all models, on both datasets, post-processing had negligible effect on DSC-TP. The largest effect

was observed from the best performing model on the Radiomics dataset, the SC Student, with an improved DSC-TP of 69.97 from 69.39. The model that achieved the highest DSC also achieved the highest DSC-TP.

For the object detection metrics: F1 Score, recall, and precision, the performance was better overall on the MSD dataset. The U-Net and SC Student achieved the highest F1 Score of 88.89 after post-processing on the MSD dataset. Post-processing did not affect recall on the MSD dataset, but improved precision. A similar trend was observed on the Radiomics dataset. Post-processing improved F1 and precision for the best performing models. The best model was the SC Student which achieved an F1 of 72.79, whereas the U-Net achieved a lower F1 of 72.06. However, the U-Net achived the highest recall of 83.82 before post-processing, whereas the SC Student had the best precision of 73.53 after post-processing. Overall, no benefit of post-processing in terms of recall was observed, whereas both precision and F1 Score always benefited from post-processing.

Table 4.2: Student Results. For each respective metric, the best performing models are highlighted in bold.

Dataset	Model	DSC	DSC-TP	F1 Score	Recall	Precision
MSD	U-Net	$\mathbf{67.31 \pm 21.17}$	$\mathbf{73.12 \pm 15.16}$	81.48 ± 31.86	$\mathbf{88.89 \pm 31.43}$	77.78 ± 34.25
	U-Net w/post	64.99 ± 27.06	$\mathbf{73.12 \pm 15.16}$	$\mathbf{88.89 \pm 31.43}$	$\mathbf{88.89 \pm 31.43}$	$\mathbf{88.89 \pm 31.43}$
	SC Student	64.27 ± 16.05	71.32 ± 8.06	81.48 ± 31.86	$\mathbf{88.89 \pm 31.43}$	77.78 ± 34.25
	SC Student w/post	63.40 ± 23.67	71.32 ± 8.06	$\mathbf{88.89 \pm 31.43}$	$\mathbf{88.89 \pm 31.43}$	$\mathbf{88.89 \pm 31.43}$
	DC Student	55.37 ± 29.03	70.49 ± 8.82	74.07 ± 40.91	77.78 ± 41.57	72.22 ± 41.57
	DC Student w/post	54.83 ± 30.32	70.49 ± 8.82	77.78 ± 41.57	77.78 ± 41.57	77.78 ± 41.57
Radiomics	U-Net	51.06 ± 28.22	68.81 ± 18.27	63.56 ± 36.36	$\mathbf{83.82 \pm 36.82}$	56.68 ± 38.65
	U-Net w/post	50.07 ± 32.74	67.73 ± 18.35	72.06 ± 44.87	72.06 ± 44.87	72.06 ± 44.87
	SC Student	$\mathbf{52.92 \pm 31.13}$	69.39 ± 19.21	64.18 ± 37.37	79.90 ± 39.66	58.76 ± 39.28
	SC Student w/post	51.76 ± 34.39	$\mathbf{69.97 \pm 18.7}$	$\mathbf{72.79 \pm 44.09}$	72.55 ± 44.26	$\mathbf{73.53 \pm 44.12}$
	DC Student	52.25 ± 30.18	68.69 ± 19.47	68.43 ± 38.71	79.17 ± 39.75	64.95 ± 40.81
	DC Student w/post	50.88 ± 33.94	69.39 ± 20.00	69.85 ± 45.49	69.61 ± 45.64	70.59 ± 45.56
Both	U-Net	52.96 ± 27.98	69.34 ± 17.97	65.66 ± 36.32	$\mathbf{84.41 \pm 36.27}$	59.14 ± 38.76
	U-Net w/post	51.82 ± 32.49	68.49 ± 18.03	74.03 ± 43.85	74.03 ± 43.85	74.03 ± 43.85
	SC Student	$\mathbf{54.25 \pm 29.99}$	69.63 ± 18.19	66.20 ± 37.19	80.95 ± 38.90	60.98 ± 39.21
	SC Student w/post	53.12 ± 33.52	$\mathbf{70.16 \pm 17.63}$	$\mathbf{74.67 \pm 43.11}$	74.46 ± 43.28	$\mathbf{75.32 \pm 43.11}$
	DC Student	52.61 ± 30.06	68.90 ± 18.59	69.09 ± 39.02	79.00 ± 39.97	65.80 ± 40.97
	DC Student w/post	51.34 ± 33.56	69.53 ± 18.95	70.78 ± 45.12	70.56 ± 45.26	71.43 ± 45.17

Figure 4.2 shows a selection of cases from the test set, with corresponding labels and output from the three models. The images were slices from the axial plane, except MSD 42 which was extracted from the sagittal plane. Figure 4.3 shows the 3D rendered output of the corresponding images shown in figure 4.2. The 3D rendered output reveals false positives that cannot be seen in a single axial slice.

In some of the images all three models produced similar images, such as MSD 27, MSD 78 and Radiomics 154. In image MSD 42 the U-Net and SC Student produced similar images and the DC Student produce a different image. Sometimes, the students produced similar results whereas the U-Net produced a different mask, seen in Radiomics 332 and Radiomics 422.

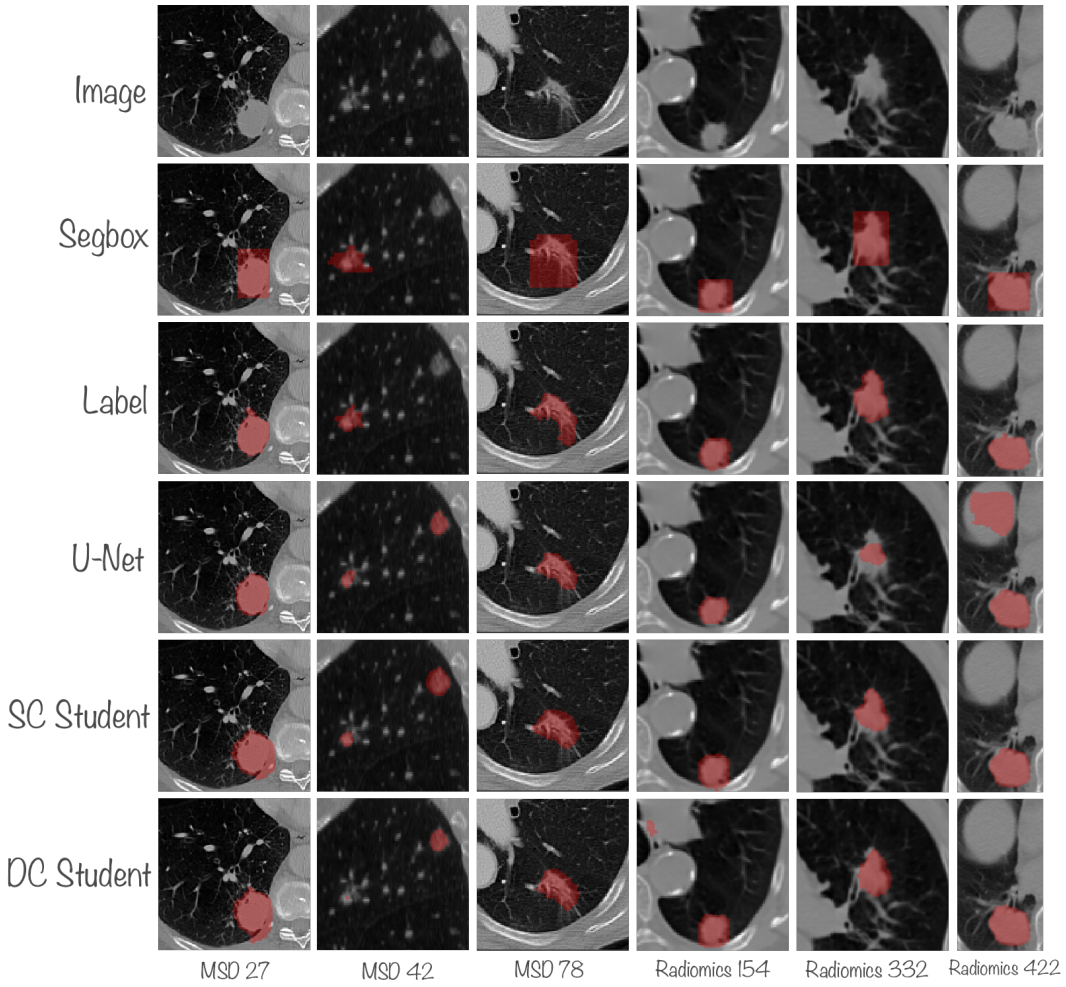


Figure 4.2: Student Result Sample. Example cases are shown column-wise. The input image is seen in the top row, followed by the segbox, GT and the three model outputs. The images were cropped around the tumour for illustrative reasons.

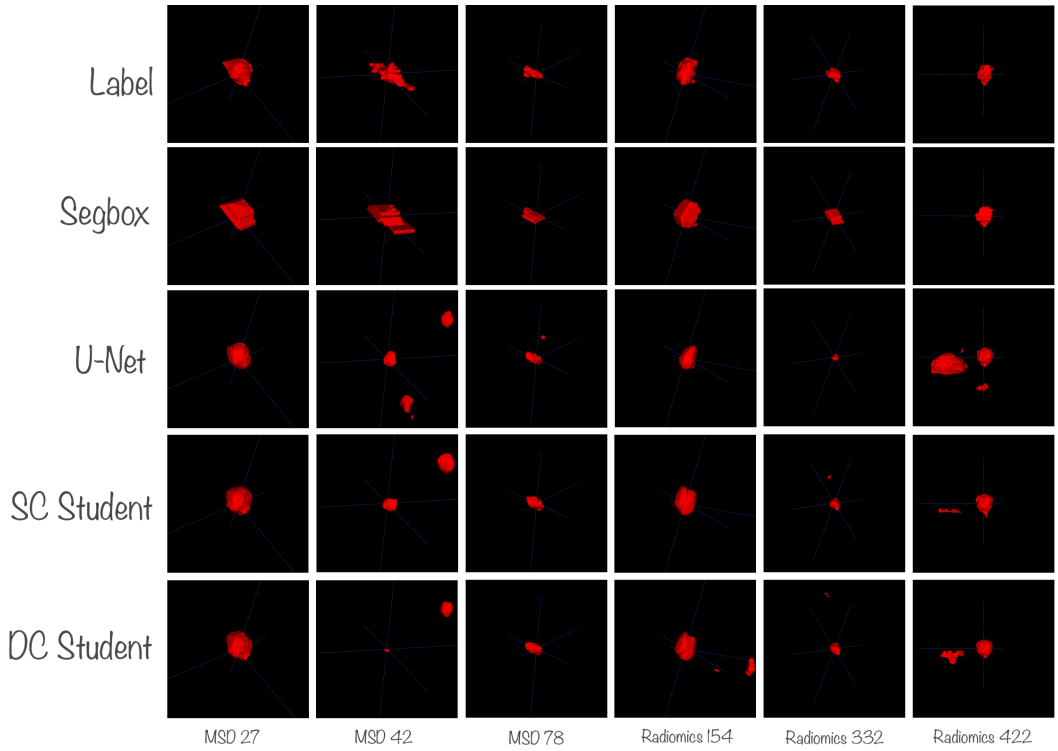


Figure 4.3: 3D Render of Student Result Sample. A 3D render of the corresponding samples shown in figure 4.2. For each image, the camera position and zoom are stationary between the outputs, such that they can be easily compared.

4.3 Sparsely Trained Models

Sparse Teacher Results

The takeaway of the results related to the sparsely trained teacher it performed on a similar level as the teacher in the first experiment, whereas the sparsely trained baseline U-Net performed worse when the baseline trained on all available data. The difference between the two sparsely trained models are larger than the difference between the two semi-automatic models previously presented.

Table 4.3 shows the results of the sparsely trained teacher. The baseline U-Net achieved a DSC of 48.52 on the MSD dataset and 43.84 on the Radiomics dataset. The teacher outperformed the baseline on both dataset with a DSC of 81.65 and 84.69 on the MSD and the Radiomics datasets, respectively.

Table 4.3: Sparsely Trained Teacher Results. The best performing model with respect to mean DSC is highlighted in bold.

Dataset	**Model**	**DSC**
MSD	Sparse U-Net	48.52 ± 31.18
	Sparse U-Net w/post	43.51 ± 35.41
	Sparse Teacher	$\mathbf{81.65 \pm 07.40}$
	Sparse Teacher w/post	$\mathbf{81.65 \pm 07.40}$
Radiomics	Sparse U-Net	43.83 ± 25.65
	Sparse U-Net w/post	44.56 ± 27.54
	Sparse Teacher	$\mathbf{84.69 \pm 06.59}$
	Sparse Teacher w/post	84.60 ± 06.68
Both	Sparse U-Net	44.42 ± 26.45
	Sparse U-Net w/post	44.43 ± 28.65
	Sparse Teacher	$\mathbf{84.31 \pm 06.77}$
	Sparse Teacher w/post	84.23 ± 06.84

Figure 4.4 shows a sample of the outputs produced by the U-Net and the teacher, respectively. The sparsely trained teacher has similarities with the teacher from the first experiment. The U-Net performs poor compared to the Teacher. One example of this poor performance can be observed in MSD 27. Here the U-Net has missed part of the tumor, whereas the teachers mask closely resembles the label. In Radiomics 89, the U-Net model completely ignores the tumor, whereas the teacher detected it.

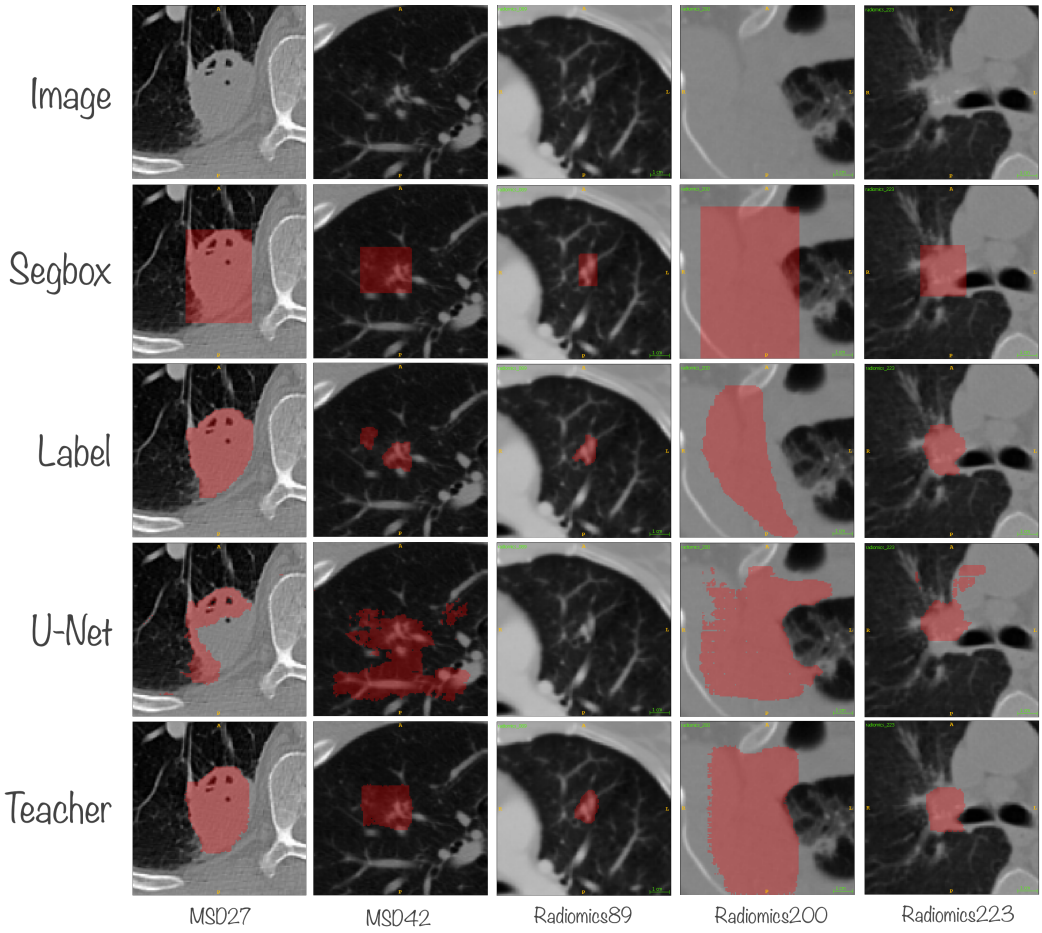


Figure 4.4: Sparse Teacher Result Sample in the Axial Plane.

Sparse Student Results

The essence of of the result of this experiment is that the sparsely trained students performed better than the sparsely trained U-Net model. The difference between the sparsely trained models are greater than the difference between the previously discussed students.

Table 4.4 shows the results of the student models trained on the sparse data. The U-Net model that is trained on the 40 human-annotated images achieved the lowest DSC on both dataset. All models performed similarly in terms of DSC-TP. Both recall and precision were poorer for the U-Net compared with the students. The SC Student achieves a perfect F1 Score of 100.0 on the MSD dataset, whereas the DC Student achieved 88.89. The U-Net performed poorer with an F1 Score of 09.10 before post-processing. A similar trend on the Radiomics dataset was observed. However, the DC Student was the best performing student in terms of DSC and F1 Score on the Radiomics dataset, whereas on the MSD dataset the SC Student performed best.

Table 4.4: Sparsely Trained Student Results. For each respective metric, the best performing model is highlighted in bold.

Dataset	Model	DSC	DSC-TP	F1 Score	Recall	Precision
MSD	U-Net	26.45 ± 26.56	75.24 ± 15.90	09.10 ± 13.26	33.33 ± 47.14	05.31±07.80
	U-Net w/post	28.27 ± 35.55	75.24 ± 15.90	33.33 ± 47.14	33.33 ± 47.14	33.33±47.14
	SC Student	64.74 ± 11.82	71.56 ± 10.40	61.85 ± 16.49	**100.0 ± 0.00**	47.22±20.79
	SC Student w/post	**71.56 ± 10.40**	71.56 ± 10.40	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
	DC Student	71.00 ± 16.01	**76.67 ± 07.08**	85.18 ± 31.86	88.89 ± 31.43	83.33±33.33
	DC Student w/post	68.15 ± 25.00	**76.67 ± 07.08**	88.89 ± 31.43	88.89 ± 31.43	88.89±31.43
Radiomics	U-Net	28.23 ± 28.05	55.39 ± 22.80	32.13 ± 36.99	51.47 ± 49.24	26.99±35.68
	U-Net w/post	27.48 ± 30.85	53.94 ± 23.25	48.04 ± 49.63	47.79 ± 49.58	48.53±49.98
	SC Student	51.06 ± 30.75	**68.56 ± 20.69**	62.65 ± 36.73	79.41 ± 39.51	56.67±38.79
	SC Student w/post	50.05 ± 34.64	68.13 ± 21.46	71.57 ± 44.74	71.32 ± 44.82	72.06±44.87
	DC Student	53.89 ± 29.75	67.44 ± 21.70	66.96 ± 35.57	**84.56 ± 35.62**	60.44±38.24
	DC Student w/post	**54.88 ± 32.22**	67.22 ± 21.75	**80.39 ± 39.29**	80.15 ± 39.43	**80.88 ± 39.32**
Both	U-Net	28.02 ± 27.89	56.91 ± 22.96	29.44 ± 35.82	49.35 ± 49.34	24.45±34.35
	U-Net w/post	27.58 ± 31.44	55.71 ± 23.48	46.32 ± 49.57	46.10 ± 49.52	46.75±49.89
	SC Student	52.66 ± 29.51	**68.98 ± 19.60**	62.55 ± 34.98	81.82 ± 37.72	55.56±37.26
	SC Student w/post	52.56 ± 33.47	68.66 ± 20.19	74.89 ± 43.03	74.67 ± 43.11	75.32±43.11
	DC Student	55.89 ± 29.01	68.56 ± 20.71	69.09 ± 35.64	**85.06 ± 35.18**	63.12±38.41
	DC Student w/post	**56.43 ± 31.75**	68.42 ± 20.72	**81.38 ± 38.55**	81.17 ± 38.68	**81.82 ± 38.57**

Figure 4.5 shows a sample of the output from the various models in the axial plane, except MSD 42 which is a slice in the sagittal plane. In MSD 27 and Radiomics 422, the U-Net partially masks the tumor. However, they are more restrictive than the students' masks. In the remaining images, the masks produced by the U-Net is either empty, not masking the tumors or masking small portions of the tumors, whereas the SC Student and the DC Student have similar masks. Figure 4.6 shows the same images where the output is 3D rendered. It can be observed that the U-Net produces large false positives for several of the images. Although false positives can occur for the students, this effect is seen more seldom and the false positive tumors are usually smaller.

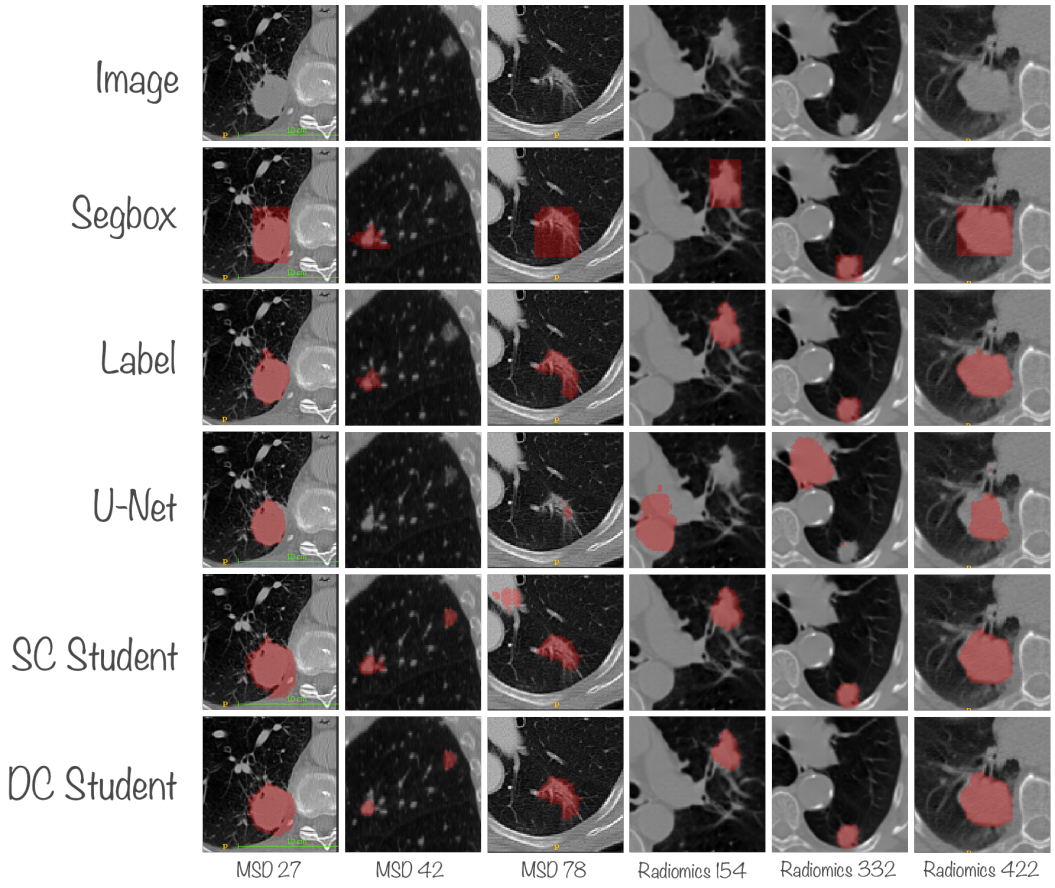


Figure 4.5: Sparse Student Result Sample

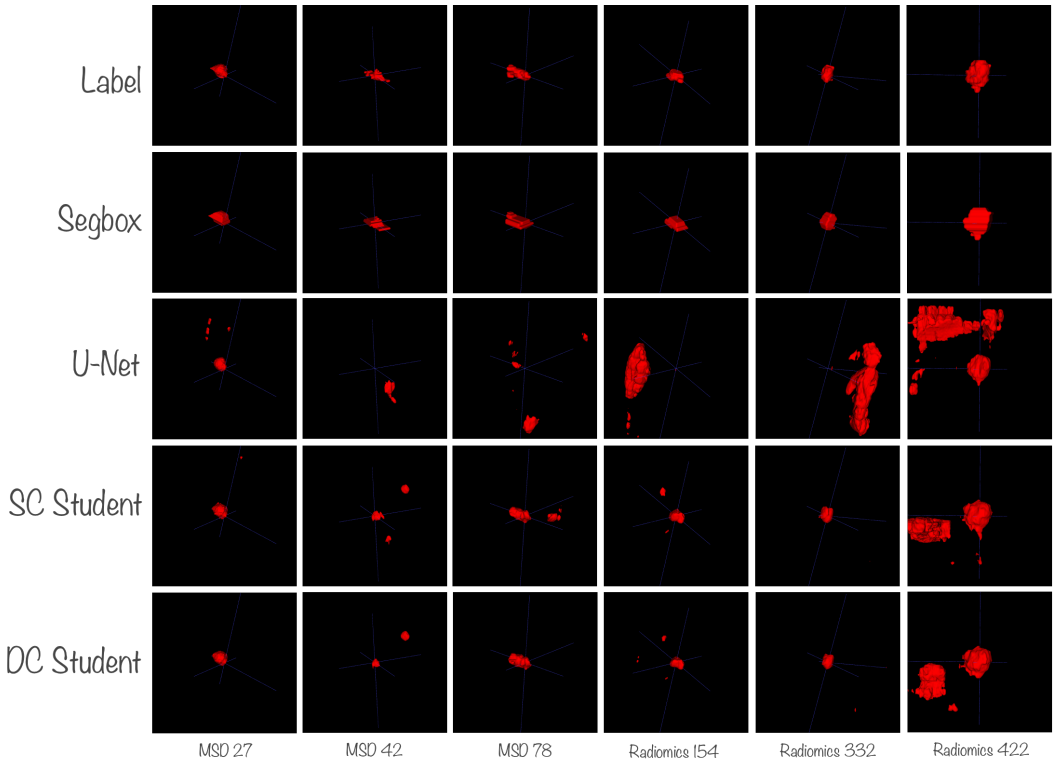


Figure 4.6: 3D Render of Sparse Student Result Sample

CHAPTER 5

DISCUSSION

In this thesis, we have explored the potential of a Teacher-Student Framework to improve the performance of automatic lung tumor segmentation models on CT images. We have conducted three separate experiments to address the research questions presented in the introduction. We found that introducing a Teacher-Student Framework did not have an effect when the imbalance between strongly and weakly labeled data was small, but that the effect was evident when the imbalance was greater.

5.1 Domain & Dataset Discussion

5.1.1 Inter-Observer Annotator Variability

The ground truth is provided by the human experts to evaluate the models in this thesis. This introduces a potential bias, as evaluation becomes dependent on the annotator. Annotating lung tumors is challenging and is prone to high variability between annotators. This type of variance is known as *inter-observer annotator variability*. As a result of this, it is infeasible to achieve perfect DSC on the data used.

There are multiple studies that have investigated this problem; for instance in annotations of prostate tumors from MR images [57] and annotations of brain tumors on CT and CT/PET-fused images [58]. Both studies conclude that there are significant difference between the segmentation annotations performed by different experts.

Figure 5.1 shows the annotations of two different tumors performed by two different experts. The top row can be said to have had a more *liberal* annotation, whereas the bottom row has a more *conservative* annotation. From these annotations, it is reasonable to assume that there would be some disagreement on how to delineate the tumor, especially as the tumor grows into tissue with similar intensities.

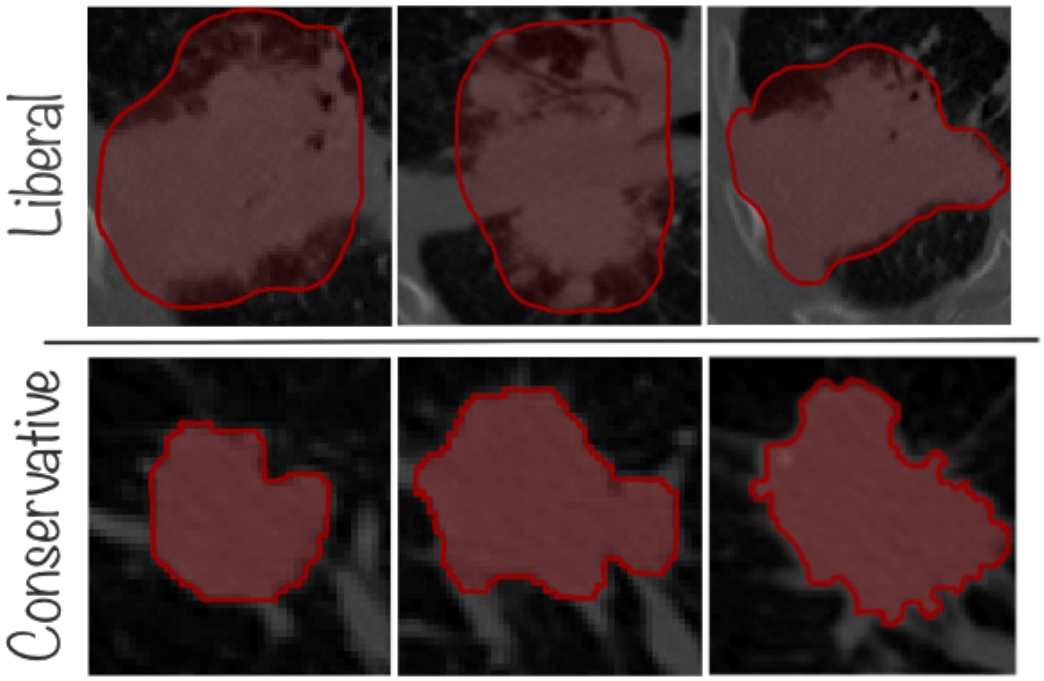


Figure 5.1: Illustration of Segmentation Variability Between Experts. The top row shows a tumor from the Radiomics dataset, whereas the bottom row shows a tumor from the MSD dataset. A curvature has been drawn surrounding the tumor annotations for illustrative reasons.

When evaluating our model, this effect must be taken into consideration. It is therefore not realistic to reach a DSC of 100.0. If a model managed to achieve this, it would imply that the model has learned to replicate the annotator, as the annotator is the gold standard. Even if a model is found to perform slightly poorer, it does not necessarily mean that the model is objectively worse, as the real ground truth is not available. Nonetheless, the annotations should serve as a good indication on lung tumor segmentation performance.

5.1.2 Artifacts in Dataset

The quality of the datasets is mostly sound. However, by randomly extracting samples from the dataset for inspection, some flaws were observed. As can be seen in figure 5.2, in the instance named *Erroneous orientation*, one of the masks were stored with the wrong orientation. By comparing the image and label in the sagittal view, it seems that the label has been flipped around the axial plane.

The instance named *Illogical Delineation*, in the same figure, shows an example of a segmentation that we doubt is correct. It is possible that the annotators had additional information when the segmentation was performed, but it seems unlikely that the tumor has grown part of the bone structure, as can be seen in the figure. With that being said, it is not for us to override the experts' annotations, but it is interesting to note these kinds of artifacts.

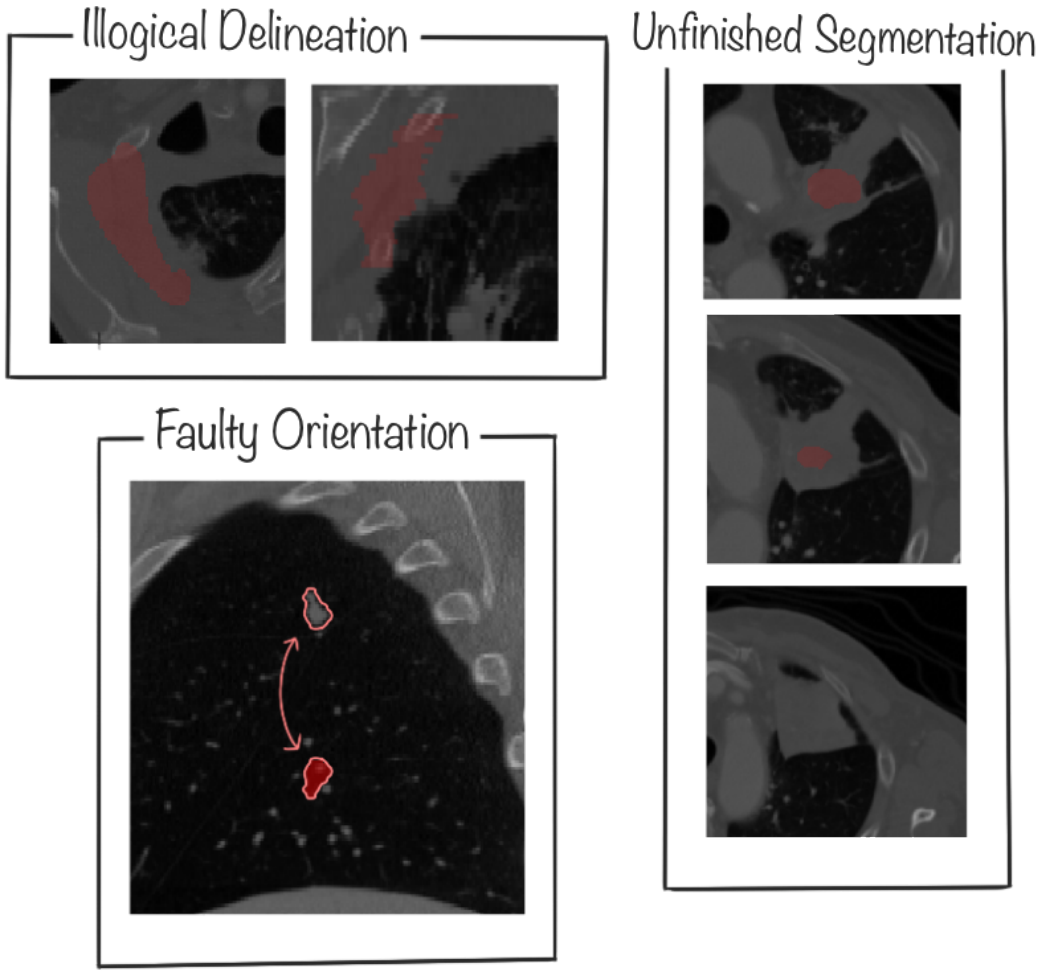


Figure 5.2: Examples of Artifcats in the Datasets

The part in the figure that is labeled *Unfinished Segmentation* shows what seems to be a half-performed segmentation. This scan is part of the Radiomics dataset. At least for the untrained eye, it is hard to see why some parts of the tissue are labeled as tumor, whereas other parts are not. It is however possible that the annotators had additional information available, like PET-scans or similar. In general, we experienced that the annotations in the MSD dataset was sound and consistent, and of much higher quality than the Radiomics dataset.

Although the vast majority of the images in the datasets are correctly labeled, there exist some images with artifacts, like the ones shown in figure 5.2. However, to make evaluation comparable to published papers, we chose to keep the dataset as is, as discarding potentially false or anomaly cases would make the comparison biased.

5.1.3 Limitations of CT

Although CT images are mostly sufficiently informative to locate the tumor, there are scenarios where the lack of information makes it challenging to separate the tumor tissue and the surrounding tissue. It is therefore possible that some scenarios require supplementary information, that CT is unable to capture. PET, MRI, or similar imaging techniques might be necessary to accurately delineate the tumor. Figure 5.3 shows an example from the LungDx dataset where

it is feasable to locate the tumor, but challenging to accurately predict the tumor perimeter without taking the PET image into account.

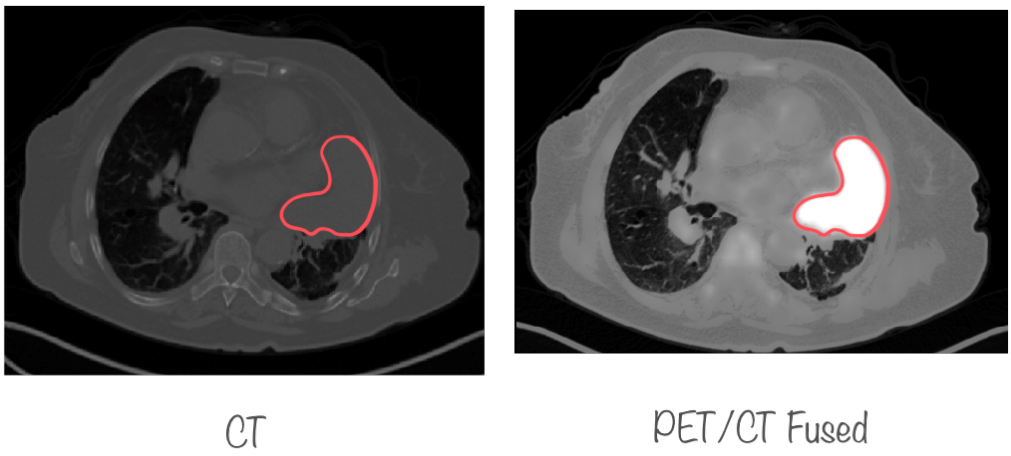


Figure 5.3: Example of the Limitations of CT. The red curvature is added to show where the delineation fits in the CT image. The white glow on the right image is the tumor glowing up in the PET scan.

5.1.4 Data Preprocessing

The **Teacher pipeline** was used to preprocess the data before feeding it to the teacher. Here a $128 \times 128 \times 128$ image was cropped around the tumor from images with voxel spacing of $(0.7 \times 0.7 \times 0.7)\ mm^3$. In a few cases, we observed that the tumors were so large that they did not fit inside this $128 \times 128 \times 128$ image. It is possible that it would be benificial to either crop a larger image or alternatively use a larger voxel spacing to fit the largest tumors within this cube.

The **Student Pipeline**, which was used to preprocess the images fed to the student, did also crop images as part of the process. There are mainly two concerns regarding the cropping around the lungs. First, we observed on occasion, that the tumor was located between the two lungs, effectively being present in both lungs. When the image was preprocessed through the student pipeline, this tumor would often be split into two and treated independently thereafter. Second, if a tumor grows outside of the lungs, it is possible that the tumor was clipped, and thus part of the tumor was lost. This is due to the way we cropped the lungs around a lung mask.

Regarding the first challenge, we experimented with an alternative but similar cropping technique that cropped the image around both lungs and treated both simultaneously. We observed poorer performance when using this preprocessing technique. However, this strategy may be viable if other parts of the pipeline or the model are changed. The second challenge can be partially solved by creating a buffer during cropping. Instead of tightly cropping the lungs, it could be possible to add a fixed amount of voxels in all dimensions to mitigate the challenge of tumors being cropped. This can, in turn, lead to issues with VRAM capacity and possibly that the images become too vast for our models to handle. However, we did not prioritize investigating this cropping technique further.

Common for both preprocessing pipelines are the thresholding and normalization of voxel intensity values. It is possible that by being more conservative with the interval used to threshold, the

model would have fewer irrelevant values to regard. Values that were outside the range [-1024, 1000] were filtered. Normalizing the intensity values with another normalization algorithm could also affect the network's performance, but we believe it could only play a minor role.

5.1.5 Output Post-Processing

The post-processing method is simple as it removes all but the largest component. Although this can reduce the number of FPs drastically, it comes with a risk. One scenario where this method can never reach optimal solutions is when the patient has more than one tumor. At least one TP will be removed in the post-processing step. The post-processing method can also remove TPs when it is only one tumor in the image as well. If the output produces two or more components where the largest component is not the TP, the TP will be removed. On average, we observed that this post-processing method seemed to improve precision but slightly degrade recall.

Nonetheless, this post-processing method can help to better understand the models by comparing the performance before and after post-processing. To increase the performance of the method further, development of another post-processing pipeline should be investigated.

5.2 RQ1: Teacher as a Semi-Automatic Method

Research Question One

RQ1: How does semi-automatic methods, that utilizes either pre-calculated bounding boxes or the center of the tumor as additional input, compare to a fully automatic method that only uses the image as input?

This section is concerned with discussing research question one, and the results seen in section 4.1. The research question is interesting from a clinical perspective as a semi-automatic method can aid experts during tumor delineation to save time. Two types of semi-automatic methods were tested. The first being where the expert selects the center of the tumor and the image is cropped around the center. This method is referred to as *U-Net* in table 4.1. The second method is dependent on bounding boxes created by an expert that indicate the tumor's location in the axial plane in every slice. This method is referred to as *Teacher* in table 4.1. Performing a delineation of the tumor takes longer than drawing bounding boxes in the axial plane. To select the center of the tumor takes even shorter than drawing the bounding boxes. As our results show, the saved time comes with a cost of less accurate delineations.

5.2.1 Discussing the Results

As seen in table 4.1, the teacher model outperformed the baseline U-Net on both datasets with respect to mean DSC. On average, It performed $\mathbf{12.98}\%$ better on the MSD dataset and $\mathbf{43.54}\%$ better on the Radiomics dataset, in terms of DSC. As there are fewer images in the MSD dataset than the Radiomics dataset, the teacher model performed $\mathbf{38.78}\%$ better on average on both datasets. The standard deviation of the teacher's output was lower than the U-Net's. Over the entire dataset, the teacher had a standard deviation of 8.50 before post-processing and 5.42 after post-processing. After post-processing, this was about $\frac{1}{5}$ of the standard deviation of the post-processed U-Net output, which saw a higher standard deviation after post-processing. This indicates that the teacher model was more stable than the U-Net model in its predictions.

By qualitatively evaluating the results, it is possible to understand why the two models perform so differently. In figure 4.1, the outputs of the two models along with the corresponding images and ground truths are shown. The figure shows the output *before* post-processing. In the figure, the second column shows a slice of the image called *Radiomics 89*. As can be observed, the U-Net model has segmented the wrong part of the lung, whereas the teacher model has confidently found the correct candidate tumor. There are quite a few images where the U-Net model completely misses the tumor, but it is not a single instance where this happens to the teacher. We argue that the teacher has learned that it is always a tumor within the bounding boxes. The U-Net model cannot lean on the bounding box and is therefore prone to false positives and false negatives. To make matters worse for the U-Net model, when it correctly identifies the tumor but unfortunately also segments a false positive, the true positive might be removed during post-processing if the false positive tumor is larger than the true positive one. This explains why the standard deviation is higher for the U-Net after post-processing than before. This effect is not seen for the teacher. When the post-processing is applied to the teacher output, this likely removes small artifacts that fall within the given bounding box but are not connected to the tumor. As table 4.1 shows, this has a positive effect on both the mean DSC and the standard deviation.

As discussed, the additional information of the bounding boxes seems to positively affect the performance. However, we observed images where the bounding boxes seem to mislead the teacher as well. In figure 4.1 there are two examples where it seems that the teacher almost blindly trusts the bounding box, likely because it was unable to find a logical delineation inside the bounding box. In the image named *MSD 42*, it seems the U-Net model does a better job segmenting the tumor than the teacher. The teacher's output closely resembles the segbox seen in the same figure. The image named *Radiomics 200* shows a similar effect. In the *MSD 42* image, it is air which is the prominent substance within the segbox. In *Radiomics 200*, however, it is body tissue. It seems that the teacher trusts the bounding boxes regardless of the surrounding tissue whenever it is unable to find the tumor within the box.

The U-Net model performed better than the SOTA method reported in the publication by Carvalho et al. [36] on the MSD dataset. However, on the Radiomics dataset, it performed poorer than the reported DSC in the study by Pang et al. [37] The teacher performed well across both datasets. To the best of our knowledge, no published, fully-automatic method achieved a DSC close to our teacher, indicating that using the pre-calculated bounding boxes is advantageous.

Possible Sources of Error

- **Test set selection**
  Due to limited data, the test set cannot be arbitrary large. It was sampled randomly, but it is plausible that by selecting another test set, the results would differ, resulting in potential selection bias. Especially for the MSD test set, each image contributes to $\frac{1}{9}$ of the average performance.
- **No Controlled Baseline**
  We did not train a fully automatic method ourselves. In the discussion we relied on published papers to compare our semi-automatic models with. Thus, the comparison might be done on the wrong premises, for example differences in the preprocessing pipeline, test set, or other factors, resulting in potential misclassification bias.

5.2.2 Usability

Given the presented results and from inspecting the output on the test set, we argue that a semi-automatic method like the teacher can be useful primarily for two tasks.

- **Faciliate Dataset Creation**
  By using the teacher to annotate datasets, which is currently only annotated with bounding boxes, the available data appropriate for segmentation task learning can be expanded. In addition, this lowers the bar for annotating new datasets. If a slight lack of precision can be tolerated, the teacher can then be used to lower the expense of creating segmentation annotated datasets.
- **Aid Experts in Clinical Use**
  As the masks produced by the teacher is sound, it is possible that the teacher can be used to aid experts performing segmentation. The expert might quickly create the bounding box annotation, then let the teacher perform an initial segmentation based on this. The expert can then inspect the result to verify its quality or enhance the segmentation. The method could be integrated as a plugin into an existing software to ease the process.

5.3 RQ2: Student as a Fully Automatic Method

Research Question Two

RQ2: By expanding the dataset using a Teacher-Student Framework, could the performance of a fully automatic model be increased compared to a model trained purely on strongly annotated data?

This section discusses the results from section 4.2 in relation with research question two. A fully automatic method was trained on both expert-labeled data from the MSD and Radiomics dataset and teacher-annotated data from the LungDx dataset.

Research question two is essential because if models that utilize both expert-labeled and teacher-labeled data perform better than models trained solely on expert-labeled data, it opens up the possibility to utilize larger datasets that have previously been regarded as *too weakly annotated* by introducing a teacher. Furthermore, the Teacher-Student Framework does not dictate the architecture or pipelines that can utilize this effect. Instead, it simply states that introducing a teacher that can generate decent pseudo-labels given extra information can boost the performance of the primary task model.

5.3.1 Discussing the Results

As can be seen in table 4.2, the Single-Channeled Student performed best on the Radiomics dataset and the baseline U-Net performed best on the MSD dataset. The table reveals that the Single-Channeled Student performed best when evaluated on both datasets. Still, as accounted for in section 3.1, the Radiomics dataset is larger than the MSD dataset, making it an unfair comparison when evaluated on both datasets overall. Interestingly, it seems that the proposed Double-Channeled Student did not gain any performance from the additional decoder branch in this experiment.

We argue that tumor volume in the datasets can explain why the U-Net model performed better on MSD and the two other models performed better on Radiomics. The tumor volume

distribution is similar between the Radiomics and the LungDx datasets. The MSD dataset, on the other hand, contains smaller tumors on average. This means that when the two students learned from the teacher-annotated LungDx dataset, they were exposed to more large tumors, which did not necessarily affect performance on the MSD dataset in a positive way, but had a positive impact on performance measured evaluated on the Radiomics dataset. Figure 5.4 shows the average DSC for each model, divided into eight buckets of tumor volume. The U-Net model performed best on the two buckets that contained the smallest tumors, whereas it performed worst on the remaining bucket sizes, except the last bucket. This suggests that the two students perform better than the U-Net model on larger tumors, which supports our initial hypothesis.

To summarize, the teacher-annotated LungDx dataset contains tumors that are similar in size to those found in the Radiomics dataset, but less similar in size as those found in the MSD dataset. Our theory is that this caused the two students to perform better on the Radiomics test set, and the U-Net model trained on only MSD and Radiomics performed better on the MSD test set.

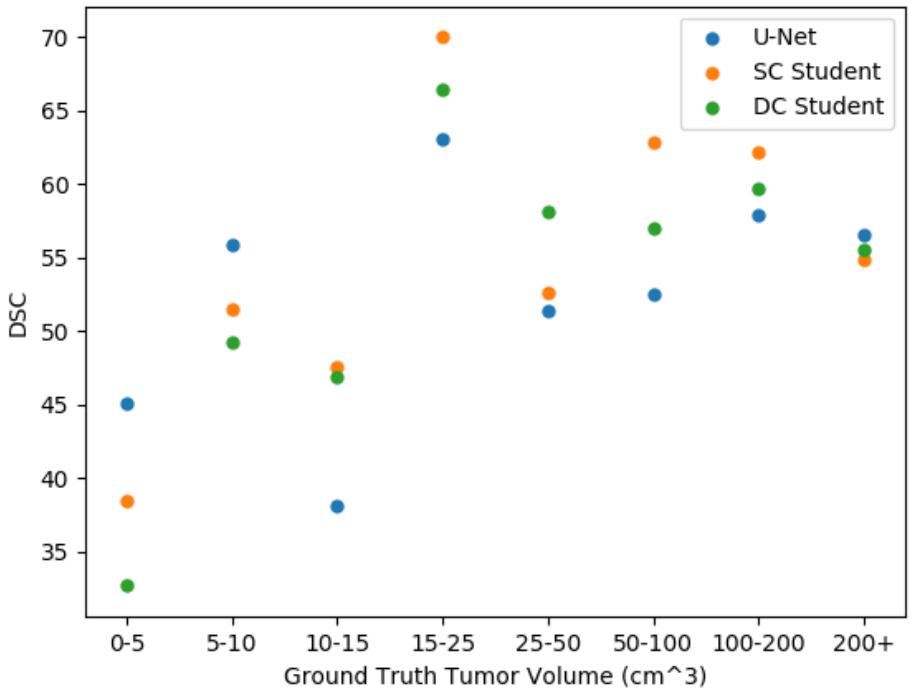


Figure 5.4: Students DSC Related to Tumor Volume on the Test Set. The average DSC is calculated within each bucket.

Our measurements suggests that the balancing of the dataset in terms of tumor volume did not work as well as desired. However, we argue that it is harder to detect and mask smaller tumors. Without performing a second experiment it is hard to quantify how successful the effort to balance the dataset were. These results suggests that our models struggles more with smaller tumors on average.

The standard deviation of the DSC produced by the models in this experiment, are in many cases ~ 30. We argue that this is because there are multiple outliers that have a DSC close to zero. Figure 5.5 shows a boxplot of the DSC evaluated on the whole test dataset. The median

DSC of the U-Net was 58.4, whereas the SC Student and DC Student achieved a median DSC of 66.1 and 61.1, respectively.

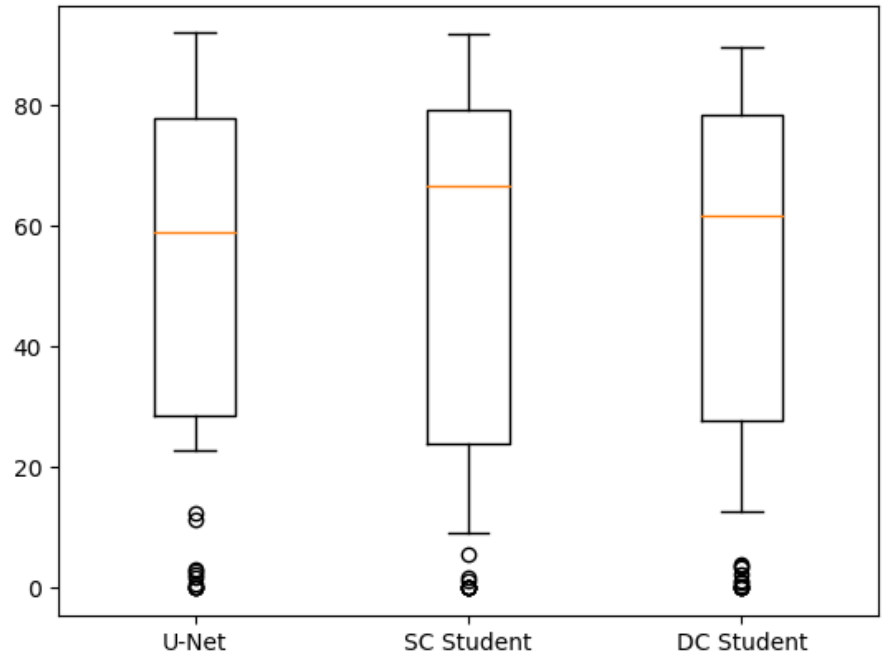


Figure 5.5: A Boxplot of the DSC for the Models in Experiment Two. The median score is indicated by the yellow line, the box indicates the middle 50%.

By qualitatively inspecting the images, we observed that the U-Net was a bit more conservative in its delineations than the two student models. In figure 4.2 three examples of this can be seen. The images MSD 27, MSD 78, and Radiomics 422 shows some examples that the two student models seem to create masks that *bleed out* of the tumor to a certain degree. On the other hand, the U-Net model seems a bit more restrictive and tends to mask a smaller portion of the tumor. This observation is supported by the data seen in table 5.1. The U-Net model creates smaller masks than the students on average.

We measured the TP volume, such that FPs would not affect the calculated volumes. From this table, it can be observed that the U-Net produces the smallest TP tumors on average, whereas the SC Student produces the biggest TP on average. This support what we saw in the images. We suspect this effect might come from the fact that the two models trained on the teacher-annotated data see more often liberal annotations than the baseline model. As discussed in section 5.2.1, when the teacher struggles to see the perimeter of the tumor, it tends to trust the bounding boxes, which in turn may lead to very liberal annotations. This behavior may have lead to the two students learning to create liberal segmentation annotations to a higher degree than the baseline U-Net only trained on supervised data.

Table 5.1: TP Tumor Sizes in Experiment Two

Model	Tumor Size TP
U-Net	57.04
SC Student	62.26
DC Student	59.96

The three different models performed differently in terms of the different metrics as covered. We have highlighted the potential cause of these behaviours. On average, the difference between the baseline model and the two student models were smaller than we anticipated. The students performed better at larger tumor on average, whereas the baseline performed better on smaller tumors. Based on this, the answer to research question two is indecisive. We hypothesis that the imbalance between the strongly and weakly labeled data was not great enough in our dataset. The imbalance seen in all of the publications addressed in section 2.4, except Sun et al. [41], were far greater than ours.

Possible Sources of Error

- **Test Set Selection**
  The same source of error as covered in the previous experiment.
- **Hyperparameter Sensitivity**
  We discovered that our student models were sensitive to changes in hyperparameters. It is possible that by using different configurations, other results can be achieved.

5.4 RQ3: Sparsely Trained Models

Research Question Three

RQ3: When shifting the balance of the dataset towards less strongly annotated data and more weakly annotated data, does this cause the Teacher-Student approach to yield higher performance than the standard fully-supervised approach that only uses the strongly annotated data?

This section is concerned with discussing the results from section 4.3 in relation to research question three. This research question is particularly interesting if one has a small dataset of strongly labeled data, or the resources to create one, and a large weakly labeled dataset at disposal.

The size of the strongly labeled dataset was artificially scaled down to contain only 48 images. Of these 48 images, 40 were appointed to the train set and the remaining 8 to the validation set. This imbalance bear closer resemblance to some of the balances seen in the papers in the related work section than the imbalance seen in the previous experiment. The results of this experiment are discussed in relation to the previous experiments.

5.4.1 Discussing the Results

The results from section 4.3 are divided into two parts; the sparsely trained teacher, and the sparsely trained students. The sparsely trained models were evaluated on the same test set as

the non-sparsely trained models in the preceding experiments. This was done to be able to evaluate the effect of the greater imbalance.

The Sparsely Trained Teacher

Table 4.3 shows the results of the sparsely trained teacher evaluated on the same test set as the non-sparsely trained teacher. The results from the non-sparsely trained teacher are shown in table 4.1. As expected, the teacher trained on more data achieved a better performance than the sparsely trained teacher. This was expected as more data generally give better results, assuming equal quality between the two sets. However, the difference between the two teachers were smaller than we anticipated. The non-sparsely teacher was trained on a dataset more than ten times larger than the sparsely trained teacher, but only achieved $\mathbf{4.0}\%$ higher DSC on the MSD dataset and $\mathbf{2.9}\%$ higher on the Radiomics dataset. Interestingly, the difference between the baseline U-Net and the teacher model was higher for the sparsely trained models than the models trained on all available data. The sparsely trained teacher performed $\mathbf{68.3}\%$ better on the MSD dataset than the sparsely trained U-Net before post-processing, and $\mathbf{93.2}\%$ better on the Radiomics dataset. This suggests that the performance of the sparsely trained teacher is even more dependent on the bounding boxes than the non-sparse teacher. The explanation to why the teacher performed better than the baseline are previously discussed.

The Sparsely Trained Students

The non-sparsely trained students performed similar to each other, and to the baseline U-Net. The sparsely trained students, on the other hand, achieved very different results compared to each other, and to the sparsely trained U-Net model, as can be observed in table 4.4.

On the MSD datset, the best performing model in terms of DSC was the SC Student after post-processing, which performed **153.1%** better than the U-Net model. The DC Student performed similar to the SC Student. A similar observation was done on the Radiomics dataset, although the DC Student achieved a better DSC than the SC Student on this dataset. Both the students performed over **80%** better than the baseline model on the Radiomics dataset. The standard deviation was similar among the three models when evaluated on both datasets. However, the boxplot shown in figure 5.6 suggests that there are different reasons for this high standard deviation. Outliers of DSCs close to zero caused the high standard deviation of the students, whereas the standard deviation of the baseline model was affected by the few good masks it was able to produce.

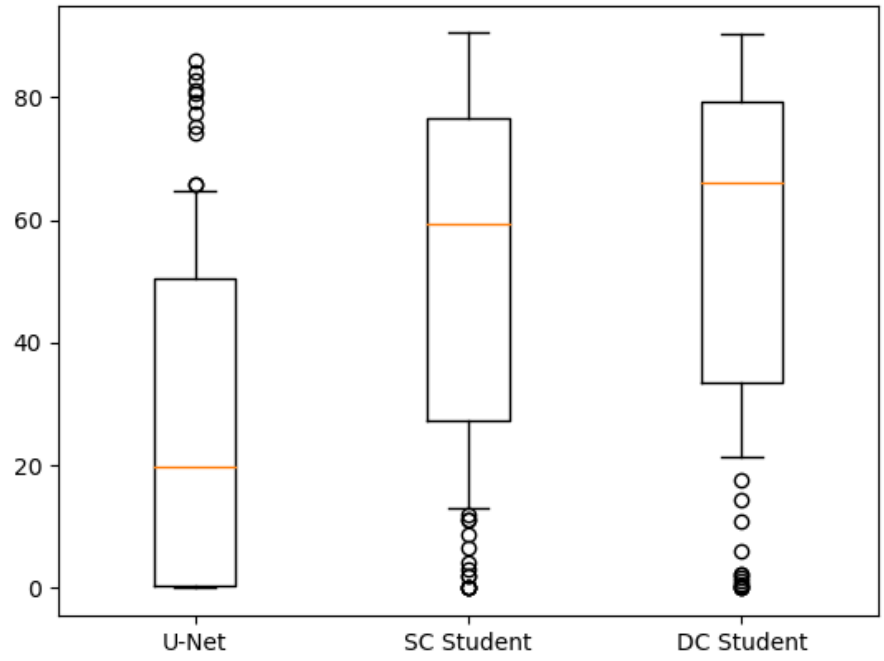


Figure 5.6: A Boxplot of the DSC for the Models in Experiment Three. The median score is indicated by the yellow line, the box indicates the middle 50%.

When the U-Net is evaluated on the MSD dataset with the DSC-TP metric, it seems to perform well. It had a higher DSC-TP score than the SC Student. By evaluating this in context with the low recall of the U-Net, it might indicate that it only finds the tumors that are easy to mask, whereas the two students also mask the complex tumors, which can negatively affect DSC-TP. Evaluated on the Radiomics dataset, it can be observed that the U-Net achieved a higher recall. Consequently, it achieved a much lower DSC-TP compared to the students. This observation supports the hypothesis that the high DSC-TP score on the MSD dataset is a consequence of the model only detecting the easy tumors, which is indicated by the low recall achieved.

As can be seen, the students have a significantly higher F1 Score than the U-Net. The two students, however, performed similar in terms of F1 score. Although the SC Student achieved a perfect F1 score of 100.0, the DC Student performed better on the Radiomics dataset. We expected the additional segbox branch in the DC Student to yield higher recall, but we cannot draw this conclusion from our experiments.

Our sparsely trained SC Student achieved a higher DSC, 71.56, than the SOTA results reported by Carvalho et al., 70.9, on the MSD dataset. However, the result achieved in this project is not in line with the official rules of the competition associated with the dataset. First of all, the images used in our test set were not the same as the official test set. This is because the official test set did not include annotations, making it impossible for us to evaluate our models on that particular dataset without attending the competition. Second, the competitors of the MSD challenge had to make a model that performed segmentation on all of the ten categories. Our models have only learned to segment pulmonary tumors. Whether it is allowed to utilize additional datasets to train the models is unclear to us, but this part of our project may be in

the gray area regarding the challenge's rules as well.

The previous experiment did not result in a clear difference between the students and the baseline U-Net model in terms of performance. However, this experiment resulted in considerable difference between the two students and the baseline U-Net model. The sparsely trained SC Student achieved a higher score than the best performing non-sparsely trained model, even achieving a DSC higher than the current reported SOTA. Although the comparison with the performance reported by Carvalho et al. is not completely fair, it is interesting that a model trained on such a limited amount of strongly labeled data can achieve a result close to the SOTA performance.

Possible Sources of Error

- **Test Set Selection**
  A different test set could impact the performance of the models. Given that the test set is similar between our experiments, similar conclusions could be drawn. However, a different test set could impact the conclusions related to other published work, as the performance could be impacted.
- **Teacher Train Set Selection**
  As the teacher train set is of such a limited size, it is very important that these images represent the whole dataset. If these images are not representative of the whole dataset, then the teacher might not generate good pseudo-soft labels. The teacher's annotation, in turn, influences the students ability to learn. Thus resulting in, potentially, poor performing students.
- **Hyperparameter Sensitivity**
  As previously covered, our student models seems sensitive to changes in the hyperparameters.

5.5 Retrospective Evaluation

This section is concerned with evaluating aspects of the way we carried out the project. Overall, we are satisfied with the progression and the results we got, but we are also left with the impression that there are certain things that could have been done differently.

Positive Aspects

- **Efficient Implementation of Working Pipeline**
  We decided early on to create a working pipeline. This enabled us to quickly start development, and even create some initial experiments related to the teacher.
- **Using MONAI**
  In retrospective, we identify the decision to use MONAI as one of the best decisions we made. MONAI allowed us to rapidly develop and test different preprocessing pipelines and architectures.
- **Modular Pipeline**
  From the beginning we decided to keep the pipeline modular such that we could experiment with multiple different architectures, datasets, and preprocessing pipelines efficiently.
- **Standardized Test Set Across all Experiments**
  Choosing a test set that was identical for all experiments enabled us to draw conclusion based on observations across different experiments. This was especially important to accurately conclude on research question three.
- **Contributing to the Open-Source Community**
  We are happy that we could publish one of our models for other researches to use.

Constructive Aspects

- **Poor Productivity during Model Training**
  During training of our models we experienced low productivity. We tried to spend this time by documenting our project, but in hindsight we acknowledge that we could have utilized these periods to better plan for upcoming phases. However, as the results of the current model training often dictated the next phase, we felt it was challenging to plan ahead.
- **Poor Internal Organization of Experiments**
  As the project developed, a vast number of experiments were conducted. Initially, we had a structure for all the experiment configurations. However, with the number of experiments rapidly growing, we experienced that our system became cluttered. A better folder structure could have helped solve the problem, for example organizing the experiments by week.

Unfortunate Events

- **High Traffic on Computing Resources**
  There were periods characterized with low progress because of queue on the computing resources. At one point, the machine Idun was under maintenance leading to long compute queues in the following weeks.
- **Virtual Machine got Deleted**
  At one point one of our virtual machines were deleted from a shared computer. This virtual machine contained large amounts of processed data, which set us back a few days. Some of the data was recovered from backups.

CHAPTER 6

CONCLUSION & FUTURE WORK

6.1 Conclusion

In this thesis, the effect of using a Teacher-Student Framework to perform segmentation of lung tumors in CT images were explored. To explore this, three research questions were formed. The first research question was related to how semi-automatic methods could utilize additional information to make accurate segmentations of tumors.

Research Question One

RQ1: How does semi-automatic methods, that utilizes either pre-calculated bounding boxes or the center of the tumor as additional input, compare to a fully automatic method that only uses the image as input?
A: A semi-automatic method can achieve significantly higher performance by utilizing predefined bounding boxes during lung tumor segmentation than both a semi-automatic method utilizing the tumor center, and a fully-automatic method. Our results can not conclude whether a semi-automatic method that utilize the tumor center will perform better than a fully-automatic method in general.

Research question two was concerned with whether introducing a teacher to expand the available dataset could improve a fully-automatic method. We used a teacher to expand our strongly labeled dataset and compared students trained on this expanded dataset, to a baseline model trained only on human-annotated segmentations.

Research Question Two

RQ2: By expanding the dataset using a Teacher-Student Framework, could the performance of a fully automatic model be increased compared to a model trained purely on strongly annotated data?
A: Given that the imbalance of the strongly vs. weakly annotated data is great enough, our results indicate that allowing a teacher to create pseudo-labels can help fully automatic models achieve an increased performance compared with models trained solely on limited expert-labeled data. However, we did not find evidence that the Teacher-Student Framework helped increase the performance when the dataset was increased from ~ 500 images to ~ 1000 images. To quantify exactly when this framework is beneficial, more research is needed.

The third and final research question was related to exploring whether the Teacher-Student Framework had a stronger effect if the imbalance of strongly and weakly labeled data was increased. We artificially shrunk our strongly labeled dataset and trained a new teacher and students to research this effect.

Research Question Three

RQ3: When shifting the balance of the dataset towards less strongly annotated data and more weakly annotated data, does this cause the Teacher-Student approach to yield higher performance than the standard fully-supervised approach that only uses the strongly annotated data?
A: Our results shows that when the strongly labeled data is of very limited size and the weakly labeled data is vast, the effect of a Teacher-Student Framework as proposed is increased significantly compared with scenarios where the two datasets are more equal in size.

The goal of this thesis was to investigate the effect of a Teacher-Student Framework on lung tumor segmentation of CT images. We conclude that this strategy is beneficial if the imbalance between strongly and weakly annotated data is great. To quantify the imbalance needed for this framework to be beneficial, more research must be conducted.

6.2 Future Work

Our goal was to investigate the effect of Teacher-Student framework, we did not focus on maximizing the DSC. However, during our work with the project we thought of some ideas which we think might push the performance a bit further.

6.2.1 Further Teacher-Student Research

Further Increase Weakly Labeled Dataset Size

It would be interesting to research if the performance of the automatic methods could be considerably increased by introducing more weakly annotated data. If a large dataset could be acquired, it would be trivial to continue the training performed in this project to research the impact of a large, weakly annotated dataset.

Multi-Teacher - Single Student

The results presented in this thesis uncovers that the the sparsely trained students performed considerably better than expected, even compared to the non-sparsely trained students. This observation led to the idea of using multiple teachers trained on different subsets of the strongly labeled data. The different teachers can then create pseudo-labels on the weakly labeled data. All teachers create pseudo-labels on all the weakly labeled data. This would result in slight variances between the ground truth created by each teacher because they are all trained on a different subset. This might lead to a more generalized student, a form of natural data augmentation. Finding more data can be challenging. It is possible that this approach can further improve the generalization of the students without the need of expanding the dataset further.

DC Student without Teacher

To further investigate the impact of introducing a teacher to a mixed supervision framework, we suggest training a third model to compare with the results presented in this thesis. By training a DC Student on both the strongly and weakly supervised data, but without having a teacher label the weakly labeled data, it can help understand the importance of the teacher. Since the weakly labeled data have bounding boxes and the DC Student can produce bounding box output and be backpropagated based on them, the teacher is not necessary to utilize both the strongly labeled data and the weak labels. However, if this proposed DC Student performs worse than the one presented in this thesis, this might suggest yet again that introducing a teacher is beneficial to get the most out of the weakly labeled dataset.

6.2.2 Improving the Methods

Improved Post-Processing Method

The risks of the proposed post-processing method are previously discussed. Other post-processing pipelines can be used with the overall method proposed in this thesis. For instance, instead of removing all but the largest object, removing all objects smaller than a predefined value makes sense. This way, the risk of removing TPs can be lowered compared with the current post-processing method, but the noise in the output is still reduced, effectively lowering the FPs.

Different Window-level in Preprocessing

In our preprocessing pipelines, HU-intensities below -1024 and above 1000 was filtered. Experimenting with different thresholds to remove negligible intensities of the image can potentially reduce the complexity of the task for the models.

BIBLIOGRAPHY

[1] National Cancer Institute. Common Cancer Types. https://www.cancer.gov/types/common-cancers, April 2021.

[2] Per Holck. Lungene i Store medisinske leksikon. https://sml.snl.no/lungene, December 2020.

[3] Americal Lung Association. Lung Cancer Fact Sheet. https://www.lung.org/lung-health-diseases/lung-disease-lookup/lung-cancer/resource-library/lung-cancer-fact-sheet, May 2020.

[4] Kreftregisteret. Lungekreft. https://www.kreftregisteret.no/Temasider/kreftformer/Lungekreft/, May 2021.

[5] American Cancer Society. Lung Cancer Early Detection. https://www.cancer.org/cancer/lung-cancer/detection-diagnosis-staging/detection.html, April 2021.

[6] Cancer Research UK. Why is early diagnosis important? https://www.cancerresearchuk.org/about-cancer/cancer-symptoms/why-is-early-diagnosis-important, January 2021.

[7] Americal Lung Association. Lung Cancer Basics. https://www.lung.org/lung-health-diseases/lung-disease-lookup/lung-cancer/learn-about-lung-cancer/lung-cancer-basics, February 2021.

[8] Amber L. Simpson, Michela Antonelli, Spyridon Bakas, Michel Bilello, Keyvan Farahani, Bram van Ginneken, Annette Kopp-Schneider, Bennett A. Landman, Geert Litjens, Bjoern H. Menze, Olaf Ronneberger, Ronald M. Summers, Patrick Bilic, Patrick Ferdinand Christ, Richard K. G. Do, Marc Gollub, Jennifer Golia-Pernicka, Stephan Heckers, William R. Jarnagin, Maureen McHugo, Sandy Napel, Eugene Vorontsov, Lena Maier-Hein, and M. Jorge Cardoso. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *CoRR*, abs/1902.09063, February 2019.

[9] Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig. User-Guided 3D Active Contour Segmentation of Anatomical Structures: Significantly Improved Efficiency and Reliability. *Neuroimage*, 31(3):1116–1128, 2006.

[10] American College of Radiology (ACR) and Radiological Society of North America (RSNA). Mammography. https://www.radiologyinfo.org/en/info.cfm?pg=mammo, April 2019.

[11] International Society for Computed Tomography (ISCT). Half A Century In CT: How Computed Tomography Has Evolved. https://www.isct.org/computed-tomography-blog/2017/2/10/half-a-century-in-ct-how-computed-tomography-has-evolved, October 2016.

[12] Frank Gaillard and Kyle Greenway et al. Hounsfield unit. https://radiopaedia.org/articles/hounsfield-unit.

[13] Standford University. Artificial Intelligence Index 2019 Annual Report. 2019.

[14] Antoine Buetti-Dinh, Vanni Galli, Sören Bellenberg, Olga Ilie, Malte Herold, Stephan Christel, Mariia Boretska, Igor V. Pivkin, Paul Wilmes, Wolfgang Sand, Mario Vera, and Mark Dopson. Deep neural networks outperform human expert's capacity in characterizing bioleaching bacterial biofilm composition. *Biotechnology Reports*, 22:e00321, 2019.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *CoRR*, abs/1502.01852, February 2015.

[16] Jason Brownlee. Loss and Loss Functions for Training Deep Learning Neural Networks. https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/, January 2019.

[17] Aditya Anaanthram. Random Initialization For Neural Networks: A Thing Of The Past. https://towardsdatascience.com/random-initialization-for-neural-networks-a-thing-of-the-past-bfcdd806bf9e, February 2018.

[18] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf, 2010.

[19] Michael Nielsen. How the backpropagation algorithm works. http://neuralnetworksanddeeplearning.com/chap2.html, December 2019.

[20] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952.

[21] Sebastian Ruder. An overview of gradient descent optimization algorithms. https://ruder.io/optimizing-gradient-descent/, January 2016.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2017.

[23] Jason Brownlee. How to Avoid Overfitting in Deep Learning Neural Networks. https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/, December 2018.

[24] Raimi Karim. Intuitions on L1 and L2 Regularization. https://towardsdatascience.com/intuitions-on-l1-and-l2-regularisation-235f2db4c261, December 2018.

[25] Jason Brownlee. A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/, August 2019.

[26] Dominic Masters and Carlo Luschi. Revisiting Small Batch Training for Deep Neural Networks. *CoRR*, abs/1804.07612, 2018.

[27] Jason Brownlee. A Gentle Introduction to Batch Normalization for Deep Neural Networks. https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/, January 2019.

[28] Mayank Agarwal. Batch Normalization, Instance Normalization, Layer Normalization: Structural Nuances. https://becominghuman.ai/all-about-normalization-6ea79e70894b, August 2020.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*, abs/1505.04597, 2015.

[30] Austin Ray. Lung Tumor Segmentation via Fully Convolutional Neural Networks. 2016.

[31] Fabian Isensee, Paul Jaeger, Simon Kohl, Jens Petersen, and Klaus Maier-Hein. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature Methods*, 18:1–9, 02 2021.

[32] Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge. *CoRR*, abs/1811.02629.

[33] The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). https://pubmed.ncbi.nlm.nih.gov/25494501/.

[34] Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. https://pubmed.ncbi.nlm.nih.gov/28872634/.

[35] Uday Kamal, Abdul Muntakim Rafi, Rakibul Hoque, Jonathan Wu, and Kamrul Hasan. Lung Cancer Tumor Region Segmentation Using Recurrent 3D-DenseUNet. *CoRR*, abs/1812.01951, 2020.

[36] Joao Carvalho, José Moreira, Mário Figueiredo, and Nickolas Papanikolaou. Automatic Detection and Segmentation of Lung Lesions using Deep Residual CNNs. In *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 977–983, October 2019.

[37] Shuchao Pang, Anan Du, Xiaoli He, Jorge Díez, and Mehmet Orgun. Fast and Accurate Lung Tumor Spotting and Segmentation for Boundary Delineation on CT Slices in a Coarse-to-Fine Framework. pages 589–597, 12 2019.

[38] David Bouget, Arve Jørgensen, Gabriel Kiss, Håkon Leira, and Thomas Langø. Semantic segmentation and detection of mediastinal lymph nodes and anatomical structures in ct data for lung cancer staging. *International Journal of Computer Assisted Radiology and Surgery*, 14:1–10, 03 2019.

[39] Stine Hansen, Samuel Kuttner, Michael Kampffmeyer, Tom-Vegard Markussen, Rune Sundset, Silje Kjærnes Øen, Live Eikenes, and Robert Jenssen. Unsupervised supervoxel-based lung tumor segmentation across patient scans in hybrid PET/MRI. *Expert Systems with Applications*, 167:114244, 2021.

[40] Pawel Mlynarski, Hervé Delingette, Antonio Criminisi, and Nicholas Ayache. Deep learning with mixed supervision for brain tumor segmentation. *Journal of Medical Imaging*, 6(03):1, Aug 2019.

[41] Liyan Sun, Jianxiong Wu, Xinghao Ding, Yue Huang, Guisheng Wang, and Yizhou Yu. A Teacher-Student Framework for Semi-supervised Medical Image Segmentation From Mixed Supervision. *CoRR*, abs/2010.12219, 2020.

[42] Patrick Bilic, Patrick Ferdinand Christ, Eugene Vorontsov, Grzegorz Chlebus, Hao Chen, Qi Dou, Chi-Wing Fu, Xiao Han, Pheng-Ann Heng, Jürgen Hesser, Samuel Kadoury, Tomasz Konopczynski, Miao Le, Chunming Li, Xiaomeng Li, Jana Lipkovà, John Lowengrub, Hans Meine, Jan Hendrik Moltz, Chris Pal, Marie Piraud, Xiaojuan Qi, Jin Qi, Markus Rempfler, Karsten Roth, Andrea Schenk, Anjany Sekuboyina, Eugene Vorontsov, Ping Zhou, Christian Hülsemeyer, Marcel Beetz, Florian Ettlinger, Felix Gruen, Georgios Kaissis, Fabian Lohöfer, Rickmer Braren, Julian Holch, Felix Hofmann, Wieland Sommer, Volker Heinemann, Colin Jacobs, Gabriel Efrain Humpire Mamani, Bram van Ginneken, Gabriel Chartrand, An Tang, Michal Drozdzal, Avi Ben-Cohen, Eyal Klang, Marianne M. Amitai, Eli Konen, Hayit Greenspan, Johan Moreau, Alexandre Hostettler, Luc Soler, Refael Vivanti, Adi Szeskin, Naama Lev-Cohain, Jacob Sosna, Leo Joskowicz, and Bjoern H. Menze. The Liver Tumor Segmentation Benchmark (LiTS). *CoRR*, abs/1901.04056, 2019.

[43] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-Training With Noisy Student Improves ImageNet Classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2020.

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[45] Soham Gadgil, Mark Endo, Emily Wen, Andrew Y. Ng, and Pranav Rajpurkar. CheXseg: Combining Expert Annotations with DNN-generated Saliency Maps for X-ray Segmentation. *CoRR*, abs/2102.10484, 2021.

[46] Antoine Rosset, Luca Spadola, and Osman Ratib. OsiriX: An Open-Source Software for Navigating in Multidimensional DICOM Images. *Journal of digital imaging : the official journal of the Society for Computer Applications in Radiology*, 17:205–16, October 2004.

[47] H. J. W. L. Aerts, L. Wee, E. Rios Velazquez, R. T. H. Leijenaar, C. Parmar, P. Grossmann, et al. Data From NSCLC-Radiomics [Data set]. The Cancer Imaging Archive. https://wiki.cancerimagingarchive.net/display/Public/NSCLC-Radiomics, 2019.

[48] H. Aerts, E. R. Velazquez, R. Leijenaar, C. Parmar, P. Grossmann, Sara Cavalho, J. Bussink, R. Monshouwer, Benjamin Haibe-Kains, D. Rietveld, F. Hoebers, M. Rietbergen, C. R. Leemans, A. Dekker, John Quackenbush, R. Gillies, and P. Lambin. Decoding tumour phenotype by noninvasive imaging using a quantitative radiomics approach. *Nature Communications*, 5, June 2014.

[49] Smith K Freymann J Kirby J Koppel P Moore S Phillips S Maffitt D Pringle M Tarbox L Prior F Clark K, Vendt B. The Cancer Imaging Archive (TCIA): Maintaining and Operating a Public Information Repository. https://link.springer.com/article/10.1007/s10278-013-9622-7, 2013.

[50] P. Li, S. Wang, T. Li, J. Lu, Y. HuangFu, and D. Wang. A Large-Scale CT and PET/CT Dataset for Lung Cancer Diagnosis (Lung-PET-CT-Dx) [Data set]. https://doi.org/10.7937/TCIA.2020.NNC2-0461, 2020.

[51] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, et al. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[52] Kikinis R, Pieper SD, and Vosburgh K. 3D Slicer: a platform for subject-specific image analysis, visualization, and clinical support. Intraoperative Imaging Image-Guided Therapy, 2014.

[53] Thomas Phil. Sikerdebaard/dcmrtstruct2nii: v1.0.19 (v1.0.19) [computer software]. zenodo. https://doi.org/10.5281/ZENODO.4037865, September 2020.

[54] Johannes Hofmanninger, Forian Prayer, Jeanny Pan, Sebastian Röhrich, Helmut Prosch, and Georg Langs. Automatic lung segmentation in routine imaging is primarily a data diversity problem, not a methodology problem. *European Radiology Experimental*, 4:50, 08 2020.

[55] Eric Kerfoot, James Clough, Ilkay Oksuz, Jack Lee, Andrew P. King, and Julia A. Schnabel. Left-Ventricle Quantification Using Residual U-Net. In *Statistical Atlases and Computational Models of the Heart. Atrial Segmentation and LV Quantification Challenges*, pages 371–380, Cham, 2019. Springer International Publishing.

[56] Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. *CoRR*, abs/1912.05848, 2019.

[57] Michael Y. Chen, Maria A. Woodruff, Prokar Dasgupta, and Nicholas J. Rukin. Variability in accuracy of prostate cancer segmentation among radiologists, urologists, and scientists. *Cancer Medicine*, 2020.

[58] Adam C Riegel, Anthony M Berson, Sylvie Destian andTracy Ng, Lawrence B Tena, Robin J Mitnick, and Ping S Wong. Variability of gross tumor volume delineation in head-and-neck cancer using CT and PET/CT fusion. 2006.

APPENDIX A

MONAI - MEDICAL OPEN NETWORK FOR AI

MONAI is an open source framework for deep learning in the medical domain[1]. It is based on the popular machine learning framework PyTorch. There are many operations that are common between medical imaging tasks. MONAI aims to standardize the workflow when dealing with machine learning in the medical domain. They do this by offering a package of methods ranging from metric-calculations to end-to-end engines for training and validating models. As most classes in MONAI are inherited directly from torch classes, it is easy to make modifications to these pipelines with torch methods whenever needed. This way, developers may choose to use MONAI for some parts of their code, while they maintain full control over other parts of their pipeline by implementing it from scratch in torch. MONAI is available as a python package and can be installed by pip. Alternatively the entire code base is openly available on MONAI's github[2].

In this project, MONAI was mainly used for networks and data-processing tasks, such as loading of images, preprocessing and post-processing. Preprocessing and data augmentation are a huge part of deep learning. MONAI offers many preprocessing and data augmentation techniques that are relevant to medical imaging. Most of the methods utilized from the MONAI framework are listed in table A.1

[1] https://monai.io/
[2] https://github.com/Project-MONAI/MONAI

Table A.1: List of Monai Methods used in this Thesis

Method	Description
LoadImage	Loads the image from the specified file. This method supports a wide range of medical image file formats, including NIfTI that was used in this project.
ThresholdIntensity	Is used to filter out intensity values outside a specified range. Useful when dealing with CT-scans where we know that all relevant tissue has a intensity value lower than 1024 and can therefore filter out every abnormality above this intensity.
NormalizeIntensity	Normalizes the image with respect to intensity. Necessary because of different settings on CT machines.
AddChannel	Adds a channel to the 'tensor'. Similar to Unsqueeze(0) in PyTorch.
Resize	Resizes the image to some specified dimensions. Supports multiple modes, including nearest neighbour and trilinear interpolation.
RandFlip	Flips the image in a specified axis with a specified probability.
RandRotate	Rotates the image in a specified plane with a specified probability
Spacing	Interpolates the image to a specified voxel spacing
SpatialCrop	Crops a 3D volume from image around a specified center with a specified height, width, and depth
DivisiblePad	Pads the image in every dimension such that it is divisible by a specified integer
KeepLargestConnectedComponent	Removes all but the largest connected component
ToTensor	Converts the image to a torch tensor
ToNumpy	Converts the image array into a numpy array

MONAI enabled us to easily test multiple types of data augmentation and to streamline our pipeline efficiently. With MONAI's *Compose()* method it is possible to specify a list of methods that should be performed on the data every time an image is requested from a dataset. Listing A.1 shows how to configure a pipeline that loads an image, thresholds its intensities, normalizes the image with respect to intensity, scales the image to desired dimensions, and spatially augments the image with a probability of 0.5. Loading an image from disk before performing all the specified transforms with only one line is then possible.

Listing A.1: Monai Compose Transforms

```
transforms = Compose(
    [
        LoadImage(),
        ThresholdIntensity(threshold=1024, cval=1024),
        NormalizeIntensity(nonzero=True, channel_wise=True),
        AddChannel(),
        Resize(spatial_size=(128, 128, 128),
        RandFlip(prob=0.5, spatial_axis=0),
        RandRotate90(prob=0.5, spatial_axes=(0, 1)),
        ToTensor()
    ]
)

image = transforms('C:\dataset\my_image.nifti')
```

APPENDIX B

THESIS WORK OVERVIEW

Naturally, we invested quite a lot of time and energy into our thesis. We continuously discussed which tasks should be completed at different intervals to stay on schedule. We had bi-weekly meetings with our supervisors from NTNU and SINTEF. Experts from St. Olavs hospital were occationally present in the meetings, mainly during the planning phases. We mapped the current progress during these meetings, discussed our results continuously, and decided on what would be the intelligent next move. Figure B.1 summarizes a timeline of our work throughout the project. Though the tasks are assigned along the timeline, no task is usually completed linearly like the timeline signals. Most of the phases placed along the timeline were conducted iteratively throughout the project.

Both of us were involved in all the parts of the project. Over time it naturally evolved into different primary responsibilities. For instance, it was natural that one person had control over the data processing, as it was superfluous that both learned all the frameworks and structures of the medical images in depth. Generally speaking, both of us were involved in all parts of the project from the start until the end.

We want to emphasize that every bit of the code is written entirely by ourselves. We used powerful libraries like PyTorch, MONAI, Numpy, and the open source lung mask repository published by Hofmanninger et al [54]. Regarding the thesis, every single chapter is written entirely by us, and all illustrations are drawn by hand by us to preserve a consistent styling throughout the thesis.

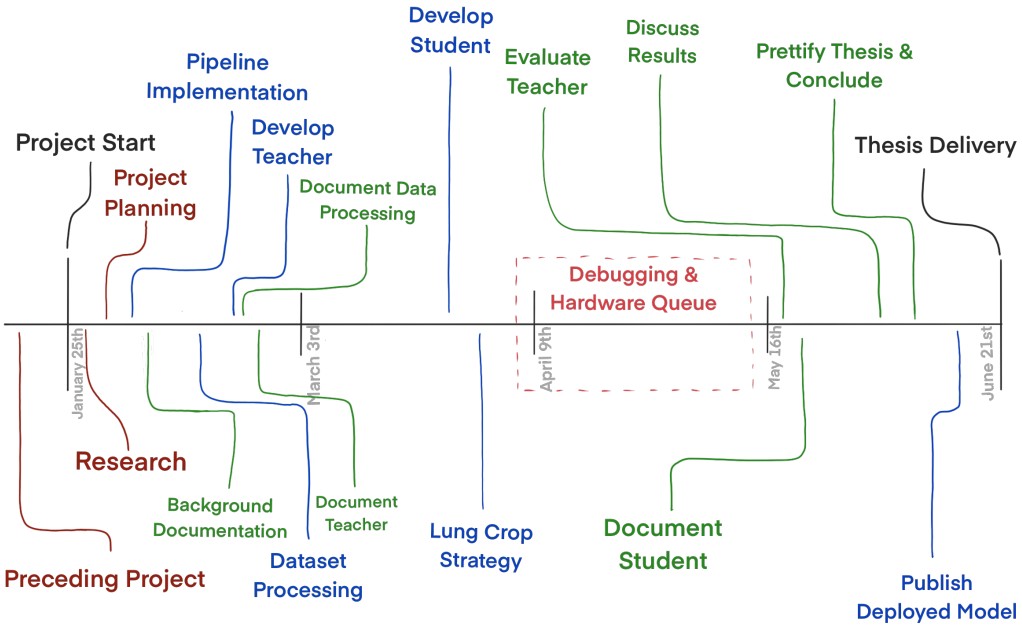


Figure B.1: Project Timeline. The timeline shows roughly when different phases was conducted during the project. Code development is highlighted in blue, thesis documentation is highlighted in green and miscellaneous tasks is highlighted in red.

As figure B.1 shows, a preceding project was conducted before the official project start. This project was performed the semester before. The project's goal was to familiarize ourselves with techniques and concepts relevant for our master thesis. During this project, we implemented and trained two different U-Net-inspired networks that segmented lungs and lung lobes from CT-scans. We learned how segmentation tasks were solved and gained in-depth experience with the PyTorch framework. All code was written from scratch during this project, more or less only using PyTorch and Numpy as third-party libraries. This project gave us great insight into the medical imaging field and 3D semantic segmentation solutions.

During the first phase of this project, an excessive research and planning job was done. We had several meetings with the supervisors and discussed different concepts and strategies worth investigating. It was a strong desire from SINTEF that we looked into the Teacher-Student approach, because of recent publications with promising results. Experts at St. Olavs hospital were interested in effective algorithms to detect and segment lung tumors. During our research phase, it became clear that no publications existed that investigated the use of a Teacher-Student framework for improving the performance of lung tumor segmentation models. We found publicly available datasets with both segmentation annotations and bounding box annotations. The sum of these discussions and discoveries led us to pursue the investigation of how mixed supervision with a teacher-student framework could affect the performance of simple convolutional networks.

At an early stage, right after the initial planning and research phase, it was decided that it would probably be wise to get a basic training pipeline running. This way, we could iteratively debug step-by-step, and it would be more manageable to monitor the progress. The first working training pipeline ready to read NIfTI-images and perform training was running within days. This part contains implementing all the steps in the training loop, configuring loss functions, optimizers, hyperparameters, and implementing standard U-Net using the fundamental building blocks of

PyTorch. While all image processing was coded from scratch using numpy in the preceding project, we used MONAI for data processing tasks during this project. It offers standardized methods for loading, transforming, and augmenting medical images. This framework enabled us to test multiple processing pipelines quickly and helped us develop the pipeline rapidly.

As the figure illustrates, documentation of our method was pretty much done continuously throughout the project. Our preceding project report heavily inspires the background theory chapter. The two projects share the same fundamental concepts and technologies like neural networks and the medical domain. As for the documentation of our method, it was continuously written and updated according to how our implementation changed.

During the implementation of our models, quite a lot of time went into debugging and running multiple training sessions to figure out the correct configurations. The figure shows that during particularly one period, most of the time went into debugging and waiting for compute time. During this phase, there was much traffic on the machines we had available at that time. This held us back quite a lot because we were unable to perform training of our networks. Fortunately, we were given access to a third computer. We did also reach a point where our models suddenly performed worse than what they did previously. Because the new machine had more VRAM available, we decided to exploit this by having images with a higher resolution. To our surprise, it seems that this had the counter-intuitive consequence that our models could not learn effectively on these vast images. We scaled the images down to the old voxel spacing and got our expected results back. During a project like this, there are several encounters like this that one never sees coming that halts the progress for days.

During the last weeks before the deadline, the results from the different models were documented and discussed, and the thesis was controlled and corrected. Finally, a plug-and-play solution for performing lung tumor segmentation was published openly on GitHub. This repository was inspired by Hofmanninger et al's lungmask GitHub repository [54]. His solution enabled us to, in a plug-and-play fashion, utilize his models in our solution. Our idea was to give back to the community the same way he did. Even though our models are not ready for clinical use, they might be helpful in a future thesis or research project.

APPENDIX C

PUBLISHED MODEL

This appendix contains a description of our published Github repository that contains a plug-and-play deployment of our pipeline. Performing end-to-end inference corresponds to step six in the overall method. During our experiments we evaluated multiple models. The model that performed best with respect to DSC was one of the models trained on the sparse dataset. Our repository contains the pre-trained weights from the best performing model.

The repository is open source and available at our Github[1]. It is intended for research purposes, and not for clinical use. It was designed to be easy to set up with a simple pip install, and to perform tumor masking, a single line in the command line interface starts the process. The model weights are automatically downloaded the first time the program is run. It is only downloaded once, thereafter stored locally.

Python, MONAI, and Hofmanninger et al's lungmask [54] must be installed in advance. By using pip, the repository can be installed as a python package.

```
# Install using pip
pip install git+https://github.com/VemundFredriksen/LungTumorMask
```

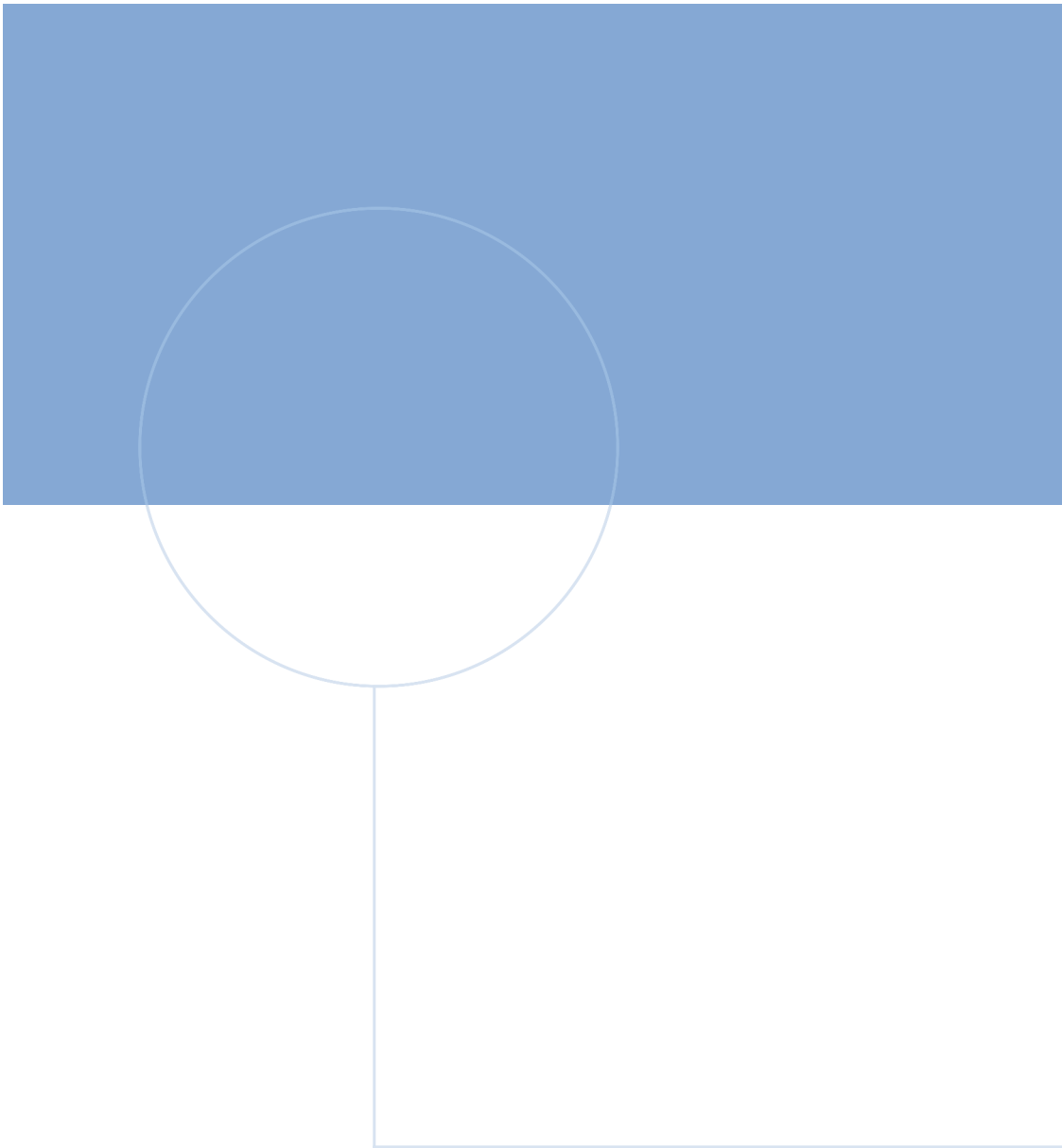
Once the package is installed, it can be used to detect and segment lung tumors from CT images. Below is an example of how to use the command line interface to initiate segmentation.

```
# Format
lungtumormask input output


# Example
lungtumormask patient_01.nii.gz mask_01.nii.gz
```

The output is stored where specified. The mask has the same resolution as the input image, orientation, and voxel spacing is inherited from the input image. Using an appropriate medical imaging software, the mask can be overlayed the CT image to be inspected.

[1] https://github.com/VemundFredriksen/LungTumorMask



NTNU
Norwegian University of
Science and Technology

SINTEF