# Automated segmentation of CT lung lesions using Deep Learning

**Master Thesis in Medical Technology**

by

**Heris Sivanesarajah**


Department of Physics and Technology
University of Bergen

June 3, 2024

# Scientific environment

This thesis was carried out at the Department of Physics and Technology at the University of Bergen, in collaboration with Mohn Medical Imaging and Visualization Centre (MMIV) Department of Radiology, Haukeland University Hospital.

The data used in the current work are medical images generously shared as an online data base. For methods development and evaluation, computational resources at MMIV were utilized throughout the project. In addition a final model was uploaded and pipelines implemented directly into a research picture archive system (PACS) at the Department of Radiology, Haukeland University Hospital facilitating also analysis of locally acquired lung CT images.



HELSE BERGEN
Haukeland universitetssjukehus

# Acknowledgements

I would like to express my gratitude towards my advisors, Frank Riemer and Renate Grüner, who provided great support, guidance, and motivation throughout this thesis. I wouldn't have been able to complete my thesis without you.

I would also like to express my gratitude to the MMIV community, especially Hauke Bartsch, for assisting me throughout the thesis, and also for the assistance in implementing the deep learning model to the clinical workflow together with Zhanbolat Satybladinov. I would also like to thank MMIV for the opportunity to participate at MMIV's annual MMIV Conference by granting me the opportunity to present a poster (Figure I.1).

I would like to thank the Ultralytics team and especially the developers of YOLOv8 (You Only Look Once version 8), which was a key component in this thesis. Additionally, I would like to thank the Cancer Imaging Archive for providing me with the LIDC-IDRI dataset, which was also a key component in this thesis.

I would like to thank my friends from Medisinsk Teknologi, my childhood friends, and other great friends from the beautiful city of Bergen. Thank you for motivating, supporting, and helping me throughout my studies; also, thanks for all the fun we have had together. I would also like to thank everyone from room 534 during the course of this thesis. I'm grateful for the company, the breaks, and the late nights we spent together.

Last but not least, I would like to express my gratitude towards my family: my mother, Suviththa Sivanesarajah, and my father, Sivanesarajah Ramanathan, for always encouraging me to strive and making me thrive, and my brother, Ketis Sivanesarajah for always being there for me. I wouldn't be here without any of you.

Heris Sivanesarajah
Bergen, June 2023

# Abstract

**Purpose:** Lung cancer, both primary and secondary, has one of the highest cancer-related mortality rates. This form of cancer can be detected using the high-resolution medical imaging technique, Computed Tomography (CT). Sometimes, even for health professionals, these lung lesions might be difficult to spot, thus providing segmentations, which may lead to late lesion detection and death. This is where deep learning (DL) can provide great assistance and prevent unnecessary deaths. In this thesis, the purpose was, therefore, to train and evaluate the DL algorithm You Only Look Once (YOLO) for CT lung lesion segmentation (and detection) and establish a local clinical workflow utilizing the algorithm.

**Methods:** This thesis was conducted by adapting the large online LIDC-IDRI dataset containing CT images of lung cancer patients to fit the utilized DL approach. After the images were pre-processed, models with various complexities were trained on datasets both with and without lesion size restrictions. Finally, a trained model was uploaded to the regional clinical research PACS, and the model was applied to local test data.

**Results:** Some of the trained models managed to detect and segment lung lesions to a high degree. The models trained with size restrictions tended to perform better. Additionally, the model uploaded to the regional research PACS was tested on unlabeled CT lung test images and was able to segment (and detect) what seemed to be tumors.

**Conclusion:** The trained model uploaded demonstrated the ability to detect and segment lung lesions, both on the online data and the test data, from the regional research PACS system, suggesting further exploring and optimizing the model developing methodology for future work. Additionally, after optimizing the model one could try to train the model to predict tumor development longitudinally.

# Contents

# Abbreviations

| | |
|---|---|
| **1D** | One-Dimensional |
| **2D** | Two-Dimensional |
| **3D** | Three-Dimensional |
| **AI** | Artificial intelligence |
| **BCE** | Binary Cross Entropy |
| **CNN** | Convolutional Neural Network |
| **CT** | Computed Tomography |
| **DNN** | Deep Neural Network |
| **DICE** | Sørensen-Dice coefficient |
| **DICOM** | Digital Imaging and Communication in Medicine |
| **DL** | Deep learning |
| **DNA** | Deoxyribonucleic acid |
| **ECG** | Electrocardiogram |
| **FN** | False Negative |
| **FP** | False Positive |
| **GT** | Ground truth |
| **HU** | Hounsfield units |
| **IoU** | Intersection over Union |
| **LDCT** | Low dose CT |
| **LIDC-IDRI** | The Lung Image Database Consortium and Image Database Resource Initiative |
| **Lr** | Learning rate |
| **LUT** | Lookup table |
| **ML** | Machine Learning |
| **MSE** | Mean square error |
| **NSCLC** | Non small cell lung cancer |
| **PACS** | Picture archive and communications system |
| **SCLC** | Small cell lung cancer |
| **SGD** | Stochastic gradient descent |

| | |
|---|---|
| **SOP** | Service-Object Pair |
| **std** | Standard deviation |
| **TN** | True Negative |
| **TP** | True Positive |
| **UID** | Unique Identifier |
| **VOI** | Values of Interest |
| **YOLO** | You Only Look Once |
| **YOLOv3** | You Only Look Once version 3 |
| **YOLOv5** | You Only Look Once version 5 |
| **YOLOv8** | You Only Look Once version 8 |

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

Lung cancer is the type of cancer with one of the highest cancer related deaths worldwide, with 1 796 144 deaths in 2020 according to GLOBOCAN. Lung cancer is also one of the most prevalent with 2 206 771 estimated new cases in 2020 [1]. This includes both primary and secondary/metastatic lung cancer [2].

Computed Tomography (CT) is a high-resolution medical imaging technique, which is used to detect lung cancers and is essential in the diagnosis and treatment of lung cancer, both primary and secondary. A CT scan is much more effective and provides more detailed information than a conventional X-ray scan [3] [4]. CTs were found to make 90% of peripheral cancers visible for heavy smokers and detectable by radiologists [4].

Reading CT scans can be time consuming and challenging for health professionals, and machine learning (ML) is a powerful tool which is more time efficient. ML tries to learn computers to solve different kinds of tasks from data, and it has, in recent years, been implemented in medical use. In medicine, ML and Deep learning (DL) (a subset of ML based on multi layer neural network) are used in e.g. automated interpretation of the electrocardiogram (ECG) and automated lung nodule detection [5].

Although the application of DL is increasing, there is still no consensus on the optimal approach of how DL can be used for lesion detection and segmentation. This thesis will provide a comparison and evaluation of a not well documented DL approach for CT lung lesion segmentation, which can provide important insights in the field of CT lung lesion segmentation and DL.

## 1.2 Objectives

The objective is to evaluate the feasibility of applying DL to perform lesion detection and segmentation on CT images in 2D through:

1. Exploring the properties of a novel DL approach (You Only Look Once (YOLO)).

2. Adapt and evaluate the application of DL (YOLO) on CT lung images.

3. Implement an optimized DL approach on an online database with predefined manually labeled lesions.

4. Establish a local, clinical workflow of integrated DL for CT lung lesions.

## 1.3 Hypothesis

In this thesis, the models are trained on Two-Dimensional (2D) images with lesions extracted from the LIDC-IDRI (Lung Image Database Consortium and Image Database Resource Initiative) dataset. The different models are then trained with unique 2D lesion size restrictions. The first collection of trained models is trained with every 2D lesion in the LIDC-IDRI dataset (without size restriction). Then the other collections of trained models are trained with specific lesion size restrictions.

The hypothesis is that the models trained with size restrictions perform better than the models trained without size restrictions because these models will be trained to identify specific 2D lesions rather than lesions of all sizes.

The most suitable trained model is then uploaded to the clinical workflow and can be used to predict lesions for each 2D image (slice) in a scan from clinical patient examinations. Furthermore, if the uploaded model predicts the existence of a lesion, with a segmentation, within a slice and the prediction is confirmed by radiologists, then the prediction can be used to retrain the model. By retraining the model with a correct prediction, a new hypothesis is that the model should get better and better for each correct prediction, and over time, the model might outperform even the best radiologist. Hence, creating a better flow in the healthcare system by saving time and money and quicker diagnosis, which can save patient lives.

## 1.4 Contribution

The contribution of this thesis is an evaluation of a DL approach for detecting and segmenting CT lung lesions through the purposed pipeline, as well as comparing these findings to existing 2D CT lung lesion detection and segmentation models. The thesis provides insights about the feasibility of using an algorithm trained on a large public heterogeneous dataset and on local clinical data. Additionally, a trained model is integrated to the local clinical workflow, which will provide assistance to radiologists and other health professionals by possibly detecting and segmenting additional lung lesions or by diagnosing new cases of lung cancer.

This thesis also overcomes obstacles such as training a segmentation model on a large, heterogeneous dataset. The dataset contains segmentations with vast differences over the same region from four different radiologists and contains CT images with various qualities. The thesis also involves converting the CT images from a DICOM (Digital Imaging and Communication in Medicine) file to a file format compatible with the proposed algorithm. Additionally, the thesis provides a solution for calculating the representative segmentation metric scores for the model, when some wrong predictions occur.

# Chapter 2

# Theory

This chapter consists of the theory necessary to understand the Method, Results Discussion, and Conclusion chapters in this thesis. Firstly, the theory behind CT is explained, which is important for generating the CT images used in this thesis and for understanding the patient risks associated with a CT scan. Secondly, some background about both primary and secondary lung cancers is provided. Thirdly, the theory behind AI, ML, and DL are explained, with emphasis on DL. DL is responsible for being able to acquire computerized assistance, to segment lung lesions not only in this thesis but also in various research papers across different fields. Finally, some background about the DL algorithm used in this thesis (YOLO), its functioning, architecture, loss functions, etc. is explained.

## 2.1 Computed tomography, CT

From the patient entering the CT machine to exiting the CT machine, various processes are happening in the complex machinery, eventually providing the user with a Three-Dimensional (3D) CT image consisting of a series of slices (2D) of the patient. This section will go through the theory needed to understand how the CT machine operates, with a detailed explanation of the X-ray production in the CT machine, photon interactions with matter, which are key for reconstructing images, the components in a CT machine, data acquisition, image reconstruction, and image display. Additionally, two different modes of CT imaging are explained in more detail, with their key differences and preferred usage. To end this section, some common CT imaging artifacts are mentioned.

### 2.1.1 CT basics

CT imaging is an imaging methodology that utilizes ionizing X-rays to generate cross-sectional images. This is achieved by directing a quickly rotating narrow X-ray beam through the patient. Then, the beam absorption (attenuation) information is acquired for each angle of the rotation and collected in a sinogram. Finally, from the sinogram, image reconstruction is done typically through filtered back projection; this is explained in more detail in Section 2.1.6. The generated 2D images are typically stacked upon one another to create a 3D image [6] [3] [7].

The visual components of a CT scanner are the gantry and a moving patient bed. The major components hidden inside the gantry are the X-ray tube, the source of X-rays, collimators, and the detector system, which collect information about the attenuated X-ray beam [8]. Each component will be described in greater detail in Section 2.1.5. To acquire the 2D images over a whole region, the bed moves slowly through the gantry, where the CT system generates these images.

The ionizing X-rays are transmitted through the patient at typical CT X-ray energies. The X-rays are created in the X-ray tube from accelerating electrons from rest with a tube potential between 80 to 140 kV, translating to typical X-ray energies between 80 to 140 kilo electron-volt (keV) [8]. The corresponding X-rays stem from two matter and matter interactions: Bremsstrahlung radiation and Characteristic X-rays. These X-rays tend to interact with matter, and these interactions are divided into three categories: Compton scattering, Photoelectric effect, and Coherent scattering. If an X-ray does not interact with matter, it will just pass through. These interactions are described in greater detail in Section 2.1.2.

The CT images created depend on the intensities of X-rays hitting the detectors compared to the initial beam intensities, where the decrease in intensity results from the various photon interactions. The images are represented in various shades of grey; when no X-rays pass through, the color of the region behind will be white, and if many/all photons pass through, the region behind will be black [6].

Since CT uses ionizing X-rays, molecules get ionized, which is when an X-ray transmits enough energy to an electron, allowing it to be emitted from the molecule, e.g., by the photoelectric effect or Compton scattering. Some of these ionized molecules are called free radicals, which can cause harm to the body and pose a risk of cancer development [3]. This can happen if those free radicals react with the DNA (Deoxyri-

bonucleic acid) itself, causing a mutation that can make a cell cancerous that the body might be unable to detect and correct on its own, which then result in cancer [9] [10]. This scenario is very unlikely because the body usually detects and corrects these mutations; it is estimated that 2 out of 10,000 CT scans (0.02%) result in the development of cancer for children and young adults. The reasons why one has to undergo a CT scan usually outweigh the risks associated with undergoing one [11].

### 2.1.2 Photon interactions

There are several ways a photon (an X-ray) can interact with matter; for low-energy photons ($\leq$ 140 keV in medical CT imaging context), these usually are the photoelectric effect, Compton scattering, and coherent scattering. The interaction type heavily depends on the energy of the initial photon and the atomic number, $Z$, of the atom the photon interacts with, as illustrated in Figure 2.1 [12].

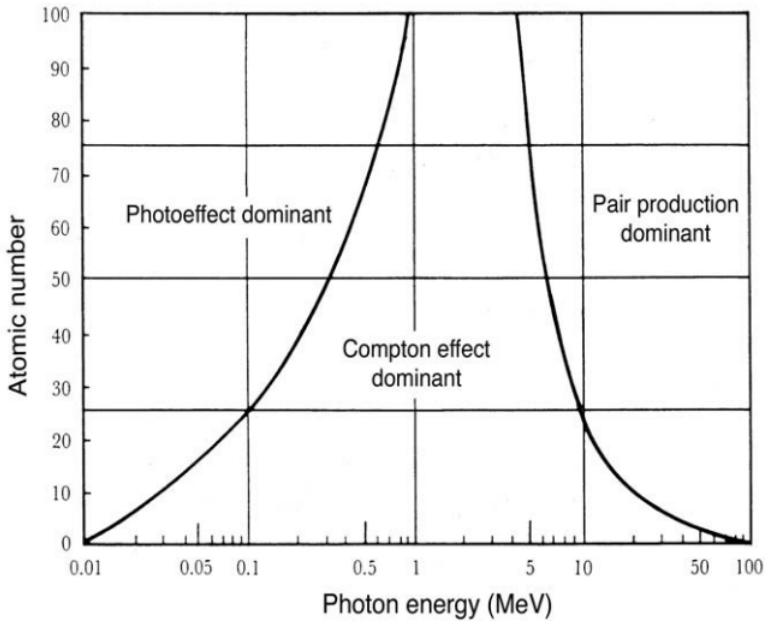


*Figure 2.1: Different regions where the different types of photon interactions dominate. The figure is acquired from [12] (FIG. 1.8).*

For CT X-ray energies, 80 keV to 140 keV, Figure 2.1 shows that the common types of photon interactions are the photoelectric effect and Compton scattering. The figure also shows that *pair production* (generation of particles with mass) is present for high-energy photons. However, these photon energies are significantly higher than the CT X-ray energies.

When dealing with energies and motions, two important conservation laws always remain true: the conservation of total energy, $E_{tot}$, and the conservation of momentum, $p$,

for every direction. These laws are expressed in Equation 2.2 and Equation 2.3, respectively. While Equation 2.1 provides the relationship between $E_{tot}$, the kinetic energy, $KE$, and potential energy, $PE$.

$$E_{tot} = KE + PE \tag{2.1}$$

$$E_{tot,i} = E_{tot,f} \tag{2.2}$$

where $i$ symbolize the initial state, and $f$ symbolize the final state.

$$p_{d,i} = p_{d,f} \tag{2.3}$$

where $d$ symbolize a spatial direction, e.g. $d = x$ in the x-direction and $d = y$ in the y-direction.

**Compton scattering**

Compton scattering is an elastic collision between a photon and a *free electron*, which is either an electron that isn't bound to an atom or a molecule or an electron that is bound to an atom or a molecule where the binding energy, $W$, is negligible. In Compton scattering, this amounts to when the binding energy to the electron is negligible compared to the incoming photon energy, $E_{p,i}$ ($E_{p,i} >> W$). Since Compton scattering is an elastic collision, the momentum and energy in the system have to be conserved. The electron will be emitted with an angle $\phi$, and a photon will be scattered with an angle $\theta$, shown in Figure 2.2, as a result of the conservation of momentum with direction, $d$, expressed in Equation 2.3. When a second photon is not scattered, then the interaction is essentially a photoelectric interaction [13], which is described in greater detail in the following section.

The energy and the momentum for the initial and final photon are expressed in Equation 2.4 and Equation 2.5.

$$E_{p,s} = \frac{hc}{\lambda_s} \tag{2.4}$$

where $c$ is the speed of an electromagnetic photon, $h$ is the Planck constant and $\lambda$ is the wavelength for a specific photon. The notion $s$ symbolizes the photon state one deals with.

$$p_s = \frac{h}{\lambda_s} \tag{2.5}$$

Assuming the binding energy is negligible, and no thermal (kinetic) energy loss, both the energy and the momentum to the emitted electron can be expressed by Equation 2.6

and Equation 2.7.

$$E^2 = (pc)^2 + (m_0c^2)^2 \tag{2.6}$$

$$p = \frac{m_0 v}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{2.7}$$

where $m_0$ is the rest mass for the emitted electron and $v$ is the speed of the emitted electron.

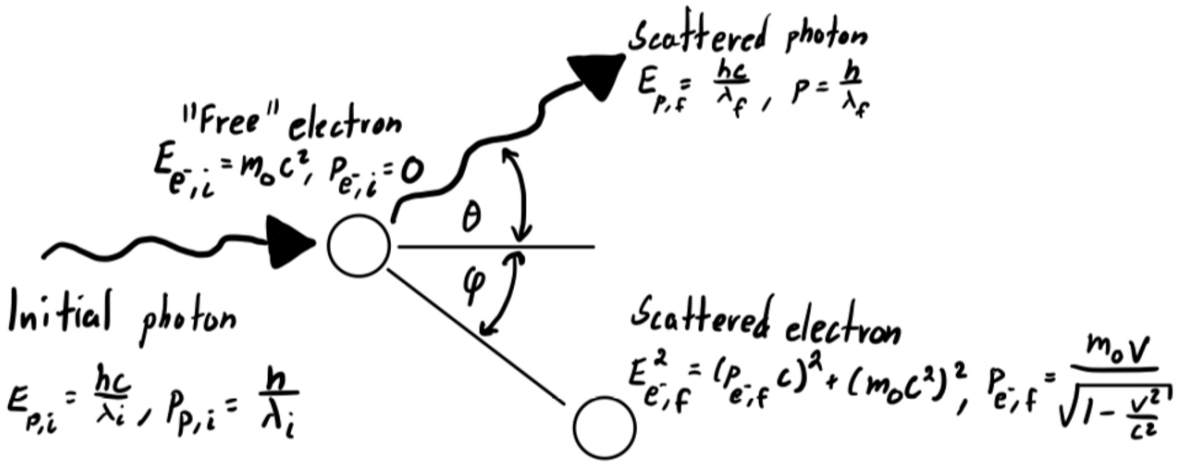


*Figure 2.2: Compton scattering: Elastic collision between a photon and a free electron. The figure was inspired from [13].*

The attenuation coefficient to Compton scattering, $\sigma$, when a photon is approaching a single electron, where the corresponding atomic number is $Z$, is proportional to $Z$ and is shown in Expression 2.8. When the photon is traveling towards a cluster of atoms (e.g., tissue, metal sheet, etc.) with the atomic number $Z$, and density $\rho_Z$, the attenuation coefficient can be modified to Expression 2.9 [12]. Since these attenuation coefficients are expressed with proportionalities, these expressions can be considered as likelihoods of occurrence.

$$\sigma \propto Z \tag{2.8}$$

$$\sigma \propto p_Z Z \tag{2.9}$$

However, the attenuation coefficient for a single electron interaction also decreases steadily from low to high photon energies [12].

### Photoelectric effect

The photoelectric effect refers to the emission of electrons from a material when exposed to photons [14]. The electrons are bound to the material with a binding energy, $W$. This binding energy can be expressed as a specific photon energy, with Equation 2.4 this can be written as Equation 2.10.

$$W = \frac{hc}{\lambda_0} \tag{2.10}$$

where $\lambda_0$ is the specific wavelength for a photon to break the bond between a given electron and its nucleus.

If the incident photon has greater energy than $W$, electrons emit from the material with kinetic energy. The maximum kinetic energy, $KE_{\max}$, to the emitted electrons is expressed in Equation 2.11 [14].

$$KE_{\max} = \frac{hc}{\lambda} - W \tag{2.11}$$

The attenuation coefficient to the photoelectric effect, $\sigma$, when interacting with a single atom with atomic number $Z$, is proportional to the expression given in Expression 2.12. From this expression, one can see that when $Z$ is fixed, the probability of the photoelectric occurring increases with lower photon energies, as also visualized in Figure 2.1.

$$\sigma \propto \frac{Z^3}{\left(\frac{hc}{\lambda}\right)^3} \tag{2.12}$$

Given the CT detectors do not detect electrons and that the photoelectric effect is more common for elements with a high atomic number, for CT X-ray ranges as shown in Figure 2.1 and Expression 2.12, this property can be exploited with contrast agents to highlight desired parts of the body. This can be achieved by using contrast agents with specific $Z$ values (higher or lower) compared to the surrounding anatomy [12]. The fact that the photoelectric effect is more common for low-energy photons also provides the opportunity to filter out such photons before entering the patient, which is explained in greater detail in Section 2.1.5.

### Coherent (Rayleigh) scattering

Rayleigh scattering occurs when the initial photon interacts with a bound orbital electron, and emits a photon with essentially the same energy as the initial photon. The

photon is emitted with an angle, $\theta$, with respect to the initial photon path [12]. The proportionality of the attenuation coefficient to Rayleigh scattering, for interacting with a single atom with atomic number $Z$, $\sigma$, is given in Expression 2.13. For a photon traveling through a cluster of atoms with the atomic number $Z$, corresponding to a density $\rho_Z$, the proportionality can be expressed as shown in Expression 2.14.

$$\sigma \propto \frac{Z^2}{\left(\frac{hc}{\lambda}\right)^2} \tag{2.13}$$

$$\sigma \propto \rho_Z \frac{Z}{\left(\frac{hc}{\lambda}\right)^2} \tag{2.14}$$

In a CT scan, the importance of Rayleigh scattering is very little to non existent compared to to the other photon interactions, as shown in Figure 2.1, only contributing to a few percent or less [12].

The Rayleigh scattering described is one simple case of Rayleigh scattering [15].

### 2.1.3 Dose

When X-rays travel through the patient, during the CT scan, the X-rays impart some (or all) of its energies to the surrounding tissue. This is known as the absorbed dose, $D$, with the SI unit gray (Gy), and is defined as the mean energy imparted, $d\varepsilon$, to matter with mass $dm$, as shown in Equation 2.15 [12].

$$D = \frac{d\varepsilon}{dm} \tag{2.15}$$

Since the absorbed dose is a measure for the energy imparted to matter. The organ dose, $D_T$, which is the mean energy distributed, $\varepsilon_T$, over an specific organ or tissue, $T$, will follow the same logic as Equation 2.15, and is expressed in Equation 2.16 [12].

$$D_T = \frac{\varepsilon_T}{m_T} \tag{2.16}$$

where $m_T$ is the mass of the organ/tissue.

In CT scans, the effective dose, $E$, is of keen interest because it provides information about the whole irradiated area and not just one specific organ. $E$ is defined as the sum of organ doses, $D_T$, with the corresponding radiation sensitivity, $w_T$. By only accounting for the dose imparted by X-rays, $E$ can be expressed by the sum in Equation

2.17 [12].

$$E = \sum_T w_T D_T \tag{2.17}$$

In a lung CT scan, with X-ray energies at max 120 keV, a typical organ dose (lung) value is 11.2 mSv and a typical effective dose is 3.8 mSv. The unit of dose given is in milli-siverts (mSv), and for X-rays 1 Gy organ dose is equivalent to 1 Sv [8] [12].

**Biological effects with CT X-rays**

The X-ray energies imparted to the matter and tissue can be imparted through the photoelectric effect and Compton scattering, which in return may result in the emission of high-energy electrons. These high-energy electrons can cause biological damage indirectly to tissue outside of the irradiated region. For X-rays in a typical CT scan, with X-ray energies typically $\leq$ 140 keV, indirect damage is the most prelevant form of biological damage to tissue and will therefore be the only one discussed in this section [12] [13].

In indirect reactions, through photon interactions, the incident X-rays may start chains of chemical reactions that produce chemically toxic free radicals, such as the hydrogen radical, $H$, and the hydroxyl radical, $OH$. These free radicals may cause changes in the cell's DNA by breaking chemical bonds. In return, the changes to the DNA can then be transferred to its daughter cells (identical copies) through mitosis or to the patient's offspring through meiosis. In mitosis, one worst-case scenario is that the change to the mother cell causes rapid uncontrolled cell growth, also known as cancer. In the case of meiosis, one worst-case scenario is that the offspring inherently have an either guaranteed or highly increased risk for developing cancer. These high energy electrons may also start a chain of chemical reactions that leads to the powerful oxidizing agent hydrogen peroxide, $H_2O_2$, but this chain is less common with CT X-rays [16] [13] [12].

The following chemical equations illustrate how the free radicals $H$ and $OH$ are created from the interaction between X-rays and water inside the body. Where Equation 2.18 illustrates the production of a high-energy electron through either the photoelectric effect or Compton scattering, and Equation 2.19 and Equation 2.20 show the final chemical reactions creating $OH$ and $H$ respectively [13].

$$H_2O + energy => H_2O^+ + e^- \tag{2.18}$$

The positive unstable water ion will slit into a positive hydrogen ion, $H^+$, and the free neutral hydroxide radical, $OH$, as shown in Equation 2.19.

$$H_2O^+ => H^+ + OH \tag{2.19}$$

For creating the hydrogen radical, $H$, the emitted electron has to fuse with another water molecule, of which there is a huge abundance inside the patient. This fusing is expressed through the following chemical equation:

$$H_2O + e^- => H_2O^-$$

The negative unstable water ion obtained will split into a neutral hydrogen radical $H$ and a negatively charged hydroxide ion, as shown in Equation 2.20.

$$H_2O^- => H + OH^- \tag{2.20}$$

### 2.1.4 CT X-ray production

The X-rays used in a clinical CT scan are generated through two primary interactions between matter and matter. Firstly, the main contributor, bremsstrahlung radiation, occurs when one decelerates a high-energy particle. Secondly, characteristic X-rays occur when one electron from an outer shell moves down to an empty "hole" in an inner shell [12].

#### Bremsstrahlung radiation

In the context of clinical CT, bremsstrahlung radiation plays a major role in X-ray production. Bremsstrahlung radiation occurs when a high-energy negative charged electron changes its trajectory due to the electrical attraction between itself and a positive charged nuclei. When the nuclei attract the electron, it changes the electron's trajectory with an angle $\theta$, causing the electron to decelerate and thus lose some of its kinetic energy. The kinetic energy lost is then emitted as an X-ray, as a result of the law of energy conservation [12].

In CT, the bremsstrahlung radiation (X-rays) produced forms a continuous energy spectrum between zero and the tube potential in keV, as illustrated in Figure 2.3. An X-ray energy at zero signifies no X-ray emission, and an X-ray energy at tube voltage indicates that the electron has come to a full stop. The X-ray energies depends on the distance between the high-energy electron and the nuclei, where when the electron col-

lides with the nuclei (full stop), the maximum X-ray energy is created [12].

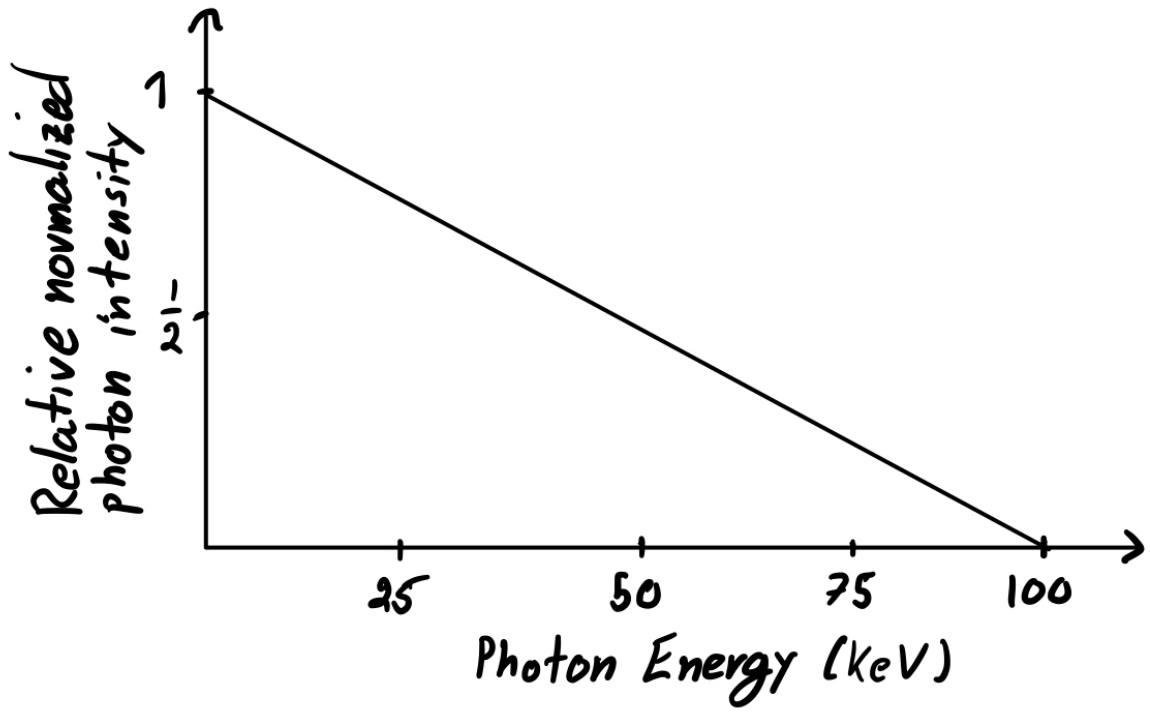


*Figure 2.3: The relative frequency of the number of X-rays produced through bremsstrahlung, with specific energies. In the figure, the electrons are accelerated with a tube potential of 100 kV, which amounts to electrons with maximum kinetic energy of 100 keV. The illustration was inspired by FIG. 5.1 in [12].*

Figure 2.3 also illustrates the linear relationship between the number of photons produced through the relative normalized photon intensity with certain X-ray energies [12].

#### Characteristic X-rays

In clinical CT, characteristic X-rays occur when a high-energy electron hits and emits an inner shell electron from an atom. Since there now is an empty "hole" in the inner shell, an electron from an outer shell with more energy has to fill the "hole" and dispose of the excess energy, which is disposed of through an emitted X-ray. Since the shell electrons have discrete energy levels, dependent on the shell localization and the atomic number, the emitted X-rays also have discrete energies, in contrast to bremsstrahlung X-rays which have a continuous spectrum of energies [12].

### 2.1.5 CT components

When looking at a CT scanner, two major components are visible: the gantry and the patient bed. However, the gantry consists of several major components, namely the

X-ray tube, the filters and collimators, and the detector system, which are the key components for generating the CT images in a CT scan. In this section, these components are described in greater detail.

### X-ray tube

The X-ray tube is the component producing the X-rays traveling through the patient during a CT scan. The X-ray tube consists of its housing, a cathode, an anode disk, a rotor, and a stator, as illustrated in Figure 2.4. The housing cools the tube components and provides electrical shielding. The rotor and the stator are components that cause the anode to rotate, and the anode and cathode are the main components for creating the X-rays [17].

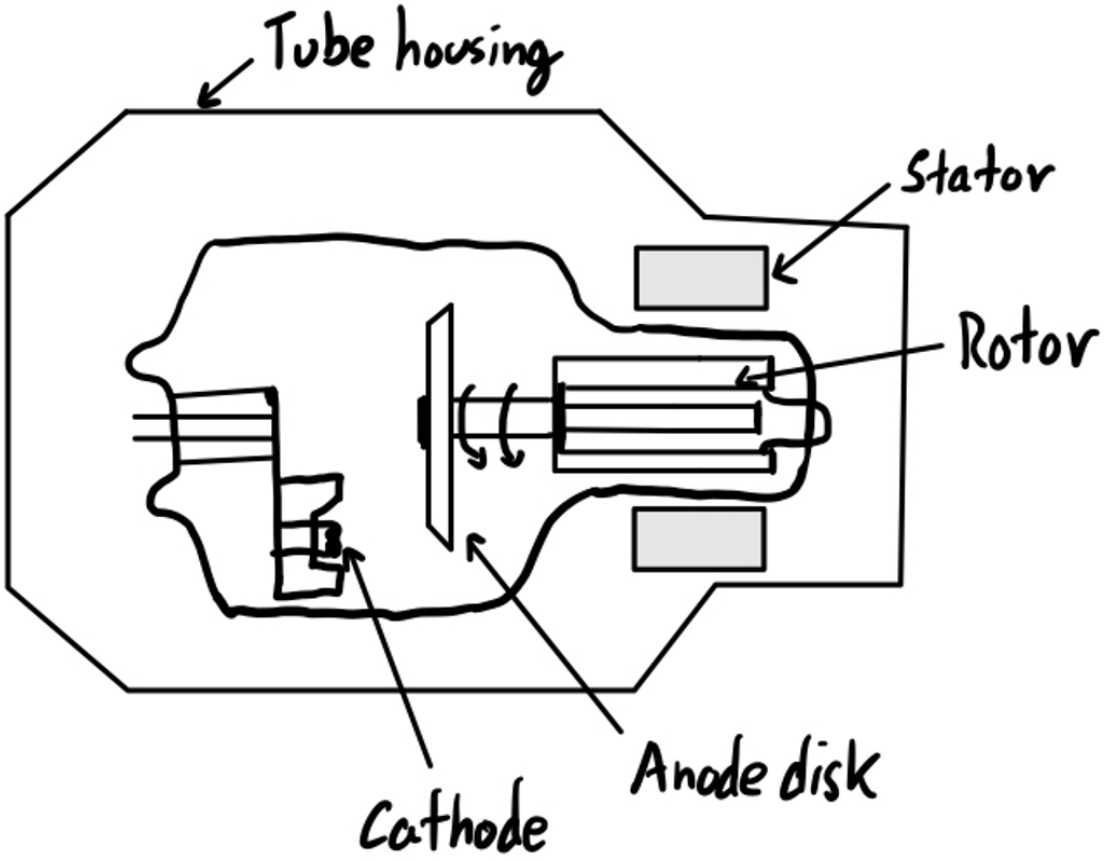


*Figure 2.4: X-ray tube with its components*

The X-rays are generated by a cathode and an anode connected to an electrical circuit, creating a potential gap. Additionally, the cathode is connected to a separate low-voltage circuit. When the low-voltage circuit generates current through the cathode, it heats up and emits electrons, the potential gap between the cathode and anode then causes the electrons to accelerate towards the anode, causing collisions at the *focal spots*

(illustrated in Figure 2.5). After the collision between the electrons and the anode, the electrons release their kinetic energies as bremsstrahlung radiation (and characteristic X-rays), heading towards the patient located perpendicular underneath the focal spots, as visualized in Figure 2.5 [17] [8] [6]. As described in Section 2.1.4, bremsstrahlung radiation is the main contributor of X-rays in a CT scan.

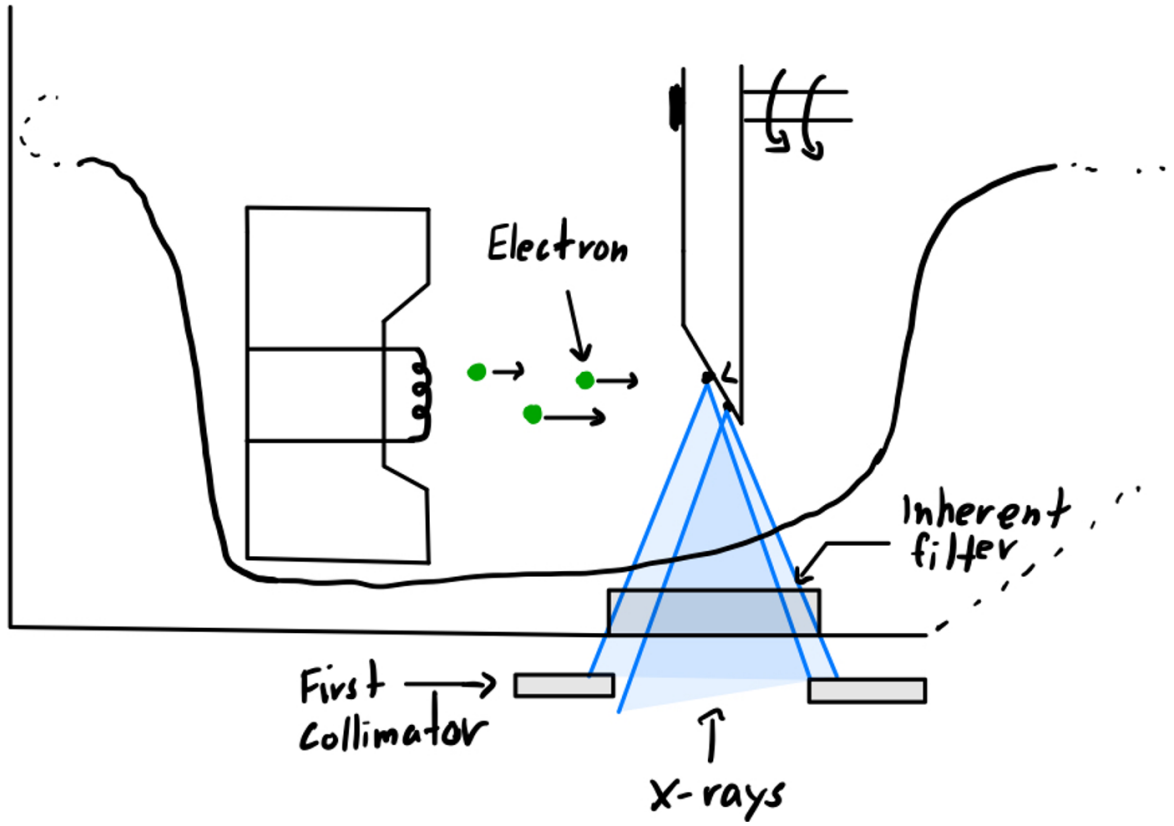


*Figure 2.5: X-ray production with electrons accelerating from the cathode to the anode.*

### Collimator

Before the X-rays reach the patient, there are several collimators that shape the X-ray beam by filtering out X-rays. An example of an illustration of a collimator with an overhead view is provided in Figure 2.6. These collimators serve three main purposes: reducing the patient dose, reducing noise in the generated images, and determining slice thickness of the images [8].

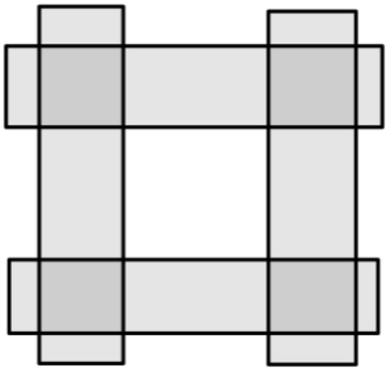

*Figure 2.6: An illustration of how a collimator might look from an overhead view.*

The first collimator is close to the anode and fixed, and it assists in shaping the initial "fan" beam to the patient, as illustrated in Figure 2.7-a). Figure 2.7-b) illustrates the case, when a collimator is not used before the X-rays enter the patient, here unnecessary areas not close to the organ of interest are radiated and receive a dose, in addition given the higher flux of X-rays the probability of Compton scattering increases which can cause increased noise to the acquired images. There can be other adjustable collimators before the X-rays enter the patient to adjust for patient size [8].

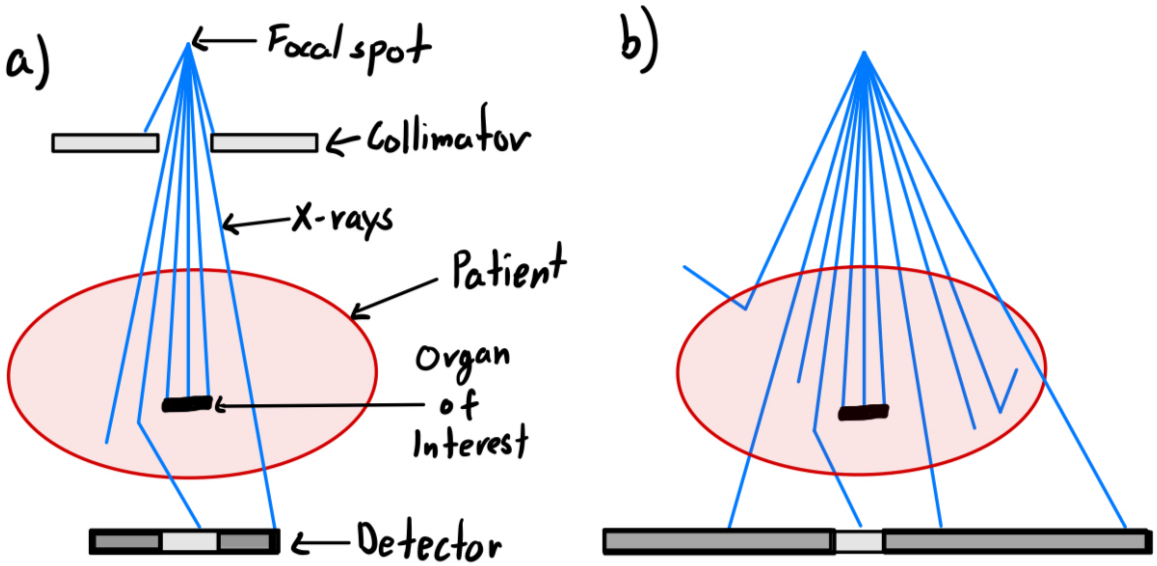


*Figure 2.7: Figure a) illustrates how the radiation to the patient is limited with a collimator. The collimator in the figure is the first fixed collimator close to the focal point. Figure b) illustrates how the patient is unnecessarily radiated without the use of a collimator.*

There are also collimators in front of the detector, these collimators serve two purposes. First, to reduce noise in the images by removing scattered X-rays. Secondly, by determining the image slice width. These slice widths are controlled by how small

an opening the collimators in front of the detector have, where a smaller opening will create smaller slice widths [8].

### Filters

In addition to collimators, filterers are used to determine what X-rays enter the patient, namely the energies of these X-rays. The filter operates by removing low-energy X-rays, through exploiting e.g. the photoelectric attenuation. The removal of these low-energy X-rays is important because these X-rays do not go through the patient and thus do not contribute to the generation of the image, and these X-rays only contribute to unnecessary radiation to the patient [8] [12].

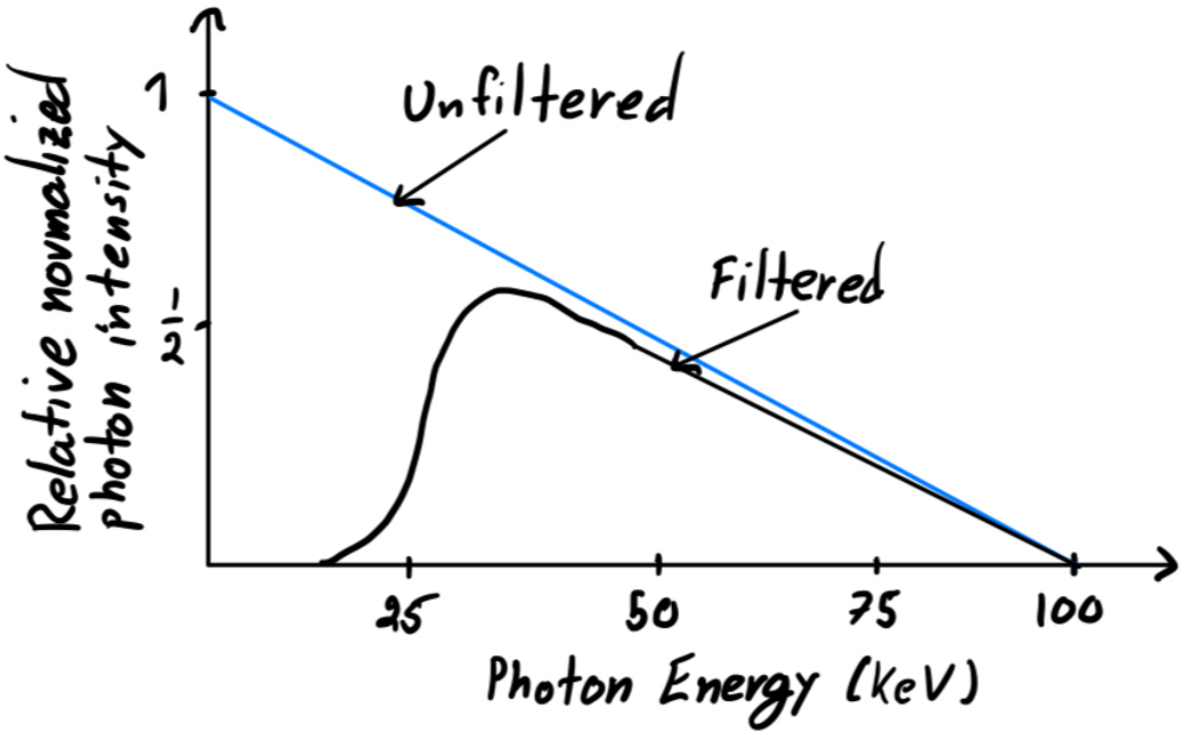


*Figure 2.8: The bremsstrahlung X-ray energy distribution without a filter (black), which is equal to Figure 2.3. The bremsstrahlung X-ray energy distribution with a filter (black), removing low-energy photons. The illustration was inspired by FIG. 5.1 in [12].*

Some filters may be monoatomic sheets with an atomic number, $Z$; these thin sheets will have some thickness, which will introduce an atom density, $\rho_Z$, for the X-ray path. Given the sheet thickness, the photoelectric attenuation coefficient (Expression 2.12), can be modified to the expression shown in Expression 2.21 [12].

$$\sigma \propto \rho_Z \frac{Z^3}{\left(\frac{hc}{\lambda}\right)^3} \tag{2.21}$$

From Expression 2.21 it is clear that thicker filters increase the likelihood of X-rays

undergoing the photoelectric effect. As a result, filters may have different shapes with different thicknesses distributed across their structure. One such filter is known as the bow-tie filter, which is thin at the center and gradually widens towards its edge. This kind of filter allows the distribution of X-rays to pass through the patient uniformly during imaging. Additionally, the filter also reduces unnecessary radiation to the patient [8].

In a CT machine the first filter is the inherent filter in the X-ray tube, which is a fixed unchangeable filter. Additional adjustable filters are also present, and the usage of these filters is dependent on the imaging purpose, the imaging location, and the patient's anatomy [8].

#### Detector system

Today, the preferred detector system utilizes scintillation crystals and photodiodes, called scintillation detectors, where one detector contains one crystal and one photodiode component [6].

When a crystal is struck by high-energy X-rays that passes through the patient without being absorbed, the crystal will absorb the incident X-rays and emit photons with lower energies in the visible light spectrum through fluorescence. The absorption coefficient for these crystals increases with the atomic number and density. Thus, the crystals are usually made from materials such as cadmium tungstate, $CdWO_4$, bismuth germinate, $Bi_4Ge_3O_{12}$, and cesium iodine, $CsI$. The emitted visible light photons then hit the photodiodes attached to the crystal, that transforms the visible light energies into electrical signals. Then the electrical signals are transformed into digital signals, which are then transferred to the computer to perform image reconstruction [6] [8].

### 2.1.6 CT imaging

The images created are dependent on the interaction of X-rays that pass through the patient, but in order to create the whole CT image, there are a few more steps needed. The steps can be divided into three segments: data acquisition, image reconstruction, and image display [6].

#### Data acquisition

Data acquisition is the process of collecting and storing the attenuation (absorption) data. In CT, this amounts to the process of collecting and storing raw data produced by

the gantry and its components described in Section 2.1.5 [6]. The main part of CT data acquisition is described in Section 2.1.5, but the attenuation of X-rays inside the patient was not described. Therefore, this section focuses on the patient-dependent part of the data acquisition.

After the X-ray beam is focused and low-energy X-rays are filtered out, with the use of collimators and filters (Section 2.1.5), the remaining X-rays enter the patient. The entering X-ray beam contains X-rays of various energies, a polychromatic beam, even after the filtering process as illustrated in Figure 2.8. The beam also has an intensity, $I_0$, which decreases throughout the patient to an intensity, $I$. Hence, some of the X-rays have either perished or have changed direction and is described as attenuation, $P$, and is expressed in Equation 2.22 [8].

$$P = ln\left(\frac{I_0}{I}\right) = \mu \cdot d \tag{2.22}$$

Inside the patient, the X-rays remaining mainly interact with tissue through the photoelectric effect and Compton scattering, as illustrated in Figure 2.1. When interacting through the photoelectric effect, the X-rays "disappear". However, when an X-ray interacts through Compton scattering, the incident X-ray is converted to an X-ray with less energy, which might still hit a detector despite the presence of collimators in front of the detector. Thus, the X-rays hitting the detectors might also be scattered photons, although ideally, the detectors want to record X-rays that pass through the patient, excluding the scattered X-rays [12] [8].

As a result, $P$ is dependent on some linear attenuation coefficient, $\mu$, that provides information about the attenuation between an X-ray and a single atom/molecule. This $\mu$ is then dependent on both the single atom/molecule photoelectric effect attenuation coefficient, $\sigma_{PE}$, and the single atom/molecule Compton scattering attenuation coefficient, $\sigma_C$, which provides the needed tissue information. Like the other photon interactions, shown in Expression 2.21 and Expression 2.9, $P$ increases with tissue thickness, $d$, as shown in Equation 2.22 [8].

The correlation between the beam intensities $I_0$ and $I$, X-ray energy, $E$, tissue/patient thickness $d$, and attenuation $\mu$ is expressed in Equation 2.23.

$$I = \int_0^{E_{max}} I_0(E) \cdot e^{-\int_0^d \mu(E)ds} dE \tag{2.23}$$

where $\mu(E)$ is the energy dependant attenuation.

Once the X-rays have struck the detector and intensity, $I$, is measured, the attenuation coefficients $\mu(x_i, y_i)$ at coordinate $(x_i, y_i)$ are computed, which provides tissue information [8].

**Image reconstruction**

Image reconstruction is the process of using raw data to create an image. There are several approaches ranging from algebraic reconstruction to statistical and ML approaches. The conventional approach is filtered back projection, and its key concepts are described in grater detail below [6].

The detectors detect how much of the initial X-ray beam, $I_0$, has been attenuated for each position, $P$, for a straight path through the patient. When every attenuation $P$ for every single X-ray path is plotted it is called an attenuation profile, as illustrated in Figure 2.9-b) [6] [7]. Now there is a One-Dimensional (1D) signal generated and stored. To illustrate the concept of back projection, one can first imagine one beam without rotation of the X-ray tube and detector. For reconstructing the image from the one 1D signal, the signal is sent back in the opposite direction; this is referred to as back projection and is illustrated in Figure 2.9-c). The back projection is done by evenly distributing the attenuation values across the whole line segment, as illustrated in Figure 2.9-c) [7].

Given that the X-ray tube and detector have a circular motion and the need to collect many 1D signals, a way to collect and store data is by the use of a sinogram. The sinogram is a collection of the attenuation profiles for each beam for multiple beam angles, where the beam angle is the angle of the beam with respect to a fixed reference axis. The beam angle changes linearly with respect to the angle of rotation to the X-ray system. Then, to reconstruct the 2D image, back projection is applied to the sinogram data. Given that the back projection distributes the attenuation values evenly across the line segment, these projections will construct artifacts called streak artifacts, which will be visible in the 2D image and the 3D image if left untreated [6] [7].

A visualization of a streak artifact can be imagined by, repeating the whole process illustrated in Figure 2.9 for multiple angles. In the end, the circular object would ap-

pear in the middle, however, not without some shade of gray in the region where the original image was black.

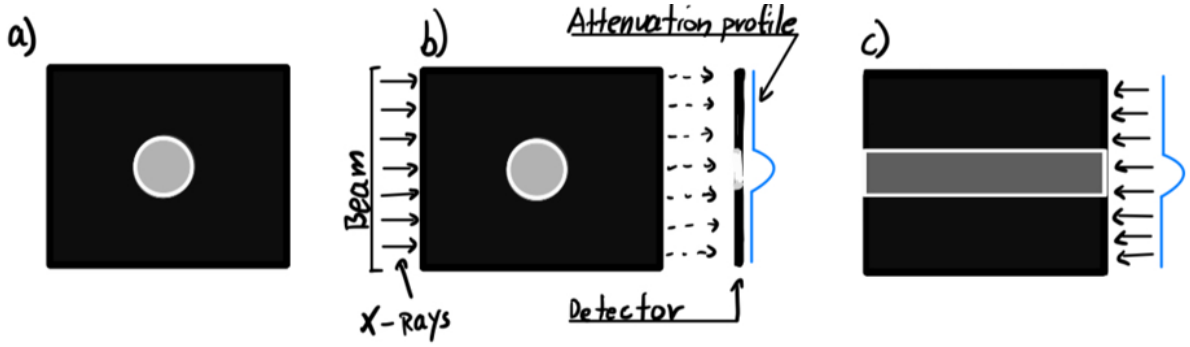


*Figure 2.9: Figure a) represents the object that is scanned. Figure b) shows how the beam from the X-ray tube passes through the object and creates an attenuation profile. Figure c) illustrates how a simple back projection works, without filtering. The illustrations are inspired by Figure 5.32 in [7]*

To reduce these streak artifacts, a process known as filtering is applied before the back projections. This is done by first transforming the data from the spatial domain to the sinogram to the frequency domain, with a Fourier transformation. Then, applying a frequency filter, e.g., the common high-pass filter, which works by amplifying high frequencies of the sinogram, in a way that edges and fine detail are more showcased. After filtering the data an inverse Fourier transformation is done to go from the frequency domain back to the spatial domain. Now, the back projection is used on the filtered sinogram, which will result in a clearer image with reduced streak artifacts [8] [7].

### Image display

To visualize CT images, the acquired linear attenuation coefficients, $\mu_T$, are displayed as CT values with the unit "Hounsfield unit (HU)" [6]. The conversion from $\mu_T$ to the corresponding CT values is expressed through Equation 2.24. Table 2.1 displays some ranges of CT values for various tissues and organs relative to the linear attenuation coefficient to water, $\mu_{water}$ [8].

$$CT\ value = \frac{\mu_T - \mu_{water}}{\mu_{water}} \cdot 1000\ HU \tag{2.24}$$

| Tissue | CT value interval (HU) |
|---|---|
| Water | [-4,4] |
| Air | [-1005,-995] |
| Lungs | [-950,-550] |
| Fat | [-100,-80] |
| Spongious bone | [50,≈300] |
| Compact bone | [≈300,>1000] |
| Kidney | [20,40] |
| Pancreas | [30,50] |
| Blood | [50,60] |
| Liver | [50,70] |

*Table 2.1: CT value ranges for various tissues and organs relative to the linear attenuation coefficient to water. The table was inspired by Figure 1.9 in [8]*

For the CT value ranges shown in Table 2.1, the overall range of CT values span from -1005 to >1000 HU (air to compact bone); in fact, medical scanners tend to provide CT values between -1024 to 3071 HU [8].

Ideally, one wants one shade of gray corresponding to one CT value. However, humans can only differentiate between 40 to 80 shades of gray at maximum. This problem is dealt with by using the "human grayscale" to only the CT value range of interest, called windowing. For example, by assigning the, e.g., 10 different shades of distinguishable grays to the CT value range [-950 HU, -551 HU], from Table 2.1 (using -511 instead of -550 for convenience), for lung examinations. When applying windowing, the whole range of CT values will then be split into e.g. 10 separate ranges, each representing one shade of gray. By using the same example as previously mentioned, the whole CT value range chosen, with HU units, will be split like this $\{\ [-950,-911],\ [-910,-871],...,[-590,-551]\}$. The shade of gray chosen to represent each range is the center value for each of those 10 ranges; in the provided example, this translates to $\{-931,-891,...,-571\}$. In windowing, CT values acquired with higher CT values than the highest CT value in the chosen range are displayed as white and black if the CT values acquired are lower than the lowest value in the chosen range. In the example provided, this translates to CT values with values > -551 HU and < -950 HU, which are displayed as white and black respectively [6] [8].

The 2D images are stacked upon one another to create the final 3D image representing the whole scan. For a scan, a vast range of CT values is stored, which allows the user to manually adjust the field of view to various ranges during the examination of the scan

[3] [6] [7].

### 2.1.7 Image modes

There exist several modes in CT imaging, and picking the right mode is influenced by what is available, where the imaging takes place, and what is the most optimal for patients. Two common modes are Helical (Spiral) CT and Low Dose CT (LDCT), where helical CT is the most common mode today, and LDCT is a safer alternative and recommended for the high-risk population [18]. These modes are described in greater detail in the following subsections.

#### Low Dose CT

LDCT gives a lower radiation dose, providing a safer alternative to the standardized one [19]. Due to the lower dose, the images do not have the same high quality as the standardized one, but this change is insignificant for the perception of lung nodules [20]. The probability of getting false positives is increased with LDCT and is therefore recommended for the high-risk population because it is also shown to reduce the mortality rate among these patients [11] [18] [21].

#### Helical CT

Helical CT, also known as spiral CT, is the most commonly used form of CT today. Helical CT scanners work so that the X-ray source and the detectors move continuously in a spiral motion along with the table. The advantages of helical scanning are increased scan speed, which in return results in improved image quality, reduction of artifacts, especially motion artifacts, and better 3D reconstruction [6].

### 2.1.8 Artifacts

Artifacts are a phenomenon when unwanted structures appear in the acquired CT images, disrupting the image qualities [8] [6]. Artifacts can appear for various reasons, e.g. patient motion, metallic implants, beam hardening (as a result of polychromatic radiation Section 2.1.6), partial volume effects, streak artifacts (Section 2.1.6), sampling errors and if the patient exceeds the limit of measurement [8]. Figure 2.10 visualizes how the mentioned artifacts negatively impact the image qualities.

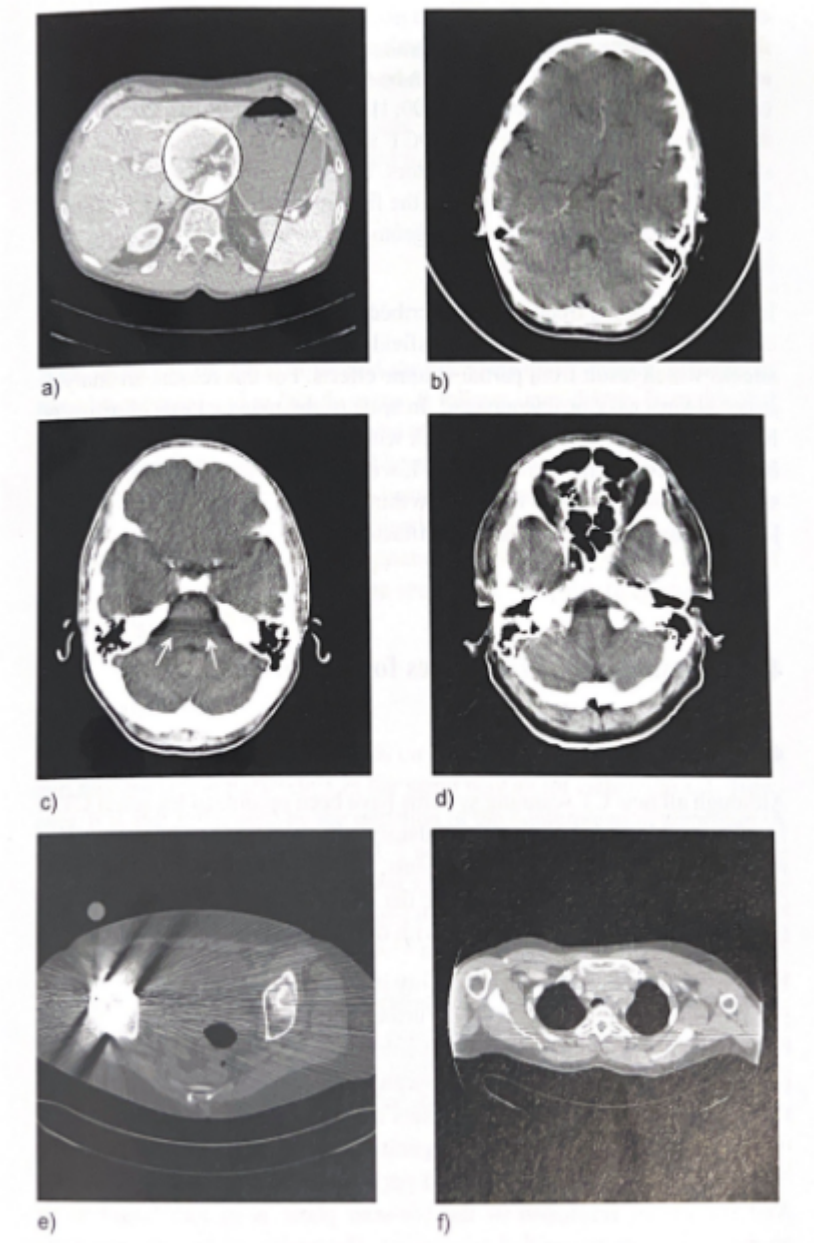


*Figure 2.10: Illustration of the common CT artifacts: sampling errors (a), patient motion (b), beam hardening (c), partial volume effects (d), metallic implants (e) or patient exceeding the field of measurement (f), which all occur in CT imaging. The images are acquired from [8] (Figure 4.10).*

However, even if the artifacts are unwanted, some patients might have permanent metallic implants, thus making such artifacts inevitable. On the other hand, some of the artifacts mentioned are intervenable such as reducing patient movement and reducing sampling errors.

## 2.2 Primary lung cancer

Primary lung cancer are cancerous lung lesions (tumors) that originates in the lungs, whereas cancer spreading to the lungs from another organ is referred to as secondary

lung cancer or lung metastasis [22] [23] [24].

Primary and secondary lung cancer have many shared symptoms. The symptoms shown below are common in both cases [25]:

- Long lasting coughs.
- Frequent chest infections.
- Coughing up blood.
- Persistent Breathlessness.
- Persistent tiredness and lack of energy.
- Loss of appetite and/or unexplained weight loss.

Primary lung cancers are divided into two groups: non-small cell lung cancer and small cell lung cancer. Whereas, non-small cell lung cancer again is divided into three main types: squamous cell carcinoma, adenocarcinoma and large cell carcinoma [22] [23].

The small cell lung cancer (SCLC) is the less common one with between 15 to 20% of all the primary lung cancers where non small cell lung cancer (NSCLS) account for the rest [22] [23]. SCLC is highly metastatic, close to 70% have a metastatic disease when diagnosed with SCLC [26]

Adenocardionma starts in the lining of the airways. Squamous cell carcinoma tends to grow in the center of the lung. Large cell carcinoma are large in appearance and can originate in any part of the lungs [23].

## 2.3 Secondary lung cancer

Secondary lung cancer (lung metastasis), are when cancerous lesions are found inside the lungs, but the lesions originate from another part of the body [24]. Metastasis to the lung is the second most common form for metastasis [27]. The identification of metastasis is of great importance, because the patient will have a primary tumor site which need to be identified if not already, to hinder metastasis other organs/structures.

Similar to primary lung cancer, secondary lung cancer also exhibits the symptoms listed in Section 2.2.

The lesions spread when tumor cells circulate from the primary lesion through the lymphatic system or through blood vessels [24]. The most common route of spread is through blood vessels, and due to this type of spread the metastatic lesions have a tendency to manifest itself at the rim of the peripheral part of the lungs and in the basal part [27].

## 2.4 Artificial Intelligence

Artificial Intelligence (AI) is when a machine or a computer is able to simulate human intelligence by making decisions from either a set of rules, previous information or experience [28]. The purpose of utilizing AI is to create AI applications which will be good at specific tasks, such as: text generation (ChatGPT [29]), CT lung image segmentation or making decisions with decision trees, etc..

There are different ways to create an AI applications. One alternative is to define a set of rules, such that the application can make predictions without using training data. Another alternative is to feed a model a dataset with information, such that a model can learn to find patterns from previous data, ML. Figure 2.11 showcases how AI and ML are related, and also introduce DL as a subset of ML, which is explained in greater detail in section 2.6.

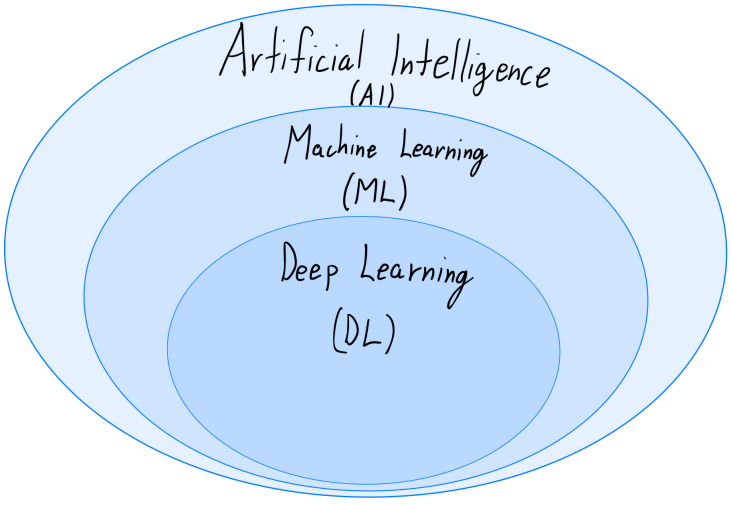


*Figure 2.11: Illustration of how AI, ML and DL is related to each other. The figure is inspired from .*

## 2.5 Machine Learning

ML is an AI-generating technique that focuses on creating computer algorithms that build models which can learn from data over time [30]. By feeding the model with

more data, the model can better predict new unseen data. The ability to make good predictions on unseen data is called generalization and is the objective of an ML model [31], and the generalization is task-dependent. An ML model must be tuned to specific applications, e.g. object detection and/or image segmentation of CT lung lesions.

ML problems are often divided into three groups: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning models train on data with labels, and the objective is to see patterns in input data and classify them with a label. The supervised model tries to generalize on labeled data and can make predictions on unseen data. Unsupervised learning models train on unlabeled data and cluster the outputs based on patterns, relationships, etc. [31]. Reinforcement learning models use the concepts of reward and punishment to train. The model gets rewarded when doing something correctly and a punishment when doing something false. By doing this, the model is forced to learn by trial and error [32]. An example of an AI model based on reinforcement learning is a model that optimizes travel paths in a maze. In the case of lesion detection and segmentation in lungs, supervised learning is the preferred path, every lung of every human has a slight deviation from one another, and by giving the training model some labeled data, it can learn to detect abnormal lesions.

### 2.5.1 Object detection

One ML task is object detection, and is an important tool in identifying the existence of specific objects in an specific region within an image. An object detection model is a model that is able to identify the existence of an object in an image. This is usually done by creating a bounding box around the area of interest [33]. Object detection is widely used in medical imaging e.g. in lung lesion detection [34] [35]. In this project one of the main goals is to segment lesions in the lungs for CT scans, and lesion detection is a crucial step for performing segmentation. In this thesis the object detection is done by YOLO as well as the segmentation.

### 2.5.2 Image segmentation

Image segmentation is a computer vision task, that classify each pixel in an image with a label, which provides information on what the given image contains. Segmentation is important when one want to know the exact region an object is located, without the additional region object detection provides [36]. Figure 2.12 provides an example of a segmented brain, where the cerebellum, brain stem and brain lobes are segmented and highlighted with distinct colors.

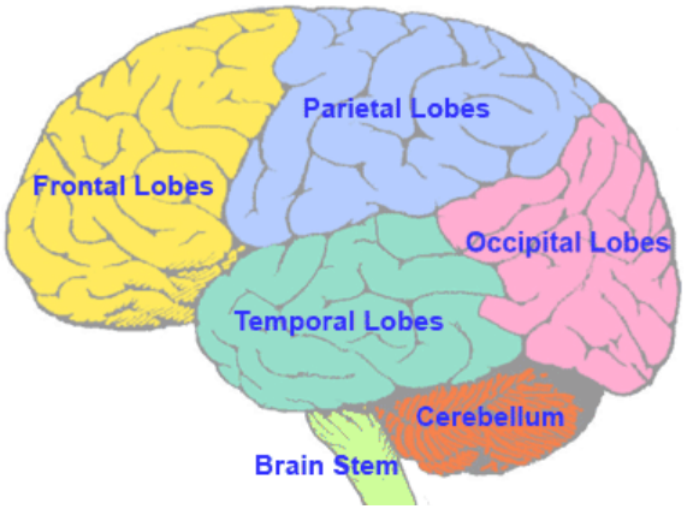


*Figure 2.12: A segmentation of the brain, where the cerebellum, brain stem and brain lobes are segmented and highlighted with distinct colors. The image are acquired from [37].*

Image segmentation is divided into semantic segmentation and instance segmentation. Semantic segmentation classifies each pixel with a label, which is useful for identifying the presence of an object in an image. However, semantic segmentation lacks the ability to distinguish between multiple objects in an image, e.g. two separate lesions, which makes it challenging to know how many objects there exist in an image. Instance segmentation builds upon semantic segmentation, by not only performing pixelwise classification, but additionally is able to distinguish between individual objects, e.g. two separate lesions, and also provides an object count [36].

The type of image segmentation method to use is task-dependent. Semantic segmentation is suitable if one want to know the presence of an object in an image. However, if the number of objects is of significance semantic segmentation will not suffice. For lung lesion segmentation tasks, the number of lesions present in a patient is of interest, and one has to use instance segmentation [36].

## 2.6 Deep learning

DL is a subgroup of ML that uses multilayered neural networks to learn from data. DL models are also referred to as deep neural networks (DNN) [31] [30]. The structure of these models is inspired by neurons in human brains and other animal brains, where one neuron in a hidden layer is analogous to a neuron in the human brain. The hidden layers provide the model with the ability to extract features without manual intervention. There are many types of DL networks. One of them is convolutional neural networks, which is the type of network used in this thesis [30] [38]. Figure 2.13 provides an

illustration of the structure of a deep neural network with its layers.

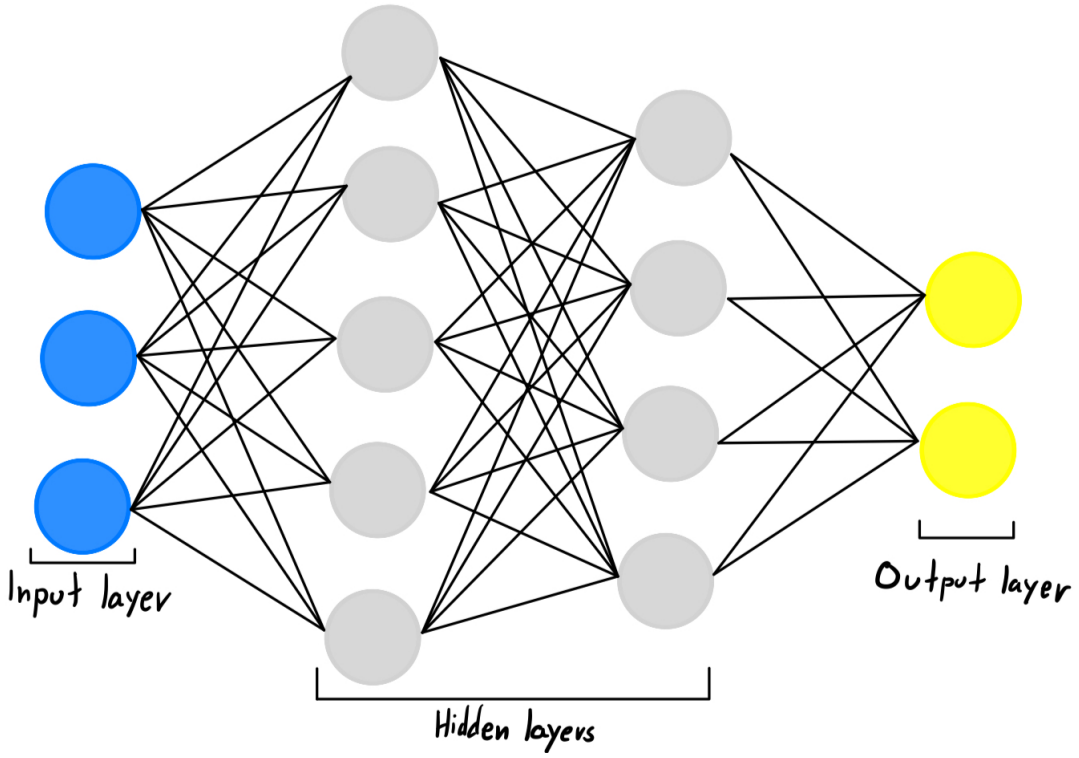


*Figure 2.13: A visual illustration of the structure to a deep neural network with its layers. The figure was inspired by [38].*

### 2.6.1 Convolutional neural network

A convolutional neural network (CNN) is a type of DNN, and is most prevelant used for image analysis. CNNs have made great success in medical analysis e.g. by object detection and segmentation, and thus wildly used in the field [39]. The huge benefit of CNN is its unique ability to identify relevant features and patterns [38] [30].

Some of the CNNs components are convolutional layers, activation functions, pooling layers, fully connected layers, and loss functions. These features are all described further below. One of the key benefits for using CNNs rather than other neural networks is CNNs weight sharing ability, which helps the generalization and reduce overfitting by reusing weights across layers [30] [38].

#### Convolutional layers

Convolution layers are the most important part of a CNN. The purpose of a convolution layer is to extract features from the data (the feature map), this is done by different kinds of kernels, also known as convolutional filters. A kernel is a matrix containing values,

in some kernels, the values are placed in a specific order to provide the model various abilities, e.g. detect edges. Other kernels may contain random values in a random order, which acts like weights to the model, the weights are then updated for each training era (epoch), such that the model gains the ability to extract unique features [30].

### Activation functions

Activation functions are a way to introduce non-linearity to the neural network. The main purpose of an activation function is to either activate or deactivate a neuron, based on its weights and biases, to allow for the network to learn complicated patterns. This is done by applying an activation function after the layers with weights [30]. The activation function for a neural network works as the action potential in animal brains, a threshold has to be reached in order to fire a neuron (in the brain) and excite a neuron (for a neural network) [31].

There are many different activation functions that are used today, but for an activation function to be good the activation function needs to be differentiable, and the derivative should be a non-zero value for the "activated" part since this allows for back-propagation [30]. Some activation functions used today are Sigmoid, ReLU and SiLU which are given in Equation 2.25 - 2.27 respectively, and also illustrated in Figure 2.14-a) - c) respectively. Additionally, Figure 2.14-d) provides a comparison between ReLU and SiLU illustrating the similarities for larger values.

$$Sigmoid: \ Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{2.25}$$

$$ReLU: \ ReLU(x) = max(0,x) \tag{2.26}$$

$$SiLU: \ SiLU(x) = x \cdot Sigmoid(x) \tag{2.27}$$

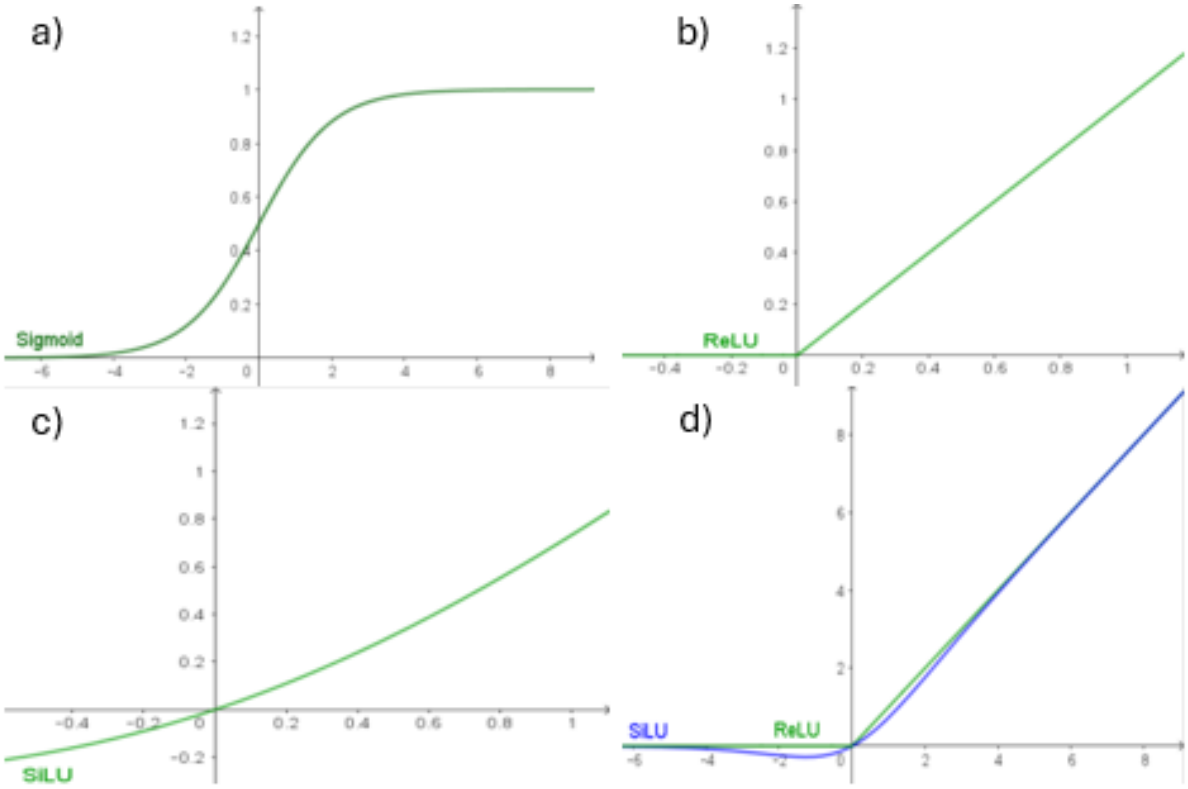


*Figure 2.14: The activation functions: Sigmoid, ReLU, and SiLU are illustrated in plots a), b), and c), respectively. Additionally, a comparison between ReLU and SiLU are provided in plot d).*

When training a DL model, there are no definite answer for which activation function to use, and in one DL model there might be more than one activation function present. The SiLU function given in Equation 2.27 is an activation function used by the DL algorithm YOLO [40], which is the algorithm utilized in this thesis. An advantage of SiLU and Sigmoid purposes is that the functions are continuous and differentiable, compared to ReLU, which is obviously discontinuous at origo. Thus making SiLU applicable for back propagation (Section 2.6.5).

#### Pooling layers

The main task of pooling layers is to shrink a feature map and create a new smaller feature map, by maintaining the majority of the dominant features [30]. Unlike convolutional layers used for feature extraction, pooling layers are used for down-sampling and reducing the feature map, by changing their dimension. The pooling operation is similar to convolution mentioned in 2.6.1, involving a kernel size [30]. However, this kernel is non-learnable, and acts like an invisible box placed over the feature map and performs pooling operations.

There are several pooling operations, the most common ones are max pooling, min

pooling, average pooling and global average pooling. Max pooling extracts the maximum value within the kernel's receptive field, while min pooling extracts the minimum value. Average pooling takes the average value within the kernel's receptive field. Global average pooling is independent of the kernel, and takes the average value of the whole feature map [30]. Figure 2.15 illustrates various pooling operations with their new feature maps.

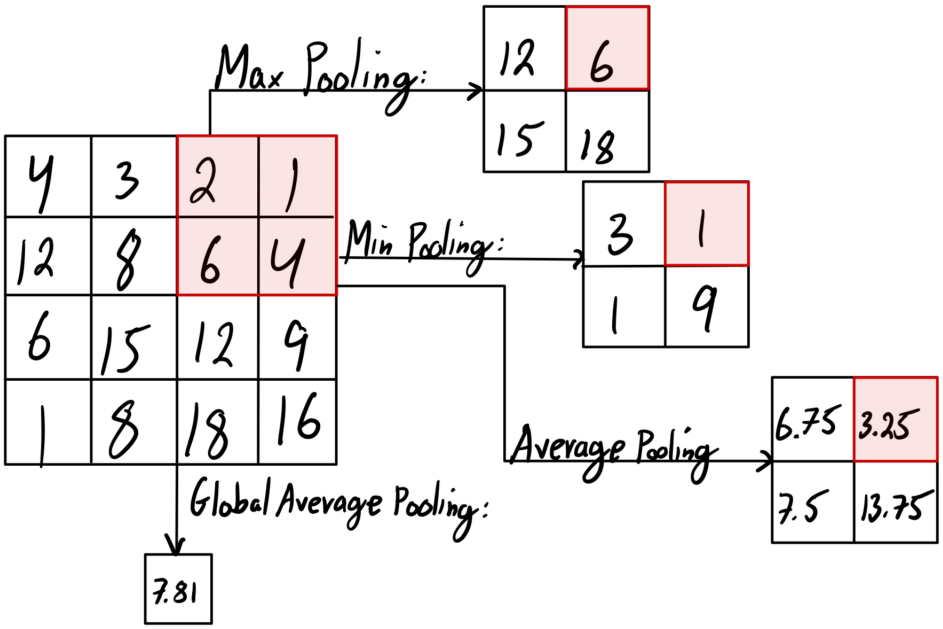


*Figure 2.15: Examples of different pooling layers. The figure was inspired from [30].*

#### Fully connected layers

Fully connected layers are usually at the end of the CNNs. These layers are unique in the sense that each neuron is connected to every neuron from the previous layer and every neuron in the next layer. This is achieved by converting the feature maps from the previous layer to a 1D vector. After applying the final fully connected layer, every neuron is connected to $n$ number of neurons, where $n$ is the number of classes/labels in the classification task [30]. In terms of lesion classification, $n$ may be equal to two, indicating lesion or no-lesion.

### 2.6.2 Loss function

The loss function is the function that provides information about the model's performance, and the loss value gets updated for each *epoch* during training. The loss function measures the difference between the output of the model and the ground truth. Hence, the goal for DL models is to minimize the value of this loss function by finding the most optimal parameters for the model. There are different kinds of methods to minimize

the loss function by parameter optimization one of which is gradient descent, which is described in greater detail in Section 2.6.6 [30].

An example of a simple loss function is the mean square error function (MSE), which is given in Equation 2.28.

$$L = \frac{1}{N}\sum_{n=1}^{N} |Y_{\text{Ground truth}} - Y_{\text{prediction}}|^2 \tag{2.28}$$

where $N$ is the number of samples in the dataset.

The lowest point in a function, defined on an unbounded interval, it is known that the gradient (derivative) at that point is zero, which the gradient descent exploits. However, there are several other cases where the gradient is equal to the zero, namely: the global and local maximas, local minimas and saddle points. These cases are the solutions one wants to avoid when minimizing the loss function.

For an arbitrary function, the global maxima is where the function has its absolute highest possible value, and vice versa for global minima. A local maxima is a point where the gradient to the function is equal to zero, but the point is not the global maxima, and vice versa for local minima. A saddle point is a point where the gradient to the function is zero, but the neighboring gradients to one side are positive, and the neighboring gradients to the other side are negative. However, for a function only defined on a specific interval (bounded), there might exist other global maximas and minimas at the end points, which are further referred to as constrained maxima and minima. Figure 2.16 illustrates every case mentioned, where the graph to the left illustrates cases when the gradient to $f(x)$ is equal to zero, with point 1. - 5. illustrating the global maximum and minimum, a local maximum and minimum, and a saddle point, respectively, and the stars marked on the graph to the right illustrate the global minimum and maximum to a function with a closed bounded interval, where the corresponding gradients are non-zero.

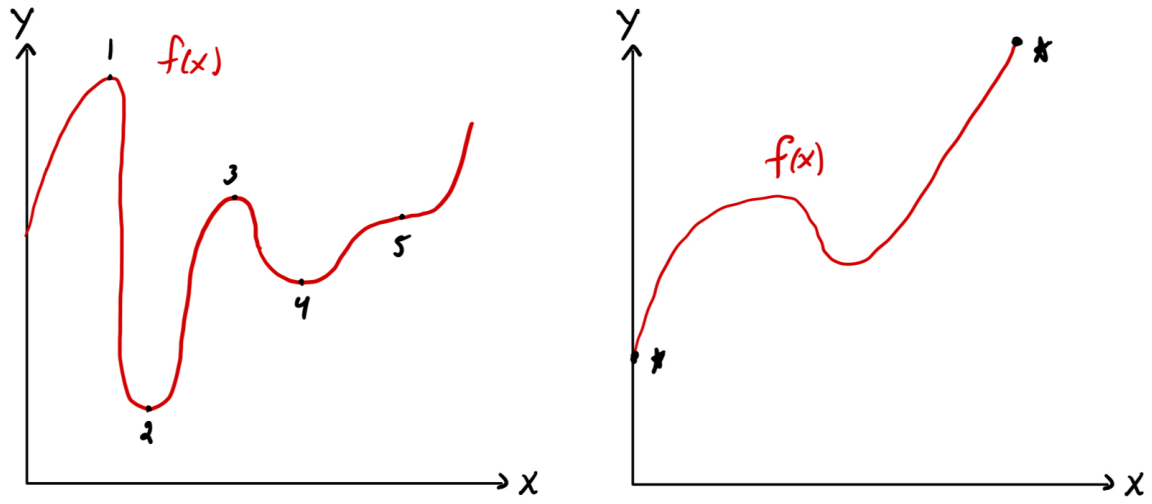


*Figure 2.16: The graph to the left illustrates examples where the gradient to $f(x)$ is equal to zero (the derivative). Point 1. - 5. illustrates the global maximum, global minimum, local maximum, local minimum and the saddle point respectively. The stars marked on the graph to the right illustrates the global minimum and maximum to a function a closed interval, where the corresponding gradient is non-zero.*

### 2.6.3 Hyperparameters

In DL, hyperparameters are parameters one has to specify before training the models, some of these hyperparameters are the learning rate, epochs, dropout, and batch size, which are described in greater detail in this section. Additionally, the hyperparameters momentum and weight decay are described in Section 2.6.6. The hyperparameters differ from the unspecified parameters, which are acquired throughout the training process [31], and will thus henceforth, only be referred to as hyperparameters.

These hyperparameters have a vast range of which values they can possess, and can as a result be difficult to pick the right hyperparameter values to obtain the absolute lowest possible validation loss, global minimum (see point 2. in Figure 2.16), this is because any change to the hyperparameter values will affect the general DL performance [30]. Methods for optimizing these hyperparameters are mentioned in Section 2.6.4.

#### Learning rate

The learning rate (lr) is with how big of a leap the model parameters get updated. If the lr is too small, the model might never reach its optimal parameters, $\theta$. In contrast, if the lr is set to high the model might diverge from the optimal solution [31]. Both cases with a small lr and a large lr are showcased in Figure 2.17.

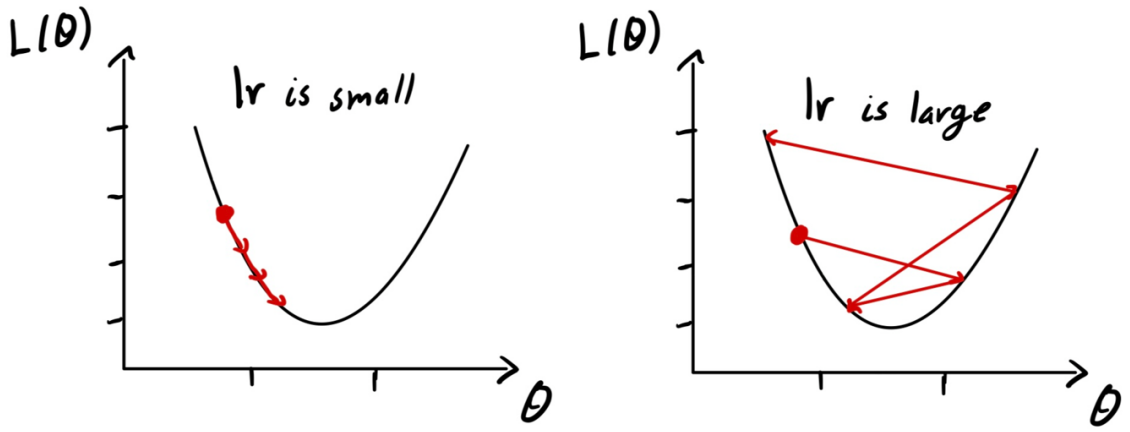


*Figure 2.17: Effect on convergence for a small and a large learning rate. The figure was inspired by [41].*

The optimal lr is usually found by trial and error, because one has to try various lrs to determine which one converges, and the convergence is also dependent on the other hyperparameters [30].

#### Epochs

The hyperparameter epochs is the number of times one wants to scan the whole training set and update the model parameters during the training process [31]. Since the optimal parameters are dependent on the combination of hyperparameters, it is important to not have too many epochs, because one might skip the optimal solution and may be computational and time expensive [30].

Its possible to apply early stopping, which will stop the training process when the validation loss increases while the training loss decreases. However, there are several issues with early stopping. Given the case of slow convergence as illustrated in Figure 2.17 the early stopping might not get applied, which when the number of epochs is very large results in a long training time. An additional problem with early stopping, is that the training process can stop in a local minima or saddle point (4. and 5. in Figure 2.16), resulting in not achieving the optimal parameters [31].

#### Dropout

Dropout is a technique used for generalization. It works by randomly inactivating neurons for a whole epoch. The portion of how many neurons to inactivate is manually chosen. By inactivating neurons, the model is forced to learn other different features by the training data presented [30].

However, the use of dropout and how big of portion of neurons to inactivate is task-dependent, and in some cases the use of dropout is not necessary for generalizing the trained model.

#### Batch size

To train the model, one could train it with every data point in the training set simultaneously for each epoch. However, this would require a huge amount of computational power/memory, which not all machines have.

To avoid this batch size is introduced, this picks out $n$ samples (the batch size) from the training set and updates the model parameters with these $n$ samples. Since an epoch is a scan of every data point in the training set, the parameters get updated $k$ times ($k$ iterations), such that the whole dataset is used before starting a new epoch [42]. To illustrate, one can use the example with $n = 10$ and the number of data points in the training set equal to 1000. Then $k$ would be $1000/10 = 100$, which means that the parameters get updated 100 times before starting the next epoch.

However, the maximum batch size value is both model complexity-dependent and computer-dependent. With a more complex model, more memory is used for processing each data point, which means that the model probably has to have a smaller batch size than for training a less complex model.

### 2.6.4 Optimization of parameters and hyperparameters

Finding the right parameters, and hyperparameters for optimizing the trained model, is a difficult task in DL and can potentially be very time expensive. There are different methods for optimizing both the hyperparameters and parameters.

The goal with optimization is to minimize the validation loss by finding the right parameters and hyperparameters. There is no guarantee that one finds the most optimal parameters or hyperparameters, one might a sub-optimal solution (local minima or saddle point), or one never even find parameters where the validation loss converges during the training time [30].

Further in this section two optimizing strategies for optimizing hyperparameters are described, in addition to a soft introduction to the optimizing strategy *gradient descent*, which optimizes the parameters.

#### Hyperparameters

There are several ways to tackle the search for hyperparameters, in this section two approaches are described in grater detail.

Grid search is a technique where one defines which hyperparameters to test on and values for each of them (the hyperparameter space). This approach can be very time expensive given the complexity of the hyperparameter space. This approach also has some limitations, given that one has to specify the values to test on for the different hyperparameters, one might get a sub-optimal set of parameters [43]. E.g., if one test on these values $[0.1, 0.2, 0.3, 0.4]$ for the hyperparameter *lr*, the grid search might suggest that the lr equal to 0.3 might be the best, but in reality the true optimal lr might be 0.25.

Random search is another algorithm that is used in DL to find parameters. Here, the hyperparameter space is also defined, but not the values of each hyperparameter, however, one can usually define a range of values that limits the hyperparameter values. Additionally one can specify how many different combinations to test. These values are generated randomly and the model compares the validation losses across every random hyperparameter combination. Unlike grid search, this approach can provide random specific values, e.g. 0.234823, that one never would think to provide in a grid search [43]. However, like grid search, random search also has the limitation that the optimal combination of hyperparameters might never be found.

#### Parameters

To find the optimal parameters, weights and biases, there exists several optimizing algorithms. Some of the algorithms are based on the concepts back propagation and gradient descent, which are both described in the subsequent sections. These algorithms try to minimize the loss function by updating the weights and biases for each iteration [30].

As always, one is never guaranteed to find the optimal set of parameters for the given task. Here, one can also encounter the possibility of reaching a sub-optimal solution or never arriving at the optimal solution.

### 2.6.5 Back propagation

Back propagation is a method to calculate the gradients to the loss function. The method propagates backwards from the end of the network and calculates each gradient by using the concept chain rule from calculus [30]. To visualize this progress a

computational graph is shown in Figure 2.18.

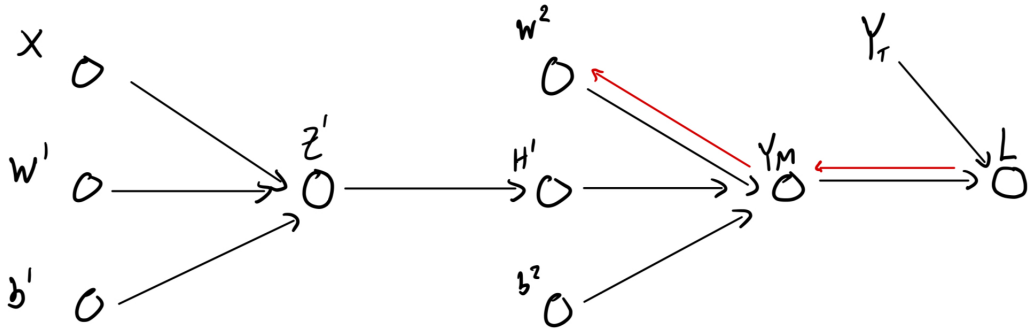


*Figure 2.18: Illustration of how a specific gradient is calculated by following the red arrows in the computational graph. Each variable have the following meaning: Input* $\mathbf{X}$*, Weights* $\mathbf{W}^i$*, Biases* $\mathbf{b}^i$*, Activation function* $\mathbf{Z}'$*, Hidden layer* $\mathbf{H}'$*, Model output* $\mathbf{Y}_M$*, Ground truth* $\mathbf{Y}_T$ *and Loss* $\mathbf{L}$*.*

In Figure 2.18 the variables provided have the following meanings: Input $\mathbf{X}$, Weights $\mathbf{W}^i$, Biases $\mathbf{b}^i$, Activation function $\mathbf{Z}'$, Hidden layer $\mathbf{H}'$, Model output $\mathbf{Y}_M$, Ground truth $\mathbf{Y}_T$ and Loss $\mathbf{L}$. The red arrows in the figure, illustrate the path taken to calculate the loss function gradient with respect to $\mathbf{W}^2$, $\frac{\partial \mathbf{L}}{\partial \mathbf{W}^2}$, by applying the chain rule, the calculation of the gradient is expressed in Equation 2.29.

$$\frac{\partial \mathbf{L}}{\partial \mathbf{W}^2} = \frac{\partial \mathbf{L}}{\partial \mathbf{Y}_M} \frac{\partial \mathbf{Y}_M}{\partial \mathbf{W}^2} \tag{2.29}$$

To acquire the other loss function gradients with respect to the corresponding variable, one can apply the same logic as illustrated in Figure 2.18 and expressed in Equation 2.29.

For calculating the loss function gradients the dimensionalities are chosen to have the same dimensionality as the corresponding variable. In Equation 2.30, this means that the dimensionality of the calculated gradient has to have the same dimensionality as $\mathbf{W}^2$. To acquire the desired dimensionality, one might have to transpose, perform element-wise multiplication, or reorder the gradients.

### 2.6.6 Gradient Descent

Gradient descent is the process of minimizing the loss function based on the training losses throughout the training process. The equation describing the gradient descent algorithm is provided in Equation 2.30 [30].

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i\text{-}1} - lr \cdot \nabla L(\boldsymbol{\theta}_{i\text{-}1}) \tag{2.30}$$

where $\boldsymbol{\theta}_i$ is the $i$-th parameter for the model (weight or bias), and $L(\boldsymbol{\theta}_i)$ is the loss function for the $i$-th parameter.

The primary goal of gradient descent is to find the most optimal parameters where the gradient of the loss function reaches zero, corresponding to the global minima of the loss function. This involves updating each parameter during every iteration [30]. To illustrate how each parameter is updated, an example where the two parameters $\theta_1$ and $\theta_2$ are updated is provided by Equation 2.31 and Equation 2.32.

$$\theta_1 = \theta_0 - lr \cdot \frac{\partial L(\theta_0)}{\partial \theta_0} \tag{2.31}$$

$$\theta_2 = \theta_1 - lr \cdot \frac{\partial L(\theta_1)}{\partial \theta_1} \tag{2.32}$$

where the gradients in the equations are acquired by back propagation.

When applying general gradient descent to find optimal parameters, the global minima is dependent on the hyperparameters chosen, such as lr and epoch, which constrain the number and which of parameters examined.

An important factor one has to consider when using gradient descent is that local minimas (local maximas and saddles points) also have gradients equal to the zero vector ($\nabla L(\boldsymbol{\theta}) = \mathbf{0}$). If the gradient of the loss function is zero, the parameters will not be updated, as shown in Equation 2.30.

Another factor to consider is the loss minimized by the algorithm presented in Equation 2.30, is the training loss, which poses the possibility of the model overfitting. To avoid this, one could implement early stopping, but this approach also can encounter some issues (Section 2.6.3). Other approaches to avoid overfitting are to implement momentum and weight decay, which are both described in the subsequent sections. These implementations also give the model an opportunity to escape local minimas.

When the gradient descent optimizer is applied without separating the training data into batches, the algorithm will calculate every new parameter, gradient, and loss, with every data point in the dataset simultaneously, which, as mentioned in Section 2.6.3, can be computationally expensive and might need more memory than the computer can provide. When applying a batch size, $n$, the computational power needed is less memory expensive, one such gradient descent based algorithm is the stochastic gradient

descent (SGD), which is described in greater detail in the following section [30].

### Momentum

Momentum is a hyperparameter utilized together with gradient descent, with the goal of streamlining the convergence rate as well as escaping local minimas, which, in return, can enhance the model's performance during training. This is done by adding a momentum, $\mathbf{m}$, with momentum weighting, $\lambda$, which tells the user how much momentum shall influence the training process and is a hyperparameter [30]. The momentum is expressed in Equation 2.33, and with substitution the momentum-implementation to gradient descent is shown through Equation 2.34 [44].

$$\mathbf{m_i} = \lambda \mathbf{m}_{i\text{-}1} - lr \cdot \nabla L(\boldsymbol{\theta}_{i\text{-}1}) \tag{2.33}$$

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i\text{-}1} + \mathbf{m}_{i\text{-}1} \tag{2.34}$$

The momentum is analogous to the velocity, whereas the momentum weighting is analogous to the inertia from classical physics [44]. Momentum can be explained by viewing the loss curve as a hill and the loss as a ball rolling down the hill [45]; when rolling downwards, the ball gains momentum (speed). If there is a small hole (local minima) in the ball's path, instead of falling inside the hole and stopping, the ball will essentially jump over (escape) and continue its path, displaying resilience for sudden changes (inertia).

### Weight decay

Weight decay is a method that prevents *overfitting* by enabling extra punishment for large weights in the loss function, providing a new loss function shown in Equation 2.35 [46].

$$L_{new} = L_{old}(\boldsymbol{\theta}) + \frac{\eta}{2}\boldsymbol{\theta}^2 \tag{2.35}$$

where $\eta$ is the weight decay weighting hyperparameter.

When utilizing weight decay and momentum together in gradient descent, the momentum is now given by Equation 2.36, which in return provides a new equation describing gradient descent, shown in Equation 2.37 [47]. By setting both $\lambda$ and $\eta$ to be zero, Equation 2.37 and Equation 2.30 are equal, resulting in Equation 2.37 being a more general description of gradient descent.

$$\mathbf{m_i} = \lambda \mathbf{m}_{i\text{-}1} - lr \cdot \nabla L_{new}(\boldsymbol{\theta}_{i\text{-}1}) \tag{2.36}$$

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i\text{-}1} + \theta \mathbf{m}_{i\text{-}1} - lr(\nabla L_{old}(\boldsymbol{\theta}_{i\text{-}1}) + \eta \boldsymbol{\theta}_{i\text{-}1}) \tag{2.37}$$

**Stochastic gradient descent**

SGD operates like gradient descent, but by separating the training data into batches with size $n$, which amounts to $k$ number of batches, which will result in less memory expensive training. Hence, some of the most important hyperparameters in SGD are the lr, the number of epochs, and the batch size [30] [48]. Since SGD are gradient descent based, the algorithm is also described by Equation 2.37.

When training the model, because the training data was split into $k$ batches, every parameter will be updated for $k$ iterations for every epoch, which in return introduces noise to the system. The noise introduced to the system can cause the convergence to the optimal parameters to become unstable, however, the noise introduced can also help the model to not converge to sub-optimal parameters [30] [48].

### 2.6.7 Data splitting

Data splitting is how the original dataset is split to one training and evaluation dataset. These splits are dependent on the size of the original dataset and are split in a way that the training set usually contains most data points; some common data splits are 50:50, 60:40, 70:30, 80:20, and 90:10 (training:evaluation). The evaluation set could be separately split into validation and test set [49].

The main purpose of data splitting is to prevent overfitting [49]. Overfitting occurs when the model has trained itself to fit the training data almost perfectly, ignoring the deviations that occur in other sets. To identify overfitting, one can examine the performance of the training set and compare it to those obtained by the validation set. An example of overfitting vs a good fit (generalized) is illustrated in Figure 2.19.

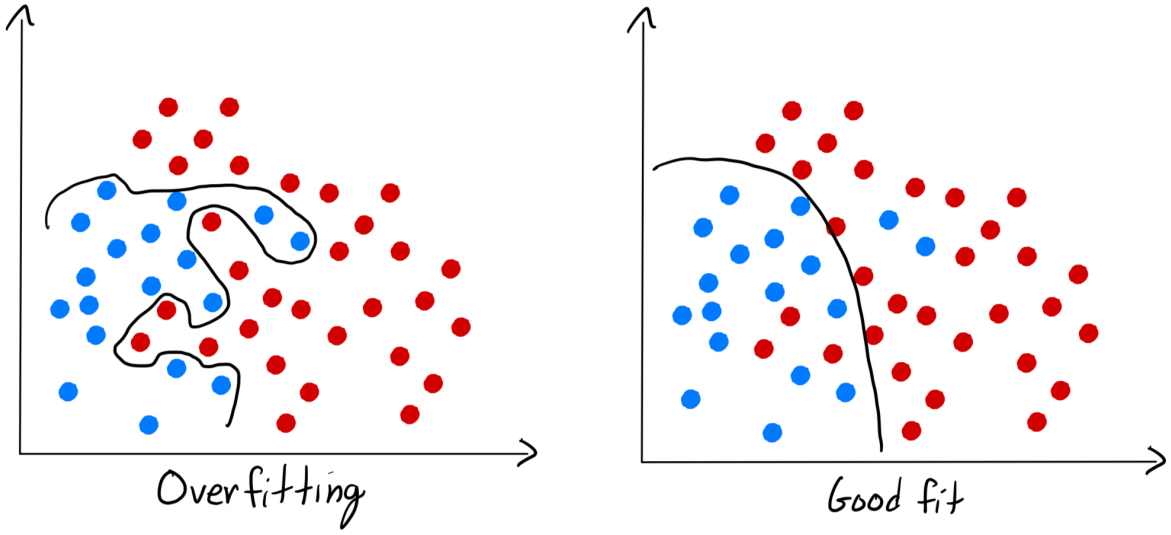


*Figure 2.19: Illustration of a overfitted model (left) and a good fit (generalized) model (right).*

Since the models are trained on the training set, it is also common practice to use the whole evaluation set as the validation set [49]. Even if this technically makes the model selected somewhat biased to the validation set, this can be done when, e.g., dealing with small datasets.

### 2.6.8 Data augmentation

Data augmentation is a method used to compensate for the lack of data. This method modifies the original data; in the case of images, this can be done by resizing the images, tilting the images, and flipping the images, etc., and adding the modified versions to the training data. In addition to compensating for data, data augmentation also provides the model additional flexibility, and can positively influence the generalization of the model by preventing overfitting [30].

## 2.7 Performance measure

To evaluate a model, different performance measures can be used; for medical segmentation, the evaluation metric Sørensen-DICE coefficient (DICE) is often used, and other evaluation metrics such as intersection over union (IoU), Sensitivity, and Precision are also used [50]. To calculate these performance metrics the terms True Positive (TP), False Positive (FP), and False Negative (FN) have to be and are introduced in the subsequent section.

### 2.7.1 True Positive, False Positive and False Negative

TP, FN, FP, and True Negative (TN) are terms commonly used in binary classification tasks, where the label is "True" if the label predicted matches the ground truth, or "False" if the label predicted does not match the ground truth. To classify a detection as TP some threshold, which is usually based on the model's object detection ability, one common threshold is the bounding box IoU [31] [30]. In segmentation tasks, however, the terms TP, FP, and TP refer to the labeling of each pixel and not the labeling of the whole object for object detection.

These terms are important for calculating the performance measures; DICE, IoU, sensitivity, and precision, as expressed in Equation 2.40, Equation 2.43, Equation 2.41 and Equation 2.42 respectively [50].

When dealing with a binary segmentation task of small objects, e.g. lung lesions within a CT lung image, the number of TN-pixels will be significantly larger than the number of TP-pixels, FP-pixels, and FN-pixels and will therefore mathematically be the main contributor if present in an equation. One such performance metric is the specificity metric, which is expressed in Equation 2.38 [50]. From the equation, it is obvious that for every small segmented object, the specificity is approximately one.

$$Specificity = \frac{TN}{TN + FP} \tag{2.38}$$

### 2.7.2 DICE score

DICE compares the area of the predicted mask and the area of the ground truth mask, which essentially measures the overlap. This metric is the most commonly used in medical segmentations, as it provides a direct comparison between the two masks. However, DICE also provides information if the predicted mask falls outside the ground truth mask providing FPs [50]. The DICE score can be calculated by using Equation 2.39, and a visual illustration of the calculation is displayed in Figure 2.20.

$$DICE = \frac{2 \cdot Intersection}{Pedicted\ area + Ground\ truth} \tag{2.39}$$

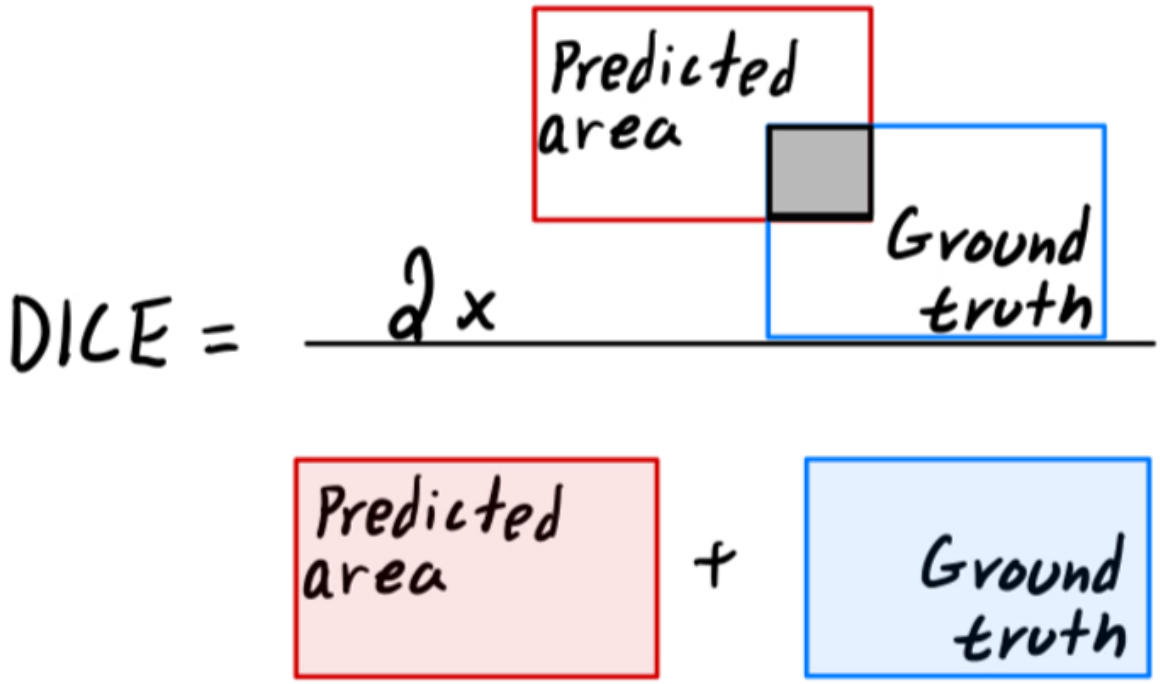


*Figure 2.20: An illustration of how the DICE score is calculated*

From Equation 2.39 one can see that the DICE score varies from zero to one, where one means perfect overlap and zero means no overlap. In terms of TP, FP, and FP Equation 2.39 is expressed as Equation 2.40.

$$DICE = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \tag{2.40}$$

### 2.7.3 Sensitivity

Sensitivity, also known as recall or true positive rate, measures the model's ability to correctly predict a mask compared to the ground truth, similar to DICE. In contrast to DICE, sensitivity, however, provides more information on how the predicted mask misses the ground truth mask by ignoring the FPs in its calculation. The formula is given in Equation 2.41. Since FPs are not included in sensitivity, DICE and sensitivity do not have a 100% correlation and are, in fact, reduced to 74% [50].

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.41}$$

In Equation 2.41, it is obvious that sensitivity can obtain values from zero and one, where one means that the predicted mask covers the whole ground truth mask, and zero means that either the predicted mask misses completely, or there is no predicted mask in that particular image. However, in the case where sensitivity is equal to one,

due to Equation 2.41, one cannot be sure if the predicted mask covers the ground truth mask exactly or if additional regions outside the ground truth mask are included in the predicted mask [50]. This is why sensitivity has to be combined with metrics such as DICE, precision, etc. or a combination of these.

### 2.7.4 Precision

Precision measures the model's ability to correctly predict a mask on the ground truth, similar to DICE. In contrast to DICE, precision, however, provides more information on how the predicted mask is located on the ground truth mask by ignoring the FNs in its calculation. The formula is given in Equation 2.42. Since FNs are not included in precision, DICE and precision do not have a 100% correlation as the sensitivity [50].

$$Precision = \frac{TP}{TP+FP} \tag{2.42}$$

Similar to DICE and sensitivity, the precision can be every value from zero to one. When the precision value is zero, then the prediction is unreliable and overlaps 0%, and when the precision value is one, the prediction mask overlaps 100%. However, when precision is equal to one, due to Equation 2.42, one cannot be sure if the predicted mask covers the ground truth mask fully or just segments of the ground truth [50]. This is why precision has to be reported together with other metrics such as DICE and sensitivity.

### 2.7.5 IoU

Intersection over Union (IoU), like DICE, also measures the overlap between the predicted area and the ground truth area. DICE score is, however, more common in medical segmentation tasks; in other segmentation tasks, IoU is also used. Additionally, IoU is a term more familiar to non-medical individuals with a general knowledge of statistics. The formula for calculating IoU is given in Equation 2.43.

$$IoU = \frac{TP}{TP+FP+FN} \tag{2.43}$$

The DICE score and IoU are, in fact, 100% correlated, and the correlation is expressed in Equation 2.44, which can be verified by substituting Equation 2.40 with DICE in Equation 2.44 [50].

$$IoU = \frac{DICE}{2-DICE} \tag{2.44}$$

Compared to the DICE score, the True Positives are less weighted in the calculation of IoU, which means that the DICE≥IoU as shown in Equation 2.44. From Equation 2.44 it is also clear that IoU and DICE are equal at zero and one.

### 2.7.6 Confusion matrix

A confusion matrix (standard confusion matrix) is a matrix that provides information on the model object detection performance. The confusion matrix sorts the model predictions on the validation set against the ground truths in a comprehensible manner. For binary classification tasks, e.g. lesion or no lesion, a confusion matrix will have a dimension 2x2, as shown in Figure 2.21 [31].

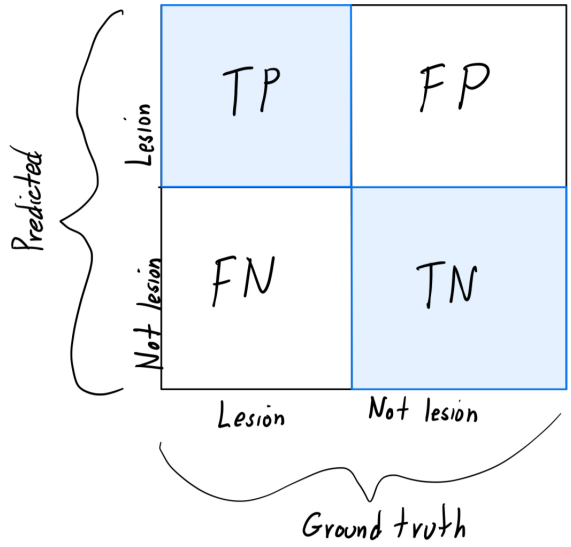


*Figure 2.21: The Figure illustrates an binary confusion matrix*

As Figure 2.21 indicates, for every DL task, it is ideal to only have non-zero values in the diagonal, from the top left corner to the bottom right corner, and zero elsewhere.

#### Normalized confusion matrix

A sub-type of the confusion matrix is the normalized confusion matrix, which is to normalize the cell values to values between zero and one. However, there are two sub-types of these normalized confusion matrices namely, the column-wise normalized matrix and the row-wise normalized matrix [51].

Depending on the layout of the standard confusion matrix, these serve different purposes. The layout used to describe the concepts follows the binary standard matrix layout illustrated in Figure 2.21, where the predicted axis is analogously placed as the

y-axis and vice versa for the ground truth axis.

The concepts are described by calculating the new "TP values" (the other values FP, TN, and FN, follow the same logic). For a column-normalized matrix, the sum of every column is equal to one. Hence, the column-normalized TP-value will represent the sensitivity metric following Equation 2.41. For a row-normalized matrix, the sum of every row is equal to one, and therefore the row-normalized TP-value will represent the precision metric following Equation 2.42 [51]. If the layout was flipped, where the predictions would represent the x-axis, the column-normalized TP-value would represent precision and vice versa. An example of a column-normalized matrix, with the layout presented in Figure 2.21, is displayed in Figure 4.3.

## 2.8 YOLO

YOLO is a popular object detection and image segmentation model developed by Ultralytics. The model not only detects objects for a single image but can also perform object detection in a video feed [52] [53].

In this thesis, the YOLO segmentation algorithm is utilized for developing an automated CT lung lesion model. The algorithm used in this these are the latest version of YOLO namely, YOLO version 8 (YOLOv8) [52]. YOLO has previously been used in a range of medical publications, where YOLO version 5 (YOLOv5) have been used in "Study on Sperm-Cell Detection Using YOLOv5 Architecture with Labeled Dataset", where the goal was to use object detection to identify normal sperm cells [54], and YOLO version 3 (YOLOv3) has been used for lung nodule detection [34].

YOLOv8 allows the user to either train just an object detection model or an instance segmentation model, which also provides information on object detection. The key difference between those models is that an object detection trained model only focuses on locating the object, whereas the segmentation model also tries to segment the objects within the bounding boxes, which can be more demanding.

In Section 2.8.1 the image segmentation model are described in grater detail, which includes the information needed to train the models, the general architecture and how object detection are incorporated within the segmentation model. Henceforth, YOLOv8 will mainly be referred to as YOLO.

### 2.8.1 Image segmentation

In this project YOLO was incorporated perform instance segmentation of CT lung lesions. The CNN algorithms provided by YOLO primarily designed for image segmentation, also has the capability to perform object detection. Hence, the algorithms designed for instance segmentation also provides bounding boxes in its predictions. The general architecture for the instance segmentation model is described in greater detail in the subsequent section.

To train these segmentation models, the model requires information about the location of the object. Since the primary objective for this specific model is to perform instance segmentations, the algorithm requires detailed information about the objects locations. The locations are presented as a detailed map of the objects edges, which is referred to as an edgemap, where each edgemap corresponding to an object is presented with the structure expressed in Expression 2.45.

$$\{CLASS\ ID,\ x1,\ y1,\ x2,\ y2,\ ....\} \tag{2.45}$$

where CLASS ID translates to the label, e.g. lesion, and x1, y1,.. translates to the 2D image coordinates (x1,y1),... representing the edgemap corresponding to the object.

#### General architecture

The general architecture of the instance segmentation model provided by YOLO consists of three main components, namely the backbone, the neck, and the head. An illustration of the object detection architecture is provided in Figure 2.22, the segmentation architecture is built upon the object detection architecture by adding the segmentation features [55] [56].

The backbone is the feature extraction component, utilizes a CNN with weights and biases. Based on the weights and biases, the features are extracted from the original images, then the images are downsampled and passed on to the next layer of the CNN [55] [56] [57].

After the features are extracted from the CNN, they are further processed by the neck. The neck is the component that may enhance the features extracted by fine-tuning them, and this fine-tuning process may executed by a feature pyramid network [56] [57].

Then, the final step of that particular iteration is to use the features extracted to pre-

dict a segmentation. This final step happens in the head, for the segmentation models provided by YOLO, there are usually two "heads", one for object detection and one for segmentation. The object detection head is responsible for predicting object type and provide the corresponding bounding boxes, whereas the segmentation head is responsible for perform pixelwise semantic segmentation inside the bounding box provided by the detection head [56] [57]. However, since the semantic segmentation is inside the restricted bounding box, the segmentation acts like an instance segmentation.

After predicting, the loss is calculated according to Equation 2.46. Before updating the weight and biases accordingly to the chosen optimizer, before undergoing the whole process again.

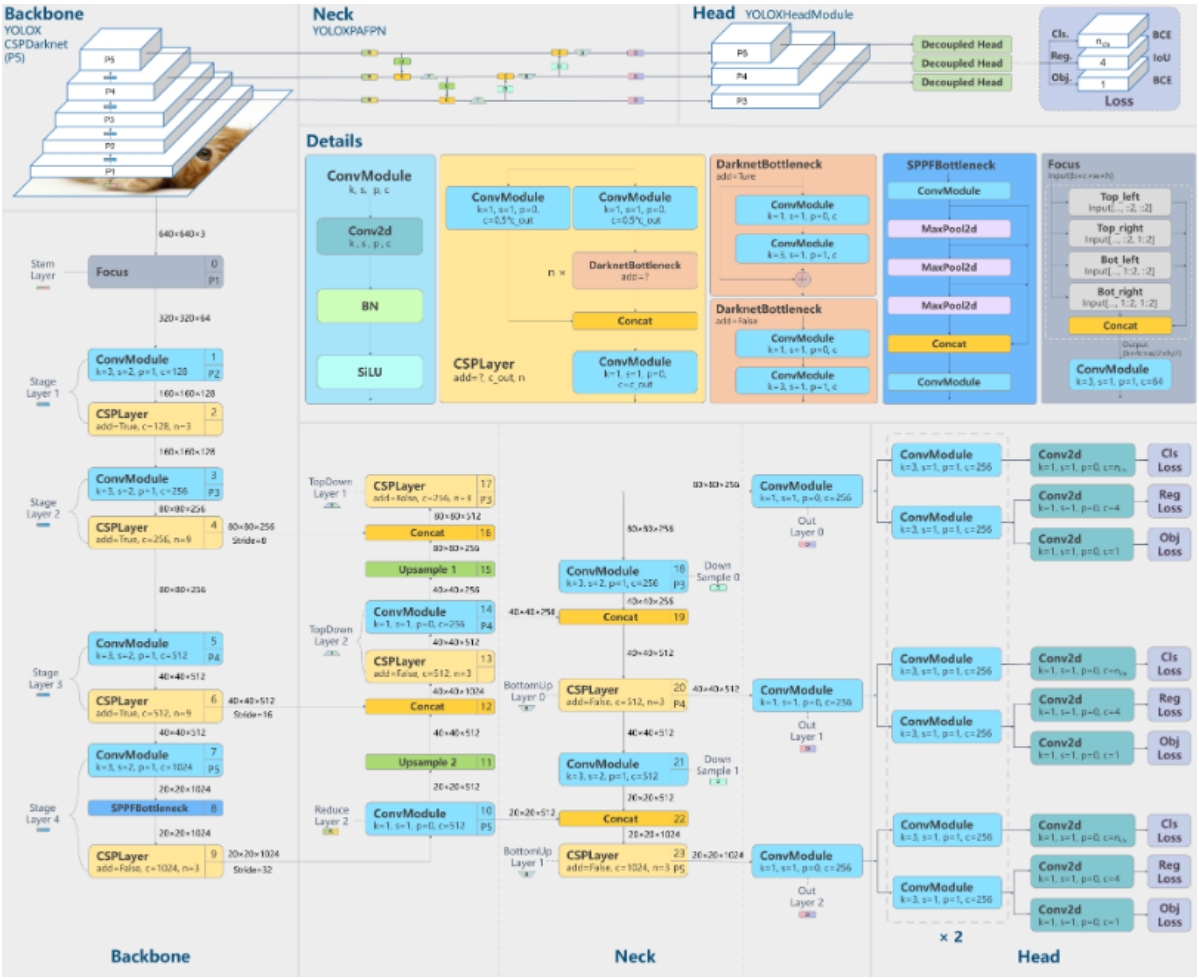


*Figure 2.22: The figure illustrates the architecture to the object detection network provided by YOLO. The architecture is similar to the segmentation architecture, only without the added segmentation features. The figure is acquired from [40]*

Additionally, YOLO provides the opportunity to choose model complexity, also referred to as a model size. The key differences between the sizes are the number of parameters, weights, and biases. The number of parameters chosen has a huge in-

fluence on the performance of the model. Since the number of parameters chosen is proportional to how many features to extract, the number of parameters influences the training time and the quality of predictions made by the model [58].

#### Object detection

Since the segmentation algorithm predicts bounding boxes, in which the semantic pixelwise segmentation is performed, information about the models performance for object detection are also extractable. The algorithm will in fact automatically provide a normalized confusion matrix, which is key to understand the trained models ability to perform object detection.

The normalized confusion matrix provided is a column-normalized matrix, with a similar layout illustrated in Figure 2.21, which again translates to the new TP value equaling the object detection metric, sensitivity. The default threshold values to count a detection as TP are a confidence score greater than 0.001 and an IoU greater than 0.7. However, these thresholds can be changed if desired.

### 2.8.2 Data augmentation

In YOLO, there are several built-in data augmentation techniques that are automatically executed when running the training protocol. Some of these data augmentations are image angle/degree rotation, image translation, image scale, image shear, image flip up-down, image flip left-right, and image cut mix. Not all of them are necessary for every task, since these are task-dependent, and it is easy to disable the undesired ones. To disable or activate the builtin augmentation types, this must be specified in the training command [59].

Image scaling scales the image up and down. This augmentation may be useful in the sense that if a small patient takes a CT scan, the lungs will be small, so by upscaling the image, it may be easier to see the potential lesion. The image flip left-right flips the image horizontally [59]. The main advantage here is introducing "new"/more data to the training.

### 2.8.3 Loss function

The total loss function used in YOLO segmentation models consists of four components: the segmentation loss, the bounding box loss, classification loss (CL), and dis-

tribution focal loss (DFL). The total loss is expressed in Equation 2.46, where segmentation is denoted "seg" and bounding box is denoted "bbox".

$$L_{\text{total}} = \lambda_{\text{seg}} \cdot L_{\text{seg}} + \lambda_{\text{bbox}} \cdot L_{\text{bbox}} + \lambda_{\text{CL}} \cdot L_{\text{CL}} + \lambda_{\text{DFL}} \cdot L_{\text{DFL}} \tag{2.46}$$

where $\lambda_X$ is the corresponding weighting, by default $\lambda_{\text{seg}}$ is equal to $\lambda_{\text{bbox}}$.

The segmentation loss quantifies how close the predicted mask is to the ground truth mask, and this component utilizes Binary Cross Entropy (BCE) loss. The bounding box loss calculates the error between the predicted bounding box and the ground truth bounding box, and this component utilizes MSE. The CL measures the error of predicting the true label within the bounding box, and this component utilizes BCE. The DFL addresses class imbalances within the dataset by weighting the objects which are difficult to classify, and this component utilizes Cross Entropy [60] [61] [62].

### 2.8.4 Hyperparameter tuning

YOLO has a builtin hyperparameter tuning algorithm to optimize the hyperparameters. The builtin hyperparameter tuner utilizes Ray Tune and focuses mainly on mutating hyperparameters before starting the training process. These mutations are randomly chosen from a given hyperparameter search space, which is a set of hyperparameters with certain ranges [63] [64] [65] [66]. An example of a hyperparameter search space is given in Expression 2.47.

$$space = lr0 : tune.uniform(10^{-5}, 0.1),\ lrf : tune.uniform(10^{-7}, 10^{-2}) \tag{2.47}$$

where $space$ is the hyperparameter search space, $lr0$ and $lrf$ are the initial and final learning rate respectively, and $tune.uniform(a,b)$ picks a random hyperparameter value in the range $a$ to $b$.

After picking the hyperparameter values, the model gets trained, and after training, the tuning algorithm selects a new set of hyperparameters and then starts the training process again. This cycle will continue for $N$ iterations, which are predefined. After each iteration, the tuning algorithm provides information on which combination of the best-performing hyperparameters in a YAML format, an example is shown in Figure 2.23. After training each model, for each iteration, every training loss is continuously plotted in graphs. However, the continuous validation loss graphs are not plotted, but only the final validation losses are given, an example is shown in Figure 2.24 [63] [64].

```
# 8/20 iterations complete ✅ (125059.11s)
# Results saved to runs/segment/tune8
# Best fitness=0.67846 observed at iteration 3
# Best fitness metrics are {'metrics/precision(B)': 0.76537, 'metri
# Best fitness model is runs/segment/train120
# Best fitness hyperparameters are printed below.

lr0: 0.01095
lrf: 0.01092
momentum: 0.92674
weight_decay: 0.00051
warmup_epochs: 2.8179
warmup_momentum: 0.89255
box: 7.5
cls: 0.46337
dfl: 1.47135
hsv_h: 0.01568
hsv_s: 0.68346
hsv_v: 0.4086
degrees: 0.0
translate: 0.0897
scale: 0.39769
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.471
mosaic: 0.99669
mixup: 0.0
copy_paste: 0.0
```

*Figure 2.23: Example of the current best-performing hyperparameters during a hyperparameter tuning process, iteration 8 of 20.*

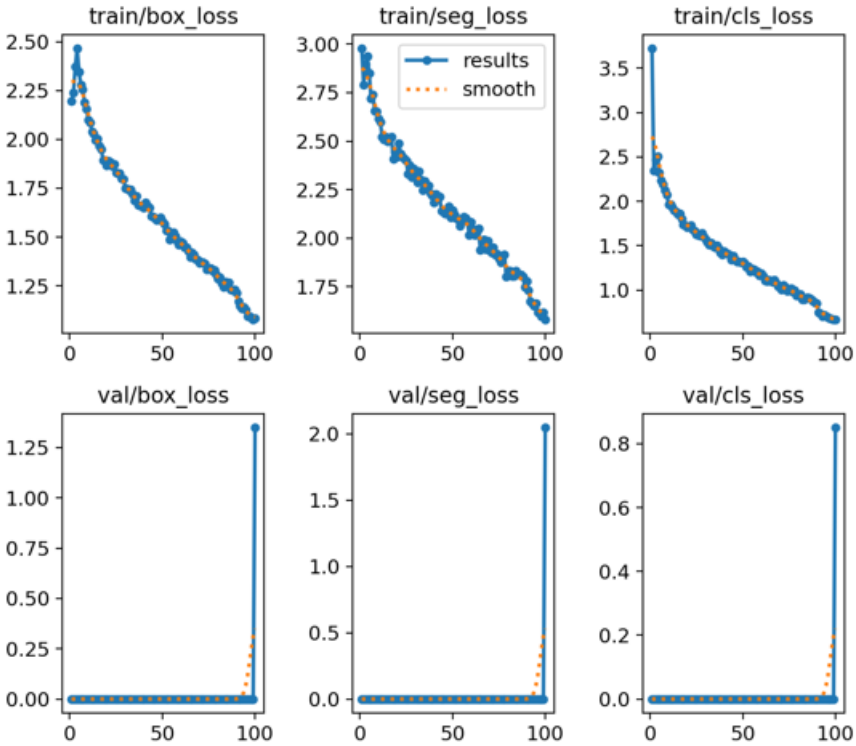


*Figure 2.24: Example of training loss curves and the lack of validation loss curves after training a model with YOLOv8's builtin hyperparameter tuner. For each curve, the x-axis represents the number of epochs, and the y-axis represents the corresponding loss.*

## 2.9 Recent advances

Automated segmentation of CT lung lesions using DL approaches has been of interest over the last decade, and there has been purposed several models, usually based on networks such as AlexNet, VGG, UNet, and GoogleNet [67], for dealing with this particular task, both for 2D slices and 3D volumes. Several studies about this topic have been conducted by various research teams and published in multiple journals [68]. However, these studies are not only restricted to lesion segmentation, but also malignancy score given lesion, lesion identification (e.g. metastasis or primary tumor), etc. Multiple of these studies, however, have specific requirements which needs to be satisfied for a lesion to be counted in the dataset, e.g. lesion diameter threshold and lesion verification from multiple radiologists as done by Pezzano et al. [67].

Zhi et al. [68] compared multiple DL networks for CT lung lesion segmentation. For 2D segmentations, the models compared in the article managed to achieve DICE scores ranging between 0.680-0.950 (one study managed to achieve a DICE score equal to 0.969. However, the training and evaluation were done on private datasets), and between the models, the mean performance for sensitivity and precision ranged between 0.584-0.874 and between 0.820-0.869 respectively. Notably, these scores are obtained after training and validating various datasets (including LIDC-IDRI), including the number of images, image sizes, and loss functions.

YOLO has previously been utilized in studies for lung lesion detection. YOLO version 3 (YOLOv3) has already been used in the LIDC-IDRI dataset by Liu et al. [34]. In this study, the objective was lung nodule detection, and the performance measure used is the sensitivity. The models trained managed to achieve an averaged sensitivity score of 87.3%. However, this study also had a specific requirement where CT scans with slice thickness above 2.5 mm were excluded.

Another similar study, where YOLO is utilized and on the LIDC-IDRI dataset by Liying et al., is one where YOLOv5 was combined with an additional feature extraction network and a feature extraction structure to train a model for detecting lung nodules. The metric score for this study was also sensitivity, and the trained model achieved a sensitivity score of 96.2% for nodule detection. However, not all CT images were used, and the model was only trained on 1987 CT images [35].

Since the YOLO has already been used and documented to detect CT lung lesions,

using YOLO, especially YOLOv8 (released 2023), for CT lung lesion segmentation is less documented. Limitations when comparing the generated models in this thesis to other studies are the various requirements used, datasets, and loss functions used to obtain the results mentioned in this section. However, the results from this thesis can provide important information and insights about YOLO in the use of CT lung lesion segmentation, and in the field of CT lung lesion segmentation in general.

## 2.10 Aim

This thesis aims to explore the feasibility of training a DL model to detect and segment CT lung lesions in 2D, and to establish a local clinical workflow utilizing an integrated DL model to segment CT lung lesions.

This is achieved by adapting the CT lung images presented in the predefined manually labeled LIDC-IDRI dataset to fit the YOLO criteria. YOLO is used because it demonstrated promising results in terms of lesion detection and since YOLO's instance segmentation is directly based on its detection performance. After training the models, the results are evaluated based on the segmentation performance measures: DICE score, IoU, sensitivity, and precision, and presented in Chapter 4, and the object detection performance measure: sensitivity.

Finally, a well-performing model will be uploaded to a clinical workflow, the regional research PACS, providing proof of concept. The results obtained by testing the model with test data from the PACS system will be presented in Chapter 4.

# Chapter 3

# Methods

The data used for training the models in this thesis is the online dataset LIDC-IDRI [69], which consists of a collection of primary lung tumors and secondary lung tumors/metastasis. This particular dataset has been used in several papers to train DL networks for both lesion segmentation and lesion detection such as: the segmentation models compared by Zhi et al., and the object detection models presented by Liu et al. and Liying et al. Since YOLOv8 introduces instance segmentation and there are no (or few) paper(s) that utilize YOLOv8 on the LIDC-IDRI dataset, YOLOv8 was chosen as the DL model for this thesis.

In the current chapter, the data used in the current work are described alongside the various pre-processing steps needed to handle the data. Additionally, the application and adaption of DL are described in addition to the evaluation metrics applied. Finally, the approach toward implementing a final pipeline for processing CT lung images in a clinical research workflow is described.

The scripts used in the methodology, and used for providing the results in Chapter 4 are provided in the GitHub repository in Appendix II.

## 3.1 Data

The data used in this project is acquired from the online dataset LIDC-IDRI from the Cancer Imaging Archive [69]. The whole dataset with 1012 patients was downloaded, in the dataset, patients with malignant primary lung cancer, with malignant lung metastasis, and patients with benign or non-malignant diseases are showcased. The dataset contains both LDCT and standard-dose CT, with both nodules and non-nodules marked by four radiologists.

A nodule is an abnormal mass that appears in a CT scan, which is considered to be either a primary lung cancer tumor, metastatic tumor, a noncancerous process, or indeterminate (analogous to 3D lesion in this thesis). These nodules will vary in shape, size, density, and location in the lung. A non-nodule is also an abnormal mass which does not satisfy the criteria to be a nodule, and these nodules are not cancerous [70].

In the dataset and XML files, the nodules and non-nodules (the labels) are divided in three categories according to the longest possible straight line between two points in the abnormal mass.

1. Nodule $\geq$ 3 mm
2. Nodule $<$ 3 mm
3. Non-Nodule $\geq$ 3 mm

In the collection of 1012 patients from the dataset, the four radiologists identified a total of 2669 nodules $\geq$ 3 mm and 4702 nodules $<$ 3 mm and non-nodules $\geq$ 3 mm combined. However, in order to be included in the list only one radiologist had to label the masses. Only 928 of the 2669 and 1012 of the 4702 were unanimously agreed upon by the four radiologists to have the same label. There are several reasons for these deviations; one reason is that two separate nodules might look like one single for some radiologists or vice versa, and another reason is that some small irregularities for a lesion with a length less than 3 millimeters, is either wrongfully labeled or not identified by some radiologists [70]. In this thesis, every DICOM image (slice) with either a nodule or non-nodule were converted from a 2D DICOM slice to a PNG image.

In the XML files to the CT scans, the non-nodules and small nodules (nodules $<$ 3 mm) only have the center of mass coordinate attached to the specific mass object, whereas the other nodules have multiple coordinates, which represent the outline of the nodule, attached to the specific nodule object. In a segmentation and an object detection task, multiple coordinated are required for generating the segmentation mask and computing the bounding box. As a result every small nodule and non-nodule were discarded after the DICOM to PNG conversion. However, given the geometry and orientation of big nodules, one 2D image, which only contains one nodule coordinate of the big 3D nodule ($\geq$ 3 mm) was also discarded.

However, not only the images with one coordinate were removed, but nodules with less or equal to four nodule coordinates were also removed. The removal of nodules

with less or equal to three nodule coordinates was done because the python library *shapely* [71] needed at least three coordinates to create a polygon with non-zero area, which is an important part in the following pre-processing steps. In the XML file, however, for nodules with more than one nodule coordinate, the first nodule coordinate is mentioned twice, for closing the polygon, which means for a nodule with two unique nodule coordinates, the XML file will provide three coordinates. Mathematically, two unique coordinates will only create a line that has no area. As a result, nodules with a total of less or equal to three coordinates had to be removed.

During the removal of nodules with less or equal to three nodule coordinates, an error occurred in the custom-made script, which accidentally removed nodules with less or equal to four nodule coordinates. However, as displayed in Table 3.1 only five images were lost, but the number of nodule slices lost is unknown. Regardless, the models were trained with the remaining images which might impact the trained model to some degree. Figure 3.1 displays one of the 7152 slices left, after the conversion from DICOM to PNG and after the removal of 2D nodule slices less or equal to four nodule coordinates. Of the 7152 slices, 5721 were used in the training set, whereas the remaining images were split equally into a validation and test set (716 and 715 images respectively).

| n-coordinates: | Number of images after removal of n-coordinates: |
|---|---|
| n=3 | 7157 |
| n=4 | 7152 |

*Table 3.1: Number of images left from the LIDC-IDRI dataset after the removal of n nodule coordinates.*

*Figure 3.1: A CT lung image in PNG format converted from DICOM format, which contains a slice of a big nodule (nodule $\geq$ 3 mm).*

The CT scans conducted on the patients and the image quality varied to a significant degree, where CT scans were taken with several parameters that differed between the scans, which in return caused the LIDC-IDRI dataset to be inhomogeneous. Additionally, the existence of image artifacts is unknown, these have the capability to negatively impact model performance. However, except patient exceeding the field of measurement artifact, these were not observed after training the models, and if present they did not effect the trained models to a significant degree (Chapter 4). For the images with the exceeding field of measurement artifact, the lungs were present.

In the dataset there were a total of four different CT scanner manufactures, where the manufactures also had several models, in which the majority of the scans were executed by GE Medical Systems LightSpeed scanner models and Siemens Definition, Emotion, and Sensation scanner models with 66% and 20% of the total scans respectively. The range of tube peak potential were from 120 kV to 140 kV, where the majority of the acquisitions taken with 120 kV and 140 kV, with 80% and 10% respectively. The most dominant slice thickness were 1.25 mm and 2.5 mm, with 34% and 32% respectively, the range of slice thickness however were from 0.6 mm to 4.0 mm. The tube current varied from 40 mA to 627 mA, and the pixel spacing, the true physical spacing between two pixels in a 2D image, ranged from 0.461 mm to 0.977 mm [70].

## 3.2 Data processing

The following section describes the pre-processing step and post-processing step, of the data, executed in this thesis needed for running the models and acquiring the results in Chapter 4, which provides the basis for the discussion in Chapter 5 and conclusion in Chapter 6.

In Section 3.2.1, the pre-processing steps taken before training each DL model are given in detail, along with the reasoning behind. First, how images of the YOLO-incompatible DICOM format were converted to the YOLO-compatible PNG format. Then, how to extract the lesion segmentation outlines from the XML file and convert them to a YOLO-compatible txt formatted label file. Subsequently, how the challenge of dealing with multiple segmentations, drawn by the various radiologists, over the same region was managed. Lastly, how to size restrict the lesions areas in retrospect to the whole image, while still including patient with various physical sizes.

In Section 3.2.2, the post-processing step taken after the training was completed, for

acquiring the results in Chapter 4. This section addresses the fact, that YOLO by default does not provide the user with the evaluation metrics listed in Section 2.7. The solution involves implementing a custom made script with necessary adjustments, to ensure that the values of the evaluation metrics are correct when segmenting a lesion correctly.

### 3.2.1 Data pre-processing

The conversion of images in DICOM format to images in PNG format were executed two times. The pre-processing method which were ultimately used to train the models is described first. The latter method will be described at the end of this section, and also provides the reason for not further pursue using these converted images to train and evaluate models. The major and most crucial difference between the two conversions is the command used to convert the images from a DICOM format to a PNG format. The whole pre-processing steps ultimately used are illustrated in the flowchart in Figure 3.2.

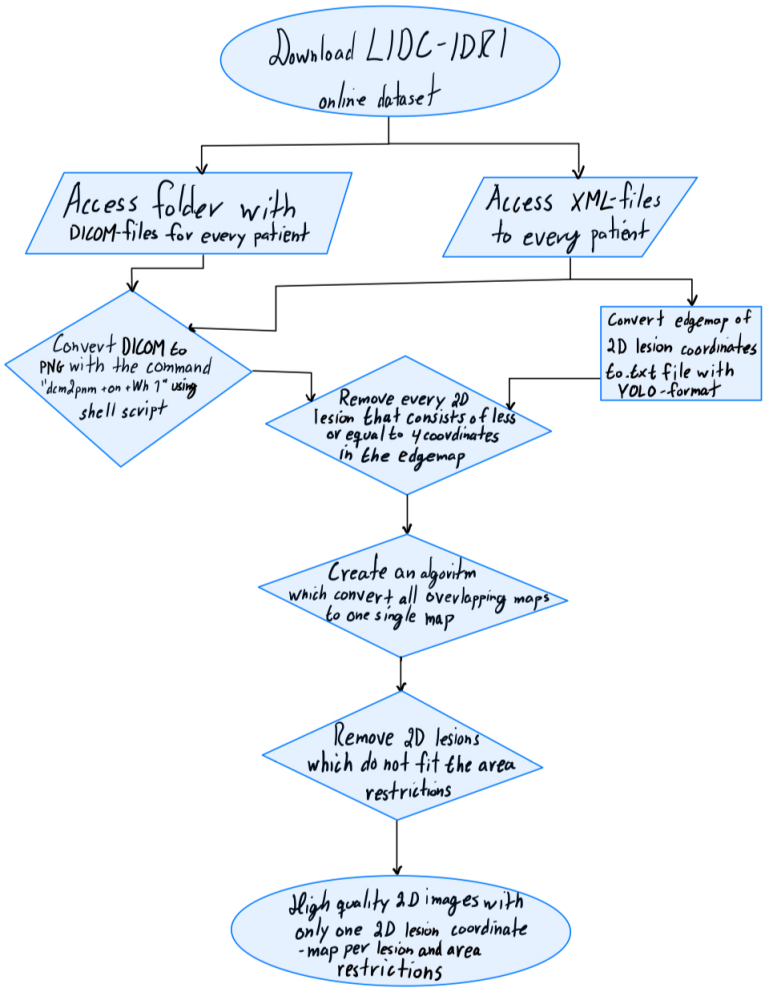


*Figure 3.2: Data conversion: The flowchart illustrates the data conversion with the pre-processing steps that were used to provide the results in this thesis.*

### The used conversion of DICOM to PNG images

After getting poor results with the other conversion process, illustrated in Figure 3.5, another attempt to convert the images was made. This conversion provided more success than the first attempt and was used further for providing the results in this thesis and uploaded to the clinical PACS (Picture archive and communications system) research system.

The first pre-processing step was to convert the DICOM images to PNG images such that they could be compatible with YOLO's detection and segmentation algorithms. This was done by locating every *SOP Instance UID* corresponding to every slice in the dataset in the XML file. The Standard Operating Procedure (SOP) Instance Unique Identifier (UID) is a unique identifier for each DICOM file that corresponds to a unique image slice [72]. After locating the SOP Instance UIDs, the corresponding DICOM images were converted to PNG images.

In this approach, four conversions were considered. These four alternatives were *+Wi*, *+Wn*, *+Wm* and *+Wh 1*, the names indicate the types of conversions and are all VOI (values of interest) LUT (Lookup table) transformations form the DICOM toolkit DCMTK package dcm2pnm [73]. In Figure 3.3-a) - d) the +Wi, +Wn, +Wm, and +Wh 1 conversions are shown respectively.

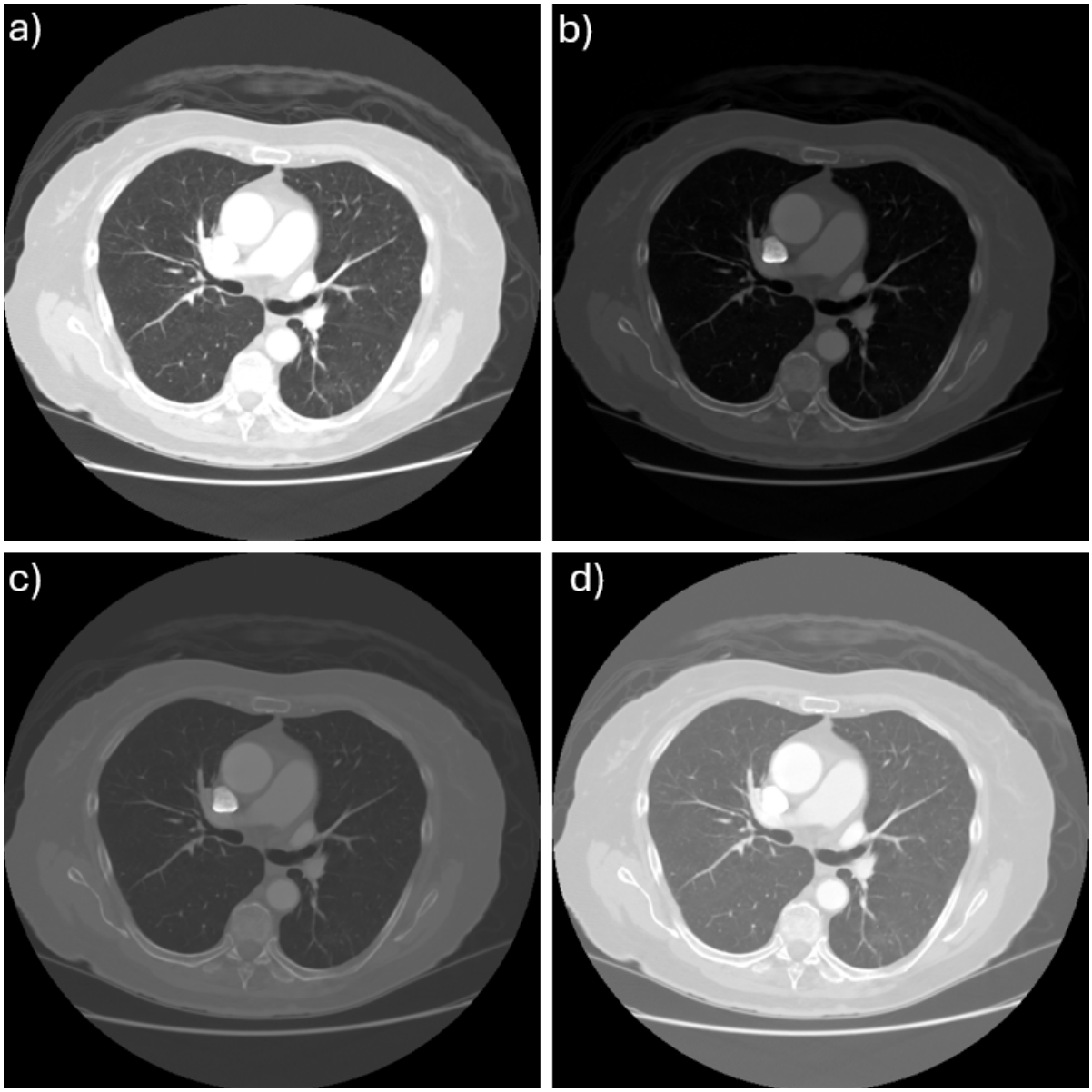


*Figure 3.3: The +Wi, +Wn, +Wm and +Wh 1 image conversions provided by [73].*

The biggest contrast between the minimum and maximum pixel values in the generated PNGs is clearly in Figure 3.3-a) and Figure 3.3-d), which provides a broader range of pixel values the DL model can extract features from. The difference between the VOI LUT transformations +Wh 1 and +Wi is that; +Wh 1 computes a new VOI window by using a histogram algorithm and ignores 1% of the extreme pixel values (brightest and darkest), in contrast to +Wi which only uses the first VOI window in the DICOM file [73].

After reviewing these different converted PNG images, the alternative +Wh 1 was chosen to be further used in this thesis. The decision was made because +Wh 1 computes a new VOI window while excluding extreme pixel values, and +Wi does not provide the

same compensation. Thus, if images have poor image qualities with extreme values, due to e.g. such artifacts illustrated in Figure 2.10-a) and Figure 2.10-e), +Wh 1 could have improved the image qualities, whereas +Wi would not. The exact command used to convert the images from DICOM to PNG in the terminal is expressed in Expression 3.1.

$$dcm2pnm\ +on\ +Wh\ 1\ \text{"}\$dcmfile\text{"}\ \text{"}\$newfilename\text{"} \tag{3.1}$$

#### Converting XML file to txt

In the XML file, there is also information about the coordinates of the lesions inside the lungs. These coordinates were generated after multiple radiologists had drawn an outline for the tumor region. In addition to an outline, the coordinates associated with a given lesion are also referred to as an edgemap. Given that multiple radiologists had drawn these outlines, one lesion could have multiple segmentations/outlines with slightly different coordinates. These outlines were later extracted and put in a txt file corresponding to the input for the segmentation model (Expression 2.45).

#### Create overlapping maps

In the beginning, every outline drawn by the radiologists was kept. However, this provided poor performing models. One probable cause for this is that the models were confused about where to predict, given several ground truths in the same region.

To eliminate this form of confusion, it was decided that only the overlapping regions the different radiologists agreed upon were kept. This was done by creating a custom made script. As a result, from every outline drawn by the radiologists, only one outline representing the overlapping region were kept, Figure 3.4 illustrates an example of the different outlines drawn by radiologists in color, and the outline for the overlapping polygons in black.

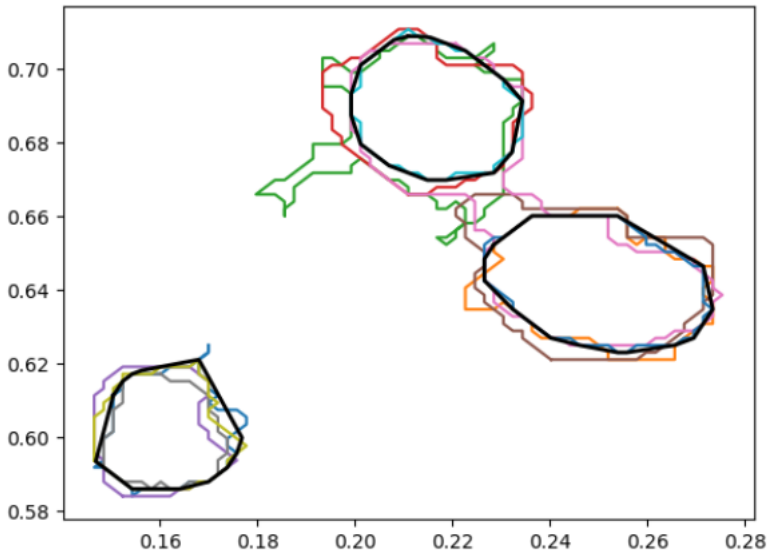


*Figure 3.4: Illustration of outlines drawn by the radiologists to mark the tumors shown in color, and the outlines of the overlapping regions shown in black.*

From Figure 3.4, one can see that the overlapping outlines slightly overshoot, but since these deviances are small, they were deemed insignificant. However, in order to create these overlapping outlines, some lesions had to be excluded, namely, the lesions that had less than four coordinates in the edgemap. The reason for removing these coordinates is that shapely needed at least three coordinates to create a polygon, as mentioned in Section 3.1.

#### Size restriction

The final step in the pre-processing process was to provide the option to size restrict lesions in the datasets. This option provides the opportunity to train different models with different and specific lesion area ranges, and could be used to compare the performance for each model trained with specific areas to one another or to the models trained on the whole dataset. Another property size restriction provides is filtering out large lesions that a radiologist can easily recognize as a lesion and small lesions that can be of less clinical relevance, which in return can improve the trained model by ignoring such cases.

The filtering process based on lesion sizes was attempted in two separate ways, each with custom programmed scripts. The first approach was to upscale the pixels in a way one got the actual physical areas of the lesions from each slice. This was done by multiplying the dimensions to the DICOM image, which in the LIDC-IDRI dataset was 512$x$512, and multiplying the pixel spacing in each direction, $x$ and $y$, which are acquired from the DICOM header to the image. The given approach provides the phys-

ical description of each 2D lesion. However, this approach has one limitation and one flaw. The limitation is that this custom made script was severally time expensive, in the script each image had to be compared to every DICOM file until the right image was found. Some examples of how time expensive the approach was are shown in Table 3.2. The flaw is that, given that the images are in 2D and the pixel spacing for each DICOM file is unique, images which seem to obtain lesions with identical areas, when visually examined, might actually have different physical areas. A comprehensible example is a CT lung scan from a child with lung cancer vs a CT lung scan from an adult with lung cancer, given that the sizes of the tumors are relative equal in retrospect to their bodies, the actual physical sizes will differ by a severe amount. As a result the flaw will likely result in confusion for the model, because the model only knows how the area of the lesion visually looks, rather than their actual physical area.

| Area range $[mm^2]$ | Conversion time $[days]$ | Acquired images |
|---|---|---|
| $\leq 49$ | $\approx 5$ | 164 |
| $49 \leq$Area range$\leq 300$ | $\approx 7$ | 52 |
| $250 \leq$Area range$\leq 2000$ | $\approx 7$ | 18 |

*Table 3.2: Demonstration of how time expensive filtering out 2D lesions based on psychical area are with the custom made script.*

The second approach to filter out lesions to the desired sizes is by the relative area of the 2D lesion from the image compared to the whole image itself. Even though this approach does not capture the actual physical size of the lesion, this method will provide lesions that seems to have the same relative area according to the whole 2D images rather than the physical, which will result in a less confused model. Also, in contrast to the first described approach, this approach will only take a few minutes at most.

In this thesis, the latter approach was chosen due to both the time expensiveness and the challenge of patient size and pixel spacing. The exact range chosen was lesions with areas between 0.05% to 0.4% relative to the whole image. In this range, both the largest and the smallest lesions were removed; the remaining dataset contained a sufficient amount of images to train and evaluate the model, with a total of 1978 images. Additionally, the difference in relative lesion areas in the remaining dataset is a factor of approximately 10, which still allows the model to segment lesions of various sizes and not just one fixed size. The large image reduction, from 7152 to 1978, also reduces the model training time, allowing for a more thorough hyperparameter search. Of the 1978 images 1582 images was used in the training set, whereas the remaining images was split equally into a validation and test set (198 images each).

### The discarded conversion of DICOM to PNG images

The discarded training and pre-processing steps are illustrated in the flowchart presented in Figure 3.5. Only by first following the steps provided in the provided flowchart, and realizing that this approach would only provide poor results, the more optimal suited approach presented in Figure 3.2 was discovered. The steps of trying and failing to produce good results with this approach is described in greater detail in this section.

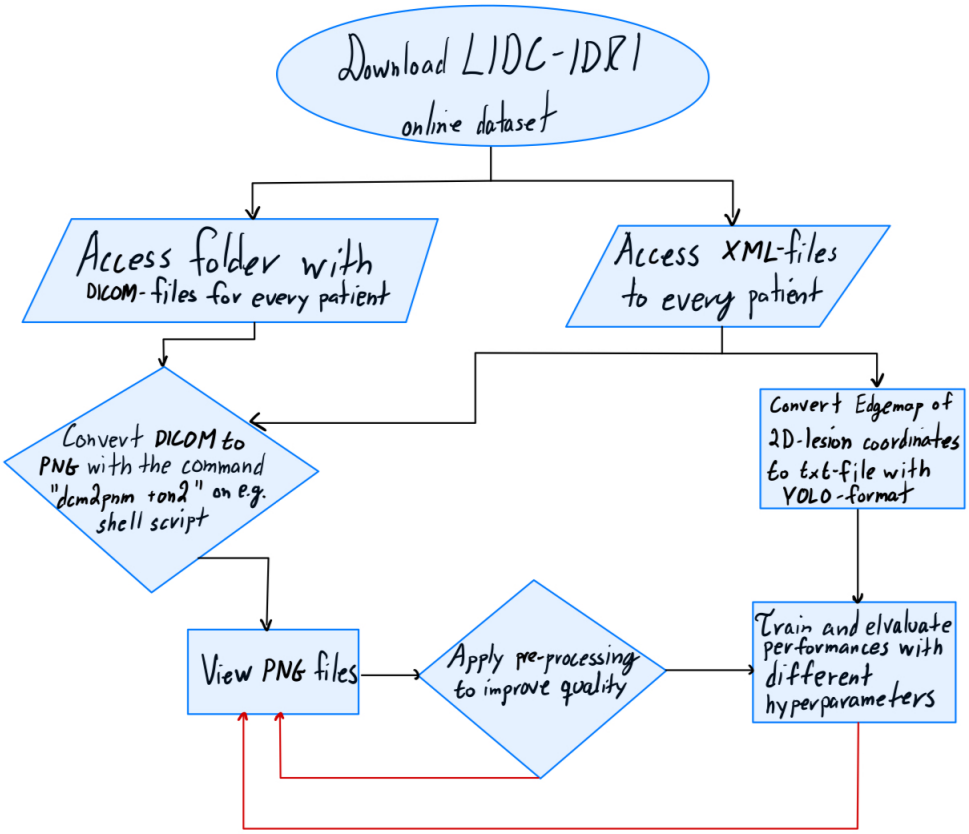


*Figure 3.5: Example of the flowchart with the conversion command "dcm2pnm +on2 "$dcmfile" "$newfilename"".*

A simple conversion from the DICOM image to an 16-bit PNG were executed with the command "dcm2pnm +on2 "$dcmfile" "$newfilename"" in bash shell script which is documented in [73]. These images had poor image quality and is shown in Figure 3.6.

*Figure 3.6: Example of an simple converted image with the command "dcm2pnm +on2 "$dcmfile" "$newfilename"".*

To improve the image quality, a custom made script were implemented to intensify different pixel values given a range, as illustrated in Figure 3.7 and Figure 3.8. The two images both have Figure 3.6 as their origin.

*Figure 3.7: Example of an image with intensified pixel values from an image of the same quality as Figure 3.6.*

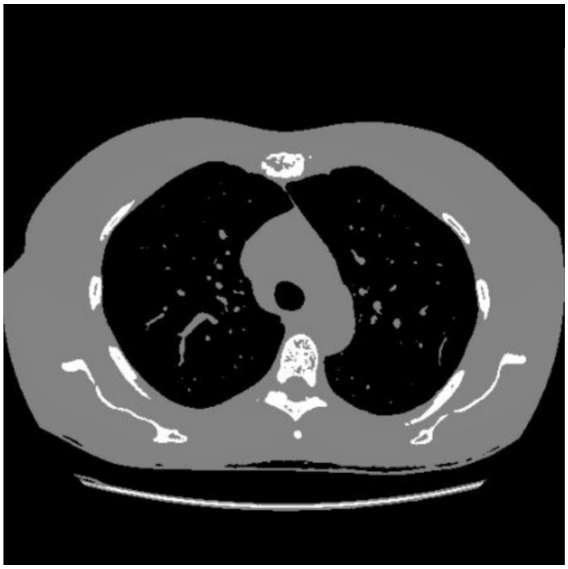

*Figure 3.8: The same image as in Figure 3.7 but the intensified pixel values conversion have different limits.*

The next step were to train different object detection models with different hyperparameters, when only obtaining very poor results, the images were reviewed modified and then trained on again from scratch. Regardless of how many iterations, the intensified pixel values were modified, every model would provide poor results. To do further work on these images and upload them to the PACS research system, would be time expensive and would not provide any advancement in terms of research on clinical data. As a result, the datasets with these images were abandoned.

Note that the dataset containing the types of images shown in Figure 3.6, Figure 3.7 and Figure 3.8 were not used in the segmentation model, and were only in object detection, due to the significant increase of image quality and results when converting with "+Wh 1".

### 3.2.2 Data post-processing

This section provides the implementation and adjustments made, for being able to evaluate the trained model's performances using one of medical segmentations most common evaluation metric, the DICE score, together with other evaluation metrics. The possible consequences of the adjustments are examined in the discussion.

#### Implementation and adjustments

The evaluation metrics, DICE, IoU, precision and sensitivity, used to evaluate the segmentation model is not calculated in YOLOv8 [52]. As a result, custom programmed scripts were created to generate these metrics.

In this thesis, there may be several predicted masks for lesions and several ground truth masks in one image. This causes some issues with the evaluation metric calculations. When the evaluation metric score is equivalent to zero, there are now four possible cases, as shown in Figure 3.9, where each case are described in the following list. In Figure 3.9, predicted masks are denoted "P" and the ground truth masks are denoted "GT".

1. The model provides a predicted mask in a region the radiologist deemed healthy (FP), as illustrated in Figure 3.9 number 1.

2. The model doesn't provide a mask where the radiologist marked a region (FN), as illustrated in Figure 3.9 number 2.

3. The predicted mask misses the GT mask by a small margin/nearly touching, as illustrated in Figure 3.9 number 3.

4. The DICE score is calculated between one GT mask and not the corresponding predicted mask. In Figure 3.9 number 4., P1 and G1 is the corresponding masks, and the DICE score calculated between P1 and G2, and between P2 and G1 are zero.

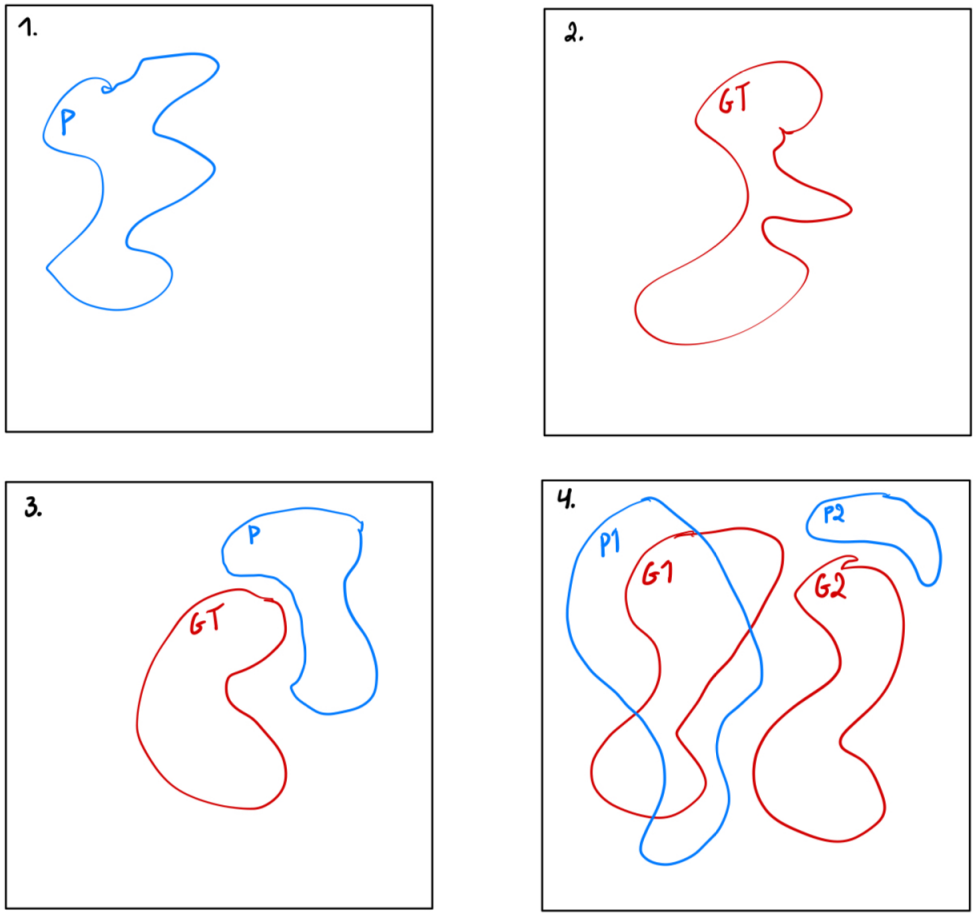


*Figure 3.9: The figures illustrates the four possible cases for obtaining metric score equal to zero. Note "P" is the predicted mask and "GT" is the ground truth. 1: The model provides a predicted mask in a region the radiologist deemed healthy (FP). 2: The model doesn't provide a mask where the radiologist marked a region (FN). 3: the predicted mask misses the GT mask by a small margin/nearly touching. 4: The metric score is calculated between one GT mask and not the corresponding predicted mask.*

As a result the evaluation metric scores were calculated without these cases, because the inclusion of these cases would not be representative for how well the model performed for the cases which it actually detected a mask that overlapped with the GT, and because by including point 4. will severely impact the performance in a poor manner. Another fact to consider is that the GT masks are only masks which the radiologists knows there exists lesions, which means that there is a possibility a lesion exists when a mask is predicted where there is not a GT mask (case 1).

Another adjustment with the evaluation metric scores is rather than calculating the overlapping area directly, counting the number of common pixels were done (Section 2.7.1). This is because a pixel has dimension 1x1, which translates to an pixel area of 1, thus the number of common pixels will represent the intersection, illustrated in Figure 2.20.

## 3.3 YOLO

This section describes how the DL algorithm, YOLO, was utilized in this thesis. First, the model size selection process is described. Then the hyperparameter search strategies were executed, and are described in the subsequent section.

Only the models primarily designed for instance segmentation was utilized, because these segmentation models also performs object detection. Additionally, after the training is done the model automatically provides a column normalized confusion matrix, with the object detection sensitivity performance (Section 2.8.1).

The model sizes utilized in this thesis are the small YOLO segmentation model (yolov8s-seg), the medium YOLO segmentation model (yolov8m-seg) and the large YOLO segmentation model (yolov8l-seg). The difference between the models are the complexities of their architecture, namely the parameters (weights and biases), where the small model, medium model and the large model have 11.8 million, 27.3 million and 46.0 million parameters respectively [58]. These sizes were chosen to visualize the impact of model sizes, thus their complexities, between the to detect and segment CT lung lesions.

### 3.3.1 Hyperparameter search

There kinds of hyperparameter searches executed. First, the grid search strategy with a custom hyperparameter search was conducted. Secondly, the builtin hyperparameter tuner algorithm, provided by YOLO, was utilized for the hyperparameter search 2.8.4.

The custom grid search was conducted by training various models with yolov8m-seg, with a vast number of hyperparameter combinations. Some of the hyperparameters used with the custom grid search algorithm, is presented in Table 3.3. The models were mainly trained with the medium model, due to the training time required by the large models. Additionally, the medium model is the model which is both second least and most complex.

| Hyperparameter | Values | Hyperparameter | Values |
|---|---|---|---|
| lrs | {0.0001, 0.001, 0.01 } | dropouts | {0, 0.2, 0.3} |
| scales | {0, 0.1, 0.2} | momentums | {0.937, 0.947, 0.957 } |

*Table 3.3: The hyperparameter space used in the grid search, where lr0 and lrf are set to be equal and are denoted as lrs*

The hyperparameter tuner utilized used the default hyperparameter search space when searching for optimal hyperparameters. The reason for not executing the tuner with custom hyperparameter spaces is due to an error when trying to create custom spaces, which seems like an error multiple YOLO users encounter [74]. Hence, the hyperparameter tuner is only applied once for each model size. The builtin hyperparameter tuner was utilized only for the dataset without size restriction, and the results acquired are given in Appendix III. The default hyperparameter space can be acquired at Ultralytics YOLOv8 documentation [63],

## 3.4 Uploading to PACS

The final part of this thesis was to establish a local clinical workflow of integrated DL for CT lung lesions. Therefore, a well-performing model was uploaded onto the regional PACS research system, which allows the model to detect and segment lesions on regional data and can thus be used for further research within the regional PACS. The implementation process follows the flowchart provided in Figure 3.10. Additionally, the flowchart presents a method for further improving the uploaded model by retraining it.

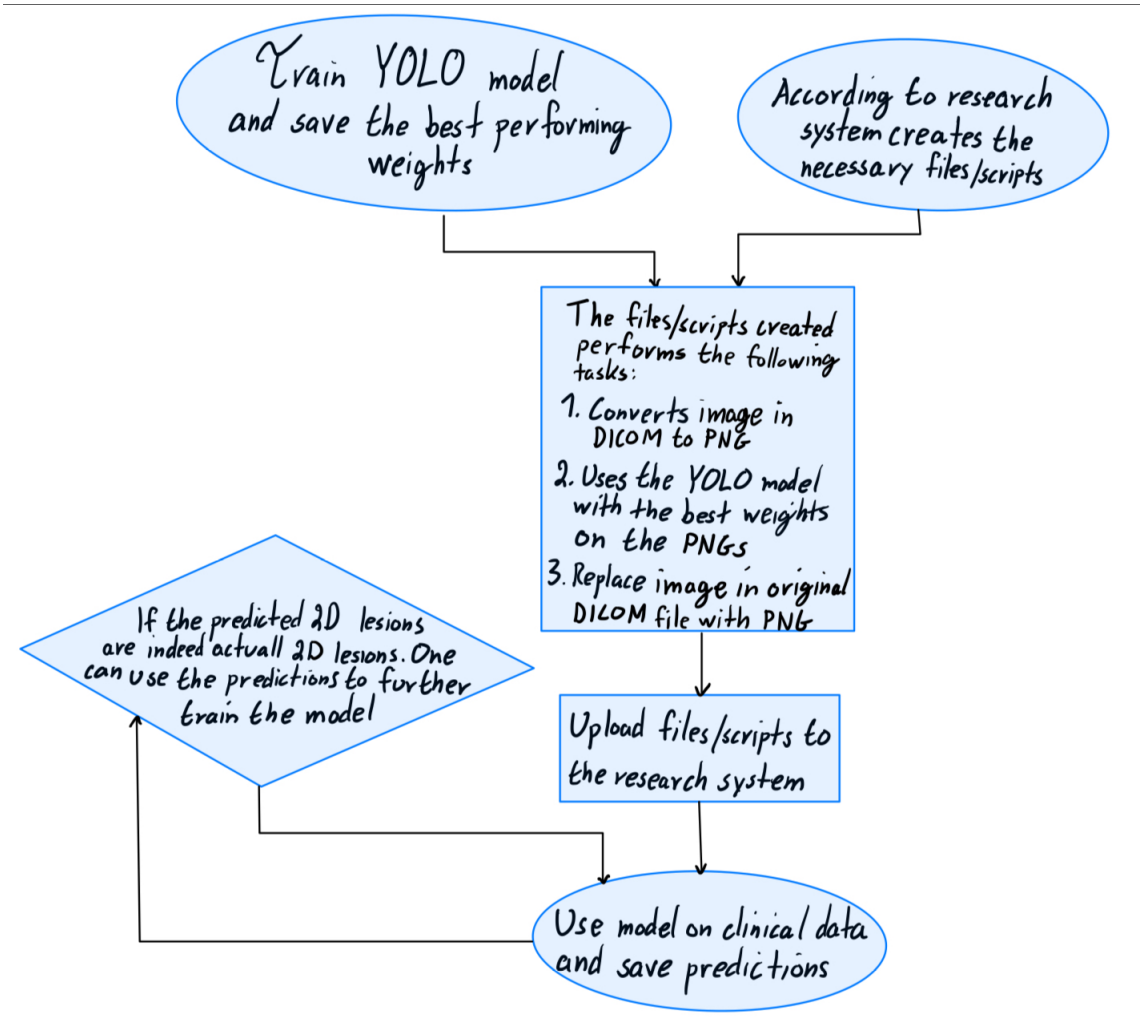


*Figure 3.10: The flowchart illustrates the process for uploading the best performing model to PACS.*

Due to patient confidentiality and other safety concerns, such as information about other research projects and models, screenshots to illustrate the complete workflow from a user perspective are not provided.

When running a prediction on local pilot test data, the model constructs a new im-

age series and presents it to the user without corrupting the original image series. After predicting on local test data, the performance was evaluated by visually inspecting the predicted lesion. However, given that the pilot test data used for the initial testing did not have ground truth masks, without healthcare professionals such as radiologists and doctors, the results on the pilot test data provided in Section 4.5 can be misinterpreted. Hence, in this thesis a lesion segmentation will be counted as successful based on the lesion localization and whether or not the model provides segmentations in the same region for at least two consecutively slices.

# Chapter 4

# Results

The online LIDC-IDRI dataset was successfully prepared, converted, and pre-processed, and one dataset containing all lesions independent of size compromising 7152 images (the whole dataset) and one size-restricted subset of the whole dataset containing 1978 images were created. The preparations, conversion, pre-processing, training, and validating were completed, and the results acquired in this thesis were through an Alienware Area-51 R4 computer with an NVIDIA GeForce GTX 1080 Ti 11GB GPU.

This chapter will present the results of the automated segmentation of CT lung lesions through the approach described in the previous sections. Firstly, the results for segmentation will be presented with the DICE score, IoU, sensitivity, and precision for the specific segmentation. The scores will be presented with three decimal points, with their corresponding standard deviation (std). Secondly, the results for object detection will be presented through the sensitivity in addition to the normalized confusion matrix. Finally, the normalized training and validation loss curves will be provided.

For both the datasets with and without size restriction, different models with different complexities (small, medium, large) were trained. To generate the following results the parameters and hyperparameters, which generated the lowest total validation loss, corresponding to each model, were used. There are multiple trained models with both the size-restricted dataset and the dataset with all lesions. The size-restricted dataset contains lesion areas relative to the whole image as explained in Section 3.2.1.

For the evaluation with the evaluation metrics scores, both the training set and validation set are displayed. If the specific metrics have approximately the same value for the training and validation set, it indicates that for the creation of the training and validation set, lesions of various shapes, areas, and locations were distributed somewhat

equally. If the specific metrics values have vast differences, this can indicate that either the training and validation set did not contain the same "types" of lesions or that the model has overfitted to the training data.

Some of the common hyperparameters utilized to train the size-restricted models are given in Table 4.1. These hyperparameters were acquired by a custom hyperparameter grid search. All models were trained with 100 epochs. The models which trained with another combination of hyperparameters, were both the large models and the small trained model for every lesion size. For the models trained without size-restriction, the only difference, from Table 4.1, is that the scale hyperparameter is set to 0.9. Additionally, the large (most complex) models have batch sizes set to one.

| Hyperparameter | Value | Hyperparameter | Value |
|---|---|---|---|
| batch size | 8 | optimizer | SGD |
| scale | 0.6 | dropout | False |
| weight_decay | 0.001 | momentum | 0.937 |
| lr0 | 0.0001 | lrf | 0.0001 |

*Table 4.1: Some of the hyperparameters utilized to train the majority of the models*

## 4.1 YOLOv8 small model

This section provides the results acquired by training the small YOLO segmentation model (yolov8s-seg), on the datasets with and without size restrictions. The small segmentation model is the least complex model utilized in this thesis. The results in this section are presented by first providing the results for the model trained without size restrictions (all lesions) and then subsequently providing the results for the trained with size restrictions.

### 4.1.1 Results from all lesions

The model's training and validation loss curves during training are displayed in Figure 4.1. From the curves provided, the losses, on average, follow a similar steady decline and seem to converge. The results provided in this section are acquired from the parameters (weights and biases) that generated the lowest total validation loss according to Equation 2.46.

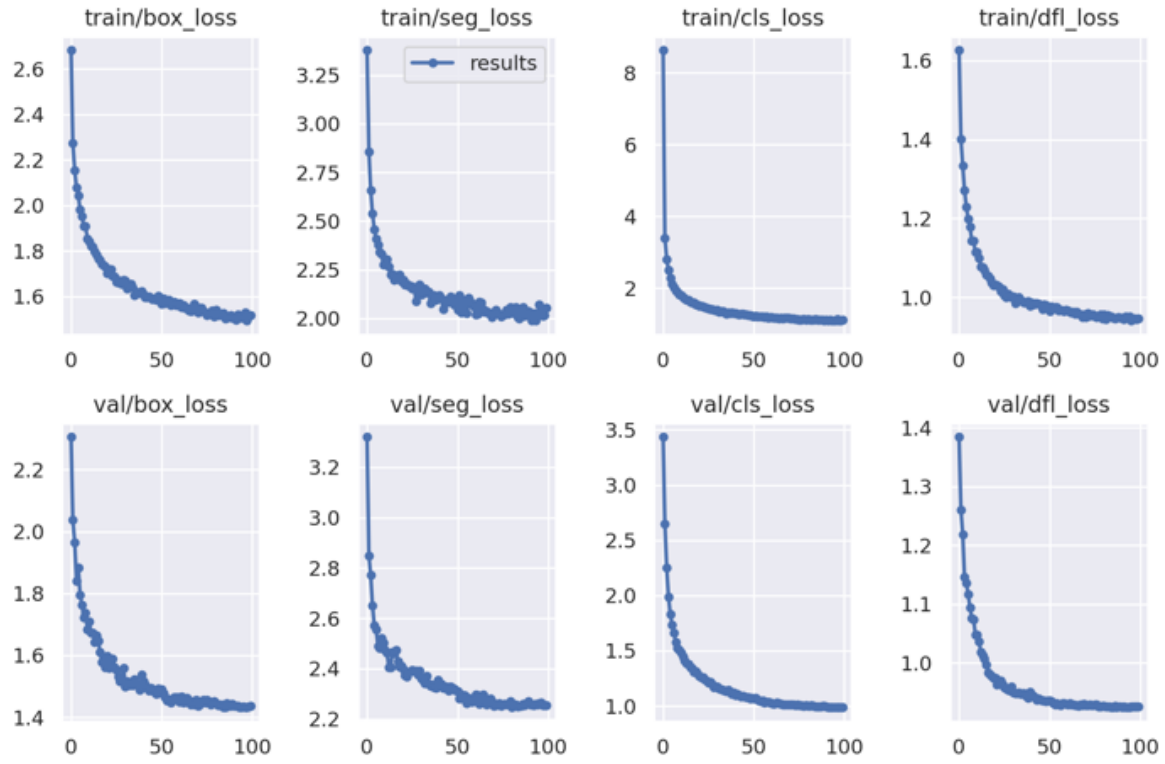


*Figure 4.1: The training and validation loss curves for the small model trained on the dataset without size restrictions. For each curve, the x-axis represents the number of epochs, and the y-axis represents the corresponding loss.*

Table 4.2 shows the performance metrics results obtained from the automated segmentation with yolov8s-seg for all lesions. The results were obtained with the modifications mentioned in Section 3.2.2. On the validation set, the model achieved a DICE score of $0.682 \pm 0.228$, an IoU of $0.556 \pm 0.230$, a precision of $0.829 \pm 0.175$, and a sensitivity of $0.653 \pm 0.270$, where the metric scores are approximately equal for the training set, indicating that the trained model did not overfit, which is also illustrated by the loss curves (Figure 4.1). The performance metrics for this trained model are compared to the other models trained in this thesis in Table 4.8.

| | Training set | Validation set |
|---|---|---|
| Evaluation metric: | score $\pm$ std: | score $\pm$ std |
| DICE | $0.681 \pm 0.234$ | $0.682 \pm 0.228$ |
| IoU | $0.557 \pm 0.235$ | $0.556 \pm 0.230$ |
| Precision | $0.822 \pm 0.169$ | $0.829 \pm 0.175$ |
| Sensitivity | $0.650 \pm 0.273$ | $0.653 \pm 0.270$ |

*Table 4.2: The segmentation performance measures with the related scores with standard deviation for the small model with all lesions.*

Figure 4.2 provides examples of predictions made by the model (bottom row) compared

to their corresponding ground truths (top row). In these examples, the model was able to correctly detect and segment three out of four lesions, with confidences between 0.4 and 0.9 (excluding the prediction where the confidence is not shown), demonstrating an ability to detect and segment lesions correctly. However, in the second prediction, the model predicts two extra cases that are not given as ground truths, indicating either that the model provides extra cases of FPs or that the ground truths provided by the radiologists from LIDC-IDRI are not complete, which is further discussed in Chapter 5.

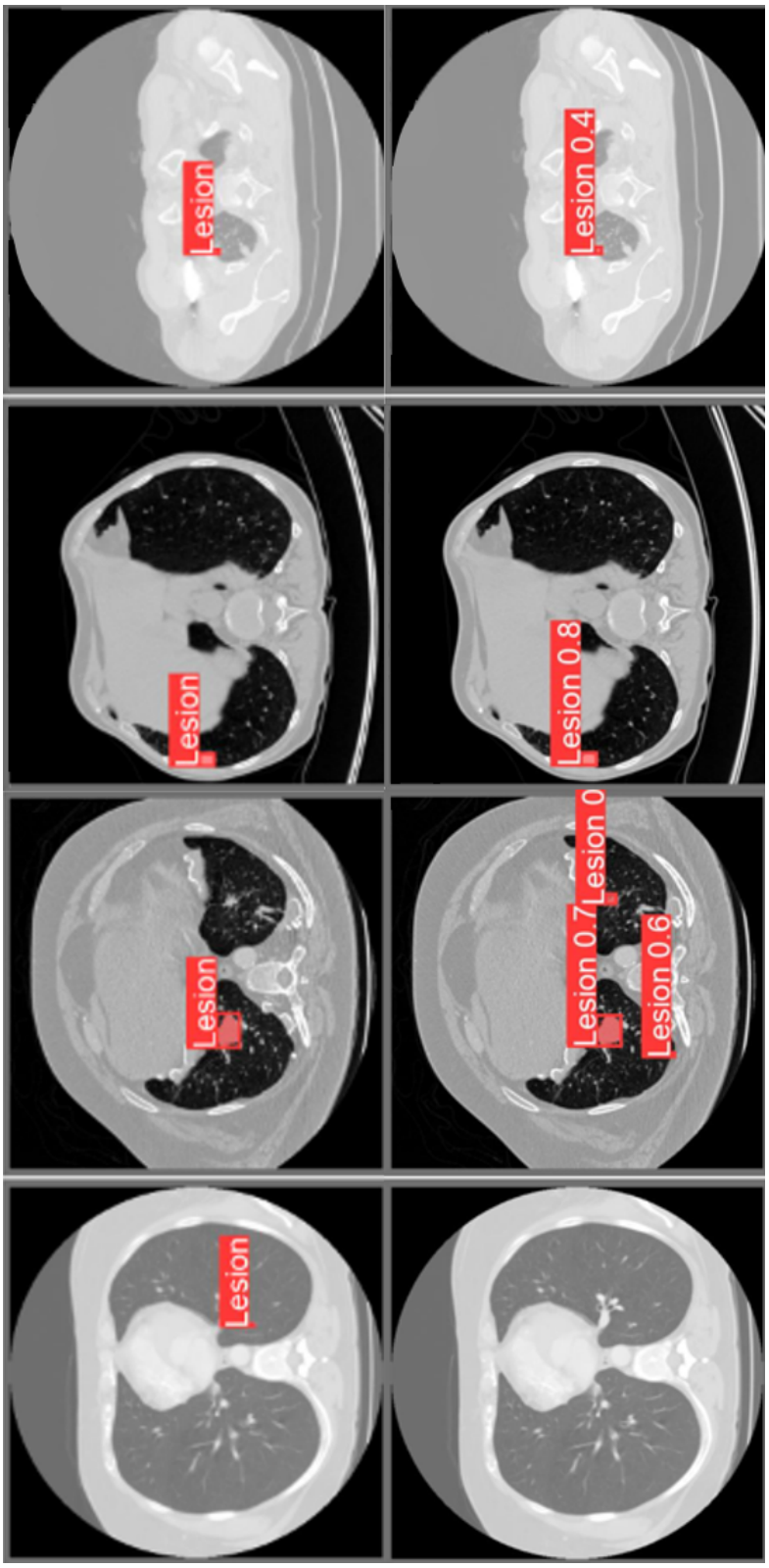


*Figure 4.2: Some model predictions (bottom row) against their corresponding ground truths (top row) by the small model trained without size restriction.*

Figure 4.3 illustrates the object detection performance with a normalized confusion matrix on the validation set. The column-normalized matrix provides a sensitivity score of 0.64, which indicates that over half of the CT lung lesions got detected. Compared to the models provided by Liu et al. (0.873) and Liying et al. (0.962) this trained model

underperformed, which is further discussed in Chapter 5.

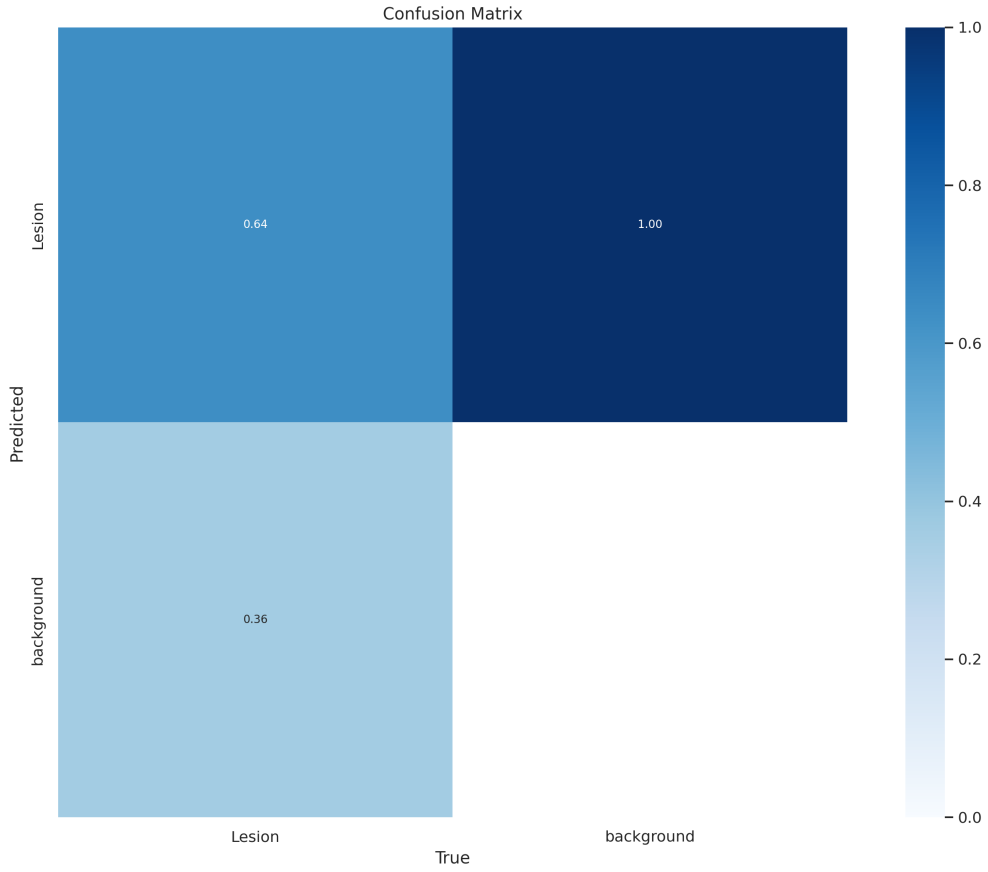


*Figure 4.3: The normalized confusion matrix displays how well the object detection performed for the small model with all lesions*

### 4.1.2 Size-restricted

The model's training and validation loss curves during training are displayed in Figure 4.4. From the curves provided, the losses follow a similar steady decline on average. The results provided in this section, are taken from the losses which generated the lowest total validation loss according to Equation 2.46.

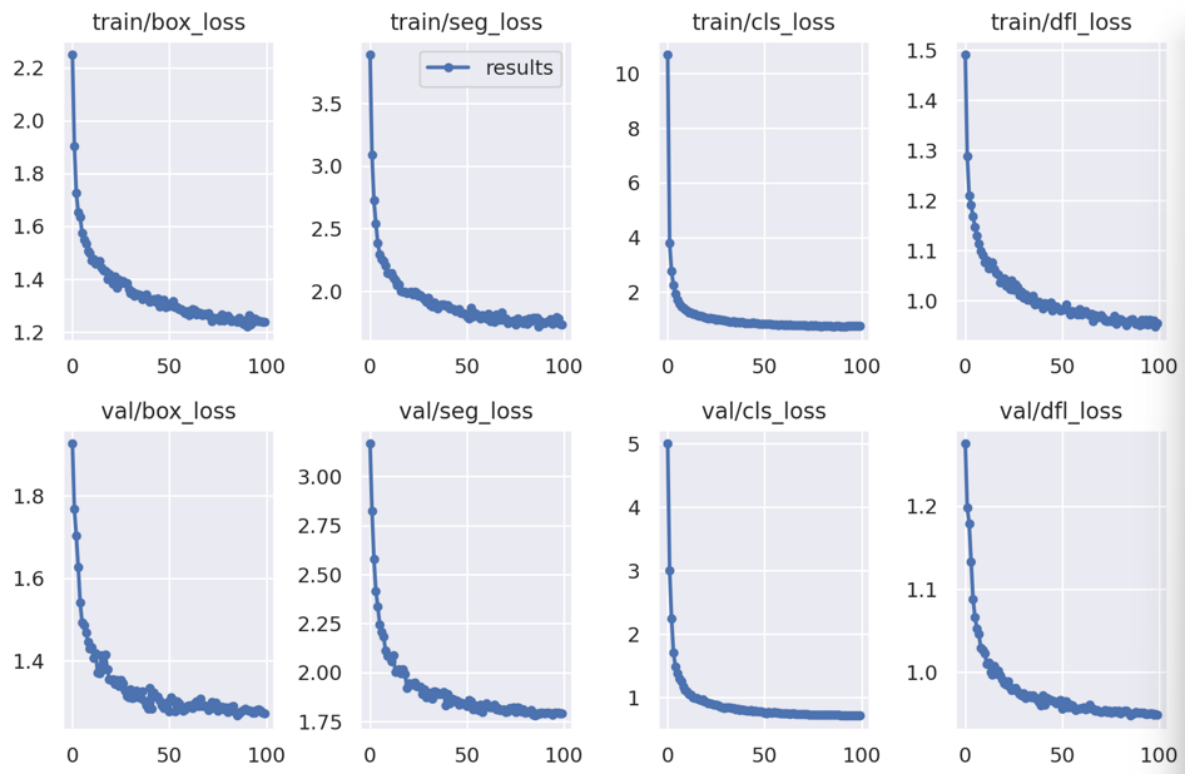


*Figure 4.4: The training and validation loss curves for the small model trained on the size-restricted dataset. For each curve, the x-axis represents the number of epochs, and the y-axis represents the corresponding loss.*

Table 4.3 provides the evaluation metric scores obtained from the automated segmentation task. On the validation set, the trained model managed to achieve a DICE score of $0.836 \pm 0.117$, an IoU of $0.731 \pm 0.138$, a precision of $0.864 \pm 0.109$, and a sensitivity of $0.836 \pm 0.153$, where the metric scores are approximately the same for the training set, indicating that the trained model did not overfit, which is also illustrated by the loss curves (Figure 4.4). The performance metrics for this trained model are compared to the other models trained in this thesis in Table 4.8.

| | Training set | Validation set |
|---|---|---|
| Performance metric | score $\pm$ std | score $\pm$ std |
| DICE | $0.844 \pm 0.108$ | $0.836 \pm 0.117$ |
| IoU | $0.742 \pm 0.130$ | $0.731 \pm 0.138$ |
| Precision | $0.858 \pm 0.112$ | $0.864 \pm 0.109$ |
| Sensitivity | $0.852 \pm 0.133$ | $0.836 \pm 0.153$ |

*Table 4.3: The segmentation performance measures with the related scores with standard deviation for the small model. For 2D lesions covering* 0.05% *to* 0.4% *of the whole image.*

Figure 4.5 provides examples of predictions made by the model (bottom row) and with the corresponding ground truths (top row). In these examples, the model was able to

predict and segment four out of four lesions, with confidences ranging between 0.6 and 0.9, demonstrating a great ability to detect and segment CT lung lesions.

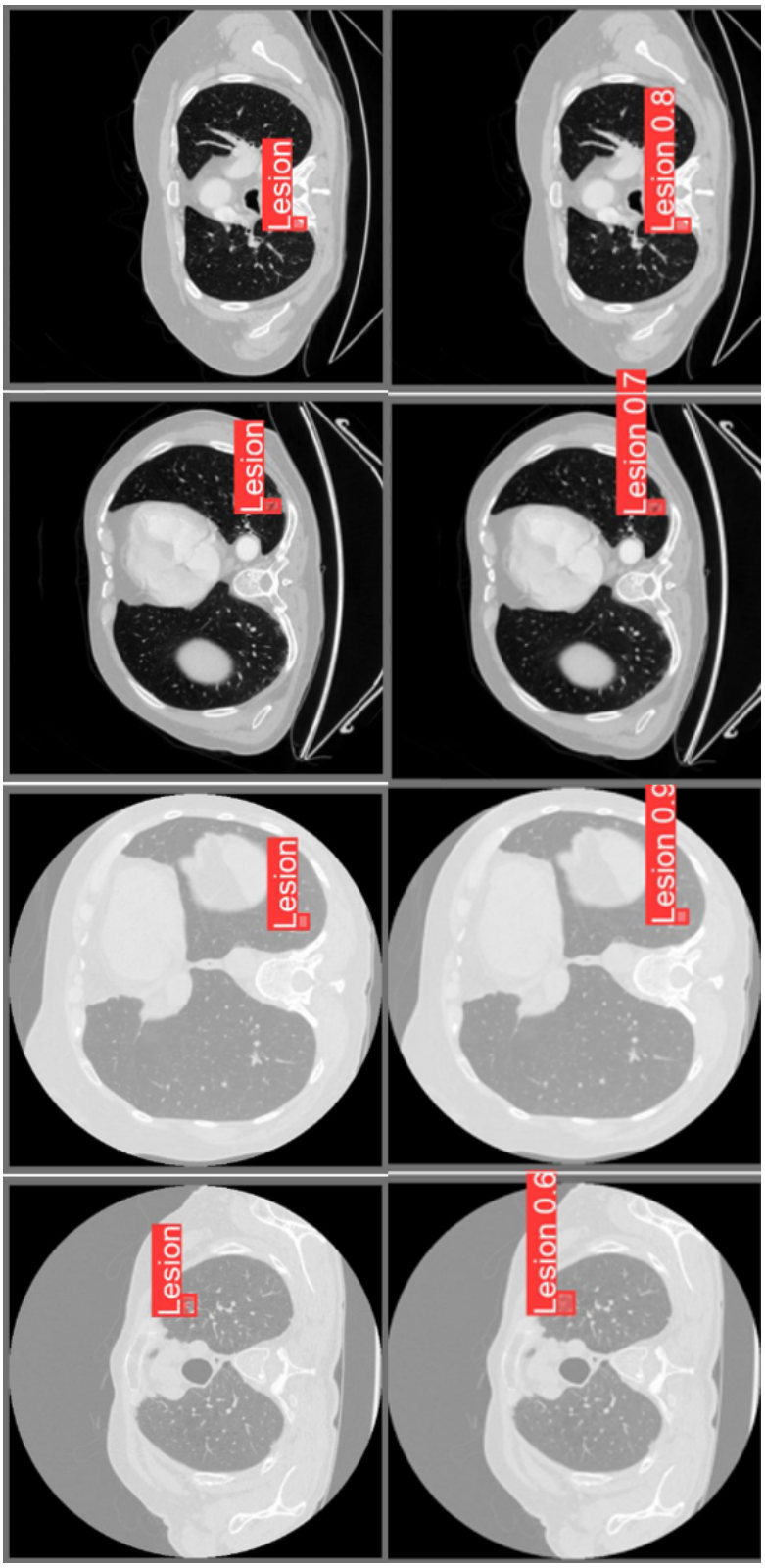


*Figure 4.5: Some model predictions (bottom row) against their corresponding ground truths (top row) by the small model trained with size restriction. 2D lesions covering 0.05% to 0.4% of the whole image.*

Figure 4.3 illustrates the object detection performance with a normalized confusion matrix on the validation set. The column-normalized matrix provides a sensitivity score of 0.91, which, as hypothesized, outperformed the model trained without size restriction.

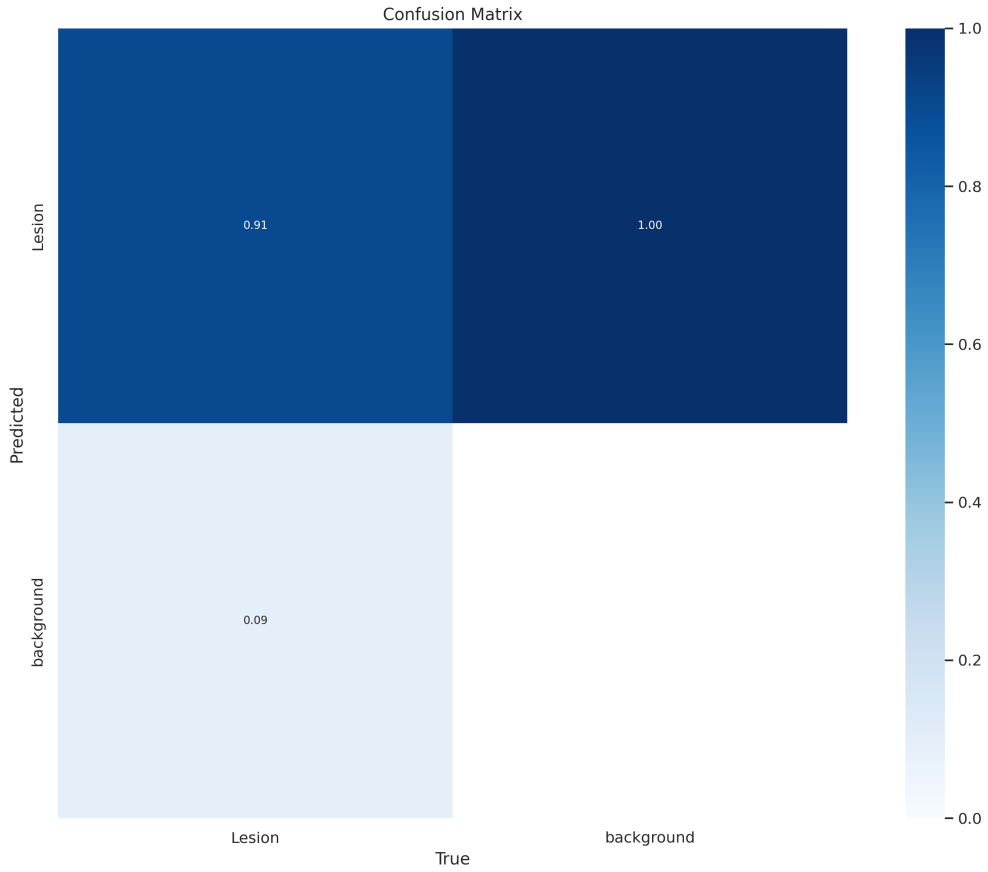


*Figure 4.6: The normalized confusion matrix for lesions covering 0.05% to 0.4% of the whole image.*

## 4.2 YOLOv8 medium model

This section provides the results acquired after training models with the medium YOLO segmentation model (yolov8m-seg), which is the second least complex model utilized in this thesis. The results are presented by first providing the results acquired by training the model on the dataset without size restriction (all lesions) and then subsequently providing the corresponding results to the model trained on the dataset with size restriction.

### 4.2.1 Results from all lesions

The training and validation loss curves for the model during training are displayed in Figure 4.7. From the curves provided, the losses follow a similar steady decline on average. The results provided in this section, are taken from the losses which generated the lowest total validation loss according to Equation 2.46.

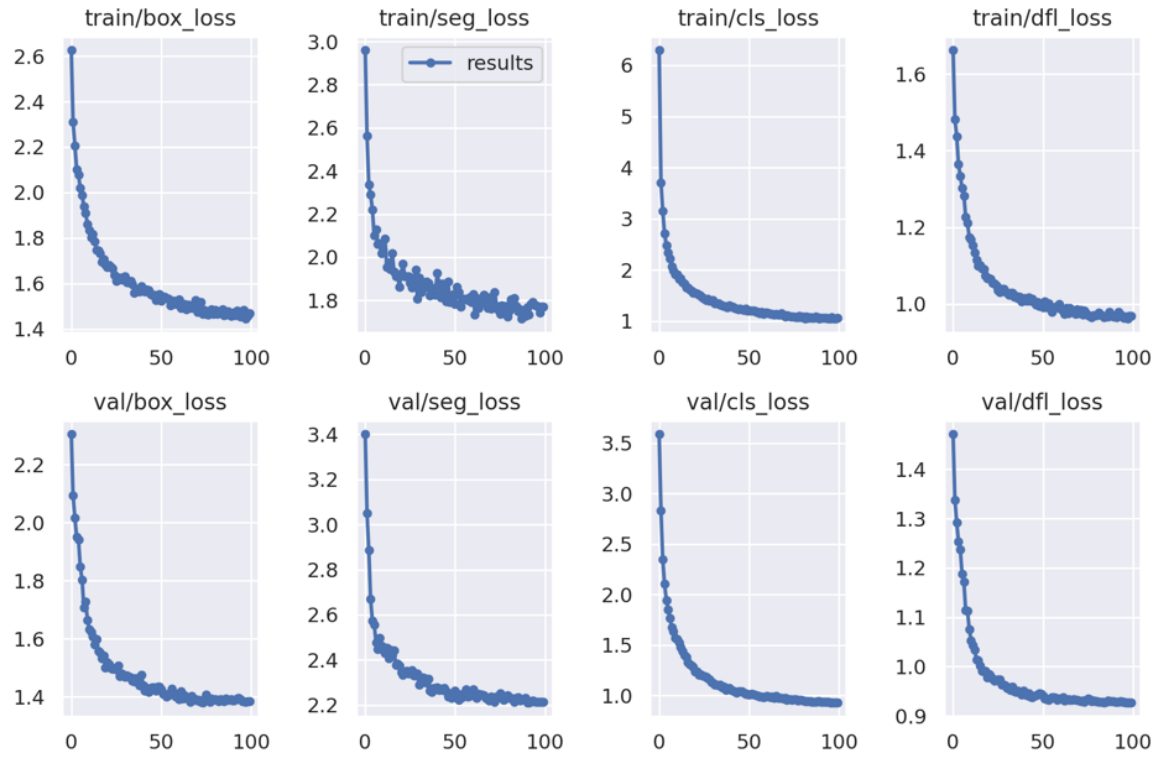


*Figure 4.7: The training and validation loss curves for the medium model trained on lesions of every size. For each curve, the x-axis represents the number of epochs, and the y-axis represents the corresponding loss.*

Table 4.4 provides and compares the various segmentation evaluation metrics scores between the training and validation set. On the validation set, the model achieves a DICE score of 0.685 $\pm$ 0.237, an IoU of 0.563 $\pm$ 0.238, a precision score of 0.823 $\pm$ 0.188, and a sensitivity score of 0.665 $\pm$ 0.272, where the scores are approximately equal to the training set, indicating that the trained model did not overfit, which is also illustrated by the loss curves (Figure 4.7). The performance metrics for this trained model are compared to the other models trained in this thesis in Table 4.8.

| | Training set | Validation set |
|---|---|---|
| Performance metric | score $\pm$ std | score $\pm$ std |
| DICE | 0.684 $\pm$ 0.233 | 0.685 $\pm$ 0.237 |
| IoU | 0.561 $\pm$ 0.235 | 0.563 $\pm$ 0.238 |
| Precision | 0.823 $\pm$ 0.167 | 0.823 $\pm$ 0.188 |
| Sensitivity | 0.655 $\pm$ 0.271 | 0.665 $\pm$ 0.272 |

*Table 4.4: The segmentation performance measures with the related scores with standard deviation for the medium model.*

Figure 4.8 provides examples of predictions made by the model compared to ground truths respectively. In these examples, the model was able to predict and segment two

out of four lesions correctly, with confidences ranging from 0.3 to 0.8, compared to the predictions made by the small model in Figure 4.2 the model seems to underperform. However, these are just some of the many predictions made by the model.

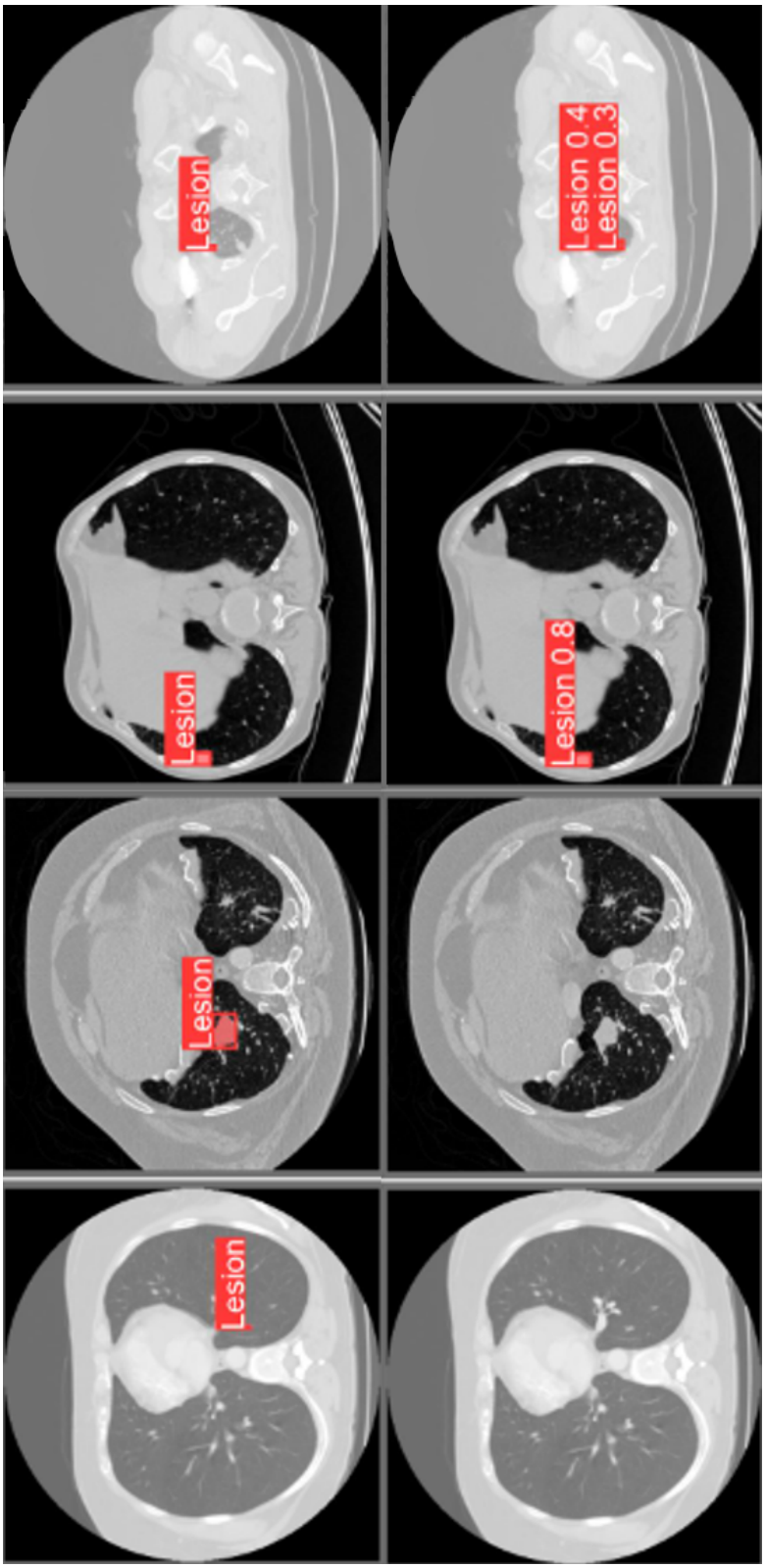


*Figure 4.8: Some model predictions (bottom row) against their corresponding ground truths (top row) by the medium model trained without size restriction.*

Figure 4.9 illustrates the object detection performance with a normalized confusion matrix, on the validation set. The column-normalized matrix indicates a sensitivity score of 0.62. In comparison to the small trained model (all lesions), the medium model seems to be slightly inferior to the small model, which might explain the lack of

lesion detection (and segmentation) in Figure 4.8.

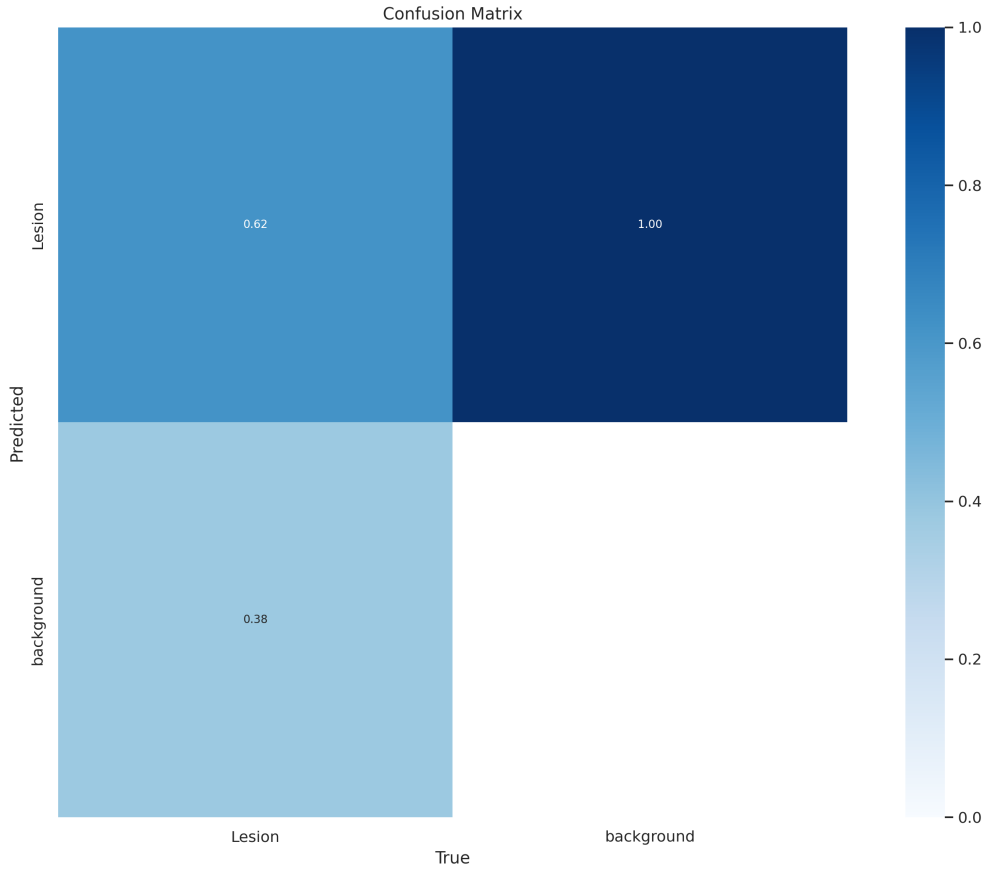


*Figure 4.9: The normalized confusion matrix displays how well the object detection performed for the medium model with all lesions*

### 4.2.2 Size-restricted

The model's training and validation loss curves during training are displayed in Figure 4.10. From the curves provided, the losses follow a similar steady decline on average. The results provided in this section, are taken from the losses which generated the lowest total validation loss according to Equation 2.46.

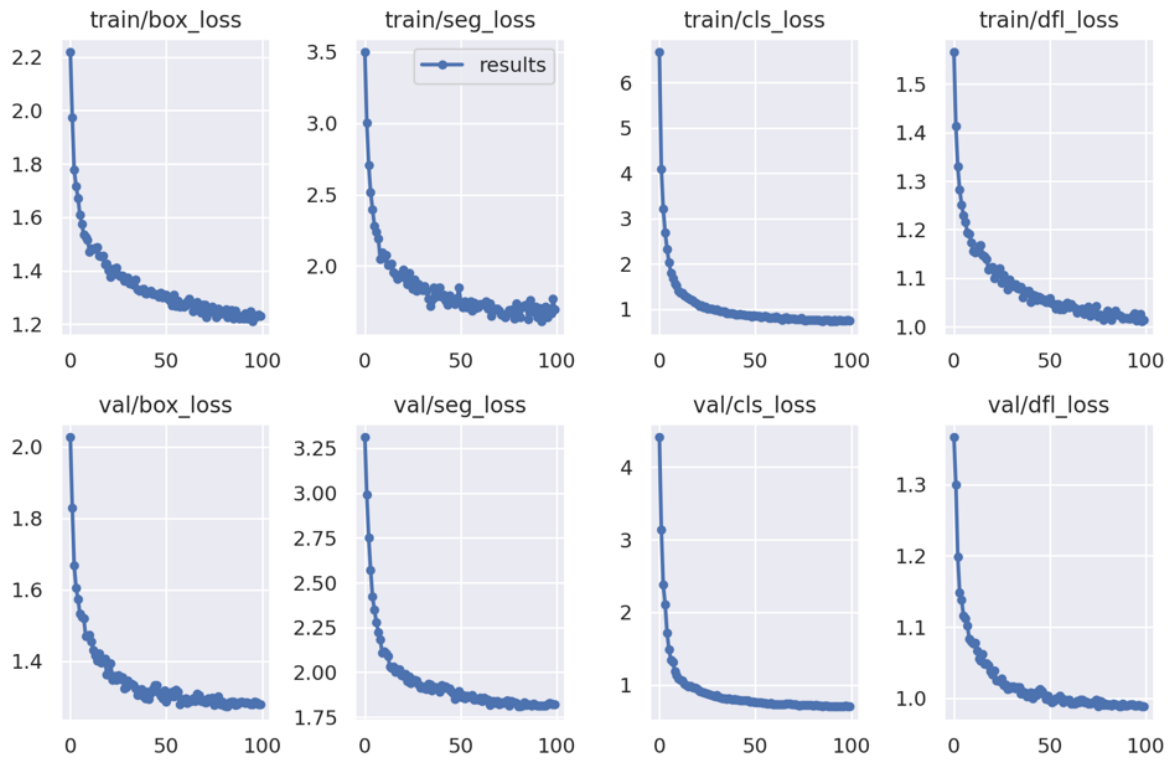


*Figure 4.10: The training and validation loss curves for the medium model trained on the size-restricted dataset. For each curve, the x-axis represents the number of epochs, and the y-axis represents the corresponding loss.*

Table 4.3 compares the different segmentation evaluation metric scores, between the training and validation set. For the validation set the trained model managed to achieve a DICE score of $0.826 \pm 0.132$, an IoU of $0.720 \pm 0.153$, a precision score of $0.850 \pm 0.120$ and a sensitivity score of $0.835 \pm 0.165$, for the training set these metric scores are approximately equal, indicating that the trained model did not overfit, which is also illustrated by the loss curves (Figure 4.10). The performance metrics for this trained model are compared to the other models trained in this thesis in Table 4.8.

| | Training set | Validation set |
|---|---|---|
| Performance metric | score $\pm$ std | score $\pm$ std |
| DICE | $0.844 \pm 0.112$ | $0.826 \pm 0.132$ |
| IoU | $0.743 \pm 0.134$ | $0.720 \pm 0.153$ |
| Precision | $0.858 \pm 0.113$ | $0.850 \pm 0.120$ |
| Sensitivity | $0.853 \pm 0.138$ | $0.835 \pm 0.165$ |

*Table 4.5: The segmentation performance measures with the related scores with standard deviation for the medium model. For 2D lesions covering* 0.05% *to* 0.4% *of the whole image.*

Figure 4.11 provide examples of predictions made by the model (bottom row) with the corresponding ground truths (top row), respectively. In these examples, the model was

able to predict and segment four out of four lesions, with confidences ranging between 0.5 and 0.9, indicating a great ability for lesion segmentation and detection. However, in the first prediction, the model detected and segmented two lesions with very close (overlapping) proximity, which practically can be ignored.

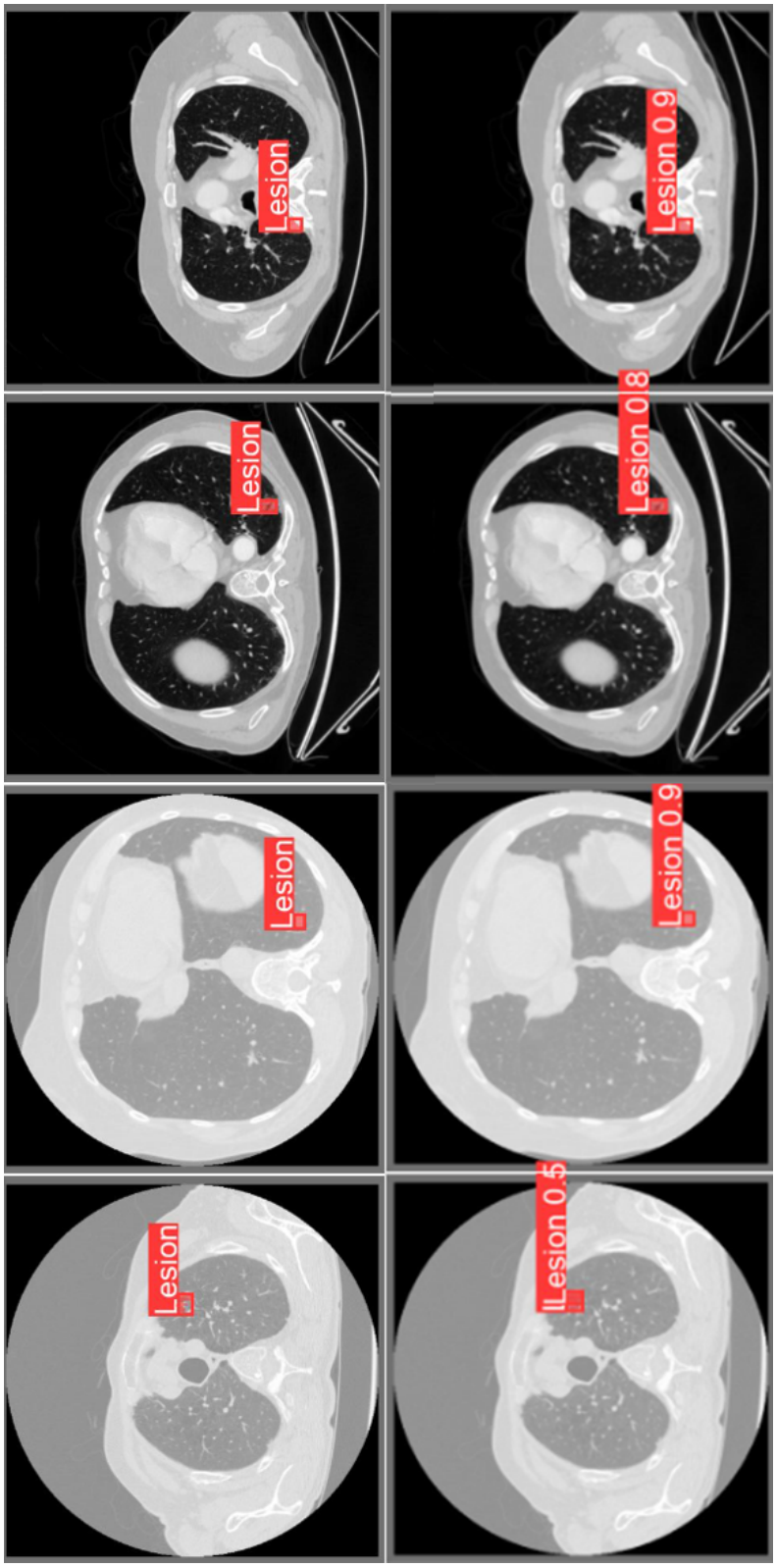


*Figure 4.11: Some model predictions (bottom row) against their corresponding ground truths (top row) by the medium model trained with size restriction. 2D lesions covering 0.05% to 0.4% of the whole image.*

Figure 4.12 illustrates the object detection performance with a normalized confusion matrix on the validation set. The column-normalized matrix indicates a sensitivity score of 0.91, which, as hypothesized, outperformed the model trained without size restriction. The model also performed equally well as the small size-restricted model, and the results are further discussed in Chapter 5.

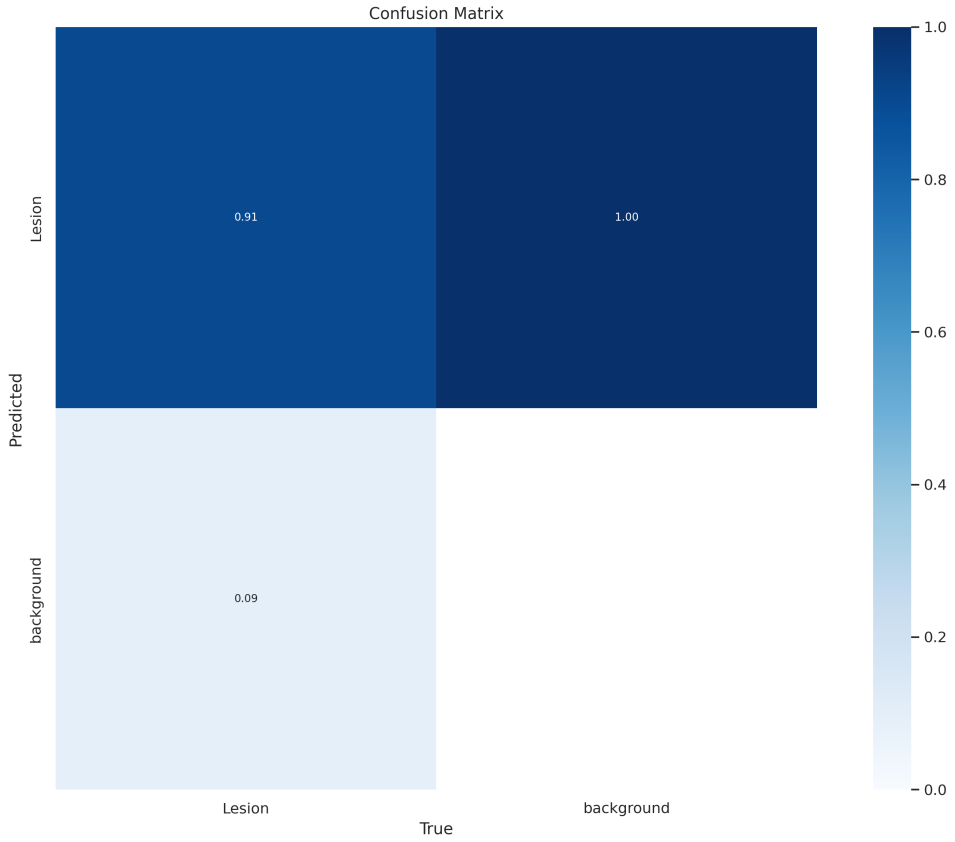


*Figure 4.12: The normalized confusion matrix for 2D lesions covering 0.05% to 0.4% of the whole image.*

## 4.3 YOLOv8 large model

This section provides the results acquired after training CT lung lesion segmentation model with the large YOLO segmentation model (yolov8l-seg), which is the most complex model utilized in this thesis. These models were both trained with every hyperparameter presented in Table 4.1, apart from the number of batch sizes being set to one.

The results are presented by first providing the results acquired after training the model with every lesion size and subsequently by providing the results acquired after training the size-restricted lesions. However, after the models were trained, the automated process for generating examples of predictions versus ground truths and generating the

normalized confusion matrix failed.

### 4.3.1 Results for all lesions

The model's training and validation loss curves during training are displayed in Figure 4.13. From the curves provided, the losses follow a similar steady decline on average. The results provided in this section are taken from the losses that generated the lowest total validation loss according to Equation 2.46.

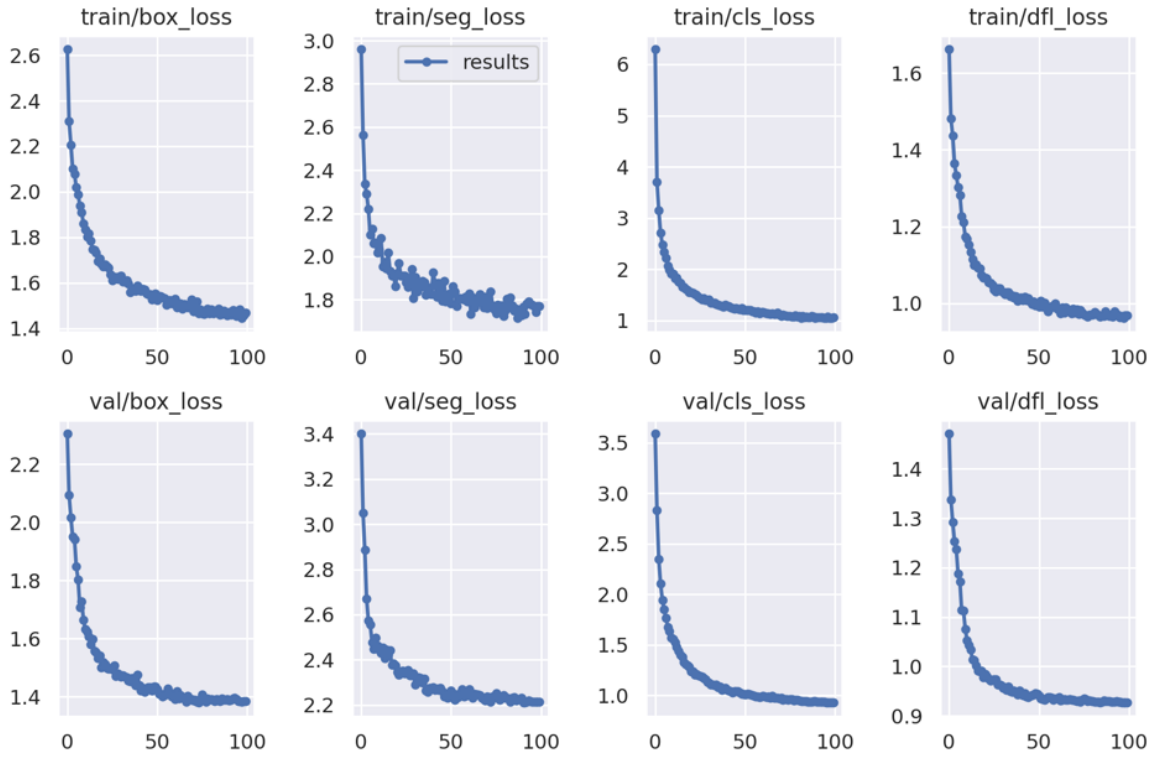


*Figure 4.13: The training and validation loss curves for the large model trained on lesions of every size. For each curve, the x-axis represents the number of epochs, and the y-axis represents the corresponding loss.*

Table 4.6 shows the results obtained from the automated segmentation with yolov8m-seg for all lesions. The results were obtained after implementing the modifications mentioned in Section 3.2.2. On the validation set, the model managed to achieve a DICE score of 0.664 $\pm$ 0.250, an IoU of 0.542 $\pm$ 0.246, a precision score of 0.816 $\pm$ 0.192, and a sensitivity score of 0.636 $\pm$ 0.282, where the scores are approximately equal to the training set, indicating that the trained model did not overfit, which is also illustrated by the loss curves (Figure 4.13). The performance metrics for this trained model are compared to the other models trained in this thesis in Table 4.8.

| | Training set | Validation set |
|---|---|---|
| Performance metric | score ± std | score ± std |
| DICE | 0.674 ± 0.236 | 0.664 ± 0.250 |
| IoU | 0.550 ± 0.237 | 0.542 ± 0.246 |
| Precision | 0.826 ± 0.171 | 0.816 ± 0.192 |
| Sensitivity | 0.641 ± 0.273 | 0.636 ± 0.282 |

*Table 4.6: The segmentation performance measures with the related scores with standard deviation for the large model.*

### 4.3.2 Size-restricted

The training and validation loss curves for the model during training are displayed in Figure 4.14. From the curves provided, the losses follow a similar steady decline on average. The results provided in this section, are taken from the losses which generated the lowest total validation loss according to Equation 2.46.

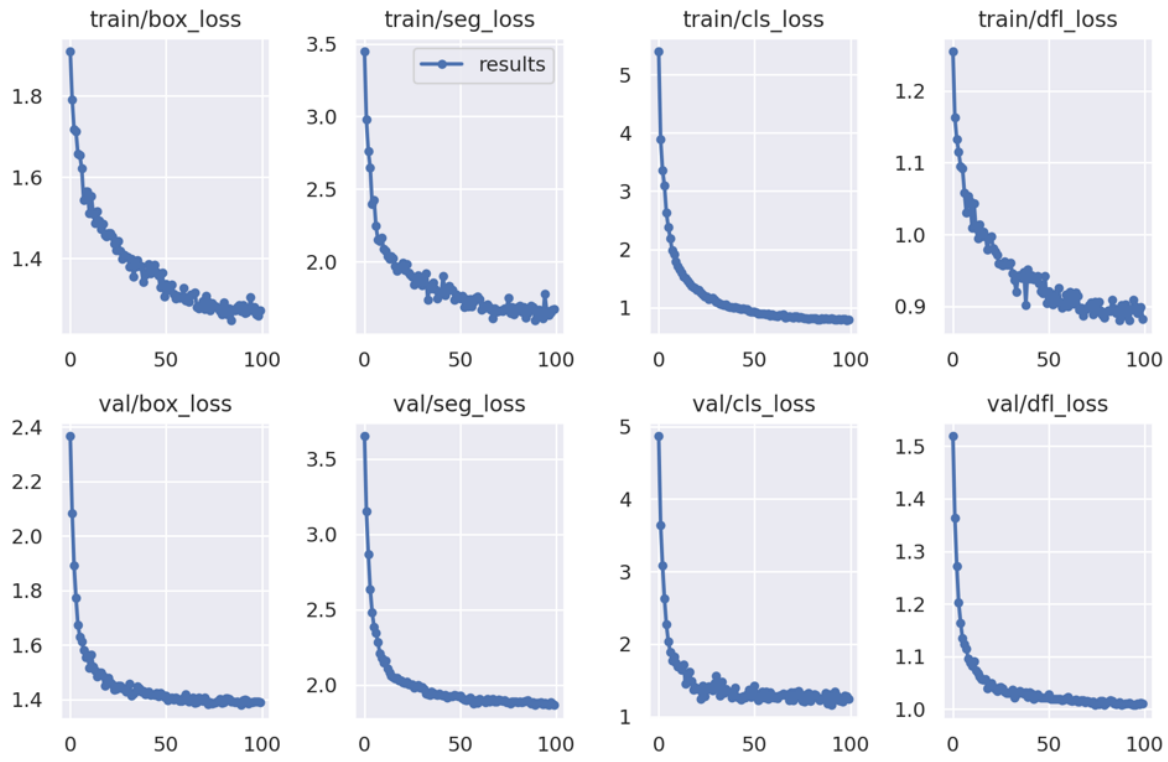


*Figure 4.14: The training and validation loss curves for the large model trained on the size-restricted dataset. For each curve, the x-axis represents the number of epochs, and the y-axis represents the corresponding loss.*

Table 4.7 shows the results obtained from the automated segmentation with yolov8l-seg for the size-restricted dataset. The results were obtained after implementing the modifications mentioned in Section 3.2.2. On the validation set, the model achieves a DICE

score of 0.840 ± 0.127, an IoU of 0.740 ± 0.142, a precision score of 0.877 ± 0.103, and a sensitivity score of 0.833 ± 0.158, where the scores are approximately equal to the training set, indicating that the model did not overfit, which is also illustrated by the loss curves (Figure 4.14) The performance metrics for this trained model are compared to the other models trained in this thesis in Table 4.8.

| | Training set | Validation set |
|---|---|---|
| Performance metric | score ± std | score ± std |
| DICE | 0.845 ± 0.114 | 0.840 ± 0.127 |
| IoU | 0.744 ± 0.135 | 0.740 ± 0.142 |
| Precision | 0.876 ± 0.110 | 0.877 ± 0.103 |
| Sensitivity | 0.836 ± 0.142 | 0.833 ± 0.158 |

*Table 4.7: The segmentation performance measures with the related scores with standard deviation for the large model. For 2D lesions covering* 0.05% *to* 0.4% *of the whole image.*

## 4.4 Comparison across models

Each model, influenced by its unique hyperparameters, parameters, and size restriction, demonstrated various performances accordingly. A comparison between the models' segmentation performances together with their corresponding size restrictions is depicted in Table 4.8. The model with the highest DICE score, IoU, and Sensitivity score was the large size-restricted trained model with the scores 0.840 ± 0.127, 0.740 ± 0.142, 0.877 ± 0.103 respectively. The model with the highest sensitivity score was the small size-restricted model with a score of 0.836 ± 0.153. Given the small deviance in sensitivity score between the large and small size-restricted model, the large model should be selected. However, since the deviances in the performances, in general, are small between the small and large models, uploading the small (least complex) model to the regional research PACS is sufficient for pilot testing.

| Size restriction: | DICE: | IoU: | Precision: | Sensitivity: |
|---|---|---|---|---|
| S: All lesions | 0.682 ± 0.228 | 0.556 ± 0.230 | 0.829 ± 0.175 | 0.653 ± 0.270 |
| S: 0.05% to 0.4% | 0.836 ± 0.117 | 0.731 ± 0.138 | 0.864 ± 0.109 | 0.836 ± 0.153 |
| M: All lesions | 0.685 ± 0.237 | 0.563 ± 0.238 | 0.823 ± 0.188 | 0.665 ± 0.272 |
| M: 0.05% to 0.4% | 0.826 ± 0.132 | 0.720 ± 0.153 | 0.850 ± 0.120 | 0.835 ± 0.165 |
| L: All lesions | 0.664 ± 0.250 | 0.542 ± 0.246 | 0.816 ± 0.192 | 0.636 ± 0.282 |
| L: 0.05% to 0.4% | 0.840 ± 0.127 | 0.740 ± 0.142 | 0.877 ± 0.103 | 0.833 ± 0.158 |

*Table 4.8: Every validation score from every model trained with and without size restrictions (All lesions, 0.05% to 0.4%), where yolov8s-seg = S, yolov8m-seg = M and yolov8l-seg = L.*

Since the large size-restricted model was the best performing model, and the small size-restricted model was chosen to be uploaded to the regional research PACS, the segmentation performances for both models on the test set are provided in Table 4.9. Except for the DICE score, the models' performance are as expected when compared with Table 4.8. There is a slight difference between the performance in the validation set versus the test set, where the test set underperforms, which might indicate that some images in the test set are dissimilar to those in the other sets or that the models are overfitted to both the training and validation set. However, given the deviations occur on the second decimals, the models are not likely to be significantly overfitted.

| | Large | Small |
|---|---|---|
| Performance metric | score $\pm$ std | score $\pm$ std |
| DICE | $0.809 \pm 0.166$ | $0.811 \pm 0.144$ |
| IoU | $0.703 \pm 0.179$ | $0.702 \pm 0.166$ |
| Precision | $0.845 \pm 0.168$ | $0.824 \pm 0.155$ |
| Sensitivity | $0.803 \pm 0.184$ | $0.838 \pm 0.162$ |

*Table 4.9: The segmentation metric scores on the test set for both the large and small size-restricted trained models.*

## 4.5 Clinical integration

The trained model chosen to be implemented in PACS was the small size-restricted model due to its simplicity, performance for the given lesion size, and to demonstrate proof of concept for a DL algorithm segmenting CT lung lesion in the clinical workflow. After uploading the trained model to PACS, with its adaptions, the model was used to run predictions on local test data. The test data did not have already existing segmentations. Hence, the performance metrics of the model's performance on these clinical images are not provided. However, the performance of the model in PACS is expected to be similar to the performance on the LIDC-IDRI dataset, and how the model performs on local data is however, beyond the scope of the current thesis.

Figure 4.15 and Figure 4.16 display an image series of predictions on one test patient from the test data. Both image series provide predictions in consecutive order from the top left image to the bottom right. These image series are both from the same test patient where the bottom right image in Figure 4.15 and the top left image in Figure 4.16 are consecutive slices. Due to an error, fine information about the structures outside the lungs was lost in the generation of these predictions, the errors have since been corrected (Figure 4.17 - 4.20). The model detects lesions in five distinct areas, whereas all

five lesions are partly or fully segmented. The model also provides corresponding lesion confidence scores where the bottom left predicted lesion has the highest confidence score with an average of 0.71, indicating a high probability for the region detected and segmented being a lesion.

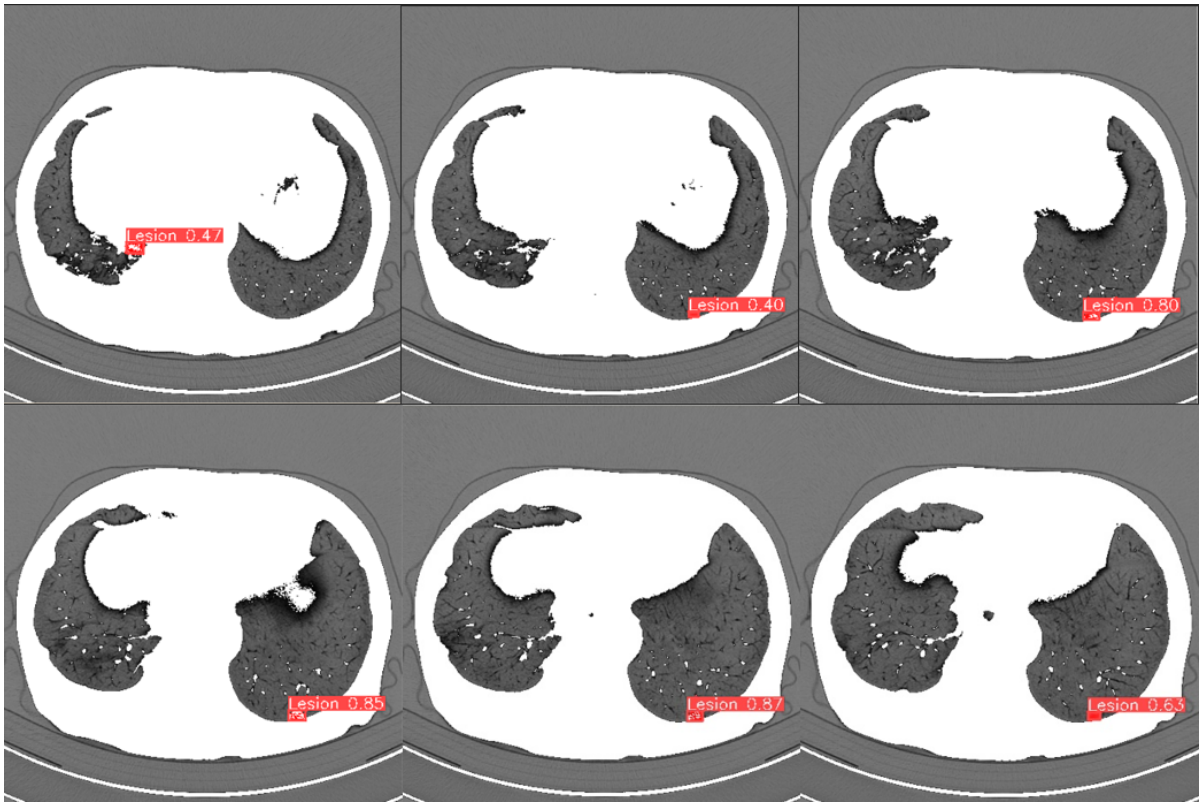


*Figure 4.15: Image series of predictions on one patient from the test data, where the images are slices in consecutive order from the top left to bottom right. The image series continues in Figure 4.16.*

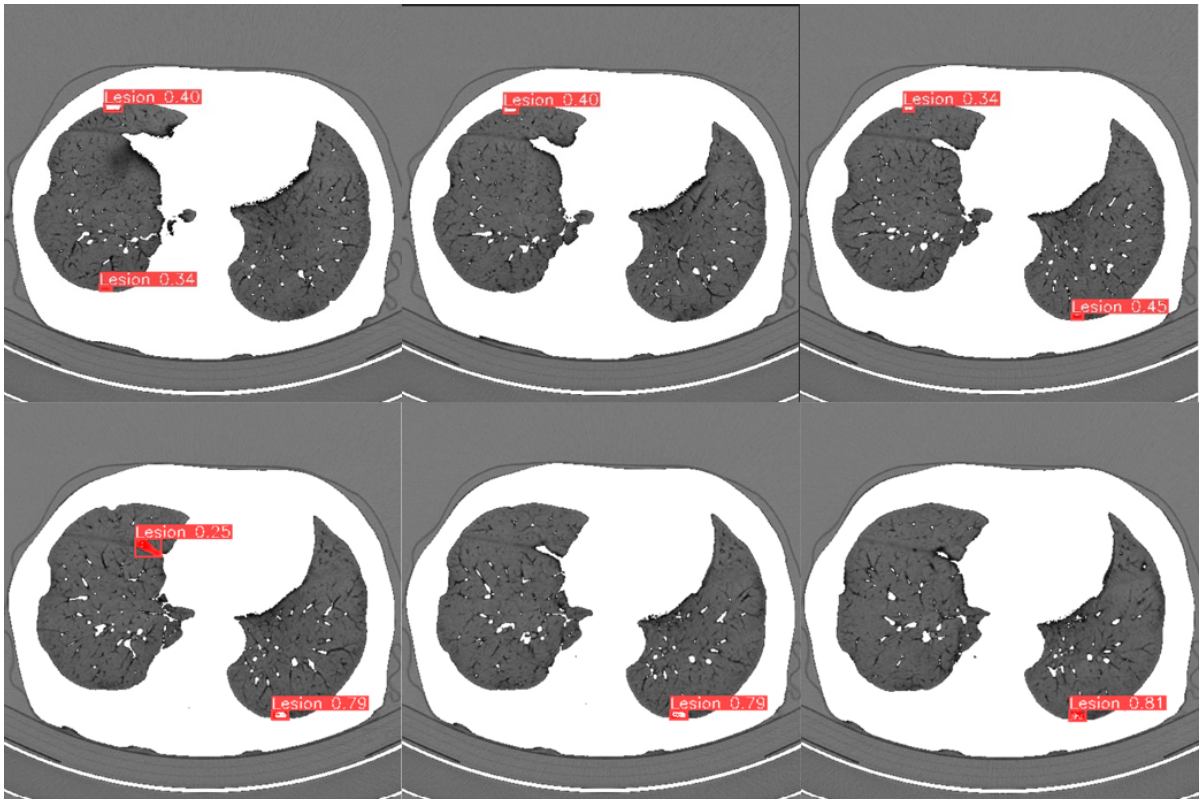


*Figure 4.16: Image series of predictions of one patient from the test data, where the images are slices in consecutive order from left to right. The image series is a continuance of the image series displayed in Figure 4.15.*

Figure 4.17 - 4.20 displays predictions on test data with the predicted images to the left and the original images to the right. In Figure 4.17 one of the segmentations provided is clearly faulty, where the rightmost segmentation is outside the patient. Figure 4.18 provides a segmentation with a possible lesion, with a confidence score of 0.31. The image series in Figure 4.19 and Figure 4.20 provides segmentations of a possible lesion, with the images in consecutive order from the top to the bottom, with an average confidence score of 0.33. These results will be further discussed in Section 5.7.

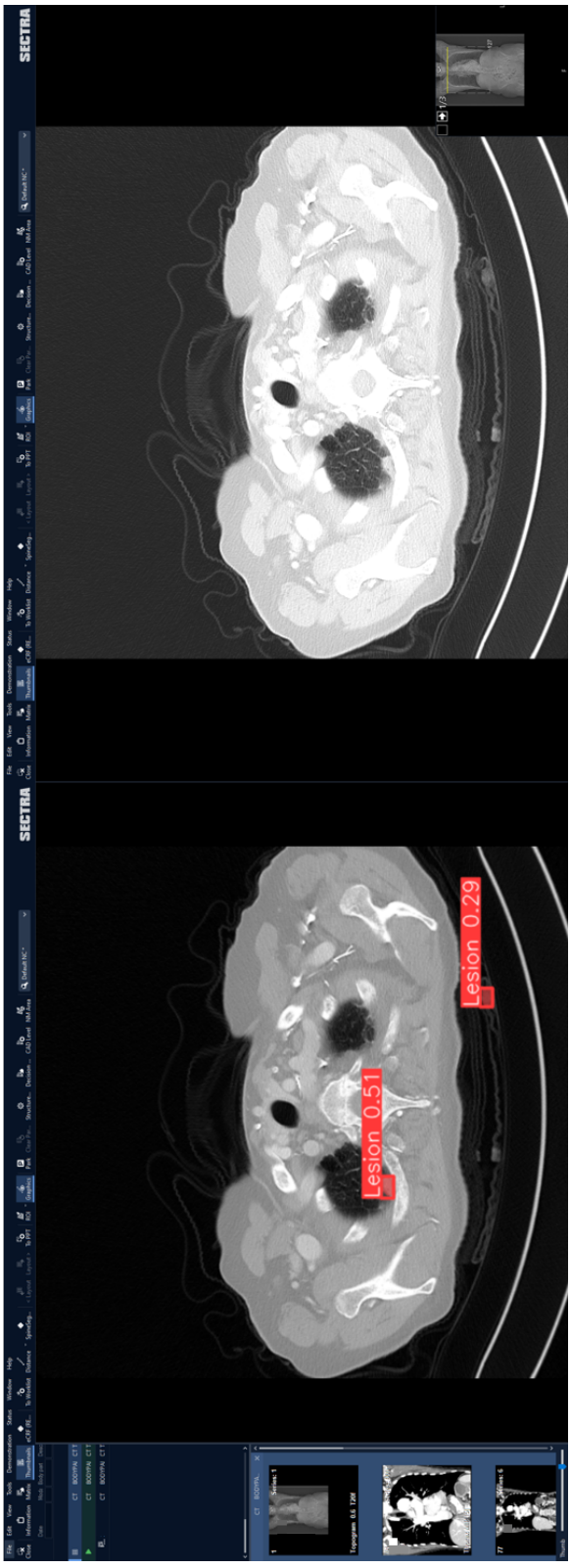


*Figure 4.17: Prediction of image slice on test data, compared to the original data. One of the segmentations is clearly wrong with a segmentation outside the patient.*

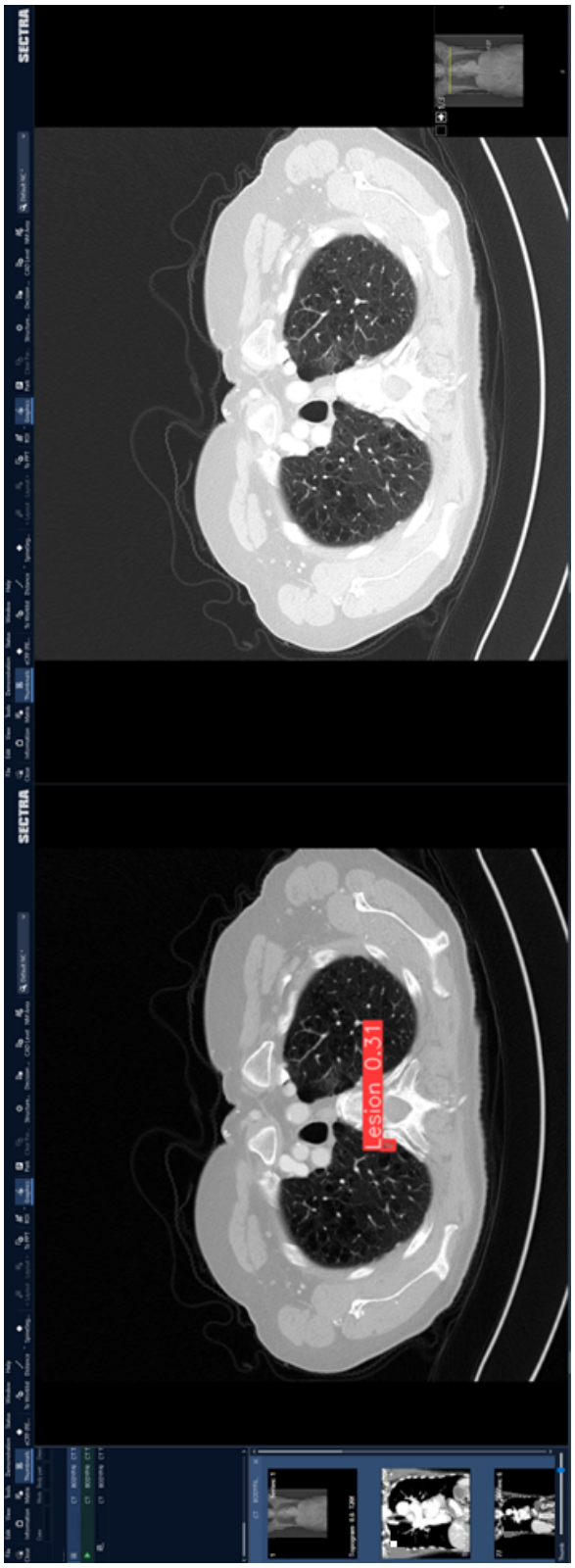


*Figure 4.18: Prediction of image slice on test data compared to the original image.*

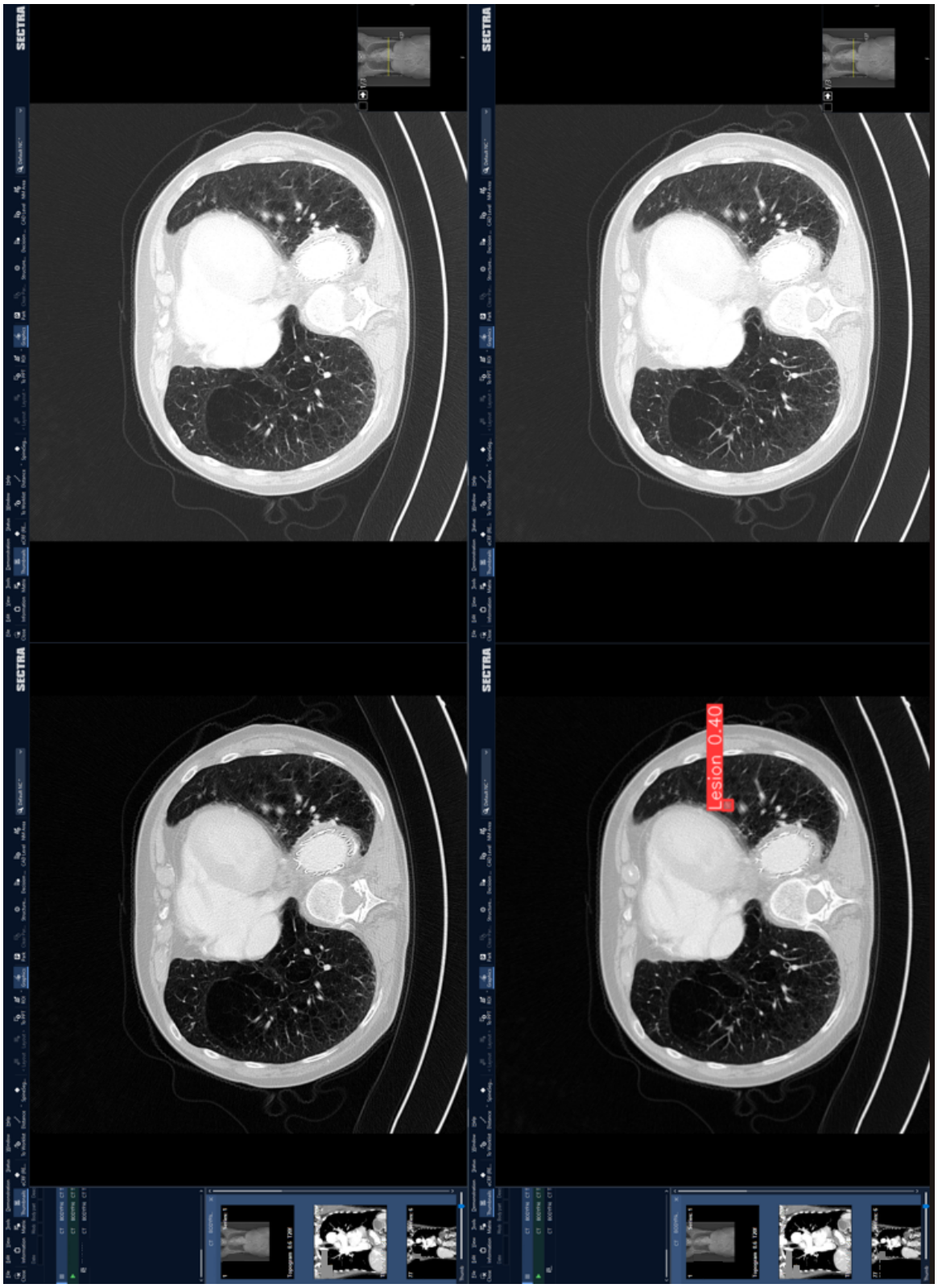


*Figure 4.19: Image series of predictions on test data, from the top to the bottom, compared to the original images. The image series continues in Figure 4.20.*

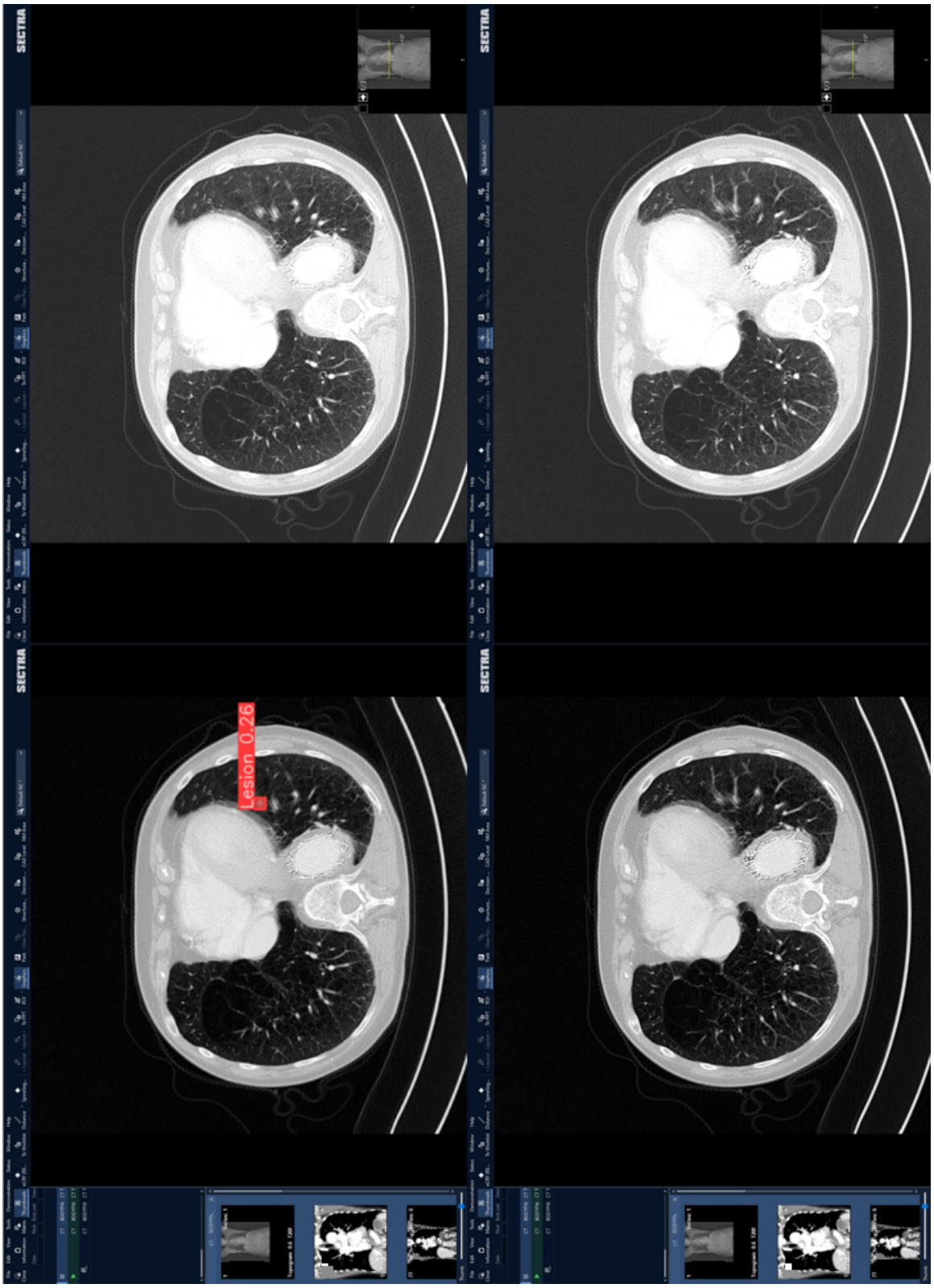


*Figure 4.20: Consecutive image series of predictions on test data, from the top to the bottom, compared to the original images. The image series is a continuance of the image series displayed in Figure 4.19.*

# Chapter 5

# Discussion

In this thesis, YOLO was successfully explored, adapted, and evaluated as a DL approach in terms of CT lung lesion segmentation and detection. This was done by extracting CT lung images and the corresponding XML files from the online dataset LIDC-IDRI, and then adapting the images and the corresponding data to match the yolov8 segmentation requirements, mentioned in Section 2.8.1, and then training the models with different hyperparameters, model sizes and lesion sizes, where the lesion is a 2D slice/part of a 3D lesion. The trained models demonstrated the ability to segment lesions from the converted 2D images with different performances. The different performance metrics used to evaluate the trained models are the DICE score, IoU, precision score, and sensitivity score, which are all metrics used in medical image segmentation tasks [50].

This thesis also successfully integrated the DL approach (YOLO) into a local clinical workflow for segmenting and detecting CT lung lesions. To achieve this, the best-performing trained model was uploaded and modified to fit the PACS structure. Further, the DL approach was tested on test data from the regional research PACS without pre-existing labels and segmentations, the DL approach was tested. The results on the test data showed promising results, where the model was able to segment lesions in 2D slice by slice consecutively.

In the following sections, the results obtained in Chapter 4 are examined, along with sources of errors and limitations from the methodology which might have influenced the results.

## 5.1 All lesions

The average segmentation-oriented performances for the models trained with every lesion size were a DICE score of 0.667 (66.7%), an IoU score of 0.554 (55.4%), a precision score of 0.823 (82.3%), and a sensitivity score of 0.651 (65.1%) on the validation set. Although the models perform to some degree with good average precision, indicating the model's ability to not overshoot the predicted masks significantly, the models also perform poorly compared to the 0.05% to 0.4% size-restricted lesions. The average sensitivity score for object detection-oriented tasks was 0.65 (65%), indicating that the trained models could detect 65% of the lesions presented in the validation set.

Compared to the models Zhi et al. [68] (with DICE score, sensitivity, and precision between 0.680-0.950, 0.584-0.874 and 0.820-0.869 respectively), the models trained in this thesis (with all lesion sizes) performed at the lower end of the spectrum. However, when accounting for standard deviation, the metrics are both at the high end of the spectra and far below the models compared by Zhi et al. There are several reasons for these discrepancies. The models compared in the article are trained with various datasets, various loss functions, and various image sizes and are trained with various images. Another important reason for these discrepancies is that many of the models are trained with special lesion requirements.

In contrast, the models trained with all lesions in this thesis only had one requirement: the number of unique coordinates to the lesion outlines, which had to be more than four coordinates, as mentioned in Section 3.2.1. With only that particular requirement, the trained models probably segmented the small lesions poorly, which would have affected the metrics poorly to a significant degree, as mentioned in Section 3.2.2. Additionally, the larger lesions probably also were not fully segmented. These claims are supported by examining the sensitivity and precision segmentation scores, especially when comparing those models to the models trained on the size-restricted dataset.

However, by training on all lesions, the models in this thesis are more generalized than the models compared by Zhi et al. and, thus, potentially more representative of what is expected in a clinical workflow. The majority of the models presented by Zhi et al. are more specified based on certain lesion requirements. In contrast, the models trained with every lesion size in this thesis will perform to some degree for a broader range of various lesions.

Compared to the object detection sensitivity scores achieved by the other YOLO-trained models, provided by Liu et al. (YOLOv3) [34] and Liying et al. (YOLOv5) [35] which scored 0.873 (87.3%) and 0.962 (96.2%), respectively, the models trained without size restriction in this thesis underperformed significantly. A reason for the underperformance may stem from the fact that the other YOLO models trained on smaller datasets with specific requirements. In contrast, the models presented in this thesis were not.

By not constraining the datasets, the models in this thesis (all lesions) are trained on data of various qualities, thus making them more applicable to CT image series with slice thicknesses greater than 2.5 mm (Liu et al.) for lesion detection. Although the majority of CT scans used to train the presented models had slice thicknesses $\leq$ 2.5 mm (over 66% of the images), they also trained on some scans with slice thicknesses greater 2.5 mm, which the models presented by Liu et al. did not. This may make the models in this thesis (all lesions) better at detecting lesions in such cases.

Similarly, if a random CT scan resembles the data used to train the model trained by Liying et al., it may outperform the models presented in this thesis (all lesions). However, due to the diverse data, the models in this thesis are likelier to perform better in lesion detection on a truly random CT scan.

Another reason for the underperformance is that the models presented by Liu et al. and Liying et al. are primarily designed to perform object detection. Whereas, the models presented in this thesis perform both segmentation and object detection. Since total loss (Equation 2.46) also contained a segmentation loss component, which may have affected the models' object detection performances during training. As a result, while the models presented by Liu et al. and Liying et al. might be better suited for locating and counting lung lesions, the models in this thesis (all lesions) can also provide segmentations of the detected lesions.

## 5.2 Size-restricted

The average performances for the models trained with size restriction 0.05% to 0.4%, relative lesion size compared to the whole image, were a DICE score of 0.834 (83.4%), a precision score of 0.864 (86.4%) and a sensitivity score of 0.835 (83.5%). These performance scores indicate that the models have a great ability to predict lesion masks that overlap with the ground truth masks, a great ability to segment points within the

ground truth without overshooting the predicted mask, and a great ability to segment points within the ground truth mask without undershooting the predicted mask respectively. The average sensitivity score for object detection was 0.91 (91%), which indicated that the trained models were able to detect 91% of the lesions presented in the validation set.

Compared to the models trained with all lesions, these size-restricted trained models perform significantly better. Even compared to the models from the paper by Zhi et al., the size-restricted trained model performs to the same standard. The reason for these similarities likely lies in the requirement for counting a lesion as a lesion in the models compared by Zhi et al. These models are trained on datasets containing lesions with the same character, making it easier for the model to extract important features than those trained on all lesions, which had broad spectra of various lesions.

The size-restricted trained models perform in the middle range of the object sensitivity scores obtained by training the models presented by Liu et al. and Liying et al. The difference in sensitivity compared to the models presented by Liu et al. only excluded CT scans with slice thicknesses greater than 2.5 mm. This makes the models by Liu et al. more suited for detecting lesions of various sizes, but the size-restricted-trained models are more suited for detecting lesions with the same relative size as those in the dataset.

The model presented by Liying et al. will likely perform better given a truly random CT scan. However, as for the models in this thesis trained without size restrictions, the size-restricted-trained models are trained for both segmentation and detection, thus changing the nature of the loss function, which is likely one reason why the model presented by Liying et al. performs better. Since Liying et al.'s model only are able to locate lesions, and the size-restricted models are able to perform both instance segmentation and lesion detection, the models in this thesis (size-restricted) are more suitable in terms of clinical use.

## 5.3 Comparison across models

When comparing the trained models, lesions of various sizes and locations were evenly distributed in the training and validation sets, which is probably the main reason why the trained models do not seem to overfit significantly to the training sets. This claim is supported by the fact that when comparing the segmentation metrics for both the

training and validation sets (Tables 4.2 - 4.7), the metrics on average do not differ by a significant amount.

The comparison across models, with Table 4.8, suggests that the size-restricted models were the most optimal for achieving the highest metric scores. However, these size-restricted trained models will probably not identify the smallest lesions, which might be a concern. Additionally, these models might not identify the larger lesions either. The overall best-performing model of the size-restricted ones is the large, most complex one. This suggests that the extra features extracted by the large model are relevant for size-restricted CT lung lesion segmentations, which raises the question of whether or not training a more complex YOLO model would have improved the results.

However, the size-restricted medium model seems to be the most overfitted, by examining Table 4.5, since the difference between the training and validation metrics are significant for three out of four metrics. By examining the equations corresponding to each performance metric, it seems like the medium size-restricted model tends not to fully segment lesions in the validation set, but since the precision metric is approximately equal for the training and validation set, it also seems like the model is capable of locating the lesions, but only has difficulty in segmenting the whole lesion to the same degree as done in the training set. To better optimize the segmentation, by making the training and validation scores more equal, the model could either have trained longer or trained with more data.

Looking at the sensitivity scores in the validation set in Table 4.8, both larger models have the lowest dataset corresponding sensitivities. By Equation 2.41, this corresponds to a higher prelevance of FNs, which indicates that the segmentations conducted by the models tend to segment less of the actual lesions compared to the smaller models. However, the difference between the large model and the other models is negligible for the size-restricted models.

Additionally, according to the metrics used in this thesis, the large model without size restrictions is the least performing model, as shown in Table 4.8. In contrast to the large model with size restriction, which holds the best-overall achieving scores, the results suggest that a more complex model might not be best suited for training a generalized CT lung lesion segmentation model. For future work, this could be explored by attempting to train a more complex YOLO model, which, if not disproved, might strengthen the hypothesis.

These results are as hypothesized: the models trained with the specific size restriction performed better than the models trained on all lesions for each model size. This is most likely due to the lower variation of lesion sizes in the size-restricted dataset, compared to the dataset with all lesions, and the fact that small deviations in the predicted masks for small lesions have a greater impact than predicted masks for larger lesions.

For further work, additional models could be trained with various size restrictions of different sizes, possibly extending the hypothesis for multiple size restrictions. However, since a small deviation in the prediction of smaller lesions has a greater impact than small deviations on larger lesions, models trained with tiny lesions will probably not achieve high performance metrics scores, which also allows further research to determine the smallest lesion detectable and segmentable, given a size restriction, with YOLO as a DL algorithm.

## 5.4 Dataset

The LIDC-IDRI dataset was the dataset used to train the models. In the dataset, there are four radiologists who identified and segmented the lesions used to train the models. With these segmented images, the models learned which features were relevant and not for detection and segmentation, which directly affected the performance of each model.

However, there are some considerations about the dataset. First, since the training was conducted by 2D images. In the dataset, and CT scans in general, for each 2D image/slice, other common objects, e.g., blood vessels, might visually appear as lesions in 2D. For the validation set during training and prediction with the test data from the regional research PACS, these objects might be labeled as lesions. However, these wrong predictions in the validation set do not interfere with the calculation of the performance metrics due to the adjustments mentioned in Section 3.2.2, although these wrong predictions should have had a negative impact on the calculated metrics. As for the predictions with test data, these cases probably are easily recognized to be FPs by radiologists and will not interfere in a significant way.

Additionally, the lesions marked by the radiologists in the XML files are only the lesions those individuals were able to locate. Thus, there might be additional lesions in the CT scans that the radiologists missed, or the radiologists might have marked lesions to have a smaller outline than they actually have. For the validation set during train-

ing, missed lesions can result in correct predictions done by the model being labeled as false. For the performance metrics, both missed lesions and wrongfully marked outlines can cause every performance metric used in this thesis to be equal to zero for the predicted tumor, as illustrated in Figure 3.2.1. However, these predictions in the validation set do not interfere with the calculation of the performance metrics due to the adjustments mentioned in Section 3.2.2, although these predictions actually should have had a positive impact on the calculated metrics.

Finally, the adjustment done in the methodology was to only keep the overlapping region of the lesions segmented by multiple radiologists from the LIDC-IDRI dataset, which also contained slight deviations as illustrated in Figure 3.4. These deviations could also negatively impacted the performance of each model trained. Without this adaption, the model would have been confused because multiple segmentations with the same label would have existed in the same region. To illustrate the effect of this adjustment, the models could have been trained with multiple labels for the segmentation, e.g., by labeling the regions according to the radiologist and one with the overlapping region. However, this would be time-consuming and unnecessary because when training a model with only the overlapping region, the trained model will be less biased, but when training with labels corresponding to each radiologist, these predictions become biased towards the radiologist. With this reasoning, by training the model with only the overlapping, the training becomes less time-consuming and less biased.

## 5.5 Performance metrics

The performance metrics are used to evaluate the models performance after training, and these metrics are also used to compare the performance of the purposed models to other models trained with the same goal, as done in Section 5.1 and Section 5.2.

Since these metrics directly affect the presentations of the various models, there are some considerations to consider according to the calculations of these metrics. Firstly, for small lesions, a small deviation in the predicted mask versus the ground truth mask will impact the performance metrics used in this thesis to a more significantly negative degree compared to the same scenario with larger lesions. This can be a reason why the performance metric scores for the models trained on the whole dataset have lower scores compared to the models trained on the size-restricted dataset.

When the performance metrics are equal to zero, the scores are further excluded for

generating the average metric scores provided in Chapter 4. Although some of these cases might be when the trained model predicts a lesion where there is none, as previously mentioned, the radiologists might also have either segmented the lesion with false outlines or the radiologist might have missed some lesions the trained model is able to identify and segment. The final case is when there are either multiple predicted lesions and equally many ground truths or an uneven number between predicted lesions and ground truths. Here, the case of determining which predicted mask corresponds to the correct ground truths is a challenging task. To further optimize the calculations one could count the frequency of every case mentioned and illustrated in Figure 3.2.1, then investigate these cases to further determine the weighting of each case. However, due to time limitations and uncertainty between radiologists, this option could not be provided in this thesis.

## 5.6 Time restrictions and computational power

During the course of this thesis, due to time restrictions and computational power, some model optimization, model modification, and application paths were not explored further during this thesis. This section explains the individual effects of not exploring these paths in greater detail.

It is not possible to determine if the hyperparameters used are the most optimal hyperparameters. One limitation of the thesis is that the results provided are just from hyperparameter searches, with only a fraction of all possible hyperparameter values and combinations. By conducting wider searches with more values and combinations, the results provided in Chapter 4 might even be optimized further. However, one can never explore every hyperparameter value and combination. Hence, the hyperparameter searches conducted in this thesis are satisfactory, but in future work, exploring other hyperparameter values and combinations may be considered to optimize the trained model even better.

For the segmentation task, the models were only trained on images converted from DICOM to PNG with the command "+Wh 1", and it is possible the trained models might have performed better with another conversion method, either those purposed in Section 3.2.1 or other conversions. This option was also not explored due to time restrictions, and since the used conversion provided satisfactory results, this option was deemed unnecessary. However, this might be considered in future work.

Evaluation metrics are not universal for particular tasks. Hence, the evaluation metrics used in Chapter 4 might be comparable for every medical professional and computer scientist. By calculating additional performance metrics the results would have become more comparable for people with different backgrounds. These calculations would have required even more scripts and code and were deemed unnecessary since the most conventional metrics in medical use were included. However, people with general knowledge of statistics are familiar with IoU, and by including the IoU metric, even though DICE and IoU are 100% correlated, the results provided in this thesis are more easily comparable for people with various knowledge from various fields.

For the small lesions in the dataset with all lesion sizes, the training might have benefited from enlarging the image sizes to a greater degree than done in this thesis (640x640), e.g. 1012x1012. By enlarging the image, the small lesions would appear bigger, and during the training, the models could have extracted some features. However, due to computational power and model complexity, the computer used to train these models in this thesis could not train such models. An attempt to enlarge the images in this dataset was made through the scale hyperparameter, but since the scale hyperparameter also shrinks some images by the same factor, which actually could have had a negative impact on the performance.

The models were only trained on two different datasets, one with all lesions of every size and one with the size restriction 0.05% to 0.4% lesion area compared to the image size. An interesting and highly recommended aspect for future work is training the models on multiple size restrictions to specify each model. This will provide insights about which size restrictions the YOLO algorithm performs best on, and in clinical practice one could run every trained model, with every size restriction, simultaneously to optimize the predictions and segmentations of lesions.

## 5.7 Clinical integration

In the regional research PACS system, the small size-restricted trained DL model was uploaded and tested. The model was chosen for its simplicity and overall good performance. The main purpose of the integration was to establish a clinical workflow for using DL/AI to detect and segment lung lesions.

The integration of the DL algorithm (YOLO) to the clinical workflow shows promising results. The consecutive image series from Figure 4.15 and Figure 4.16 are predictions

with segmentations of distinct areas. The bottom left segmented area in this series, which also has a high confidence score, indicates a possibility of an actual 3D tumor existing in the area.

However, not every prediction was equally successful, where some segmentations obviously are faulty. Furthermore, some blood vessels and other normal regions are segmented as lesions and obvious tumors do not get segmented. Predictions that are obviously false are also displayed in this thesis, where in Figure 4.17, a region outside the patient was segmented and identified to be a lesion. Another obviously faulty prediction displayed in this thesis is shown in Figure 4.16, where the top left segmented part of the bottom right image is also faulty, which is obvious when viewing the image series from the start. Segmentations which do not seem faulty when only viewing the image with the predicted lesion, but by closer inspection might seem like a blood vessel or other normal healthy regions is the image series provided in Figure 4.19 and Figure 4.20. The reason why this segmented lesion raises suspicion is the rapid displacement from slide to slide, which indicates the lesion predicted actually is a blood vessel. However, the segmentation could be inspected by a radiologist to confirm. The wrongful predictions inside the lungs most likely stem from the fact the model was trained on 2D images, and in a particular slice and certain angle, healthy objects, e.g. blood vessels, might look like lesions to the model.

Although, every segmentation of the model is not guaranteed to be a lesion of interest, in cases such as the image series in Figure 4.15 and 4.16, an interesting aspect would have been to try to interpolate between the 2D lesion segmentations to create a 3D lesion volume, and also extract other features such as physical volume and lesion diameter. The DL algorithm integrated has the potential to increase performance by retraining on correct segmentations, which over time could lead to the detection and segmentation of lesions of various sizes.

# Chapter 6

# Conclusions and Future Work

The evaluation of YOLO as a lung lesion segmentation and detection model in CT imaging was successful and showed promising results. As expected, the model trained on the dataset with size-restricted lesions performed better than the model trained on the whole dataset. In comparison to other models trained for 2D lung lesion segmentation, the results with the size restriction trained model, on average, exhibit similar performance to those of Zhi et al., where the majority of the models are based on the U-Net architecture. In terms of object detection, size-restricted trained models exhibit better performance than the models presented by Liu et al., but inferior to the model presented by Liying et al. However, since the models in this thesis are trained for both lesion segmentation and detection, they are more suitable in terms of clinical usage.

The local clinical workflow integration also showed promising results, where the model uploaded was able to successfully segment and detect several 2D lesion slices of a 3D lesion consecutively, as displayed in Figure 4.15 and Figure 4.16. Hence, integrating the pipeline into the regional research PACS was successful, which provides the opportunity to detect and segment lesions that healthcare professionals fail to notice. As mentioned in Section 5.7, retraining the model with correct predictions has the capability to improve the model's ability over time, and it might be able to detect and segment smaller and larger lesions over time.

To further improve the models and the results in Chapter 4, a larger hyperparameter search can be conducted to further minimize the total validation loss which theoretically should provide a better segmentation and detection model. Another addition may be to add weighting for the calculation of performance metrics relative to the lesion size, since small deviations in smaller lesion predictions more negatively impact the calculation of performance metrics compared to small deviations in bigger lesion pre-

dictions.

Additionally, the models could be trained with various size restrictions, creating a set of models that together can provide detection and segmentation for lesions of every size. By adding weighting for the performance measures based on the predicted lesion sizes, the models trained have the potential to achieve this.

Considering that the model demonstrates a certain ability to detect and segment lung lesions from slices of a CT scan in consecutive order, further work could be to construct 3D segmentations from the already existing 2D segmentations, e.g. by interpolation. Other additions worth exploring are to extract certain features from the segmented lesions such as: malignancy status, calcification, and whether or not the segmented lesion is a primary tumor, secondary tumor, or another anomaly within the lungs. With these features added, the model will be able to provide more detailed information and possibly be used to detect diseases other than lung cancer and segment other regions of interest associated with that particular disease.

After successfully creating 3D segmentations, and considering that patients tend to undergo follow-up examinations, one could potentially train a model to predict tumor development using the 3D segmentations along with the corresponding longitudinal data. This approach could ultimately lead to groundbreaking research in understanding tumor development patterns and metastasis. In a clinical workflow, this would translate to both preparation and prevention of potential metastasis, resulting in preventing casualties. Additionally, this could provide estimations about treatment responses based on factors such as patient anatomy, tumor localization, etc., which would help shape the treatment plan to the patient and thus potentially save lives by eliminating ineffective treatment plans.

# Appendix I

# MMIV Conference 2023 poster

The poster provided in Figure I.1 is the poster presented at the MMIV Conference, displaying this thesis's preliminary results.

# Longitudinal assessment of lung metastasis using machine learning

Heris Sivanesarajah[1], Hauke Bartsch[2], Frank Riemer[2], Renate Grüner[1,2]
1. Department of Physics, University of Bergen. 2. MMIV, Department of Radiology, Haukeland University Hospital

## Introduction

- Lung metastasis is cancer cells traveling from another organ to the lungs, and the cells manifests there.
- Worldwide, about 1500 people die from lung metastasis daily, due to the lack of treatment [1].
- This is why early detection and longitudinal assessment is key to treat patients with lung metastasis.
- Disease management includes early detection of metastasis and ability to predict disease development.
- The Goal: To use AI models on CT-images of the thorax to segment lesions and foretell development.

## Methods

- For methodological development, an online database with labelled lung lesions was used, LIDC-IDRI dataset [2] with over 1000 patients, mostly primary lung lesions
- Both an object detection model and a segmentation model were trained on the online data set using the YOLOv8 algorithm [3].
- Data pre-processing included data conversion (png) and feeding the algorithm with information about the edges of the nodules in the format [Class-ID, x0, y0, x1, y1,….], which were extracted from the XML-file to each CT-scan.
- To enhance performance the smallest nodules were omitted, and the minimum nb of "edgepoints" set to 41.
- The segmentation model displayed in this poster were trained with a train/val-test split 80/20%, 60 epochs, learning rate of 0.01 and batch size 30.

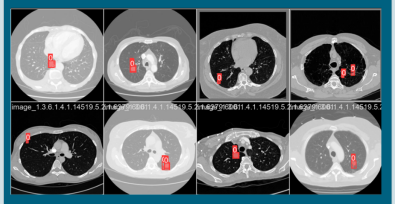

***Figure 1: An example of a training batch for the segmentation and object detection model with «edgepoints» > 40.***

## Results

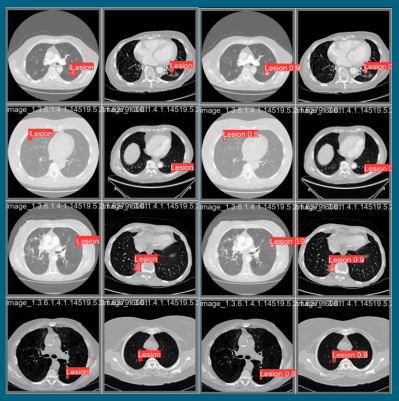

*Figure 2: The true labelled segmented lesions in the two colums to the left and the predicted to the right. Note that some of the predicted only have the bounding-boxes and some of them also have a red segmented lesion.*

- The preliminary results from the segmentation and object detection model is visualized in Figure 2, 3 and 4.
- The results are promising with some segmented nodules, but by viewing Figure 2 and 3, we can see there is still room for improvement.

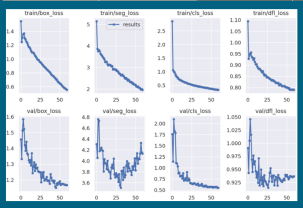

*Figure 3: Training and validation losses plotted in retrespect to number of epochs.*

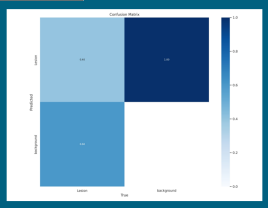

*Figure 4: Normalized confusion matrix from the bounding boxes of the segmentation model.*

## Discussion and Conclusion

- Preliminary results suggest that segmentation of lung lesions is feasible using the YOLOv8 algorithm.
- Data augmentation can be applied to better the results.
- Subsequently, there is a need to transfer these 2D bounding-boxes and segmentation to 3D, and train a model that can predict tumor development over time.
- The developed methodology will also be applied to other data (lung metastasis).

## References

1. A. Jamil and A. Kasi, "Lung metastasis" (2023), https://www.ncbi.nlm.nih.gov/books/NBK553111/
2. CANCER IMAGING ARCHIVE "Data from The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A completed reference database of lung nodules on CT scans (LIDC-IDRI)" (accessed 2023), https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=1966254
3. Ultralytics "Ultralytics YOLOv8 Docs" https://docs.ultralytics.com/

## Acknowledgments

- The CANCER IMAGING ARCHIVE for the dataset that got the project started.
- The National Cancer Society for funding the MALIGN project (to R.G)
- The master students at medical technology and medical physics, who have kept me with good company during this project (still ongoing).
- Thanks to the employees at MMIV who have been kind enough to help me and for the good environment at the office.

MMIV HELSE BERGEN Haukeland universitetssjukehus

*Figure I.1: The poster presented at the 2023 MMIV Conference.*

# Appendix II

# GitHub repository

The code used in this thesis to execute the methodology presented in Chapter 3, and the code used to generate the following results are provided in the GitHub repository accessible through the following url:

https://github.com/herissiv/lung-ai

# Appendix III

# Results with Hyperparameter tuner

This appendix provides the results obtained after utilizing YOLO's builtin hyperparameter tuner on the dataset without size restriction with the corresponding model complexities provided in Chapter 4. For the small and medium model trained (after tuning), the object detection sensitivity scores are provided through the column-normalized confusion matrices (Figure III.2 and Figure III.4) indicating sensitivity scores of 0.68 and 0.69 respectively. The segmentation performance metrics are provided for all model complexities (small, medium, and large) from Table III.1 - III.3. Examples of predictions made by the models with their corresponding ground truths are given in Figure III.1, Figure III.3, and Figure III.5.

## 1 All lesions small model

| | Training set | Validation set |
|---|---|---|
| Performance metric | score $\pm$ std | score $\pm$ std |
| DICE | $0.696 \pm 0.226$ | $0.673 \pm 0.238$ |
| IoU | $0.573 \pm 0.232$ | $0.548 \pm 0.235$ |
| Precision | $0.822 \pm 0.174$ | $0.791 \pm 0.219$ |
| Sensitivity | $0.665 \pm 0.259$ | $0.676 \pm 0.253$ |

*Table III.1: The performance measures obtained by the builtin hyperparameter tuner, with the related scores with standard deviation, from the small trained segmentation model.*

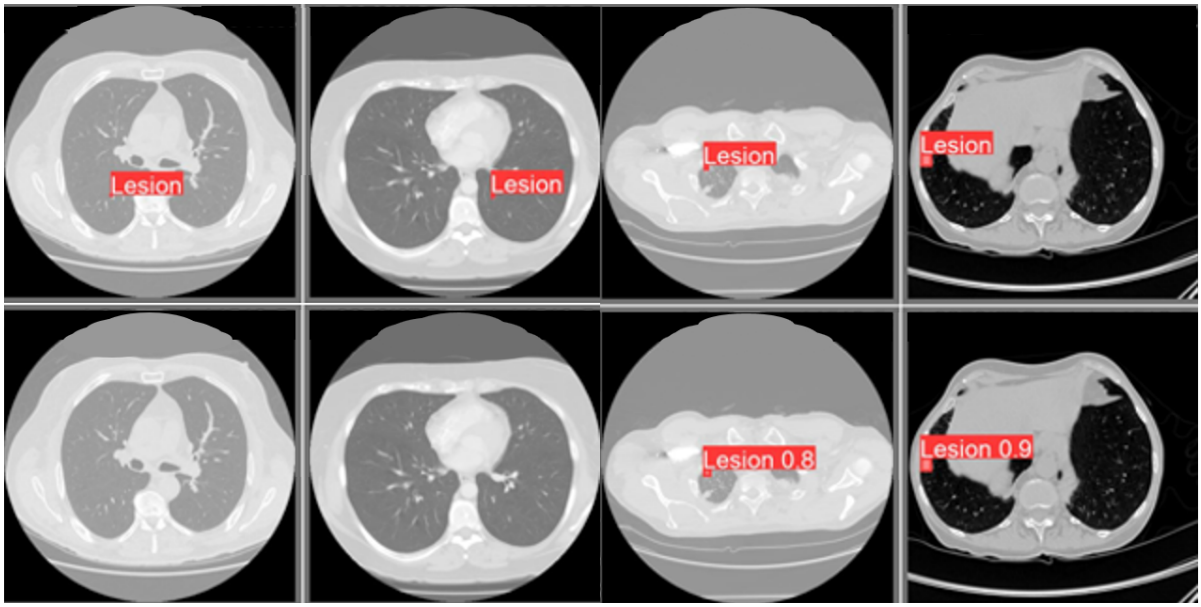


*Figure III.1: Predictions made by the hyperparameter tuner trained model (small model) in the bottom row, with the corresponding ground truths at the top row.*

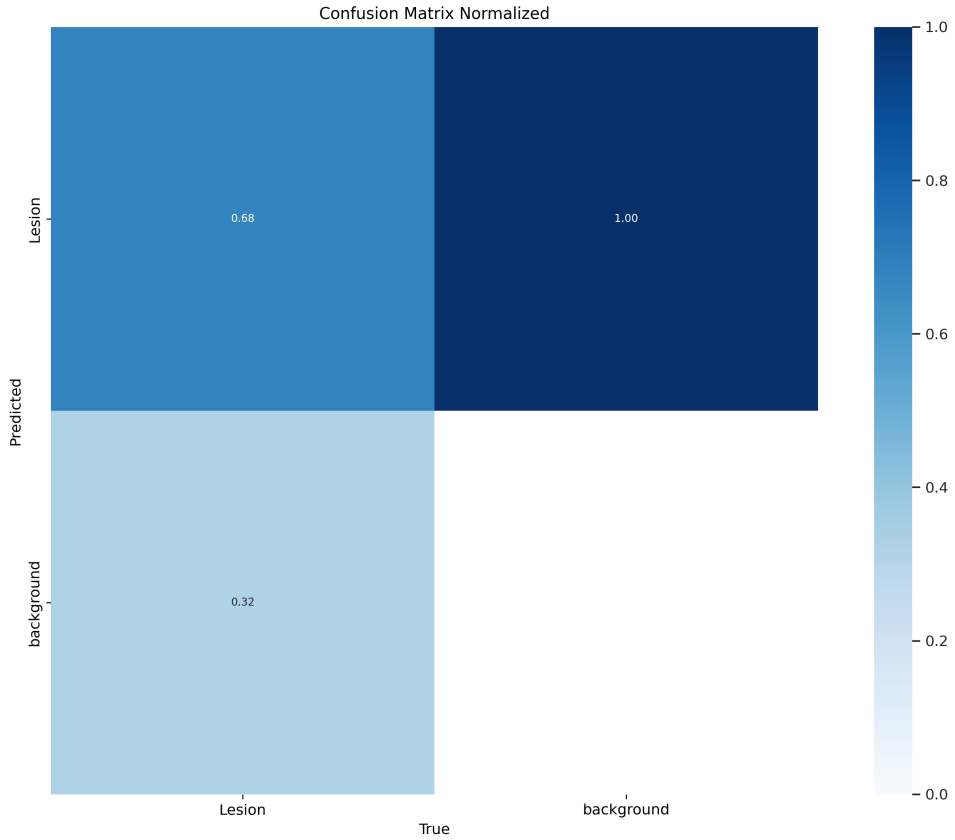


*Figure III.2: The normalized confusion matrix obtained by the builtin hyperparameter tuner from the small trained model. From the confusion matrix, the object detection sensitivity score is 0.68.*

# 2 All lesions medium model

| | Training set | Validation set |
|---|---|---|
| Performance metric | score $\pm$ std | score $\pm$ std |
| DICE | $0.714 \pm 0.207$ | $0.678 \pm 0.233$ |
| IoU | $0.591 \pm 0.219$ | $0.553 \pm 0.230$ |
| Precision | $0.827 \pm 0.167$ | $0.782 \pm 0.227$ |
| Sensitivity | $0.683 \pm 0.242$ | $0.690 \pm 0.244$ |

*Table III.2: The performance measures obtained by the builtin hyperparameter tuner, with the related scores with standard deviation from the medium trained segmentation model.*

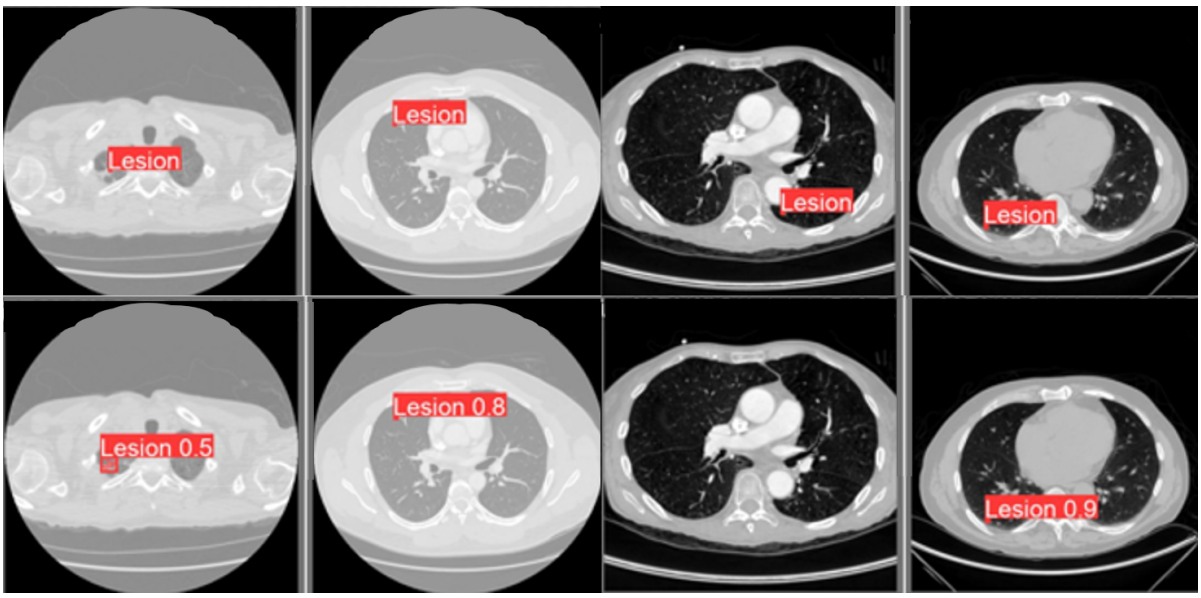


*Figure III.3: Predictions made by the hyperparameter tuner trained model (medium model) in the bottom row, with the corresponding ground truths at the top row.*

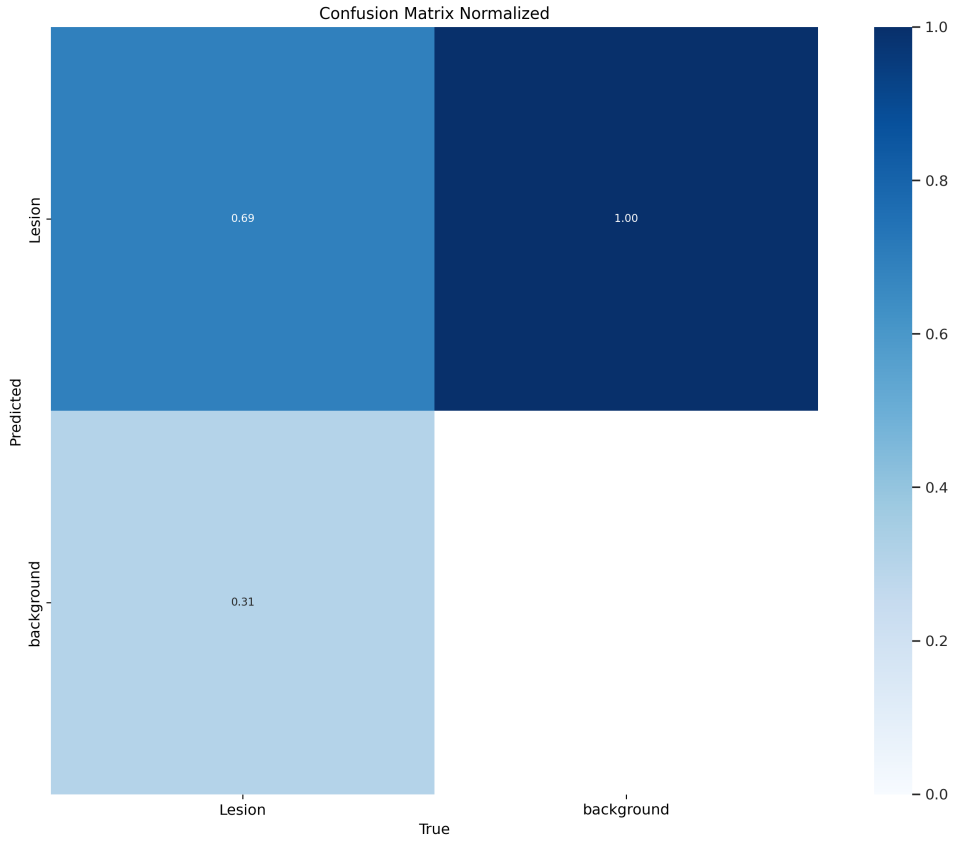


*Figure III.4: The normalized confusion matrix obtained by the builtin hyperparameter tuner from the medium trained model. From the confusion matrix, the object detection sensitivity score is 0.69.*

# 3 All lesions large model

| | Training set | Validation set |
|---|---|---|
| Performance metric | score $\pm$ std | score $\pm$ std |
| DICE | $0.687 \pm 0.239$ | $0.663 \pm 0.253$ |
| IoU | $0.566 \pm 0.242$ | $0.541 \pm 0.226$ |
| Precision | $0.820 \pm 0.173$ | $0.785 \pm 0.226$ |
| Sensitivity | $0.654 \pm 0.271$ | $0.670 \pm 0.268$ |

*Table III.3: The performance measures obtained by the builtin hyperparameter tuner, with the related scores with standard deviation, for the large segmentation model.*

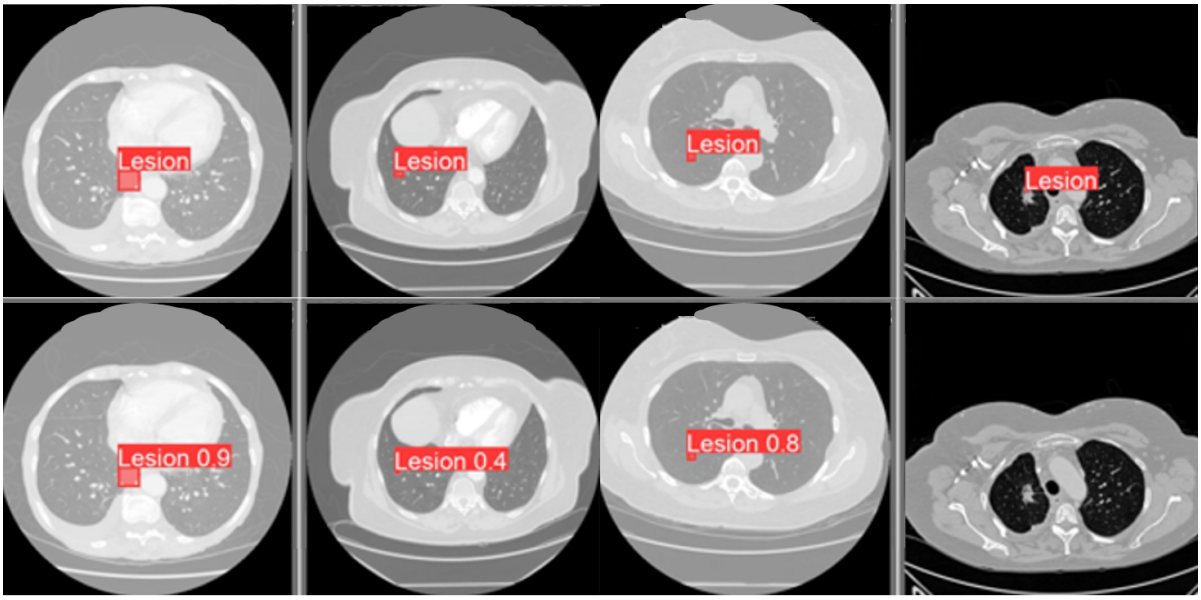


*Figure III.5: Predictions made by the hyperparameter tuner trained model (large model) in the bottom row, with the corresponding ground truths at the top row.*

# Bibliography


[1] G. C. Observatory, "Estimated number of new cases in 2020, world, both sexes, all ages (excl. nmsc)." [Online]. Available: https://gco.iarc.fr/today/home 1.1

[2] K. C. Thandra, A. Barsouk, K. Saginala, J. S. Aluru, and A. Barsouk, "Epidemiology of lung cancer," *Contemporary oncology (Poznan, Poland)*, 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8063897/ 1.1

[3] N. I. of Biomedical Imaging and Bioengineering, "Computed tomography (ct)," 2022. 1.1, 2.1.1, 2.1.6

[4] A. M. A. Gindi, T. A. Attiatalla, and M.-S. M. Mostafa, "A comparative study for comparing two feature extraction methods and two classifiers in classification of early-stage lung cancer diagnosis of chest x-ray images," *Journal of American Science*, 2014. 1.1

[5] R. C. Deo, "Machine learning in medicine," *Circulation*, 2015. 1.1

[6] L. E. Romans, *COMPUTED TOMOGRAPHY for TECHNOLOGISTS*. Wolters Kluwer, 2011. 2.1.1, 2.1.5, 2.1.5, 2.1.6, 2.1.6, 2.1.6, 2.1.6, 2.1.6, 2.1.7, 2.1.8

[7] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. Global Edition, 2018. 2.1.1, 2.1.6, 2.9, 2.1.6, 2.1.6

[8] W. A. Kalender, *Computed Tomography*, 3rd ed. Publicis, 2011. 2.1.1, 2.1.3, 2.1.5, 2.1.5, 2.1.5, 2.1.5, 2.1.5, 2.1.5, 2.1.5, 2.1.6, 2.1.6, 2.1.6, 2.1.6, 2.1.6, 2.1, 2.1.6, 2.1.8, 2.10

[9] R. M. Sainz, F. Lombo, and J. C. Mayo, "Radical decisions in cancer: redox control of cell growth and death," *Cancers*, 2012. 2.1.1

[10] L. Eldridge, "Free radicals: Definition, cause, and role in cancer," October 2022. [Online]. Available: https://www.verywellhealth.com/information-about-free-radicals-2249103 2.1.1

[11] N. C. I. (NCI), "Computed tomography (ct) scans and cancer," 2019. [Online]. Available: https://www.cancer.gov/about-cancer/diagnosis-staging/ct-scans-fact-sheet 2.1.1, 2.1.7

[12] *Radiation Oncology Physics*, ser. Non-serial Publications. Vienna: INTERNATIONAL ATOMIC ENERGY AGENCY, 2005. [Online]. Available: https://www.iaea.org/publications/7086/radiation-oncology-physics 2.1.2, 2.1, 2.1.2, 2.1.2, 2.1.2, 2.1.2, 2.1.2, 2.1.3, 2.1.3, 2.1.3, 2.1.3, 2.1.3, 2.1.4, 2.1.4, 2.3, 2.1.4, 2.1.4, 2.1.5, 2.8, 2.1.5, 2.1.6

[13] T. E. Johnson, *Introduction to Health Physics*, 5th ed. McGraw-Hill Education, 2017. 2.1.2, 2.2, 2.1.3

[14] J. C. Morrison, *Modern Physics for Scientists and Engineers*, 2nd ed. Academic Press, 2015. 2.1.2, 2.1.2

[15] B. J. McParland, *Nuclear Medicine Radiation Dosimetry*. Springer, 2010. 2.1.2

[16] H. Lodish, A. Berk, C. A. Kaiser, M. Krieger, A. Bretscher, H. Ploegh, A. Amon, and K. C. Martin, *Molecular Cell Biology*, 8th ed. w.h.freeman, 2016. 2.1.3

[17] S. Prabhu, D. K. Naveen, S. Bangera, and S. B. Bhat, "Production of x-rays using x_ray tube," *Journal of Physics: Conference Series*, 2020. 2.1.5, 2.1.5

[18] C. Rampinelli, D. Origgi, and M. Bellomi, "Low-dose ct: technique, reading methods and image interpretation." *Cancer imaging : the official publication of the International Cancer Imaging Society, 12(3)*, 2013. 2.1.7, 2.1.7

[19] N. C. I. (NCI), "Ldcy." [Online]. Available: https://www.cancer.gov/publications/dictionaries/cancer-terms/def/ldct 2.1.7

[20] S. Iranmakani, A. R. Jahanshahi, P. Mehnati, T. Mortezazadeh, and D. Khezerloo, "Image quality and pulmonary nodule detectability at low-dose computed tomography (low kvp and mas): A phantom study." *Journal of medical signals and sensors, 12(1), 64–68.*, 2021. 2.1.7

[21] A. Bonney, R. Malouf, C. Marchal, D. Manners, K. M. Fong, H. M. Marshall, L. B. Irving, and R. Manser, "Impact of low-dose computed tomography (ldct) screening on lung cancer-related mortality," *The Cochrane database of systematic reviews, Issue 8. Art. No.: CD013829.*, 2022. 2.1.7

[22] N. N. H. Service), "Lung cancer." [Online]. Available: https://www.nhs.uk/conditions/lung-cancer/ 2.2, 2.2

[23] C. R. UK, "Types of lung cancer." [Online]. Available: https://www.cancerresearchuk.org/about-cancer/lung-cancer/stages-types-grades/types 2.2, 2.2

[24] A. Jamil and A. Kasi, "Lung metastasis," *StatPearls [Internet]*, January 2023. 2.2, 2.3

[25] N. H. S. (NHS), November 2022. [Online]. Available: https://www.nhs.uk/conditions/lung-cancer/symptoms/ 2.2

[26] J. Ko, M. M. Winslow, and J. Sage, "Mechanisms of small cell lung cancer metastasis." *EMBO molecular medicine*, 2021. 2.2

[27] G. M. Stella, S. Kolling, S. Benvenuti, and C. Bortolotto, "Lung-seeking metastases," *Cancers*, 2019. 2.3

[28] X. Du-Harpur, F. M. Watt, N. M. Luscombe, and M. D. Lynch, "What is ai? applications of artificial intelligence to dermatology." *The British journal of dermatology*, 2020. 2.4

[29] OpenAI, "ChatGPT: OpenAI's Conversational AI Model," OpenAI, Technical Report, 2019. [Online]. Available: https://openai.com/chatgpt 2.4

[30] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challanges, applications, future directions," *Journal of Big Data*, 2021. 2.5, 2.6, 2.6.1, 2.6.1, 2.6.1, 2.6.1, 2.15, 2.6.1, 2.6.2, 2.6.3, 2.6.3, 2.6.3, 2.6.3, 2.6.4, 2.6.4, 2.6.5, 2.6.6, 2.6.6, 2.6.6, 2.6.6, 2.6.6, 2.6.8, 2.7.1

[31] Z.-H. Zhou, *Machine Learning*. Springer, 2012. 2.5, 2.6, 2.6.1, 2.6.3, 2.6.3, 2.6.3, 2.7.1, 2.7.6

[32] M. R. Bonyadi, R. Wang, and M. Ziaei, "Self-punishment and reward backfill for deep q-learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2.5

[33] C. Bhagya and A. Shyna, "An overview of deep learning based object detection techniques," in *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*. IEEE, 2019, pp. 1–6. 2.5.1

[34] C. Liu, S. C. Hu, C. Wang, K. Lafata, and F. F. Yin, "Automatic detection of pulmonary nodules on ct images with yolov3: development and evaluation using

simulated and patient data." *Quantitative imaging in medicine and surgery*, 2020. 2.5.1, 2.8, 2.9, 5.1

[35] L. Han, F. Li, H. Yu, K. Xia, Q. Xin, and X. Zou, "Birpn-yolovx: A weighted bidirectional recursive feature pyramid algorithm for lung nodule detection." *J Xray Sci Technol*, 2023. 2.5.1, 2.9, 5.1

[36] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE*, 2022. 2.5.2, 2.5.2

[37] Q. Health, "Brain map," 2022. [Online]. Available: https://www.health.qld.gov.au/abios/asp/brain 2.12

[38] M. M. K. Aarhus, "Machine learning in automated segmentation of small lesions in magnetic resonance imaging for multiple sclerosis," 2023. 2.6, 2.13, 2.6.1

[39] L. Lu, Y. Zheng, G. Carneiro, and L. Yang, *Deep Learning and Convolutional Neural Networks for Medical Image Computing*. Springer, 2017. 2.6.1

[40] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021. 2.6.1, 2.22

[41] theletz stackoverflow, "small learning rate vs big learning rate," 2020, only figure inspiration. 2.17

[42] P. M. Radiuk, "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets," *Information Technology and Management Science*, vol. 20, no. 1, pp. 20–24, 2017. 2.6.3

[43] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for nas," 2019. 2.6.4

[44] G. Nakerst, J. Brennan, and M. Haque, "Gradient descent with momentum—to accelerate or to super-accelerate?" *arXiv preprint arXiv:2001.06472*, 2020. 2.6.6, 2.6.6

[45] S. Ruder, "An overview of gradient descent optimization algorithms," 2017. 2.6.6

[46] K. Nakamura and B.-W. Hong, "Adaptive weight decay for deep neural networks," 2019. 2.6.6

[47] R. Wan, Z. Zhu, X. Zhang, and J. Sun, "Spherical motion dynamics: Learning dynamics of neural network with normalization, weight decay, and sgd," 2020. 2.6.6

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2017. 2.6.6

[49] I. Muraina, "Ideal dataset splitting ratios in machine learning algorithms: general concerns for data scientists and data analysts," in *7th International Mardin Artuklu Scientific Research Conference*, 2022, pp. 496–504. 2.6.7, 2.6.7

[50] A. A. Taha and A. Hanbury, "Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool," *BMC medical imaging*, 2015. 2.7, 2.7.1, 2.7.2, 2.7.3, 2.7.3, 2.7.4, 2.7.4, 2.7.5, 5

[51] B. He, "A machine learning approach for data unification and its application in asset performance management," 2016. 2.7.6

[52] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics 2.8, 3.2.2

[53] Ultralytics, November 2023. [Online]. Available: https://docs.ultralytics.com/#where-to-start 2.8

[54] D. M, B. J, L. J, K. O, and S. A, "Study on sperm-cell detection using yolov5 architecture with labaled dataset," *Genes (basel)*, 2023. 2.8

[55] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," 2019. 2.8.1

[56] G. Jocher, "summary of yolov8-seg model structure," 2024, gitHub issue. [Online]. Available: https://github.com/ultralytics/ultralytics/issues/1710 2.8.1

[57] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2017. 2.8.1

[58] Ultralytics, November 2023. [Online]. Available: https://docs.ultralytics.com/tasks/segment/ 2.8.1, 3.3

[59] ——, November 2023. [Online]. Available: https://docs.ultralytics.com/usage/cfg/ 2.8.2

[60] G. Jocher, "Are class and box losses calculated the same in yolov8 and yolov5?" 2023, gitHub issue. [Online]. Available: https://github.com/ultralytics/ultralytics/issues/2789 2.8.3

[61] ——, "What is the loss used for yolov8-seg? what is the formula?" 2023, gitHub issue. [Online]. Available: https://github.com/ultralytics/ultralytics/issues/3882 2.8.3

[62] Ultralytics, "loss.py - ultralytics," 2024, source code. [Online]. Available: https://github.com/ultralytics/ultralytics/blob/main/ultralytics/utils/loss.py 2.8.3

[63] G. J. (Ultralytics), "Efficient hyperparameter tuning with ray tune and yolov8," 2024. [Online]. Available: https://docs.ultralytics.com/integrations/ray-tune/ 2.8.4, 2.8.4, 3.3.1

[64] ——, "Ultralytics yolo hyperparameter tuning guide," 2024. [Online]. Available: https://docs.ultralytics.com/guides/hyperparameter-tuning/ 2.8.4, 2.8.4

[65] Ultralytics, "tuner.py - ultralytics," 2024, source code. [Online]. Available: https://github.com/ultralytics/ultralytics/blob/main/ultralytics/utils/tuner.py 2.8.4

[66] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018. 2.8.4

[67] G. Pezzano, V. Ribas Ripoll, and P. Radeva, "Cole-cnn: Context-learning convolutional neural network with adaptive loss function for lung nodule segmentation," *Computer Methods and Programs in Biomedicine*, vol. 198, p. 105792, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260720316254 2.9

[68] L. Zhi, W. Jiang, S. Zhang, and T. Zhou, "Deep neural network pulmonary nodule segmentation methods for ct images: Literature review and experimental comparisons," *Computers in Biology and Medicine*, vol. 164, p. 107321, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482523007862 2.9, 5.1

[69] T. C. I. Archive, "Data from the lung image database consortium (lidc) and image database resource initiative (idri): A completed reference database of lung nodules on ct scans (lidc-idri)." 3, 3.1

[70] S. G. r. Armato, L. G., Bidaut, M. F. McNitt-Gray, C. R. Meyer, A. P. Reeves, B. Zhao, D. R. Aberle, C. I. Henschke, E. A. Hoffman, E. A. Kazerooni, H. MacMahon, E. J. Van Beeke, D. Yankelevitz, A. M. Biancardi, P. H. Bland, M. S. Brown, R. M. Engelmann, G. E. Laderach, D. Max, and B. Y. Croft, "The lung image database consortium (lidc) and image database resource initiative

(idri): a completed reference database of lung nodules on ct scans." *The Medical physics*, 2011). 3.1, 3.1, 3.1

[71] S. Gillies *et al.*, "Shapely: manipulation and analysis of geometric objects," toblerity.org, 2007–. [Online]. Available: https://github.com/Toblerity/Shapely 3.1

[72] T. M. I. T. A. (MITA), "Store (stow-rs)," NEMA. [Online]. Available: https://www.dicomstandard.org/using/dicomweb/store-stow-rs 3.2.1

[73] T. O. D. team, "dcm2pnm: Convert dicom images to pgm/ppm, png, tiff or bmp." [Online]. Available: https://support.dcmtk.org/docs-snapshot/dcm2pnm.html 3.2.1, 3.3, 3.2.1, 3.2.1

[74] G. Jocher, "How to auto-finetune hyperparameters in yolov8?" 2024, gitHub issue. [Online]. Available: https://github.com/ultralytics/ultralytics/issues/9007 3.3.1