University of Bergen
Department of Physics and Technology

Vessel recognition in ultrasound images using machine learning techniques

Author: Julia Cat-Vy Nguyen
Supervisors: Nello Blaser, Robert Hurry, Håkon Grøthe

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

June, 2023

Abstract

Purpose: Ultrasound is an imaging modality that is commonly used during cardiovascular surgeries globally. The purpose of this thesis is to investigate how machine learning techniques can be used to identify vessel properties and probe orientation in cardiac ultrasound images. The ultimate goal is developing a machine learning algorithm that can automatically recognize vessels in the region of interest with high mean average precision, identify vessel orientation, and run in near real-time.

Method: This thesis present a thoroughly data exploration of ultrasound images acquired from a multicenter study. A pilot study of three different object detection models; Yolo, RetinaNet and EfficientDet, was done to find the best model fit for the dataset in the thesis. The three object detection models were trained, tuned and evaluated on the ultrasound data. The object detection model that performed the best after the pilot study was explored further. Yolo outperformed the other models and was therefore chosen as the object detection model for the final study. To overcome the dataset's class imbalance and size problem, data augmentation, resizing and upscaling of the ultrasound images were employed. The resulting data was used to train multiple yolo models with varying hyperparameter tunings. Model selection was then performed on these trained models, and the final model was evaluated on test data.

Results: The final model achieved an overall mean average precision at 50% at 71.77%. The vessel orientation achieved a mean average precision at 64.6% for the longitudinal orientation and 75.8% for the transversal orientation. The model found it easier to locate the aorta compared to the anastomosis, which proved to be more challenging. The speed of the inference of all of these task was 5.6 milliseconds. Although the overall mean average precision was lower than the objective in this thesis, the model excelled in terms of speed.

Conclusion: In conclusion, this thesis explored the application of machine learning techniques on ultrasound data for vessel recognition and orientation. Although the final model did not improve the state of the art, the research from this master thesis can serve as a starting point for future reasearch in the field. It represents pioneering work in utilizing a multicenter dataset for machine learning on ultrasound images, providing valuable groundwork and shedding light on the feasibility and potential of machine learning in intraoperative ultrasound.

Acknowledgements

I would like to express my gratitude to my supervisor Nello Blaser for his invaluable guidance, discussions, problem-solving, support and feedback throughout the last year writing this thesis. His expertise and dedication have been a constant source of inspiration and motivation for me. Thank you for being so patient and helpful during this project.

I am also grateful to my co-supervisors Håkon Grøthe and Robert Hurry for their valuable contributions to my thesis. Their insights and support have been important in shaping my research. I would like to extend my appreciation to the innovation team at Medistim for providing me with the opportunity to work with their ultrasound systems and their research data during my two summers working there. I am grateful for the knowledge and experience gained, and for the team's support and encouragement.

I would like to thank my dear friend and classmate, Linnea Frang, for our insightful discussions, collaborative efforts, support and laughs throughout the five years of my study. I would also like to acknowledge my fellow students at Medical Technology for the memorable experience, knowledge and growth that we have shared together the last five years. I am excited to see where everyone's journey takes them after graduation.

Finally, I would express my heartfelt thanks to my family, partner and friends for their endless support, motivation and encouragement throughout the last five years, and especially throughout my thesis.

Julia Cat-Vy Nguyen

Bergen, June 2023

Contents

List of Acronyms and Abbreviations

AI Artificial Intelligence.

AP Average precision.

BiFPN Bidirectional feature pyramid network.

CABG Coronary Artery Bypass.

CNN Convolutional Neural Network.

ESVH Endoscopic saphenous vein harvesting.

FN false negative.

FP false positive.

FPN Feature Pyramid Network.

HFUS High Frequency Ultrasound Imaging.

IoU Intersection over Union.

mAP Mean average precision.

ONCAB On-pump coronary artery bypass surgery.

OPCAB Off-pump coronary artery bypass surgery.

REQUEST The Registry of Quality Assessment with Ultrasound Imaging and Transit-time Flow Measurement in Cardiac Bypass Surgery.

RoI Region of Interest.

SiLU Sigmoid-Weighted Linear Unit.

TECAB Totally endoscopic robotically assisted coronary artery bypass grafting.

TN true negative.

TP true positive.

TTFM Transit Time Flow Measurement.

Yolo You Only Look Once.

List of Figures

List of Tables

Chapter 1

Introduction

Cardiovascular diseases are a leading cause of death globally [52]. It is estimated that around 17.9 million people die each year with a cause of death attributed to a cardiovascular disease, accounting for 31% of all deaths worldwide [52]. One treatment option for advanced cardiovascular disease is (Coronary Artery Bypass (CABG)) surgery, which can be life-saving. According to a study by Roth et al., over five million CABG surgeries (see section 2.1) were performed worldwide in 2017 [59]

During a CABG surgery, ultrasound is commonly used to monitor the procedure in real-time. Ultrasound has become one of the core diagnostic imaging modalities because of its several advantages over other medical imaging methods [5]. Ultrasound is a low-cost, non-ionizing, real-time imaging modality and more cost-effective than popular modalities like computed tomography (CT) and magnetic resonance imaging (MRI). The advantages make ultrasound a popular choice for particular purposes like intra-operative surgeries. Even though this is the case, ultrasound also presents unique challenges. These include operator skill dependence, noise, and artifacts. [5]. Automated ultrasound imaging could play a crucial role in addressing some of these challenges.

Medistim ASA is a company working with intra-operative open surgery ultrasound imaging. Medical professionals use the ultrasound systems that Medistim provides to reduce risk and enhance the quality of cardiac, vascular, and transplant surgery. Today, most systems are operated by nurses or other medical professionals in the operation room (OR), helping the surgeon. A long-term goal of Medistim is to make the ultrasound systems more surgeon-directed and not dependent on other medical professionals to control the system. This is to increase the effectiveness of their ultrasound systems and minimize communication errors between the surgeon and the operator of the system.

In recent years, there has been a growing interest in the use of machine learning in ultrasound systems for various applications. Artificial Intelligence (AI), and more specifically, supervised machine learning algorithms, enable systems to learn and improve from experience without being explicitly programmed. Combining machine learning techniques with Medistim's existing products could increase the effectiveness of their ultrasound systems. This will also be an important step towards a surgeon-directed system. The main purpose is to assist users in using the ultrasound modality by automatically identifying properties of the vessels in the Region of Interest (RoI) being scanned, thereby adjusting application settings accordingly to best match the specific usage of the system. This study will therefore primarily focus on the identifying vessel properties in ultrasound images in a cardiac setting using machine learning techniques based on existing clinical ultrasound data acquired using the existing system.

1.1 Objective

Broadly speaking, the aim of this thesis is to investigate how machine learning could be used on existing intra-operative ultrasound data, contribute to vessel recognition on the ultrasound images, and identify vessel properties and vessel orientation indicating the RoI. Exploration of classification and object detection techniques is essential to complete this goal.

An article published in December 2022 using ultrasound video frames with object detection techniques achieved a mean average precision of 82.10% [18] for vessel recognition in ultrasound (see section 1.3). To improve the current state of the art, the machine learning model in this thesis has to have a Mean average precision (mAP) over 82.10%. In addition to this objective, a successful machine learning model that is used for clinical practice should have a mean average precision of more than 95%, according to domain experts. The model has to be general enough to be used in different medical scenarios but still be precise. Another important requirement for the machine learning model is speed. The model has to be able to run in near real-time to be useful as operator assistance.

The overall goal of the thesis is thereby finding an algorithm that can automatically identify the properties of the vessels on the ultrasound images. More specifically, the objective of this thesis is to develop a machine learning algorithm with the following properties:

- Improve the current state of the art mean average precision for object detection of vessel recognition in ultrasound with 82.10%
- Recognize vessels in RoI with 95% mean average precision to make it feasible to implement in clinical practice.
- Identify vessel orientation with 80% mean average precision in both transversal and longitudinal directions.
- Solve these tasks in less than 10 milliseconds.

1.2 Structure of the thesis

The thesis starts reviewing some of the field's past work. Chapter 2 continues by going through the central heart structure in the data used, as well as explaining the physics behind medical ultrasound. This is key to understanding how the images are created and which artifacts and elements are important to remember while working with medical ultrasound data. Because this thesis uses machine learning to increase the effectiveness of ultrasound systems, the principles of supervised machine learning and deep learning are described in section 2.3. This summarizes the important background information that the methods used in this thesis require.

Chapter 3 consists of pipelines, including data acquisition and different network-based solutions. The object detection algorithms described in this thesis are Yolo, RetinaNet and EfficientDet. Chapter 4 will present the results, while chapter 5 discusses and concludes the thesis as well as proposing some further work.

1.3 Related work

In medical imaging, machine learning for CT and MRI images have been in primary focus for the past few years, leaving ultrasound slightly behind. Nevertheless, machine learning in ultrasound is at an early stage but is rapidly progressing. A review article written by Brattain et al. [5] is focused on research in ultrasound where machine learning, particularly deep learning, has been in center. Table 1 in [5] shows that until 2018, 1 paper about segmentation and 8 papers about classification around the heart with machine learning were published. The paper about segmentation [6] presented a new supervised learning model designed for automatic segmentation of the left ventricle of the heart in

ultrasound images. The issues with needing a large annotated training set and a complex search process were also addressed. According to the results in the article [6], the model was robust to the training set and showed a reduction in the search complexity.

Six of the eight articles about classification around the heart with machine learning mentioned in [5] were centered around the fetal heart. Four of the articles, [3], [34], [7], and [65], were about localization of the fetal heart and detecting a standard scan plane, which is the cross-sectional view of the body that is being imaged. Two articles [3], [34], used deep Convolutional Neural Network (CNN) to create automated systems to find the scan planes. In one article [7], transfer learning with recurrent neural network was used, while another article [65] used fully convolutional networks. One of the articles [76] used guided random forest to identify critical fetal anatomy on the ultrasound images. The last article about the fetal heart was about transfer learning on object detection with videos from childbirth [29]. Natural images were used on a CNN to initialize the network to see if this would enhance the model used on ultrasound data afterward and reduce overfitting.

The two other classification articles were about classification of echocardiography videos. One article [33] used a discriminative learning dictionary to automatically classify standard echocardiogram views. The other article [28] used CNN to classify viewpoint classes in echocardiography automatically. All of these studies have promising results, when using machine learning on ultrasound data. This indicates that machine learning techniques can be implemented on ultrasound images and have potential in automating different processes.

In 2016 Erik Smistad published an article about vessel detection in Ultrasound images using deep convolutional neural networks [62]. Ultrasound images from the femoral region were used, and the method was also validated on a dataset of carotid arteries. The method from the article, performed an ellipse fitting at each pixel of the image, to create a real-time vessel detection. A simplified AlexNet network was used as the deep convolutional neural network classifier, and the vessel candidate search was performed on every fourth pixel. The average accuracy of this method was achieved at 94.5% [62]. In the article written by Smistad, an alternative to the proposed ellipse fitting method was proposed for further work. General object detection, such as the method used in this master thesis, was mentioned as part of this.

A problem with machine learning in ultrasound is the number of patients and images available for training and testing. Another common problem in the articles described in

[5] is how the ultrasound images are often collected at a single location by a single type of ultrasound device.

In a recent work by Iriani et al [18], object detection of nine different fetal heart substructures in ultrasound videos were tested. The machine learning framework You Only Look Once (Yolo), with version Yolov7, was used on 60 fetal echocardiography videos. The results achieved the mean average precision at 82.10%, with 17 frames per second for the nine substructures in 0.3 ms. The main finding from this study was that the machine learning model Yolov7 could detect fetal substructures even with a small dataset. In this thesis, the data used to train and test the machine learning model is from The Registry of Quality Assessment with Ultrasound Imaging and Transit-time Flow Measurement in Cardiac Bypass Surgery (REQUEST) study. This study is a multicenter study was done a few years back, with over 13 000 images from 12 000 videos.

Table 1.1 contains information about the task, method, number of images, and performance of the articles mentioned in this section.

Table 1.1: The different tasks, methods, number of ultrasound images used and the reference to the articles mentioned in Related Work. Abbreviations for the methods translates to: Convolutional Neural Network (CNN), Transferred Recurrent Neural Network (T-RNN) and Fully Convolutional Neural Network (FCN)

Task	**Method**	**# Images**	**Performance**	**Cite**
Segmentation	Deep Belief Networks	400	35.2% error rate	[6]
Classification	CNN	1003	71% accuracy	[3]
Classification	CNN	7568	38.7% accuracy	[34]
Classification	T-RNN	12343	90.8 % accuracy	[7]
Classification	FCN	12343	23.48% error rate	[65]
Classification	Guided Random Forest	29858	91% accuracy	[76]
Classification	CNN	10820	91.5% accuracy	[29]
Classification	Cuboid Detector	309	95% accuracy	[33]
Classification	CNN	432	89.4% accuracy	[28]
Classification	AlexNet	12 804	94.5% accuracy	[62]
Object Detection	Yolov7	60 videos	82.10% mAP	[18]

Chapter 2

Background

2.1 The human heart and CABG procedure

The human heart is the organ that pumps blood throughout the human body through the vessels in the circulatory system, also called the cardiovascular system [51]. The cardiovascular system consists of the *heart*, which is a muscular organ, *vessels*, which consist of arteries, veins and capillaries, and the *blood* [1]. Cardio stands for the heart, while vascular refers to blood vessels. The blood flow from the cardiovascular system supplies the tissues with oxygen and nutrients. Another important task of the cardiovascular system is to remove carbon dioxide and other waste from the cells and organs and carrying de-oxygenated blood back to the lungs. This is important to keep the cells in the human body healthy, and keep us alive [1].

The heart is a four-chambered double pump situated between the two lungs in the center of the chest in the human body, slightly to the left [51]. The heart is divided down the middle into a right and left heart, which in turn are divided into two chambers, as shown in fig. 2.1. The upper chambers is called an atrium, while the lower chambers is called a ventricle [51]. Low-oxygen blood from the right ventricle is sent to the lungs. The blood cells pick up oxygen in the lungs, and the blood is then carried to the heart's left atrium. The left atrium sends the oxygenated blood to the left ventricle, which is the muscular part of the heart. The blood is pumped out of the heart through arteries [63].

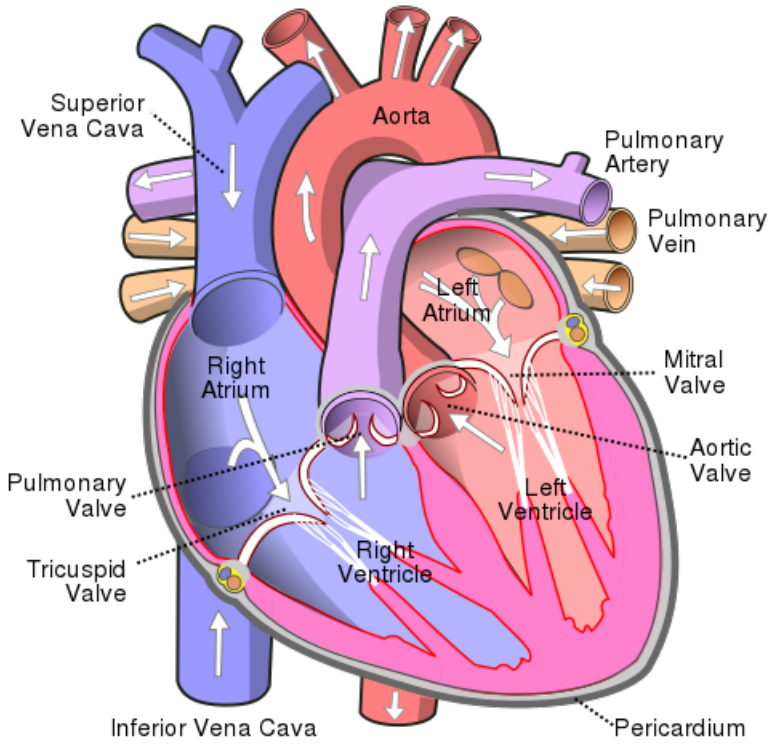


Figure 2.1: This is a diagram of the anatomy of the human heart. The image shows the four different chambers in the heart, as well as the main arteries and veins [11].

Credit: Image from [11], reproduced from public domain.

The heart muscle is supplied blood through the left and right coronary arteries. These are the first branches of the aorta, the body's main artery. The coronary arteries are both wrapped around the outside of the heart and inside the heart muscle. The left and right coronary arteries contain smaller branches, which dive into the heart muscle to bring blood to the heart [63]. Any form of coronary artery disease can reduce the supply of nutrients and the oxygen flow to the heart muscle [63]. An example of a coronary artery disease is plaque buildup inside the arteries. This can lead to clogged arteries and prevent blood from getting to the heart, leading to a heart attack.

CABG grafting has been a central procedure established at the end of the 1960s [72]. This surgical procedure is used to treat coronary heart diseases by diverting blood around narrowed or clogged parts of the major arteries. The surgery aims to improve the blood flow and oxygen supply to the heart [72]. If the heart has a deficiency of oxygen-rich blood, the deficiency can cause a coronary heart disease and chest pain can appear. This is often a warning sign that the patient could be at risk of a heart attack. CABG is one of the procedures recommended for selected patient groups, especially patients with

multi-vessel diseases, to reduce the chance of having a heart attack from coronary heart disease.

The CABG surgery involves taking a blood vessel from another part of the body and attaching it to the coronary artery above and below the narrowed area or blockage [49]. The blood vessels can either be taken from inside of the chest from the vessels called internal mammary arteries, from the leg from the vessels called saphenous veins, or from the arm from the radial arteries [49], as fig. 2.2. These areas are chosen because the other blood vessels in these areas are able to compensate for the loss of the blood vessels.

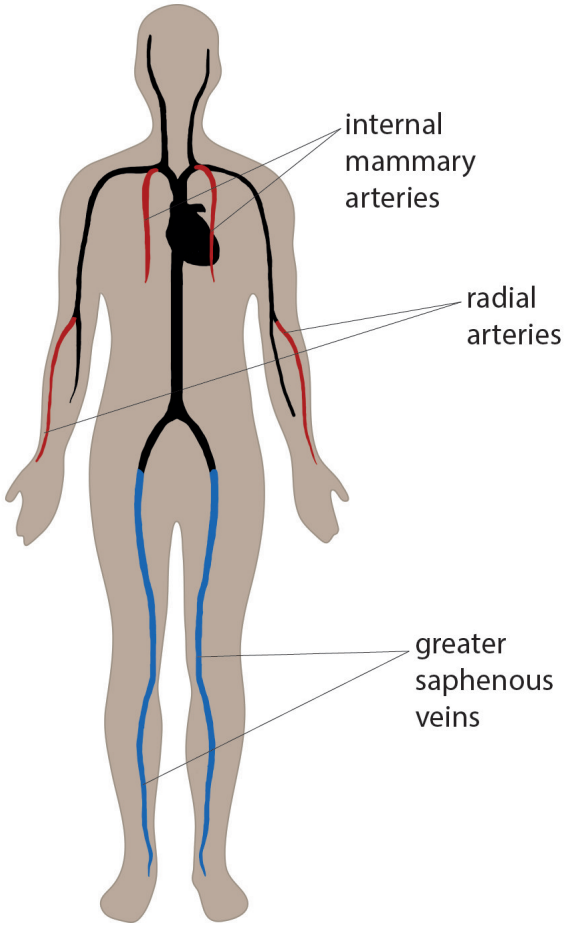


Figure 2.2: The blood vessels used in the CABG procedure are taken either from the internal mammary arteries, radial arteries or the greater saphenous veins [49].

Credit: Image from [49], reproduced from public domain.

The new blood vessel is called a graft, and the procedure can include several grafts. The surgical connection between the blood vessels is called an anastomosis. The procedure

includes different techniques along with the different configurations of where the blood vessels are taken from [49]. The four main techniques are Off-pump coronary artery bypass surgery (OPCAB), On-pump coronary artery bypass surgery (ONCAB), Endoscopic saphenous vein harvesting (ESVH), and Totally endoscopic robotically assisted coronary artery bypass grafting (TECAB). During OPCAB, a heart-lung bypass machine is not used, and the heart still beats during the surgery [49]. ONCAB is when a heart-lung machine is used during the surgery. The ESVH procedure is a less invasive method of using the veins from the legs. Keyhole surgery is done to access the saphenous vein instead of making a large cut in the leg [49]. The last procedure, TECAB, is a keyhole method of heart bypass [49]. The surgeon deflates the lungs and makes small cuts between the ribs. Using robotic arms controlled by the surgeon, the TECAB procedure can be more effective and lower the wound infection rate. This is however, a newer technique, only carried out on a small number of people [49].

The CABG procedure creates new paths for the blood to flow from the aorta through the coronary arteries around the blockage. Ultrasound is used on the graft to measure the blood flow with Transit Time Flow Measurement (TTFM), and the imaging module is used to look at the graft. The medical images used in this thesis are from a study done by Medistim on CABG procedures, and they are therefore images of the aorta, anastomosis, coronary arteries and conduit vessels.

2.2 Ultrasound

Ultrasound is one of the core diagnostic imaging modalities in medicine. It is a non-invasive way of visualizing different structures of tissues in the body with high frequency sound waves. Ultrasound images are real-time images that can capture the movement of the body's internal organs, and the blood flow through the vessels. The probes used in ultrasound are called transducers, which create the high frequency sound waves [31]. Ultrasound can be used by placing the probes on the skin or inside the body. However, the images used in this thesis are taken by ultrasound during surgery. This is done by placing a sterile probe into the area that is being operated. The Region of Interest (RoI) on the images are often vessels, anastomosis, or other anatomic structures.

The main advantages of ultrasound are that it is very accessible and relatively cheap in comparison to other imaging modalities. Diagnostic ultrasound images are very good on soft tissue, and there are no side effects with the use of ultrasound. There are,

however, some disadvantages with ultrasound as well. Ultrasound is very dependent on the operator, and the images can be hard to interpret. A high level of understanding is required to perform and interpret ultrasound images. Even though ultrasound is good on soft tissue, it is not very good on bone or when gas appears. There are also several artifacts that can influence the ultrasound images.

2.2.1 Sound waves and acoustics in ultrasound

Sound waves are longitudinal waves, where the particles in the medium move back and forth parallel to the direction the wave travels [43]. Ultrasound waves are waves with a frequency of over 20 000 Hz. These are sound waves outside of humans' audible range of sound and will spread out in liquid and soft tissue in the body. The transducer converts electrical impulses to mechanical vibrations that propagate through the tissue with microscopic movements. Medical ultrasound uses waves with frequencies around 2-20 MHz [71]. The velocity of the sound is dependent on the material the wave is propagating through. The density ρ and rigidity κ determine the sound velocity in the material through the relationship

$$c = \sqrt{\frac{\kappa}{\rho}}, \tag{2.1}$$

where c is the sound velocity. The sound velocity in tissue is given as 1540 m/s relative to the velocity of sound in bones, which is 4080 m/s [43]. The velocity of sound can also be defined with the wavelength λ and frequency f of the wave. The wavelength is proportional to the velocity of sound with the relationship

$$c = f \cdot \lambda, \tag{2.2}$$

and inversely proportional to the frequency [43]. In ultrasound, the frequency is often set by the transducer used in imaging, the wavelength is therefore mostly affected by the velocity of sound.

The intensity I of an ultrasound wave is defined by power P per unit area A. The equation

$$I = \frac{P}{A}, \tag{2.3}$$

describes this relationship. Power is the rate at which energy is transferred by the sound wave [38]. As the distance increases from the source of the sound wave, the energy from the wave will spread out over a larger area. This causes the intensity to decrease as the distance increase. The unit for intensity is $\frac{W}{m^2}$. The sound intensity level β of a sound measured in decibels dB,

$$\beta(dB) = 10 \cdot log_{10}\frac{I_1}{I_2} \tag{2.4}$$

is the ratio between two intensities [38]. The amplitude and intensity of an ultrasound wave decrease as the wave travels through different materials. This is known as attenuation [43]. Attenuation is the result of several mechanisms, known as absorption, reflection, refraction, and scatter when a sound wave propagates through two materials with different physical properties.

Absorption is the main cause of attenuation [43]. The intensity decrease is caused by energy transmission to the material as heat. The higher frequencies are absorbed more rapidly and the attenuation will therefore be quicker for waves with higher frequencies.

The difference in acoustic impedance between two materials can also cause attenuation. Acoustic impedance is a physical property of tissue and describes how much resistance an ultrasound beam encounters as it passes through a tissue [48]. Acoustic impedance Z is dependent on the density ρ and the velocity of sound c with the equation

$$Z = \rho \cdot c = \sqrt{\rho\kappa}. \tag{2.5}$$

Two materials with different densities or velocities of sound will cause a reflection because of the difference in acoustic impedance. This is the reason why ultrasound is not good at imaging bones. Bones have a high density, which makes most of the sound wave reflected. This can cause artifacts in the ultrasound image (see section 2.2.4). The

reflection coefficient R is described with the acoustic impedance to two different materials [43], Z_1 and Z_2, with the equation

$$R = (\frac{Z_2 - Z_1}{Z_2 + Z_1})^2. \tag{2.6}$$

The amount of energy reflected depends on the difference in the acoustic impedance of the tissues. This is called acoustic impedance mismatch. The difference in acoustic impedance is also why a gel is often used in ultrasound. The velocity of sound in air is 340 m/s which is different from the speed of sound in the tissue. The gel minimizes the acoustic impedance difference and excludes the air between the ultrasound transducer and the skin.

The remaining ultrasound wave that is not reflected will transmit further into the tissue. When the wave passes through the following material, the velocity of the wave changes. This causes a refraction, also known as a bending of the ultrasound wave [43], as shown in fig. 2.3. The refraction angle depends on the wave's velocity change.

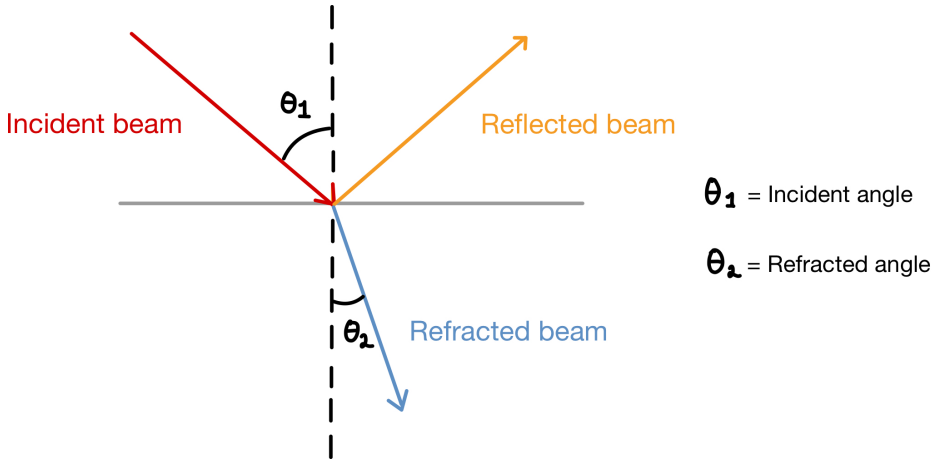


Figure 2.3: The angle of refraction depends on the velocity change of the wave.

Credit: The image was drawn by the author.

Scattering of an ultrasound wave occurs when the sound wave interacts with objects smaller than a wavelength. These objects are called "diffuse reflectors" [43], an example being red blood cells. Since the objects are smaller than a wavelength, most of the waves will not interact with the object, and the ultrasound wave will be scattered. This can cause artifacts in the ultrasound image, called speckles (see section 2.2.4).

2.2.2 Imaging formation in ultrasound

The probes used in ultrasound are called transducers. These produce the high frequency ultrasound waves and register the reflected waves used to form the ultrasound image. The transducer is made up of piezoelectric material. The material can be natural crystals, like quartz, or artificial crystals, like lead zirconate titanate (PZT) [4].

The piezoelectric material has two essential properties that are very useful in ultrasound; The first property called the piezoelectric effect, is when mechanical stress is applied to the material. The mechanical stress is the return of the ultrasound waves that hit the piezoelectric material. When this happens, the material starts to vibrate, causing generation of electrical current, which is possible to measure [71]. This is shown in the first box in fig. 2.4.

The second property called the inverse piezoelectric effect, is when an electrical field is present near the piezoelectric material. The electrical field causes the material to vibrate and thus generate an ultrasound wave. The electrical energy from the electrical field is converted to mechanical energy, which causes a sound wave to propagate into the tissue. This is shown in the second box in fig. 2.4, where the vibration makes the displacement of the piezoelectric material. The piezoelectric material can therefore be both the transmitter and the receiver of the ultrasound signals [71].

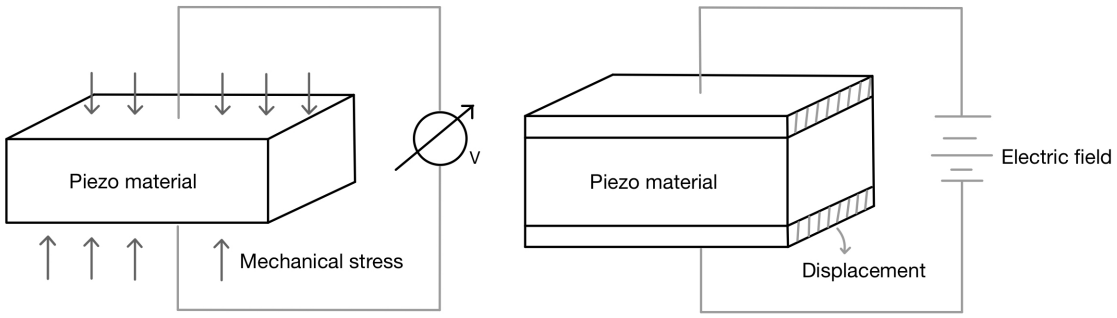


Figure 2.4: The piezoelectric material have two important properties that are very useful in ultrasound: the piezoelectric effect and the inverse piezoelectric effect.

Credit: The image was drawn by the author.

The thickness of the piezoelectric element determines the resonance frequency of the crystal. The thickness will be equal to half the wavelength of the desired frequency.

A thicker element will produce a lower frequency, while a thinner element will produce a higher frequency [4]. The piezoelectric material comprises polarized crystals, with a positive electrode in the back of the element and a negative electrode in the front of the element. This allows for an electrical connection when mechanical stress is applied to the element [4]. The transducer may consist of a single piezoelectric element or a band of multiple elements.

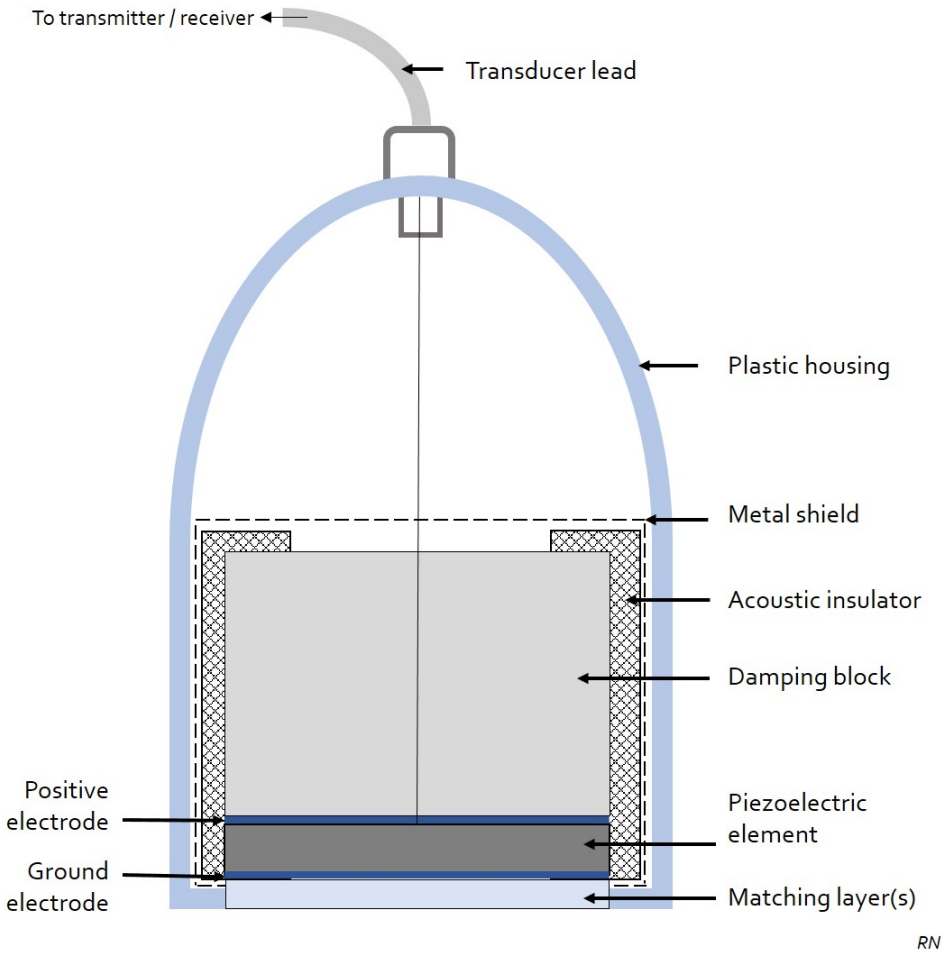


Figure 2.5: This image shows an example of an ultrasonic transducer from [4].

Case courtesy of Dr Rachael Nightingale, Radiopaedia.org, rID: 54040 from [4]

The image in fig. 2.5 shows an example of an ultrasonic transducer. The blue lines around the piezoelectric element are the positive and negative electrodes that allow the electrical signals to form. At the bottom of the transducer, a matching layer is shown. This is the part of the transducer touching the tissue and is therefore a layer to increase the acoustic coupling between the tissue and the element. The matching layer is made of materials with acoustic impedance between the soft tissue and the transducer. This can be one or

multiple layers, and each layer is one-quarter wavelength thick at a chosen frequency [4]. As the image in fig. 2.5 shows, a damping block is used in the transducer as backing. This is to dampen the wave released from the element's backside. If the damping block had not been there, this wave would have been reflected back and forth [4]. The image also shows an acoustic insulator, which stops the transducer from vibrating in the hand.

The ultrasound waves are produced when an electric current is applied to the piezoelectric crystals. This causes mechanical stress on the crystals, producing the sound wave. The ultrasound waves are produced in pulses, where the pulse length is the distance each pulse travels. The pulse repetition frequency is how often the transducers send out the pulses. The transducer emits around 1000 pulses each second, but each pulse is only around one microsecond. The rest of the time between each pulse, the transducer acts as a receiver [71]. At the top of the transducer, a cable that leads the electrical signal to the transmitter is shown in fig. 2.5. When a returning echo sound wave reaches the crystals, the piezoelectric crystals start to vibrate, causing the production of the electric signal that can be measured. The intensity of the sound wave is proportional to the amount of voltage produced. This can therefore be used to produce different ultrasound images.

It is possible to form images from different angles with the ultrasound transducer. The ultrasound waves created from one point will spread out as circles in the water [71]. This is called spherical waves. When placing multiple sources beside each other, a wavefront is created. This principle is called the Huygens principle. If multiple piezoelectric elements are used, and the elements are activated one by one with a time interval in between, the wavefront will be angled when propagated through the tissue. This is shown in the image in fig. 2.6 below.

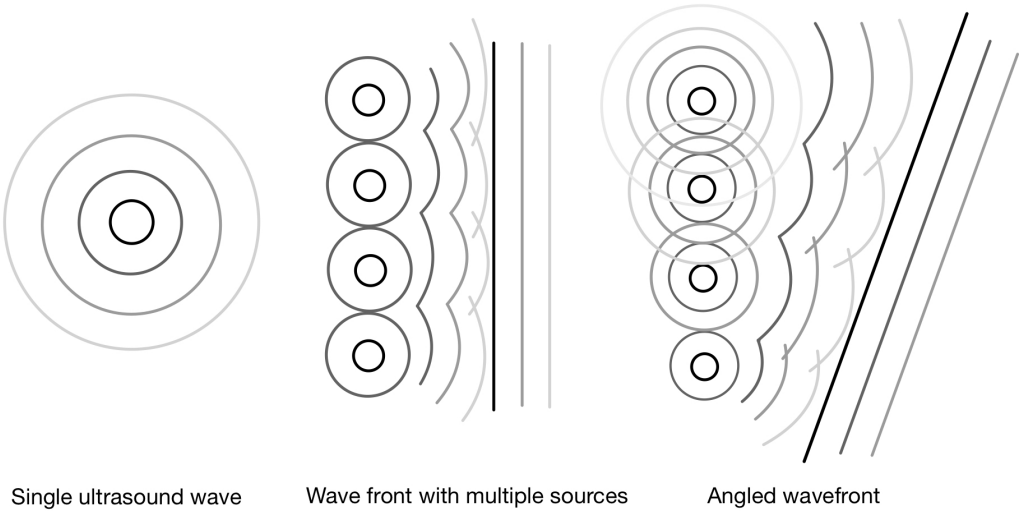


Figure 2.6: By activating multiple wave sources with a time interval in between, an angled wave front is created.

Credit: The image was drawn by the author, with inspiration from [71]

Imaging resolution determines the clarity of the ultrasound image. Resolution in ultrasound is dependent on the frequency and the pulse length. A higher frequency with a shorter pulse length will give better resolution than a lower frequency. When the frequency is high, the attenuation will be high, and the resolution will be high. Low frequencies will however, penetrate further into the tissue. One must therefore find a compromise between the desired resolution and the ability to penetrate deeper into the tissue.

Using different imaging probes in ultrasound creates a varying scattering of the ultrasound beam, which leads to different properties for each probe. The images in this master thesis are taken with a linear probe. A linear probe produces ultrasound waves in straight parallel lines [42]. The probe has high frequencies, which provides better resolution and less penetration. The probe is often ideal for imaging superficial structures and in ultrasound guided procedures [42]. In this thesis, the linear probe L15 - High-frequency Ultrasound Imaging Probe from Medistim was used for all the images. The probe operates at frequencies from 8 - 18 MHz, which allows for extreme near-field resolution and is approved for direct contact with the cardiac tissue [2].

2.2.3 Different ultrasound modalities

Ultrasound imaging is often divided into different modalities; A-mode, B-mode, and M-mode [44]. Fig. 2.7 shows examples of the three different modalities. A-mode is called the amplitude modality and is the simplest form of displaying the received echo. This is imaging in one dimension, and the echoes are represented with the amplitude of peaks indicating the signal strength. The height of the peaks depends on the strength of the signal.

B-mode is the brightness modality, where the echoes are displayed as pixels, with the brightness of each pixel corresponding to the strength of the reflected signal [44]. Each of the lines in the B-mode image is an A-mode line. The ultrasound images with B-mode show a slice of the tissue and is the most used modality in medical ultrasound to visualize anatomical features. If the scan rate is high enough, real-time imaging is possible with the B-mode. The images used in this master thesis are B-mode images with anatomical structures.

M-mode is the motion modality. The M-mode image is the motion from one of the A-mode lines in the B-mode image [44]. Instead of peaks indicating the signal strength, the echoes are represented with a vertical line of motion as a function of time. The advantage of this modality is that the time resolution is very good, making it helpful in studying movement.

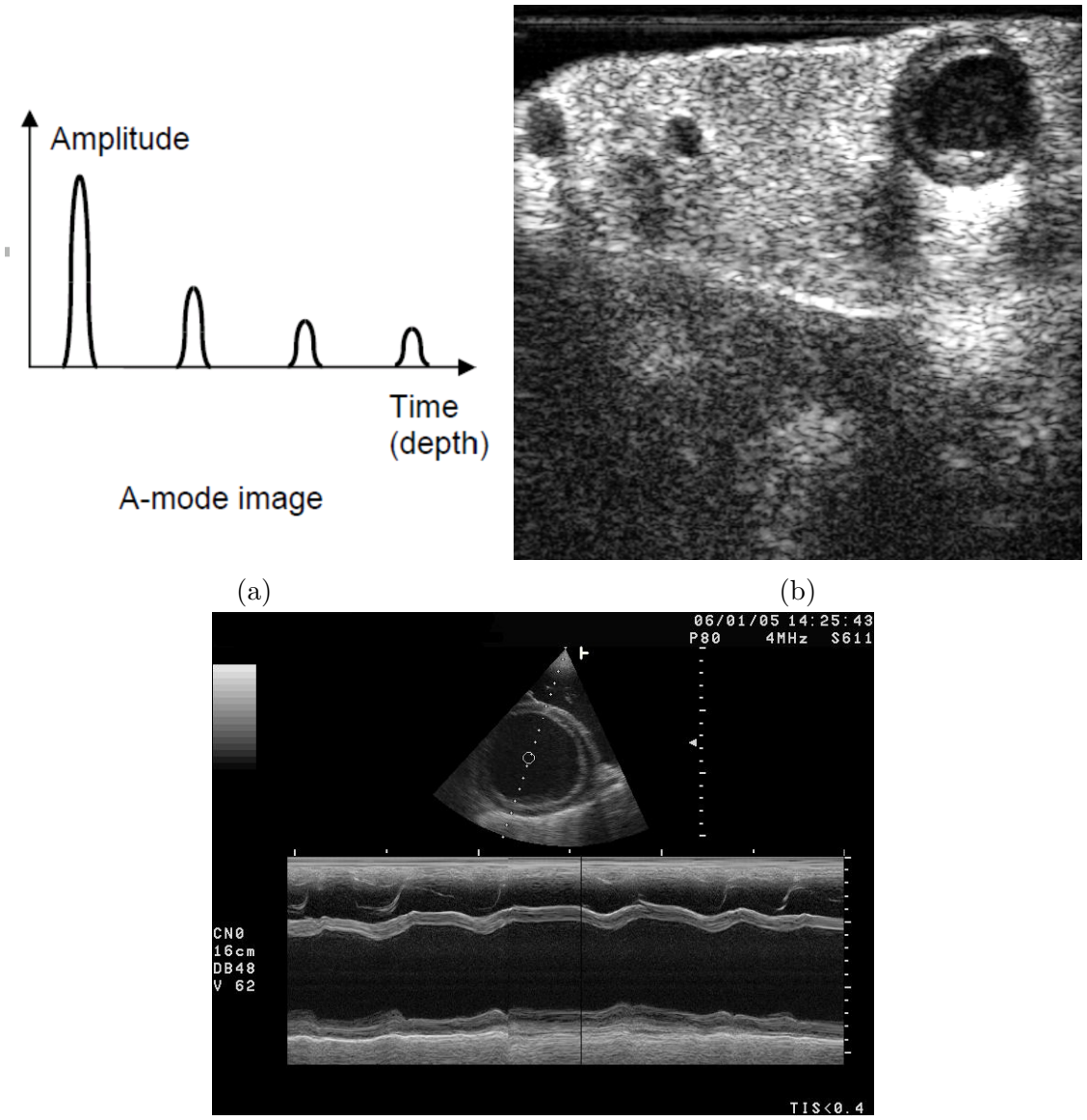


Figure 2.7: Different examples for the modalities in Ultrasound mentioned above. Image (a) is an example of an A-mode image which is the display of amplitude spikes in the vertical axis, and time required for the return of the echo in the horizontal axis. Image (b) is an example of B-mode, with the two-dimensional map of the data. Image (c) is an example of M-mode, with a B-mode image displayed over. The M-mode captures the returning echoes in one line of the B-mode image displayed over time.

Credit: (a) from [12] and (c) from [10] reproduced from public domain, (b) REQUEST study from Medistim.

2.2.4 Artifacts

As mentioned in the introduction, ultrasound imaging presents a few unique challenges. One of these is identifying and reducing the artifacts that occur during scanning.

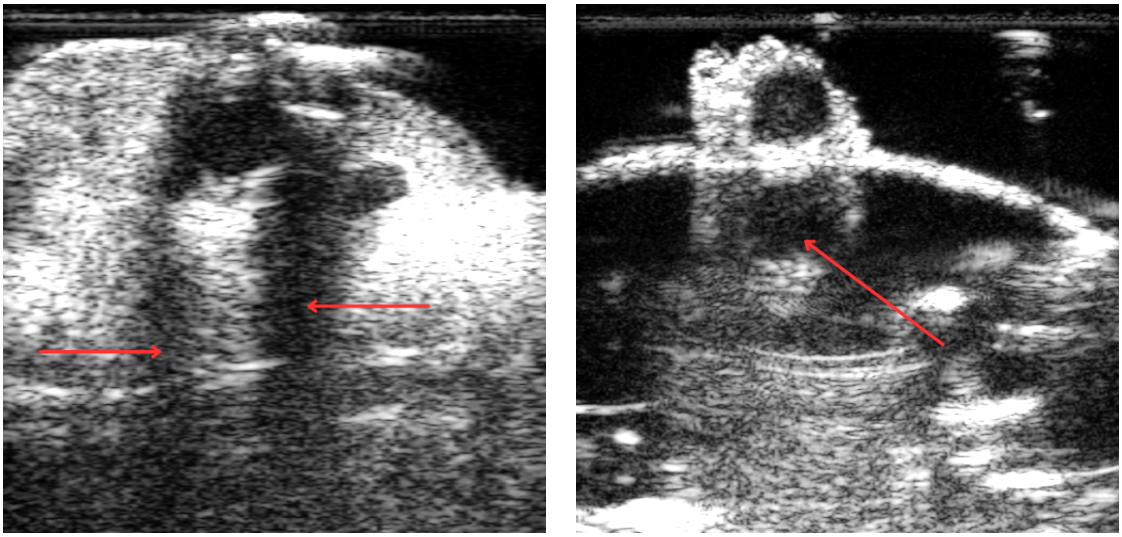
One common artifact in ultrasound is called speckle [71]. This occurs when sound waves are reflected in different directions, resulting in spots or specks appearing in the ultrasound image. Speckle is an artifact that is more influenced by the imaging system rather than the object being imaged. Speckle arises from differences in interference between waves and resolution. Although speckle reduces the imaging contrast, it can be useful for observing tissue motion.

While ultrasound is effective for visualizing soft tissue, it encounters difficulties when the ultrasound wave encounters materials with significantly different acoustic impedance than soft tissue. The reflected wave becomes very strong in such cases, causing ultrasound attenuation. Because of attenuation, a shadow can appear behind the structure in the ultrasound image. This artifact is referred to as the attenuation effect or shadowing [44]. The attenuation effect or shadowing is commonly observed with materials like gas and bone.

Reverberation is another common artifact in ultrasound images. Reverberation occurs when a reflected pulse strikes a highly reflective surface, such as the skin or the ultrasound probe itself [44]. The ultrasound wave then re-enters the tissue, creating a deeper echo that does not correspond to the anatomical features. This is one of the reasons why a dampening block is used on the transducer, as explained in fig. 2.5.

Similar to reverberation, the mirror artifact occurs when multiple ultrasound beams are reflected between an organ in the body and a strongly reflective surface [44]. This artifact appears as a virtual mirrored object.

In the B-mode modality, a common artifact known as side lobes can occur. This happens when anatomical structures outside of the main ultrasound beam are mistakenly mapped into the main beam [44].



(a) (b)

Figure 2.8: This is two images with examples of some of the artifacts mentioned above. Both Image (a) and (b) have a lot of structure in the image, where the structure is an example of the speckle artifact. Image (a) is also an example of attentuation artifact, where you can see a black shadow going down from the round vessel at the top. Image (b) is an example of mirroring, where you can see a vessel being reflected, appearing as a virtual object in the image.

Credit: REQUEST study from Medistim

2.3 Machine Learning and Deep Learning

Machine Learning is a type of AI where the algorithm learns and improves from input data to make future predictions [32]. The goal of machine learning is to be able to solve problems more effectively and more efficiently. Machine Learning is often divided into supervised, unsupervised, and reinforcement learning [32]. In supervised machine learning, the algorithm learns from labels or targets in the training data. In unsupervised machine learning, the input is unclassified and unlabeled. The algorithm has to uncover the patterns in the data. Reinforcement learning is a goal-oriented algorithm [32]. The method has an agent with a goal, and when action is taken, the agent receives a reward or punishment depending on the action relative to the goal. The algorithm will learn a behavior that aims to maximize the reward. Deep learning is also a subdivision of machine learning. Deep learning refers to the algorithms learning from examples by mimicking the human brain with neural networks. This section will focus on supervised machine learning and deep learning since this is used in this master thesis.

2.3.1 Fundamentals of supervised machine learning

As mentioned above, supervised machine learning is a type of machine learning where the model is trained on labeled data. This means the data has been labeled with the correct output, and the model is trained to predict the output for a given input. An example of a supervised machine learning problem is creating a model that can predict whether an image is of a human heart or brain. Suppose we have a dataset of images of hearts and brains. We would use a supervised machine learning algorithm to train the model on the labeled dataset. After training, the model could classify new images of hearts and brains based on the features it learned from the training data. For example, if we showed the model an image of a human heart with smooth surfaces, it would predict that the image contains a human heart. If we showed it an image of a brain with a dimpled texture, it would predict that the image contains a brain.

Supervised machine learning is often used for classification or regression. Classification is the process where a model predicts the most probable category, class, or label. Some examples of supervised classification algorithms include Random Forest and Support Vector Machine [32]. Regression is used to analyze the relationship between different independent variables and a numeric dependent variable or outcome.

Object detection is a machine learning task that involves identifying objects in images or videos with bounding boxes around the objects along with a label of the category of the object [45]. This can be used to detect abnormalities in ultrasound images or identify specific structures in ultrasound images, as done in this thesis. In this thesis the specific structures are vessels, aortas, or anastomosis.

The basic idea behind object detection is to train a model to recognize specific objects in the images by feeding it a large number of labeled examples. These examples typically consist of images with bounding boxes drawn around the objects of interest, and corresponding labels. The model learns to associate specific patterns with different object classes during training. These patterns are called features and are extracted with the help of deep learning algorithms such as convolution neural networks (see section 2.3.2) [45]. For example, the algorithm might learn that a particular texture in an ultrasound image corresponds to plaque or that a particular shape and intensity pattern indicates a vessel. Once the model has been trained, it can be applied to new images to identify the objects they contain.

One of the main concepts of object detection with multiple objects is anchor boxes. Anchor boxes are predefined bounding boxes that represent the ideal location, shape and size of the object it specializes in predicting [9]. They are fixed initial boundary box guesses. During the training of an object detection model, the algorithm uses these anchor boxes as references to predict the location and size of objects in the image or the RoI in the case of an ultrasound image. Instead of directly predicting the bounding boxes, the model predicts the probability and refinements corresponding to the anchor boxes.

During detection, thousands of predefined anchor boxes for each predictor are tiled across the image. For each anchor box, the Intersection over Union (IoU) (Defined precisely in section 2.3.1) is calculated with the objects bounding boxes. If the IoU is higher than a specific threshold, the anchor box detects the object with the highest IoU. If the highest IoU is less than the threshold, then the content of the anchor box is detected as no object [9].

One of the key advantages of using anchor boxes is that they allow object detection algorithms to handle objects of different sizes and aspect ratios, which is vital in real-world applications where objects can vary widely in size and shape [9]. Anchor boxes enable the algorithm to detect overlapping objects as well.

There are several methods for generating anchor boxes, including manually defining them based on knowledge of the objects in the dataset. In recent years, deep learning methods such as CNN have been used to generate anchor boxes automatically, allowing for greater flexibility and scalability in object detection algorithms.

Pipeline of a supervised machine learning model

In machine learning, a pipeline divided into three different parts is one of the fundamental concepts. The three parts are training, validation, and testing, and the model's input data is also split similarly [47]. The training and validation are part of the model selection process, while the testing is done in the model evaluation process. The training set is used to train the model, the validation set is used to evaluate the model during the training and tune the hyperparameters, and the test set is used to evaluate the final model.

The pipeline and the data are divided into three parts to prevent overfitting and underfitting and improve generalization of the machine learning model [57]. Overfitting is when a model performs well on the data it was trained on but poorly on new unseen data [57]. This can happen when a model is too complex and starts picking up on random noise in the training data instead of learning the true underlying pattern. On the other hand, underfitting is when a model needs to be more complex to capture the true underlying pattern in the data. This can lead to poor performance on the training data [57].

The aim of the training process is generalization, which means that the model should perform well on new and unseen data [32]. The bias-variance tradeoff is a fundamental concept in machine learning, and it refers to the balance between two sources of error in the model: bias and variance. Bias is the error introduced by simplifying assumptions made by the model, while variance is the error introduced by the model's sensitivity to the specific training data it was trained on [15]. In machine learning, the goal is to minimize both bias and variance, but lowering one often increases the other. In order to achieve good generalization, a model must strike a balance between bias and variance. A model with too much bias will be underfitted, while a model with too much variance will be overfitted [15]. The evaluation of the model is done on new unseen data, and not the same data that the training was done on, to ensure generalization. Fig. 2.9 shows how the bias-variance tradeoff can influence the model.

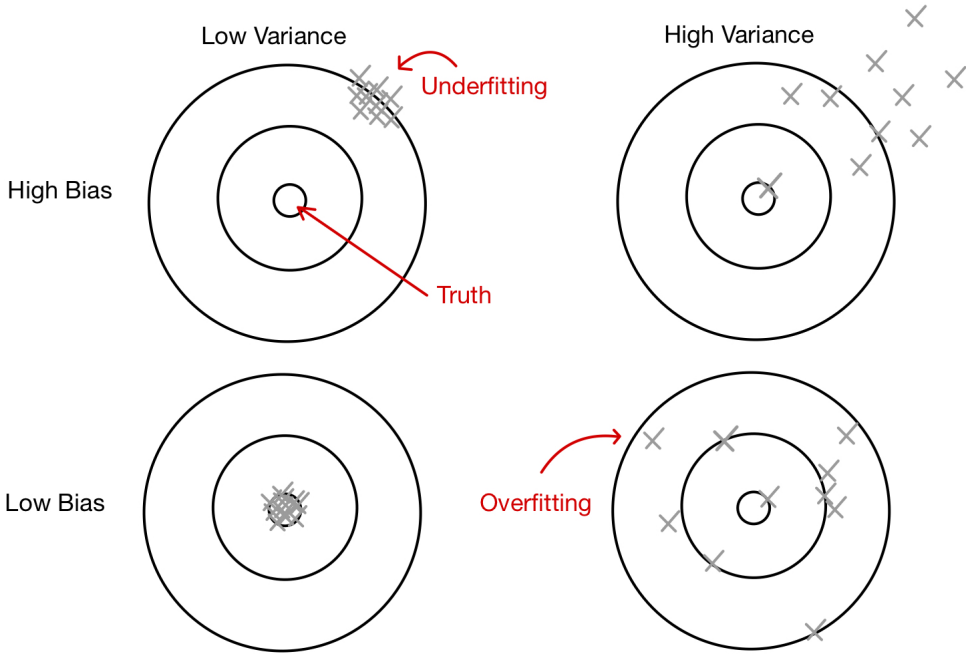


Figure 2.9: The images shows a bulls-eye diagram of bias and variance. The image show the center of the target as the ground truth, and moving away from the center gives worse predictions. [15]

Credit: Image made by author, inspired by Bill Howe UW from [15].

The no free lunch theorem is a mathematical result that shows that, in general, no machine learning algorithm can be expected to perform better than any other on all possible data sets [30]. In other words, the performance of a machine learning algorithm depends on the specific data it is trained on, and there is no "one size fits all" solution that will work for all data sets.

In light of this theorem, it is important to carefully evaluate the performance of a machine learning algorithm on the specific data set that it will be used on rather than assuming that a particular algorithm will always be the best choice. Dividing the data into training, validation, and test sets can help with this evaluation by providing a way to measure the performance of a model on unseen data. However, the no free lunch theorem tells us that even with careful evaluation, there is no guarantee that a particular machine learning algorithm will always perform well [30].

Parameters and hyperparameters

In supervised machine learning, a model is trained predict new data based on labeled training examples. Parameters are the internal values that are learned by the model

during training [47]. The model typically has one or more parameters adjusted during training to minimize the error in the training data. These parameters are often referred to as the model's weights [47]. Hyperparameters are the external values set before training, such as the learning rate, which controls how quickly the model updates its weights [47].

Tuning the hyperparameters of a learning algorithm is an important part of the training process, and it can significantly impact the performance of the resulting model. In general, we use a validation set to evaluate the performance of different hyperparameter values and to choose the combination that produces the best performance on the validation set (see section 2.3.1). This can help ensure the model has a good bias-variance tradeoff and can generalize well to new data.

Loss function

In machine learning, the loss function is related to the prediction of the machine learning model. The loss is a number indicating how bad the models' prediction was on a single example [47]. The higher the loss output, the worse the model performs. When a model is trained on a dataset, it adjusts its parameters in order to minimize the loss. In other words, the model strives to make predictions that are as close as possible to the true values in the dataset.

There are many types of loss functions, and the appropriate loss function depends on the specific task and the model type. For example, in a classification task, where the model predicts the class of an input, a common loss function is the cross-entropy loss. In a regression task, where the model predicts a continuous value, the mean squared error (MSE) is often used as the loss function.

The mean square error loss also called the L2 loss, is measured as the average of the squared difference between predictions and actual observations [56]. Due to the square, predictions far from the actual values are penalized heavily compared to less deviated predictions. If n equals the number of samples, y_i is the correct value, and $\hat{y}_i$ is the predicted value for sample i, the equation for L2 loss is given as

$$L2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{2.7}$$

L1 loss, also called Mean Absolute Error Loss (MAE), is a loss function used for regression tasks. This is the absolute difference between the predicted and the actual values. If y

is the actual value, $\hat{y}$ is the predicted error, and n is the number of samples, the L1 loss can be described as

$$L1 = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{2.8}$$

The disadvantage of L1 loss is that the loss is not sensitive to outliers because it is an absolute value. The loss called smooth L1 loss is an answer to this problem along with the problem of the L2 loss, where the L2 loss can have an exploding gradient when the actual and the predicted values are too different. The smooth L1 loss is a version of the L1 and L2 loss that are more robust to outliers and can prevent exploding gradients [66]. The smooth L1 loss is described mathematically as

$$L1_{smooth} = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & if|y - \hat{y}| \leq \delta \\ |y - \hat{y}| - \frac{1}{2}, & otherwise \end{cases} \tag{2.9}$$

where δ is a hyperparameter to define the range for L1 and L2 loss. The intuition is to use the L1 loss when the difference between the actual value and the predicted value is small and use the L2 loss for larger errors to enforce a high penalty to decrease the impact of outliers.

Another common loss function is cross entropy loss, divided into binary and categorical cross entropy. Binary cross-entropy (BCE) is used for binary classification tasks. If we keep n as the number of samples, y as the correct value, either 0 or 1, and $\hat{y}$ as the predicted probability that the class is 1, the BCE loss can be described as

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot \log(\hat{y_i}) + (1 - y_i) \cdot \log(1 - \hat{y_i})) \tag{2.10}$$

Categorical cross entropy (CCE) loss is essentially BCE loss expanded to multiple classes[47]. The CCE loss can be described as

$$CCE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} y_{ij} \cdot \log(\hat{y_{ij}}), \tag{2.11}$$

where c is the number of classes, y_{ij} is 1 if sample i is in class j and 0 otherwise, $\hat{y_{ij}}$ is the predicted probability for sample i of class j.

The loss function is an important part of the training process for a machine learning model. By minimizing the loss function during training, the model can improve its performance and make more accurate predictions on unseen data.

Performance measures and metrics for classification

Performance measures are used in machine learning to evaluate the accuracy and effectiveness of machine learning models. There are multiple different techniques to evaluate a model in machine learning.

Some general terms for evaluating binary classification are true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP refers to the number of instances that are correctly predicted as positive by the model [54]. TN refers to the number of instances that are correctly predicted as negative by the model [54]. FP refers to the number of instances that are incorrectly predicted as positive by the model [54]. For example, if the model is a binary classifier that predicts whether an ultrasound image is of a heart or not, a false positive would be an image that is incorrectly predicted as a heart but is actually not a heart. FN refers to the number of instances that are incorrectly predicted as negative by the model [54]. For example, if the model is a binary classifier that predicts whether an ultrasound image is of a heart or not, a false negative would be an image of a heart that is incorrectly predicted as not a heart but is actually an image of a heart.

Precision and recall are two common metrics used to evaluate the performance of a machine learning model. Precision measures the model's ability to identify positive instances [54] correctly. It is calculated as the number of TP divided by the number of TP plus FP, that is

$$Precision = \frac{TP}{TP + FP} \tag{2.12}$$

In other words, it measures how often the model is correct when the prediction of an instance is positive.

Recall measures the model's ability to correctly identify all positive instances in the dataset [54]. It is calculated as the number of TP divided by the number of TP plus FN, that is

$$Recall = \frac{TP}{TP + FN}. \tag{2.13}$$

In other words, it measures how often the model can find all the positive instances in the dataset.

Ideally, when both precision and recall are high, the detected object is correct, and the model can detect all the occurrences of a class. The precision-recall curve shows the tradeoff between the precision and recall values for different thresholds [54]. The probability that the bounding box in an object detection problem belongs to a certain category and the threshold is called the confidence threshold [54]. A precision-recall curve plots the value of precision against recall for different confidence threshold values [54] and gives us a visualization of which confidence threshold is the best. An example of a precision-recall curve is shown in fig. 2.10, with precision along the y-axis and recall along the x-axis.

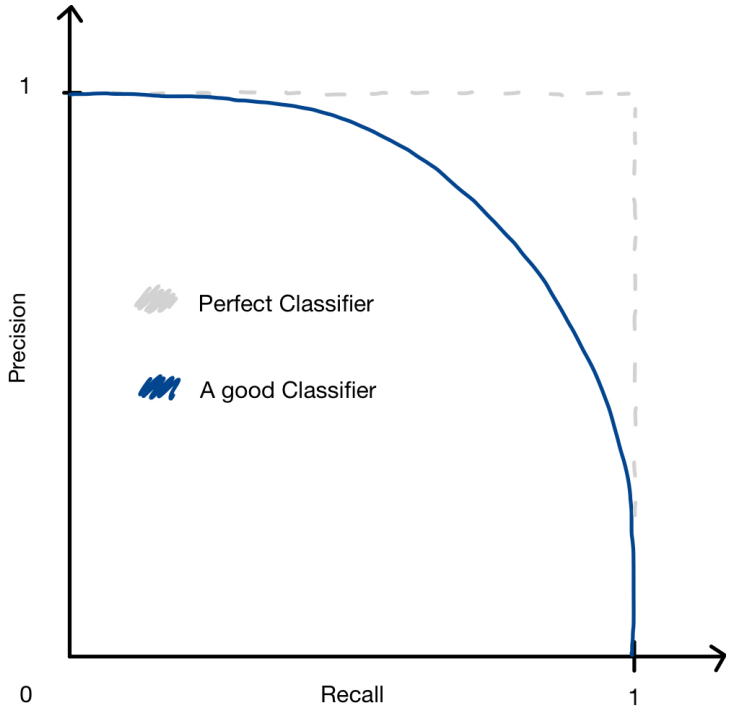


Figure 2.10: This is an example of a precision-recall curve, with precision and recall along the axis.

Credit: Image made by author

Performance measures and metrics for object detection

For object detection, average precision, Intersection over Union IoU, and mAP are common performance measures. IoU measures the overlap between the predicted bounding box and the ground truth bounding box for an object [54]. It is calculated as

$$IoU = \frac{IntersectionArea}{UnionArea}, \tag{2.14}$$

which is the area of the intersection of the two bounding boxes divided by the area of the union of the two bounding boxes, as shown in fig. 2.11. A high IoU score indicates that the predicted bounding box closely matches the ground truth bounding box, while a low IoU score indicates that the two bounding boxes do not overlap very much.

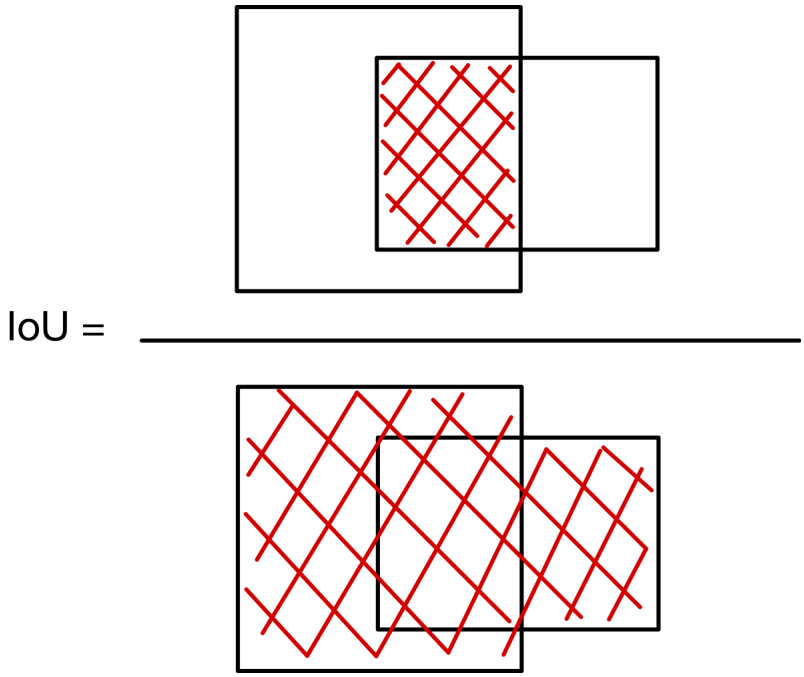


Figure 2.11: IoU is the Intersection Area over the Union Area [54].

Credit: Image made by author

Object detection often uses a loss function called Complete Intersection over Union (CIoU) loss. This is an extension of the IoU metric. The CIoU loss takes into account the distance between the predicted bounding box and the ground truth bounding box, in addition to their IoU [77]. This helps improve the model's accuracy by penalizing predictions far from the ground truth, even if they have a high IoU. If B and B^{gt} are the

predicted bounding box and ground truth bounding box, S is the overlap area denoted by $S = 1 - IoU$, D is the normalized distance IoU loss between the center point of the predicted and ground truth boxes, and V is the consistency of aspect ratio, CIoU can be described as

$$CIoU = S(B, B^{gt}) + D(B, B^{gt}) + V(B, B^{gt}). \tag{2.15}$$

Confidence scores and confidence thresholds are often used in object detection to evaluate the reliability of the predictions and classifications made by the model [54]. The confidence score is a numerical value between 0-100% that represents the model's confidence level in its prediction. The higher the value, the higher the confidence the model has about the prediction.

The confidence threshold is a value that is determined before the evaluation. This is a threshold that is used to determine whether a prediction or classification should be accepted or rejected. If the confidence score for an input is above the threshold, the model will label the prediction. If the confidence score is below the threshold, the prediction or classification is rejected as unreliable or uncertain.

Average precision (AP) is a metric calculated with the help of several other metrics, such as TP, FP, FN, precision, recall, and IoU [54]. AP is the area under the precision-recall curve, and it summarizes the precision-recall curve to one scalar value. When both precision and recall are high, the AP is also high. The AP is low when either precision or recall is low across a range of confidence threshold values. The precision-recall curve is described with the equation $p(r)$, and the AP is therefore described by

$$AP = \int_{r=0}^{1} p(r)\, dr. \tag{2.16}$$

mAP, or mean average precision, is another common performance metric in object detection tasks. It is a variant of the average precision metric, which averages the precision of the model at different recall levels [54]. If the equation for AP is eq. (2.16), then the mAP can be described with AP_k as the average precision of class k and m as the number of classes with the formula

$$mAP = \frac{1}{m} \sum_{k=1}^{m} AP_k. \tag{2.17}$$

mAP50 specifically refers to the average precision of the model at 50% recall. A high mAP50 score indicates that the model has a high average precision at 50% recall, which means that it is able to accurately detect objects in an image with a high degree of precision. A low mAP50 score, on the other hand, indicates that the model cannot accurately detect objects in an image.

In conclusion, performance measures are essential tools for evaluating the effectiveness of machine learning models. The mAP50 measure is commonly used in object detection tasks, while precision, recall, accuracy, and the confusion matrix are also important measures for evaluating the performance of machine learning models.

2.3.2 Neural networks

Deep learning is a subset of machine learning that uses multilayered neural networks to recognize complex patterns in the data. The neural networks are inspired by the structure of the human brain, specifically how the layers are connected with nodes, allowing information to flow through the network and be transformed and processed at each step.

Neurons, or nodes, are the basic units of a neural network. They are inspired by the neurons in the brain, which receive input, perform computation, and produce an output. In a neural network the input is inserted into the network's input layer. This layer has as many nodes as there are features in the dataset. After the input layer, the data is passed on to one or more hidden layers for further processing. These are the layers that find the pattern in the data. The more hidden layers the network has, the more complex patterns can be found [60]. The output of the final hidden layer is the neural network output. The number of layers is also equal to the depth of the network. A network with only one layer is called a shallow network, while a network with many layers is called a deep neural network. The neural network's prediction is based on the patterns the network learned from the training data.

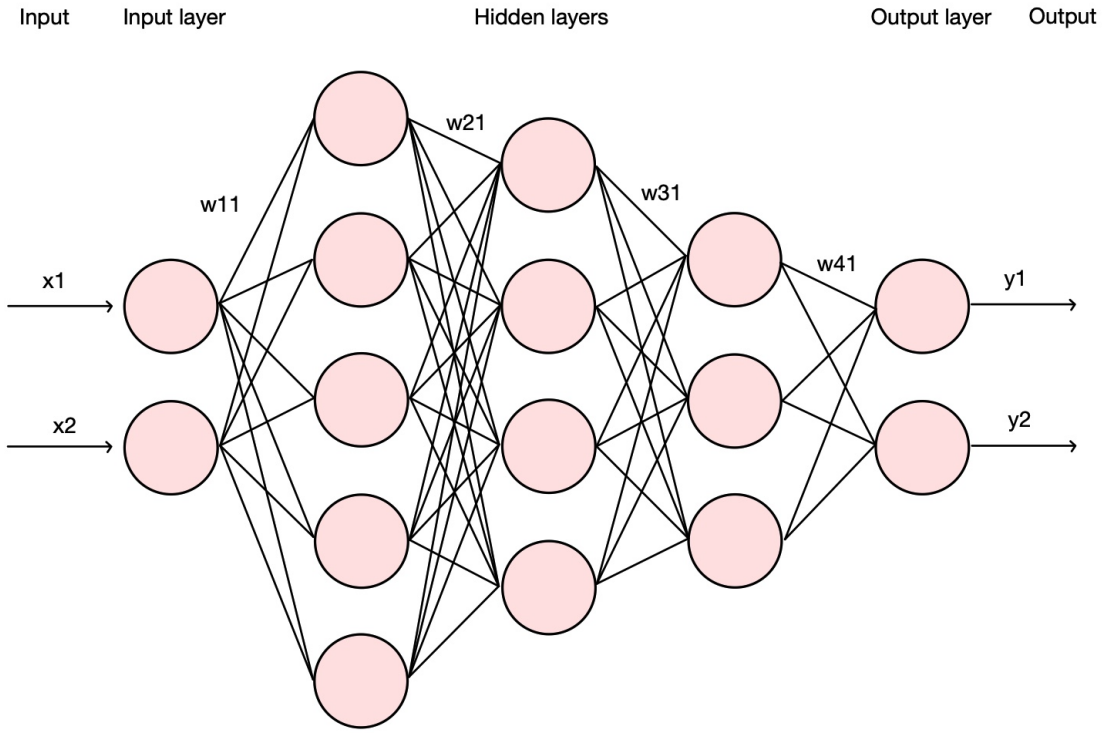


Figure 2.12: This is an illustration of a simple neural network with an input layer, hidden layers and an output layer.

Credit: Image made by author

Fig 2.12 shows a simple neural network with nodes, an input layer, hidden layers, and an output layer. In a neural network, each node receives input from the other nodes and produces an output, which is again passed to other nodes. The connection between two nodes is determined by the weights and a constant bias between the nodes [47]. The node computes the weighted sum z of the inputs. If the inputs are given as $x_1, x_2, ..., x_n$, and the weights are $w_1, w_2, ..., w_n$, the weighted sum is computed as

$$z = x_1 w_1 + x_2 w_2 + ... + x_n w_n + bias. \tag{2.18}$$

The output of the neuron is calculated using an activation function, which is a mathematical function that takes the weighted sum as input and produces an output. There are many different activation functions to choose from, and each activation function has its properties and is suitable for different types of tasks. The sigmoid activation function is a widely used function for binary classifications. It is used in the output layer of the network to convert the output of the last hidden layer into a probability distribution over

the different classes of the problem. The output values are between 0 and 1 [26]. The mathematical formula for sigmoid is given as

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}. \tag{2.19}$$

Sigmoid-Weighted Linear Unit (SiLU) is also one of the activation functions used in this thesis. SiLU is defined as the sigmoid function multiplied by its input [17],

$$SiLU(x) = x \cdot Sigmoid(x). \tag{2.20}$$

The SiLU function is smooth and has a non-zero derivative for all inputs, which can help to improve stability and performance for deep neural networks.

Rectified linear unit activation function, also called ReLU, is often the preferred activation function in the hidden layer. The range of the ReLU function is from 0 to infinity [26]

$$ReLU(x) = \max(0, x) \tag{2.21}$$

One of the activation functions used in this thesis is an extension of ReLU called Leaky ReLU. This function is similar to ReLU but has a small slope for negative values and is given as

$$LeakyReLU(x) = \max(0.1x, x) \tag{2.22}$$

Fig. 2.13 below shows an example graph of four of the activation functions explained.

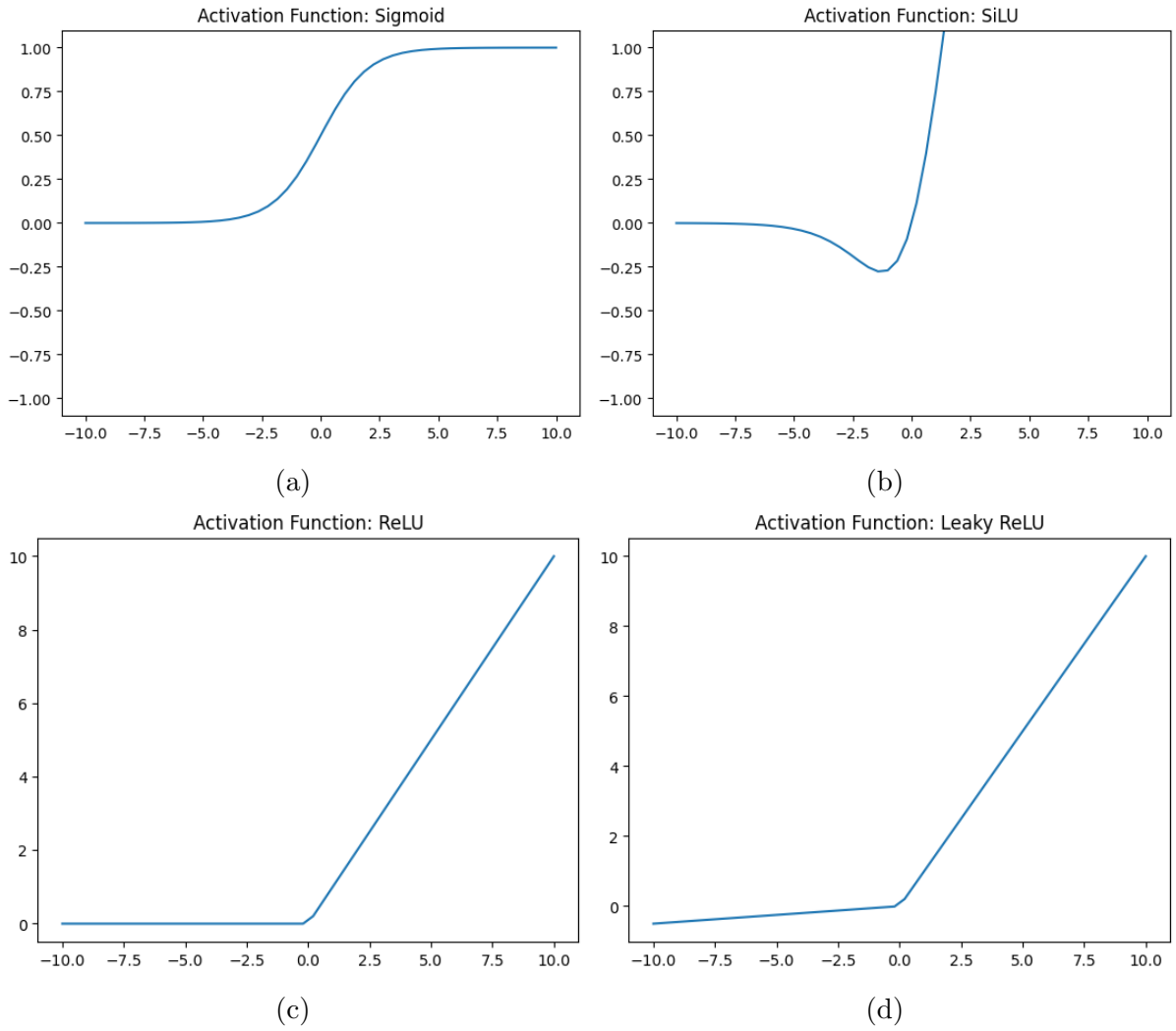


Figure 2.13: Example plots of the different activation functions described above. The activation functions sigmoid (a), SiLU (b), ReLU (c) and (d) Leaky ReLU are plotted.

Credit: Image made by author with matplotlib

When the neural network is trained on the training set, the weights are first set to some initial values. During training, these weights are optimized to get the best weights for the problem. The goal of optimization is to find the set of weights and biases that minimize the error of the network, i.e. minimize the loss function on the training data. This process is known as learning for the neural network. Several optimization algorithms can be used to train a neural network.

Gradient descent is the most common optimization algorithm and is an iterative first-order optimization algorithm used to find a local minimum of a given function [64]. The gradient descent optimization calculates the next point by using the gradient at the current position and scaling it with the learning rate with respect to each parameter in

the model. If our initial weight is w_n, our new weight is w_{n+1}, and the learning rate is given as α, gradient descent is described as

$$w_{n+1} = w_n - \alpha \nabla l(w_n) \tag{2.23}$$

Stochastic Gradient Descent is a variant of gradient descent where randomness is introduced to make the algorithm faster in terms of less computations [64]. Stochastic gradient descent only picks out one data point from the whole data set at each iteration.

Convolutional neural networks

CNN is a type of neural network particularly effective for image analysis. This is because they can be tailored to recognize image patterns [53]. CNNs are widely used in a variety of applications, including object recognition, face detection, image segmentation, and image generation.

The main difference between other neural networks and CNN is that the neurons in CNN layers are made up of neurons organized into three dimensions [53]. These neurons are not fully connected but are connected to a small region of the layer preceding the neuron. The three dimensions are height and width, which are the spatial dimensionality of the input, and the depth. Overall, the architecture of a CNN network is made up of three types of layers. These are convolutional layers, pooling layers, and fully-connected layers.

The convolutional layer is the building block of a CNN. At the heart of a convolutional layer is the convolution operation performed by a set of learnable filters, also known as kernels. The kernels are usually small in spatial dimensionality but spread out in the depth of the input [53]. The main goal of the kernel is to extract essential features of the input. The kernel is placed over the upper right corner of the images and is moved across the input image. The convolutional process is performing a dot product with the local region of the image that the kernel is covering. This operation results in a 2D activation map, essentially a transformed and smaller version of the input image. The activation map is then passed on to the next layer in the network. The number of filters in a convolutional layer determines the depth of the layer. The more filters there are, the more complex features the layer can learn.

In general, deeper layers in a CNN can learn higher-level features that are more abstract and potentially more useful for the task. In a CNN, the network learns what values to

use in the kernel for training. Fig. 2.14 shows a 6x6 image with a 3x3 kernel and the activation map after the convolution process.

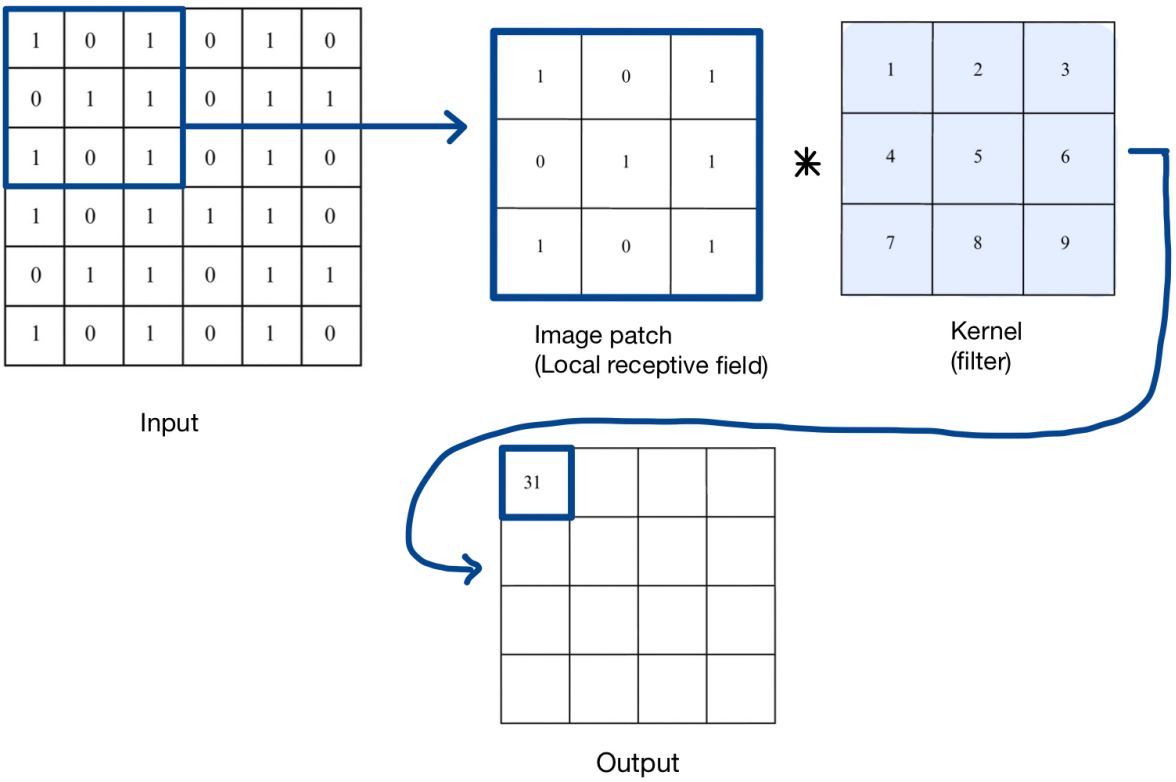


Figure 2.14: An 6x6 image with a 3x3 kernel and the activation map at the end.

Credit: Image made by author

The filter is moved across the input image with a certain stride and padding. The stride is the number of pixels the filter moves each time, and it determines how much the filter jumps over between applications on the input image [53]. It has a big impact on the size of the resulting activation map. Zero padding is when an image is padded with zeros around as a border before convolution is applied. It allows controlling the dimensionality of the output volumes [53].

Pooling layer is another layer in a CNN. The pooling layer will perform down sampling along the spatial dimensionality of the given input. This will reduce the number of parameters and the computational complexity of the model [53]. Pooling happens the same way as a convolution layer, where a kernel is passed over the image. Two types of pooling layers are commonly used: max-pooling and mean pooling. Max pooling returns the maximum value in the kernel. Mean pooling returns the average of all values in the filter.

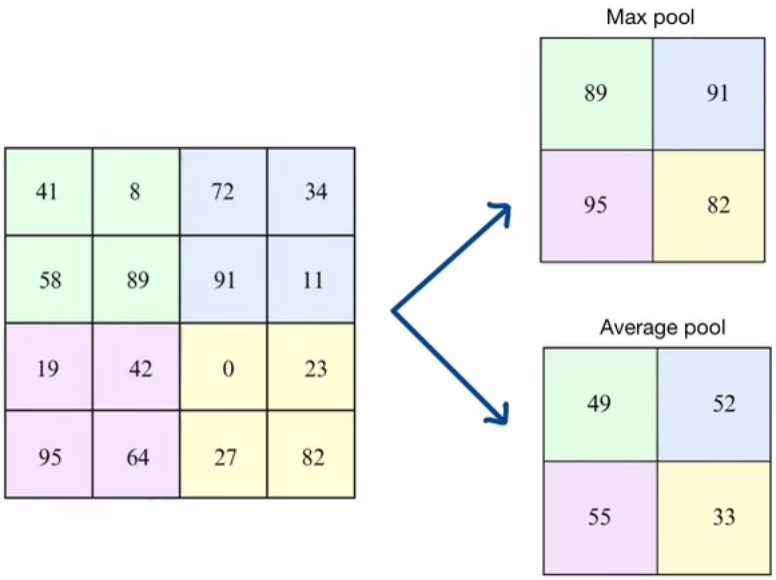


Figure 2.15: This figure shows how max pooling and mean pooling work.

Credit: Image made by author

Fig. 2.15 shows how to do average pooling and max pooling over a 4x4 image with a 2x2 filter and stride 2. The pooling layer is often added after the convolution layer to further downsize the image, while retaining the important properties.

After downsizing the data with convolutional layers and pooling layers, the data is fed into a fully connected layer. The fully connected layer has neurons that are arranged as a feedforward neural network [53]. This layer will make predictions based on the convolutional layer and pooling layer. The last fully connected layer will have the same number of outputs as the classifications in the dataset.

2.3.3 Data augmentation

Because of limited data in the medical field, data augmentation can be used to increase the size of the dataset used for deep learning models. Increasing the data improves the sufficiency and diversity of the data [75]. This can make the model more robust.

Data augmentation can also help reduce overfitting. By doing data augmentation, the deep learning model is exposed to a wider variety of data, which can help the model generalize better to new data. Data augmentation can also address the class imbalance

in object detection [78], where some classes are more common than others. After data augmentation, the distribution of the classes can be more balanced, which can cause improved results.

The general methods of data augmentation for object detection include transforming the original images in various ways, such as image manipulation [75]. Image manipulation can be rotation, flipping, scaling, or cropping. Rotation includes rotating the image by a certain angle to generate new images with different orientations. Flipping can be horizontal or vertical, putting the objects in different positions in the image. Scaling is also a common image manipulation method, where the image is scaled up or down to create images with different sized images. Cropping the image to remove parts of the edge of the image, can also be done to generate images with different perspectives.

Noise injection and different color operations are also common data augmentation methods [78]. Noise injection adds noise into the image, with for example Gaussian noise. Adding Gaussian noise to the image, generates new images with random variations. Color operations such as changing the brightness, contrast, or saturation are also common to generate new images with different color variations. Figure 2.16 shows an example of different augmentation methods done on an ultrasound image.

Some research has been done for data augmentation specifically for ultrasound images. The article by Tirindelli et al. [23] describes a more physics inspired approach for ultrasound images. Instead of doing the traditional augmentation methods mentioned above, methods inspired by deformation, reverberation, and signal-to-noise ratio were used. After comparing the traditional and the physics inspired methods, the difference in accuracy was small, with traditional methods actually being better than the physics inspired methods when all of them were combined.

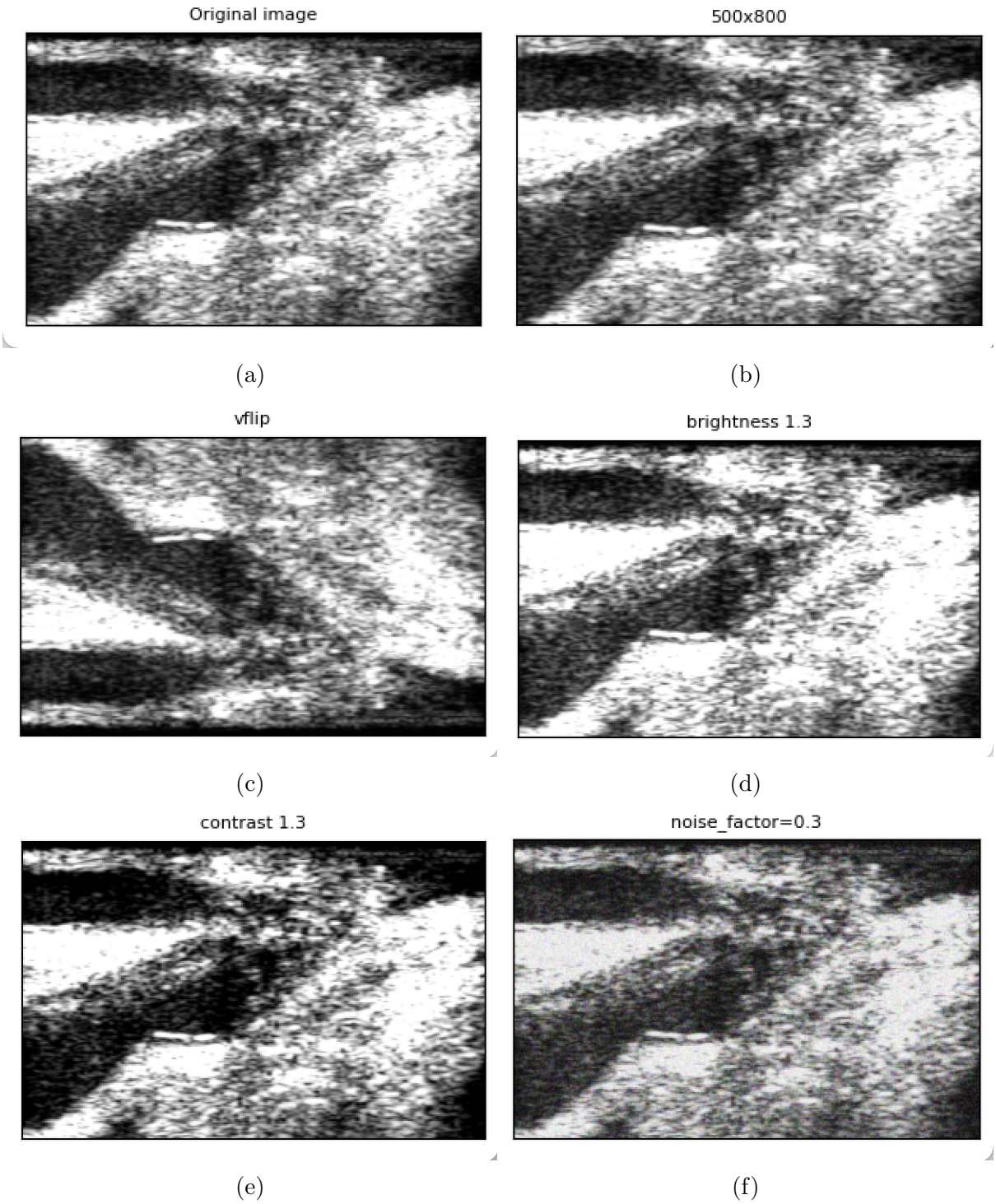


Figure 2.16: Example of different data augmentation methods done on an ultrasound image. (a) shows the original image, (b) is center cropped, (c) is flipped vertically, (d) has added brightness, (e) has added contrast and (f) has added Gaussian noise.

Credit: Image made by author with PIL

2.3.4 Yolo algorithm

You Only Look Once (Yolo) is a state-of-the-art object detection algorithm introduced by Joseph Redmon et al. in 2015 [58]. As the name of the algorithm suggests, Yolo only forward propagates through the neural network one time to detect objects, which means that the image's prediction is made in a single algorithm run. The article [58] reframes object detection as a single regression problem. This allows for real-time prediction and an efficient and effective algorithm.

The Yolo algorithm is a CNN, where the CNN architecture is used to predict the various class probabilities and bounding boxes. It consists of a series of 24 convolutional and max pooling layers, followed by 2 fully connected layers [58], see fig. 2.17. The convolutional layers extract essential features from the input image, while the max pooling layers are used to reduce the spatial dimensions of the feature maps, which increases the computational efficiency of the network. The fully connected layers are responsible for making the final prediction of the object's class and the coordinates of the bounding boxes.

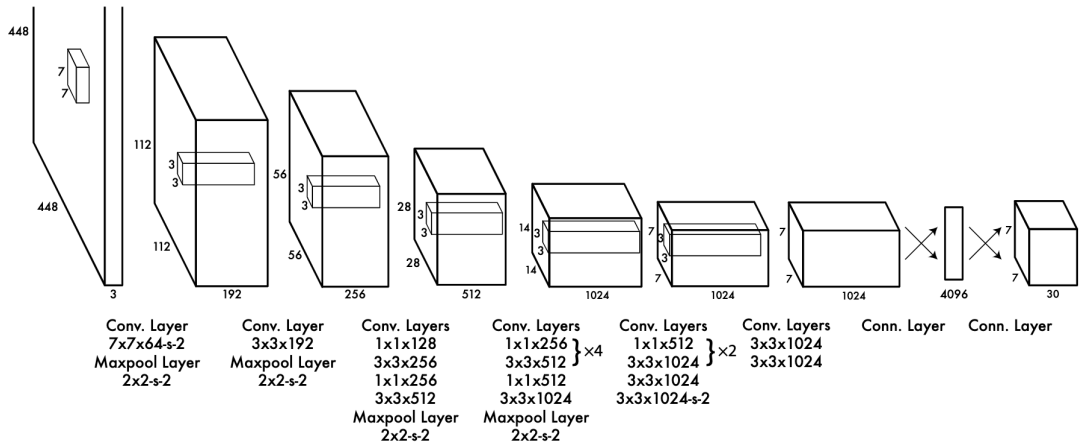


Figure 2.17: The images shows the architecture of a yolo network, with 24 convolutional layers with max pooling, followed by 2 fully connected layers

Credit: Figure 3 from [58] reproduced with permission.

The algorithm creates a grid pattern out of the input image [58]. The grid pattern in Yolo is used to divide the input image into a grid of cells. Each cell predicts a fixed number of bounding boxes and associated confidence scores. The coordinates of the bounding boxes are predicted relative to the coordinates of the cell in which they are located. This allows Yolo to make predictions with respect to the entire image at once. When an object's center falls inside a grid cell, that grid cell is in charge of that object. Each bounding

box has five predictions with *x, y, w, h, and the confidence score.* The *(x,y)* coordinates represents the center of the box, while the w is the box's width, and h is the height of the box. If no object exists in the cell, the confidence score is zero. If not, the confidence score should be equal to the IoU between the predicted box and the ground truth. Yolo uses features from the entire image to predict each bounding box and predicts all bounding boxes across all classes for an image simultaneously [58]. The algorithm is very fast since Yolo frames detection as a regression problem. The base network runs at 45 frames per second with no batch on a Titan X GPU [58] while maintaining a high average precision.

The Yolo algorithm has been constantly improved over the years, and one of the latest versions Yolov5, was released in 2020 by Ultralytics [20]. This is the version of Yolo used in this master thesis. The Yolov5 algorithm was released with five different model sizes, where the number of layers and parameters are different. The different sizes are

- n for extra small (nano) size model, with approximately 1.9 million parameters,
- s for small size model, with approximately 7.2 million parameters,
- m for medium size model, with approximately 21.2 million parameters,
- l for large size model, with approximately 46.5 million parameters,
- x for extra large size model, with approximately 86.7 million parameters.

In later versions of Yolo, like Yolov5, the bounding boxes are predicted by displacements from anchor boxes. In Yolov5, anchor boxes are used to help the model predict bounding boxes of different shapes and sizes. The anchor boxes are pre-defined boxes of different sizes and aspect ratios that are placed at each grid cell of the image. The model then predicts the offsets of the anchor box and the probability that an object exists within the box. Yolov5 introduces the concept of "auto anchor boxes", which are generated automatically based on the dataset being used [8]. The model first clusters the ground truth bounding boxes into k groups based on their similarity in size and aspect ratio. Then, the model uses the k-means algorithm to find the center of each group, which becomes the anchor box for that group. This helps improve the model's precision by ensuring that the anchor boxes are tailored to the specific dataset [8].

Yolov5 uses the activation functions SiLU, Leaky ReLU, and Sigmoid [20]. SiLU and Leaky ReLU is used with the convolution operations in the hidden layers, while Sigmoid is used in the output layer of the architecture. Since Yolov5 returns three different outputs, the objectness score, the classes of the detected objects, and the bounding boxes, the loss is a combination of three different losses [20]. Binary cross-entropy loss is used to compute

the classification loss and objectness loss. Complete Intersection over Union loss is used to compute the localization loss for the bounding boxes. The localization loss measures the error in predicting the bounding box coordinates, while the objectness loss measures the probability that an object exists within the bounding box. The classification loss measures the probability that the object belongs to a particular class.

2.3.5 RetinaNet algortihm

RetinaNet is an object detection algorithm introduced by Tsung-Yi Lin et al in 2017 [37]. Just as mentioned for Yolo, RetinaNet is an one-stage object detection algorithm with only one single forward pass, making the algorithm fast and efficient. The difference is that RetinaNet identifies class imbalances during training, a commonly encountered problem in object detection [37].

The class imbalance problem occurs when there are a lot more background pixels in an image compared to the pixels belonging to the objects of interest due to dense sampling of anchor boxes. RetinaNet addresses this problem by using a loss function called focal loss. Focal loss is a dynamically scaled version of cross entropy loss, such that it down-weights the loss for easy examples and focuses on the loss for hard examples [37]. A modulating factor $(1 - y_i)^\gamma$ is added to the cross entropy loss, with a tunable focusing parameter $\gamma \geq 0$. pt is the model's estimated probability for the class. When pt is small and an example is misclassified, the modulating factor is close to 1, and the loss is unaffected. The modulating factor changes to 0 as pt approaches 1 and the loss for correctly identified cases is down-weighted. The focusing parameter smoothly adjust the rate at which easy instances are down-weighted.Focal loss is the same as cross entropy loss when $\gamma = 0$. When γ is increased, the effect of the modulating factor is also increased. The focal loss is defined as

$$FL = -\sum_{i=1}^{i=n} (i - y_i)^\gamma \cdot \log(\hat{y_i}). \tag{2.24}$$

The RetinaNet network architecture is built on a feedforward ResNet backbone, a type of CNN widely used for image classification tasks. RetinaNet features Feature Pyramid Network (FPN) on top of the ResNet backbone to extract the features from the input

image [37]. FPN is a CNN that creates a pyramid of feature maps by combining high-level and low-level features from different stages of the backbone network [36]. The input image is first passed through the ResNet backbone, where the features are extracted, and a series of feature maps at different spatial resolutions are generated. The FPN then uses a top-down pathway to combine the high-level features with lower-level features at different spatial scales. The output of this process is a set of feature maps, each corresponding to a different scale. With a FPN with a top-down approach and lateral connections added to a regular CNN, the FPN allows the RetinNet network to effectively build a rich, multiscale feature pyramid from a single input image.

RetinaNet also includes subnetworks that are task specific, one for classification and one for bounding box regression. These two subnetworks use anchor boxes to predict the class and bounding boxes in the input. The classification subnet predicts the probability that an object is present at each position for each anchor box [36]. This subnet consists of a small fully connected network attached to each FPN level to share the parameters across all pyramid levels [37]. At each pyramid level, four 3x3 convolution layers are applied, followed by the ReLU activation function. A convolutional layer with a filter for each anchor and object is then added. The output of the classification subnetwork is attached to the sigmoid activation function.

Along with the classification subnet, another fully connected network called Box Regression Subnet is included [37]. The subnet for bounding box regression regresses the offset from each anchor box to a nearby ground truth object if one exists. The architecture of this subnet is designed the same as for the classification subnet, with the exception that it produces four linear outputs per spatial location [37]. For each anchor at a given spatial location, these four outputs predict the offset between the anchor and the corresponding ground-truth box. Although the classification subnet and the box regression subnet have a shared structure, they use separate parameters.

The architecture of the RetinaNet network is described in fig. 2.18.

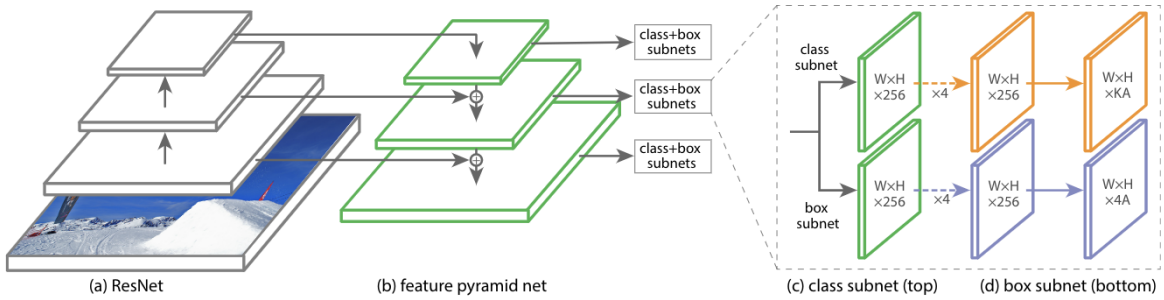


Figure 2.18: The images shows the architecture of a RetinaNet network, with a feature pyramid network backbone on a feedforward ResNet architecture.

Credit: Figure 3 from [37] reproduced with permission.

The loss function used in RetinaNet is a combination of focal loss and smooth L1 loss [37]. The focal loss is addresses the class imbalance problem in object detection, where the number of negative samples far outweighs the number of positive samples. The smooth L1 loss is used to handle the issue of outliers in bounding box regression.

2.3.6 EfficientDet algortihm

EfficientDet is an object detection algorithm that was introduced in 2019 by Mingxing Tan and Quoc V. Le [69]. Tan and Le saw that the usual approach to increase the accuracy of object detection models was to make the network deeper, diminishing the accuracy gain. EfficientDet was therefore created as a more systematic way of model scaling after seeing that balancing network depth, width, and resolution could lead to better performance. The concept of model scaling was first introduced in an EfficientNet paper [68] meant for image classification. By implementing this technique for object detection, the object detection algorithm EfficientDet was born.

EfficientNet forms the backbone of the EfficientDet architecture with the model scaling in center. As fig. 2.19 shows, there are many possibilities to scale a model. Each layer can be wider, the number of layers can be deeper, images with higher resolution could be used, or a combination of all of these could be done with compound scaling [68].

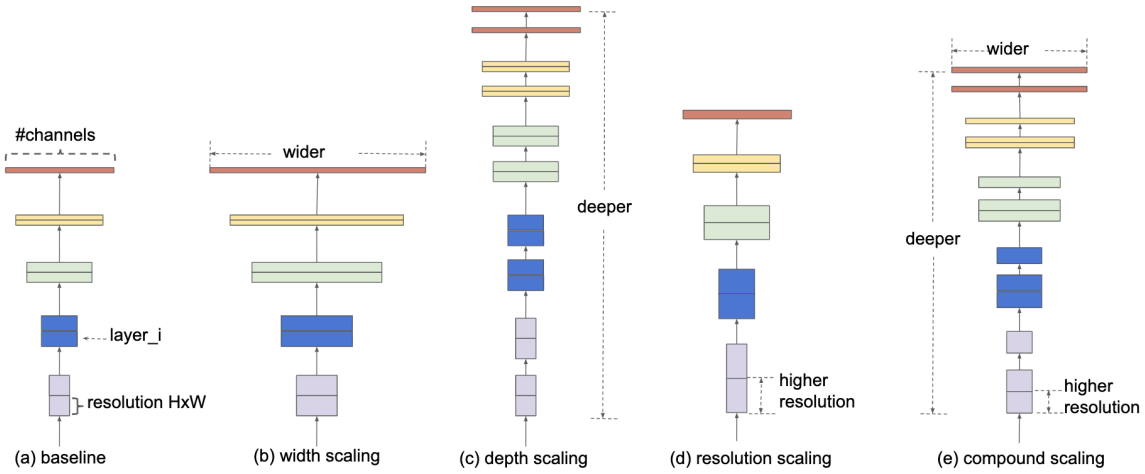


Figure 2.19: The images shows how model scaling works. (a) is a baseline network example, (b) - (d) are conventional scaling that only increases one dimension of network width, depth or resolution. (e) is the proposed compund scaling method that uniformly scales all three dimension with a fixed ratio [68].

Credit: Figure 2 from [68] reproduced with permission.

EfficientNet uniformly scales all dimensions of depth, width, and resolution using a set of fixed compound coefficients. Conventional practice is to scale these dimensions arbitrarily, but by finding a balance and simply scaling each of the dimensions with a constant ratio, the accuracy and efficiency of the CNN are increased [68]. This means that the network depth can be increased by α^N, width by β^N, and image size by γ^N if we want to use 2^N times more computational resources. [68] defined α, β, γ as constant coefficients determined by a small grid search on the original small model. EfficientNet uses a compound coefficient ϕ to scale the network width, depth, and resolution uniformly.

As fig. 2.20 shows, the EfficientNet is the backbone of the EfficientDet object detection model. The next part of the architecture of EfficientDet is the feature network called the Bidirectional feature pyramid network (BiFPN).

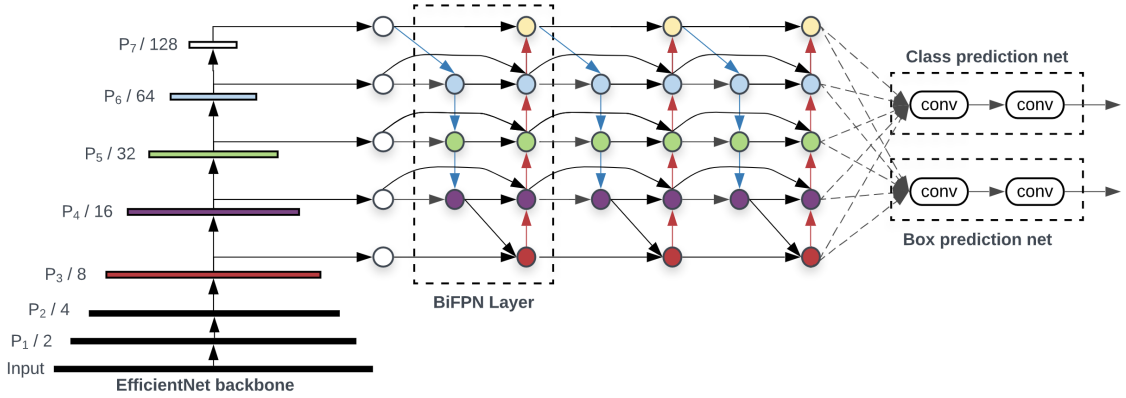


Figure 2.20: The images shows the architecture of a EfficientDet network, with EfficientNet as the backbone network, BiFPN as the feature network and one class and one prediction network.

Credit: Figure 3 from [69] reproduced with permission.

Instead of a FPN feature network as in RetinaNet section 2.3.5, EfficientDet introduces BiFPN to aggregate features at different resolutions [69]. Because different resolutions frequently contribute differently to the output features, the BiFPN is an essential component of the EfficientDet architecture that helps to combine features across different scales efficiently.

One of the key advantages of BiFPN is its use of cross-scale connections, which allows information about the input image to be passed between different scales in the feature pyramid without loss of resolution. Cross-scale connections refer to the links between different levels of the feature pyramid, which are used to propagate information both top-down and bottom-up [69]. Information can move from high to low resolution levels and from low to high resolution levels thanks to the BiFPN network's cross-scale connections between each level of the pyramid and its two neighboring levels. These connections are important for capturing context and integrating information across different scales of an image. By excluding nodes with a single input edge, adding an edge from the original input to the output node if they are on the same level, and treating each bidirectional channel as one feature network layer, BiFPN optimizes these cross-scale connections. In order to enable more high-level feature fusion, the same layer is repeated multiple times [69].

Each input feature gets an additional weight from the BiFPN, which instructs the network on how important each feature is [69]. This is called weighted feature fusion, which is used to combine features from different levels of the pyramid in a way that balances the contribution of each level to the final output.

Overall, the combination of cross-scale connections and weighted feature fusion in the BiFPN network allows EfficientDet to efficiently gather and incorporate information from various image scales, improving performance in object detection tasks. The bidirectional approach BiFPN helps develop a more reliable and precise feature representation for object detection, which can enhance the network's overall performance.

EfficientDet uses a modified focal loss function from RetinaNet that balances the contributions of positive and negative samples during training. The focal loss assigns higher weights to complex examples and down-weights the loss of easy examples, which helps improve the accuracy of the model. Along with the focal loss from RetinaNet, the smooth L1 loss is also used in EfficientDet to handle the outliers.

As RetinaNet (section 2.3.5), EfficientDet has two fully connected convolutional subnetworks for classification and bounding box prediction, where the class and box weights are shared across all levels of features [69].

Chapter 3

Method

3.1 Implementation details

Most of the training, model selection, and model evaluation was done on the high-performance 3D controller called Birget from the Department of Informatics at the University of Bergen. The hardware vendor for Birget is NVIDIA Corporation with the GA100 [A100 SXM4] Graphic processing unit (GPU) model with 80 GB computation capability.

All the data preparation was done using jupyter notebook with Python version 3.7. Pillow, the Python Imaging Library by Fredrik Lundh et al., was used to work with the png images [41]. Pandas DataFrame was used to work on an excel file with the annotation for each image [50]. This was used to read the csv file and divide the data into one DataFrame for the images and one DataFrame for the labels.

All code can be found in Appendix A.

3.2 Data

3.2.1 REQUEST study

The data used in this project are part of a study called The Registry of Quality Assessment with Ultrasound Imaging and Transit-time Flow Measurement in Cardiac Bypass Surgery (REQUEST). To be able to understand this data better, this section is a brief introduction to the REQUEST study. The REQUEST study is a multicenter study documenting if Transit Time Flow Measurement (TTFM) and High Frequency Ultrasound Imaging (HFUS) could improve surgical strategies in CABG [67]. This study aimed to evaluate the influence of TTFM with HFUS in patients undergoing CABG procedures. This was measured by looking at any changes in surgical procedure based on TTFM and HFUS parameters performed with Ultrasound devices from Medistim ASA. HFUS was used during the procedure to scan and identify diseased areas before aortic manipulation. It was also used to identify potential target-vessel anastomosis sites free of plaque. The ultrasound device was also used after the anastomosis to identify potential limit of graft flow due to the anastomosis [67].

1016 patients from 7 different international centers were included in the final analysis done between 2015-2017 [67]. The surgeons were trained to use and interpret TTFM and HFUS results using a structured REQUEST study protocol. There were 36 surgeons trained, and they were recommended to perform more than 20 cases with TTFM and HFUS before participating in the REQUEST study [67]. A study protocol was recommended with surgical steps for the surgery. Even though it was strongly recommended to follow the study protocol, it was not mandatory.

In conclusion, surgical changes in strategy were made in 25% of the patients [67]. More than three out of four of these changes were due to TTFM and HFUS. This was associated with low operative mortality and low major morbidity. The results show that transit-time flow measurement and high-frequency ultrasound may improve the CABG procedure's quality, safety, and efficacy. While using HFUS during the procedure, the surgeons were trained to save the ultrasound images leading to a database of images. The medical images used in this thesis originate from the REQUEST study and are therefore intra-operative ultrasound images of the vessels on the heart during coronary artery bypass grafting.

3.2.2 Annotation of data

The data in this thesis was annotated the summer before this project started. There were 12 850 videos saved from the REQUEST study, which led to 1 480 642 frames. A Python algorithm picked out the most unique frames from each video based on the pixel difference in the images. This was because the frames saved from each video were often similar. To increase the diversity of the dataset, each video frame was compared to the middle frame. The middle frame was saved, along with the frames with a pixel difference of 20%. This resulted in 14 742 unique frames which were annotated with the corners of the bounding boxes in a Python program written by Medistim. The frames were also annotated with the direction of the probe: transversal or longitudinal, and if the image consisted of vessels, an aorta, or an anastomosis. Three different individuals performed the labeling.

There were also images annotated as unknown, blank, or questionable. The blank images were mostly black or grey, which happens at the start of an ultrasound video before the probe has touched the skin. Questionable were images where the individuals annotating the images did not know what to annotate on the images, or that the images contained some text or drawn figures, making them stand out in the dataset. Unknown images were images that were not blank but had no object. In this thesis, the questionable results were ignored.

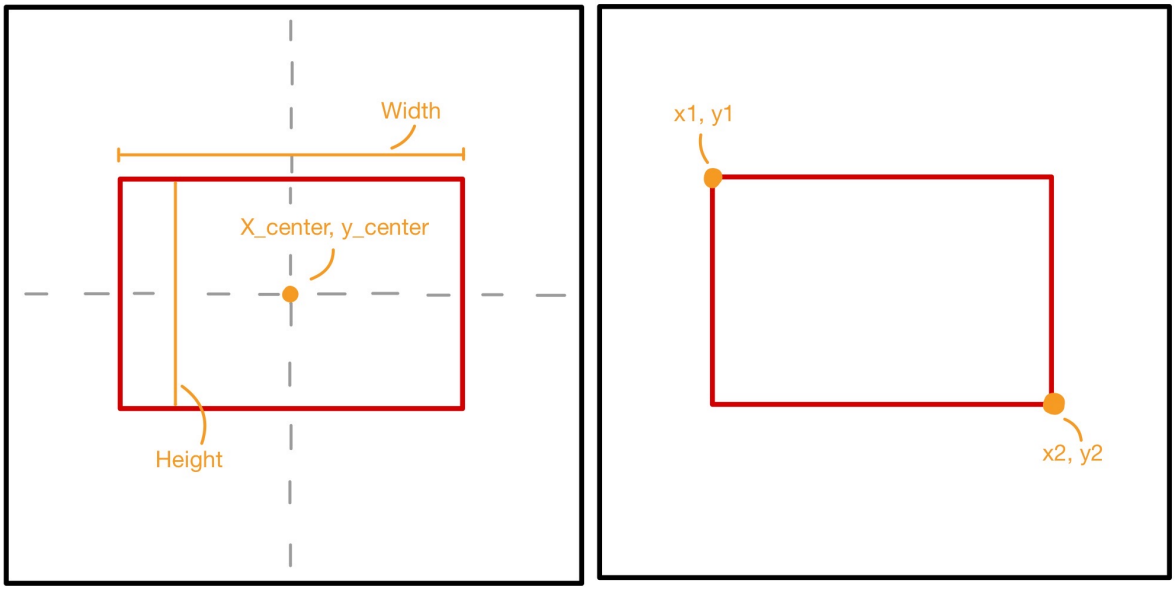
The data was stored in folders for each patient and subfolders for each measurement set. Each image was converted to a png format earlier in the annotation process. A single file with all the annotation information, including patient, case number, measurement set, number of objects in the image, labels, and bounding box coordinates for every image, was made as an Excel file.

3.2.3 Preprocessing of data

The data was divided into training, validation and a test set with Scikit learn [61]. 70% of the data was used for training, 15% for validation, and the rest 15% images were used for the test set. The same validation set was used to validate all of the models to keep a consistent validation in the model selection. The test set was only used to test the best model in the model evaluation. After splitting the dataset, the data was copied to different folders to keep the training, validation, and test set the same throughout the thesis. This was done using Shutil, a high level file operation module [27].

The data was then labeled according to each network. Yolo, described in section 2.3.4, has one number for the labeled class and four numbers for the bounding box. The four numbers are x_{center}, y_{center}, $width$, and $height$, as fig. 3.1a visualizes. Yolo uses normalized values between 0 and 1 for the bounding box. The class label and labels for the bounding box were saved in a text file with the same name as the image. Each row in the text file correspondeds to one object. Only images with an object had a text file.

RetinaNet, described in section 2.3.5, collects all the labels for each image in a csv file divided into training, validation, and testing. The box coordinates are two corner points in the bounding box $x1$, $y1$, $x2$, $y2$, as fig. 3.1b visualizes. The csv file also includes the path to the image and a class name for the class label.



(a) (b)

Figure 3.1: Figure (a) is an example of how Yolo was annotated with x_{center}, y_{center}, $width$ and $height$, and figure (b) is an example of how RetinaNet was annotated with $x1$, $y1$, $x2$, $y2$.

Credit: Made by Author

EfficientDet, described in section 2.3.6, uses COCO JSON labeling format. A JSON file with information about all the images in the split is made for each training, validation and testing set. The information includes annotations of class labels and the bounding box coordinates.

3.2.4 Exploratory data analysis

An exploratory data analysis was done to gain further insights into the distribution of classes, vessels, and annotations in the dataset. In order to visualize the class balance of each label, a pie chart was created using the matplotlib library. This allowed for a quick and easy comparison of the relative frequency of each class. Additionally, the distribution of classes was examined before and after data augmentation, and for the training data. This provided a more comprehensive understanding of how data augmentation impacted the distribution of classes.

Furthermore, a bar chart was generated to analyze the variation in the number of vessels per image for those labeled with vessels. This visualization method allowed for the easy identification of any outliers or trends that may exist within the dataset, which could be further explored and investigated. The bar chart was made for the distribution after data augmentation.

In order to visualize the annotations of the bounding boxes for the training set, a Python code was developed. This code enabled the visualization of both the original annotations and the annotations after undergoing Yolo transformations, which ensured that the transformation process had been executed accurately and effectively. To ensure a comprehensive analysis, fifty random images were selected from the training set and plotted using the code. This enabled a thorough examination of the variety of images within the dataset and allowed for the identification of any potential outliers or trends that may exist within the data.

Upon reviewing the fifty images with the ground truth bounding boxes, a few examples of particularly well-annotated structures were selected for further analysis. These structures serve as ideal examples of the types of objects that the model is designed to identify using object detection techniques.

3.3 Pilot study

A pilot study of three different models was done to make sure the dataset was fit for the models. Yolo was tested first, RetinaNet second, and EfficientDet was tested last. After the pilot study, the model with the most progress in the results was chosen as the final model to further explore because of time and resource limitations.

3.3.1 Pilot study of Yolo

The official implementation of Yolov5 by Ultralytics [20] was used in this pilot study. The Yolo implementation was released in 2020, and the latest update used in this thesis was released in August 2022.

The Yolov5 network had five different pretrained models, which were used as the initial model to start training. The models ranged from the smallest Yolov5n model, with 1.9 million parameters, to the largest model Yolov5x model, with 86.7 million parameters (see section 2.3.4). All models were tested on each hyperparameter combination to find the best initial model for the data during training.

In this master thesis, the Yolov5 training process was optimized by tuning several hyperparameters. Five different hyperparameters were tuned during the pilot study, while the rest of the hyperparameters were set to default. The hyperparameters chosen for further tuning were learning rate, momentum, weight decay, flip up down, and flip left right. Initially, each of the five pretrained models was trained with default values set to 0.01 for learning rate, 0.937 for momentum, 0.0005 for weight decay, 0.0 for flip up down, and 0.5 for flip left right.

Along with training the five different models on these default values, a hyperparameter room for further tuning was defined around the default values. For each training, one number for each hyperparameter in the hyperparameter room was randomly picked out, leading to a random hyperparameter search. For learning rate, the values 0.0001, 0.001, 0.01, and 0.1 were tested. For momentum, a range between 0.750 and 1.000 was set. A number sampled uniformly at random in the range was then rounded to three decimals. For weight decay, the range was set between 0.0000 and 0.0010. The number that was sampled randomly in this range was then rounded to four decimals. The flip up down and flip left right parameters were tested for values between 0.00 and 1.00 with two decimals.

After training several models, it was found that the flipping parameters decreased the mean average precision. Therefore, flip up down and flip left right were set to zero for the rest of the training. The number of epochs was set to around 35 for every run. The default values were also trained with 300 epochs, but this did not significantly change the mean average precision.

A loop including the machine learning pipeline for training and validation was written to automatically train different combinations of hyperparameters and pretrained weights,

while also doing model selection. The loop included training on the training set, validation of all of the models on the validation set, and then picking out the model with the highest mean average precision based on the results from the validation. The results of the best model from each training loop, along with the hyperparameters, were saved in a collected csv file. Detection was also done on the best model to print out the predicted bounding boxes on the validation images.

In the end, eight training loops were trained and validated with random search. Each training loop contained about five to ten combinations of hyperparameters, and each combination was tested on the five different models. The CSV file containing the best models from each training loop registered eight models after the pilot study.

3.3.2 Pilot study of RetinaNet

A Keras implementation of RetinaNet [22] was used as a base for this pilot study. The implementation was based on the official RetinaNet paper [37]. The backbone of the network was set to ResNet50, which is a ResNet model that is 50 layers deep. This model has around 36.46 million parameters. This was used along with the Feature Pyramid Network that RetinaNet is known for. Pretrained weights given by the Keras implementation were used to train and initialize the model. The weight used was called ResNet50_coco_best_v2.1.0.h5.

Another essential step in the RetinaNet training process was the hyperparameter selection. The hyperparameters batch size, step between each epoch, number of epochs, and learning rate was tested. The step size was decided by dividing the number of images in the training set by the batch size so the model could go through each image in the dataset. The default values were 16 as batch size, 897 as the steps between each epoch, the number of epochs given as 100, and the learning rate 0.00001. The hyperparameter room for further tuning was defined based on the default values from the implementation. The batch size was tested for 8, 16, 32, and 64. The steps were tested according to the batch size, and the learning rate was tested for 0.000001, 0.00001, 0.0001, and 0.001. The model was also tested with different amounts of epochs varying from 35 epochs, 100 epochs to 500 epochs

The loop of the RetinaNet training process involved training different models on different hyperparameters, converting the trained models to inference models, and evaluating them on the validation set. After this was done, the model with the highest mean average precision was picked out. The results and the hyperparameters were saved to a CSV file for the best model of the training loop.

3.3.3 Pilot study of EfficientDet

EfficientDet was the last model tested on the REQUEST data. A PyTorch re-implementation of the official EfficientDet based on the original paper [79] was used. This was the highest rated PyTorch implementation of EfficientDet on paperswithcode [55].

The EfficientDet model was trained with pretrained weights provided by the repository of the model. Eight different pretrained weights with parameters ranging from 3.9 million to 77.0 million parameters were tested. A training loop for EfficientDet was also started with default values of the batch size and learning rate, where batch size was set to 12 and learning rate set to 0.00001. The number of epochs was set low for the testing at 5. A hyperparameter room was set up for a training loop with the range 0.000001, 0.00001, 0.0001, 0.001 for learning rate and 4, 8, 12, and 16 for batch size. Different batch sizes were also tested to run the other weights without getting any further results. Because of time, further training and validation were not done for EfficientDet.

3.4 Further data preprocessing

Data augmentation, resizing and upscaling were done after the pilot studies to fit the final model better.

3.4.1 Data augmentation

After experiencing an imbalance in the dataset while testing the data in the pilot studies, which could impact the performance of the different networks being trained, data augmentation was done to improve the quality and diversity of the dataset. Before data augmentation, the images without elements were 50,3% of the total number of images. The data augmentation was only done on the images containing an element to decrease this percentage and balande the dataset more.

A center crop function was used on all of the images to be augmented, where 25 pixels from each side of the image were removed. The cropped images were then separated into two parts, with half of the images in each part. Each part had three different augmentations done on each image in order to further increase the diversity of the dataset.

For the first round of augmentation, half of the images had their brightness changed, while the other part had their contrast changed. The second round of augmentation divided the images into a part with no further modification other than cropping and a part with Gaussian noise applied. The last round of augmentation consisted of a combination of Gaussian noise and either contrast or brightness adjustments. The first half of the images had Gaussian noise applied first, before the contrast of the image was changed. The second half of the images had Gaussian noise added first before the brightness was changed. An example of augmented images can be seen in fig. 3.2

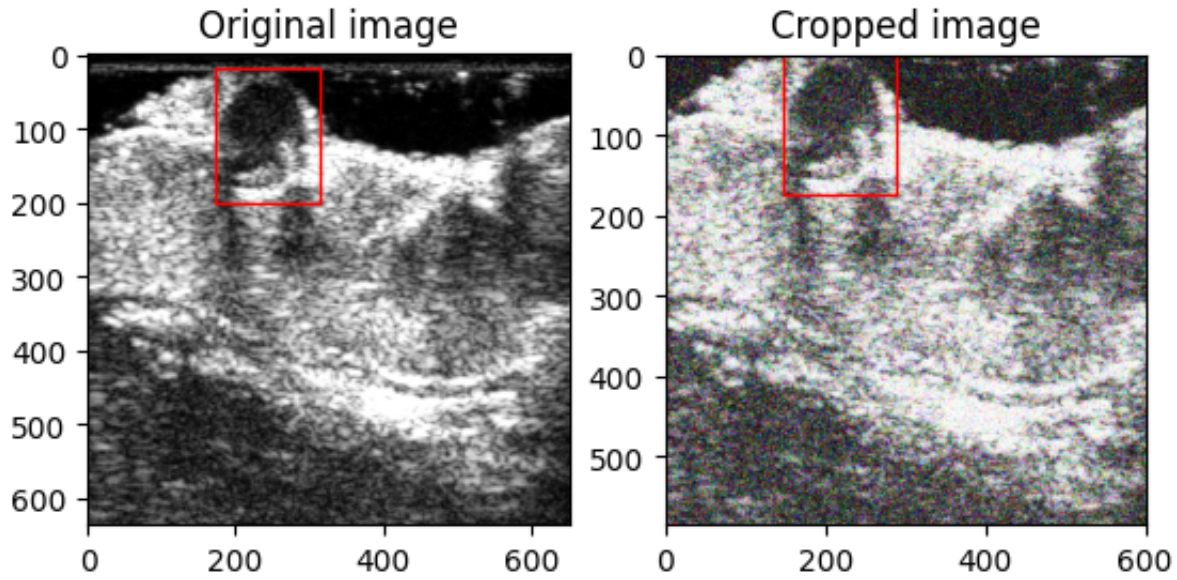


Figure 3.2: The images show an example of how the data augmentation was done on the training data. The right image has been cropped with 25 pixels on each side, Gaussian noise has been applied and it has a 3% increase in brightness.

Credit: Image made by author with matplotlib.

Once all the augmentation techniques were applied, the augmented images were added to the dataset, resulting in a larger and more diverse dataset for training the model. The final dataset included four copies of the images containing an element, where one copy was the original image and two of the copies were cropping and one other augmentation. The last copy included a combination of cropping, Gaussian noise, and either brightness or contrast adjustments. This resulted in the background images only being 24.76% of the total number of images, now increased to 28 028 images.

3.4.2 Resizing and upscaling of images

During the pilot studies, data analysis was done before and after training and validation of the models to ensure that the models correctly interpreted the annotations of the bounding boxes. A Python code was developed to collect the labels from the models during the different stages to visualize them. After plotting the ground truth and predicted bounding boxes during exploratory data analysis of the Yolo model, the ground truth did not look the same as when annotated. This was because Yolov5 resized each image to be a square at 640 x 640 pixels without resizing the bounding boxes.

To fix this issue, upscaling the images to 640 x 640 pixels and resizing the labels were done. The height and width of the original image were compared to find the biggest side. The images were then resized on the biggest side to fit 640 pixels. The labels were adjusted accordingly to the percentage change in pixel difference. For the other sides, a gray border were inserted. This resulted in 640 x 640 images, with labels being resized. Fig. 3.3 shows the upscaled image with a resized bounding box.

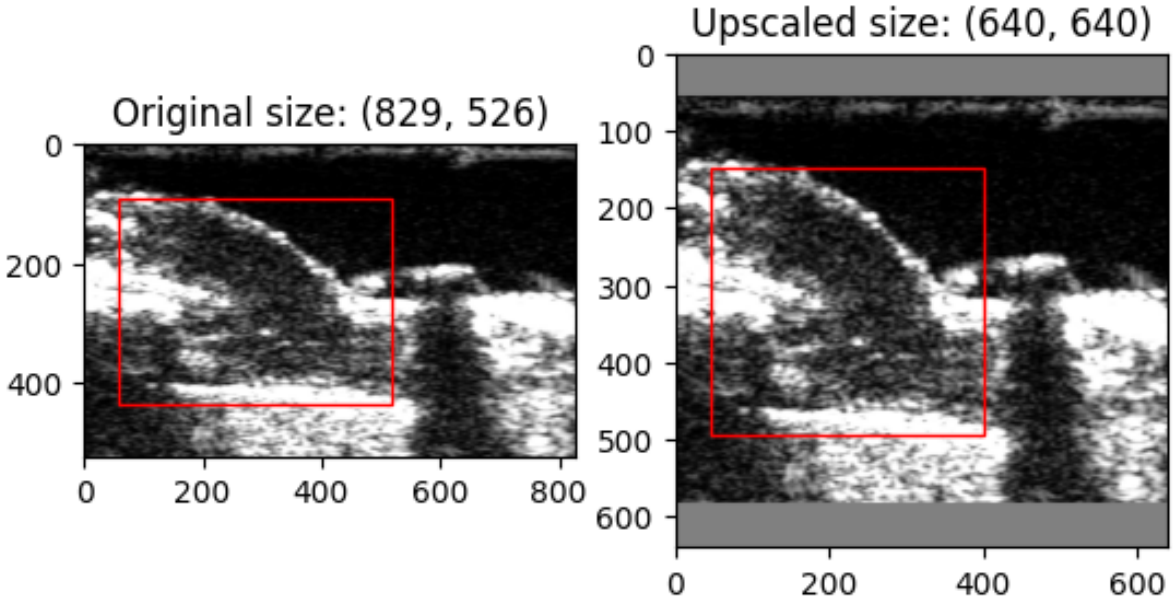


Figure 3.3: Upscaled image from training dataset.

Credit: Images made by author with matplotlib.

3.4.3 Pilot study of Yolo part 2

Before testing the new data with the complete dataset, another pilot study of a muted data augmentation with only two copies of each image with an element was tested. No

resizing or upscaling of the images was done for this pilot study. This was done with the default hyperparameters and the same tuning of the hyperparameter room described in section 3.3.1. The flip parameters, flip up down and flip left right, were set to zero in this short pilot study to avoid the complexity involved in data augmentation of ultrasound images.

The training loop and machine learning pipeline stayed the same as during the first pilot study.

3.5 Yolo-based solution - Final model

After performing data augmentation and upscaling on the ultrasound images in the dataset, the Yolo algorithm was tested on the final dataset. In order to optimize the performance of the algorithm, the five pretrained models mentioned in section 3.3.1 were trained on different hyperparameter combinations using random hyperparameter search. The hyperparameter room was kept the same as in the pilot study. The default values were also tested with both 35 epochs and 300 epochs for each model.

Early stopping patience was used for the final model, where the number of epochs without improvement was set to 5 for the models trained on 35 epochs, and 15 for the models trained on 300 epochs.

The same training and validation loop as in the pilot study was used to train and validate on the models with the validation set. In total, 55 models were trained. The results of the best model from each loop were saved in a csv file, along with the corresponding hyperparameters and results. After all the models were trained and validated, the model with the highest mean average precision at 50% was selected as the final model.

The final model was then evaluated on the test set, which had been set aside at the beginning of the study to ensure unbiased results. Detection was performed on the test set using the final model, which yielded predicted labels, confidence scores, and a precision-recall curve. These results provided valuable insights into the performance of the final model.

Overall, the testing and evaluation process involved a systematic approach to selecting and optimizing the final model. By carefully tuning hyperparameters and evaluating the performance of each model on the validation set, the method was able to identify the most effective approach for vessel recognition on ultrasound images. Using a separate test set for final evaluation ensured that the results were unbiased and provided a reliable evaluation of the model's performance.

Chapter 4

Results

4.1 Exploratory data analysis

In this thesis, the total number of unique images was 14 778. The division of the data is described in the pie chart in fig. 4.1. 24% of the data was of one or more vessels, 12% of the data was of an aorta, while 11% of the data was of an anastomosis. 45% of the data was unknown, 2% were blank, and 7% were questionable. After removing the questionables the total number of images in the thesis was 13 745.

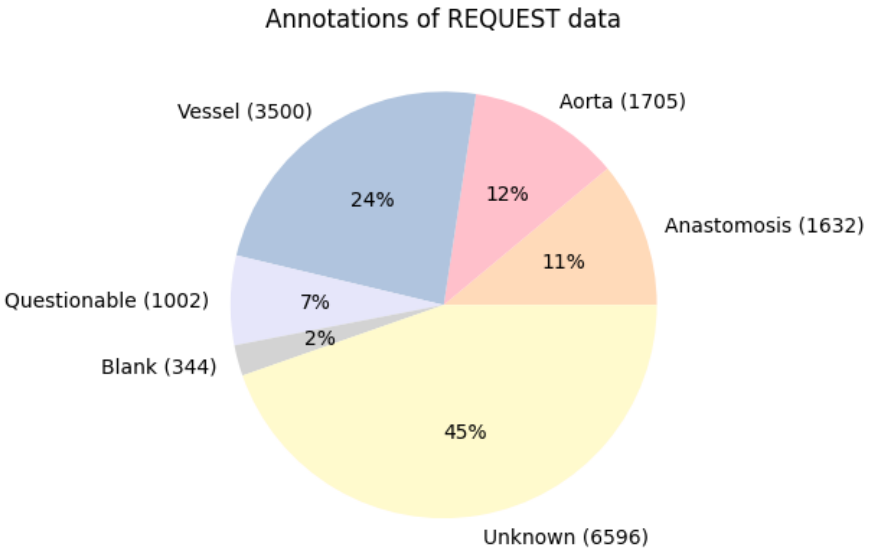


Figure 4.1: The pie chart shows an overview of what the annotated data in the REQUEST study consists of. The number beside each element in the pie chart, is number of images containing the element.

Credit: Image made by author with matplotlib

For the training data, the division of the classes is shown in fig. 4.2. The percentage of images without objects in the training data before data augmentation was 51%. The training data was 70% of the total data.

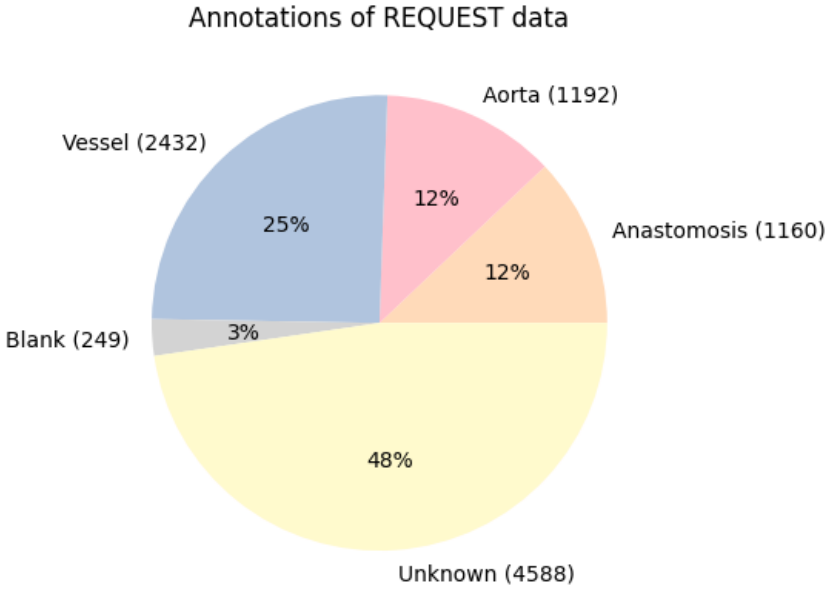


Figure 4.2: The pie chart shows the division of training data before data augmentation. The number beside each element in the pie chart, is number of images containing the element.

Credit: Image made by author with matplotlib

After data augmentation explained in section 3.4.1, the division of the complete data was changed to the pie chart in fig. 4.3. Questionables were removed, and the images containing vessels, aortas, and anastomosis were increased. Total number of images used after data augmentation was 28 028. 5241 of the images were of aortas, 5103 were of anastomosis, 10744 were of vessels, 6596 were unknowns, and 344 of the images were blank. Percentage of images without objects went from 50.3% to 24.76% of total images.

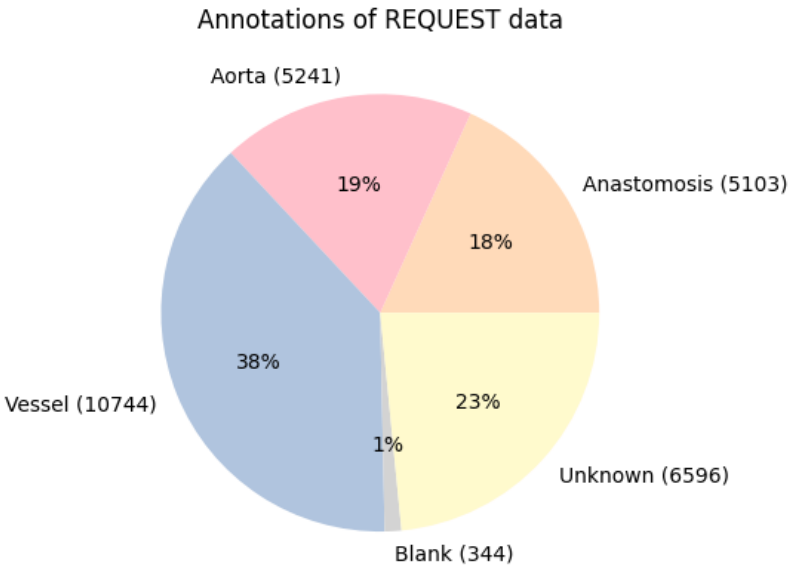


Figure 4.3: The pie chart shows the division of data after data augmentation. The number beside each element in the pie chart, is number of images containing the element.

Credit: Image made by author with matplotlib

The labels in the training data after data augmentation was divided as shown in fig. 4.4. The data augmentation was only done on the training data. After data augmentation the percentage of the images without an object in the training data was changed from 51% to 20%.

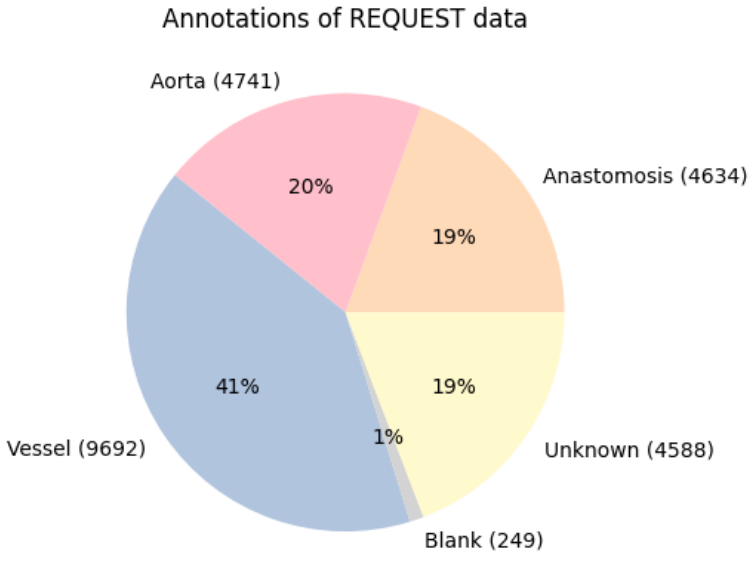


Figure 4.4: The pie chart shows the division of the training data after data augmentation.

Credit: Image made by author with matplotlib

Most of the images containing elements only had one bounding box per image, and one bounding box for the direction of the probe. The images containing vessels, could have multiple elements in one image. Before data augmentation 548 of the images had 2 vessels per image, 162 of the images had 3 vessels, 21 of the images had 4 vessels per image and 1 of the images had 5 vessels per image. The total number of vessels in this dataset before data augmentation was 4429. After data augmentation the division of the number of vessels in the ultrasound images was changed to the numbers shown in fig. 4.5

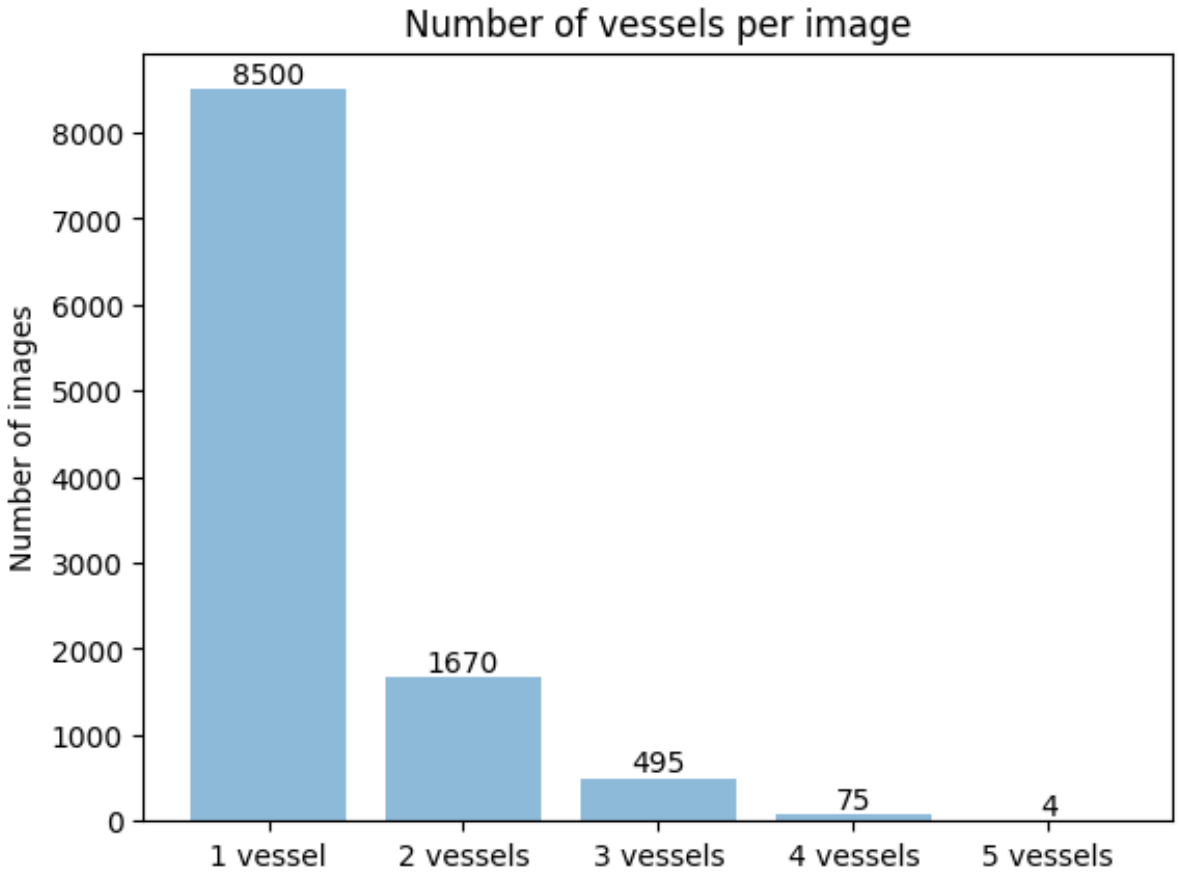
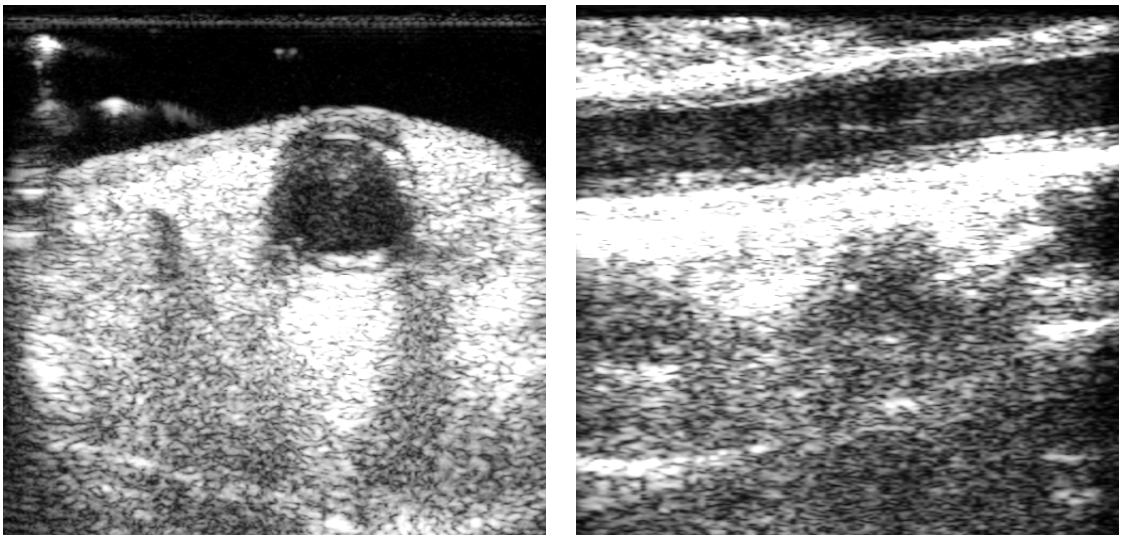


Figure 4.5: The bar chart shows number of images after data augmentation with the different number of vessel per image.

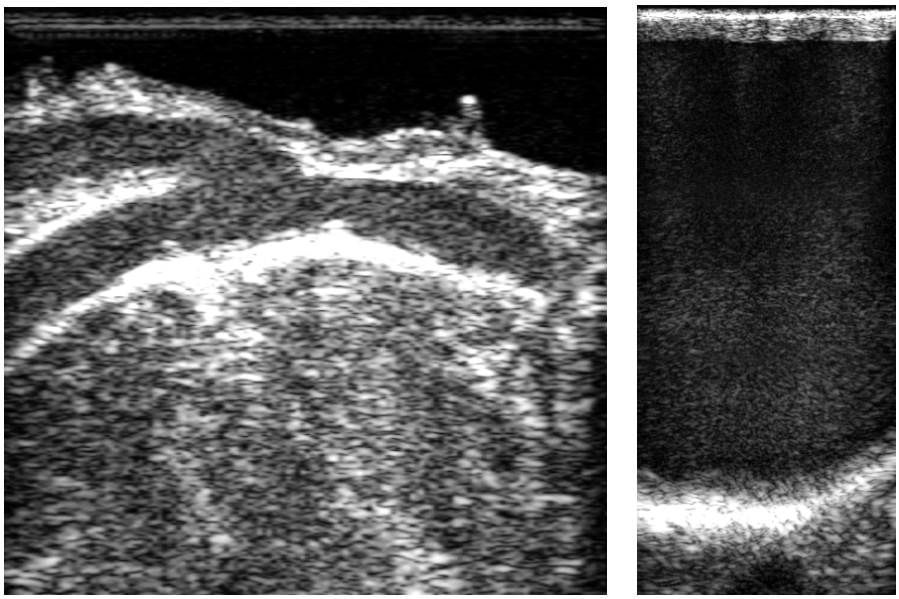
Credit: Image made by author with matplotlib

The total number of vessels in the dataset after data augmentation was 13 645.

Fig. 4.6 shows typical good examples of the structures that can be found in the dataset. The typical examples are taken from the training set. The structures are (a) transversal vessels, (b) longitudinal vessels, (c) anastomosis, (d) aortas.



(a) (b)

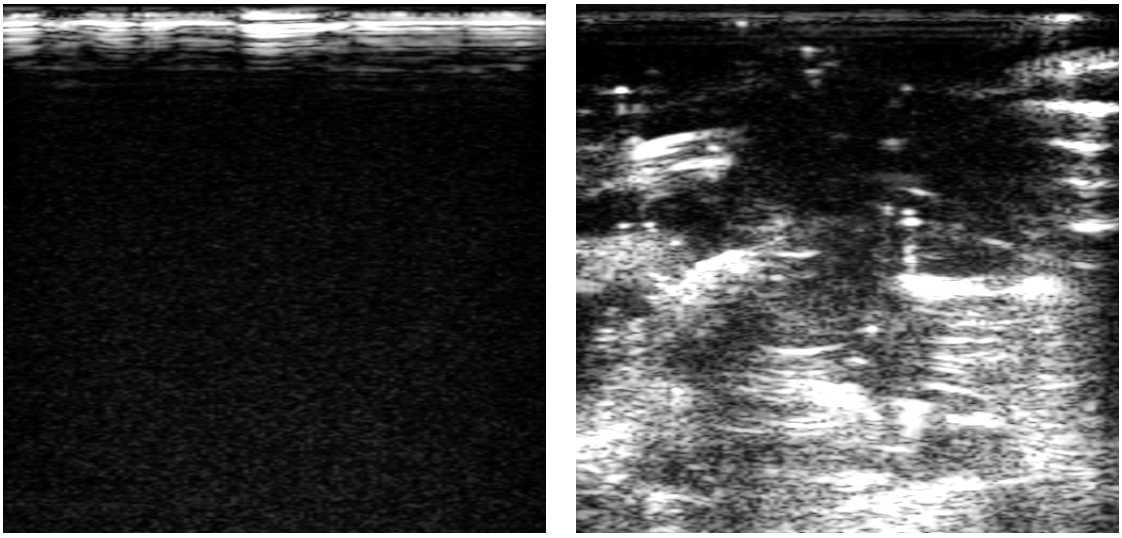


(c) (d)

Figure 4.6: These images are typical good examples from the training dataset. The images are used to clearly show what kind of structures that was looked at in this thesis. Image (a) is a typical example of an transversal vessel, image (b) is a typical example of an longitudinal vessel, image (c) is a typical example of an anastomosis and image (d) is a typical example of an aorta.

Credit: REQUEST study from Medistim

Typical examples of a blank image and an unkown image without any elements are show in fig. 4.7. These images are from the training dataset.



(a) (b)

Figure 4.7: Image (a) is a typical example of an blank image, image (b) is a typical example of an image labeled unkown.

Credit: REQUEST study from Medistim

As seen in fig. 4.7, the blank images are mostly black with a white area at the top from the air touching the ultrasound probe. The unknown image is an image without any noticeable structures. These images often appear at start and at the end of an ultrasound video before the probe has touched the structure that is being imaged.

The size of the elements in the images varied with the type of element. This was because the annotations represented the suggested position of the RoI box on the ultrasound system. The bounding boxes of the vessels were smaller squares or rectangles if the vessels were transversal. If the vessels were longitudinal, the bounding boxes were a longer and narrower horizontal rectangle. Anastomosis was a bigger rectangle that took up a bigger piece of the image, but not the whole image. The aorta images were more vertical, and the bounding boxes of the aortas took up most of the image with a vertical rectangle. Figure 4.8 shows a typical example of annotations of a transversal vessel, longitudinal vessel, anastomosis, and an aorta.

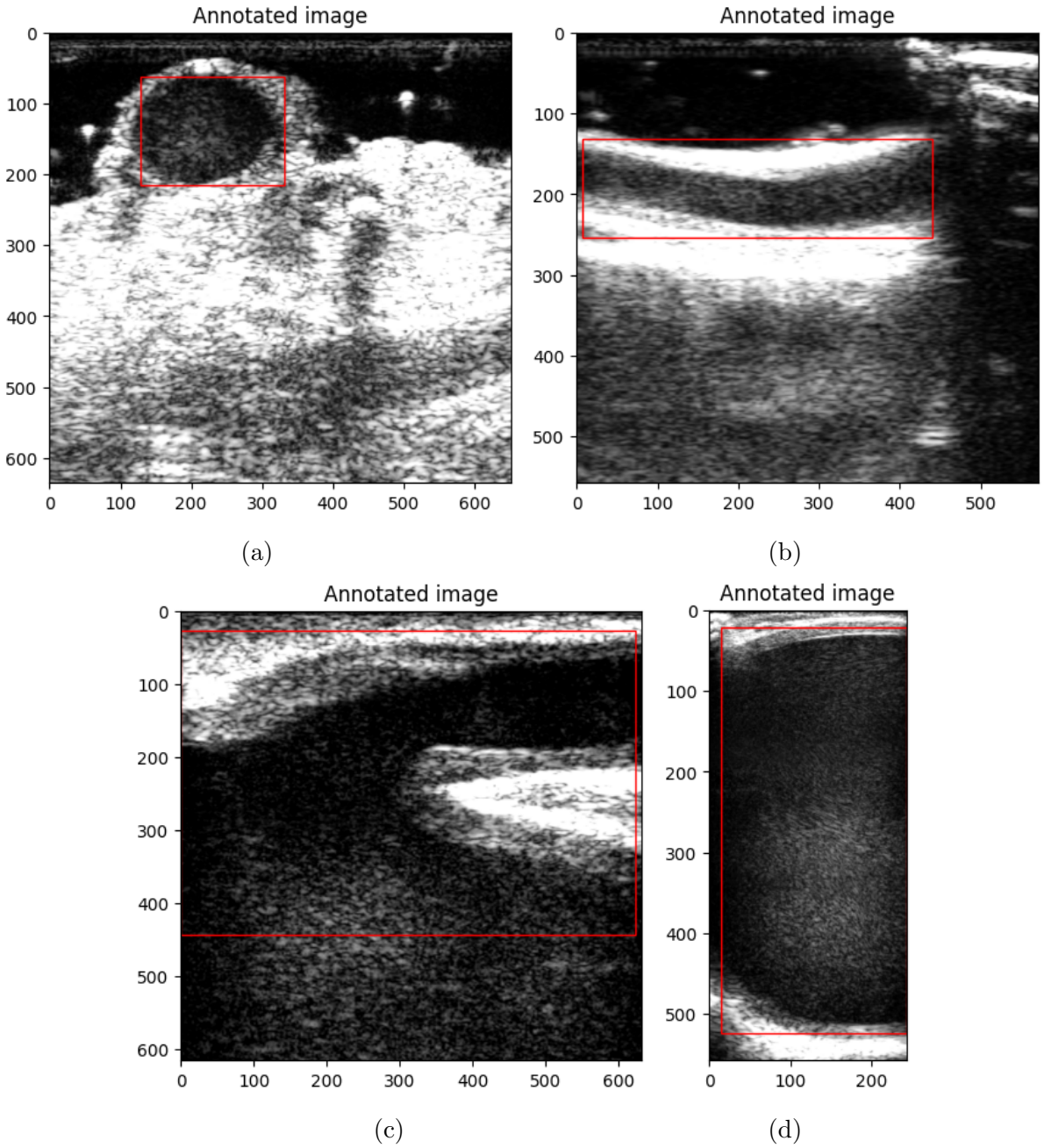
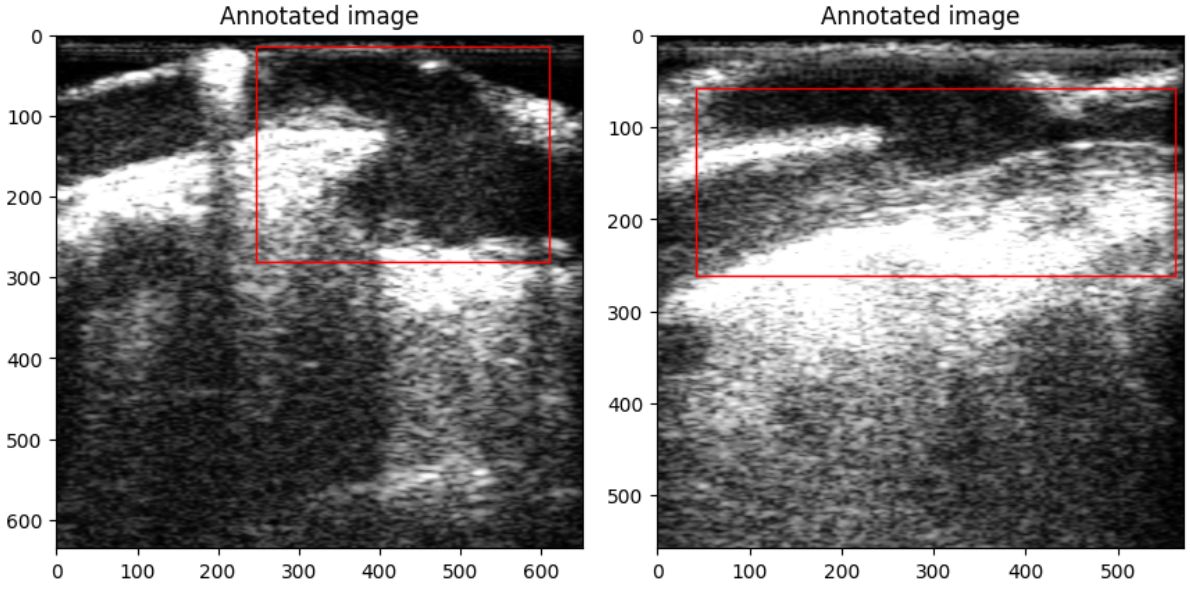


Figure 4.8: These images are good examples of the structures with annotation from the training dataset. Annotations with (a) a typical example of transversal vessel, (b) a typical example of a longitudinal vessel, (c) typical example of an anastomosis and (d) a typical example of an aorta.

Credit: REQUEST study from Medistim

As mentioned in section 3.2.2 the annotations of the data were done by three different individuals. Because of this, the annotations varies a little in some cases. In fig. 4.9 annotations of two similar anastomosis are shown. In the first image the bounding box

covers the entire anastomosis but not the rest of the vessels which belongs to the anastomosis. In the second image, the bounding box is wider and covers the vessels around the anastomosis as well.



(a) (b)

Figure 4.9: Both of the images are annotations of anastomosis. Image (a) shows a smaller but higher annotation of an anastomosis. Image (b) is longer but cuts of the upper vessel.

Credit: REQUEST study from Medistim

4.2 Results of pilot study of Yolo

The best results of the first Yolo pilot study are shown in table 4.1. The first run was done while the flip up down and flip left right was still tuned. The second run was done with the flip parameters set to 0, and with tuned hyperparameters. The last best run was done with default hyperparameters and flip set to 0, but with 300 epochs.

Table 4.1: Best of the results from the Yolo pilot study.

	Best run 1	**Best run 2**	**Best run 3**
Model architecture	Yolov5m	Yolov5l	Yolov5n
Learning rate	0.1	0.1	0.01
Momentum	0.84	0.786	0.937
Weight Decay	0.0009	0.0008	0.0005
Flip Up Down	0.07	0.0	0.0
Flip Left Right	0.14	0.0	0.0
Epochs	35	35	300
Batch	16	16	16
Precision	0.4582	0.4245	0.4360
Recall	0.6311	0.6178	0.6634
mAP50	0.4367	0.4127	0.4440

As table 4.1 shows, the first best run used the middle sized weights with 21.2 million parameters, the second best run used the large sized weights with 46.5 million parameters, while the last best run used the smallest weights with 1.9 million parameters. The mean average precision at 50% for the first best run was 43.67%. The mean average precsion at 50% for the second best run was 41.27%, and the last best run achieved a mean average precision at 50% at 44.40%. The mean average precision was tested on validation data.

4.3 Results of pilot study of RetinaNet

After training and testing several different instances of RetinaNet in the pilot study, it was found that the mean average precision given by the network was very low. This was an indicator that there was something wrong with either the data or the set up of the network. A training loop without pretrained weights was also tried, but this did not increase the mean average precision. To gain further insight and improve the network, additional analysis and debugging was necessary.

Table 4.2 shows three of the best runs for the RetinaNet model, the hyperparameters for these runs. All of the three runs had default hyperparameters but with different number of epochs or pretrained weights. The first run was with pretrained weights and 100 epochs, the second run was without any pretrained weights and 100 epochs, and the third run was with 500 epochs and pretrained weights. The pretrained weight used was $ResNet50\_coco\_best\_v2.1.0.h5$

Table 4.2: Best results from RetinaNet pilot study

	Best run 1	**Best run 2**	**Best run 3**
Weights	pretrained	Not pretrained	pretrained
Learning rate	0.00001	0.00001	0.00001
Batch size	16	16	16
Steps	897	897	897
# of epochs	100	100	500
mAP_ weighted	0.0708	0.0737	0.0648

As table 4.2 shows, the first run with 100 epochs and pretrained weights achieved a mean average precision at 7.08%. The second run without any pretrained weights achieved a mean average precision at 7.37%. The last run with 500 epochs achieved a mean average precision at 6.48%. The mean average precision was tested on the validation data.

4.4 Results of pilot study of EfficientDet

The Birget controller used for training the EfficientDet models could only handle the first three pretrained weights without being out of memory. In the end, the hyperparameters in table 4.3 along with the three smallest weights were trained, but further training did not proceed because of time and resources.

Table 4.3: Best results with the belonging hyperparameters for EfficientDet. AP is the average precision with an IoU at 50%

	Best run 1	**Best run 2**	**Best run 3**
Weights	efficientdet-d0.pth	efficientdet-d1.pth	efficientdet-d1.pth
Learning Rate	0.001	0.00001	0.00001
Batch Size	12	12	12
Epochs	10	10	10
AP at 50% IoU	0.296	0.351	0.325

Table 4.3 shows the best results from the three different weights that was tested. The models were evaluated with average precision (AP) with an IoU at 50%. The smallest weight with 3.9 million parameters managed an AP at 50% at 29.6 %. The two next weights with parameters respectively at 6.6 million and 8.1 million parameters had the same hyperparameters, with AP at 50% at 35.1% and 32.5%. The models were evaluated with the validation data.

4.5 Results of pilot study of Yolo part 2

The results of the pilot study number 2 of Yolo is shown in table 4.4. This pilot study was done with a less advanced data augmentation with two copies of each image with an element. No resizing or upscaling was done for this pilot study. Total number of images in this pilot study was 17 323. The table shows the best run of one of the models that was trained.

Table 4.4: Best of Yolo results from pilot study 2.

	Yolo best run
Model architecture	Yolov5s
Learning rate	0.01
Momentum	0.937
Weight Decay	0.0005
Flip Up Down	0.0
Flip Left Right	0.0
Epochs	35
Batch	16
Precision	0.676
Recall	0.689
mAP50	0.6996

The best run for pilot study 2 for Yolo had default parameters for hyperparameters, along with 35 epochs. The pretrained weight used was the small sized modell with 7.2 million parameters. The mean average precision at 50% achieved was 69.96% for this run. The mean average precision was tested on validation data.

4.6 Results of Yolo-based solution - Final model

After the pilot studies, Yolo had the most progress in the results and ended up working the best with the REQUEST data. Because of resources and time, RetinaNet and EfficientDet were not explored any further.

The three best models from the final Yolo-based solution is shown in table 4.5. The models were trained on the final dataset with full data augmentation and resizing of the ultrasound images.

Table 4.5: Best of Yolo results after data augmentation and resizing of the ultrasound images. The models were evaluated on validation data.

	Yolov5 run 1	**Yolov5 run 2**	**Yolov5 run 3**
Model architecture	Yolov5x	Yolov5x	Yolov5m
Learning rate	0.01	0.01	0.1
Momentum	0.937	0.937	0.867
Weight Decay	0.0005	0.0005	0.0003
Flip Up Down	0.0	0.0	0.0
Flip Left Right	0.0	0.0	0.0
Epochs	35	300	35
Batch	16	16	16
Precision	0.6517	0.6853	0.6575
Recall	0.7070	0.7191	0.7242
mAP50	0.7097	0.7185	0.7141

Run 1 for the final Yolo-based solution had default hyperparameters. The pretrained weights was the largest model Yolov5x with 86.7 million parameters. The mean average precision at 50 % achieved was 70.97% for this run. Run 2 had the same default parameters with the largest pretrained weights, but the model was trained with 300 epochs. The mean average precision at 50% achieved was 71.85%. The last model in table 4.5 had the medium pretrained weights with 21.2 million parameters. The hyperparameters was tuned and number of epochs was set to 35. The mean average precision at 50% achieved was 71.41%. The mean average precision was tested on validation data.

As the table 4.5 shows, the run named Yolov5 run 2 had the highest mean average precision at 50%. This model was therefore picked as the final model in the model selection. The final model was then tested and evaluated on test data.

4.6.1 Results of final model

The hyperparameters, precision, recall and mean average precision at 50% for the final model that was tested on the test set is shown in table 4.6.

Table 4.6: The results of the final model. The model was tested on test data.

	Final model
Model architecture	Yolov5x
Learning rate	0.01
Momentum	0.937
Weight Decay	0.0005
Flip Up Down	0.0
Flip Left Right	0.0
Epochs	300
Batch	16
Precision	0.6604
Recall	0.7294
mAP50	0.7177

After evaluating the final model, the outputs of the model were precision, recall, and mean average precision, given in table 4.6. The precision of the final model was 66.04%, while the recall was 72.94%. The mean average precision at 50% was 71.77% for the final model.

Table 4.7: The results of the final model. The model was tested on test data.

	Precision	**Recall**	**mAP50**
Transversal	0.672	0.748	0.758
Longitudinal	0.626	0.665	0.646
Vessel	0.676	0.686	0.699
Anastomosis	0.624	0.552	0.515
Aorta	0.704	0.996	0.971
All	0.660	0.729	0.718

Table 4.7 shows the precision, recall, and mean average precision at 50% for the orientation of the probe and the three anatomic structures that was labeled in the dataset. The final model identified the vessel oritentation with 75.8% mean average precision for the transversal images, and 64.6% mean average precision for the longitudinal images. The

final speed of inference in both recognizing vessels and identifying the vessel orientation was 5.6 milliseconds per image.

Fig 4.10 shows the precision-recall curve for the final model from the evaluation of the final model on test dataset.

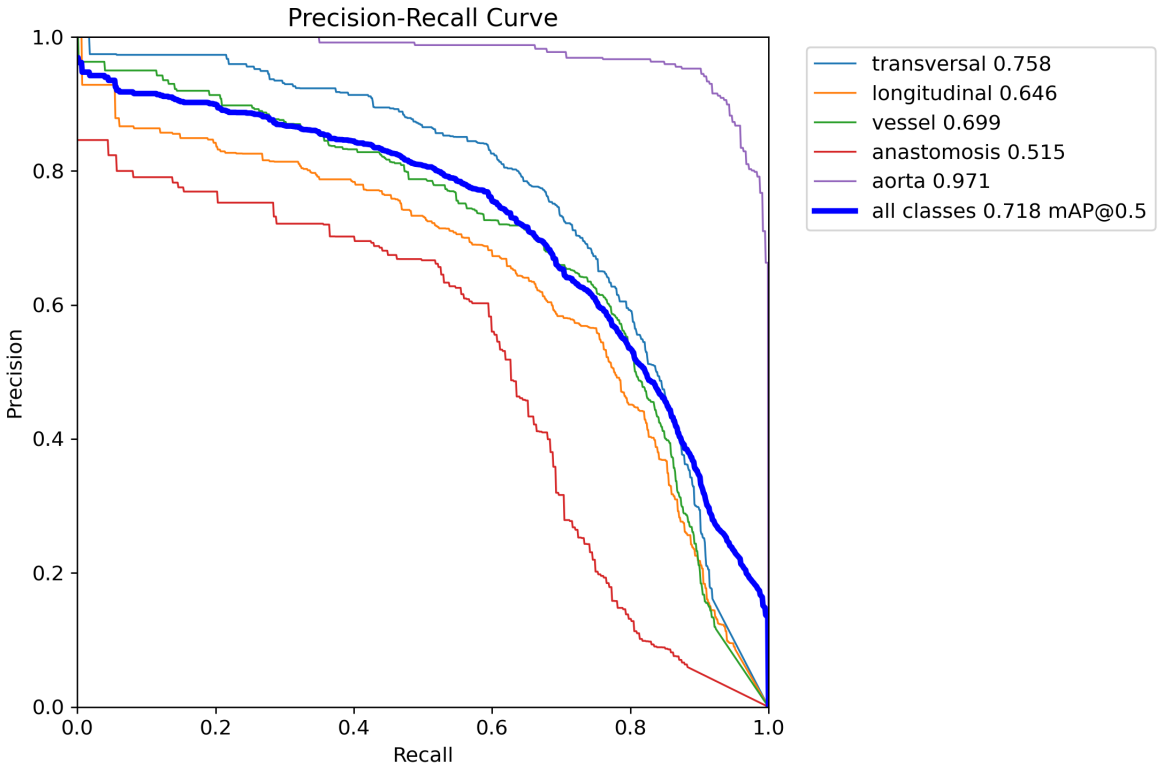
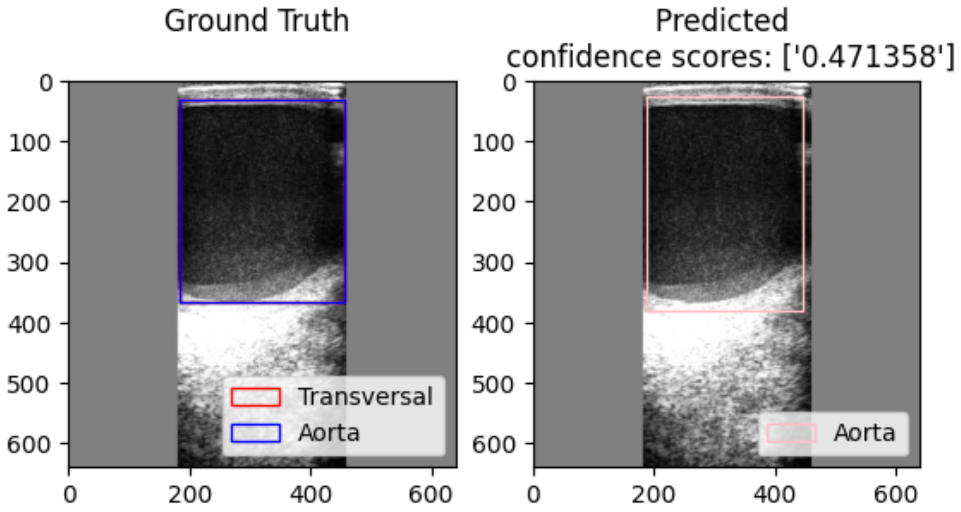


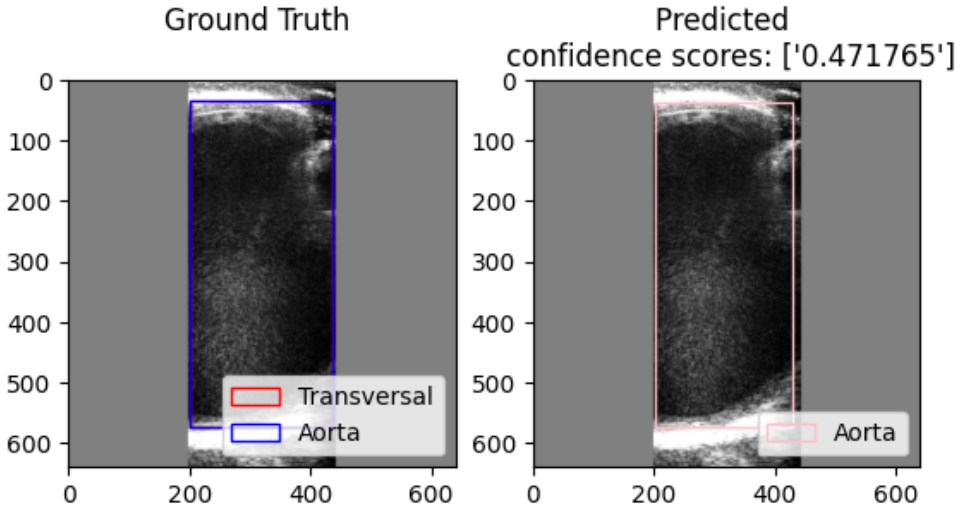
Figure 4.10: The figure shows the precision-recall curve for the final model on test set.

Credit: Image made by author

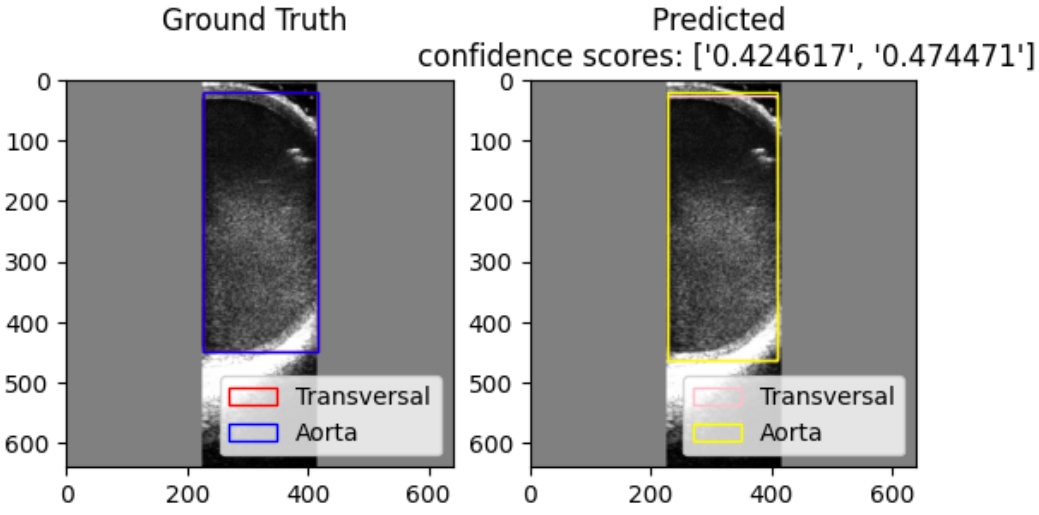
The model also outputed detection images with bounding boxes around the region of interest (RoI) and the predicted labels. Figure 4.11 shows the 3 images from the test set that had the highest confidence score. The ground truth and the predicted labels are also visualized. Figure 4.12 shows the 3 images with the lowest confidence scores from the test set with a confidence threshold at 0.1. Figure 4.13 shows 3 random images in the test set with the corresponding ground truth and predicted labels.



(a)



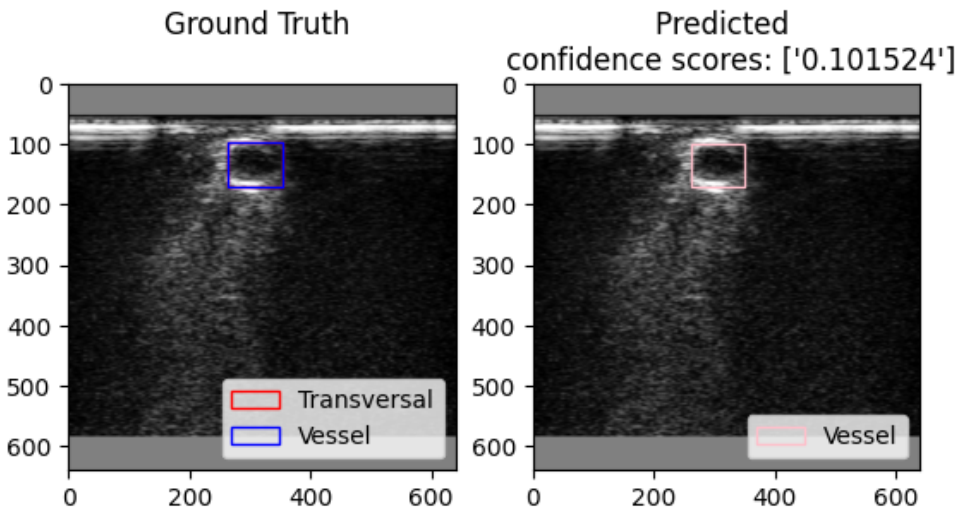
(b)



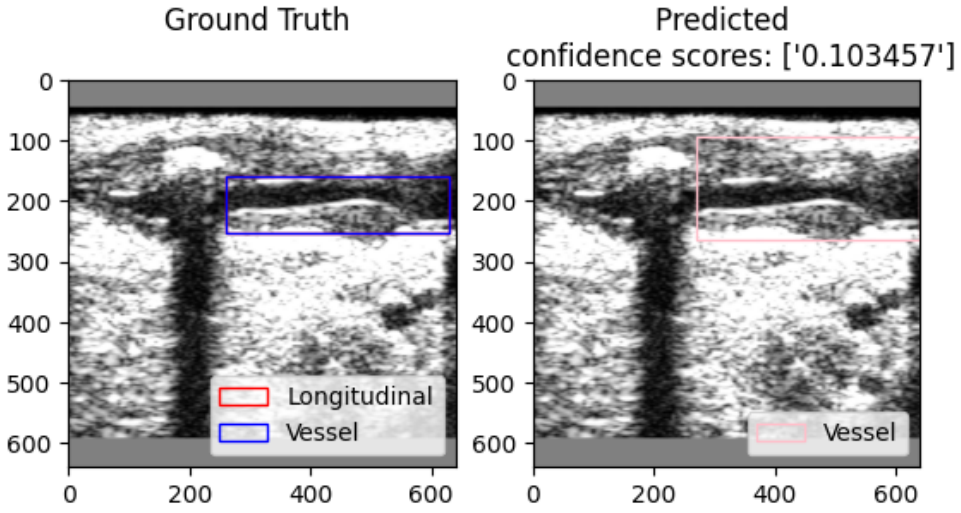
(c)

Figure 4.11: The images from the test dataset with the highest confidence scores with bounding boxes.

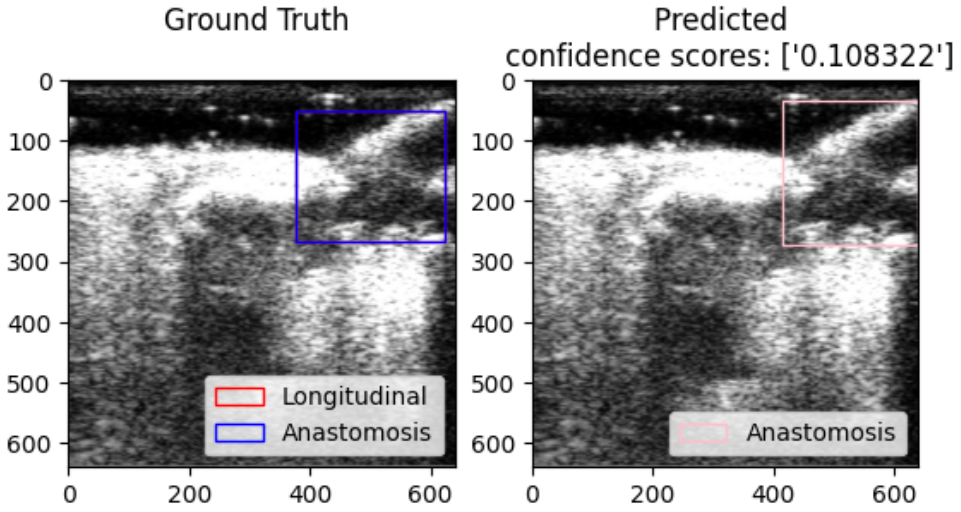
Credit: REQUEST study from Medistim



(a)



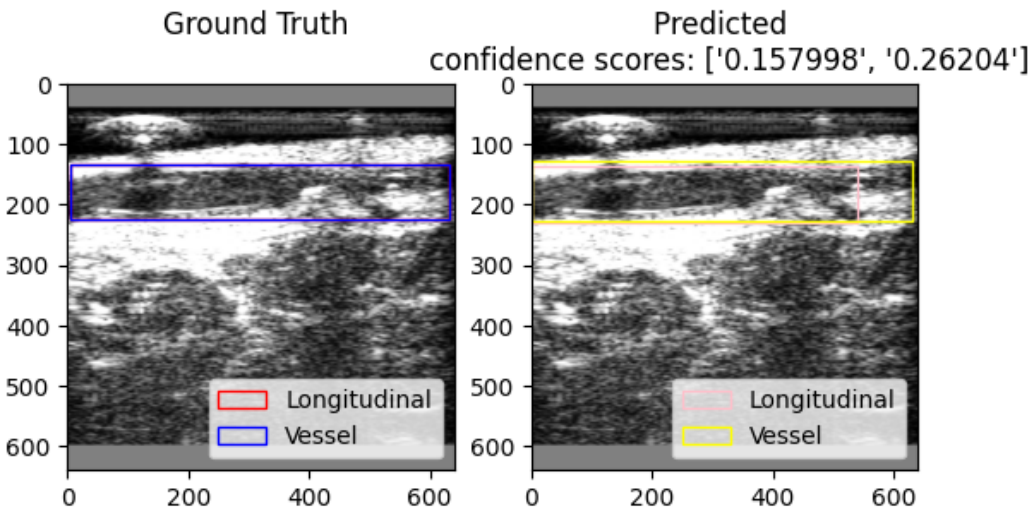
(b)



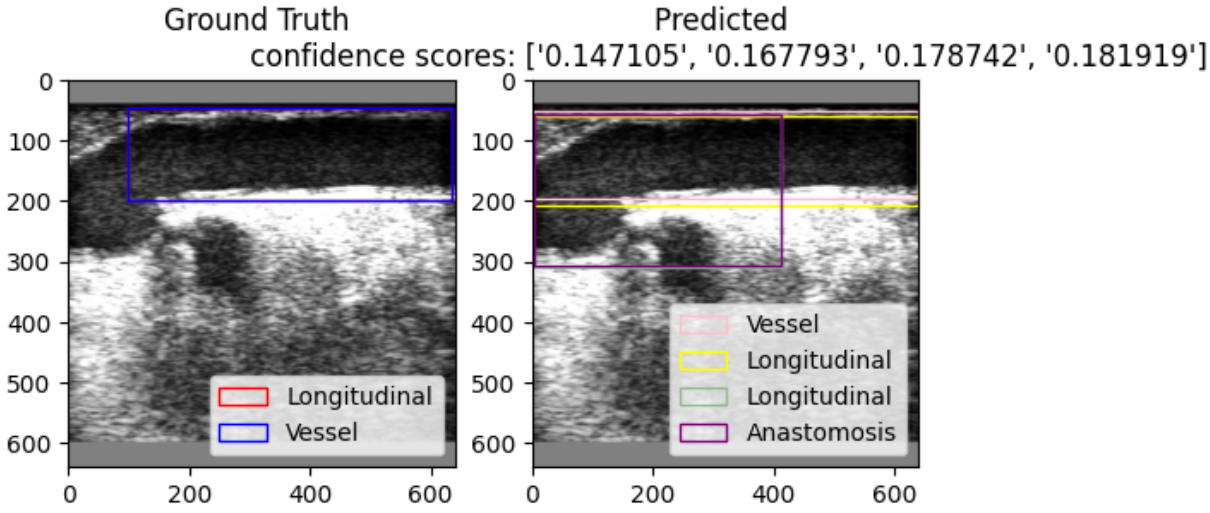
(c)

Figure 4.12: The images from the test dataset with the lowest confidence scores with bounding boxes.

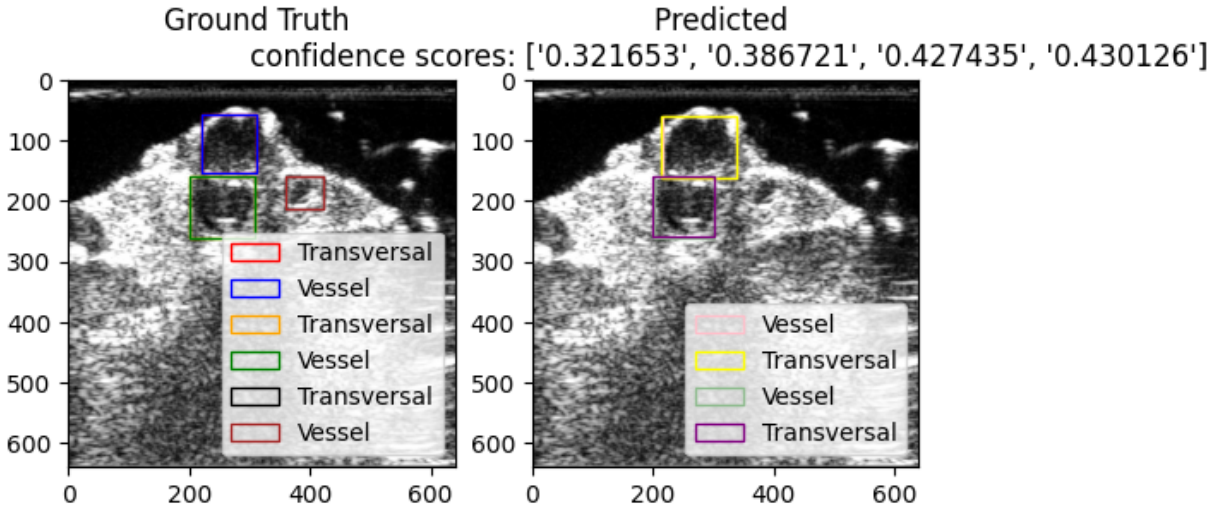
Credit: REQUEST study from Medistim



(a)



(b)



(c)

Figure 4.13: Three random images of predicted bounding boxes along with the confidence scores.

Credit: REQUEST study from Medistim

Chapter 5

Discussion

The goal of this master thesis was to investigate how machine learning could be used on existing intra-operative ultrasound data to contribute to vessel recognition and classifying vessel properties on ultrasound images. Three machine learning models were tested, but only Yolo was used in the final study. The final Yolo model achieved good results after data augmentation and upscaling of the images. The model recognized the vessels with 71.77% mean average precision, identified the vessel orientation with 75.8% mean average precision for transversal images, and 64.6% mean average precision for longitudinal images. The final model solved both tasks with a speed of inference per image at 5.6 milliseconds. The objective of vessel recognition, aiming for 82.10% mean average precision, and the objective of achieving 95% mean average precision for feasibility in clinical practice were not completed. Additionally, the objective of identifying vessel orientation with 80% mean average precision was not achieved. However, the objective of solving all of these tasks in less than 10 milliseconds was successfully accomplished.

In this chapter, the results presented in chapter 4 are discussed, including what went well and what could have been improved. Furthermore, the discussion will include how the results from this master thesis could be used for future research and in clinical practice.

5.1 Final model

The final model achieved an overall precision of 66.04% and recall of 72.94%, with a 6.9% difference. This indicates a balanced performance between accurately predicting

the structures and orientation and capturing the majority of the structures. The overall mean average precision achieved was 71.77%. Even though the mean average precision was less than the objective of this thesis, it is important to acknowledge that this research still holds a significant value.

Achieving a mean average precision of 82.10% would have been an important milestone, as it would have represented an improvement over the current state-of-the-art. However, the target user of the ultrasound systems is typically a nurse or other medical professional rather than a surgeon or diagnostician. As such, the machine learning model is intended to serve as an aid to the system's operator rather than as a diagnostic tool. By incorporating a machine learning algorithm to assist the operator, the ultrasound system progresses towards reducing reliance on the system's operator. While a higher mean average precision is be desirable, it may not be strictly necessary for the purpose of providing effective assistance to the user.

Previous research has yet to use the data from the REQUEST study for machine learning purposes before. Furthermore, this research stands out as one of the first attempts to conduct a multicenter study in the context of intra-operative ultrasound and the application of machine learning techniques. Including data from multiple medical centers enhances the diversity of the results, making them more representative and applicable to real-world scenarios. While the achieved precision, recall, and mean average precision may not have reached the initial objective, it is crucial to recognize that the model can still serve as a valuable supporting tool. These results provide a starting point and valuable insights into the feasibility and potential of machine learning in the context of intra-operative ultrasound.

5.1.1 Results of the individual structures

When analyzing the performance of the object detection model, a small difference and a slight decrease in performance were observed for the test set compared to the validation set. The overall mean average precision for the validation set was 71.85%, while the test set achieved a mean average precision of 71.77%. Although the difference was notably small at 0.8%, it is worth considering the potential implications of this performance decline. Such a small difference could indicate variance in the data in the form of imbalanced class distribution or the possibility of overfitting the model. To maintain the model's overall performance and generalization abilities, it is essential to be aware of the

effects of any potential variances or overfitting. Further exploration of these factors was not done due to the small difference in this case.

Analyzing the individual structures with the results in table 4.7 shows that the results varied. Aorta had the highest mean average precision at 97.1%, while anastomosis had the lowest mean average precision at 51.5%. This indicates that the aorta performed better than the other structures. The vessel and the aorta had a higher recall than precision, with the difference being 1% for the vessel and 29.2% for the aorta. The aorta had a higher precision than all other structures with a precision of 99.6%. A high recall suggests a lower rate of false negatives implying that most of the images of the aorta were predicted as an aorta. The anastomosis was the only structure with a lower recall than precision, with a difference of 7.2%. Since the precision was higher for anastomosis, the model made more accurate positive predictions but still missed a number of actual images of anastomosis.

Considering the orientation of the probe, the transversal orientation had a higher performance in mean average precision. The transversal orientation accomplished a mean average precision of 75.8%, while the longitudinal orientation accomplished a mean average precision of 64.6%. Both the transversal and the longitudinal orientation demonstrated a higher recall than precision.

These results can also be reflected in fig. 4.10 where the precision-recall curve for the aorta is closer to an ideal model with a line closer to the right corner than the other structures. Anastomosis is the structure with the less ideal line in the precision-recall curve. For the orientation of the probe, the transversal orientation is closer to an ideal model than the longitudinal orientation.

As shown in fig. 4.11, the highest confidence score in the test set was 47.14%. According to an article by Microsoft [46], a confidence score between 30-50% is a low confidence, which gives a typical related answer. A confidence between 50-70% is a medium confidence which gives a fairly good answer that should answer the main intent of the object detection model. A confidence score over 70% is a high confidence that gives a good answer. The confidence for the final model on the test set was therefore lower than ideal. The images of the aorta had higher confidence scores than the vessels and anastomosis. This is in accordance with the results for precision, recall, and mean average precision for the labels in table 4.7.

The confidence threshold for the final model was set as default 10%. The model had to be 10% sure that the prediction was right before labeling the prediction. Setting a low

confidence threshold would result in more predictions being accepted, but this could also increase the rate of false positives for the model. A higher confidence threshold would ensure that only certain predictions or classifications would be accepted, which could result in a higher rate of false negatives. Changing the confidence threshold could have an impact on the overall performance of the model. The three images with the lowest confidence scores in the test set are shown in fig. 4.12 were still predicted with the right label, even if the confidence scores were at the threshold of 10%.

5.1.2 Implementation of the final model

The performance of the models on the validation dataset in both the pilot study shown in table 4.1 and the final model shown in table 4.5 shows that the default values of the hyperparameters had the best results most of the time. Because the difference between the results for the default values and the hyperparameter tuning was small, hyperparameter tuning should still be considered to improve the model.

The way the hyperparameter tuning was done in this thesis was also a choice that could have affected the final results. There are two common methods for hyperparameter tuning in machine learning; grid search and random search [35]. Grid search involves selecting a hyperparameter room and testing every possible combination of values. Random search involves selecting hyperparameters randomly from the hyperparameter room and training the models on those values.

One of the advantages of grid search is that it is guaranteed to find the optimal hyperparameters within the hyperparameter room, provided enough computational resources are available. However, grid search can be computationally expensive, particularly when working with a large number of hyperparameters or when the ranges in the hyperparameter room are wide. Random search, on the other hand, has several advantages. Firstly, it is more computationally efficient than grid search, as it only requires running a subset of the possible combinations. Secondly, it has been shown to be effective at finding good hyperparameters, particularly when the number of hyperparameters is large [35]. This thesis used a random hyperparameter search because of limited computational resources and time.

Early stopping was also implemented for the final model due to time constraints and hardware challenges (see section 5.6). Changing the number of epochs without improvement could significantly impact the results. Increasing the patience level could allow for

more epochs without improvement before stopping the training, potentially leading to better performance. Conversely, decreasing the patience level could prevent overfitting and promote generalization. However, early stopping may limit the model's learning potential. Experimenting with the early stopping could have led to better performance for the final model.

5.2 Selection of machine learning task

One of the primary objectives in this thesis was to find a machine learning model that could achieve high performance on ultrasound data and later be implemented on the ultrasound systems of Medistim. Specifically, the two key performance indicators were improving the current state-of-the-art mean average precision with 82.10% and a mean average precision of 95% to make the model feasible to implement in clinical practice. The selection of the machine learning task was based upon the use of the machine learning model as an assistant to the user interface of the ultrasound systems. Object detection with bounding boxes was chosen as the approach for this thesis because of the interest in identifying and localizing the objects in the RoI within the ultrasound images.

Alternatively, segmentation approaches such as U-Net can also be used for vessel recognition on ultrasound images. These algorithms divide the image into smaller regions and identify the vessels within each segment. While they may be less effective at detecting complex vessel structures or identifying vessels in cluttered backgrounds, they can provide highly accurate segmentation maps useful for downstream analysis. Another approach that has been explored is ellipse fitting, as demonstrated by Smistad in 2016 [62]. This method involves fitting ellipses to the vessels in the ultrasound images and using these shapes to extract vessel features for classification. While this approach can be highly effective for specific vessel types, it may be less robust in the face of variability in vessel shape.

Ultimately, the choice of machine learning task will depend on a number of factors, including the specific application, the availability of data, and the desired level of interpretability. Because of the main objective and usage of the machine learning model on the ultrasound systems, the object detection models Yolo, RetinaNet, and EfficientDet were chosen in this thesis.

5.3 Selection of object detection models

This thesis's choice of object detection algorithms was based on two main criteria, achieving a high mean average precision and near real-time prediction capabilities. The models selected were Yolo, RetinaNet, and EfficienDet, which are all state-of-the-art object detection models that have demonstrated high mean average precision on various benchmark datasets.

5.3.1 Selection of Yolo

Yolo is a popular and efficient object detection family with promising state-of-the-art performance, high mean average precision, and fast inference speeds. The latest version at the start of this master thesis, Yolov5, was released in 2020, and updated for the last time at the end of 2022. In September 2022, the paper for Yolov6 was released [19], and in July 2022, the paper for Yolov7 was released [70]. In November 2022, the newest version of Yolov8 was released by the same developers as Yolov5 [21]. The implementation of these versions was not tested as much as Yolov5 at the start of this master thesis. Yolov5 was therefore chosen to be explored further.

Yolov5 is designed to be easily customizable and adaptable to different applications. It has a modular architecture that allows for easy modification and customization, making it suitable for fine-tuning specific datasets. This is particularly useful in medical imaging, where datasets can be diverse and unique. Furthermore, YOLOv5 is well-known for its simplicity and ease of use. The model is lightweight and simple to train, with a simple pipeline that requires little data preprocessing. This makes it accessible to researchers with limited machine learning experience while still delivering state-of-the-art performance. The model has been used in medical imaging before on computed tomography (CT) [25] and magnetic resonance imaging (MRI) [14], but it has also been tested on ultrasound images with good results [13], [73]. This makes Yolov5 a reliable and effective tool for medical image analysis.

5.3.2 Selection of RetinaNet

RetinaNet was also chosen because of its speed and comparative object detection performance. RetinaNet's Feature Pyramid Network (FPN) allows for detecting objects of

varying sizes and resolutions, which is a desirable characteristic for ultrasound images. The hierarchical feature extraction in FPN enables the model to identify objects with high precision, even in noisy ultrasound images. Additionally, RetinaNet employs a focal loss function that addresses the issue of class imbalance, which is often present in medical imaging datasets.

The architecture of RetinaNet is highly efficient, with fast inference speeds and low computational requirements. The model has been tested on ultrasound images with good results [40], [24]. This makes it a practical choice for real-time object detection on ultrasound images.

5.3.3 Selection of EfficientDet

EfficientDet uses a compound scaling technique that balances the model's depth, width and resolution. This enables EfficientDet to efficiently detect objects of different sizes and shapes, which is an important characteristic for ultrasound images. Furthermore, EfficientDet is known for its efficiency, which is important for ultrasound imaging which is a real-time imaging modality.

EfficientDet has also been tested on ultrasound images with promising results [16], [39]. EfficientDet was therefore one of the three models chosen for this master thesis.

5.4 Pilot studies

This master thesis started with three different pilot studies of three different models to test the fit of the dataset on the different models. Pilot studies are a type of exploratory investigation that is frequently carried out prior to the start of major research studies. Pilot studies are designed to verify the feasibility and validity of the study approach, and to identify any potential issues or constraints that may develop during the data collection and analysis phase. In this case, the pilot study was used to help ensure that the final model was well-designed and identify any necessary modifications to the data. Although it would have been ideal to test all models, time and resources were insufficient. By doing the pilot studies on each of the models, a better understanding of their strengths and weaknesses was gained.

The ultimate goal of the pilot studies was to ensure that the final model chosen was appropriate for the objective of this thesis and gain an understanding of how to use it more effectively. Yolo was chosen as the final model after the three pilot studies because of the higher mean average precision and most development in the results during the pilot study.

In the first pilot study of Yolo, the results were not as promising as anticipated. The best model achieved a mean average precision at 50% of 44.40%, which was only half of the desired outcome. Notably, the precision value was considerably lower than the recall, with a difference of 22.74%. This indicates that the model's ability to accurately identify the structures and orientation and classify the objects could have been more consistent compared to its ability to detect objects. Despite achieving less than ideal results, Yolo still outperformed the other models in terms of overall performance, progress, and stability throughout the three pilot studies. The model had the highest level of improvement in the results throughout the pilot study.

The results of the pilot studies of RetinaNet and EfficientDet could have been better. RetinaNet achieved an unexpectedly low mean average precision, despite implementing various techniques, which did not improve. Different batch sizes and the number of epochs were tested but failed to make any enhancements. The best performing RetinaNet model achieved a mean average precision below 10%, indicating that more debugging and data exploration had to be done to use RetinaNet.

EfficientDet had computational challenges from the start, as the model failed to train with the pretrained weights due to memory constraints. After experimenting with different batch sizes and configurations, the first three weights of EfficientDet were successfully used in training. However, the best model of EfficientDet achieved an average precision at 50% of 32.5%. Due to the computational difficulties and time constraints, further exploration of EfficientDet was therefore not done.

A number of factors may have influenced these results. One significant reason for the superior performance of Yolo, compared to the other models, could be the extensive data preprocessing and analysis done for Yolo. Data preparation, which involves cleaning, transforming, and preparing the data for analysis, is an essential step in machine learning. Because Yolo was the primary model of interest before the pilot studies more data and annotation analytics were done for Yolo compared to RetinaNet and EfficientDet. This made it easier to check if the model interpreted the annotation correctly when it was used in the model.

Another factor that could have impacted the pilot studies is the configuration of the machine learning models. The performance of a model is heavily dependent on various configuration settings such as hyperparameters, number of layers, and activation functions used. Improper configuration of the models could have contributed to their poor performance.

Hardware and resource limitations could have played a role in the incomplete pilot experiments, particularly in the case of EfficientDet. Significant computational resources, such as specialized hardware or extensive amounts of memory, may be required to train and evaluate machine learning algorithms. Unfortunately, because this was not a controllable variable in the thesis project, the system may not have had the resources required to run the model properly

5.5 Dataset challenges

One of the primary challenges in using ultrasound data for object detection was the inconsistency in the data itself. The quality of ultrasound images could vary significantly based on the patient's body type, the operator's experience, and the equipment used. The images varied in resolution, contrast, and signal-to-noise ratio. These variations can make it challenging to develop object detection models that can be generalized to serve diverse ultrasound images. In particular, a low signal-to-noise ratio can make identifying and distinguishing small or low-contrast objects challenging. Another challenge was operator inconsistency, which can be caused by a lack of ultrasound experience or training. In some cases, cardiac surgeons may not have received adequate ultrasound training, which can lead to inconsistencies in the collected data, such as variations in imaging settings or scan planes.

In ultrasound data, the objects in RoI like anastomosis, aorta, and vessels, are less common than background images without any object of interest. This leads to class imbalance and could make it difficult to train models that can accurately detect these objects. Data augmentation was done to address the class imbalance and create a more diverse training dataset in this thesis.

Upon reviewing the results from pilot study 1 of Yolo in table 4.1 and pilot study 2 of Yolo in table 4.4, the mean average precision at 50% increased from 44.40% to 69.96%. This improvement indicates that the implementation of data augmentation techniques

positively impacted the performance of Yolo. After resizing, upscaling and intensifying the data augmentation, the mean average precision at 50% increased from 69.96% to 71.85% on the validation data. Comparing the improvements achieved between the pilot studies and the final model shows that the initial data augmentation had a more substantial impact on enhancing the model's performance than the full data augmentation implementation with resizing and upscaling.

The data augmentation techniques, such as rotation and horizontal and vertical flip, were less effective for the machine learning model. In fact, studies by Tirindelli et al. [23] and Wulff et al. [74] have highlighted that applying these techniques could result in unrealistic ultrasound images inconsistent with the physical model of attenuation and displacement between the transducer's location and the image content. These factors can create shadow regions and gaps between the wave source and reflective tissue, leading to highly unrealistic synthetic samples. This was also why the hyperparameters for flipping the image up and down, left and right, led to worse results when these were not set to 0. Wulff et al. also mention how contrast could be an issue with data augmentation on ultrasound images. However, some ultrasound systems, including the systems the images in this thesis were taken from, have contrast adjustments as one of the on screen controls. Different systems can also use different presets with different contrast adjustments. Contrast adjustments would in our case not create an unrealistic image and were therefore adjusted in the data augmentation part in this thesis.

In addition to the previously discussed challenges, another challenge faced by the ultrasound dataset used in this thesis was the difference in size between the images of the aorta and the images of the anastomosis and vessels, based on the depth set for the ultrasound probe. Such differences in image size can create difficulties in developing models that can accurately detect objects across a range of ultrasound images. However, it is noteworthy that operators typically adjust their imaging modality to visualize the aorta accurately. Therefore, in this thesis, the size of the aorta was left as it is to avoid altering the imaging modality and compromising the accuracy of the ultrasound data.

5.5.1 Annotation noise factors

Annotation noise factors were also one of the main challenges during this master thesis. The annotation of the ultrasound images in the dataset was carried out by three different people. This approach can have advantages and disadvantages in terms of the quality and reliability of the dataset. Multiple annotators can result in a more diverse and

representative dataset, as different people may interpret the images in differently based on their expertise, experience, and personal biases. This helps capture the real-world variability and complexity of ultrasound images, which is critical for developing object detection models that can generalize to different settings and populations. Furthermore, incorporating multiple annotations per image can help improve the precision and consistency of the dataset, as disagreements and errors can be identified and resolved through agreement.

On the other hand, involving multiple annotators can also introduce inconsistencies and errors in the dataset, particularly if there is no clear standard or guideline for annotation. The annotation of ultrasound images requires careful attention to detail, as small differences in the labeling or measurement of features can have significant implications for the precision and usefulness of the dataset.

The annotation in this thesis was done by non-experts and non-clinicians, and the level of experience of the annotators could affect the quality of the annotations. Less experienced annotators may miss subtle or complex features in the ultrasound image, leading to inconsistencies and subjective interpretation of the images. Most of the papers mentioned in section 1.3 had annotations performed by experts [34], [65], [76], [33],[28], [3], [18]. Generally, annotations on ultrasound images are performed by trained clinicians. Performance by non-experts could lead to mislabeling and misinterpretation of image features and introduce noise into the dataset.

As shown in fig. 4.9, the annotation of the ultrasound images in this thesis led to some variation in how the images were labeled. This noise could impact the performance of object detection models trained on the dataset, as the models may learn to detect or classify features that are not relevant or representative of the true underlying patterns in the images.

To address these challenges, it is important to carefully consider the annotation process and develop clear guidelines and standards for annotation. This helps to ensure that the dataset is consistent, accurate, and representative of the real-world variability of ultrasound images. Ultimately, the goal of annotation should be to minimize the possibility of annotation noise and other kinds of bias or mistakes while producing a high-quality dataset that can assist with the development of accurate and dependable object detection models for ultrasound imaging.

5.5.2 Ultrasound noise factors

One major source of noise in ultrasound images is the presence of artifacts, which can arise from various factors, such as the physical properties of the tissue being imaged, the configuration of the ultrasound machine, and the positioning and movement of the transducer. These artifacts can create false signals or distortions in the image, which can in turn, lead to inaccurate or misleading object detection results.

One common artifact in ultrasound imaging is mirroring, which occurs when the ultrasound beam reflects off a strong interface, such as a bone, and produces a mirror image of the object. This can create false positives in object detection, as the algorithm may mistakenly identify the mirror image as a real object. Similarly, reverberation is an artifact that occurs when the ultrasound beam bounces back and forth between two strong interfaces, producing a series of overlapping echoes that can obscure or distort the true image. This can make it difficult for object detection algorithms to accurately identify the position and size of objects in the image.

Given the potential for artifacts to create noise in ultrasound images, it is important to consider their impact on object detection algorithms. For example, the presence of mirroring or reverberation artifacts can create false positives or make it difficult to distinguish between real and false objects, while attenuation or speckle artifacts can reduce the signal-to-noise ratio and make it more challenging to detect and localize objects accurately.

5.6 Hardware challenges

During the master thesis, various hardware challenges were encountered while working with the Birget high-performance 3D controller provided by the Department of Informatics at the University of Bergen. The hardware was shared among multiple master students at the faculty. Unfortunately, the hardware's stability posed significant obstacles throughout the thesis, by frequent overloading leading to system downtime.

The initial method of this thesis involved the need to familiarize oneself with the hardware and learn how to write slurm bash files to utilize it effectively. However, issues arose from the start as the hardware experienced frequent downtime. This resulted in slow training

of the object detection models and constant interruptions. These challenges persisted during the first half of the year, adversely affecting the progress of the work.

Although the hardware slightly improved stability during the latter half of the year, the situation worsened significantly in the final three months of the thesis. Periods of downtime became a recurring problem, with periods of downtime lasting over 14 days. This ongoing problem severely impacted the training, evaluation, and technical aspects of the master thesis, causing disruptions and delays.

As a result, the instability and overload of the hardware directly impacted the effectiveness and advancement of the study. The unpredictable performance of the technology remained a persistent challenge despite deliberate efforts to reduce the impact on the thesis, such as looking for alternate solutions and changing methods. Throughout the thesis, careful management and emergency preparation were necessary to overcome the difficulties brought on by the overloaded hardware and guarantee the accomplishment of the research.

5.6.1 Improvement of the state of art

Out of all of the articles mentioned in section 1.3, only one article by Iriani et al. was about object detection on ultrasound data [18]. This was also the only article using mean average precision as a performance measure. Iriani et al. used a newer version of Yolo, Yolov7, and achieved a mean average precision of 82.10%. Frames from 60 ultrasound videos were used in the article by Iriani et al. In this thesis, 12 850 videos were used, leading to more unique frames than in the study by Iriani et al.

The difference between the data in the article by Irani et al. and the REQUEST data used in this thesis is mainly that the REQUEST data was from a multicenter study. The images were taken by multiple operators and surgeons, and on different ultrasound systems. The results may have been impacted by the larger number of videos in the REQUEST data compared to the data in the article by Irani et al.

One potential approach to enhance the results and improve mean average precision beyond that reported by Irani et al. with the REQUEST data would be to adopt a transfer learning methodology. This would involve initially training the model on ultrasound images from a single ultrasound machine and center. The model could then be further trained on ultrasound images obtained from other centers by fine-tuning the weights obtained from this initial training. This sequential training process would enable the model

to adapt and generalize better across different centers, accounting for the variations in image acquisition techniques and equipment used.

Additionally, considering the larger number of videos in the REQUEST data compared to the dataset in the article by Irani et al, it is crucial to examine the impact of video frame selection. Exploring different strategies for frame extraction, such as selecting key frames or employing a more diverse set of frames throughout the video, may yield improved results. The model could potentially learn more robust and representative features by capturing a broader range of vessel orientations and anatomical structures.

The majority of the other articles mentioned in section 1.3 used accuracy as the performance measure, which can not be compared to mean average precision. Classification was also the primary goal of most of them and not object detection. As shown in table 1.1, most of the articles mentioned had significantly fewer images in the dataset than in this thesis. The articles with the same amount of images [7], [65], [76], [62] had images coming from one hospital and were not a multicenter study such as the REQUEST study.

5.7 Future work

The results of this master thesis provide a good starting point for further development of machine learning applications on intra-operative ultrasound images. One of the key findings of this study is that thorough preprocessing of the dataset could have yielded better results. A suggestion for further work could be to develop a robust preprocessing pipeline specifically tailored to ultrasound images, including removal of noise or artifacts that could affect the model's performance negatively, and include data augmentation as part of this pipeline to balance the dataset.

The next step in this research project would have been to explore RetinaNet and EfficientDet more with the same dataset. A more custom preprocessing of the dataset to each network, and more data exploration around the two models could lead to more combarable results to Yolo.

Another step for further work is to explore object detection on ultrasound videos. Since the goal of introducing machine learning with ultrasound is to increase the effectiveness of ultrasound systems, object detection on real-time videos would be particularly useful. Yolov5 can be used on inference of mp4 videos [20], allowing the results of this master thesis on static images to be used for further research on videoes. The study done by

Iriani et al. in December 2022 [18] used Yolov7 for real-time detection of cardiac objects with fetal ultrasound video, achieving a mean average precision of 82.10%. Exploring this field with newer versions of Yolo on the REQUEST dataset could improve the state of the art of object detection on ultrasound videos.

However, there are several challenges with object detection on ultrasound videos. First, as Yolo is trained on images, the ultrasound videos would have to be extracted into a dataset of individual frames before beginning training. Additionally, accurate labeling can be difficult due to ultrasound videos' low contrast and poor image quality, making it time-consuming and difficult to annotate. Finally, real-time processing of ultrasound videos can be computationally demanding, which could limit the practicality of using Yolo for object detection on real-time ultrasound.

In addition to exploring object detection on ultrasound videos, another direction for further research is to implement the object detection model on ultrasound systems. This would involve integrating the trained model into the ultrasound system to allow real-time object detection during ultrasound examinations. The implementation of object detection on ultrasound systems would require addressing some technical challenges. For example, the computational requirements of running the object detection model in real-time on an ultrasound system may be quite high, so optimizing the model's architecture for deployment on such systems would be crucial. Additionally, the model would need to be integrated into the ultrasound system's workflow, which could involve modifying the system's software and hardware components.

Despite these challenges, real-time ultrasound videos with object detection present interesting possibilities for using ultrasound systems. In comparison to static imaging modalities, real-time ultrasound with object detection can offer a more dynamic and thorough view of anatomical structures and pathology, which could increase diagnostic precision and guide treatment choices. Future research should therefore look into ways to maximize the computational resources required for real-time processing and ways to get over the difficulties associated with training and implementation of machine learning on ultrasound systems.

5.8 Conclusion

This master thesis explored the application of machine learning techniques on existing ultrasound data for vessel recognition and classification of vessel properties. A pilot study with three different object detection models was tested, with Yolo emerging as the model with the best performance and further investigation.

Although the final model did not achieve the intended objectives regarding the mean average precision for vessel recognition or vessel orientation, the final model successfully accomplished the goal of speed optimization. Despite the setback, the research conducted in this thesis holds significant value and relevance. Notably, this research is the pioneering work utilizing a dataset derived from a multicenter study for machine learning on ultrasound images. Additionally, it represents the first research to use machine learning techniques on intra-operative ultrasound images. These aspects contribute to the significance of the findings and the advancement of knowledge in this field.

While the results may have yet to improve upon the existing state of the art, the insights gained through this thesis serve as a vital starting point for future research. The feasibility and potential of machine learning in the context of intra-operative ultrasound have been illuminated, offering valuable groundwork for further exploration and development of machine learning on intra-operative ultrasound.

Bibliography

[1] P. I. Aaronson. *The cardiovascular system at a glance*. At a glance series. Wiley-Blackwell, Chichester, 4th ed. edition, 2013.

[2] M. AS. Medistim ultrasound imaging probe intraoperative surgical guidance. `https://medistim.com/our_products/high-frequency-imaging-probe/`, 2020. Accessed 16-11-2022.

[3] C. F. Baumgartner, K. Kamnitsas, J. Matthew, S. Smith, B. Kainz, and D. Rueckert. Real-time standard scan plane detection and localisation in fetal ultrasound using fully convolutional neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9901 LNCS:203–211, 10 2016.

[4] D. Bell and R. Nightingale. Ultrasound transducer. *Radiopaedia.org*, 6 2017. Accessed 31-10-2022.

[5] L. J. Brattain, B. A. Telfer, M. Dhyani, J. R. Grajo, and A. E. Samir. Machine learning for medical ultrasound: status, methods, and future opportunities. *Abdominal Radiology*, 43:786–799, 4 2018.

[6] G. Carneiro, J. C. Nascimento, and A. Freitas. The segmentation of the left ventricle of the heart from ultrasound data using deep learning architectures and derivative-based search methods. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 21:968–982, 3 2012.

[7] H. Chen, Q. Dou, D. Ni, J. Z. Cheng, J. Qin, S. Li, and P. A. Heng. Automatic fetal ultrasound standard plane detection using knowledge transferred recurrent neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9349:507–514, 2015.

[8] O. Chernytska. Training yolo? select anchor boxes like this. `https://towardsdatascience.com/training-yolo-select-anchor-boxes-like-this-3226cb8d7f0b`, 8 2022. Accessed 03-03-2023.

[9] A. Christiansen. Anchor boxes — the key to quality object detection. https://towardsdatascience.com/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9, 10 2018. Accessed 22-02-2023.

[10] W. Commons. Dilated cardiomyopathy m-mode.jpg. https://commons.wikimedia.org/wiki/File:Dilated_cardiomyopathy_M-Mode.jpg, 2005. Accessed 27-04-2023.

[11] W. Commons. Diagram of the human heart (cropped). https://commons.wikimedia.org/wiki/File:Diagram_of_the_human_heart_(cropped).svg, 2006. File: Diagram of the human heart (cropped).svg, Accessed 07-10-2022.

[12] W. Commons. Ultrasound amplitude scan.jpg. https://commons.wikimedia.org/wiki/File:Ultrasound_Amplitude_scan.png, 2020. Accessed 27-04-2023.

[13] E. Dandil and T. et al. Fetal movement detection and anatomical plane recognition using yolov5 network in ultrasound scans. *Avrupa Bilim ve Teknoloji Dergisi*, pages 208 – 216, 2021.

[14] N. M. Dipu, S. A. Shohan, and K. M. Salam. Deep learning based brain tumor detection and classification. *2021 International Conference on Intelligent Technologies, CONIT 2021*, 6 2021.

[15] S. Doroudi. The bias-variance tradeoff: How data science can inform educational debates. *AERA Open*, 6(4):2332858420977208, 2020.

[16] R. Du, Y. Chen, T. Li, L. Shi, Z. Fei, and Y. Li. Discrimination of breast cancer based on ultrasound images and convolutional neural network. *Journal of Oncology*, 2022, 2022.

[17] S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2 2017.

[18] A. I. S. et al. Deep learning-based real time detection for cardiac objects with fetal ultrasound video. *Informatics in Medicine Unlocked*, 36:101150, 1 2023.

[19] C. L. et al. Yolov6: A single-stage object detection framework for industrial applications, 2022.

[20] G. J. et al. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, Nov. 2022.

[21] G. J. et al. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, Nov. 2022.

[22] H. G. et al. fizyr/keras-retinanet 0.5.1, June 2019.

[23] T. et al. Rethinking ultrasound augmentation: A physics-inspired approach. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12908 LNCS:690–700, 2021.

[24] T. T. et al. The feasibility to use artificial intelligence to aid detecting focal liver lesions in real-time ultrasound: a preliminary study based on videos. *Scientific Reports 2022 12:1*, 12:1–12, 5 2022.

[25] H. A. Ewaidat and Y. E. Brag. Identification of lung nodules ct scan using yolov5 based on convolution neural network, 2022.

[26] J. Feng and S. Lu. Performance analysis of various activation functions in artificial neural networks. *Journal of Physics: Conference Series*, 1237(2):022030, jun 2019.

[27] P. S. Foundation. shutil — high-level file operations — python 3.10.7 documentation. `https://docs.python.org/3/library/shutil.html`, 2022. Accessed 07-10-2022.

[28] X. Gao, W. Li, M. Loomes, and L. Wang. A fused deep learning architecture for viewpoint classification of echocardiography. *Information Fusion*, 36:103–113, 11 2017.

[29] Y. Gao, M. A. Maraci, and J. A. Noble. Describing ultrasound video content using deep convolutional neural networks. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 787–790, 2016.

[30] Y. C. Ho and D. L. Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications*, 115:549–570, 2002.

[31] B. Ihnatsenka and A. P. Boezaart. Ultrasound: Basic understanding and learning the language. *International Journal of Shoulder Surgery*, 4:55, 7 2010.

[32] T. Jo. *Machine Learning Foundations*. Springer International Publishing, 1 edition, 2 2021.

[33] H. Khamis, G. Zurakhov, V. Azar, A. Raz, Z. Friedman, and D. Adam. Automatic apical view classification of echocardiograms using a discriminative learning dictionary. *Medical image analysis*, 36:15–21, 2 2017.

[34] A. Kumar, P. Sridar, A. Quinton, R. K. Kumar, D. Feng, R. Nanan, and J. Kim. Plane identification in fetal ultrasound images using saliency maps and convolutional neural networks. *undefined*, 2016-June:791–794, 6 2016.

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

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

[37] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:318–327, 8 2017.

[38] S. J. Ling, J. Sanny, W. Moebs, O. College, and O. T. Library. *University physics. Volume 1*, volume 1. OpenStax, 9 2016.

[39] G. Liu, J. Tan, H. Yang, Y. Li, X. Sun, J. Wu, and B. Luo. Breast ultrasound tumor detection based on active learning and deep learning. *Communications in Computer and Information Science*, 1700 CCIS:1–10, 2022.

[40] G. S. Liu, P. Y. Huang, S. S. Zhuang, X. P. He, M. L. Wen, and J. Hua. Application of endoscopic ultrasonography for detecting esophageal lesions based on convolutional neural network. *World Journal of Gastroenterology*, 28:2457, 6 2022.

[41] F. Lundh. Pillow (pil fork) 9.2.0 documentation. `https://pillow.readthedocs.io/en/stable/`, 2022. Accessed 07-10-2022.

[42] J. Markowitz. *Probe Selection, Machine Controls, and Equipment*, chapter Chapter 4. The McGraw-Hill Companies, New York, NY, 2011.

[43] D. J. Martin, I. T. Wells, and C. R. Goodwin. Physics of ultrasound. *Anaesthesia & Intensive Care Medicine*, 16:132–135, 3 2015.

[44] K. Matre and O. H. Gilja. Medical imaging, 2011.

[45] U. Michelucci. *Object Classification: An Introduction*, pages 195–220. Apress, Berkeley, CA, 2019.

[46] Microsoft. The confidence score of an answer. `https://learn.microsoft.com/en-us/azure/cognitive-services/qnamaker/concepts/confidence-score`, 2021. Accessed 09-05-2023.

[47] O. A. Montesinos López, A. Montesinos López, and J. Crossa. *Fundamentals of Artificial Neural Networks and Deep Learning*, pages 379–425. Springer International Publishing, Cham, 2022.

[48] A. Murphy and M. Morgan. Acoustic impedance. *Radiopaedia.org*, 11 2014.

[49] NHS. Coronary artery bypass graft (cabg) - nhs. https://www.nhs.uk/conditions/coronary-artery-bypass-graft-cabg/, 11 2021. Accessed 28-11-2022.

[50] I. NumFOCUS. pandas.dataframe — pandas 1.5.0 documentation. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html, 2022. Accessed 07-10-2022.

[51] T. E. of Encyclopaedia Britannica. heart — structure, function, diagram, anatomy, & facts — britannica. https://www.britannica.com/science/heart, 8 2022. Accessed 22-09-2022.

[52] W. H. Organization. Cardiovascular diseases. https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1, 2023. Accessed 02-03-2023.

[53] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *International Journal for Research in Applied Science and Engineering Technology*, 10:943–947, 11 2015.

[54] R. Padilla, S. L. Netto, and E. A. B. da Silva. A survey on performance metrics for object-detection algorithms, 2020.

[55] PapersWithCode. https://paperswithcode.com/paper/efficientdet-scalable-and-efficient-object. Accessed 21-04-2023.

[56] R. Parmar. Common loss functions in machine learning. https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23, 9 2018. Accessed 15-12-2022.

[57] R. Ranjan. Overfitting and underfitting in machine learning. https://towardsdatascience.com/overfitting-and-underfitting-in-machine-learning-89738c58f610, 4 2020. Accessed 12-12-2022.

[58] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:779–788, 6 2015.

[59] G. A. Roth, C. O. Johnson, K. H. Abate, F. Abd-Allah, M. B. Ahmed, K. Alam, T. Alam, N. Alvis-Guzman, H. Ansari, J. Ärnlöv, et al. The burden of cardiovascular diseases among us states, 1990-2016. *JAMA Cardiology*, 3(5):375–389, 2018.

[60] I. H. Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2:1–20, 11 2021.

[61] D. scikit learn. sklearn.model_selection.train_test_split — scikit-learn 1.1.2 documentation. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`, 2022. Accessed 07-10-2022.

[62] E. Smistad and L. Løvstakken. Vessel detection in ultrasound images using deep convolutional neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10008 LNCS:30–38, 2016.

[63] D. E. Spicer, D. J. Henderson, B. Chaudhry, T. J. Mohun, and R. H. Anderson. The anatomy and development of normal and abnormal coronary arteries. *Cardiology in the Young*, 25(8):1493–1503, 2015.

[64] A. V. Srinivasan. Stochastic gradient descent. `https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31`, 9 2019. Accessed 09-01-2023.

[65] V. Sundaresan, C. P. Bridge, C. Ioannou, and J. A. Noble. Automated characterization of the fetal heart in ultrasound images using fully convolutional neural networks. *Proceedings - International Symposium on Biomedical Imaging*, pages 671–674, 6 2017.

[66] A. R. Sutanto and D. K. Kang. A novel diminish smooth l1 loss model with generative adversarial network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12615 LNCS:361–368, 2021.

[67] D. P. Taggart, D. J. Thuijs, G. D. Giammarco, J. D. Puskas, D. Wendt, G. D. Trachiotis, T. M. Kieser, A. P. Kappetein, and S. J. Head. Intraoperative transit-time flow measurement and high-frequency ultrasound assessment in coronary artery bypass grafting. *The Journal of Thoracic and Cardiovascular Surgery*, 159:1283–1292.e2, 4 2020.

[68] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:10691–10700, 5 2019.

[69] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10778–10787, 11 2019.

[70] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 7 2022.

[71] S. Weum. Ultralyd for dummies. `http://www.radiolog.no/wp-content/uploads/2015/08/161020-Ultralyd-for-dummies.pdf`, 2002. Accessed 21-09-2022.

[72] S. Windecker and P. K. E. al. 2014 esc/eacts guidelines on myocardial revascularization: The task force on myocardial revascularization of the european society of cardiology (esc) and the european association for cardio-thoracic surgery (eacts)developed with the special contribution of the european association of percutaneous cardiovascular interventions (eapci). *European Heart Journal*, 35:2541–2619, 10 2014.

[73] H. Wu, B. Wu, S. He, and P. Liu. Congenital heart defect recognition model based on yolov5. *Proceedings of the International Conference on Anti-Counterfeiting, Security and Identification, ASID*, 2022-December:148–151, 2022.

[74] D. Wulff, M. Mehdi, F. Ernst, and J. Hagenah. Cross data set generalization of ultrasound image augmentation using representation learning: A case study. *Current Directions in Biomedical Engineering*, 7(2):755–758, 2021.

[75] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen. Image data augmentation for deep learning: A survey, 2022.

[76] M. Yaqub, B. Kelly, A. T. Papageorghiou, and J. A. Noble. Guided random forests for identification of key fetal anatomy and image categorization in ultrasound scans. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9351:687–694, 2015.

[77] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren. Distance-iou loss: Faster and better learning for bounding box regression. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pages 12993–13000, 11 2019.

[78] B. Zoph, E. D. Cubuk, G. Ghiasi, T. Y. Lin, J. Shlens, and Q. V. Le. Learning data augmentation strategies for object detection. *Lecture Notes in Computer Science*

(including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12372 LNCS:566–583, 2020.

[79] Zylo117. Yet-another-efficientdet-pytorch: The pytorch re-implement of the official efficientdet with sota performance in real time and pretrained weights. `https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch`, 2020. Accessed 21-04-2023.

Appendix A

Programming code

All of the code can be found at this Github link:
`https://github.com/juliacv/Masterthesis.git`

The data used in this thesis belongs to Medistim and can not be made publicly available. As a result, running the code in this repository with the original data is not possible. In order to run the code, one must clone the repositories for Yolov5 [20], RetinaNet [22], and EfficientDet [79] and add the appropriate data for each model. The validation code for Yolo and RetinaNet are slightly changed, and can be found in the git repository. Conda enviroment files can also be found in git.

The Preprocessing folder contains separate codes for Yolo, RetinaNet, and EfficientDet. These codes are responsible for splitting the data and performing data augmentation. In addition, the exploratory data analysis files contain code for visualizing the bounding boxes, testing the resizing and data augmentation, and creating pie charts to analyze the data.

Lastly, the Birget folder includes the slurm controller and Python files necessary to run the networks after cloning the original repositories and adding the data for each model. Folder names and paths have to be changed to be able to run the codes.

Read the *README.md* file for more information of the most important files.