Master's thesis

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

Amanda Kathrine Jansen

Automatic annotation of structures in echocardiography using deep learning

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Lasse Løvstakken
June 2021

NTNU
Norwegian University of
Science and Technology

Amanda Kathrine Jansen

Automatic annotation of structures in echocardiography using deep learning

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Lasse Løvstakken
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Abstract

Today, the cardiac ultrasound examination is typically performed by a clinician who has acquired specialized training in the interpretation of ultrasound images. As a result, non-experts may find it challenging to utilize echocardiography. Receiving instant feedback on which structures are in the frame during the examination can benefit the user in interpreting the internal view of the human heart.

This master thesis explores the use of deep learning to automatically detect the different structures of the heart in cardiac ultrasound images. The object detection network YOLO version 5 is implemented, trained and assessed on a dataset containing echocardiography images in the apical two-chambers, four-chambers and long-axis cardiac views. For simplicity, the structures used were the left ventricle, left atrium and mitral valve. The data was provided from 62 different patients, which included 195 recordings, resulting in a total of 1260 images with corresponding ground truth annotations. This thesis intends to obtain a robust object detection model which can be integrated into the ultrasound examination. Therefore, implementations and analyses are performed in order to find the best model capable of detecting the cardiac structures in all types of situations.

The results achieved from the best model are a mean average precision of 0.984 for an IoU equal to 0.5 and 0.631 for an IoU in the interval of 0.5 to 0.95. In addition, the detection gave a confidence of 82% on LV, 84% on MV and 94% on LA in the apical long-axis view, 67% on LV, 82% on MV and 69% on LA in the apical two-chambers view and 88% on LV, 77% on MV and 78% on LA in the apical four-chambers view. However, the model can fail to locate the structures in cases where the image quality is poor and other structures are in focus. As a conclusion the model shows promising results in detecting the structures. The performance and robustness can be increased with putting more work in data preprocessing in addition to experimenting more with data augmentation.

Sammendrag

I dag utføres ultralydundersøkelse av hjertet vanligvis av en lege som har tilegnet seg spesialisering i tolkning av ultralydbilder. Som et resultat kan det være utfordrende for ikke-eksperter å bruke ekkokardiografi. Å motta en rask tilbakemelding på hvilke strukturer i hjertet som er på bildet under undersøkelsen, vil være en god hjelp for brukeren til å tolke hjertets anatomi.

Denne masteroppgaven utforsker bruken av dyp læring for automatisk å oppdage de forskjellige strukturene i hjertet i ultralydbilder. Objektdeteksjonsnettverket YOLO versjon 5 er implementert, trent og vurdert på et datasett som inneholder ekkokardiografibilder i de apikale 2-kammer, 4-kammer og langakse visningene. For enkelhets skyld ble de tre strukturene brukt: venstre ventrikkel (LV), venstre atrium (LA) og mitralventil (MV). Dataen var hentet fra 62 ulike pasienter, som inkluderte 195 opptak, noe som resulterte i totalt 1260 bilder med tilsvarende fasitannoteringer (eng: ground truth annotations). Hensikten med oppgaven er å skaffe en robust objektdeteksjonsmodell som kan integreres i ultralydundersøkelsen av hjertet. Derfor utføres implementeringer og analyser for å finne den beste modellen som er i stand til å oppdage hjertestrukturene i alle typer scenarier.

Resultatene oppnådd fra den beste modellen er en gjennomsnittlig snittpresisjon (eng: mean average precision) lik 0.984 for en IoU lik 0.5 og 0.631 for en IoU i intervallet mellom 0.5 og 0.95. I tillegg, ga resultatet i deteksjonen en konfidens på 82% på LV, 84% på MV og 94% på LA i den apikale langaksevisningen, 67% på LV, 82% på MV og 69% på LA i den apikale 2-kammervisningen og 88% på LV, 77% på MV og 78% på LA i den apikale 4-kammervisningen. Derimot kan modellen mislykkes med å finne strukturene i tilfeller der bildekvaliteten er dårlig og/eller at andre hjertestrukturer er i fokus. Som en konklusjon viser modellen lovende resultater for å oppdage de ulike hjertestrukturene. Ytelsen og robustheten kan økes ved å legge mer arbeid i data prosesseringen, i tillegg til å eksperimentere mer med data augmentering.

Preface

I want to start by thanking the Department of Circulation and Medical Imaging for making this master thesis possible by providing relevant data for the experiments and giving me access to the remote server. A special gratitude goes to David Pasdeluop (Ph. D., CIUS) for his assistance with data preparation, data annotations, implementation of the deep learning algorithm, and helpful comments and advice throughout this semester. Finally, I wish to thank my supervisor Lasse Løvstakken for his encouragement, guidance and support during this master thesis.

Table of Contents

List of Tables

List of Figures

Abbreviations

A2C	=	Apical two chamber
A4C	=	Apical four chamber
AI	=	Artificial intelligence
ALAX	=	Apical long axis
ANN	=	Artificial neural network
AoO	=	Area of overlap
AoU	=	Area of union
AoV	=	Aortic valve
AP	=	Average precision
B-mode	=	Brightness modality
CNN	=	Convolutional neural network
COCO	=	Common objects in context
CSP	=	Cross stage partial
CSPnet	=	Cross stage partial network
DA	=	Data augmentation
DL	=	Deep learning
ECG	=	Electrogardiogram
ED	=	End of diastole
EDV	=	End of diastole volume
ES	=	End of systole
ESV	=	End of systole volume
FNN	=	Feed-forward neural network
FP	=	False positive
FPS	=	Frames per second
FN	=	False negative
HUNT	=	The Trøndelag Health Study
IoU	=	Intersection over Union
LA	=	Left atrium
LV	=	Left ventricle
mAP	=	Mean average precision
ML	=	Machine learning
MLP	=	Multilayer perceptron
MV	=	Mitral valve
NMS	=	Non maximum suppression
NN	=	Neural network
PANet	=	Path aggregation network
PLAX	=	Parasternal long axis
PSAX	=	Parasternal short aixs
RA	=	Right atrium

ReLU	=	Rectified linear unit
RGB	=	Red Green Blue
RV	=	Right ventricle
SPP	=	Spatial pyramid pooling
TN	=	True negative
TP	=	True positive
SPP	=	Spatial pyramid pooling
TN	=	True negative
TP	=	True positive
TTE	=	Transthoracic echocardiography
YOLO	=	You Only Look Once

Chapter 1

Introduction

1.1 Motivation and background

Echocardiography, often referred to as cardiac ultrasound, has been the most widely accessed medical imagining method for examining cardiac function and anatomy. A commonly used cardiac ultrasound method is the Transthoracic echocardiography (TTE). TTE uses a probe, often termed as a transducer, which is placed on a patient's chest and captures the reflections from the human heart and then produces an image [1, Chapter 12.7.5]. The images produced by this technique can be used by clinicians to determining the health of the heart muscle, identifying abnormalities and diagnosing various cardiac disorders [2]. For that reason, it is critical that the ultrasound examination generates anatomically accurate images in order to achieve adequate measures so the clinicians can make the right diagnosis. These days, the cardiac ultrasound examination is typically performed by a radiologist. Radiologists are doctors who have received special training in the analysis of ultrasound images [3]. As a result, non-experts may struggle to utilize echocardiography. For starters, accurately positioning the probe might be challenging. Furthermore, interpreting the structures of the cardiac anatomy from the echocardiograpic image can be confusing and hard to evaluate. Receiving feedback on which structures are in the frame during the examination could give valuable guide to the user in interpreting the structures of the human heart.

In recent years, deep learning (DL) techniques have emerged as an essential aspect in medical imaging. Multiple articles point out that DL methods can help to improve medical image analysis and processing. For instance, Kim et al. [4] stated that "the deep learning is expected to help radiologists provide a more exact diagnosis, by delivering a quantitative analysis of suspicious lesions, and may also enable a shorter time in the clinical workflow." Furthermore, the Cuocolo et al. [5] also stated that the use of machine learning (ML)

can aid in early detection and correct interpretation of findings. This indicates that DL can benefit clinicians in detecting diagnoses earlier, as well as making medical imaging techniques more robust for non-experts to utilize.

There are different DL approaches that can and have been applied in medical imaging. Object detection, image segmentation and classification are a few examples. To my knowledge, there are only a few studies on locating the different structures in the human heart by using object detection. Nevertheless, Yang et al. [6], have shown that using one-stage object detection one can locate the left ventricle (LV). The authors also stated that "Left ventricle detection from multiview echocardiography images can help clinicians diagnose heart disease more comprehensively and accurately". This shows that it is important and need of object detection algorithms to detect the structures in the human heart in cardiac ultrasound. In addition, Zeng et al. [7] demonstrate promising results in using region detection on various ultrasound images.

1.2 Problem statement

The aim of this master thesis is to explore the use of deep learning method to automatically locate cardiac structures in echocardiographic images, so non-experts can utilize cardiac ultrasound. This by using the one-stage object detector You Only Look Once (YOLO) to draw bounding boxes around the left ventricle (LV), mitral valve (MV) and left atrium (LA). The object detection algorithm is trained and tested on data annotated by trained experts in the circulation and medical imaging field. The main objective is to find a robust enough model that can detect the structures in both ideal and non-ideal scenarios. Therefore the following implementation and analysis will be addressed:

- Use cardiac ultrasound knowledge to implement new features such that the model is reliable enough to be used in various ultrasound situations.
- Evaluate different object detection models due to speed-accuracy tradeoff.
- Explore the use of data augmentation to increase the variation in the data and thus performance of the model.
- Investigate if the model is adequate by testing on different sizes of dataset.
- Check if the model can be used in real world scenarios by applying it on new unseen data in both standard and nonstandard cardiac views.

1.3 Outline

The first chapter includes the motivation and problem statement for this master thesis. The theory required for the thesis is covered in Chapter 2. It starts with the cardiac structure and

function and ultrasound imaging and deep learning, in addition to object detection. The methodology is presented in Chapter 3, which includes the dataset and data preparation, as well as preprocessing, implementations, and the architecture of the object detection model. Chapter 4 provides all of the metric evaluation and detection results from various object detection model analyses. In Chapter 5, the results are discussed, and future work is suggested. Lastly, Chapter 6 brings this thesis to a close.

Chapter 2

Theory

In this chapter we will go through theory about the human heart, ultrasound imaging, machine learning, some deep learning metrics and the object detection model YOLO. The theory about the human heart, echocardiography, deep learning and object detection are adapted and extended from the theory section of the specialization project [8].

2.1 The human heart

The human heart is a muscular organ located between the lungs in the thoracic cavity of the body, with its main purpose is maintain a constant blood flow throughout the body [9]. Figure 2.1 illustrates the structure of the human heart. The heart is divided into four chambers: right atrium, right ventricle, left atrium and left ventricle. The right atrium receives blood from the veins, and will pump deoxygenated blood through the tricuspid valve to the right ventricle. The blood is further pumped through the pulmonary valve to the pulmonary trunk and then into the lungs, where it is filled with oxygen. The oxygenated blood is then gathered by the left atrium and pumped through the mitral valve to the left ventricle. Following this, the left ventricle forces the oxygenated blood through the aortic valve into the aorta, where it is distributed to the rest of the body [10, Chapter 19.1].

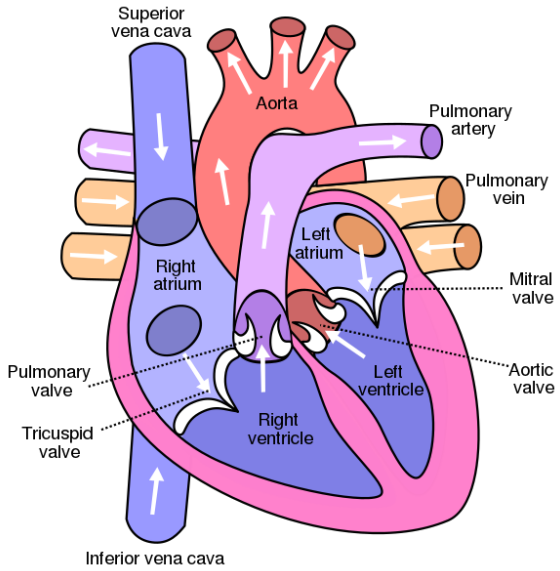


Figure 2.1: Illustration of the cardiac structure with the blood flow direction marked with arrows. Graphic by Wikimedia user Wapcaplet, reproduced under the CC BY-SA 3.0 license [11].

2.1.1 The cardiac cycle and function

The cardiac cycle consist of two phases: diastole and systole phase. The systole phase is the period when the heart contracts and is followed by the diastole phase, which occurs when the atria and ventricles relax. The muscles in the atria and ventricle contract rhythmically at a pace that depends on the activity in the rest of the body. The phases can be further divided into four different events: The atrial systole, ventricular systole, atrial diastole, ventricular diastole [10, Chapter 19.3].

The cardiac cycle begins with a relaxation period where both the atria and ventricles are in rest. The cycle then continues into the atrial systole, followed by the ventricular systole, atrial diastole and the ventricular diastole. During *atrial systole*, the atria contracts and the atrial pressure rises, causing a small amount of blood to fill the ventricles through the tricuspid and mitral valve, also known as the atrioventricular valves. After the atrial systole, right before the beginning of ventricular systole, is the period known as the end of diastole (ED), and the volume of blood remaining in the ventricle is defined as the end of diastole volume (EDV).

The *ventricular systole* can be divided into two different phases. First phase is the *isovolumetric contraction*, that takes place while the atrioventricular, aortic and pulmonary valves are closed. Due to the valves are closed, the blood volume will not change. Nevertheless, the ventricles contracts, causing the ventricular pressure to rise. The rise in the ventricular pressure causes the atrioventricular valves to bulge into the atria, resulting in a slightly rise in both the left and right atrial pressures. The cycle then continues to the second phase in the ventricular systole: the *ventricular ejection*. As the blood pressure in the ventricles rises above the pressure in the aorta and pulmonary arteries, the aortic and pulmonary valves open. The blood is then ejected to the lungs through the pulmonary valve and to the rest of the body via the aorta. The aortic valve then closes, marking the end of systole (ES), and the remaining volume of blood in the ventricle is called the end systolic volume (ESV).

The cycle then progresses to the *atrial diastole* where the atria relaxes and is then filled with blood through the pulmonary veins, the superior and the interior vena cava. When the atrium pressure exceeds the ventricle pressure, the tricuspid and mitral valve open.

Ventricular diastole, such as ventricular systole, can be separated into two phases. The first is the *isovolumetric relaxation*, where both the tricuspid and mitral valve is closed. Due to a pressure fall in the ventricles, blood flows back towards the heart, making the pulmonary and aortic valve close in order to prevent backflow into the heart. In the next phase, known as *late ventricle diastole*, the pressure in the ventricle falls below the pressure in the atria. At this point the atria start to fill the relaxed ventricles with blood, forcing the atrioventricular valves to open. The phase ends with the semilunar valves closed, atrioventricular valves open and both the ventricles and atria in diastole, marking the end of the cardiac cycle.

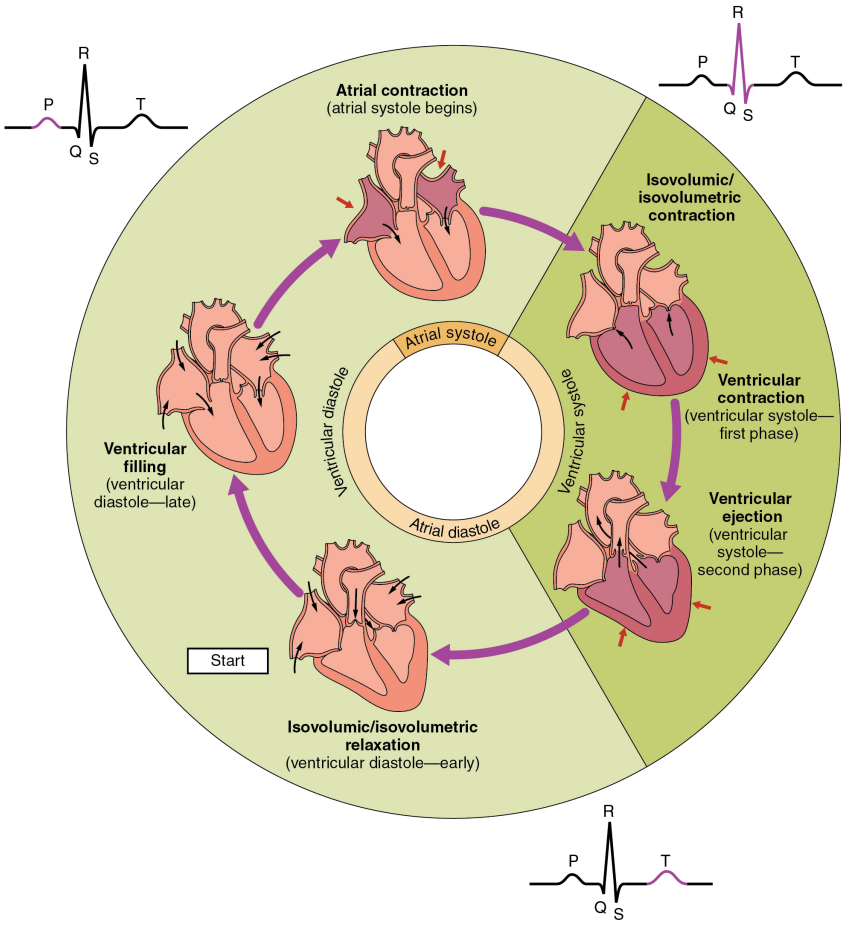


Figure 2.2: Illustration of the cardiac cycle, where the arrows indicate the cycle's direction and start marking where the cycle begins. Graphic by Wikimedia user OpenStax College, reproduced under the CC BY-SA 3.0 license [12].

2.2 Ultrasound imaging

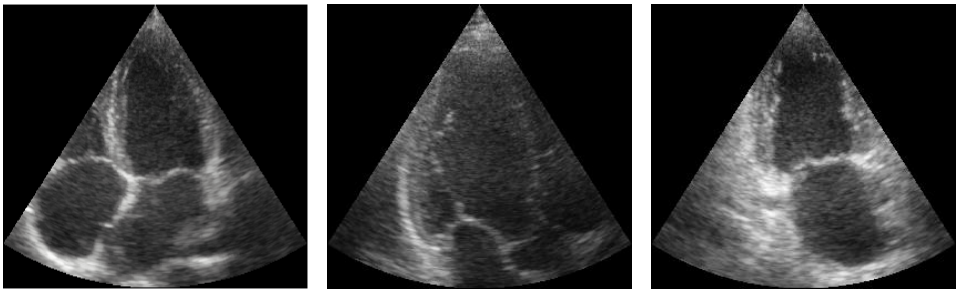
Ultrasound Imaging, also known as diagnostic ultrasound, is a non-invasive medical procedure that uses images to view the inside of the human body. The images are produced in real-time by high-frequency sound waves [10, Chapter 1.7]. For examining the human heart, one uses a type of ultrasound which is known as echocardiography or cardiac ultrasound.

2.2.1 B-mode

The ultrasound image can be created in a variety of methods. The most common is the 2D imaging mode B-mode, that stands for brightness modality. The B-mode image is created by an ultrasound probe that transmit multiple pulses into the tissue one by one from various angles. Echoes are then generated as these pulses are scattered and reflected. Some of the echoes are recorded by the transducer as they return back. The intensity of these echoes will vary with depth and the type of tissue being imaged. The depth, which is the distance between the transducer and the target, can be determined using the following equation: $d = \frac{ct}{2}$. In which c is the speed of sound, that is set to approximately $1540ms^{-1}$ in the human tissue, and t is the time it takes for the echo to return to the transducer. These echoes are joined together to form a single scan-line, called B-mode line. The 2D B-mode image is then created by multiple B-mode lines which is generated as the probe is swept over the patient's chest [13].

2.2.2 Echocardiography

Echocardiography is an ultrasound technique used to study the human heart. By using ultrasound one can see how the muscles and valves of the heart function. Transthoracic Echogardigraphy (TTE) is one of the forms of echocardiography used to evaluate cardiac function. TTE is performed by placing a probe, often described as a transducer, between the ribs on the patient's chest. The transducer generates sound waves that reflects off the heart. It then records the reflected sound wave echoes, and then generate images, as described in the previous section. TTE uses various imaging windows, or views, of the heart to evaluate specific cardiac structures [14]. The most common cardiac views are apical two-chamber (A2C), apical four-chamber (A4C) and apical long-axis (ALAX), as well as the parasternal long-axis (PLAX) and parasternal short-axis (PSAX) [15].



(a) A4C view. **(b)** ALAX view. **(c)** A2C view.

Figure 2.3: Example of still 2D B-mode images in the three of the standard views. In (a) the LV, MV and LA is displayed, alongside with the aortic valve, right ventricle and atrium on the left. In (b) the LV, MV and LA is displayed, alongside with the aortic valve on the right. In (c) the LV, MV and LA is displayed.

2.3 Deep learning

Artificial intelligence (AI) is a type of information technology that changes its own behaviour and thus appears intelligent. The goal of AI is to create computer systems that can adapt from their own experiences and solve complex problems in a variety of scenarios and environments [16, Chap 1.1]. A subcategory of AI is machine learning (ML). ML is an artificial intelligence specialization in which statistical methods are used to enable computers to identify patterns in large quantities of data. Instead of being programmed, the computer "learns" by training on a certain amount of data. Furthermore, ML is divided into three different categories: supervised, unsupervised and enhanced learning. The difference between these methods is that supervised learns to understand that the input data predict the output values, unsupervised tries to find the structure of the input values without the knowledge/access to output values. Lastly, in enhanced learning the model interacts explicitly with an environment that provides punishment or reward. It can be used in situations where there are several paths to the goal, and no indication of which is the strongest [17].

An important approach in ML is deep learning (DL). DL trains multilayered artificial neural networks (ANN) to solve various tasks such as object detection in images. ANN is based on the biological neural network in the human brain, where algorithms are inspired by the organization of nerve cells in the brain. DL has the potential to learn directly from given data and would need less interference from humans than traditional ML algorithms. This implies that a DL algorithm can automatically extract features and learn by its errors [18, Chap 1.2].

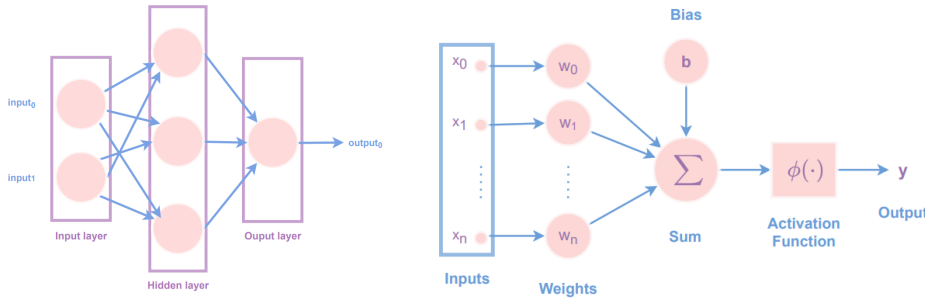
2.3.1 Neural network

A neural network (NN) receives data, trains itself to identify patterns in the data, and then predicts the output [19]. The core of a NN is the neurons, which are data elements that can receive and transmit numerical values to each other. The neuron can be defined as a mathematical function as shown in Figure 2.4b. Here the neuron receives input data, x which is multiplied with a assigned weight, w. The sum is then sent to the neurons, which are each assigned a potential bias word, b, and then applied to the input sum. The sum is further sent through an activation function, $\phi(\cdot)$, to achieve a non-linear behavior, and is transferred to the output-vector, y. The activation function describes the output behaviour by activating it. It exists different kind of activation function, but two of the most common ones are the Rectified Linear Unit (ReLU) function, shown in (2.1), and the Sigmoid function, shown in (2.2) [20]. Where in both cases, x is the input value and both produces an output shape same as the input shape.

$$ReLU(x) = max(0, x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.1}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

In Figure 2.4a a simple NN is displayed, with one input layer, one hidden layer and an output layer. As shown in the figure, the two input neurons are fully connected with the three hidden neurons which are then connected to one output neuron. The information received at the input is transferred through the hidden layer and then to the output. This type of NN is known as a feed-forward neural network (FNN), or multilayer perceptron (MLP), because there is no feedback from the output layer to the input layer. The input data will therefore only pass through the NN once without looping [19][18, Chap 6].



(a) Neural network.

(b) Neuron as a mathematical function.

Figure 2.4: Example of a neural network with one hidden layer and the math of a neuron. The arrows indicates the direction. The illustrations are reproduced from the Figures in Haykin [21, Chap I.3, Chap 4.2]

For training a FNN, there are different techniques one can use. One of the simplest and most used technique is the back-propagation algorithm. The concept of this algorithm is to proportionally modify each weight in the network based on how much it contributes to overall error. After several iterations, the error in the weights will minimize, resulting in a set of weights that improve the predictions. This is done by calculating the gradient of the error in the weight, by using the three equations: output layer error, hidden layer error and cost-weights derivative [18, Chap 6.5]. Figure 2.5 shows a visual explanation of the calculation. Here H and O represents the hidden layer and output layer activation, and X the input. $C'(W_h)$ and $C'(W_0)$ is the cost derivative for the weight on the hidden layer (W_h) and the weight on the output layer (W_o). Furthermore, E_h the hidden layer error and E_o the output layer error, where $R'(Z_h)$ and $R'(Z_o)$ is the derivative of the ReLU activation of the layers' input Z.

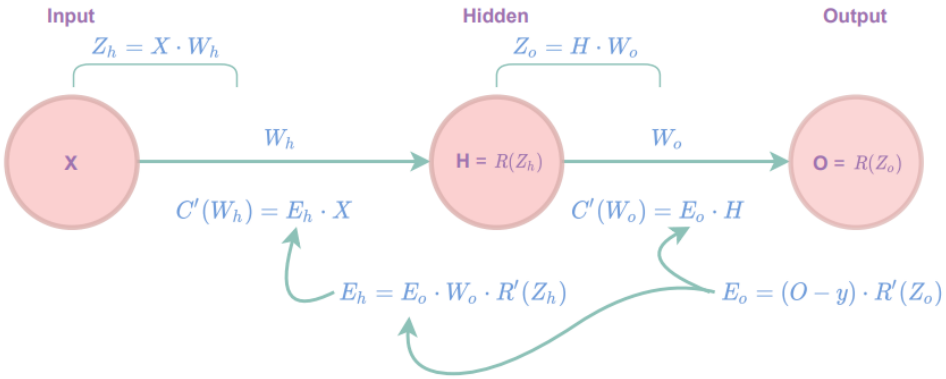


Figure 2.5: Visualization of the calculations of the back-propagation in a NN. The illustration is adapted from [22].

2.3.2 convolutional neural network

The convolutional neural network (CNN) is a deep learning algorithm used in different tasks as image recognition, object detection, segmentation etc [23]. In for example object detection, the CNN takes an input image, applies weights and biases and then create relevant image features, which are then extracted. These features are then used for recognizing patterns in the image, such as edges, textures and contours. The structure of the CNN algorithm is displayed in Figure 2.6. The CNN is built up by input layer, various hidden layers and an output layer, where the most important hidden layers are the convolutional layers [24].

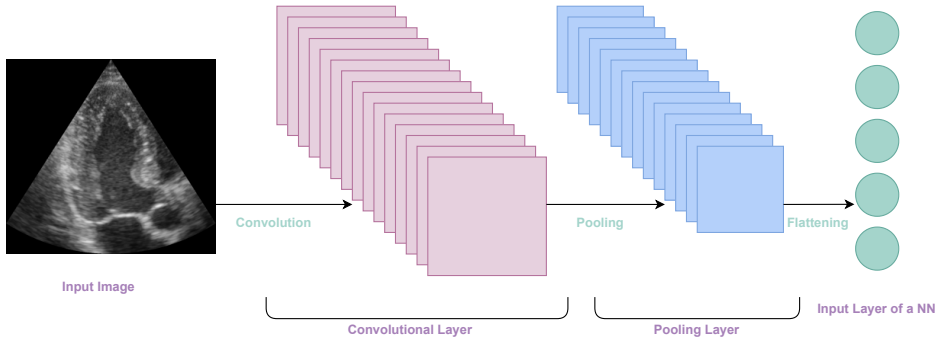


Figure 2.6: A fully connected CNN consisting of two hidden layers: one Convolutional and one Pooling Layer, where both have several feature maps. Illustration adapted from Wikimedia by user Aphex34 under CC BY-SA 4.0 license [25].

A convolution is a mathematical approach used in image processing to gather information about the arrangement of pixels in an image using filters, also known as *kernels*. Simply defined, a convolution receives an input image and applies a kernel on it before producing an output image. The kernel is a weighted matrix that is multiplied with the input as it moves across the pixels in the input image. The pixel values are then multiplied and added by using kernels. This results in a convolution, that will produce a feature map as an output. This method is visualized in Figure 2.7, and in Equation (2.3) one can see how the convolution is defined mathematically [18, Chap 9].

$$\mathbf{a}^{l+1} = \sigma(\mathbf{b}^l + \mathbf{w}^l * \mathbf{a}^l) \tag{2.3}$$

Here $\sigma(x)$ is the activation function, $\mathbf{b}^l$ is the bias and $\mathbf{w}^l$ is the weight at layer l. In addition, $\mathbf{a}^l$ is the set of input activations at layer l, and the $*$ is the convolution operation. $\mathbf{a}^{l+1}$ is the output activations from a feature map. A convolutional layer is made up of one or more kernels, producing multiple feature maps.

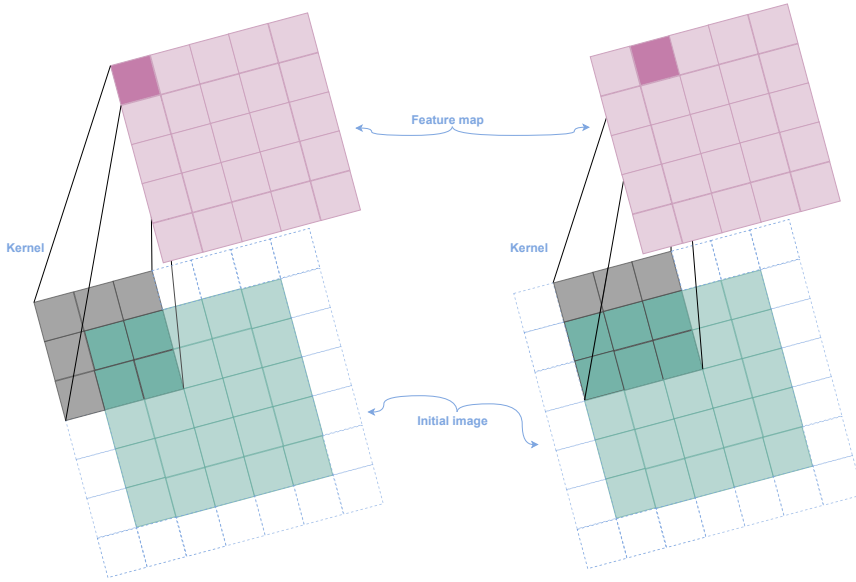


Figure 2.7: Convolution of a 3 x 3 kernel with the initial image, yielding a feature map. Illustration adapted from Wikimedia user Omegatron under MIT license [26].

A CNN, in addition to the convolutional layers, comprises another layer known as *pooling layer*. This layer is typically implemented after the convolutional layers, as seen in Figure 2.6. The pooling layer is implemented for reducing the dimension in the input image, which results in lower amount of parameters. This is done by downsampling every feature map, minimizing the height and width while maintaining depth [18, Chap 9.3]. There are different kinds of pooling functions one can use. Two of the most popular ones are *max*

pooling and *average pooling*. Max pooling returns the maximum value of the input at each stride, while average pooling returns the average value [27].

2.3.3 Performance metrics

IoU - Intersection over Union

A commonly used metric to evaluate the performance of a deep learning network is the IoU, which stands for Intersection over Union [28]. IoU will measure the object detector's accuracy on a given dataset, by using the formula as shown in Equation (2.4), where AoO stands for Area of Overlap and AoU is Area of Union. AoO is the intersection between the ground truth and predicted bounding box, while AoU is the union of the two bounding boxes. In Figure 2.8, one can see how the AoU and AoO is represented.

$$IoU = \frac{AoO}{AoU} \tag{2.4}$$

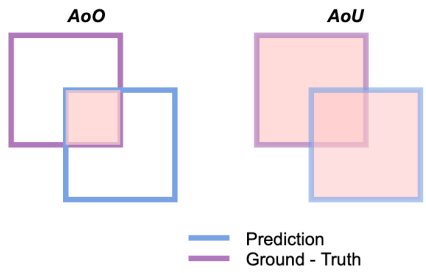


Figure 2.8: Intersection over Union. Here blue is the prediction and pink is the ground truth. The AoO demonstrated overlap and AoU the union between ground-truth and predictions.

The bounding boxes' IOU values will be between 0 and 1, where the closer the value is to 1, the more accurate the predictions are. For example if the IOU score is zero, the two bounding boxes do not converge, while if the score is one, the two boxes fully intersect.

Confusion Matrix and F_1-score

Another metric one can use to evaluate the a deep learning network is the confusion matrix. The metric uses the four elements: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). In for example object detection the metrics can be interpreted as [29]:

- **TP**: The model identifies an object, which is correct.

- **TN**: The model does not identify an object, which is correct.
- **FP**: The model identifies an object, which is incorrect.
- **FN**: The model does not identify an object, which is incorrect.

In Figure 2.9, one can see how the relationship between the true class versus the predicted class by using the four components, and the corresponding metrics.

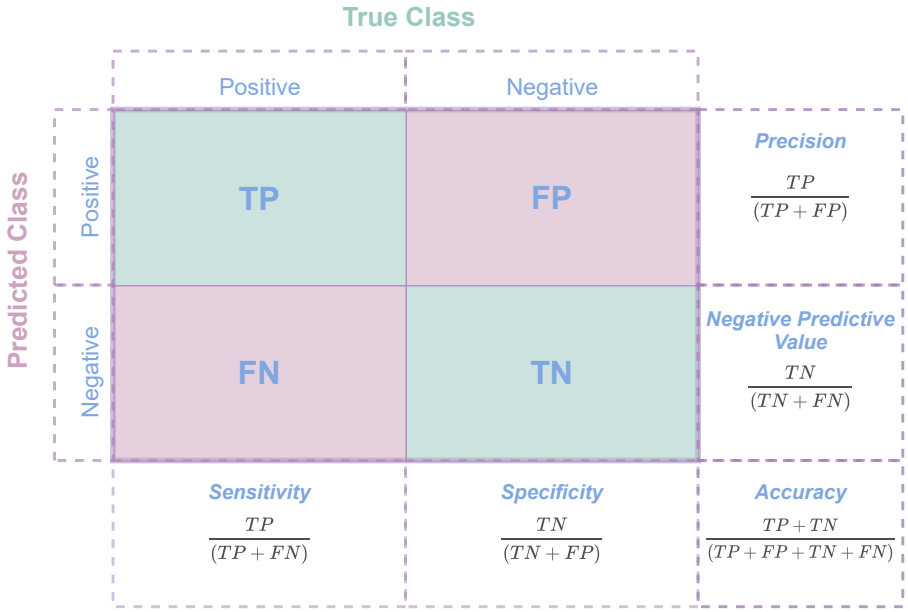


Figure 2.9: Confusion matrix with corresponding performance metrics. The matrix demonstrates the true versus the predicted class.

Precision and sensitivity, also known as recall, are two fundamental metrics for computing other essential metrics in object detection model evaluation. Precision is a classifier's ability to recognize just relevant objects, and is calculated by the ratio between the TP and all the detections. Recall refers to the classifier's ability to identify all the ground-truth, and is the ratio between the TP and the ground truth. By using the recall and the prediction, we can calculate the F_1-score, as shown in Equation (2.5) [30].

$$F_1 - \text{score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.5}$$

The F_1-score varies between 0 to 1, where 1 represents the greatest level of precision.

mAP - mean Average Precision

The mean Average Precision (mAP) is one of the most used metric for evaluating object detection models. It calculates the mean of the average precision (AP) of all the given classes [31]. Firstly, the AP needs to be determined before calculating the mAP. It can be explained mathematically by using the Equation (2.6). AP is the area under the Precision Recall (PR) curve, in other words, is the weighted sum of predictions at each point in which the weight is the increase in recall. The latest precision, $Precisions(k)$, is multiplied with the discrepancy between the present, $Recalls(k)$, and following recall, $Recalls(k+1)$. The number of thresholds is n, and the class is k.

$$AP = \sum_{k=0}^{k=n-1} [\text{Recalls}(k) - \text{Recalls}(k+1)] \cdot \text{Precisions}(k) \tag{2.6}$$

The mAP is then computed by the mean of the AP, shown in (2.7), where the number of classes is n and the average precision of the class, k, is AP_k.

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \tag{2.7}$$

2.3.4 Overfitting and underfitting

When training a neural network, it is important to analyse the model's performance by comparing the training loss to the validation loss. Figure 2.10 depicts three different fits that can be used to evaluate the performance of the DL algorithm. A model is underfitted when the validation loss is to close to the training loss, in addition to both losses will not achieve the optimal loss. As a result, the model will make inadequate predictions on both training and new data. An overfitted model, on the other hand, will experience an increase in validation loss during training, leading to a discrepancy between training and validation loss. When overfitting occurs, the model can learn from erroneous in the data, which results in inaccurate predictions. Model with a good fit is somewhere between underfit and overfit, in which the model outputs low errors and is right before the validation set starts to increase [18, Chap 5.2]. To avoid both underfitting and overfitting one can increase the data in the dataset. Furthermore, increasing the complexity of the model and epochs can help to avoid underfit, and decreasing the complexity and epochs can prevent overfit.

Figure 2.10: Underfitting, good fit and overfitting. A model is underfit if the training loss and validation loss are close to each other. It is a good fit if training loss is slightly lower than validation loss. A much lower training loss than validation loss indicates an overfit. Here epochs represents the training steps.

2.4 Object Detection

Object detection is a computer vision task that aims to find and recognize instances of objects of a specific category within an image. It combines both image classification and object localization, by finding the presence of an object in an image and placing a bounding box around the object. The bounding box is then assigned with a label representing the object's class [32]. One can divide object detection into two categories: one-stage and two-stage detectors. The different between these is that the two-stage first select a limited regions of interest and predict bounding boxes from these regions, while the one-stage detector predicts bounding boxes by only one run through its network [33].

2.4.1 YOLO

YOLO is a fast real-time one-stage object detection model, and stands for You Only Look Once. Redmon et al. [34] first published YOLO in 2015, with the YOLOv1 model. This model managed to predict several object bounding boxes with corresponding class probabilities by only using a single CNN, therefore its name. This type of object detector is called a one-stage object detector, as it predicts bounding boxes in an image by a single pass through its network. Since the first publication new versions of the object detection model have been released, all introducing new features. YOLOv2 [35] was released in 2016, and two years after in April 2018 YOLOv3 [36] was introduced with a new classifier network improving the predictions and mAP values. In April 2020, Alexey Bochkovskiy published the fourth version of the YOLO family: YOLOv4 [37]. YOLOv4 outperforms previous versions and other state-of-the-art object detectors in terms of speed-accuracy trade-off. The authors of the fourth version also introduced a new data augmentation and hyperparameters by employing generic algorithms which was a concept developed by Glenn Jocher. Not only did Jocher contribute new ideas for improving the fourth edition,

but the same year, in June 2020, his team at Ultralytics team launched the latest version of YOLO, version 5 (YOLOv5) [38]. This version has not yet been published in any official paper. However, Ultralytics demonstrates remarkable improvements in both speed and accuracy. In addition there are papers exploring the use of YOLOv5, for instant [39] stated that: ”Experimental results show that YOLOv5 algorithm is superior in almost all indicators. Especially, YOLOv5 algorithm is superior to Faster R-CNN algorithm in terms of speed, memory occupancy, and accuracy of object position prediction.”

2.4.2 Architecture

An overview of YOLOv5's architecture is displayed in Figure 2.11. As seen in the figure, one can divide the architecture into three parts: *Backbone*, *Neck* (PANet) and *Head* (YOLO layer) [39]. The Backbone is made up of a CNN that combines and produces image features at various granularities, while the Neck is made up of a sequence of layers that will include the image features. By using these features, the Head collects the class and box prediction steps. YOLOv5 uses a special kind of CNN in both the Backbone and Neck, called Cross Stage Partial Network (CSPnet). CSP enhance the learning capability of the CNN by extracting the important features from the input image. As the authors, Wang et al. [40], stated ”the CSPnet can greatly reduce the amount of computation, and improve inference speed as well as accuracy.” Additionally, the network reduces memory costs and allows for the usage of both CPU and mobile GPUs for training the network. The Backbone also consist of a Spatial Pyramid Pooling (SPP) layer, which obtain both coarse and fine information by pooling on various kernel sizes at the same time [41].

As one can see in the Figure 2.11, Path Aggregation Network (PANet) forms the neck. The PANet uses several different features for improving the information flow in the YOLOv5 framework. PANet is made up by a bottom-up path augmentation, adaptive feature pooling and a fully connected fusion. These are techniques for decreasing the information path between the lower layers and the top feature, connect feature grids at all feature levels and for improving the predictions for the object detection [42]. The head of the network is the output in Figure 2.11, and consists of the YOLO layer. The YOLO layer is the same as is used in the previous YOLO versions 3 and 4 [36][37]. This layer is the detection part of the network, which is done with anchor-based detection stages that detects the bounding box coordinates and the corresponding class predictions.

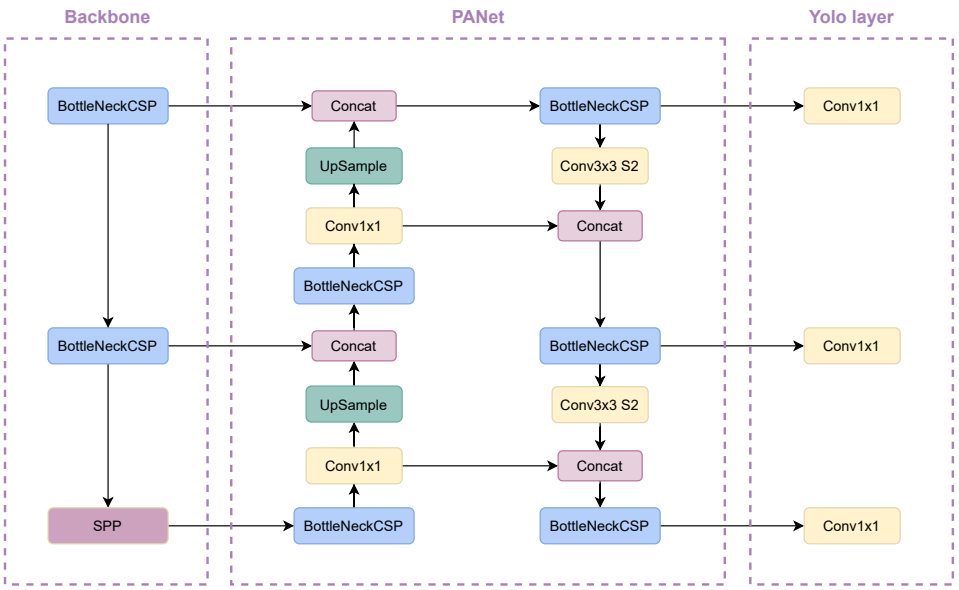


Figure 2.11: Overview of YOLOv5's Architecture, where the backbone is built up by the CSPnet and SPP layer, the neck is the PANnet and the head/output is the yolo layer containing 1 dimensional convolutional layers. (Illustration is reproduced from Figure 4 in [43])

2.5 Data Augmentation

A key technique for improving the robustness of the deep learning network, is by using data augmentation. Data augmentation ensures that the relevant data increases, while irrelevant data decreases. In addition to increase the variation of the data, in terms of views and different scenarios. As a result, the machine learning model will train on more data, resulting in more accurate predictions. This also prevents the network from learning irrelevant patterns, which improves the model's accuracy. Furthermore, by using more data augmentation one will reduce and delay overfitting. This results in longer training and then a higher mean average precision [44].

Below one can see a few of the most popular used data augmentations in object detection:

- **Image Flip**: Flipping the images either horizontally or vertically or both.
- **Image Rotation**: Rotating the image with different angles. Here one need to be careful, due to the dimension of the image can change when rotating the image.
- **Image Translation**: Translate by moving the image in either y or x direction or both directions. This force the network to look everywhere in the image, as the objects can be localized different places in the image.

- **Image Scale**: Scaling the image inwards or outwards. When scaling, the image size will either increase or decrease.
- **Image Crop**: Random cropping the image and resize the image size so it is similar to the original image dimensions.
- **Image Shear**: Shifts parts of the image, similar to a parallelogram.
- **Mosaic**: Combining four different images into one.

Because certain data augmentations can cause the image size to change, it is important to be cautious when adjusting the hyperparameters. Interpolation is often used by deep learning models to ensure that the most of the image is used or to avoid losing essential parts of the image.

In addition to the most popular data augmentations, one has the data augmentation *Random Gamma*. This augmentation random applies brightness to the image with different intensities. This is done by the using the Equation (2.8a), where γ is given in (2.8b).

$$i_{out} = c \cdot i_{in}^{\gamma} \tag{2.8a}$$

$$\gamma = \gamma_{min} + (\gamma_{max} - \gamma_{min}) \cdot x_{random}$$

Here i_{out} is the output intensity and i_{in} is the input intensity. c and γ are both positive constants, and gamma can vary between given values from $gamma_{min}$ and $gamma_{max}$. This equation is called the power-law transformation [45]. Random gamma can be an important augmentation when it comes to using ML on grayscale images, as it can improve the object detection model's performance by giving it more variations in the training data.

During training and testing using YOLOv5, the object detection model creates a dataloader for loading the data. In this dataloader one can specify different hyperparameters for using the data augmentation techniques. The dataloader generates augmented views of the dataset on demand, which are then used for training only once. This means that the augmented views will never be repeated.

Chapter 3

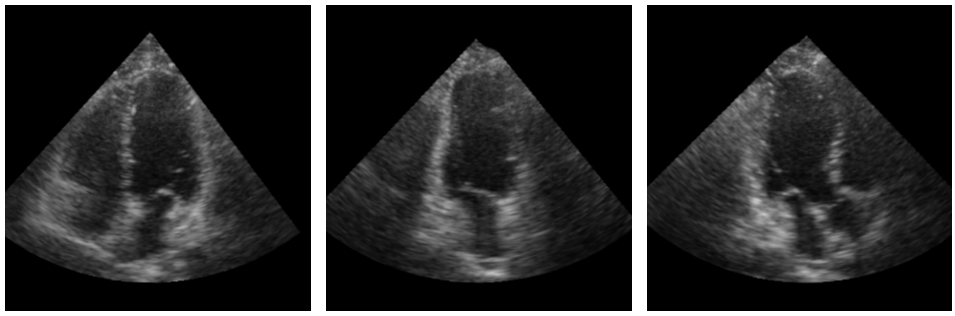
Method

In this chapter, the methodology and material used are represented. Some of the methodology is modified from the specialization project [8]. For detecting the different structures of the human heart from echocardiography images, relevant data and their corresponding annotations are required. The ground truth annotations are done by clinicians using an annotation tool from the multipurpose application EchoSearch developed by the the Department of Circulation and Medical Imaging at NTNU.

3.1 Data

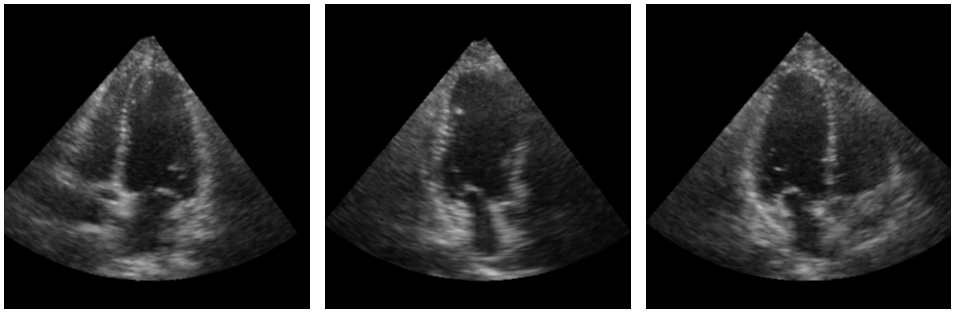
The data used in this project was retrieved from the data collection HUNT4 from The Trøndelag Health Study (The HUNT Study). This data was provided from 62 different patients, which included 195 recordings. The HUNT study is one of the world's largest health studies consisting of a unique database of data, clinical measurements and samples from the county of Trøndelag (NTNU) [46]. Patients are examined here for research purposes. The data obtained is in the form of 2D-echocardiography images. These images are all in a DICOM file format, which includes recordings of varying lengths as well as additional meta data. The recording's frame rate is normally 50 frames per second, however it might vary depending on the image's height and width. The images varies in the A4C, A2C and ALAX cardiac views.

In addition, two additional datasets were created to test how the model performed on both standard and nonstandard views. The dataset was obtained from the Forshortening2021 study Pettersen et al. [47], which contains 3D echocardiography images. To obtain the right data for object detection the 3D data is rotated in different ways and then sliced creating 2D images in different views. Further explained, this is achieved by first specifying the LV long axis and then generating 360 slices around the axis, one per degree. These slices are then sent to a classification network, which decides which of these slices corresponds to the desired views. In this particular case, the view sliced were A4C, A42C, A2C, ANS, ALAX and ANE. The 2D images were then categorized into two dataset, standard viwes: A4C, A2C and ALAX and nonstandard views: A42C, ANS and ANE. Both datasets contains the total of 158 images each.



(a) A4C **(b)** A2C **(c)** ALAX

Figure 3.1: Examples of the standard views in 2D images captured from the 3D data.



(a) A42C **(b)** ANS **(c)** ANE

Figure 3.2: Examples of the nonstandard views in 2D images captured from the 3D data.

3.1.1 Annotation process

As mentioned in the beginning of this chapter, the ground truth annotations were done by trained experts in the cardiac and medical imaging field. The annotations were created for segmentation purpose, therefore bounding boxes was drawn around the LA and LV masks. Furthermore, a bounding box was generated for the MV in the transition between LA and LV, with two-thirds of the box in LV and one-third in LA. In Figure 3.3 an example of the ground truth annotations on the A4C, ALAX and A2C views are shown.

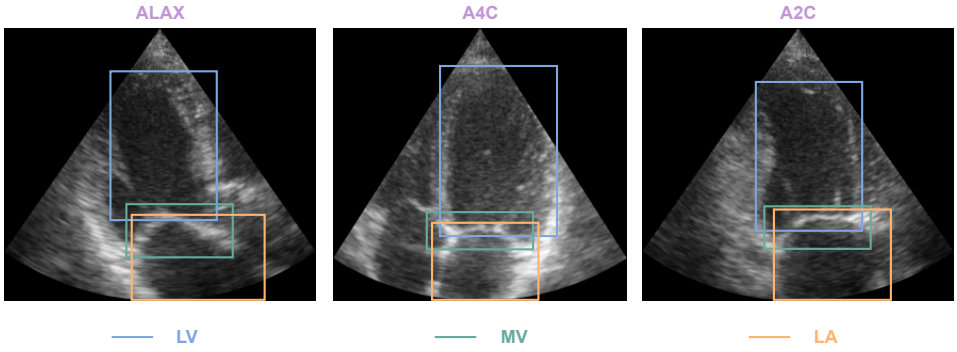
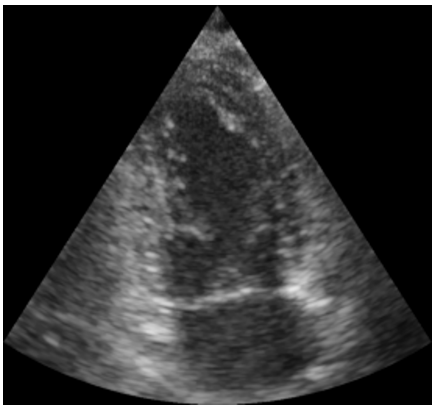


Figure 3.3: Example of ground truth on the echocardiographic images made for the object detection.

3.1.2 Data preparation

The data preparation for the object detection included converting the image format as well as altering the annotation data. The required format of the images are in .jpg/.png and etc. and the corresponding annotations in .txt-files. The image's resolution ranges from 275 x 256 to 362 x 256 pixels. When it comes to the annotations in the txt-file, the structure is

one bounding box annotation per row, with its class-label, bounding box x- and y-center and their height and width. The bounding box coordinates are normalized by dividing the bounding box center of x with the width of the image and the center of y with the height of the image. Furthermore, the class labels are transformed to zero-indexed class numbers. Figure 3.4 shows an example of how the images with corresponding annotation in txt-file looks like. The dataset was then divided into training, validation and test sets, where the images and the corresponding annotations are in different folders. The data was divided such that 70% of the data for training, 20% for validation and 10% was for testing. This resulted in a data distribution presented in Table 3.1.



```
0 0.52 0.46875 0.33 0.55
1 0.5583333333333333 0.84875 0.3333333333333333 0.2525
2 0.5341666666666667 0.711875 0.4216666666666667 0.12625
```

(a) Image in .png **(b)** Annotations in txt-file

Figure 3.4: 2D B-mode image from HUNT data and corresponding annotations in txt-file. The annotations in (b) is the following format: <class> <x_center> <y_center> <width> <height>.

Table 3.1: The HUNT data split into training, validation and test set.

Training	Validation	Testing	Total
902	251	107	1260

When evaluating the object detection model, it is critical to consider if there is adequate data to train the model on. To test this, an analysis of the entire dataset and two-thirds of the dataset was performed. Therefore, a new dataset was created by reducing the amount of data in the training and validation sets to two-thirds of the original size. This yielded a total of 602 images and corresponding annotations for the training set and 168 for the validation set.

3.2 Network - YOLO

YOLOv5 (version 4) was cloned from Ultralytics' github[1]. The Python-based object detection model YOLOv5 uses as the open source machine learning library PyTorch. The network is tested and trained on an NVIDIA Quadro P5000 GPU.

3.2.1 Architecture

As mentioned in 2.4.1 the architecture of YOLOv5 consist of three parts: Backbone, neck and head.

The head produces predictions for the bounding boxes, object and classes, called *box loss*, *objectness loss* and *classification loss*. Box loss gives a value for the discrepancy between the ground truth and predicted bounding box. Objectness provide a value indicating how probable an object is to exist in a cell, while classification determine whether or not an object is present in the image and what class the object represent. The network employs logistic regression for computing the predictions for the objectness loss per bounding box and binary cross-entropy with logistic loss for the classification predictions [48]. The box loss is calculated by using the IoU metric. The output from the head is in a form of a vector as shown in (3.1), where t_x, t_y, t_w and t_h are the predicted bounding box coordinates and pr_o is the objectness loss and pr_c is the class loss.

$$[\, t_x, t_y, t_w, t_h, pr_0, pr_c] \tag{3.1}$$

The predicted bounding box coordinates from the output vector is then used for finding the true bounding box coordinates. This is done by using the equations and method shown in Figure 3.5.

The network uses two activation functions: Leaky ReLU and Sigmoid. Leaky ReLU is a type of rectified linear activation function where it accepts small negative numbers if the input is below zero [49]. The function is shown in Equation (3.2), where a_{neg} is a parameter for controlling the angle of the negative slope. The Leaky ReLU activation function is employed in the hidden layers, while the sigmoid activation function is implemented in the YOLO layer.

$$\text{LeakyReLU}(x) = max(0, x) + a_{\text{neg}} \cdot min(0, x) = \begin{cases} x & \text{if } x \geq 0 \\ a_{\text{neg}} \cdot x & \text{if } x < 0 \end{cases} \tag{3.2}$$

[1] https://github.com/ultralytics/yolov5/releases/tag/v4.0

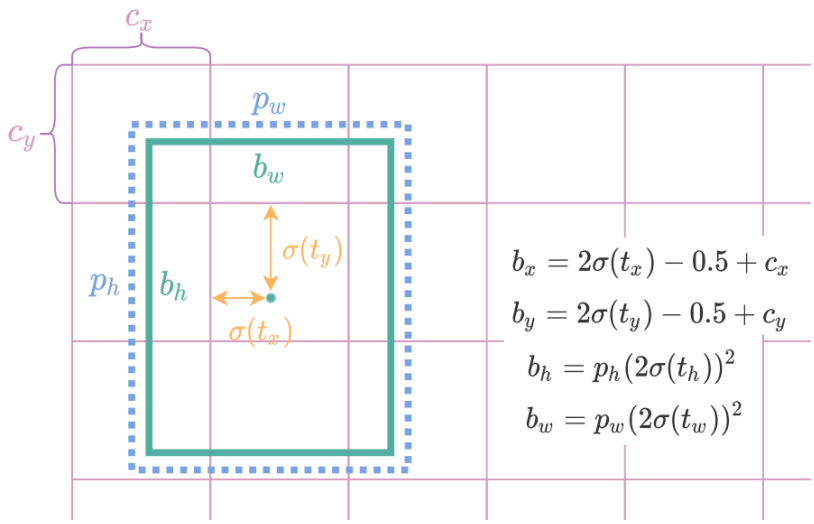


Figure 3.5: Bounding box prediction. Here the pink grid is the image, p_w and p_h is the width and height of the bounding box. The blue dotted box represents the anchor box and the green box is the output box. The figure is reconstructed from Figure 3 in [35, p. 4].

3.2.2 Model

The network structure for YOLOv5 consists of four different models, small, medium, large and xlarge. The authors of YOLOv5 have tested all the models on the dataset Common Objects in Context (COCO). The dataset is released by Microsoft for "advancing the state-of-the-art in object recognition. [...] This is achieved by gathering images of complex everyday scenes containing common objects in their natural context." (Lin et. al, 2015 [50]). The results from training and testing on COCO are given in Table 3.2, and are retrieved from the authors', Ultralytics, github [51]. Here a V100 GPU is used with a batch size of 32 and all the models are tested with the resolution of 640.

Table 3.2: YOLOv5's (version 4.0) models compared. Here with a batch size of 32, image size of 640. The mAP values are in percent and speed in ms/img.

Model	mAP^{val}	mAP^{test}	mAP_{50}	$Speed_{V100}$	parameters	GFLOPS
YOLOv5s	36.8	36.8	55.6	2.2ms	7.3M	17.0
YOLOv5m	44.5	44.5	63.1	2.9ms	21.4M	51.3
YOLOv5l	48.1	48.1	66.4	3.8ms	47.0M	115.4
YOLOv5x	50.1	50.1	68.7	6.0ms	87.7M	218.8

Several conditions must be considered while selecting the model to apply for the object

detection on our custom dataset. First of all the images in the HUNT dataset have a small resolution and are therefore small in size. Additionally, the images are in grayscale format, meaning one can use less parameters when training on the data. Furthermore, the inference and processing should be fast in comparison to the task's complexity. Since the COCO dataset contains the total of 80 classes and images in RGB, the HUNT dataset was tested on all the different models to see how the network behaves on grayscale images with only three classes.

3.2.3 Evaluation metrics and loss function

YOLOv5 offers a variety of metrics and losses to evaluate the object detection model's performance during training, testing and inference. The loss functions, which are box, objectness and classification loss, are used for demonstrating how the model performs and whether overfitting occurs. Furthermore, YOLOv5 use recall, precision, IOU and mAP, which were represented in the theory chapter 2.3.3, for analyzing the training and testing of the model. The mAP is represented as mAP@.5, implying that the mean average at IoU equals 0.5, and mAP@.95, implying that the mean average over various values of IoU in the range of 0.5 to 0.95 [31]. The values gives us an indication of how stable, precise and reconcilable the model is. In addition, the confusion matrix is also used to evaluate the classifier's performance, including the F1-score for the model's performance.

3.2.4 Training, testing and inference

Before training the desired YOLOv5 model some configurations and adjustments were done. First the model's configuration file, `yolov5*.yml`, was modified by replacing the number of classes to three. Thereafter, a configuration file, `yolo_HUNT.yml`, was made containing the paths to the train and validation set, the number of classes and the class names: `[LV, LA, MV]`. Furthermore the training parameters needs to be specified. These are the width of the image, batch size, numbers of epochs, the YOLO configurations, the data, the desired model, desired weights and which device one is using. The following command is an example on how to start the training on the server containing NVIDIA Quadro P5000 GPU:

```
python train.py --data yolo_HUNT.yml --cfg yolov5*.yml --weights ''
--img 256 --batch-size 64 --epochs 150 --device 0
```

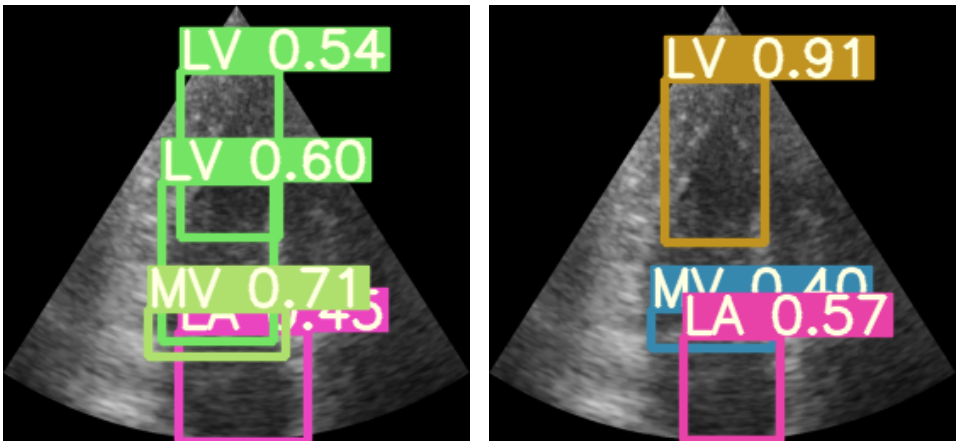
The weights used in training session was random initialized weights. Furthermore, the number of epochs were 150 and image width of 256.

The same commands implies for the test (`test.py`) of the object detection model. However here the best weights from the training is used. Here one can choose what kind of task (`--task `) one wants to do, here either test how the training went, or check the speed of the model or study the different model sizes. Furthermore, when doing the inference

(`detect.py`) the best weights from the desired training were employed and specified along with the path to the test set and a confidence threshold (`--conf-thres`).

3.3 Preprocessing

A problem that occurs several times is that the network predicts several boxes in a class. To prevent this, YOLOv5 uses a technique called non maximum suppression (NMS). NMS choose one bounding box from a set of multiple overlapping bounding boxes. The chosen bounding box is the one with the best prediction. But the algorithm implemented by the authors of YOLOv5 will still give more than one bounding box per class during testing and inference. This is probably due to the model believes there is more than one object per class in the images. In this particular case, we only want one bounding box per class, since there is for example only one left ventricle in the human heart. Therefore, some implementation were done for making the NMS to only predict one box per class in an image. The adjustment done was by making the NMS choose the box with the highest objectness score, then comparing the box's IoU with the other overlapping boxes from the same class. Then choosing the box with highest score and repeat this until all predicted boxes from a particular class had been considered. Then moving on to the next class. The result of the improvement is shown in Figure 3.6, where (a) is before and (b) is after.



(a) Before: with multiple bounding boxes per class.

(b) After: only one bounding box per class.

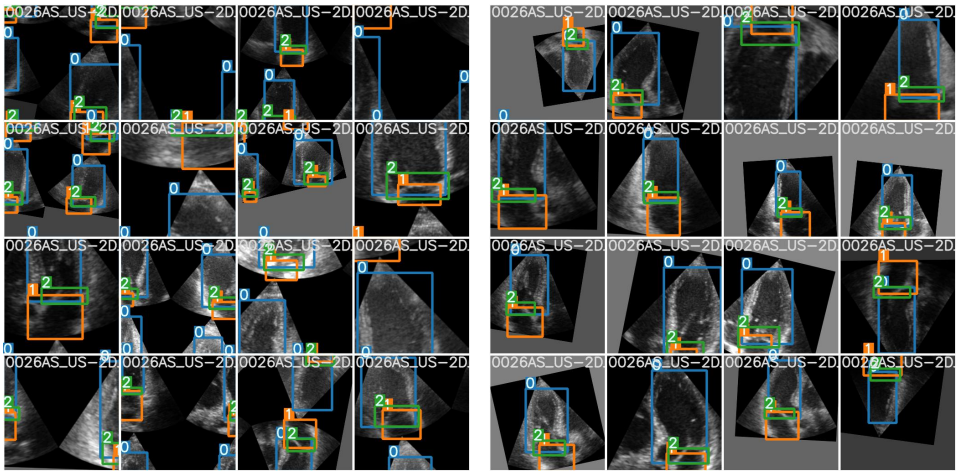
Figure 3.6: Before and after NMS improvement resulting in one bounding box for each class.

3.4 Data Augmentation

Different data augmentation techniques were assessed, implemented and used while training the model on the dataset. As YOLOv5 have different data augmentations, these were tested for increasing the variation in the data, which can result in more accurate predictions in different scenarios.

3.4.1 Mosaic

As mentioned in 2.4.1, the authors of YOLOv4 introduced to a new data augmentation hyperparameter called *Mosaic* implemented by Glenn Jocher in 2020, [37]. This augmentation technique was made for improving the mAP on the dataset COCO. The mosaic augmentation is shown to increase the performance of the object detection model, therefore mosaic was tested on the HUNT data. Figure 3.7 shows how the training batch appears with and without mosaic augmentation. The Table 3.3 shows an increase in the mAP value with the IoU between 0.5 and 0.95 for both the training and testing when not using mosaic augmentation. The same applies for the mAP value for the testing when the IoU is 0.5. As a result, the mosaic augmentation is not employed for further training of the object detection model.



(a) With mosaic **(b)** Without mosaic

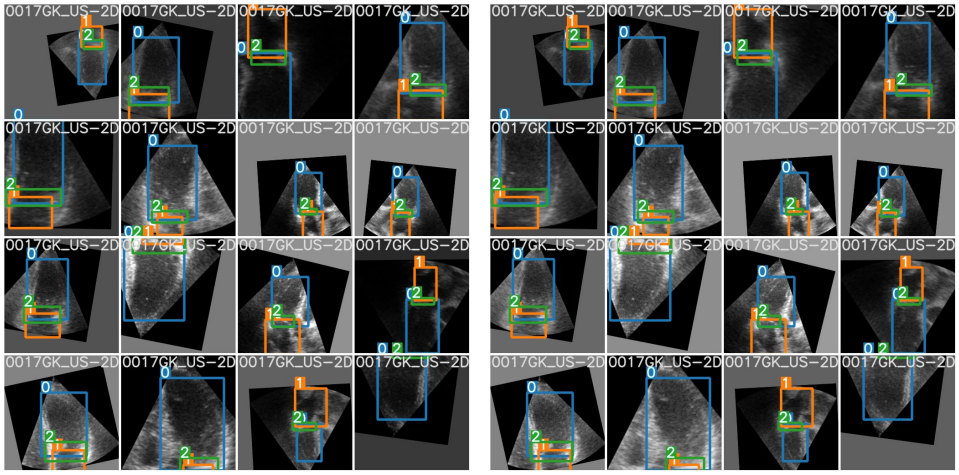
Figure 3.7: The figures shows training batches with and without mosaic. Figure (a) one can see that there is two, three or four images in one, while in (b) there are only one image at the time.

Table 3.3: The mAP compared for model with (1.0) and without (0.0) mosaic augmentation. Here the best values are in bold.

Mosaic	Train		Test	
	mAP@.5	mAP@.95	mAP@.5	mAP@.95
0.0	0.9941	**0.633**	**0.9941**	**0.633**
1.0	**0.9956**	0.6061	0.9816	0.6247

3.4.2 Random Gamma

The random gamma augmentation was implemented into the YOLOv5 network. This augmentation adds various brightness to the image, making the network train on different levels of luminance. Random gamma is an important data augmentation when it comes to grayscale 2D B-mode images. This because applying different brightness and intensities to the images make the network learn how to predict the structures in the human heart in various scenarios. And as shown in Table 3.4, the implementation of random gamma augmentation results in an increase in the mAP values in the training process.



(a) Without γ **(b)** $\gamma_{min} = 0.75$ and $\gamma_{max} = 1.6$

Figure 3.8: Training batch without (**a**) and with (**b**) Random Gamma.

Table 3.4: The mAP compared for model with and without random gamma augmentation. Here the best values are in bold.

Random Gamma	Train		Test	
	mAP@.5	mAP@.95	mAP@.5	mAP@.95
With	**0.984**	**0.631**	**0.984**	**0.631**
Without	0.983	0.622	0.983	0.622

3.4.3 Hyperparameters

The augmentations which were modified is displayed in the Table 3.5, where the value shows the final adjustments made to the hyperparameters in the `hyp.scratch.yml` file.

Table 3.5: Data augmentation values motified and used for training the object detection model.

Augmentation	Value
Shear	0.5 °
Scale	0.7
Mosaic	0.0
Translate	0.25
Rotation	15 °
Random Gamma	γ_{min} = 0.75 γ_{max} = 1.6

Chapter 4

Results

In this chapter the results from training, testing and inference are presented. First the results from training with different YOLOv5 models on our dataset are given. Following that, results from different analysis are presented, demonstrating how the model performs in various scenarios.

4.1 YOLOv5 - model comparison

Table 4.1 shows the how the different YOLOv5 models behaves on the HUNT dataset. Here the resolution of 256 and batch size of 32 is used including the test is done on the NVIDIA Quandro P5000 GPU. The best mAP values and speed are marked in bold in the table. The speed in ms/img versus accuracy in mAP@.5 for all the models is displayed in the graph in Figure 4.1. The graph indicates that as the resolution increases, the accuracy decrease.

Table 4.1: YOLOv5's models compared with training on the HUNT dataset. Here with a resolution of 256 and a batch size of 32. The speed is given in ms per image, and the best results are marked in bold. The table indicates that the x-large model is more accurate, but the small model is faster.

Model	mAP^{val}_{50}	mAP^{test}_{50}	mAP^{test}_{50-95}	$Speed_{P5000}$	parameters	GFLOPS
YOLOv5s	98.4	98.4	63.1	**2.61 ms**	7.1M	16.3
YOLOv5m	99.31	99.31	64.47	4.78 ms	21.0M	50.3
YOLOv5l	99.85	99.85	69.35	7.47 ms	46.6M	114.1
YOLOv5x	**99.89**	**99.89**	**71.65**	11.78 ms	87.2M	217.1

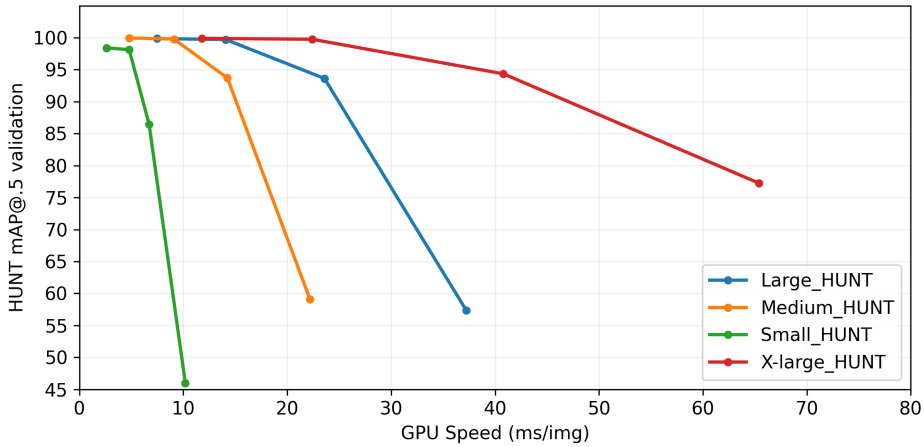


Figure 4.1: The YOLOv5 models compared on accuracy versus speed on GPU, when training on the HUNT dataset. The accuracy is in mAP@.5, meaning the accuracy when IoU is equal to 0.5. The figure shows that the accuracy decrease when increasing the resolution.

4.2 Data Augmentation

In this section results from the analysis of with and without data augmentation improvement is represented. The results without data augmentation, predefined weights are used and hyperparameters customized for the COCO dataset. In the results with data augmentation, the hyperparameters have been modified and new data augmentations implemented.

4.2.1 Metric evaluation

Table 4.2 shows the overall mAP values for the training and testing with and without data augmentation. Whereas the results in Table 4.3 shows a comparison of the mAP values for per class for both with and without data augmentation. In both tables the best predictions are marked in bold.

Table 4.2: The overall mAP values for training with versus without data augmentation. The best results are marked in bold.

Training set size	mAP@.5 train	mAP@.95 train	mAP@.5 test	mAP@.95 test
With	0.984	**0.631**	0.984	**0.631**
Without	**0.994**	0.62	**0.994**	0.62

Table 4.3: The overall mAP values per class for training with versus without data augmentation. The best results are marked in bold.

Class	With				Without			
	Train		Test		Train		Test	
	mAP@.5	mAP@.95	mAP@.5	mAP@.95	mAP@.5	mAP@.95	mAP@.5	mAP@.95
LV	**1**	0.721	**1**	0.721	**1**	**0.728**	**1**	**0.728**
LA	**1**	**0.67**	**1**	**0.67**	0.998	0.63	0.998	0.63
MV	0.951	0.502	0.951	0.502	**0.984**	**0.508**	**0.984**	**0.508**

Figure 4.2 shows the metric from the training process with data augmentation, while Figure 4.3 shows the metrics evaluation for training without data augmentation. Furthermore, the confusion matrix is compared for both cases in Figure 4.4, alongside with the F1-curve in Figure 4.5.

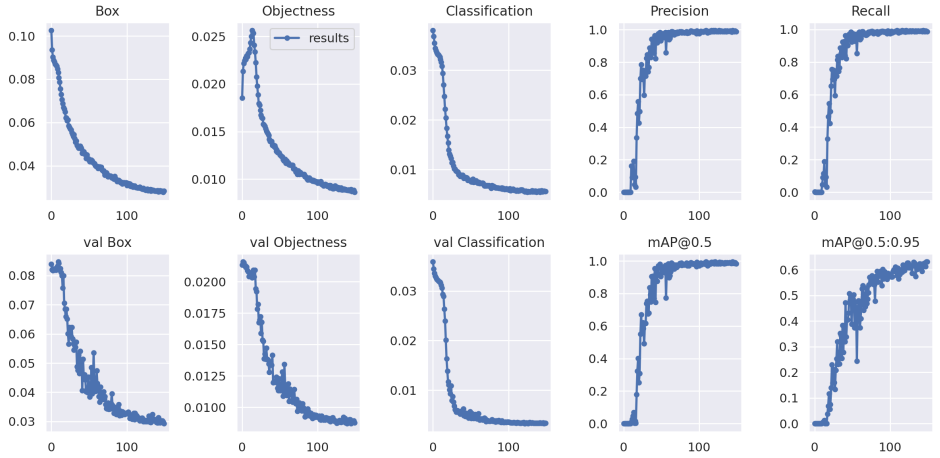


Figure 4.2: Metric evaluation from training with data augmentation with 150 epochs. The training set compared to the validation set shows a good fit.

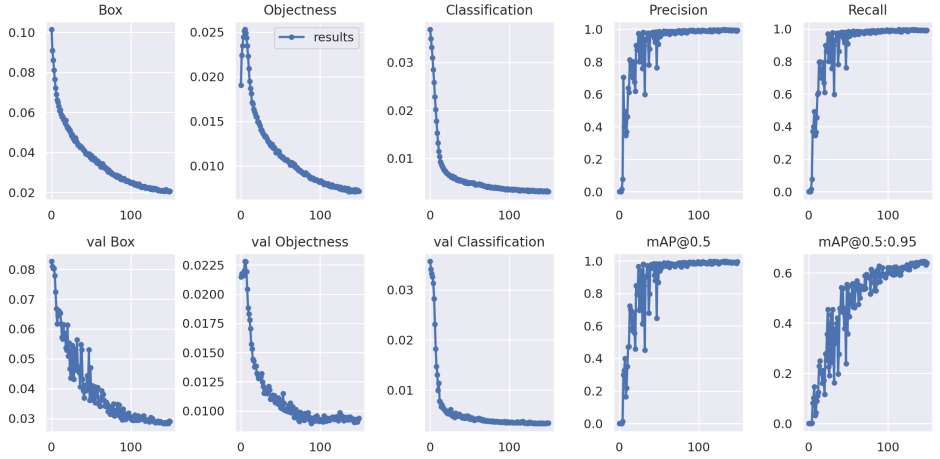
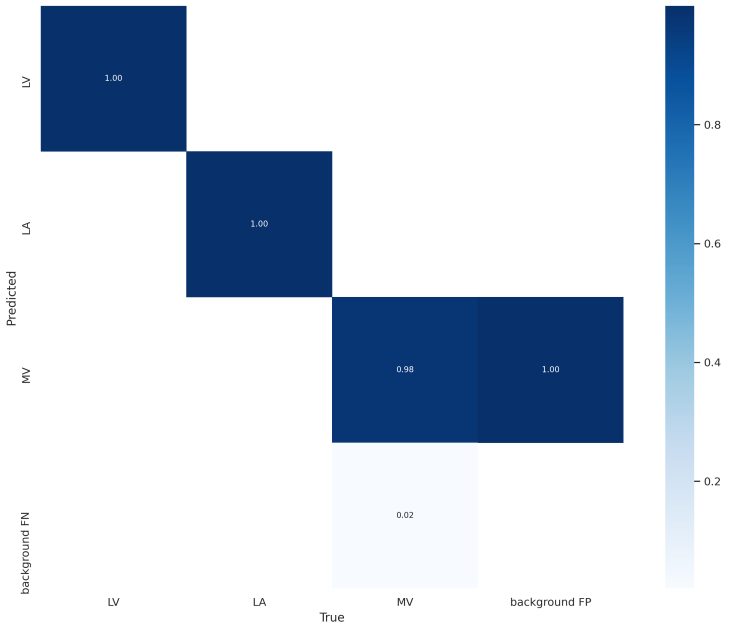
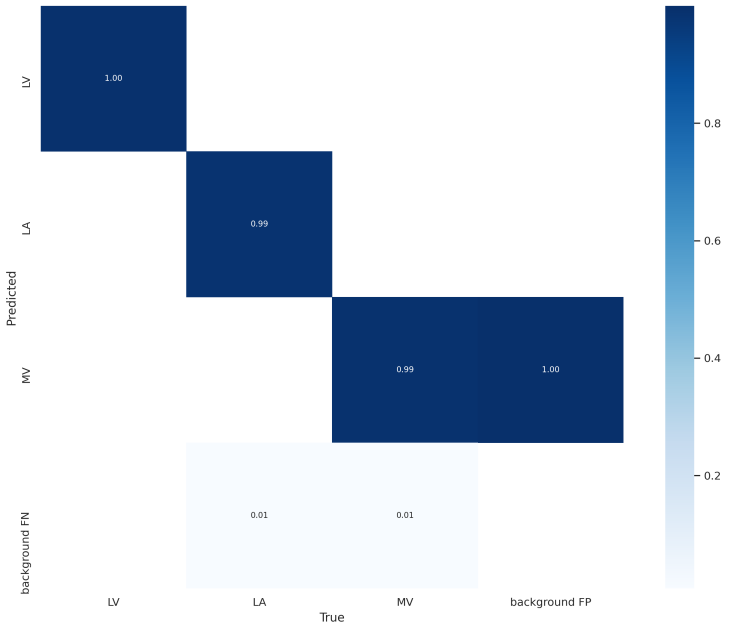


Figure 4.3: Metric evaluation from training without data augmentation with 150 epochs. The training set compared to the validation set shows a slight possibility of overfit during training.

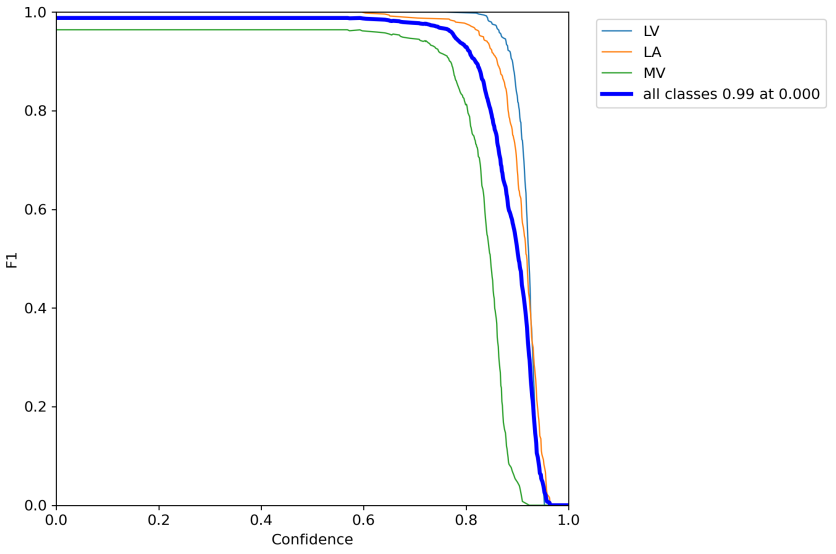


(a) With data augmentation. Specifies a background FP 100% and FN 2% caused by MV.

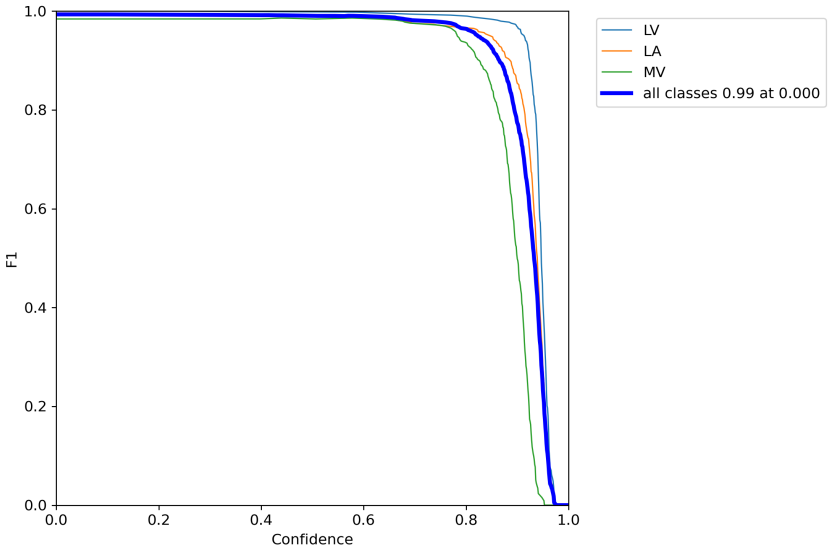


(b) Without data augmentation. Specifies a background FP 100% and FN 1% caused by MV. In addition a background FN 1% caused by LA.

Figure 4.4: Confusion matrix from training with and without data augmentation.



(a) With data augmentation.

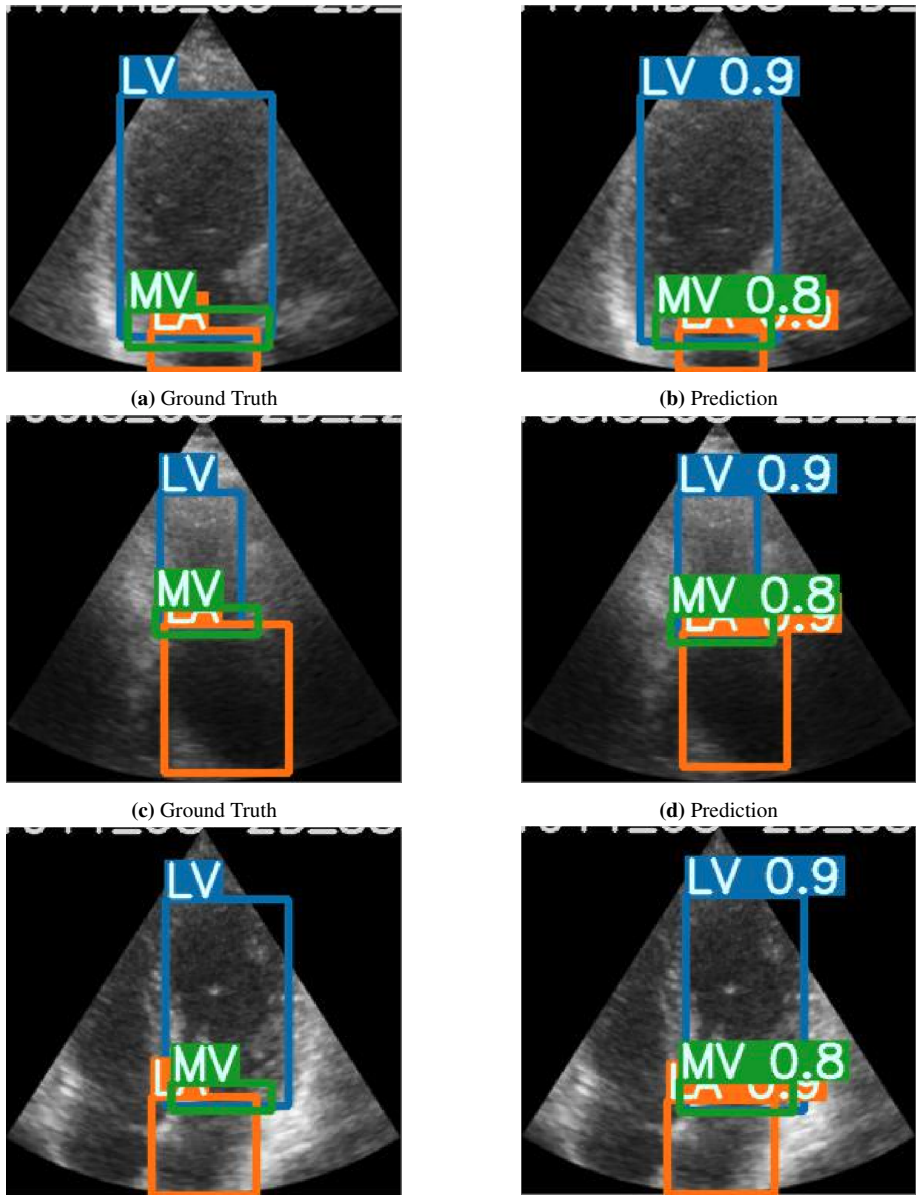


(b) Without data augmentation.

Figure 4.5: F1 - score for with and without data augmentation. A slightly higher score for MV on the training without data augmentation.

4.2.2 Ground truth vs. predicted

Figure 4.6 shows how the testing of the model with data augmentation predicts compared to the ground truth annotation on the validation set.



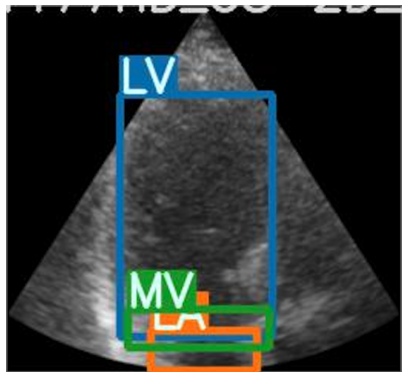
(a) Ground Truth **(b)** Prediction

(c) Ground Truth **(d)** Prediction

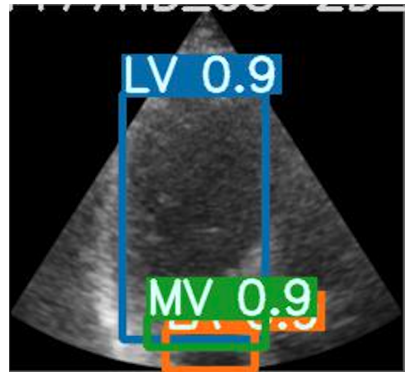
(e) Ground Truth **(f)** Prediction

Figure 4.6: The ground truth versus predicted value with data augmentation.

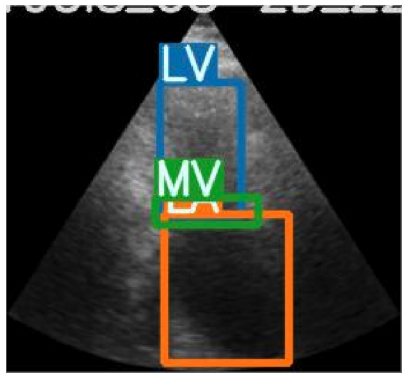
The ground truth is also compared to the predicted in Figure 4.7, with the model without data augmentation on the validation set.



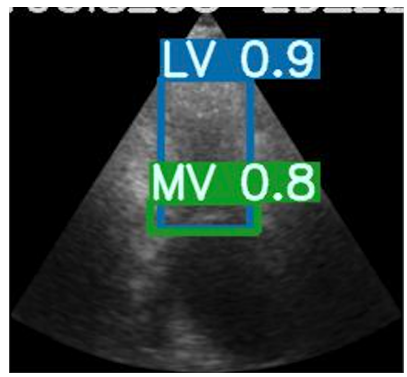
(a) Ground Truth



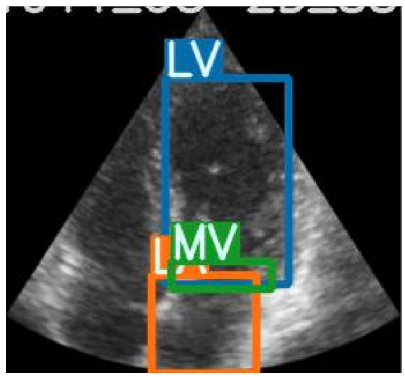
(b) Prediction



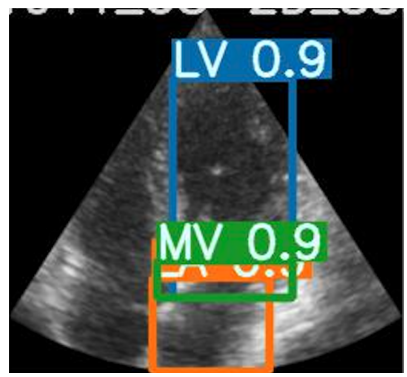
(c) Ground Truth



(d) Prediction



(e) Ground Truth



(f) Prediction

Figure 4.7: The ground truth versus predicted value without data augmentation. In (d) the model fail to locate LA.

4.2.3 Inference

The inference on the HUNT dataset with and without data augmentation are compared in Figure 4.8.

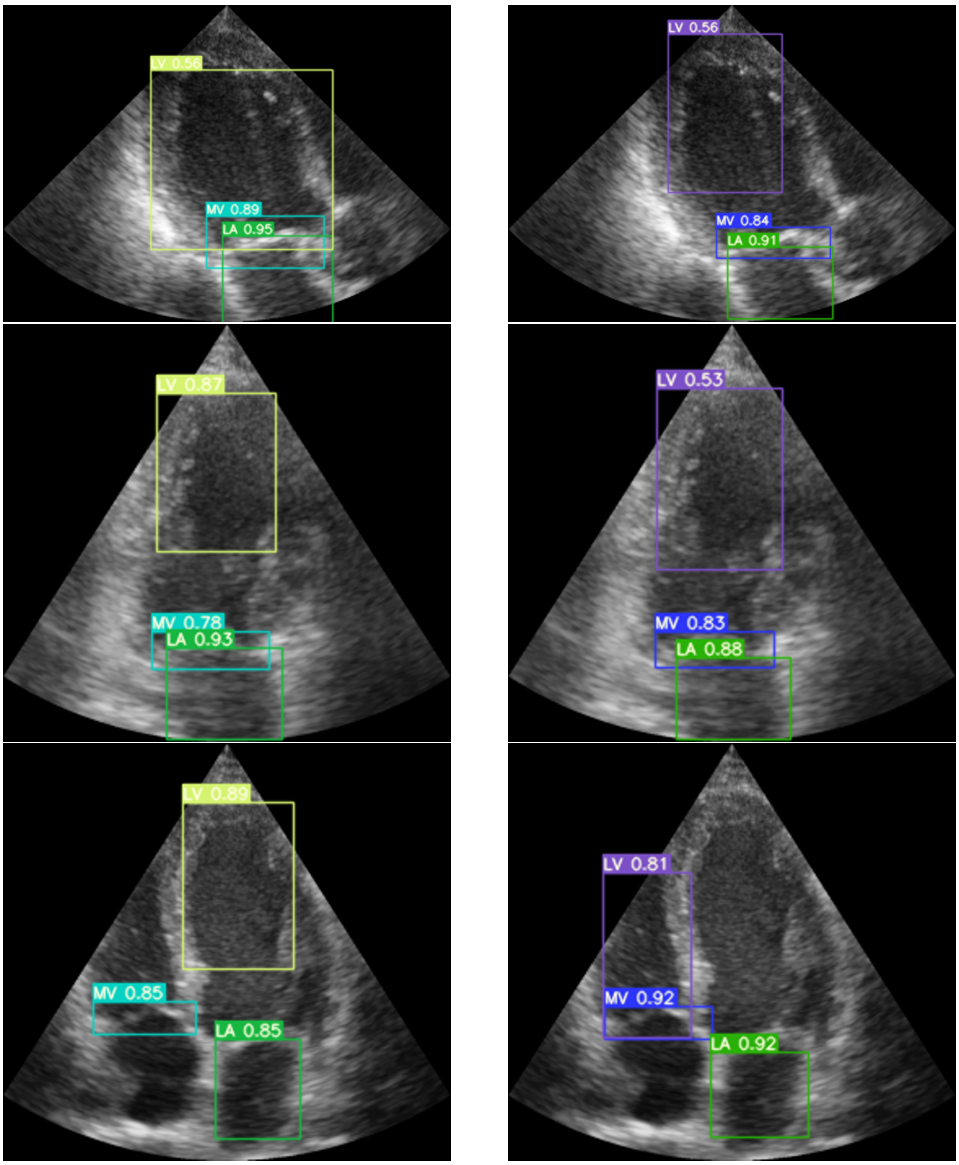


Figure 4.8: Inference with (left) and without (right) augmentation. The cardiac views shown in right order: ALAX, A2C and A4C. Both fail to locate MV in A4C, and without data augmentation fail to detect the LV in A4C.

4.3 Dataset size

This section represent the analysis of training, testing and inference of models with the whole HUNT dataset compared to two-third of it. Both cases are trained on the data augmentation modifications presented in the method section. In both tables the best predictions are marked in bold.

4.3.1 Metric evaluation

The overall mAP values for both the training and testing with the two datasets are displayed in 4.4. In addition the mAP values for each class in both cases are shown in Table 4.5.

Table 4.4: The overall mAP values for training with two-third versus the whole dataset. The best results are marked in bold.

Training set size	mAP@.5 train	mAP@.95 train	mAP@.5 test	mAP@.95 test
Whole	0.984	**0.631**	0.984	**0.631**
2/3	**0.993**	0.63	**0.988**	0.606

Table 4.5: The overall mAP values per class for training with two-third versus the whole dataset. The best results are marked in bold.

Class	Whole dataset				2/3 datatset			
	Train		Test		Train		Test	
	mAP@.5	mAP@.95	mAP@.5	mAP@.95	mAP@.5	mAP@.95	mAP@.5	mAP@.95
LV	**1**	0.721	**1**	0.721	**1**	**0.752**	**1**	**0.728**
LA	**1**	**0.67**	**1**	**0.67**	**1**	0.66	0.996	0.622
MV	0.951	**0.502**	0.951	**0.502**	**0.98**	0.479	**0.967**	0.468

The metric evaluation of the training process with the whole and small dataset is shown in Figure 4.9 and Figure 4.10. In addition both the confusion matrix and the F1-curve comparison of the two dataset are shown in Figure 4.11 and 4.12.

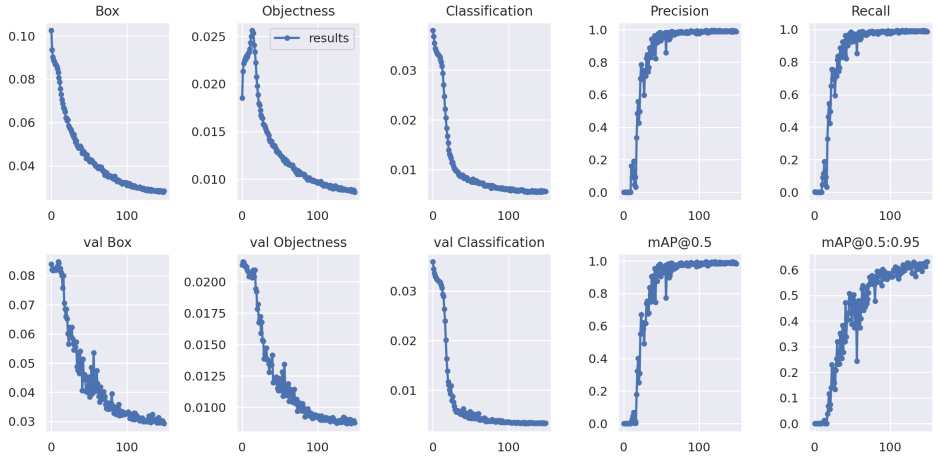


Figure 4.9: Metric evaluation from training with the whole dataset and with 150 epochs. The training set compared to the validation set shows a overall good fit.

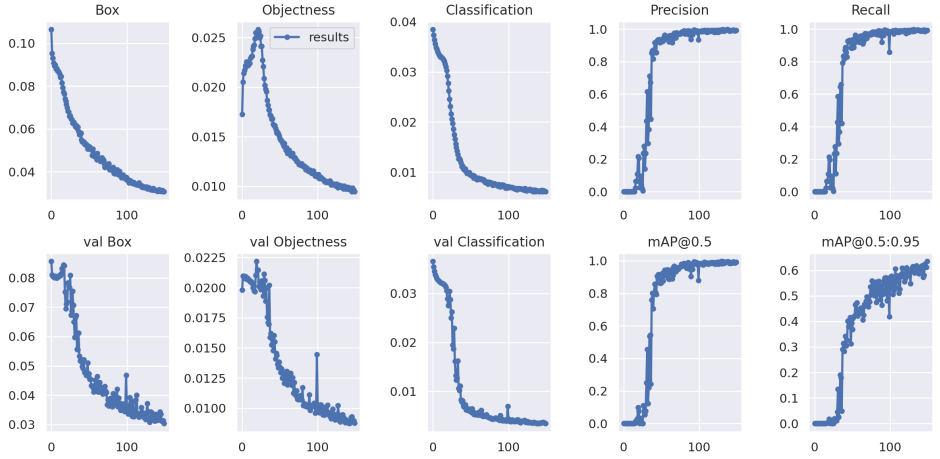
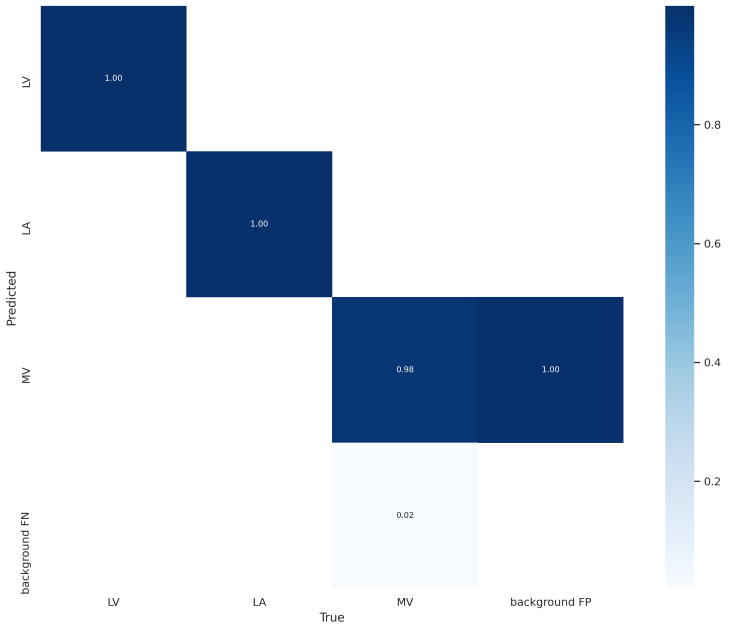
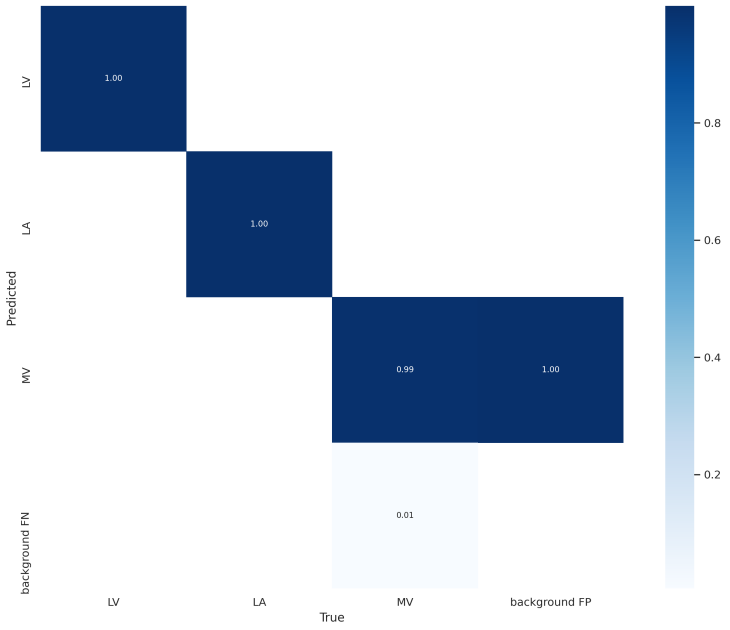


Figure 4.10: Metric evaluation from training with the small dataset and with 150 epochs. The training set compared to the validation set shows a overall good fit.

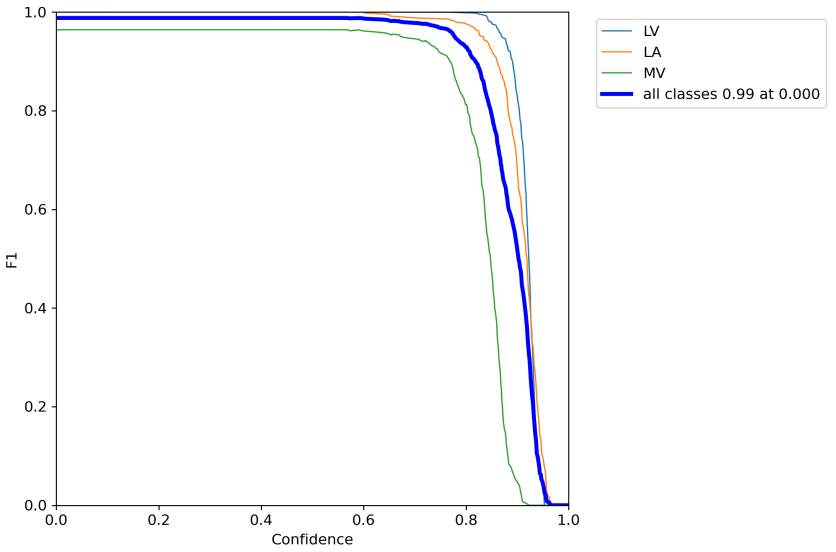


(a) The whole dataset. Specifies a background FP 100% and FN 2% caused by MV.

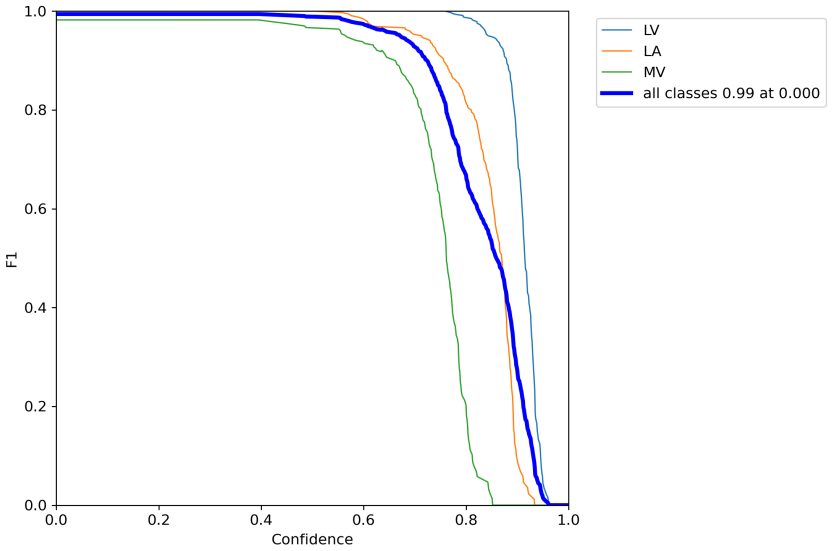


(b) The small dataset. Specifies a background FP 100% and FN 2% caused by MV.

Figure 4.11: Confusion matrix from training with the different dataset sizes.



(a) The whole dataset.

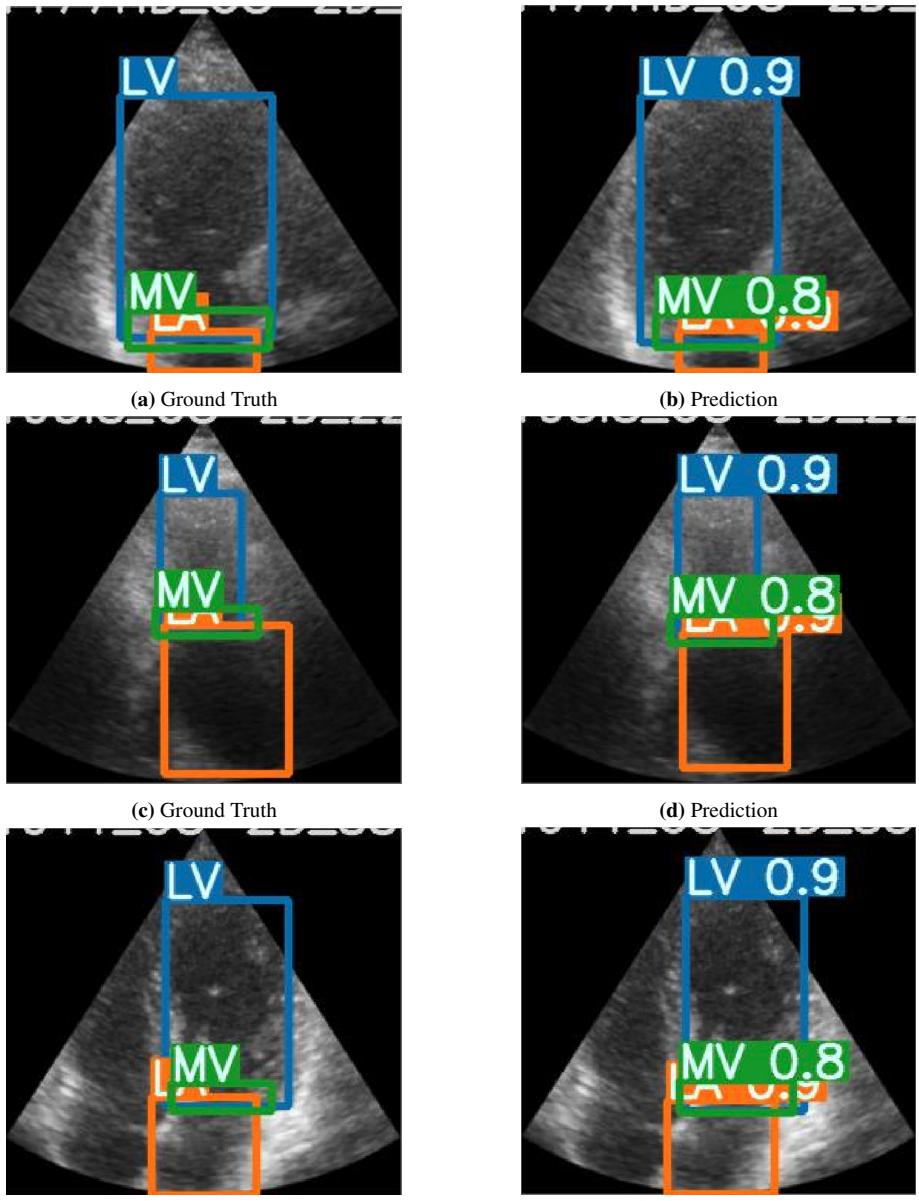


(b) The small dataset.

Figure 4.12: F1 - score for the two datasets. The score is overall higher for the model trained on the whole dataset.

4.3.2 Ground truth vs. predicted

In Figure 4.13 the prediction versus the ground truth annotation of the test of the model with the whole dataset is shown. Here tested on the validation set.



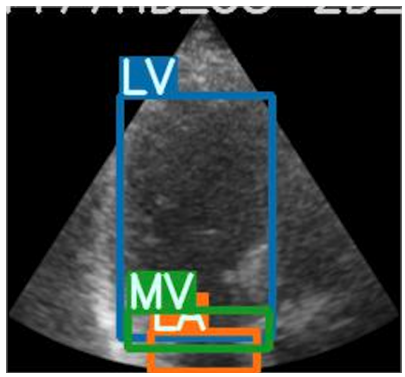
(a) Ground Truth **(b)** Prediction

(c) Ground Truth **(d)** Prediction

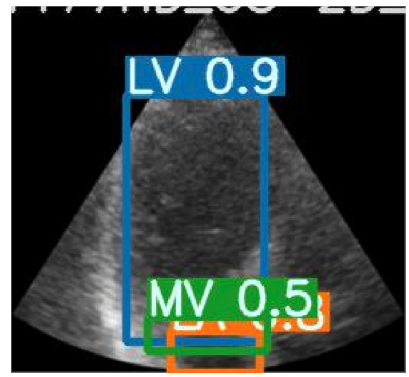
(e) Ground Truth **(f)** Prediction

Figure 4.13: The ground truth versus predicted value on the whole dataset.

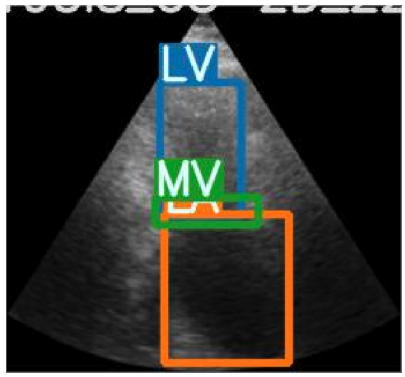
Figure 4.14 shows a comparison of the ground truth and the prediction with the model tested on 2/3 of the dataset. Here also on the validation set.



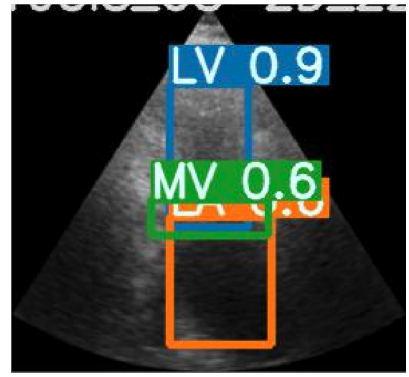
(a) Ground Truth



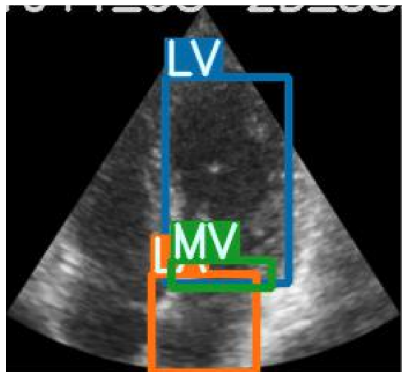
(b) Prediction



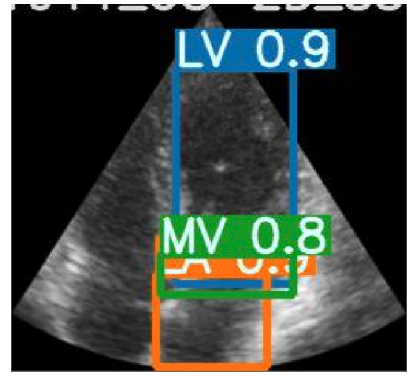
(c) Ground Truth



(d) Prediction



(e) Ground Truth



(f) Prediction

Figure 4.14: The ground truth versus predicted value on two-third of the dataset. Shows low predictions of MV in (b) and (d).

4.3.3 Inference

The inference on both the whole and 2/3 of the HUNT dataset are compared in Figure 4.15.

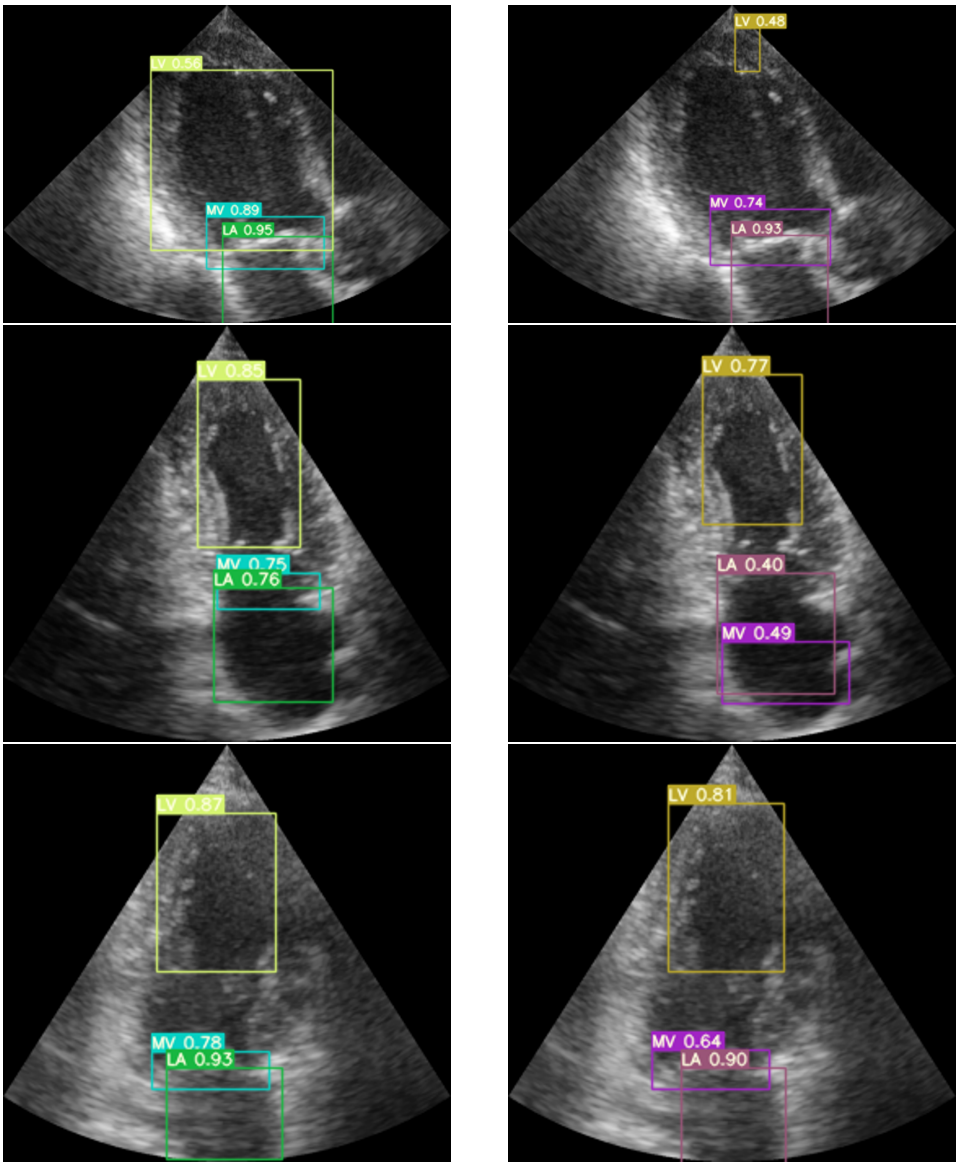


Figure 4.15: Inference on the whole (left) and the small (right) dataset. The cardiac views shown in right order: ALAX, A2C and A2C. The small dataset is unable to fully locate the LV in ALAX. Further, it gives low predictions on LA and MV in the first image in A2C.

4.4 Standard and nonstandard cardiac views

This section shows how the object detection model behaves on never seen and new data in both standard and nonstandard views. Figure 4.16 displays inference with standard views A4C, A2C and ALAX with the best weights from the training with the whole HUNT dataset and with data augmentation. Figure 4.17 represents the inference with nonstandard views named A42C, ANE and ANS, here also with the best weights from training with data augmentation on the whole HUNT dataset.

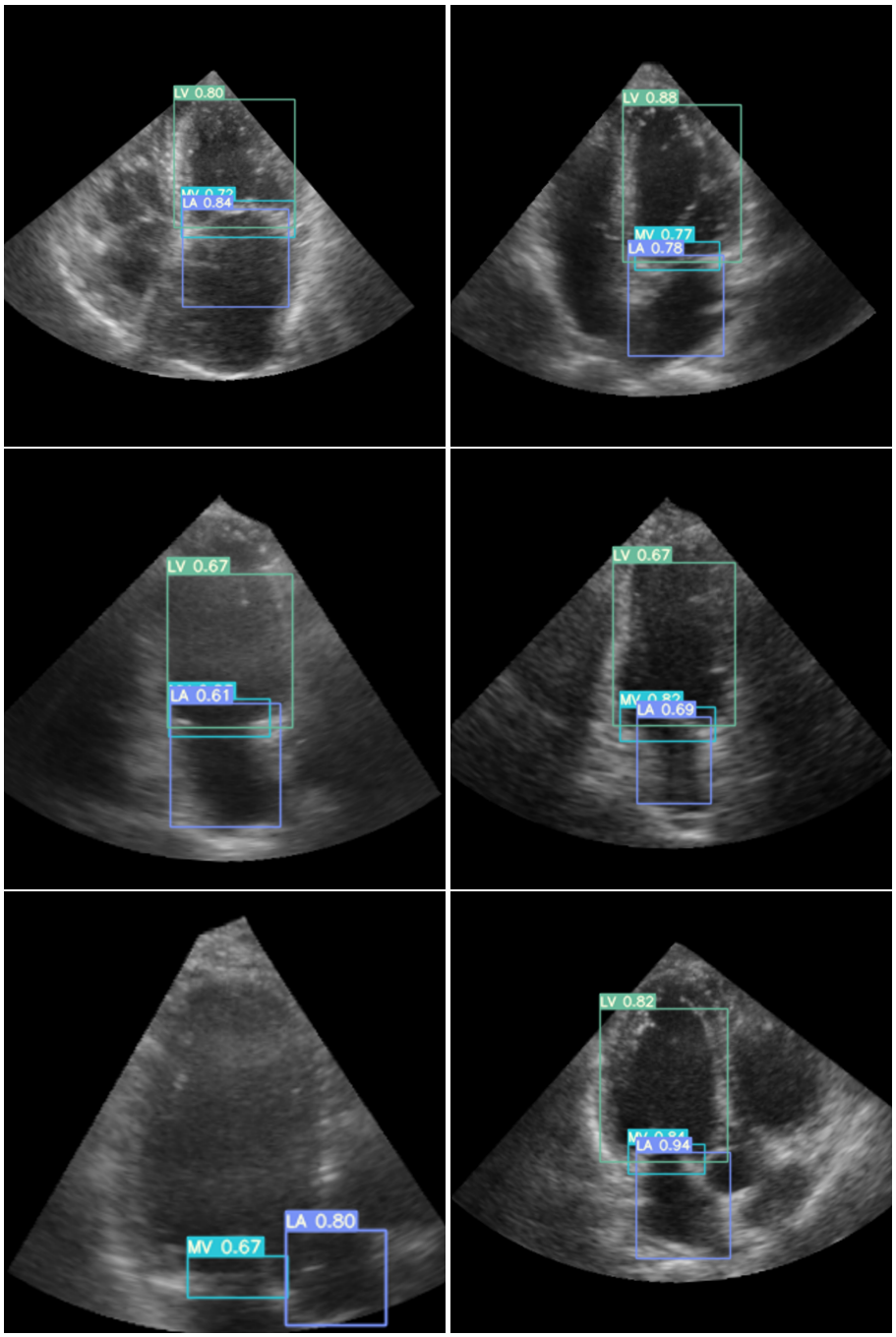


Figure 4.16: Inference of the sliced 3D data with the standard views. The cardiac views shown in right order: A4C, A2C and ALAX. The model fails to detect LV and LA in the first image in ALAX.

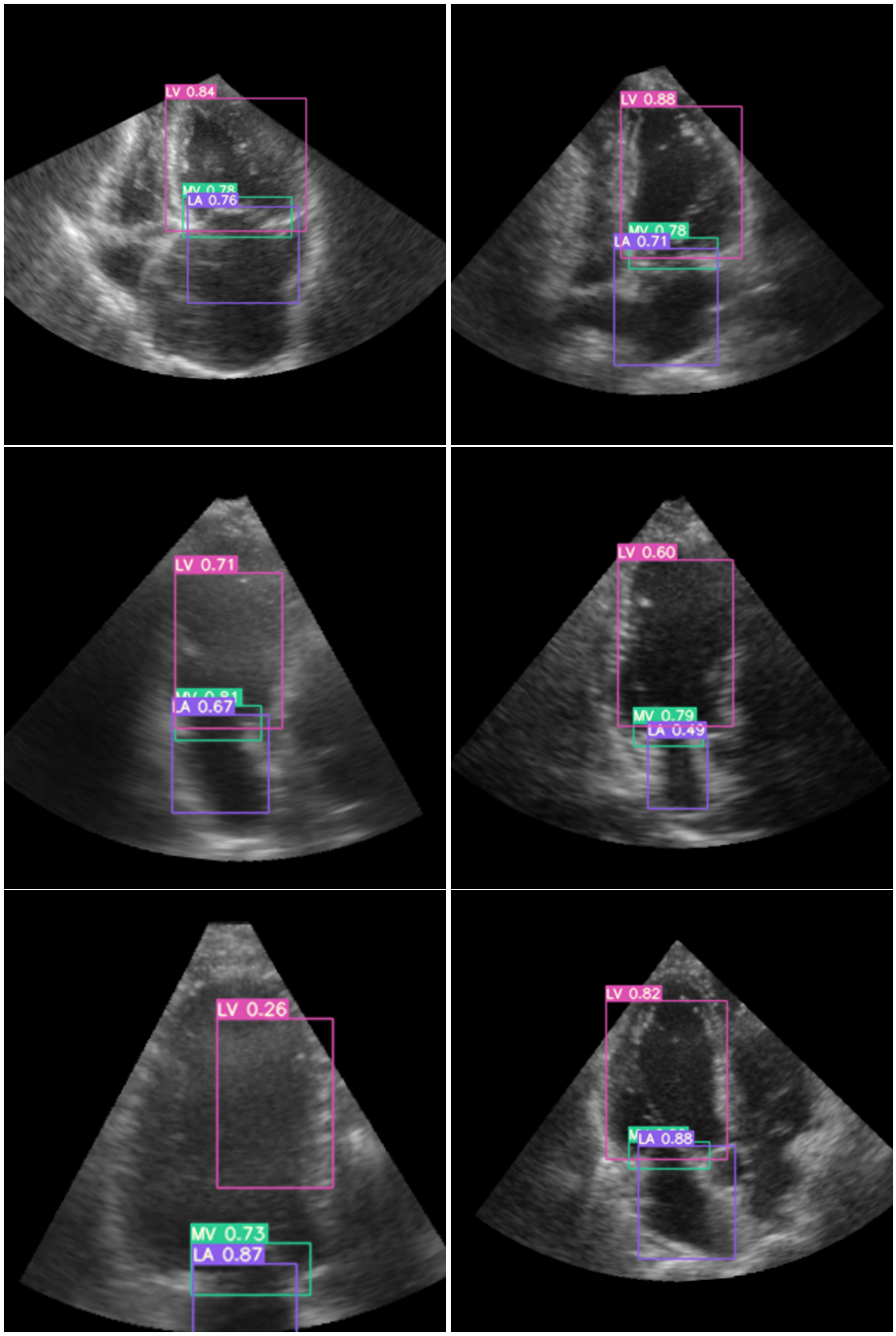


Figure 4.17: Inference of the sliced 3D data with the nonstandard views. The cardiac views shown in right order: ANE, ANS and A42C. The model gives low prediction score on LV in the first image in ALAX.

Chapter 5

Discussion

In this part of the project thesis the method and results are described and evaluated. First the different YOLOv5 models are reviewed using the HUNT dataset. Then, an analysis of how the network behaves on smaller dataset, with and without data augmentation and on standard and nonstandard views are presented. In the end, the data, alongside the data preparation, are discussed.

5.1 YOLOv5 - model comparison

Figure 4.1 shows how the accuracy (mAP@.5) versus the speed (ms/img) is for the different YOLOv5 models. Here one can see that the speed is significant better for the small model. The accuracy on the other hand, is slightly higher for the larger models. Even though, the small model's accuracy is not far behind for a resolution of 256 or 640. Therefore, comparing the speed versus the accuracy the small model appears to be a better choice if one needs a fast, but still quite accurate model when training on images with a resolution of 256.

In Table 4.1, one can see the different YOLOv5 models compared to each other. The overall mAP values are better for the YOLOv5x, both with an IoU equal to 0.5 and between 0.5 and 0.95. However, the mAP values for the other models are fairly close the accuracy generated by the x-large model. The amount of parameters are significant larger as we increase the size of the models. Furthermore, the GFLOPS (giga floating-point operations per second) values also increase as the models increase. The GFLOPS gives a value of how many computations the models do and indicates the model's computational power. A high value of GLFOPS and parameters makes the model more complex, requiring more GPU memory.

The HUNT dataset contains images that are small in size and stored in grayscale. Because the dataset is small, there is a higher risk of overfitting. As a result, complex models with a large number of parameters should be avoided. The small model, YOLOv5s, is therefore the best choice when training on the HUNT dataset. If one were to increase the data and resolution in the dataset, it is probably beneficial to choose a slightly more complex model with more parameters, like YOLOv5m.

5.2 Data Augmentation

5.2.1 Evaluation metric

The mAP@.5 values in Table 4.2 are greater in both training and testing for the model without any data augmentation. On the other hand, the mAP@.95 score are better for the model with data augmentation compared to the model without. The mAP scores per class, shown in Table 4.3, indicate that the model with data augmentation have better overall predictions on LA on both training and testing compared to the model without data augmentation. However, the model with data augmentation appears to have lower predictions for the MV in comparison to the other model. Furthermore, the predictions for LV when IoU is 0.5 are the same for the models in both training and testing, where both score a mAP@.5 equal to 1. The mAP@.95, on the other hand, is slightly better for the model without augmentation.

In Figure 4.3, the metrics for the training of the model without data augmentation is shown. Here it appears that the model is overfitted as the objectness loss's curve at the validation set start to increase a bit compared to the objectness loss for the training set. In addition, the Precision, Recall and both mAP values have a "rocky" start before it stables out to its final score, compared to the model with data augmentation, shown in Figure 4.2.

The confusion matrix for the models are quite similar, as seen in Figure 4.4. The matrix specifies that the background FP for both models are 100% caused by the MV class. The background FN is also caused for MV in both cases, with an exception of the model witout augmentation where both LA and MV is the cause. Nevertheless, the values are only between 1-2%. The background FP and FN can be caused by the way the annotations are done for the MV. During training, the model may define space within bounding boxes to be positive for MV, but also area outside of boxes to be negative for that class. Even so, the true versus the predicted values are high above for 0.98 for all the classes for both of the models.

The F1-score is shown in Figure 4.5. With the exception of MV, which appears to have a slightly higher F1-score for the model without augmentation, the score for both models appears to be relatively equal.

5.2.2 Testing on validation set

When comparing the ground truth versus the predicted bounding boxes with the model without data augmentation, as shown in Figure 4.7, the prediction score is quite accurate for all classes. Additionally, the prediction score for MV is occasionally greater than the prediction score for the model with data augmentation, Figure 4.6. Except for LA and LV, where the score is the same for both models. Nevertheless, the model without data augmentation sometimes fails to predict LA, as shown in the image in the middle in Figure 4.7. This may be due to the possibility of overfitting as mentioned in the section over.

5.2.3 Inference

In Figure 4.8, one can see how the two models behave on never seen data from the test set. The first image, which is in ALAX view, shows that the prediction score for LV for both models are the same. However, the detection of the LV is more accurate at model with data augmentation as the bounding box seem to localize the LV better than the other model. The prediction of the LV on the next image, in the A2C view, shows that both models have problems locating the whole left ventricle. Here the model without data augmentation have a lower precision than the one with data augmentation. For the last image, which is in the A4C view, both models fail to locate the MV as they both believe MV is the aortic valve (AoV). The model without data augmentation also fails to locate the LV, and rather detects LV as right ventricle.

After taking everything into account, the model with the data augmentations is chosen as the one to proceed with. This model seems to be more robust for detecting the different structures in the human heart when it comes to the overall mAP@.95 and @mAP.5 for both LV and LA, as well as the prediction versus the ground truth and prediction and results in the inference. Not to mention the possibility of overfitting from the model without data augmentation.

5.3 Dataset size

5.3.1 Evaluation metrics

The best mAP values for both the whole dataset and 2/3 dataset from training and testing on HUNT data is shown in Table 4.4. Here one can see that the mAP@.5 is slightly greater on both training and testing for the small dataset compared to the dataset containing all the data. However when it comes to the mAP@.95, the model trained with whole dataset gets a higher value than the model with the small dataset. Even so, based on the mAP@.5 values, the small dataset appears to perform better than the larger dataset. When comparing the mAP per class, shown in Table 4.5, the models achieve the same score for both LA and LV in training for IoU equal to 0.5. The same implies for LA in test, but for LV the training with the whole dataset generates a higher mAP@.5. MV's mAP@.5, however, is better for the small dataset.

The mAP@.95 for both the LA and MV have a higher score for the whole dataset compared to the small dataset, with exception to the LV where the small dataset computes a higher mAP@.95. Nevertheless, this points out that the model with the whole dataset gets an overall higher mean average precision for a IoU ratio between 0.5 to 0.95.

The metric curves in Figure 4.9 and 4.10, indicates that the model with the whole dataset has a higher overall recall and precision score. This also implies for the F1-score in Figure 4.12 which shows that the model with the large dataset have a better accuracy over all the classes than the model with the small dataset. This is especially the case for the MV class. Since the mAP score is higher for the model with two-thirds of the data, but the recall, precision, and F1-score are higher for the model with the entire dataset, it means that the confidence threshold is better for the model with the large dataset. This indicates that as the confidence threshold increases, the recall value decreases. The confusion metrics in Figure 4.11 demonstrates that the background FP is a 100% caused by the MV. The same implies for the background FN, but only by a value of 1% for the small dataset and 2% for the large. As stated in 5.2.1, this can be due to the ground truth annotations made on the data or that the network defines space around the MV in the bounding boxes as positive and space outside of the bounding box as negative.

5.3.2 Testing on validation set

To get a better understanding of how the model behaves it is important to look on the predicted bounding boxes versus the ground truth annotations for the validation set. Figure 4.13 shows that the training with the whole dataset generates good predictions for the three classes: LV, MV and LA. In addition, the predicted bounding boxes appear to be quite equivalent to the ground truth bounding boxes. The small dataset's ground truth versus predicted comparison, shown in Figure 4.14, predicts lower precision in some cases for MV than the test with the whole dataset. Furthermore, the bounding boxes drawn for MV in Figure 4.14 (b) is more accurate than the model precision value tells us (MV = 0.5). By comparing the results obtained from the evaluation metrics and the test on validation set, it seems that the model with the small dataset's predictions is not as accurate as the other model. This because it manages to locate the MV nearly perfectly, but still gives out a low score. It is probably due to the fact that the recall value decreases for higher confidence as stated over.

5.3.3 Inference

In Figure 4.15, the inference of the two datasets is presented. This part of the analysis demonstrates how the model performs on new data, which in this case is the HUNT dataset's test set. It appears that the model trained on the entire dataset is better at detecting the different classes. The first image in the figure shows that the small dataset is unable to locate LV, but still gives a precision of 0.48. This estimate is considerably off when compared to how much of the bounding box actually detects the LV. The large dataset yields a value of 0.65 for locating the LV on the same image. This value is relatively accurate because the bounding box does not capture the entire left ventricle and locate some of the MV.

The next image, the model with the large dataset seems to locate the classes better than the other model. The LV is drawn more around the left ventricle, including having a higher prediction score. The LA is also more accurately drawn for the model with the whole dataset, even though they are quite similarly drawn. The prediction score is also higher and more accurate than the model with the small dataset. The model with the small dataset generate a prediction below 0.5 for both MV (0.49) and LA (0.4), which indicates a bad detection. In addition, the detection of MV for the model trained on the small dataset is far off, and is localized in the end of the left atrium. In the last image the models seem to locate the different classes nearly the same. Both detections are quite accurate for both the LA and MV. By looking close, one can see that the LA is better drawn for the model with the large dataset and the prediction score for these two classes are slightly higher as well. When it comes to locating the LV, both models seem to have problems drawing a bounding box around the whole left ventricle. In addition, the models give out a prediction score that is higher than what they actually detects. This could be caused by the blur in the image and the lack of clear view of the different structures.

Overall, the different results indicate that the model is not robust enough for a small amount of data. Despite that the metrics are not particularly poor for LV and LA, the inference indicates that the model needs to be trained on more data when training on this particular dataset.

5.4 Standard and nonstandard cardiac views

The inference on the standard, Figure 4.16, and nonstandard views, Figure 4.17, shows if the chosen model with data augmentation trained on the whole dataset is robust enough. All of the classes are correctly located by the inference on the first two images in the standard views, which in this case are in A4C view. In the first image, a precision score of 0.84 on LA is quite good, but looking at the prediction one can see that the bounding box is not fully drawn over the left atrium. Therefore a score of 0.84 is too high in this particular case. The LV on the other hand gives a rather accurate score of 0.8 as it appears to locate the whole left ventricle, but gets some of the mitral valve in its bounding box. The next image gives good predictions and localizes all the classes quite accurate to the score given. The third image (A2C view) gives accurate predictions compared to the bounding boxes drawn. Furthermore, the model manages to detect all the classes, despite the image being hard to interpret because of the blur. The same occurs for the three first images in the nonstandard views as well, which are here in the A42C and ANS view.

In the fourth image (A2C and ANS view) both standard and nonstandard detect all the classes and this with a suitable prediction for the MV (0.82, 0.79) and LV (0.67, 0.60), as the bounding boxes are not fully drawn. Despite this, the model gives a lower prediction to the LA than what it actually draws around the left atrium. In the next image (ALAX view) for the standard view, one can see that the LV is not predicted at all and that the bounding box for LA is not correct. In the nonstandard one on the other hand (ANE view), the model predicts all the classes correct for the fourth image. The only issue, is that the bounding box is not completely drawn over LV and this results in a bad prediction of 0.26, which indicates a bad detection. Why the model cannot seem to fully localize the left ventricle can be due to the left ventricle being rather large, as it takes up nearly the entire echocardiographic image.

The inference of the last image for both the standard and nonstandard dataset, the bounding boxes are entirely drawn around the MV and LA, and almost so for LV. The predictions are also quite accurate and reliable. Overall, the inference indicates that the model can localize the classes when the LA, MV, and LA are in focus and the image quality is high.

5.5 Data

It is necessary to analyze the data used in training, testing, and inference in order to evaluate the performance of the object detection model. The data retrieved from the HUNT study consist of a total of 1260 images and corresponding annotations. Furthermore, the echocardiographic images are either in the A4C, A2C or ALAX view, meaning that in some cases the right atrium, right ventricle and aortic valve is present in the image. This seems to be a problem in some cases for the A4C, where the models locate the LV as the right ventricle and the MV as the aortic valve. This also happens to the LA in the ALAX view in the analysis on the Forshortening2021 dataset, where it is located at the right atrium. That gives us an indication that the model have either not trained enough on images in the A4C and ALAX view or that there is not enough variation in the data in the training, validation and test set. Another reason can be the ground truth annotations. Even though they turned out quite accurate, there may be some small margins that could be improved or in some instances there could be too much space in the bounding boxes. The latter can cause the model to learn undesirable patterns, resulting in false positive detections.

Looking at the results and discussion for the analysis for the entire dataset versus two-thirds of the dataset, it appears that more data is possibly required. The data should probably contain a more variation in angles of the views, as this seems to be a problem for the model, as stated in the analysis of the standard and nonstandard views in the Forshortening2021 dataset. In addition more variation of the image quality could help the model to increase its learning ability. In some of the images it can be difficult to interpret the different structures as there is noise or blur in the image. The model can therefore learn from this noise, resulting in bad predictions.

5.6 Further work

There is still some adjustment and improvements that should be done in the further work for improving the model's performance in both training and inference. For starters, to avoid potential overfitting and underfitting and achieve more accurate prediction, gathering more data can be beneficial. The data provided for training should not contain images with a lot of noise or interference. Additionally, there should be more variations in both views and the focus point, meaning which chambers that has focus in an image. This might help to get more accurate detection in the inference. Another improvement on the data preparation can be the ground truth annotations. As mentioned in the methodology in 3.1.2, the ground truth are done by clinicians for segmentation purpose. From here bounding boxes are made for the LA and LV masks, and MV gets a box in the transition between LA and LV, with 2/3 of the box in LV and 1/3 in LA. Despite the fact that the annotations were relatively precise, inaccuracies might have occurred. By refining these, the accuracy of training and inference may increase. It is also worth mentioning that the data splitting in training, test and validation, can be explored further in terms of splitting

percentage or greater variation of views and frames of the 2D images.

Even though improvement of the NMS in inference made the model only choose one bounding box for each class, there is still some other implantation one can look into. An interesting take, could be to give the network information about which view and frame the images is in. This allows the model to make more accurate predictions while avoiding being distracted by other parts of the structure of the human heart when detecting the different classes/structures. Furthermore, this could make it easier to interpret where the model may fail. Despite the fact that data augmentation has been thoroughly evaluated in this thesis, it may be useful to explore more tuning of the hyperparameters or to introduce new data augmentations.

When training the object detection model, it is critical to be cautious and precise while tuning the different training parameters. Experimenting with the training steps, batch size, and image width, for example, can be interesting. The image width, is especially an interesting experiment as the resolution of the 2D images are only 256, and by up-scaling the images while training can be beneficial. However, the model will most likely become more complex, requiring the use of a larger model. Furthermore experimenting with the inference parameters can be interesting. By for example increasing the confidence threshold one can discard bad prediction.

At last, a natural step would be to check how the object detection model behaves on videos of echocardiographic images. This way one can see how it behaves in real-time. This was not done in this thesis as it is crucial to have a model that is robust enough on different, unseen echocardiographic images before testing video segments.

Chapter 6

Conclusion

In this master thesis, a deep learning method was presented for automatically detection of the different structures of the human heart in echocardiographic images. The method used was the real-time object detection algorithm YOLOv5. The algorithm was trained and tested for locating the left ventricle, left atrium and mitral valve. The acquired data was retrieved from the HUNT study which contains images in the A2C, ALAX and A4C views. For improving the adaptability of the model, different data augmentation techniques were implemented and used while training the model. This proved to be useful as the model's performance increased. Furthermore, for testing if the final model is robust enough, two dataset was created with both standard and nonstandard views.

The metric evaluation in the results, indicates that YOLOv5s produces overall adequate predictions when training on the echocardiographic images. The model managed to detect the left ventricle (LV), mitral valve (MV) and left atrium (LA) when they are in focus or there is no noise or interference in the image. However, the model tends to fail to detect the LV and MV in images in the apical four-chambers view, where the right ventricle, right atrium and aortic valve appears. This also occurs for the LA in the apical long-axis view. The model still manages to detect the structure with good predictions on new, unseen data in both standard and nonstandard views. Although, the bounding boxes are sometimes not entirely drawn around the structures. The results showed by implementing and using more data augmentations can benefit the model to learn from more varying data. Hence, using more data and more variation of the different views would most definitely be a solution to the issues addressed. As a result, the model's learning ability will improve, leading to better detection on new and unseen data.

In conclusion, the object detection model YOLOv5s can fully detect the different structure in the human heart in images of high quality and when the desired structures are in focus. Therefore, more data with better quality and more variation in echocardiographic

views and viewpoints should be employed to enhance the model's robustness and overall performance. If the model becomes more complex, a larger YOLO model should be implemented.

Bibliography

[1] P. Davidovits. *Physics in Biology and Medicine, 5th ed.* 5th. Academic Press - Elsevier Inc, July 2019. ISBN: 978-0-12-813716-1. DOI: https://doi.org/10.1016/C2016-0-03726-9.

[2] A. Støylen. *Strain rate imaging. Myocardial deformation imaging by ultrasound / echocardiography.* [accessed 14-April-2021]. 2018. URL: https://folk.ntnu.no/stoylen/strainrate/.

[3] M. Brekke and A. Borthne. *radiograf*. [accessed 12-June-2021]. URL: https://sml.snl.no/radiograf.

[4] M. Kim et al. "Deep Learning in Medical Imaging". In: (Dec. 2019). DOI: https://doi.org/10.14245/ns.1938396.198.

[5] R. Cuocolo et al. "Current applications of big data and machine learning in cardiology". In: *Journal of Geriatric Cardiology* 16 (Aug. 2019), pp. 601–607. DOI: http://www.jgc301.com/en/article/doi/10.11909/j.issn.1671-5411.2019.08.002.

[6] M. Yang et al. "Deep RetinaNet for Dynamic Left Ventricle Detection in Multiview Echocardiography Classification". In: *Scientific Programming* (Aug. 2020). DOI: https://doi.org/10.1155/2020/7025403.

[7] X. Zeng et al. "Deep Learning for Ultrasound Image Caption Generation based on Object Detection". In: *Neurocomputing* 392 (Apr. 2019). DOI: 10.1016/j.neucom.2018.11.114.

[8] A. K. Jansen and L. Løvstakken. "Automatic annotation (of structures) in echocardiography using deep learning (Project thesis)". In: (2021).

[9] T. P. Rangul and K. Bøhle. *Hjertet*. [accessed 29-May-2021], License: CC BY-SA. June 2019. URL: https://ndla.no/subject:1:22dee9ab-5b1a-4c23-8c97-c68107b881bb/topic:2:77162/topic:2:188655/resource:1:109776.

[10] J. G. Betts et al. *Anatomy and Physiology*. OpenStax, Houston, Texas, 2013. URL: https://openstax.org/books/anatomy-and-physiology.

[11] Wapcaplet. *Diagram of the human heart*. License: CC-BY-SA-3.0. 2019. URL: https://en.wikipedia.org/wiki/File:Diagram_of_the_human_heart.svg.

[12] O. College. *Phases of the Cardiac Cycle*. License: CC-BY-SA-3.0. 2013. URL: https://upload.wikimedia.org/wikipedia/commons/3/38/2027_Phases_of_the_Cardiac_Cycle.jpg.

[13] P. R. Hoskins, K. Martin, and A. Thrush. *Diagnostic Ultrasound : Physics and Equipment. 2nd ed.* Cambridge University Press, 2010.

[14] S. T. Reeves et al. "Basic perioperative transesophageal echocardiography examination: a consensus statement of the American Society of Echocardiography and the Society of Cardiovascular Anesthesiologists". In: *Journal of the American Society of Echocardiography : official publication of the American Society of Echocardiography* 26,5 (2013). PMID: 23622926, pp. 443–456. DOI: https://doi.org/10.1016/j.echo.2013.02.015.

[15] C. Mitchell et al. "Guidelines for Performing a Comprehensive Transthoracic Echocardiographic Examination in Adults: Recommendations from the American Society of Echocardiography". In: *Journal of the American Society of Echocardiography : official publication of the American Society of Echocardiography* 32,1 (2019). PMID: 30282592, pp. 1–64. DOI: https://doi.org/10.1016/j.echo.2018.06.004.

[16] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. Prentice Hall Press, 2010. ISBN: 0136042597.

[17] A. Tidemann and A. C. Elster. *Maskinlæring*. [accessed 18-May-2021]. 2019. URL: https://snl.no/maskinl%C3%A6ring.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[19] H. Dvergsdal. *nevralt nettverk*. [accessed 19-May-2021]. 2019. URL: https://snl.no/nevralt_nettverk.

[20] C. Nwankpa et al. *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. 2018. arXiv: 1811.03378 [cs.LG].

[21] S. Haykin. *Neural Networks and Learning Machines*. 3rd. Pearson Education, 2009. ISBN: 0131471392.

[22] M. Cheatsheet. *Backpropagation*. 2017. URL: https://ml-cheatsheet.readthedocs.io/en/latest/backpropagation.html.

[23] L. Alzubaidi et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *Journal of Big Data* 53,8 (Mar. 2021). DOI: https://doi.org/10.1186/s40537-021-00444-8.

[24] A. Karpathy. *Convolutional Neural Networks: Architectures, Convolution / Pooling Layers*. [accessed 21-May-2021]. URL: https://cs231n.github.io/convolutional-networks/.

[25] Aphex34. *typical CNN architecture*. License: CC-BY-SA-4.0. 2015. URL: https://upload.wikimedia.org/wikipedia/commons/6/63/Typical_cnn.png.

[26] V. Dumoulin and F. Visin. *Convolution arithmetic - Padding strides odd*. License: MIT, via Wikimedia Commons. 2019. URL: https://commons.wikimedia.org/wiki/File:Convolution_arithmetic_-_Padding_strides_odd.gif.

[27] H. Gholamalinezhad and H. Khosravi. *Pooling Methods in Deep Neural Networks, a Review*. 2020. arXiv: 2009.07485 [cs.CV].

[28] H. Rezatofighi et al. *Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression*. 2019. arXiv: 1902.09630 [cs.CV].

[29] T. K.M. "Confusion Matrix". In: *Encyclopedia of Machine Learning and Data Mining*. Ed. by S. C. and W. G.I. Springer, Boston, MA, 2017, pp. 260–260. ISBN: 978-1-4899-7687-1. DOI: https://doi.org/10.1007/978-1-4899-7687-1_50.

[30] C. Goutte and E. Gaussier. "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation". In: vol. 3408. Apr. 2005, pp. 345–359. ISBN: 978-3-540-25295-5. DOI: 10.1007/978-3-540-31865-1_25.

[31] R. Padilla, S. L. Netto, and E. A. B. da Silva. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2020, pp. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.

[32] Z.-Q. Zhao et al. *Object Detection with Deep Learning: A Review*. 2019. arXiv: 1807.05511 [cs.CV].

[33] P. Soviany and R. T. Ionescu. "Optimizing the Trade-Off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction". In: *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. 2018, pp. 209–214. DOI: 10.1109/SYNASC.2018.00041.

[34] J. Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. DOI: 10.1109/CVPR.2016.91.

[35] J. Redmon and A. Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242 [cs.CV].

[36] J. Redmon and A. Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[37] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[38] Ultralytics. *YOLOv5*. 2021. URL: https://ultralytics.com/yolov5.

[39] L. Wang and W. Yan. "Tree Leaves Detection Based on Deep Learning". In: Mar. 2021, pp. 26–38. ISBN: 978-3-030-72072-8. DOI: 10.1007/978-3-030-72073-5_3.

[40] C.-Y. Wang et al. *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. 2019. arXiv: 1911.11929 [cs.CV].

[41] K. He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916. DOI: 10.1109/TPAMI.2015.2389824.

[42] S. Liu et al. *Path Aggregation Network for Instance Segmentation*. 2018. arXiv: 1803.01534 [cs.CV].

[43] R. Xu et al. "A Forest Fire Detection System Based on Ensemble Learning". In: *Forests* 12 (Feb. 2021), p. 217. DOI: 10.3390/f12020217.

[44] D. M. Montserrat et al. "Training Object Detection And Recognition CNN Models Using Data Augmentation". In: *electronic imaging* 2017 (2017), pp. 27–36.

[45] D. Kumar and A. Ramakrishnan. "Power-law transformation for enhanced recognition of born-digital word images". In: *2012 International Conference on Signal Processing and Communications (SPCOM)*. 2012, pp. 1–5. DOI: 10.1109/SPCOM.2012.6290009.

[46] H. R. C. (NTNU). *The HUNT Study - a longitudinal population health study in Norway*. 2019. URL: https://www.ntnu.edu/hunt.

[47] H. Pettersen et al. *The Forshortening2021 Study*. 2021.

[48] Z. Qin and W. Yan. "Traffic-Sign Recognition Using Deep Learning". In: Mar. 2021, pp. 13–25. ISBN: 978-3-030-72072-8. DOI: 10.1007/978-3-030-72073-5_2.

[49] B. Xu et al. *Empirical Evaluation of Rectified Activations in Convolutional Network*. 2015. arXiv: 1505.00853 [cs.LG].

[50] T.-Y. Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[51] Ultralytics. *yolov5 - version 4.0*. 2021. URL: https://github.com/ultralytics/yolov5/releases/tag/v4.0.

NTNU

Norwegian University of
Science and Technology