

Kåre Fosli Obrestad

Aortic Valve Localisation in 3D Transesophageal Echocardiography Volumes using Deep Learning

Master's thesis in Computer Science

Supervisor: Gabriel Hanssen Kiss

Co-supervisor: Frank Lindseth

June 2022

Kåre Fosli Obrestad

Aortic Valve Localisation in 3D Transesophageal Echocardiography Volumes using Deep Learning

Master's thesis in Computer Science
Supervisor: Gabriel Hanssen Kiss
Co-supervisor: Frank Lindseth
June 2022

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

Abstract

Intraoperative cardiac imaging and the use of hybrid operating theatres for cardiac surgery are becoming more common. 3D transesophageal echocardiography (TEE) enables a harmless way for the physician to see the finer anatomical structures of the heart in real-time. A fully automated alignment of the acquired ultrasound images would allow the images to be presented to the surgeon in an effective manner. A way to do this would be to detect anatomical landmarks of the heart. This could also help in producing standard TEE views for diagnostic purposes.

In recent years, deep learning methods have been applied to increasingly advanced computer vision tasks, such as object detection. The invention of the convolutional neural network (CNN) marked a breakthrough in this area of research. In the last two years, a new class of deep learning methods known as vision transformers have also been introduced, competing with CNNs for the best performance.

Motivated by these innovations in deep learning, this thesis presents a study on the development of such a method for automatic localisation of the aortic valve in 3D TEE volumes. The study wishes to compare the performance of CNNs against transformers and evaluate the strengths and weaknesses of the two network types.

The thesis presents a CNN, a transformer and a hybrid architecture to represent the different network types. The proposed methods use the entire 3D volumes as inputs. They are used as feature extractors for a simple, fully connected regression head, predicting bounding boxes around the aortic valve. Several experiments are done to research how hyperparameter tuning affects the networks. They are evaluated on the mean euclidean distance of the corner points of the bounding boxes.

The models are trained on 84 3D TEE recordings provided by the University Hospital of St. Olav, Trondheim, and annotated by the thesis author.

The best performing method is a 3D version of ResNet-50, which obtained an error of 17.40mm on the test set.

Sammendrag

Intraoperativ bildetagnings av hjertet og bruken av hybride operasjonsstuer for hjerteoperasjoner, blir stadig vanligere. 3D transøsofageal ekkokardiografi (TEE) gir kirurger en måte å se de detaljerte anatomiske strukturene i hjertet i sanntid, som ikke skader pasienten. Automatisk justering av disse ultralydbildene vil gjøre det slik at bildene kan presenteres effektivt til kirurgen. En måte å gjøre dette på, vil være å detektere anatomiske landemerker i hjertet. Dette kan også gjøre det enklere å produsere standardutsnitt av TEE-bilder til diagnostisk bruk.

I løpet av de siste tiårene har "deep learning"-metoder blitt brukt til mer og mer avanserte oppgaver innen datasyntese, slik som objektdeteksjon. Oppfinnelsen av Konvolusjonelle Nevrale Nettverk (CNN), markerte et gjennombrudd i forskningen innen datasyntese. I løpet av de siste to årene har det blitt introdusert en ny klasse nevralt nettverk som heter Vision Transformers. Disse kjemper mot CNNer om hvem som gjør det best i ulike oppgaver innen datasyntese.

Basert på disse nyvinningene innen deep learning, presenterer denne avhandlingen en studie innen utviklingen av en metode for automatisk lokalisering av aortaklaffen i 3D TEE-bilder. Studien forsøker å sammenligne ytelsen til CNNer mot transformere, og evaluere styrkene og svakhetene til de to nettverkstypene.

Avhandlingen presenterer et CNN, en transformer og en hybrid av disse to. Disse skal representere de ulike nettverkstypene. De fremlagte metodene bruker hele 3D volumet som input. De brukes som egenskapsutvinnere til et enkelt fullkoblede regresjonshode som predikerer avgrensingsbokser rundt aortaklaffen. Flere eksperimenter ble utført for å undersøke hvordan hyperparametersøk påvirker nettverkene. Nettverkene evalueres på gjennomsnittet av den euklidiske avstanden fra hjørnepunktene i avgrensingsboksene.

Modellene trenes på 84 3D TEE-bilder gitt av Universitetssykehuset St. Olavs Hospital i Trondheim. Disse ble annotert av forfatteren av avhandlingen.

Modellen med de beste resultatene er en 3D-versjon av ResNet-50, og oppnådde en gjennomsnittlig feilverdi på 17.40mm på testdataen.

Acknowledgements

I would like to offer my gratitude and appreciation to my supervisor and co-supervisor, Associate Professor Gabriel Hanssen Kiss and Professor Frank Lindseth, respectively. They have provided excellent guidance and displayed much patience during the process of writing my thesis. Also, much thanks to PhD student Anders Austlid Taskén for feedback, guidance and assistance. Thank you all for the time you have invested in helping me complete this work. I have learned a lot during this process, and I am thankful to have been able to work with such an exciting and meaningful project.

Finally, I would like to thank my wife, Linda Fosli Obrestad, for your patience and continued love and support during the writing process and my time at NTNU.

Contents

Abstract	iii
Sammendrag	v
Acknowledgements	vii
Contents	ix
Figures	xiii
Tables	xv
Acronyms	xvii
1 Introduction	1
1.1 Background	1
1.2 Goal and Research Questions	2
1.3 Structure of the Thesis	3
2 Theoretical Background	5
2.1 The Human Heart	5
2.1.1 The Heart's Anatomy	5
2.1.2 The Cardiac Cycle	6
2.1.3 The Aortic Valve	7
2.2 Ultrasound Imaging	8
2.2.1 Ultrasound Basics	8
2.2.2 Echocardiography	9
2.2.3 Cardiac Views	10
2.3 Artificial Neural Networks	11
2.3.1 Artificial Neurons	11
2.3.2 Feed-Forward Networks	12
2.3.3 Activation Functions	13
2.3.4 Training Neural Networks	13
2.3.5 Convolutional Neural Networks	18
2.3.6 Residual Connections	21
2.3.7 Transformer Architectures	22
2.4 Related Work	26
2.4.1 Landmark Localisation	26
2.4.2 Segmentation	27
3 Method	29
3.1 Data Set	29
3.1.1 Annotation	30

3.1.2	Train-, Validation- and Test Split	31
3.1.3	Pre-Processing	31
3.1.4	Post-Processing	31
3.1.5	Data Augmentation	32
3.2	Aortic Valve Localisation	32
3.2.1	Baseline	32
3.2.2	CNN Approach	33
3.2.3	Transformer Approach	33
3.2.4	Hybrid Approach	35
3.3	Experimental Setup	36
3.3.1	Hardware and Software	37
3.3.2	Input Size	37
3.3.3	Loss Function	37
3.3.4	Optimiser	37
3.3.5	Learning Rate Optimisation	38
3.3.6	Pre-Training	38
3.3.7	Data Set Configuration	39
3.3.8	Data Augmentation	39
3.3.9	Evaluation	39
4	Results	41
4.1	Comparison of Loss Functions	41
4.2	Effect of Pre-Training on Transformers	42
4.2.1	Training Performance	42
4.2.2	Network Performance	42
4.3	Learning Rate and Scheduling	44
4.4	Data Set Configuration	45
4.5	Data Augmentation	46
4.6	Inference Time	47
4.7	Final Localisation of the Aortic Valve	47
4.8	Visualisations	48
5	Discussion	51
5.1	Data Set and Pre-Processing	51
5.2	Annotations	51
5.3	Network Architecture	52
5.4	Parameter Search and Experiments	53
5.4.1	Loss Function	53
5.4.2	Pre-Training	53
5.4.3	Learning Rate	54
5.4.4	Data Set Configuration	54
5.4.5	Data Augmentation	54
5.5	Insights from the Training Process	55
5.6	Evaluating Final Performance	55
5.6.1	Inference Time	56
5.6.2	Error distribution of test set	56

5.7	Limitations	58
6	Conclusion and Future Work	59
6.1	Conclusion	59
6.2	Future Work	60
	Bibliography	61
A	Additional Material	69

Figures

2.1	Anatomy of the human heart	5
2.2	The Wiggers Diagram	6
2.3	The Aortic Valve	7
2.4	Comparison of 2D and 3D TEE images.	9
2.5	Illustration of Transesophageal echocardiography	10
2.6	Cardiac Views	10
2.7	Artificial and biological neurons	12
2.8	An artificial Feed-Forward neural network	12
2.9	Gradient Descent	15
2.10	The convolution operation	20
2.11	The max-pooling operation	21
2.12	A Residual Block	22
2.13	The Transformer	23
2.14	The Vision Transformer	25
2.15	Architecture for 3D CTA landmark localisation	27
2.16	U-Net architecture	28
3.1	Example of annotated data	30
3.2	Pipeline overview	32
3.3	3D ResNet-50 architecture	34
3.4	The Video Swin Transformer	35
3.5	3D window based multi-head self-attention visualised	35
3.6	The nnFormer	36
4.1	Training metrics of nnFormer	42
4.2	Training metrics of swin transformer	43
4.3	Learning rate scheduling on ResNet	44
4.4	Example of a good prediction	49
4.5	Visualisation of the worst prediction	50
5.1	Final scores single frame data set	57
5.2	Final scores on full test set	57
A.1	Visualisation of the best prediction	70

Tables

3.1	Data set split into train-, validation- and test data sets.	31
3.2	HeartNet architecture for inputs sized 128x128x128.	33
3.3	Data set size with- and without all frames.	39
4.1	Results of training runs comparing loss functions.	41
4.2	Pre-training impact on transformers	43
4.3	Cosine Decay Learning Rate Scheduler parameters	45
4.4	Network performance for different data set configurations	45
4.5	Network performance with data augmentation when trained on single frame- or multi frame data set, and evaluated on both data sets.	46
4.6	Inference times	47
4.7	The final results after training each model with its best configura- tion for 100 epochs.	47
4.8	Best and worst final results	48
5.1	Model parameter count	53

Acronyms

- 2D** two-dimensional. xiii, 2, 8, 9, 19–21, 26, 27, 30, 55, 59
- 3D** three-dimensional. iii, v, xiii, 1–3, 8, 9, 19, 26–30, 32, 34–36, 38, 47, 48, 52, 56, 58–60
- AdamW** Adam with Decoupled Weight Decay Regularization. 37, 38
- AI** Artificial Intelligence. 11
- ANN** artificial neural network. 2, 3, 11–14, 16–19, 21, 22, 31, 37, 47, 51, 58
- CNN** convolutional neural network. iii, v, 2, 3, 18–22, 26, 27, 32, 33, 35, 36, 47, 55, 56, 58–60
- Conv3D** three-dimensional convolutional layer. 33
- FC** fully connected layer. 33
- FP** floating point. 31, 37, 55
- GELU** gaussian error linear unit. 25, 35, 36
- GPU** graphics processing unit. 58
- L-ReLU** leaky rectified linear unit. 20, 33
- LAX** Long Axis. 9, 10, 29
- MAE** mean absolute error. 14
- MaxPool3D** three-dimensional max-pooling layer. 33
- ME** Mid Esophageal. 9, 10, 29
- ML** Machine Learning. 11
- MLP** Multilayer perceptron. 25

MSA multi-head self-attention. 35

MSE mean squared error. 14, 37, 39, 41

PhD philosophiae doctor. vii

PRF pulse repetition frequency. 8

ReLU rectified linear unit. 13, 17, 20, 22, 25, 33

SGD Stochastic Gradient Descent. 15, 16, 38

TEE transesophageal echocardiography. iii, v, xiii, 1–3, 9, 10, 29, 39, 40, 47, 51, 59

TTE transthoracic echocardiography. 1, 9

W-MSA window based multi-head self-attention. xiii, 35, 36

Chapter 1

Introduction

1.1 Background

It is estimated by the World Health Organization that cardiovascular disease represents 32% of global deaths, making it the number one global cause of death [1]. By detecting such diseases early, they can often be treated, either through medicine or operative procedures. The diagnosis and treatment of cardiovascular diseases can be greatly assisted by ultrasound imaging of the heart, called echocardiography. Echocardiography is a widely used imaging modality because it does not utilise ionising radiation, making it harmless compared to other modalities.

Advances in ultrasound imaging now enable cardiologists to acquire three-dimensional (3D) ultrasound images of the heart, allowing visualisation of anatomical structures in greater detail. TEE is a commonly used method to acquire 3D images of the heart [2]. It is acquired by inserting an ultrasound probe into the esophagus, resulting in more detailed images than transthoracic echocardiography (TTE), where the probe is placed on the patient's chest.

Because of the complex nature of cardiac operative procedures, such as valve repair or cardiac interventions, hybrid operating theatres are becoming more common. A hybrid operating theatre involves using intraoperative medical imaging to guide the surgeon during the procedure [3]. This enables the procedures to be minimally-invasive, as opposed to open-heart surgery. During such a procedure, it is important to obtain the instruments' position in relation to the anatomical structures of the heart. While instruments are best visualised using X-ray or CT imaging, finer anatomical details are more often visualised using ultrasound. This is partly due to the difference in the field of view and quality of the two different imaging modalities. However, the main reason is that ultrasound imaging can be presented to the surgeon in real-time and is harmless to the patient. To alleviate interventionalists from the time and energy required to align images manually according to instrument position, it would be helpful to devise a method capable of doing automatic alignment. To do such alignment, one would need to acquire the position and pose of the ultrasound probe from X-ray images or by using the positions of anatomical landmarks, such as the aortic valve, from the echocardi-

ography acquisitions as reference. Anatomical landmarks are also crucial in the process of generating standard TEE views. This thesis aims to propose methods for the latter detection task, namely automated detection of the aortic valve from 3D TEE volumes.

For the last decades, the field of machine learning has experienced breakthroughs in many tasks. The field of computer vision is relevant to this thesis because it tackles challenges related to image processing. The invention of convolutional neural networks (CNN), marked a milestone in computer vision performance. These are a form of artificial neural network (ANN) that mimics the way the visual cortex processes images. In 2015, He *et al.* [4] proposed a CNN that surpassed human performance in the ImageNet image classification task. An alternative ANN architecture named vision transformers, was introduced by Dosovitskiy *et al.* [5] in 2020, marking a new family of ANNs boasting of comparable performance to CNNs.

Motivated by these recent improvements in machine learning for computer vision tasks, we propose to investigate whether such methods could be used to automate aortic valve localisation.

1.2 Goal and Research Questions

The work presented in this thesis is centred around an overall goal, which is to develop a deep learning-based method to perform automated aortic valve localisation in 3D TEE volumes. As a continuation of the author's preliminary project, several research questions are proposed:

Research Question 1

Can one of the proposed deep learning methods achieve sufficient accuracy in detecting the aortic valve?

Research Question 2

Can the inclusion of attention-based architectures, like transformers, outperform traditional CNN-based methods as feature extractors?

Research Question 3

How do different training configurations and hyperparameters affect the performance of the proposed methods?

Research Question 4

It is possible to use the whole 3D volume as input directly, as opposed to 2D or patch-based methods, and how will this impact performance and hardware requirements?

To achieve these goals, three deep learning methods for aortic valve localisation are presented. The methods all represent different kinds of ANN architectures, namely a CNN, a transformer, and a hybrid architecture. These are compared to each other and with a functioning method from the author's project work as a baseline. This is done through several experiments, and testing on a 3D TEE data set. The localisation is done in the form of bounding box regression surrounding the aortic valve, where the proposed methods are used as backbones with a simple regression head.

1.3 Structure of the Thesis

Chapter 1

The introduction chapter presents the background and motivation for the work committed in the study. It also presents the overall goal and research questions of the thesis.

Chapter 2

The second chapter is a presentation of necessary theoretical knowledge for being able to understand the contents presented in the study. An introduction to cardiac anatomy, ultrasound and relevant machine learning theory is provided. The chapter ends with a presentation of previous work done within related fields.

Chapter 3

This chapter presents relevant details about the actual work committed and the specifics of the experiments planned to answer the proposed research questions. A detailed explanation of the proposed methods for aortic valve localisation is also presented.

Chapter 4

The results of the study are collected, summarised and presented in a structured manner in this section. In addition, short explanations will be provided for each portion of the data.

Chapter 5

This chapter is a discussion of the whole thesis, including the methodology and experiments proposed, as well as an analysis of the acquired results from the previous section. The weaknesses and limitations of the study will also be discussed.

Chapter 6

The entirety of the study is summarised in this chapter. The obtained results and discussion is formed into a brief conclusion presenting the most important findings of the study. Suggestions for future research are also presented.

Chapter 2

Theoretical Background

2.1 The Human Heart

The human heart is a muscular organ located in the mediastinum, I.e. the centre of the thoracic cavity. It is surrounded by the lungs to the sides, the sternum in front and the spine at the back. It is cone-shaped and positioned with its apex downwards, angled slightly to the left. The heart functions as the pump of the circulatory system and is responsible for blood flow throughout the body. Blood flow is essential to keep tissue and cells alive.

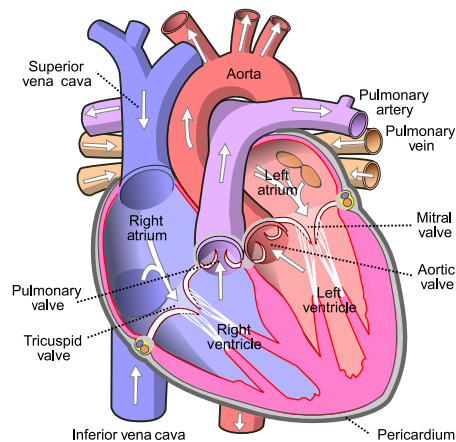


Figure 2.1: The figure shows the main components of the human heart [6].

2.1.1 The Heart's Anatomy

The heart is divided into two parts by the septum, where each part receives blood from the connected vein and pumps it out through its corresponding artery. The left part transports oxygen-rich blood from the pulmonary veins out through the aorta to the whole body. The right part experiences an influx of blood from the

venae cavae, the cardiac veins and the coronary sinus before pumping it to the lungs through the pulmonary artery for reoxygenation.

Each half of the heart is separated by a valve into two chambers called the atrium and the ventricle. Together, this partitions the heart into a total of four chambers, as shown in Figure 2.1. Blood flow is unidirectional from the atria to the ventricles. In the left heart, this is facilitated by the mitral valve, and in the right heart, it is ensured by the tricuspid valve. The valves are opened or closed according to the difference in blood pressure across the valve.

2.1.2 The Cardiac Cycle

The continuous process of pumping blood around the circulatory system is facilitated through what is known as the cardiac cycle. The cardiac cycle can be described as the process the heart undergoes from the start of one heartbeat to the start of the next and is a series of pressure changes within different compartments of the heart [7].

The cycle consists of two alternating phases, namely the diastole, where the heart relaxes and fills with blood, and the other named the systole, in which the heart contracts and pumps blood. The process is controlled by electric signals originating from special pacemaker cells in the sinoatrial node located in the heart wall of the right atrium. These signals propagate through the atria stimulating contraction of the heart muscle, i.e. the myocardium, in the atria. The signals then travel to the atrioventricular node, located in the septum in between the atria and the ventricles, where they, after a brief delay, propagate through the myocardium of the lower heart. This causes the ventricles to contract.

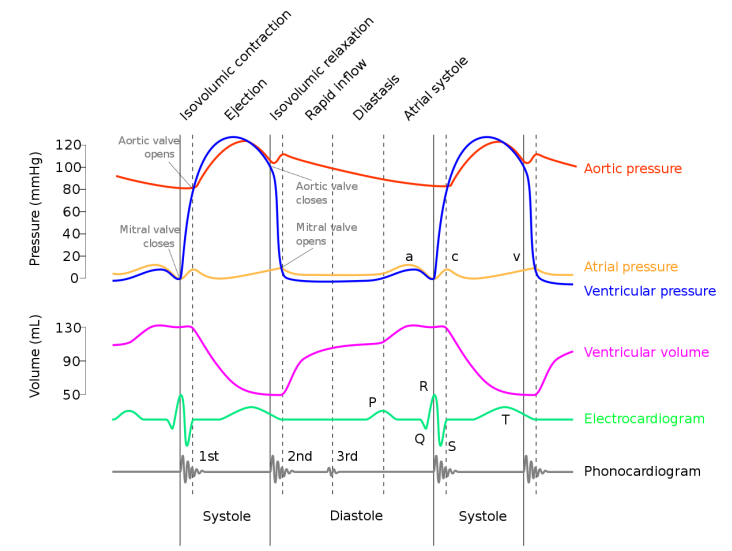


Figure 2.2: The Wiggers Diagram, showing the variation of several cardiac parameters throughout the cardiac cycle [8].

Directly after contraction and subsequent emptying of blood, the myocardium relaxes, allowing intake of new blood. Because of the delay at the atrioventricular node, the atria and the ventricles undergo systole and diastole alternately with a slight overlap. This means that near the end of the ventricular diastole, the atria begin to contract. At the end of this contraction, the ventricular systole is initiated.

The relations between different parameters of the cardiac cycle, such as pressure, volume and valve opening, can be described in detail through a Wiggers diagram, as shown in Figure 2.2.

2.1.3 The Aortic Valve

The aortic valve connects the left ventricle to the aorta. It consists of several anatomical structures, which can be seen in Figure 2.3. The aortic annulus is a circular crown-shaped structure formed by the attachment of three semilunar leaflets to the tissue connecting the left ventricular outflow tract to the aorta[9]. The three leaflets function as the valve mechanism and cover what is known as the three sinuses of Valsalva. These are widenings in the wall of the ascending aorta, i.e. the first part of the aorta. The leaflets are named the right-, left- and non-coronary cusps, after the respective sinus that they cover. The term aortic root is also often used to describe the part of the aortic valve from the top of the left outflow tract to the junction between the sinuses of Valsalva with the ascending aorta[10].

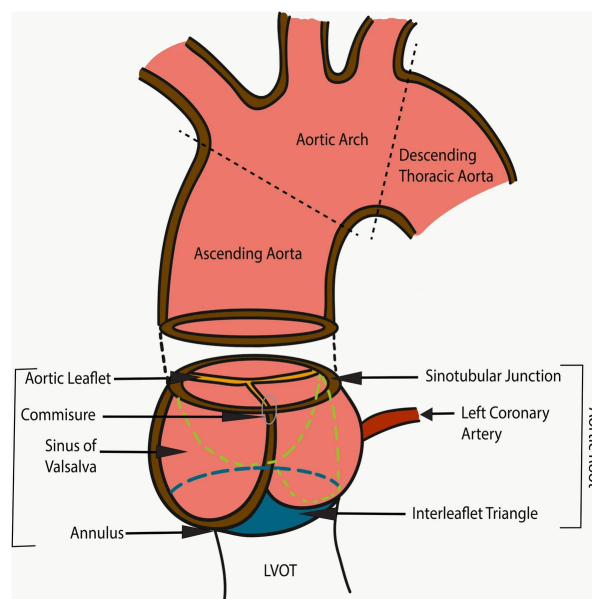


Figure 2.3: The aortic valve and its anatomical structures. [11]

The aortic valve opens after ventricular contraction, which leads to increased pressure in the left ventricle compared to the aorta. This enables oxygen-rich blood flow through the aorta out to all body systems. After the ventricular pressure falls below aortic pressure, the valve closes to stop the backflow of blood. This marks

the beginning of the diastole. As the valve controls blood flow to the primary circulatory system, it is considered to be one of the most important parts of the heart. Incidentally, it is also located in the centre of the heart, making it an anatomical landmark to navigate from when studying medical images of the heart.

2.2 Ultrasound Imaging

For over half a century, ultrasound imaging has been a major revolution within several fields, e.g. manufacturing, security and medicine. Especially within medicine, ultrasound examinations have proven to be both non-invasive and harmless to patients, compared to other medical imaging modalities, and are therefore widely used [12]. Modern ultrasound technology has the capability to produce images both in 2D and 3D and can show anatomical structures in great detail.

2.2.1 Ultrasound Basics

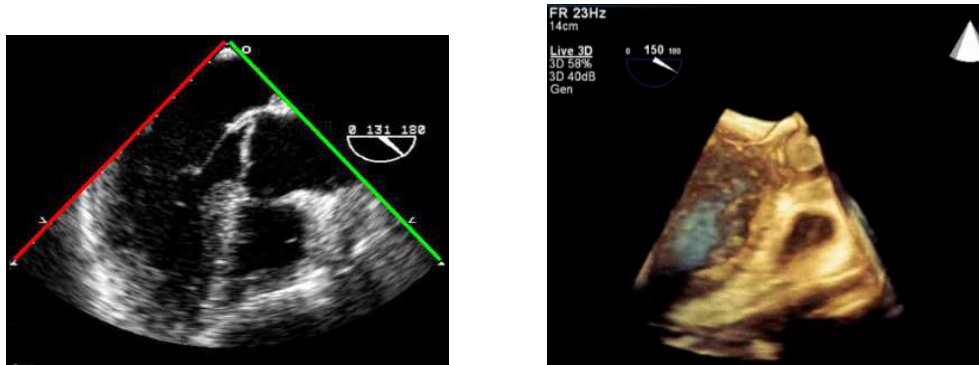
The term ultrasound describes high-frequency sound waves. These sound waves are in a frequency range above what is audible to the human ear. When used for medical imaging, the sound waves are typically emitted at a frequency in the range of 1 to 15 MHz. The sound waves are produced by a probe containing piezoelectric crystals, which vibrate when receiving electrical current. The probe both emits and receives sound waves in pulses. The frequency at which this happens is called pulse repetition frequency (PRF).

As the emitted sound waves pass through the body, they are partly reflected. The strength of the reflected signals is determined by the acoustic properties of the type of tissue that is encountered. In the time between emitted pulses, the probe receives incoming reflected signals. It is, therefore, important that the PRF is low enough to allow the emitted sound wave to reach the target tissue and return to the probe before the next pulse. The normal range for medical imaging lies between 1 and 10 kHz. Received signals are converted into electrical current. The strength of the received pulse is proportional to the amount of current, while the elapsed time between emission and reception determines the distance to the reflective tissue. In the most common form of ultrasound imaging, known as B-mode, a grayscale ultrasound image is constructed by translating these properties into pixel intensity and vertical axis position, respectively. This thesis will focus solely on B-mode imaging.

To produce a 2D image, the emission of ultrasound pulses is divided into several scan lines across the probe surface. The scan lines are constructed by activating all elements in the probe with varying delays. These are used to construct the horizontal axis of the resulting image. With pixel intensities across both the horizontal and vertical axes, one can construct a grayscale 2D image.

Recent advances in ultrasound technology have enabled the acquisition of 3D images. This is achieved using the same principles as in B-mode imaging. However, the ultrasound emitters are arranged in a 2D matrix array. Using a signal

processing technique known as beamforming, one can sample data points spatially in a pyramid-shaped manner to create a 3D volume. The difference between a 2D image and a 3D image can be observed in Figure 2.4.



(a) Example of 2D-TEE ME LAX image.

(b) Example of 3D-TEE ME LAX image.

Figure 2.4: Side by side comparison of a two-dimensional- and three-dimensional-transesophageal echocardiography image showing the Mid Esophageal Long Axis view [13]. The aortic valve is clearly visible in the centre of both acquisitions.

2.2.2 Echocardiography

Echocardiography is the application of ultrasound imaging to examine the heart. Such imaging can be used to gather information about cardiac function. The two most commonly used forms of echocardiography are called transthoracic echocardiography and transesophageal echocardiography.

TTE is the most common method, as it is both non-invasive and quicker to perform than TEE. It is performed by placing the ultrasound probe on the exterior of the patient's thorax or abdomen, sending pulses inwards. TEE however, is performed by inserting a probe into the patient's esophagus and sending pulses outwards as shown in Figure 2.5. The probe's increased proximity to the heart and great vessels usually means that TEE provides images of higher quality and resolution compared to TTE. This is useful for examining the finer anatomical structures of the heart, i.e. the aortic valve or the mitral valve.

Because of the invasive nature of TEE, the patient is often placed under local anaesthesia during the examination. This procedure is therefore often only used as a follow-up after an initial TTE examination if a greater level of detail is required. During cardiac surgery, where the patient is under general anaesthesia, the probe can be left in the patient's esophagus, and in this situation TEE is also often used to provide live images to the surgeon.

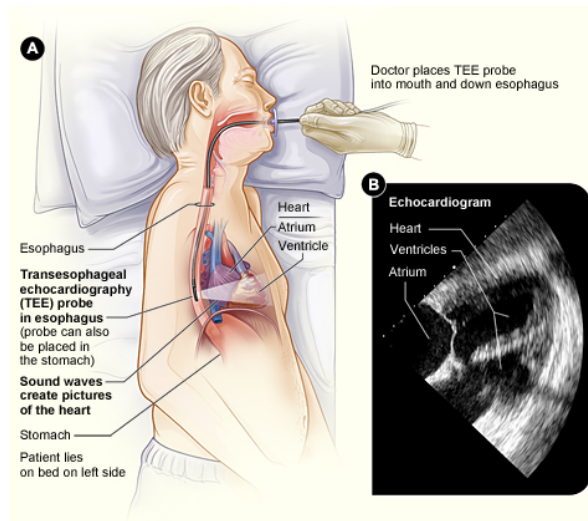


Figure 2.5: (A) Illustration showing the procedure of transesophageal echocardiography. (B) The resulting grayscale image. [14]

2.2.3 Cardiac Views

In the practice of transesophageal echocardiography, there is the need to standardise the description of cardiac views. Such standards are provided by the American Society of Echocardiography in the form of a publication describing 28 views [15]. The views are described in the form of cardiac cross-sections from different angles and probe positions in the esophagus. Examples of four such view definitions are shown in Figure 2.6. Figure 2.4 also shows the Mid Esophageal (ME) Long Axis (LAX) view, which is one of the most common views for surveying the aortic valve.

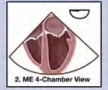
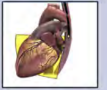




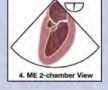





			Transducer Angle: ~ 0 - 10° Level: Mid-esophageal Maneuver (from prior image): Advance ± Retroflex	Left atrium/Right atrium IAS Left ventricle/Right ventricle/IVS Mitral valve (A ₂ A ₁ -P ₂ P ₁) Tricuspid valve
			Transducer Angle: ~ 50 - 70° Level: Mid-esophageal Maneuver (from prior image): NA	Left atrium Coronary Sinus Left ventricle Mitral Valve (P ₂ -A ₂ A ₁ -P ₁) Papillary muscles Chordae tendinae
			Transducer Angle: ~ 80 - 100° Level: Mid-esophageal Maneuver (from prior image): NA	Left atrium Coronary sinus Left atrial appendage Left ventricle Mitral valve (P ₂ -A ₂ A ₁)
			Transducer Angle: ~ 120 - 140° Level: Mid-esophageal Maneuver (from prior image): NA	Left atrium Left ventricle LVOT RVOT Mitral valve (P ₂ -A ₂) Aortic valve Proximal ascending aorta

Figure 2.6: The American Society of Echocardiography’s presentation of four common cardiac views detailing probe position, angle and the anatomical structures visible from the specific view. [15]

2.3 Artificial Neural Networks

Artificial Intelligence (AI) is a broad field within computer science, focused on creating seemingly intelligent computer programs able to solve tasks usually associated with intelligent beings. Machine Learning (ML) is a sub-field of artificial intelligence focused on making computer programs that can learn to solve a specified task and improve themselves without being explicitly programmed. In the field of machine learning, there are several different methods for achieving this; however, this thesis will focus on what is known as supervised learning. In supervised learning, the objective of a machine learning method is to approximate an unknown function by looking at a vast number of function inputs and corresponding outputs. Later innovation in machine learning has largely been driven forward by research in artificial neural networks (ANNs). This approach is inspired by how the brain functions as a network of interconnected neurons. The first physical application of an ANN, invented in 1958, was a one-layer network called the "Mark 1 perceptron. It was built as a machine designed for image recognition [16]. The main building block of the perceptron was linear threshold units, also called perceptrons. They were first proposed by McCulloch and Pitts in 1943 and laid the foundations for the mathematical model of ANNs [17]. Today these units are known as artificial neurons.

2.3.1 Artificial Neurons

The artificial neuron has remained nearly unchanged since its origin and is the building block of many modern ANN architectures. These neurons are said to be a simplified mathematical model of the neurons in the human brain. Even though a biological neuron is substantially more advanced, the basic concepts are the same, as can be seen in Figure 2.7. The functionality of an artificial neuron is to accept a range of discrete input signals, which are outputs from other neurons. This is similar to the dendrites of a biological neuron. A weighted sum of these inputs is calculated and passed to a non-linear threshold function, also known as an activation function. The activation function determines if the neuron is to send an output signal or not. This can be likened to the biological neuron in which the soma acts as a summation and activation function, causing the outgoing axons to fire electrical signals if the electrical potential in the soma reaches a certain threshold.

Mathematically the artificial neuron can be defined as in Equation (2.1), where y_j denotes the output of the j -th neuron. b is a learnable bias term representing the threshold value of the neuron. x_i is the output from the i -th previous neuron with w_i denoting the learnable weight associated with x_i . Lastly $\varphi()$ represents the activation function.

$$y_j = \varphi \left(b + \sum_{i=1}^n w_{ij} x_i \right) \quad (2.1)$$

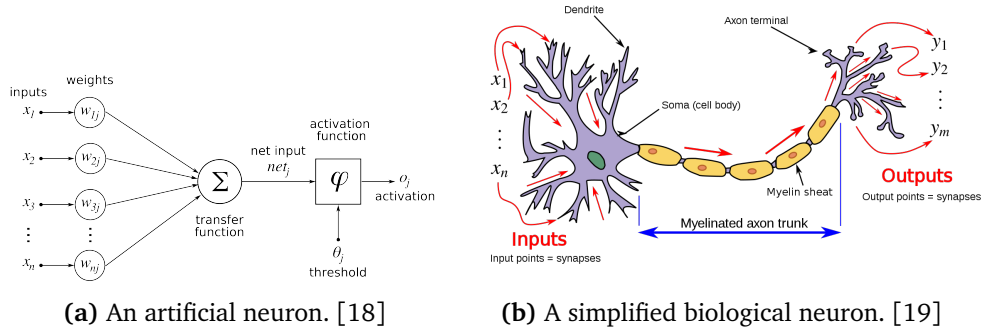


Figure 2.7: Comparison of an artificial and a biological neuron. Notice the similar structure.

2.3.2 Feed-Forward Networks

Artificial neurons can be stacked on top of each other, connected and arranged in layers as seen in Figure 2.8. In this configuration, all inputs are passed forward throughout the layers of nodes to the final outputs without cycles. This is the most basic kind of ANN and is known as feed-forward networks. The perceptron from 1958, was in fact, a one-layer feed-forward network. A layer in a feed-forward network is often called a fully-connected layer since the output of each neuron is connected to the input of every neuron in the next layer.

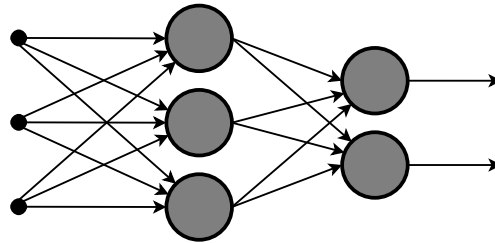


Figure 2.8: Visualization of an artificial Feed-Forward neural network. [20]

Each layer of neurons in a feed-forward network can be formulated as a mathematical function. Given x as input, and θ as the set of learnable parameters for the layer, i.e. w and b , you have $y = f(x; \theta)$. With the same notation, one can define a neural network as a combination of several such functions, as shown in Equation (2.2).

$$\hat{y} = f(x; \theta) = f_3(f_2(f_1(x; \theta))) \tag{2.2}$$

In recent years, as computing power has been made more available, the amount of layers in a feed-forward network has been able to increase considerably, leading to ANNs being able to model more complex functions. The field of study associated with many-layered ANNs is known as deep learning.

2.3.3 Activation Functions

Since the weighted sum of inputs is a linear combination, the addition of a non-linear activation function is essential for the network's ability to approximate non-linear functions. A neural network with neurons lacking activation functions would essentially be a linear combination of linear combinations, making it only able to approximate linear functions. The usage of such activation functions combined with many layers enables an ANN to learn highly complex functions. A plethora of activation functions exist, from simpler threshold functions like the Heaviside unit step function, to sigmoid functions like the hyperbolic tangent. However, for deep learning, a common choice is the rectified linear unit (ReLU)[21] or variations thereof.

The ReLU function is classified as a ramp function, and is defined in Equation (2.3). It acts as a threshold function, only outputting values for inputs greater than zero. This somewhat mimics the functionality of the axons of the brain and is shown to improve performance in the context of computer vision [22]. ReLU is often considered the default choice of activation function in the deep learning community, both because of its simplicity and effectiveness [23]. However, it is not without problems. As a function, it is not differentiable at zero. It is also observed that in deep neural networks utilising ReLU, some neurons end up in a state where they are never activated because of small gradients. There have been proposed several variations of ReLU to counter this, among them the functions mentioned in the headings about ReLU and GELU, found in Section 2.3.5 and Section 2.3.7 respectively.

$$f(x) = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{otherwise} \end{cases} \quad (2.3)$$

2.3.4 Training Neural Networks

To be able to solve a given task, an ANN first needs to be trained. Tasks can usually be divided into two categories, namely classification and regression. In a classification task, the network tries to predict what class the input belongs to, and the output is usually a vector with probabilities for each defined class. For regression tasks, one trains the network to map an input variable x to a continuous output value y . The end goal of training is to find the ideal parameters θ that minimise a measure of the difference between the network output $\hat{y} = f(x; \theta)$ and the ground truth y . The training process includes several concepts presented in the following headings.

Loss Function

There are several ways to define the network error, suitable for different tasks. Expressing the network error as a function $\mathcal{L}(y, \hat{y})$ is useful, as it defines the problem in terms of a function that one wants to minimise. Such a function is called

a loss function. For regression tasks it is common to use either mean absolute error (MAE) or mean squared error (MSE) defined in Equation (2.4) and Equation (2.5). These are sometimes known as L1- and L2-loss, respectively. L2-loss is often used because of its speed of converging. However, L1-loss is considered more stable. It is imperative that the chosen loss function is differentiable. The reason for this is described in the header about backpropagation.

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum |y_n - \hat{y}_n| \quad (2.4)$$

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum (y_n - \hat{y}_n)^2 \quad (2.5)$$

Data

To learn an accurate representation of a function, an ANN is dependent on large amounts of data. For learning a mapping $f : X \mapsto Y$, the network is shown many examples $\{x_i, y_i\} \subset \{X, Y\}$, and tries to approximate this.

When training, it is customary to divide the data into three sets for specific purposes, namely training, validation and testing. The training set is usually the largest of the three and is the data the network uses to learn from. One iteration of the training set is called an epoch. Validation is usually run at the end of each epoch. This is a process where the network is fed the data of the validation set, but no training is done. This is simply to evaluate the performance of the ANN during training. Finally, when the network is finished training, the network is evaluated on the test set using a metric of choice. This is considered the final benchmark of network performance, since the test data has not previously been seen by the network, and should be a real indicator of performance on data in the real world.

Under- and overfitting

Even though maximum accuracy is desired, because of the use of a specified training set, one runs the risk of training the network to approximate only the examples of the training set. This causes bad performance on real-life cases and is generally not desired. Overfitting can generally be discovered if the training error is getting lower each epoch, while the validation error is rising.

Underfitting is the opposite case. This is when the network is unable to approximate even the samples of the training set. This means that the model is unable to capture the relationship between input and output and might mean that the specific network architecture is unsuited for the task.

Gradient Descent

The process of tuning the learnable parameters θ of an ANN such that the loss function is minimised, can be likened to finding the global minima of a non-convex hyperplane with θ dimensions, as shown in Figure 2.9. This is a problem which

has no feasibly computable analytical solution and is therefore solved as an optimisation problem in an iterative manner. In most recent machine learning research, variations of an algorithm known as gradient descent are used.

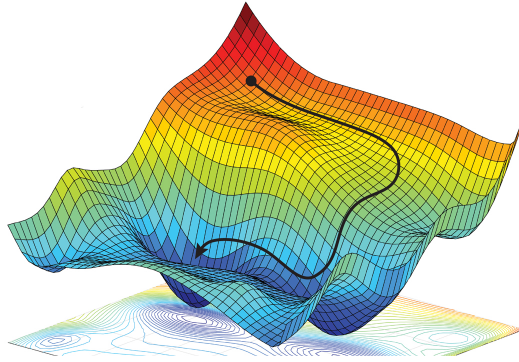


Figure 2.9: Illustration of gradient descent down the slope of a non-convex function. [24]

The method was first described by Cauchy in 1847 [25] and is an iterative optimisation algorithm used to minimise a differentiable function. The principle of gradient descent is to calculate the gradient of the loss function with respect to the learnable parameters as defined by Equation (2.6).

$$\nabla_{\theta} \mathcal{L} = \left[\frac{\partial \mathcal{L}}{\partial w}, \frac{\partial \mathcal{L}}{\partial b} \right]^T \quad (2.6)$$

The gradient shows in which direction the loss function \mathcal{L} decreases the fastest, namely the direction of the negative gradient $-\nabla_{\theta} \mathcal{L}$. Adjustments of the learnable parameters θ can then be made following the update rule, as defined in Equation (2.7), to shift the loss one step closer to a local minimum. The parameter η represents the step size of the algorithm, more commonly known as the learning rate in the context of machine learning.

$$\theta = \theta - \eta \nabla_{\theta} \mathcal{L} \quad (2.7)$$

This is the basic form of gradient descent called batch gradient descent. However, it is computationally expensive to calculate the gradient with respect to the parameters for the entire data set. It is, therefore, more common to use other optimisers based on variations of this principle. These often contain slight modifications to the update rule.

Optimisers

The most commonly used optimiser in deep learning is a variation of batch gradient descent called Stochastic Gradient Descent (SGD). Instead of computing the

true gradient of the function by utilising the entire data set, the gradient is calculated from a random subset of the data. This resembles a stochastic approximation of the batch gradient descent method. This gives an update rule as defined in Equation (2.8), where $f()$ represents the ANN output and $x_i, y_i \in \{X, Y\}$ are network inputs and labels respectively.

$$\theta = \theta - \eta \nabla_{\theta} \mathcal{L}(f(x_i), y_i) \quad (2.8)$$

Because SGD always follows the direction of the negative gradient with a fixed step size, it runs the risk of getting stuck in a local minimum instead of the global one. To counteract this, a principle called momentum is introduced. This results in the common optimisation algorithm Stochastic Gradient Descent with momentum, introduced by Rumelhart *et al.* [26]. The modified update rule with momentum is defined in Equation (2.9), where α denotes the momentum decay factor. The main principle here is to calculate the weight update as a linear combination of the gradient and previous parameter update, in fact giving the parameters a form of acceleration. This can be thought of in terms of a rock falling down a hill carrying a certain momentum. If the momentum is large enough, the rock can roll past small local mountain shelves, potentially avoiding getting stuck. The method has proven very successful and is widely used among researchers.

$$\theta = \theta - \eta \nabla_{\theta} \mathcal{L}(f(x_i), y_i) + \alpha \Delta \theta \quad (2.9)$$

Even though SGD with momentum is considered one of the most popular and stable optimisers, convergence can be slow. An adaption to counter this is to use adaptive learning rates for individual parameters. A frequently used method that implements this is the optimisation method by Kingma and Ba [27], known as Adam. In Adam, the learning rates of each parameter are updated based on running averages of the first and the second moments of the gradients. If t denotes the time-step of the training epoch, β_1, β_2 are moment decay factors, and m and v are the first and second moments respectively, one can define the update rule as shown in Equation (2.10). ϵ is added as a small parameter to hinder division by zero. The gradient of the loss function is denoted as $\nabla_{\theta} \mathcal{L}$ for simplicity, but is identical to the gradient as defined in Equation (2.8).

$$\begin{aligned} m_{\theta}^{t+1} &= \beta_1 m_{\theta}^{(t)} + (1 - \beta_1) \nabla_{\theta} \mathcal{L}^{(t)} \\ v_{\theta}^{t+1} &= \beta_2 v_{\theta}^{(t)} + (1 - \beta_2) (\nabla_{\theta} \mathcal{L}^{(t)})^2 \\ \hat{m}_{\theta} &= \frac{m_{\theta}^{(t+1)}}{1 - \beta_1^t} \\ \hat{v}_{\theta} &= \frac{v_{\theta}^{(t+1)}}{1 - \beta_2^t} \\ \theta^{(t+1)} &= \theta^{(t)} - \eta \frac{\hat{m}_{\theta}}{\sqrt{\hat{v}_{\theta} + \epsilon}} \end{aligned} \quad (2.10)$$

Backpropagation

When using an artificial neural network to transform an input x to an output \hat{y} , the input is propagated through the layers of the network. This is known as forward propagation. Backpropagation is essentially the opposite process and is a mechanism that is used during gradient descent to calculate the gradients of the loss with respect to the learnable parameters of the ANN throughout all layers. Because of the derivation involved, it is essential that all functions used in the ANN are differentiable. The term was first announced by Rumelhart *et al.* [26] in the context of machine learning and is used in most modern machine learning applications.

The goal of backpropagation is to calculate the gradient of the loss function with respect to any learnable parameter as defined in Equation (2.6). The gradient calculation starts at the last layer of the ANN, calculates intermediate gradients for each layer, and updates the layer parameters based on this intermediate gradient. First the derivative of the loss function with respect to the activations of the last, i.e. N th, layer $\nabla_{a^N} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial a^N} = \mathcal{L}'(a^N)$ is calculated. By utilising the chain rule of calculus, this term can in term be used to calculate $\delta^N = \frac{\partial \mathcal{L}}{\partial z^N} = \frac{\partial \mathcal{L}}{\partial a^N} \circ \sigma'(z^N)$, where σ is the activation function, and z^N is the input to the activation function. Applying the chain rule and the product rule, we can get the same term expression but generalised for any layer, as seen in Equation (2.11). This term acts as an intermediate derivative and is used to calculate the derivative with respect to w and b . These expressions are shown in Equation (2.12) and Equation (2.13) respectively.

$$\delta^{n-1} = \frac{\partial \mathcal{L}}{\partial z^{n-1}} = ([w^{(n)}]^T \frac{\partial \mathcal{L}}{\partial z^n}) \circ \sigma^{n-1\prime}(z^{n-1}) \quad (2.11)$$

$$\frac{\partial \mathcal{L}}{\partial w^n} = \frac{\partial \mathcal{L}}{\partial z^n} * [a^{(n-1)}]^T = a^{n-1} \delta^n \quad (2.12)$$

$$\frac{\partial \mathcal{L}}{\partial b^n} = \frac{\partial \mathcal{L}}{\partial z^n} = \delta^n \quad (2.13)$$

With the derived expressions for $\nabla_{\theta} \mathcal{L}$, one can propagate this information up through all layers of the ANN and update the parameters according to the update rule presented in Equation (2.7).

Weight Initialisation

Weight initialisation is a crucial step of the training process. This step happens when the ANN is first created and specifies what values to give the weights before training. It turns out that random initialisation is not the most optimal way to initialise network parameters. Several methods exist for this and what method is most suitable to use varies depending on what type of layer the weights are for and the activation functions used. For example, for fully-connected layers with ReLU

activation, it is often common to utilise He-initialisation, also known as Kaiming-initialisation as seen in Equation (2.14) [4]. It samples weight values from a zero-centred Gaussian with a standard deviation of $\sqrt{\frac{2}{n_l}}$, where n_l can be either the number of neurons coming into the layer or coming out from the layer. The bias values are all initialised to zero.

$$w_l \sim \mathcal{N}\left(0, \frac{2}{n_l}\right) \quad (2.14)$$

Regularisation

There are several methods to improve training performance and accuracy. To improve model generalisation and hinder over-fitting, one often uses a form of regularisation, also known as weight decay. The most common forms of regularisation are known as L1 and L2 regularisation, and use the same calculation as the L1- and L2-loss. However, the calculation is done over the magnitude of the network's weights instead of the loss function. L1 regularisation calculates the sum of the absolute value of all the weights, while L2 regularisation calculates the sum of the squared value of all the weights. Then the calculated score is added to the total loss during training, effectively punishing the network for having large weights. The goal of this is to counteract overfitting, as networks with huge weights often show signs of overfitting and, therefore, low generalisation.

Batch-Normalisation

During training, a neural network encounters what is known as internal co-variate shift [28]. This effect is caused by the change of network parameters causing overall changes in the distributions of network activations. When this phenomenon happens, each layer will, in turn, have to compensate for the change in distribution, resulting in low training performance.

To counter internal co-variate shift, a method known as batch normalisation is used. The normalisation scales each layer input to have a mean of zero and variance of one as seen in Equation (2.15). Each layer gets learnable parameters γ, β that control the level of shift and scale of the normalised data. The mean and standard deviation are calculated before normalisation for each dimension over each mini-batch. This method or variants of it are used in most modern artificial neural networks.

$$y = \frac{x - E[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta \quad (2.15)$$

2.3.5 Convolutional Neural Networks

Convolutional neural networks (CNNs) marked a revolution in computer vision research. They are a family of feed-forward networks originally inspired by the

visual cortex. CNNs saw their first popular application in a computer vision context by Lecun *et al.* [29] in 1989, in the form of a network performing optical character recognition in documents. However, they gained increased interest after the CNN AlexNet beat every other image classification method in the ImageNet challenge by Google [30].

The introduction of CNNs was motivated by the increased complexity of working with 2D data like images. If a normal feed-forward network were to work on a square image of size 256 with RGB colour, the ANN would have to keep track of $256 \times 256 \times 3 = 196\,608$ weights per neuron in the first layer alone. This rapid increase in computational complexity means that fully connected layers are not suitable for working with such data. CNNs solve the complexity issue by interacting with the pixels in a sparse manner and sharing weights such that the next layers only see a subset of the input pixels. Convolutional neural networks normally consists of four main parts:

1. Convolutional Layers.
2. ReLU layers.
3. Pooling Layers.
4. Fully-Connected Layers.

Convolution

Instead of the normal weighted summation of fully-connected layers, convolutional layers utilise a set of learnable filters known as kernels. These kernels often span a smaller size than the input itself. During forward propagation, the kernel is slid across the input image as a window while calculating the dot product of kernel weights and input values within the window. This is known as the convolution operation; however, mathematically speaking, it is actually implemented as the cross-correlation operation. The difference between the two is minor and only involves flipping the kernel. For simplicity and historical reasons, the operation will still be referred to as the convolution operation in this thesis.

Convolution over a 2D input X and 2D kernel is defined in Equation (2.16). In this equation, the kernel size is given by $2d + 1 \times 2c + 1$, and k, j denotes the row and column index of the convolution output Y . The output Y of a convolution is often called a feature map. The formula can be extended to a three-dimensional setting by adding another dimension, as shown in Equation (2.17).

$$Y(j, k) = (X * w)(j, k) = \sum_{\gamma=-c}^c \sum_{\delta=-d}^d X(j + \gamma, k + \delta) w(\gamma, \delta) \quad (2.16)$$

$$Y(i, j, k) = (X * w)(i, j, k) = \sum_{\beta=-b}^b \sum_{\gamma=-c}^c \sum_{\delta=-d}^d X(i + \beta, j + \gamma, k + \delta) w(\beta, \gamma, \delta) \quad (2.17)$$

The convolution operation starts in the upper left corner of X and applies the kernel to create the upper-left entry of Y . The kernel is shifted horizontally

according to a given step size called stride, and the process repeats for the entire row. After completing a row, the kernel is shifted one column down and returned to row start. The calculation of one feature map index is illustrated in Figure 2.10.

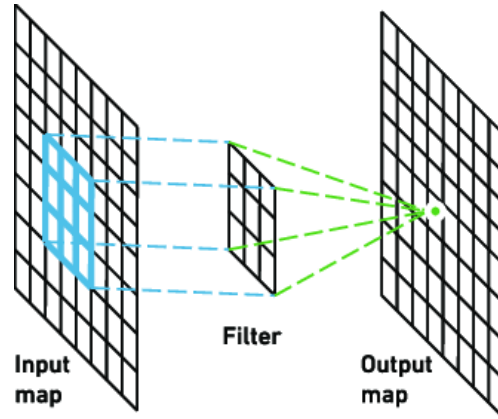


Figure 2.10: The application of the convolution operation on a 2D input to create a feature map [31].

The rectified linear unit (ReLU)

After a feature map has been created by the convolutional layer, it is common to utilise a layer with a non-linear activation function. The most utilised family of activation functions for CNNs are ReLU, as presented in Equation (2.3). However, a problem often encountered using the standard ReLU is something known as the dying neuron problem. This is because of the activation function's property that it only passes forward positive inputs. Since negative inputs lead to a zero-activation, it will also lead to a zero gradient, which can, in some cases, hinder backpropagation from changing the corresponding weight values. In some cases, this will lead to the neuron "dying" and being stuck in a state of never activating. To counter this, variations such as the leaky rectified linear unit (L-ReLU) is introduced [32]. The definition is shown in Equation (2.18)

$$f(x) = \begin{cases} x, & \text{if } x > 0, \\ 0.01x, & \text{otherwise} \end{cases} \quad (2.18)$$

Instead of outputting zero for negative inputs, L-ReLU will output the input scaled by a small factor of 0.01. This causes the activation to have a non-zero gradient in its entire domain. In effect, this counters the dying neuron problem. This property makes the L-ReLU a commonly used activation function for CNNs.

Pooling

Another central part of CNNs are pooling layers. They are usually added after a sequence of convolutional layers. Their purpose is to counter overfitting and re-

duce the number of activations. They function by extracting the most important information from a feature map, essentially downscaling the feature map. This is done by sliding a kernel across the input in the same manner as in a convolutional layer. However, for pooling layers, the kernel does not contain learnable weights. Instead, they perform a specified calculation on the inputs in the current position of the filter. The most common type of pooling layer, namely max-pooling, takes the most significant number in the area covered by the filter and passes it as output. However, there is also average-pooling, which calculates the average of the inputs in the filter region. It is common to use max-pooling after consecutive convolution layers, with a filter of size 2×2 and a stride value of 2, as shown in Figure 2.11.

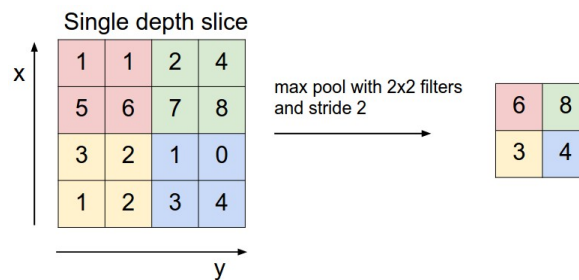


Figure 2.11: The application of the max-pooling operation on a 2D input to down-scale a feature map, and extract the most important information from it [33].

A layer of this configuration will discard 75% of the activations from the previous layer without removing too much of the encoded information from the feature map.

Fully-connected layer

It is common to put one or more fully-connected layers at the end of a CNN. This structure is often called the head of the CNN, producing the final output prediction. The feature map is flattened into a one-dimensional vector before being passed through the layer. The fully-connected layers may function as either a classifier head or a regression head. It can also be configured to perform object detection or segmentation.

2.3.6 Residual Connections

As computational power has gone through a massive increase in recent years, so has the depth of artificial neural networks resulting in a great increase in network accuracy. However, much deeper networks have proven hard to train. Increasingly deep networks result in what is known as the degradation problem[34]. When gradients pass through many layers of non-linear activation functions during backpropagation, they often approach zero in early layers. This excludes some weights from being updated. This can be partially solved through a normalised

initialisation procedure, but deep networks still have challenges with learning. It is reported that once a deep network converges, accuracy saturates and sharply decreases [35]. One method to counter this phenomenon is residual connections, also called skip-connections, introduced by He *et al.* [34].

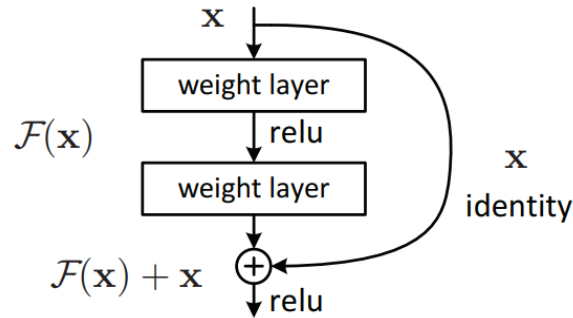


Figure 2.12: Illustration of a residual block [34].

A residual connection is, in essence, an identity mapping, which skips a couple of layers as a side-step in the network graph, and is then concatenated back to the main branch following the skipped layers. In He *et al.* [34], the original desired mapping is defined as $\mathcal{H}(x)$. The skipped layers is formulated as a mapping $\mathcal{F}(x) := \mathcal{H}(x) - x$. The original mapping can then be reformulated into $\mathcal{F}(x) + x$, as shown in Figure 2.12. Inserting such residual connections into a network with set intervals, helps gradient flow through the network to the earlier layers. This makes optimising deep networks much more efficient.

2.3.7 Transformer Architectures

Transformer architectures are ANNs that are created to learn context and track relationships within sequential data. It was first utilised with great success in the machine learning applications relating to language and translation [36]. However, the introduction of the vision transformer in 2020 by Dosovitskiy *et al.* [5] sparked great interest in transformers within computer vision. Recently transformers employed to computer vision tasks have shown to perform comparable to, or better than, CNNs given enough training data [37]. Most applications utilising transformers are trained on massive data sets.

Transformers are based on an encoder-decoder structure, meaning the network is divided into two parts, where the encoder creates an intermediate representation of the input sequence, which is then fed to the decoder that produces the output sequence. The transformer is mainly composed of a form of positional embedding, followed by stacks of self-attention layers and fully-connected layers with ReLU activations. An overview of the original transformer structure can be seen in Figure 2.13. The decoder takes both the output of the last encoder block and the output of the previous masked multi-attention layer.

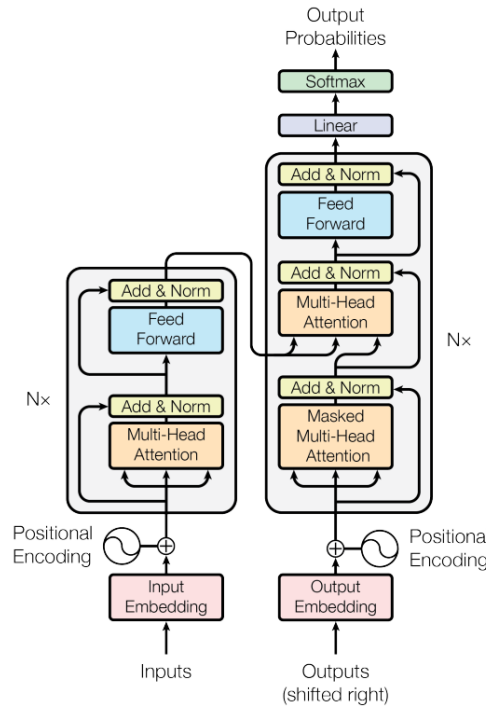


Figure 2.13: Overview of the original Transformer architecture [36].

Positional embedding

As mentioned earlier, the transformer accepts sequences as input. In the original transformer, the sequences represented sentences of words. These sequences have to be split up into a set of tokens, i.e. words, which are then embedded into vectors of dimension d using learned embeddings. The vectors are then injected with positional information to allow the model to utilise the order of the sequence. In the original transformer, this positional encoding is based on sine and cosine, using the formulas shown in Equation (2.19), where p represents the position and i the dimension. The positional embedding is of the same dimension d as the embedding, so that the two can be directly summed.

$$\begin{aligned} \text{Pos}(p, 2i) &= \sin\left(\frac{p}{10000^{2i/d}}\right) \\ \text{Pos}(p, 2i + 1) &= \cos\left(\frac{p}{10000^{2i/d}}\right) \end{aligned} \quad (2.19)$$

Self-Attention

The main building block of the transformer is based on a mechanism known as self-attention. Attention can be interpreted as a measure of similarity, or correlation, between the elements of the input sequence. The original transformer used dot-product attention, which functions by mapping a query Q , and key-value pairs

K, V to an output by a weighted summation. The weights are computed as a function of the correlation between the query with the corresponding key. The values Q, K, V are created by matrix multiplication between the input embedding X with three different weight matrices W^Q, W^K, W^V . The attention values are then computed using Equation (2.20). $\sqrt{d_k}$ is added as a scaling factor to hinder the vectors from growing large.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.20)$$

The softmax operator is defined in Equation (2.21) where $\exp()$ is the standard exponential function e^x . It can be interpreted as scaling its input into a probability distribution.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (2.21)$$

The computed matrix $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$ can be interpreted as which part of the input embedding to attend to, while the matrix V represents the values that are wanted as outputs. The linear combination of these two then yields the final output.

The authors expand on this idea by introducing multi-head attention. Instead of using one single attention calculation using Q, K, V of dimension d , they instead calculate h different attention values with different Q_i, K_i, V_i of dimension d_q, d_k, d_v . The output of each calculation Z_i is called a head. These are independent of one another and can therefore be calculated in parallel. The heads are concatenated and transformed using the weight matrix W^O of dimensions $d \times d$, where $d = h * d_k$. The procedure for this calculation is given in Equation (2.22).

$$\begin{aligned} Q_i &= XW^{Q_i}, K_i = XW^{K_i}, V_i = XW^{V_i}, \\ Z_i &= \text{Attention}(Q_i, K_i, V_i), i = 1 \dots h \\ \text{MultiHead}(Q, K, V) &= \text{Concat}(Z_1, Z_2, \dots, Z_h)W^O \end{aligned} \quad (2.22)$$

The motivation behind the introduction of multi-headed attention was, according to the authors: "Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this." [36]

In the decoder, the attention calculations of the first layer are supplemented with masking. The embeddings that have been generated so far are masked by a masking matrix M consisting of zeros and infinitely low values for the values that are to be masked out. This has the effect of removing the correlation between masked queries and values. The masked attention mechanism is defined in Equation (2.23).

$$\text{MaskedAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T + M}{\sqrt{d_k}}\right)V \quad (2.23)$$

Layer Normalisation

Each layer of the transformer is followed by a layer denoted "Add & Norm" in Figure 2.13. This layer is, in fact, the addition of residual connection around the previous sublayer, as well as layer normalisation. The layer normalisation functions in nearly the same manner as batch normalisation. However, the mean and variance are computed across channels and the vectors.

A series of fully-connected layers usually follow the encoder-decoder sequence to function as a classification or regression head.

Vision Transformers

As noted in Section 2.3.7, the introduction of the vision transformer sparked interest in transformers for computer vision tasks. The structure of the vision transformer is similar to the original transformer. However, the vision transformer is only utilising a transformer encoder without the decoder. As such, it is only creating a feature map and not decoding it. This is done by inserting a head at the end of the transformer encoder. This makes the vision transformer flexible and able to create feature maps for a range of different architectures.

Also, instead of tokenising a sentence into a sequence of words, it splits an image into a sequence of patches. These patches are flattened and embedded into a d dimensional space using a trainable linear projection. These are injected with positional encodings, which are used as input for the encoder.

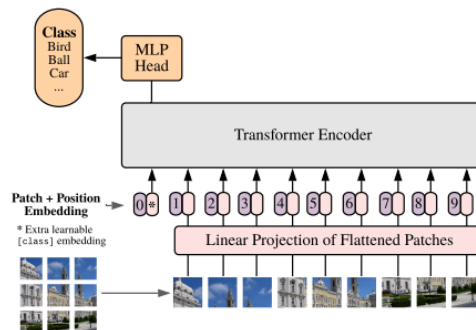


Figure 2.14: Overview of the Vision Transformer architecture [5].

All the feed-forward layers of the vision transformer, which are referred to as Multilayer perceptrons (MLPs) by the authors, use an activation function known as GELU.

The gaussian error linear unit (GELU)

The Gaussian error linear unit is an activation function introduced by Hendrycks and Gimpel [38], commonly used in most transformer type architectures. It is a variation of ReLU, in that it gates inputs by their value as opposed to their sign. The

original authors claim that: "increased curvature and non-monotonicity may allow GELUs to more easily approximate complicated functions than can ReLUs" [38]. It is defined in Equation (2.24), where ϕ denotes the Cumulative Distribution Function for Gaussian Distribution.

$$f(x) = \Phi(x)x \quad (2.24)$$

2.4 Related Work

The field of computer vision has, for the most part, been focused around 2D applications. This means that most well known public data sets, like ImageNet [39] or COCO [40], are 2D images. In recent years however, 3D methods are becoming more common [41]. In the author's experience, this also holds true for computer vision within medical imaging. This has been a natural development because of easier access to increasingly powerful hardware.

The methods proposed in this study are inspired by previous research within deep learning methods for landmark localisation, object detection and segmentation. Methods used in this thesis will be explained in more detail, both in Chapter 3. Within medical imaging, the most relevant research has been done within landmark localisation and segmentation.

2.4.1 Landmark Localisation

Most research within landmark localisation has centred around CNN-based methods. Noothout *et al.* [42] presented a method for anatomical landmark localisation in 3D CTA scans of the heart. The localisation pipeline was to extract small 3D patches of the full volume and feed these to a CNN with a fully connected regression head and a classification head. The network architecture is shown in Figure 2.15. The classification head classifies which patches contain a landmark, while the regression head predicts the 3D displacement vector from the patch to the landmark position. An average of the displacement vectors from patches classified to contain the landmark, are then used to calculate the final landmark positions. Using this method, they locate the three aortic valve commissures, which is the space between each leaflet anchored to the aortic wall, with an average error of 1.94mm.

Wester [43] used this same method for performing landmark localisation in 3D ultrasound images, where the purpose was to align ultrasound data with CT scans. Utilising a 3D version of ResNet-18 as the feature extractor, the model was able to predict the anatomical landmarks with an average error of 8.78mm.

Some studies have proposed methods for automatic estimation of cardiac parameters based on detected landmarks from ultrasound data. One such proposal is made by Taskén [44], who presented a method for automatic MAPSE estimation by segmenting the mitral annulus. This was based on tracking landmarks of the mitral annulus in 2D planes extracted from the 3D volume around the Long

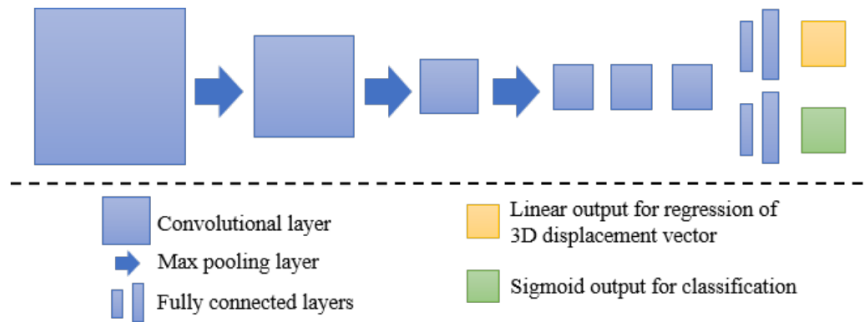


Figure 2.15: Overview of the architecture proposed for anatomical landmark localisation in 3D CTA scans by Noothout *et al.* [42]

Axis. The pipeline included several steps. However, the landmark localiser, an encoder-decoder model, was based on ResNet-50 as a feature extractor and an upsampling decoder to produce probability maps for the output coordinates. After post-processing, the model was able to predict landmarks of the mitral annulus with an average error of 3.2 ± 3.5 mm. The proposed model functioned as a 3D extension of work done by Nordal [45] on 2D images.

As for bounding box detection of the aortic valve in ultrasound images, no work has been done on 3D volumes. However, Ahmad Nizar *et al.* [46] surveys different well-known object detection methods for aortic valve detection in short axis 2D ultrasound images. The methods SSD [47] and Faster R-CNN [48] are compared as detectors, with the best accuracy score being 0.949.

At the time of writing, no transformer methods were found for landmark localisation or object detection in medical images of the heart. However, several transformer approaches have been proposed for the segmentation of volumetric images.

2.4.2 Segmentation

For volumetric segmentation tasks in general, it is common to use a U-Net-like architecture, as proposed by Ronneberger *et al.* [49]. These types of architectures are structured in an encoder-decoder fashion, with residual connections from the encoder layers to the decoder layers, as shown in Figure 2.16.

These methods are often based around CNNs; however, an increasing number of transformer U-Nets have been proposed for medical image segmentation. An example is the method UNETR, as presented by Hatamizadeh *et al.* [50], which utilises a vision transformer [5] as the encoder part of a U-Net to perform medical image segmentation on various 3D data sets. This method improves the score of the original U-Net on all data sets tested by the authors.

This work is expanded upon by Hatamizadeh *et al.* [51], which changes the encoder of UNETR with a specific transformer, well suited for object detection tasks,

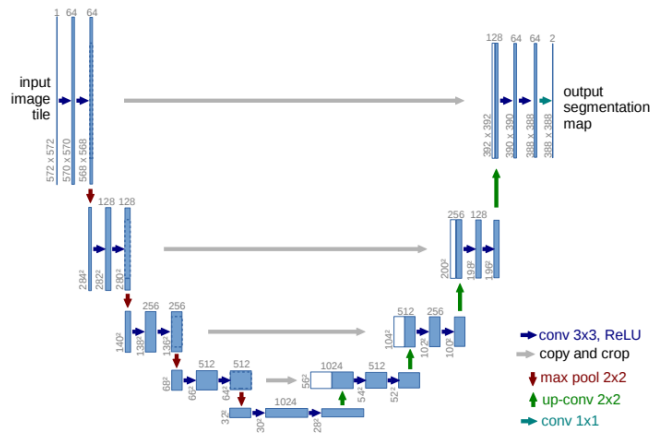


Figure 2.16: Overview of the U-Net architecture [49].

known as Swin transformer [52]. In their work, they propose Swin UNETR for brain tumour segmentation in volumetric MRI images. With these modifications, the proposed model outperforms UNETR.

The nnFormer, proposed by Zhou *et al.* [53], is an alternate approach to medical image segmentation in 3D volumes. It utilises the same principles as Swin UNETR, but modifies it by interleaving convolutional layers between the attention layers of the encoder. This enables higher accuracy and faster convergence as opposed to Swin UNETR.

Chapter 3

Method

3.1 Data Set

The data set for this study consisted of three-dimensional transesophageal echocardiography B-mode recordings from 80 patients. The images were acquired by cardiologists at St. Olavs University Hospital in Trondheim, Norway. The recording equipment used to produce the images consisted of state-of-the-art clinical scanners, named E9, E95 and S70, from GE Vingmed, Horten, Norway. The ultrasound images were recorded with a 6VT-D probe, and the depth of the scans was adjusted such that the record displays as much of the patient's left ventricle as possible. As such, a Mid Esophageal Long Axis view should be producible from the recordings.

An ethics approval was in place with regards to this study, and all patients have consented to participation in scientific studies. In addition, all recordings are anonymised and contain no personal data. The raw ultrasound beam data were scan-converted and exported from DICOM- to HDF5- format and made available by associate professor Gabriel Kiss. The HDF5 files available to the thesis author contained relevant meta-data such as image size, voxel size, probe angle, origo coordinates and ECG times. Each recording included one full heart cycle, amounting to between 3 and 18 frames per recording. The image data itself consists of a four-dimensional integer array formatted as a series of 3D grayscale images. The grayscale values range from 0 to 255. For each patient, there were between one and four recordings, leading to a total of 168 recordings.

The image size of the recordings varied greatly; however, most recordings were above 128x128x128. Most images had the same resolution in all three dimensions, but some were shorter in terms of depth than height and width. The grid-array configuration of the 3D ultrasound probes acquires images in a cone extending from the probe. As a result of this and poor alignment, the aortic root cannot be fully seen in some recordings. In addition, a few recordings had issues with image artefacts occluding the view of some of the anatomical structures. This led to some recordings being discarded from the study. A total of 54 recordings were discarded because of the low quality or visibility of the aortic valve. After dis-

carding these images, 114 remaining recordings were to be utilised in the study.

3.1.1 Annotation

All annotation of the data set was done manually by the project author using self-written software, providing the user with a graphical interface. In this interface, the ultrasound volumes were visualised as individual 2D slices through all three axes. However, a 3D visualisation could also be produced using attenuated Maximum Intensity Projection. Locating the aortic valve during the annotation process was guided by the online educational tool TEE: 3D Standard Views [13] as well as illustrations found in the ASE Guidelines and Standards for performing Transesophageal echocardiography [15].

The annotations were in the form of three-dimensional bounding boxes, as this was suggested by the thesis supervisor as an appropriate annotation format. During annotation, the coordinates of the lower-left corner, as well as the top right corner of the aortic valve, were located by slicing through the x-axis to the view where the aortic valve is at its most visible. Two additional points denoting the front and back of the aortic valve were located by slicing the volume along the x-axis to the closest and farthest visible views of the aorta. These points were combined and saved on the format $((x_{min}, y_{min}, z_{min}), (x_{max}, y_{max}, z_{max}))$. These coordinates denote the closest lower-left corner and the farthest high-left corner of a 3D bounding box surrounding the entire aortic valve. As mentioned in Section 3.1, recordings that the author was unable to annotate due to noise were discarded. A +2D visualisation of an annotated recording is shown in Figure 3.1.

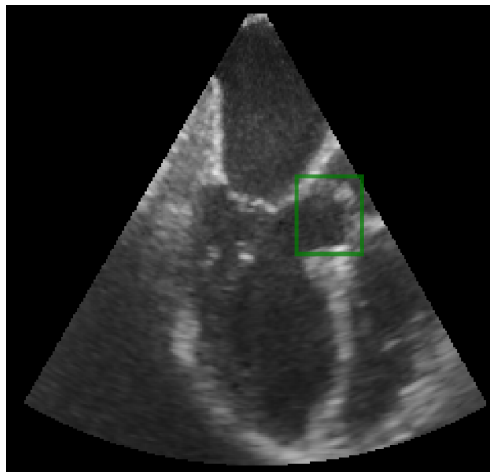


Figure 3.1: Example of annotated data.

Because of the relatively small movement of the aortic valve throughout the cardiac cycle, an annotation was only done in one frame but set to count for all frames of the recordings, saving a lot of annotation time. Thus, the bounds of the bounding box were set wide enough to make sure that the aortic valve was

contained throughout all frames of the recording.

3.1.2 Train-, Validation- and Test Split

As mentioned in Section 2.3.4, it is standard practice to split the data into three distinct sets for training, validation and testing. The decided split is shown in Table 3.1. In terms of percentages, the chosen split represents almost a 74-17-9 split, not far from the common 70-20-10 split. Because of the limited amount of data, it was chosen to prioritise the training split to ensure as good variation in the training data as possible.

Table 3.1: Data set split into train-, validation- and test data sets.

Data set	Recordings
Train	84
Validation	20
Test	10
Total	114

3.1.3 Pre-Processing

Before data can be used as input to a machine learning model, it has to undergo several pre-processing steps. A pre-processing pipeline was designed to handle this.

Because of the variation in image size in the data, the first step in the pre-processing pipeline was to resize all image volumes to the same size. The ratio between the original image size and the new size was then used to scale the labels accordingly.

After re-sizing, the images were normalised using min-max normalisation, as defined in Equation (3.1). This ensures that all grey tone values lie between zero and one, which helps the ANN to keep activations at a reasonable size.

As a final step, images are converted to tensor datatypes with 32-floating point (FP) precision and batched into mini-batches.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (3.1)$$

3.1.4 Post-Processing

Because of the resizing involved in the pre-processing step, the bounding box output from the network was on a different scale compared to the original ultrasound recording. To use the bounding box in a real-life scenario, it was necessary to resize the bounding box to match the original scale. This was completed by multiplying the output coordinates with the reciprocal of the ratio between the resized scale and the original scale for all dimensions. The post-processing was

not a part of the training pipeline because of efficiency considerations. Therefore the loss values were calculated on the down-scaled bounding boxes and labels.

3.1.5 Data Augmentation

Because of the limited data set, a technique known as data augmentation was utilised to artificially expand the amount of training data. Such methods commonly involve random rotations, flipping the image around an axis, applying distortion or varying image intensity. In this study, methods for applying random rotations and axis flips were implemented using affine transforms. Since an objective of the study was to examine the impact of data augmentation on the models, when training on limited data, the augmentation was made optional. This way, one could run experiments with or without data augmentation.

3.2 Aortic Valve Localisation

The main objective of this study was to propose and compare methods for localisation of the aortic valve in volumetric data, both using CNN-based, transformer-based, and hybrid architectures. As such, three different options using state-of-the-art methods from all three categories will be presented and tested. They will be compared using the method proposed in the author’s preliminary project work as a baseline. To ensure fair comparisons, all network architectures will be treated as feature extractors and serve as a backbone for the same type of regression head, consisting of fully-connected layers. The use of a simple regression head means that the network does not contain a classification head. A naive assumption is made that the network will find the aortic valve in every recording, negating the need for classification, so it will always output coordinates. This pipeline structure is shown in Figure 3.2.

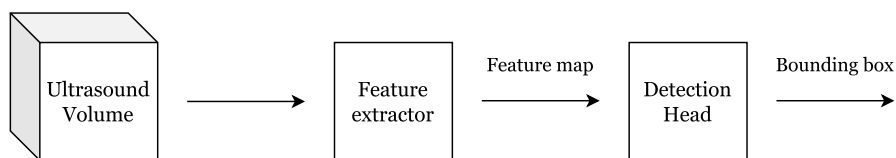


Figure 3.2: Overview of the proposed pipeline.

3.2.1 Baseline

The author’s preliminary project proposed a method called HeartNet. This was a lightweight 3D CNN, inspired by the VGG [54] family of neural networks. Note that this method is only included as a baseline to compare against and should not be considered a part of the work done for the thesis. The architecture of HeartNet, as proposed in the project, is shown in Table 3.2.

Table 3.2: HeartNet architecture for inputs sized 128x128x128.

Layer-type	Size	Input channels	Output channels
Conv3D	3x3x3	1	32
L-ReLU	N/A	N/A	N/A
MaxPool3D	2x2x2	N/A	N/A
Conv3D	3x3x3	32	64
L-ReLU	N/A	N/A	N/A
MaxPool3D	2x2x2	N/A	N/A
Conv3D	3x3x3	64	64
L-ReLU	N/A	N/A	N/A
MaxPool3D	2x2x2	N/A	N/A
Conv3D	3x3x3	64	128
L-ReLU	N/A	N/A	N/A
MaxPool3D	2x2x2	N/A	N/A
Flatten	64x64x64	128	1
Dropout	N/A	N/A	N/A
FC	65536x6	N/A	N/A

3.2.2 CNN Approach

Convolutional neural networks represents some of the most widely researched approaches to object detection methods. The ResNet architecture, as proposed by He *et al.* [34], is widely regarded as an effective and widely used backbone for feature extraction in images gaining first place in the ImageNet localisation and COCO detection challenges in 2015. The network is a residual network and consists of residual blocks with stacks of convolutional- and batch-normalisation layers followed by ReLU activations.

For this thesis, the ResNet-50 configuration was chosen as it provides a nice compromise between computational requirements and accuracy. Resnet-50 utilises stacks of a type of residual blocks called bottleneck blocks. The bottleneck block consists of three convolutional layers with batch normalisation and ReLU layers in between. Each bottleneck contains a residual connection from block input to block output, as in described in Section 2.3.6. An overview of this configuration is provided in Figure 3.3.

3.2.3 Transformer Approach

The standard vision transformer has largely seen use in the field of image classification. The increased resolution required for image analysis compared to text translation has proved a challenge when adopting transformers to the image domain. Using traditional vision transformers for image tasks have a quadratic computation complexity with respect to image size, because of global self-attention calculations. One approach to more effectively extract features from images, is the Swin Transformer, proposed by Liu *et al.* [52]. The Swin transformer is a general-purpose backbone for computer vision tasks and has reported great accur-

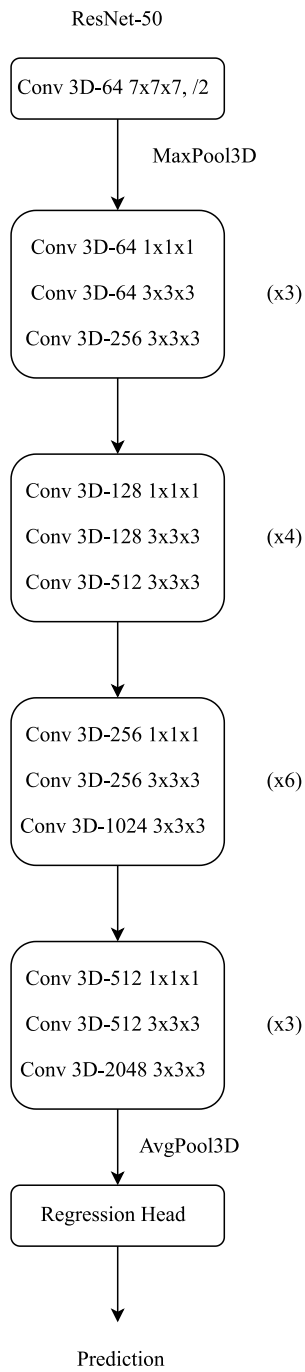


Figure 3.3: Overview of the architecture of 3D ResNet-50. Conv 3D-64 7x7x7, /2 denotes a 3D convolution layer with 64 kernels of size 7x7x7 and a stride value of 2. The number beside each bottleneck block denotes how many blocks are repeated of the same type.

acy in object detection tasks such as the COCO detection task. It utilises a modified version of the transformer’s MSA called window based multi-head self-attention (W-MSA). The concept behind W-MSA is to calculate attention only in local windows and then shift the windows between attention layers. It also merges image patches in deep layers. This enables the Swin Transformer to calculate attention in a hierarchical manner, without linear complexity with respect to the input size. The Swin Transformer uses GELU as activation functions.

A 3D version of the original swin transformer called Video Swin Transformer is presented in Liu *et al.* [55]. The architecture of this version can be seen in Figure 3.4.

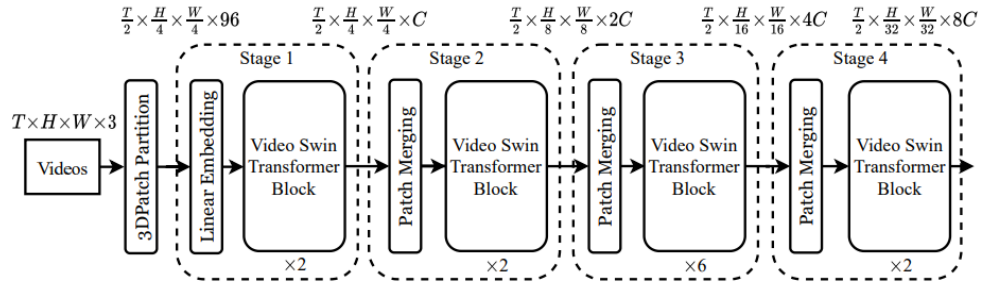


Figure 3.4: Overview of the architecture of the Video Swin Transformer (Swin-T) [55].

The Video Swin Transformer is identical to the Swin Transformer, but extends all concepts to account for an extra dimension. The W-MSA mechanism for 3D can be viewed in Figure 3.5. This transformer is made to handle video, where the extra dimension is the temporal dimension. However, in our case, we propose to utilise the same architecture but handle the extra dimension as spatial.

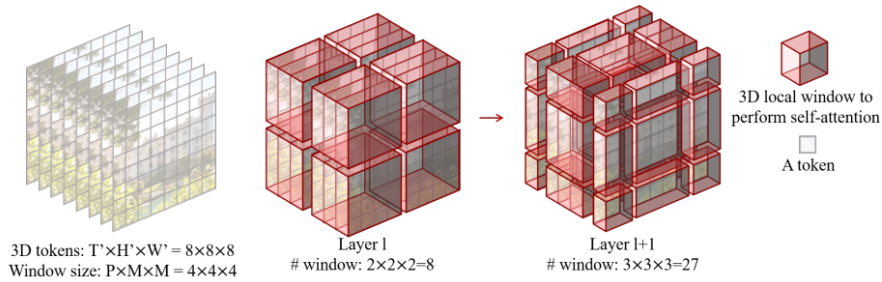


Figure 3.5: Illustration of 3D window based multi-head self-attention. Notice the shifting of windows occurring in layer $l + 1$ [55].

3.2.4 Hybrid Approach

As a hybrid approach between CNNs and transformers, the method proposed by Zhou *et al.* [53] called nnFormer is used. nnFormer is a segmentation network

for medical images, based on a U-Net architecture. It borrows a lot of principles from the Swin Transformer by using slightly modified 3D versions of the W-MSA. In addition, the nnFormer utilises several layers of small convolutional kernels as an embedding block, instead of the single large convolutional layer of Swin Transformer.

The main idea of nnFormer is to use several blocks of W-MSA layers interleaved with residual convolutional blocks. In this way, theoretically, the network is able to utilise properties from both CNNs and transformers. In similarity with Swin Transformer, nnFormer used GELU as activation functions.

Because nnFormer is a U-net, it is composed of several downsampling encoding layers creating a feature map, before these are upsampled to create a segmentation output. There are also skip connections between each encoding-block and the decoding-layers. The network structure is shown in Figure 3.6.

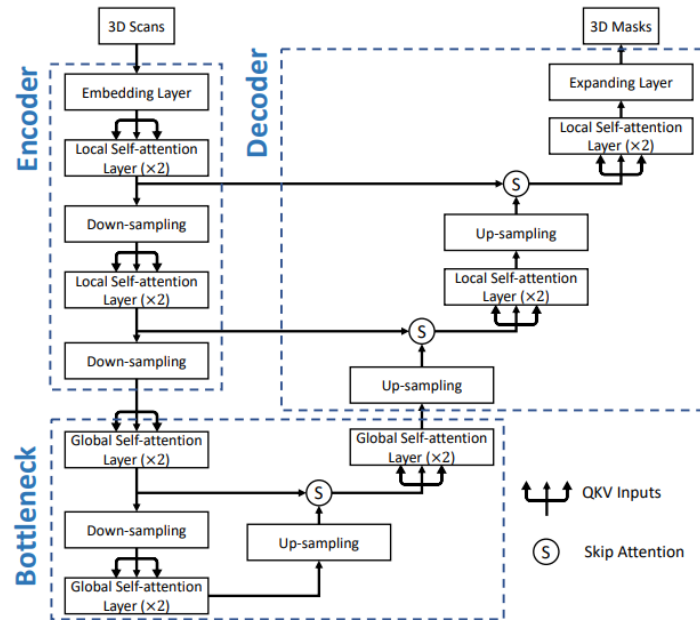


Figure 3.6: Illustration of the U-Net like structure of nnFormer. In this study, only the encoder and bottleneck part will be used. [53].

To make nnFormer act as a feature extractor for this thesis, we propose to only utilise the down-sampling encoder part of the nnFormer, and append a regression head at the end. In theory, this approach should be close to a hybrid between ResNet and Swin Transformer.

3.3 Experimental Setup

To compare the different network architectures against one another, several experiments were completed. This study intends to examine different parameters and

configurations impact on the performance of each network type. These experiments will be the grounds for discussing the proposed architecture in Chapter 6 and answering the research questions of the thesis.

3.3.1 Hardware and Software

All models are trained on an Nvidia Quadro RTX 8000 with 48GB of graphics memory. For each model, the maximum available batch size given the memory constraints is used. All ANNs are created and trained using Python 3.8.10, and PyTorch compiled for CUDA 10.2. PyTorch is a library for machine learning and contains operations necessary for building ANNs, along with methods for back-propagation using reverse-mode automatic differentiation.

Training with 16-FP half precision was initially tested, to increase training speed and batch size. However, this resulted in numerical instability due to small gradients. Subsequently, it was decided to do all training using 32-FP full precision.

3.3.2 Input Size

Because of limited computing power in the author's project work, the baseline model was most effective when trained on 64x64x64 images. However, because of the increased memory of the Quadro RTX 8000, input models were trained with a larger input size. After initial testing, it was concluded that 128x128x128 represented a reasonable trade-off between input size and batch size. Because of the complexity of both the video Swin Transformer and the nnFormer, anything higher than 128x128x128 was deemed unfeasible.

3.3.3 Loss Function

For bounding box regression, the choice of loss function was not straightforward. Some common object-detectors, such as R-CNN [48], use MSE loss. However, as noted in Girshick [56], a modified version of L1, called smooth L1, has been noted to perform better on such tasks. The smooth L1 loss function is defined in Equation (3.2) In the author's project work, it was found that the proposed model performed best when trained using MSE loss. However, a test was conducted to evaluate if this holds for the proposed methods of this study as well. Thus, initial training runs were done for all models using both MSE and smooth L1.

$$L = \frac{1}{N} \sum l_n, \quad l_n = \begin{cases} \frac{0.5(x_n - y_n)^2}{\beta}, & \text{if } |x_n - y_n| < \beta \\ |x_n - y_n| - 0.5 * \beta, & \text{otherwise} \end{cases} \quad (3.2)$$

3.3.4 Optimiser

A variation of the Adam optimiser known as Adam with Decoupled Weight Decay Regularization (AdamW) was used as the optimiser for all training runs. This op-

timiser was first presented by Loshchilov and Hutter [57] and performs the weight decay step in an alternative manner compared to the original Adam. The choice of optimiser was motivated by previous results from the author's project work, in which AdamW performed significantly better than SGD with momentum.

3.3.5 Learning Rate Optimisation

The choice of learning rate usually has a significant impact on training performance. A systematic approach was taken to test a range of different learning rates. The recommended learning rate of the AdamW optimiser is 0.001. The different learning rates explored ranged from 0.0001 to 0.01.

Even though the AdamW optimiser uses variable learning rates for each parameter, it can be beneficial to vary the base learning rate throughout training iterations. This is called learning rate scheduling. For this study, additional training runs were used with a scheduler called cosine decay with restarts, as presented in Loshchilov and Hutter [58]. This scheduling also included a warm-up period. Theoretically, this allows the network to converge faster in earlier epochs without exploding gradients. Moreover, the gradual decrease of learning rate throughout training has been shown to have a positive effect on training performance [58].

3.3.6 Pre-Training

Transformers are notoriously hard to train and often require large amounts of data to perform well. However, it has been shown that by using proper optimisation and hyperparameters, one can train transformers from scratch on small data sets and obtain reasonable results [59]. The most common approach, however, is to fine-tune models pre-trained on a larger corpus of data. To examine if pre-training makes a positive difference, both transformer architectures were tested with and without pre-training.

The data set used to pre-train the nnFormer is the synapse multi-organ CT scan set [60]. This is a 3D data set, consisting of 30 abdominal CT scans with segmentation labels for eight organs. Even though this is a segmentation task, it still consists of medical image volumes, and the hypothesis is that the pre-training using this data should provide a good foundation for further training. For the video Swin transformer, however, no relevant data sets could be found in which pre-training was performed. In general, it was hard to find models pre-trained on volumetric data sets. The Swin Transformer was therefore pre-trained on the kinetics-400 set [61]. This is a dataset consisting of short videos of persons doing an activity and labels for activity classification. It is not a directly volumetric dataset, although it can be treated as such by treating the temporal dimension as a spatial one.

3.3.7 Data Set Configuration

As mentioned in Section 3.1, each TEE recording span one heart cycle. The author’s project work was only concerned with detection in the first frame of each recording, where the aortic valve is in a closed position. Subsequently, the model proposed in the project was only trained on the first frames. It is, however, desirable to be able to detect the aortic valve in all stages of the heart cycle.

As mentioned in Section 3.1.1, the bounding boxes were made wide enough to include the aortic valve throughout all the frames. This meant that it was possible to train on all the included frames of each recording, making the total data set many times larger. Theoretically, this should boost performance and make the models able to detect the aortic valve in all frames. Separate training runs were done for all models on both the first frame of each recording and the whole data set. The resulting models were evaluated individually on both data set configurations. The number of frames in each data set is presented in Table 3.3.

Table 3.3: Data set size with- and without all frames.

Data set	Single frames	With all frames
Train	84	1156
Validation	20	274
Test	10	117
Total	114	1547

3.3.8 Data Augmentation

To evaluate the impact of data augmentation on the performance of the models, all experiments of this study were performed both with- and without data augmentation. The type of data augmentation implemented was random rotation and flipping the image around a random axis. Each augmentation was performed with a probability $p = 0.2$, and the degree of rotation θ was picked at random within the interval $[-0.15, 0.15]$.

3.3.9 Evaluation

To evaluate network performance, several metrics were used. Since experiments were done using both MSE- and smooth L1 as loss functions, these were also used as metrics for each model to compare the effect of the models’ ability to minimise one loss function on the other.

It is also desirable to have a more intuitive evaluation metric. In this case, we propose to use the euclidean distance of the predicted points to the ground truth points, given in millimetres (mm). Since the task of this study is to find a bounding box, and not a single coordinate, the average euclidean distance of both corner coordinates of the bounding boxes is used as the final score. The precise formula is found in Equation (3.3), where $\hat{y}_n = [\hat{x}_1, \hat{x}_2, \hat{x}_3]$ denotes the n -th corner of the

predicted bounding box and $y_n = [x_1, x_2, x_3]$ represents the n -th corner of the ground truth annotation.

$$f_{avgeuc}(\hat{y}, y) = \frac{1}{N} \sum_n \|y_n - \hat{y}_n\|_2 \quad (3.3)$$

As mentioned in Section 3.1.4, output bounding boxes are not in the same scale as the original TEE recordings. To get a metric applicable in a real-life scenario, the average euclidean distance is also calculated after post-processing. This metric is referred to as scaled euclidean distance.

Chapter 4

Results

During the experimental phase of the study, more than a total of 88 different training runs were finished. The results from the training relevant to the proposed experiments are collected and presented in this section. Brief comments about the results will be made after each experiment is presented. However, the results will be discussed in detail in Chapter 5.

4.1 Comparison of Loss Functions

Table 4.1, presents the evaluation scores for eight different training runs. These are categorised after model and loss function, showing the effect of smooth L1 and MSE on model accuracy. All models were trained for 50 epochs.

Table 4.1: Results of training runs comparing loss functions.

Architecture	Loss Function	L1	MSE	Euclidean (mm)	Scaled Euclidean (mm)
HeartNet	Smooth L1	5.14	83.42	8.59	20.11
	MSE	5.34	92.01	8.79	20.52
ResNet	Smooth L1	4.45	77.00	7.70	16.76
	MSE	4.76	88.59	7.99	18.11
Swin Transformer	Smooth L1	8.70	179.22	14.12	20.39
	MSE	9.03	182.83	14.48	20.94
nnFormer	Smooth L1	8.46	183.64	14.10	20.98
	MSE	8.78	184.21	14.21	20.92

From the results, it is clear that the models trained with the smooth L1 loss function outperformed the others. It is interesting to note that the models trained with smooth L1 were better in minimising MSE than the models trained with MSE. This shows that smooth L1 may be a better and more stable loss function for bounding box regression, as proposed by Girshick [56]. However, all other experiments were performed with models trained using MSE, due to them being scheduled before these results were analysed.

4.2 Effect of Pre-Training on Transformers

4.2.1 Training Performance

It was noted for both the swin transformer and the nnFormer that training runs with pre-trained weights started with lower loss values, compared to the ones trained from scratch. However, even though the training was at a better starting point due to the pre-training, it failed to make an impact on the final loss values. This can be seen for the nnFormer in Figure 4.1 and the swin transformer in Figure 4.2.

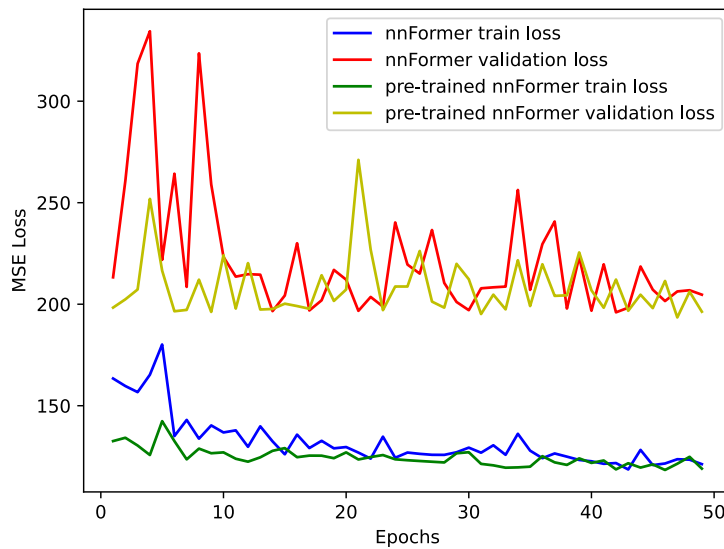


Figure 4.1: Training metrics of nnFormer.

For each model, the pre-trained losses start at a lower value than the typical losses for both models. This holds for both training- and validation loss.

4.2.2 Network Performance

Table 4.2 presents the impact that pre-training had on model accuracy. The table shows that the swin transformer performed better when utilising pre-trained weights than when trained from scratch. The nnFormer, on the other hand, performed worse when pre-trained. Subsequent experiments were performed with pre-training for the swin transformer and without pre-training for the nnFormer.

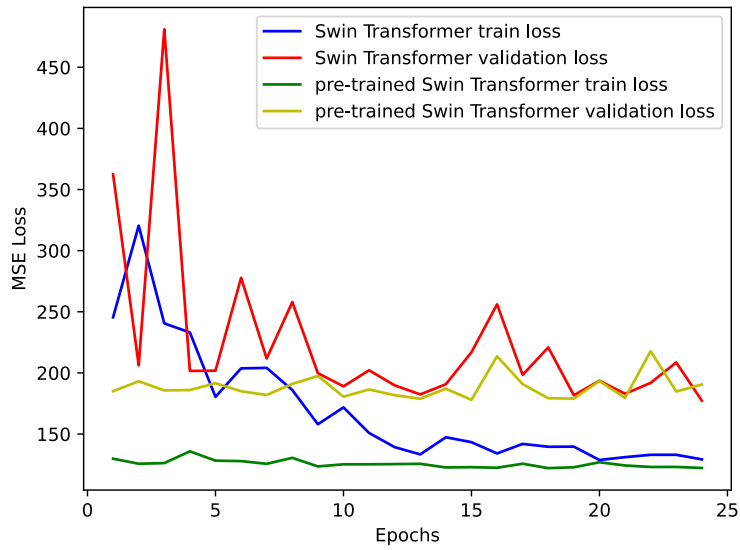


Figure 4.2: Training metrics of swin transformer.

Table 4.2: Results of training runs comparing pre-training and training from scratch. The scores are from evaluation done on the test set.

Architecture	Pre-trained	L1	MSE	Euclidean (mm)	Scaled Euclidean (mm)
Swin Transformer	No	9.03	182.83	14.48	20.94
	Yes	8.19	174.99	13.63	20.17
nnFormer	No	8.78	184.21	14.21	20.92
	Yes	9.67	198.14	15.12	21.99

4.3 Learning Rate and Scheduling

Hyperparameter tuning of the learning rate was done for all four models. The evaluation of the learning rate influence was done on a smaller subset of the total data set to reduce computing time. For HeartNet, the optimal learning rate of $\gamma = 0.001$ was already known from the preliminary project. Adding learning rate scheduling to this provided no additional increase in performance.

It was found that ResNet benefited from training with a lower learning rate than HeartNet. This is consistent with the optimal learning rates used by Nordal [45]. However, it was also found that including learning rate scheduling resulted in marginal improvements. The impact of learning rate scheduling on the training of ResNet can be seen in Figure 4.3.

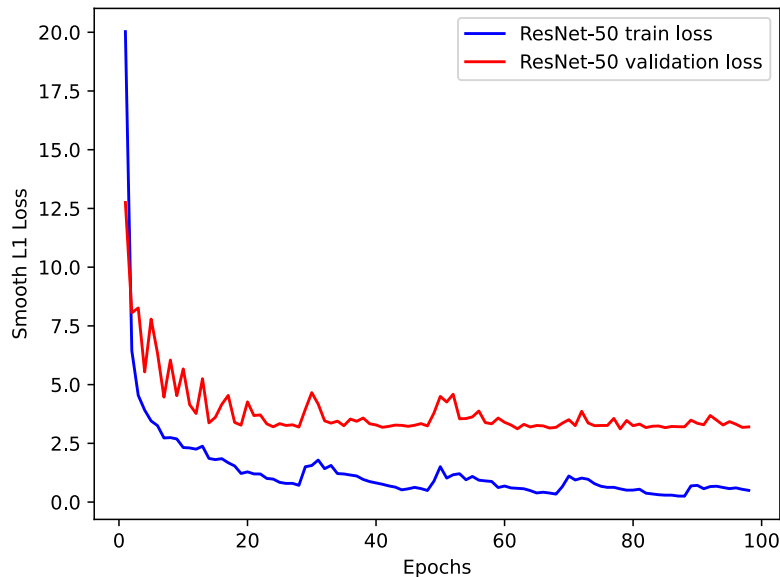


Figure 4.3: Impact of learning rate scheduling on ResNet training. Notice the little bumps in the graph indicating where the scheduler performs restarts.

Both the Swin Transformer and the nnFormer plateaued early in the training process when using the default learning rate of 0.001. The Swin Transformer was found to have an optimal learning rate of $\gamma = 0.0001$, with no additional performance increase from learning scheduling.

The nnFormer generally struggled to converge when using static learning rates. Adding learning rate scheduling with restarts helped convergence rates across later epochs in the training runs in some cases. However, it should be noted that training convergence was not always stable, even when using learning rate scheduling. The optimal parameters for learning rate scheduling were the same for both ResNet-50 and the nnFormer. The scheduling parameters are presented in

Table 4.3.

Table 4.3: Configuration of cosine decay learning rate scheduler with warmup and restarts. These values were used both for ResNet-50 and the nnFormer.

Parameter	Value
warmup-start-lr	0.0001
warmup-epochs	10
init-decay-epochs	20
min-decay-lr	0.00003
restart-interval	20
restart-lr	0.0009

4.4 Data Set Configuration

Training runs were performed for all models on the single-frame data and the whole sequence data. The performance of each model was evaluated on both data sets to see the impact of the extra data. Table 4.4 presents the results from this experiment.

Table 4.4: Network performance when trained on single frame- or multi frame data set and evaluated on both data sets. The "training"-column denotes which dataset the models were trained on, and the "evaluation"-column presents which data set it is evaluated on.

Architecture	Training	Evaluation	L1	MSE	Euclidean (mm)	Scaled Euclidean (mm)
HeartNet	Single	Single	4.90	77.60	8.21	19.45
		Multi	5.47	82.58	8.97	20.95
	Multi	Single	5.29	83.54	8.62	20.10
		Multi	5.29	81.22	8.68	20.24
ResNet	Single	Single	5.50	88.41	8.95	20.34
		Multi	5.98	96.99	9.62	21.41
	Multi	Single	4.83	78.40	8.02	17.34
		Multi	4.94	79.83	8.19	17.84
Swin Transformer	Single	Single	6.55	110.43	10.56	23.17
		Multi	6.92	120.35	11.03	23.42
	Multi	Single	5.00	85.17	8.35	18.65
		Multi	5.23	85.23	8.48	18.62
nnFormer	Single	Single	8.48	177.53	14.04	20.76
		Multi	8.49	178.63	14.08	20.45
	Multi	Single	5.02	85.60	8.48	18.10
		Multi	5.35	88.50	8.81	18.13

For all models, it can be seen that training on the larger data set with all

frames results in better accuracy. The difference is especially noticeable for the swin transformer and nnFormer. The only case in which training on the single-frame data set results in better performance, is for HeartNet. However, this is only the case when evaluating on the single-frame data. Another point worth noting is the fact that there are primarily minuscule differences between evaluations done on the single-frame data and the multi-frame data.

4.5 Data Augmentation

As mentioned in Section 3.3.8, all experiments were run twice, both with- and without data augmentation enabled. To maximise the impact of data augmentation, the evaluation of this effect was conducted on the two data set configurations. As such, the evaluation of data augmentation impact will be done using a comparison of the results from Section 4.4 with data augmentation disabled and enabled. This is presented in Table 4.4 and Table 4.5 respectively.

Table 4.5: Network performance with data augmentation when trained on single frame- or multi frame data set, and evaluated on both data sets.

Architecture	Training set	Evaluation	L1	MSE	Euclidean (mm)	Scaled Euclidean (mm)
HeartNet	Single	Single	6.11	91.60	10.07	22.56
		Multi	6.71	97.64	10.74	23.49
	Multi	Single	5.38	84.58	8.81	20.17
		Multi	5.40	82.86	8.92	20.15
ResNet	Single	Single	6.14	92.56	9.95	21.81
		Multi	6.25	97.15	10.07	21.29
	Multi	Single	4.35	81.38	7.89	18.46
		Multi	4.71	84.58	7.95	18.34
Swin Transformer	Single	Single	6.91	101.39	10.91	21.70
		Multi	8.14	129.86	12.61	24.17
	Multi	Single	6.65	110.21	10.63	22.42
		Multi	6.61	107.74	10.66	22.88
nnFormer	Single	Single	6.23	95.50	10.12	22.17
		Multi	7.87	129.33	12.22	24.43
	Multi	Single	6.00	89.11	9.37	20.80
		Multi	5.89	94.70	9.59	20.24

The effects of data augmentation were somewhat counter-intuitive. Generally, across all experiments, data augmentation has detrimental effects on accuracy, even though one would expect the opposite. From the results presented in this section, it is found that every model performs worse with data augmentation. The only exception is the nnFormer, which performs slightly better when trained on the single-frame data set with augmentation. However, the nnFormer trained on the multi-frame data without augmentation outperforms all other nnFormer runs.

4.6 Inference Time

Since the proposed methods for aortic valve localisation might have intraoperative applications, inference times are a relevant evaluation metric to consider. The average inference times measured over the whole data set are presented in Table 4.6 for each model.

Table 4.6: Network performance measured in inference time. Values are given in frames per second. Inference time is measured on the Nvidia Quadro RTX 8000, with the models loaded in evaluation mode and gradient calculations disabled. Pre-processing is included in the inference time.

Architecture	Inference time (FPS)
HeartNet	9.75
ResNet	7.18
Swin Transformer	7.08
nnFormer	7.17

It is seen from Table 4.6 that HeartNet is the model with the best performance. However, this is most likely due to its lower complexity in comparison with the other models. Swin Transformer is by far the largest model, but also the slowest.

4.7 Final Localisation of the Aortic Valve

As a final evaluation of the models' ability to localise the aortic valve in whole 3D TEE volumes, the parameters and configurations with the best performance from the previous experiments were utilised. Each ANN architecture presented in the thesis was trained with the model's best found configuration for 100 epochs. The model performances are presented in Table 4.7.

Table 4.7: The final results after training each model with its best configuration for 100 epochs.

Architecture	L1	MSE	Euclidean (mm)	Scaled Euclidean (mm)
HeartNet	4.86	76.46	8.20	18.65
ResNet	4.91	88.13	8.37	17.40
Swin Transformer	5.22	87.56	8.60	18.15
nnFormer	5.40	88.14	8.69	18.52

As can be seen from Table 4.7, ResNet-50 is the best performing network when taking scale into account. Generally, the CNNs outperform the transformer and hybrid architecture, but not by much. Table 4.8 shows the best and the worst scores for each model.

Table 4.8: The average final results with the scores for the best and worst frame. The number in the columns represent the index of the recording in the data set, with the euclidean distance in the parentheses.

Architecture	Scaled Euclidean (mm)	Best Recording (mm)	Worst Recording (mm)
HeartNet	18.65	7 (5.79)	6 (30.37)
ResNet	17.40	4 (6.41)	6 (36.17)
Swin Transformer	18.15	4 (4.80)	6 (35.04)
nnFormer	18.52	4 (4.55)	6 (35.34)

4.8 Visualisations

This section showcases a visualisation of the best and the worst prediction made by the top-performing model ResNet-50. Figure 4.4 shows the best prediction and Figure 4.5 shows the worst. The red bounding box is the prediction done by the network, while the green bounding box is the ground truth annotation. The visualisations consist of three orthogonal slices of the 3D volume through the aortic valve. The lines represent the positions of the other views.

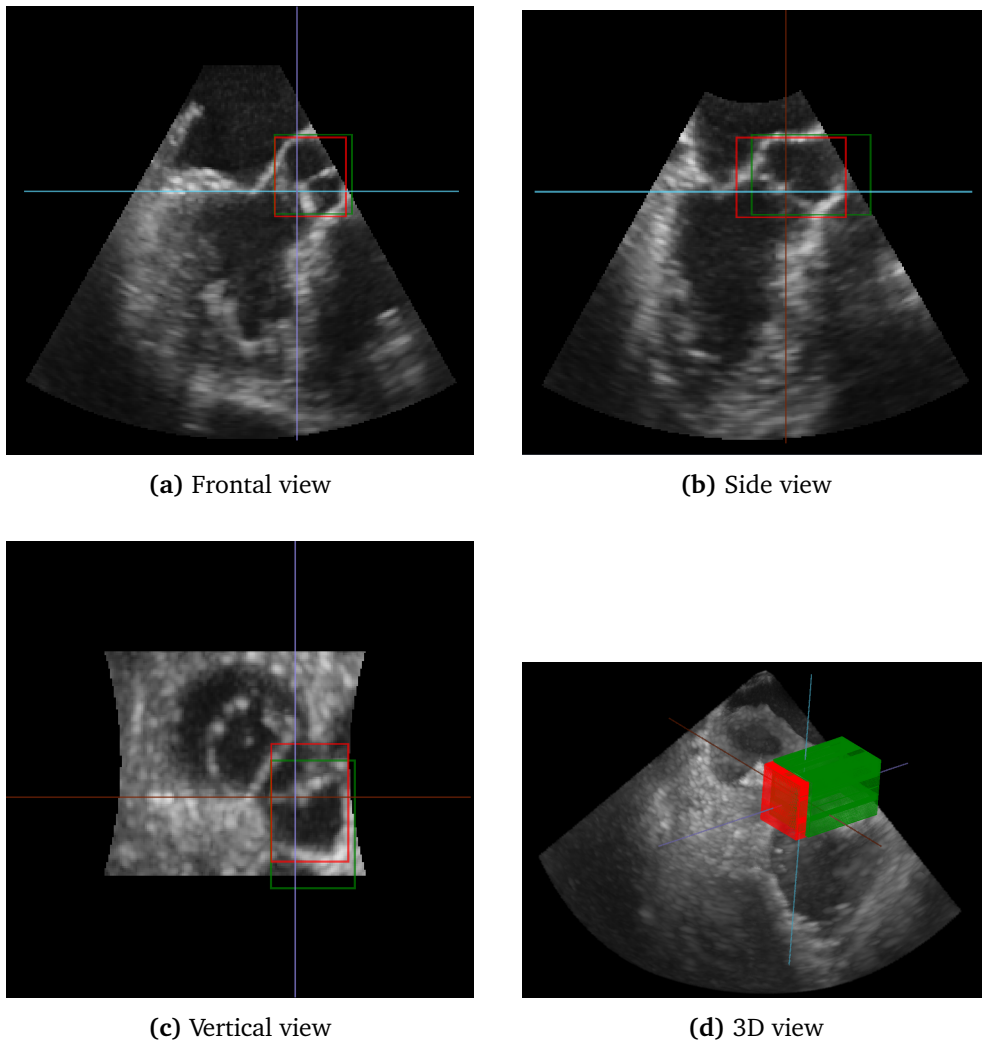


Figure 4.4: Visualisation of the best prediction by ResNet-50.

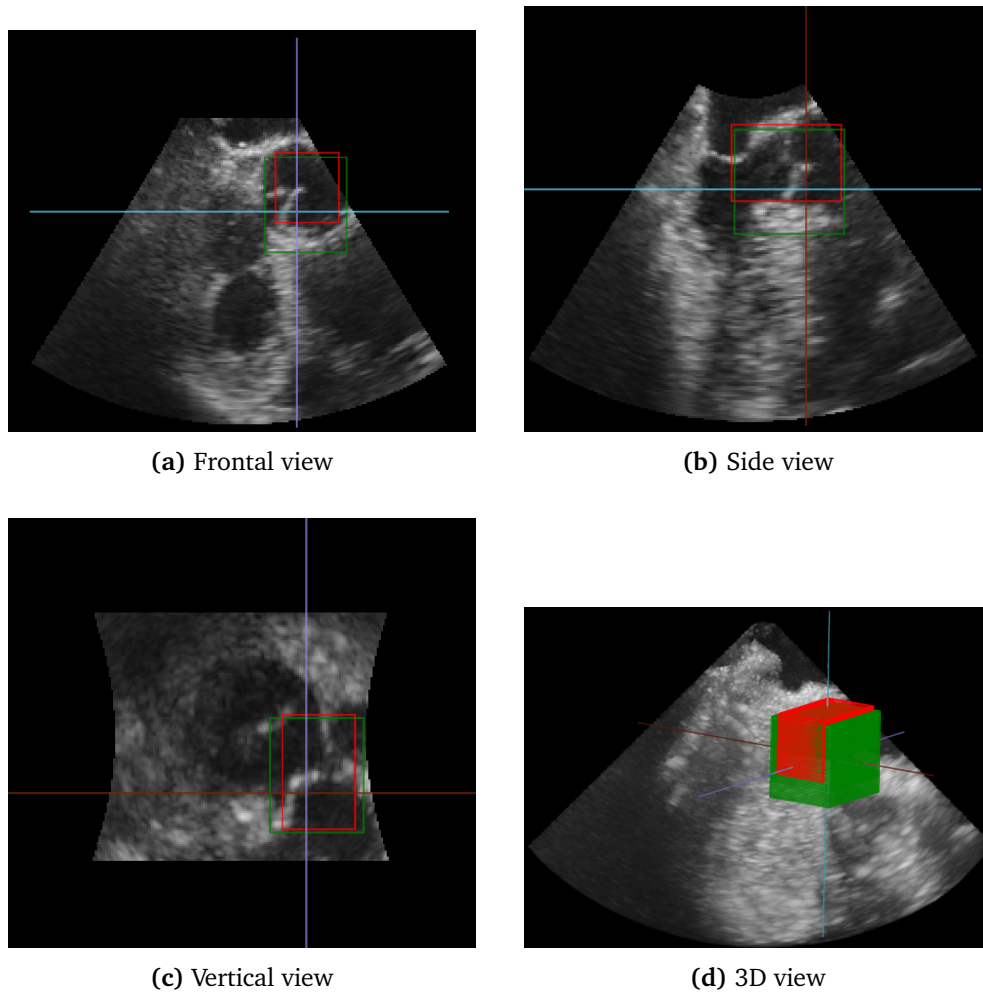


Figure 4.5: Visualisation of the worst prediction overall. Done by ResNet-50.

Chapter 5

Discussion

5.1 Data Set and Pre-Processing

All TEE recordings where the author was able to locate the aortic valve were included in the data sets and applied throughout the experiments. Recordings, where poor image quality made the author unable to do annotations, were discarded. As a result, the network is only trained on data where the author was able to locate the aortic valve. However, it should be noted that the data included in the data set, were also of varying quality, so it should resemble the real-life scenario of varying data quality, to a certain degree. Discarding some recordings from the data set ultimately reduces the available training data. Due to the fact that most machine learning pipelines benefit from increased amounts of data, it is likely that collecting more data would result in increased accuracy of the proposed methods.

The recordings acquired are expected to represent a wide variety of patients, as no patient selection was made. The patients from which the data was obtained, underwent TEE as a part of standard care, with no selection based on cardiac function. It can therefore be said that the data should closely resemble real-world cases in which the image quality is sufficient enough to see the aortic valve.

The pre-processing pipeline is designed to closely resemble pre-processing done in other traditional object detection methods. Re-sizing is necessary to enable the use of whole image volumes as input for all methods. And normalisation is needed to scale the voxel values to a scale suitable for ANNs. The re-sizing and normalisation methods used are considered standard practice and should not contribute negatively to method performance.

5.2 Annotations

Since all annotation of the TEE recordings were done by the thesis author, it can be claimed that the annotations do not hold to the same standards as an expert cardiologist would be able to provide. However, as mentioned in Section 3.1.1, the annotation work was guided by the online educational tool TEE: 3D Standard

Views [13] as well as illustrations found in the ASE Guidelines and Standards for performing Transesophageal echocardiography [15]. The process was done with the author's best effort, and recordings where annotation could not be done with confidence, were discarded. Consequently, the annotations should provide a basis accurate enough to perform training on.

The annotation process represents a significant amount of work. This is especially the case when the annotations are in the form of 3D bounding boxes. As mentioned in Section 3.1.1, it was decided to use one annotation per recording for all frames of the recording, to drastically reduce the workload of manual annotation. This was thought to be a reasonable compromise between reduced workload and accuracy, due to the relatively low movement of the aortic valve throughout the aortic cycle. Under the assumption that the ultrasound probe remains stationary in relation to the patient during a full recording, the only movement of the valve should be from the heart itself. Even though this assumption is made for the study, one can not guarantee that this is the case. To facilitate the use of one annotation for the whole recording, the bounds of the bounding boxes were extended slightly such that the aortic valve was contained by the bounding box throughout the entire recording. As a consequence, this decision might have caused an impact on localisation accuracy.

A challenge with the usage of bounding boxes arises when the objects are not perfectly vertically aligned. To encapsulate the entire aortic valve in an axis-aligned bounding box, it is sometimes necessary to make the bounds of the annotation wider and higher than the aortic valve would have been if it was vertically aligned. This may inflict additional accuracy penalties on the training of the proposed localisation methods.

Both of the above reasons for widening the bounding boxes do, in fact, negatively affect performance. This is clearly shown in Figure 4.5, where it is seen that the predictions do not look far off. The red prediction is closely centred around the aortic valve. The green annotation bounding boxes, however, are wider than they have to be. The effect of these wide bounding box labels is worth noting when evaluating the final numerical results. This might suggest that network performance could be more accurately evaluated visually than numerically.

5.3 Network Architecture

The proposed methods are meant to fulfil the overall thesis goal, which is to function as a method for automated aortic valve localisation. In addition, they are to utilise the whole 3D volume as input, according to Research Question 4. All the presented methods consisted of a feature extractor backbone and a lightweight, fully connected regression head. The regression head was kept simple and consistent across architectures to ensure that the evaluation of the different architectures would be based on the backbone. It was assumed that using a more advanced bounding box detection head would compensate for differences in architecture and not enable a thorough comparison. Because of this, it is expected that the

methods could achieve higher accuracy if more sophisticated bounding box detection heads such as RCNN [48] or SSD [47] were to be used.

The proposed architectures differ in size. The transformer-based models are much larger than the other models. Total parameter count for each model is shown in Table 5.1.

Table 5.1: Number of trainable parameters in each model.

Architecture	Parameter count
HeartNet	781 734
ResNet	46 167 366
Swin Transformer	430 461 112
nnFormer	130 609 954

5.4 Parameter Search and Experiments

To answer Research Question 3, several experiments regarding parameter search were carried out. The insights gained from the experiments are presented in this section.

5.4.1 Loss Function

Table 4.1 show that all architectures showed improved performance as a result of training with Smooth L1 instead of MSE. This experiment echoes the sentiment from Girshick [56] that Smooth L1 loss is generally to be preferred over MSE for bounding box detection tasks.

5.4.2 Pre-Training

As transformers are notorious for being hard to train with limited datasets, the pre-training experiment was only executed for the Swin Transformer and nnFormer. The results are a bit counter-intuitive in that the nnFormer, which was trained on the most relevant data set (synapse), did not gain anything from pre-training, except for lower loss values in the initial training epochs. The swin transformer, however, did see a marginal performance increase, even though the data set for pre-training (kinetics-400) was from a completely different domain. It is hard to say what made these results the opposite of what is expected; however, it might be due to different normalisation procedures. If pre-trained weights trained on more relevant data sets could be acquired, it would probably be beneficial to utilise those in further research.

5.4.3 Learning Rate

Learning rate tuning is a time-consuming task, especially for three different models. Because of the time taken to evaluate a single choice of learning rate, it is only feasible to test a limited amount of learning rates. This means that the learning rates found for each model might not be the actual optimal parameters. Given more time, one might be able to test parameters with finer increments than what was done for this thesis. It is interesting to note that nnFormer benefited greatly from scheduling, while the Swin Transformer performed optimally with a static learning rate of 0.0001. It is hard to say what attributes make a model benefit more from scheduling versus another. However, since the nnFormer can be interpreted as a hybrid between the Swin Transformer and ResNet, it might inherit properties from both of them.

5.4.4 Data Set Configuration

The utilisation of all frames in each recording had a significant impact on accuracy for most models. Table 4.4 shows that all three architectures proposed in this thesis gain at least 2mm in scaled euclidean score by training on the expanded data set, as compared to only one frame of each recording. The Swin Transformer gains almost 5mm in the scaled euclidean metric and is the model with the largest gain from the extra data. This is probably due to the Swin Transformer being the largest model. As shown in Table 5.1, the swin transformer has almost four times the amount of trainable parameters as nnFormer and ten times the amount of ResNet. The amount of trainable parameters usually corresponds with longer training times, but can also be seen as an indication of how much training data the model needs. This assumption was partly confirmed during the experimental phase. As can be seen when comparing Table 5.1 and Table 4.4, the performance gain from extra data is somewhat corresponding to the model size of both transformer architectures.

ResNet also sees an increase of 3mm in accuracy when training on extra data. However, it is notable that nnFormer roughly matches ResNet performance when trained on the single-frame data set, even though it is a much larger model. Intuitively one would argue that ResNet would perform better than nnFormer on limited data, since nnFormer uses transformer concepts. This shows that one of the strengths of the hybrid architecture is that it possibly makes the model less data-hungry. As noted earlier, all models would probably perform better if given access to more training data.

5.4.5 Data Augmentation

When comparing Table 4.4 and Table 4.5, one is presented with the somewhat counter-intuitive result that all models perform worse with data augmentation. It is especially puzzling that the models are not able to take advantage of data augmentation in the evaluation of single-trained models on the largest data set.

Theoretically, the data augmentation should make most of an impact on that specific case.

It could be proposed that increasing or decreasing the aggressiveness of the augmentation would lead to better results. Additionally, the introduction of other augmentation methods such as intensity jittering, random cropping or noise addition could possibly have made the data augmentation more effective. However, because of time constraints, this could not be tested as part of the thesis.

5.5 Insights from the Training Process

In general, because of the magnitude of the proposed models, training times per epoch were long, and the convergence rate was slow. This made hyperparameter tuning a slow and cumbersome process. The transformer architectures, in particular, proved to be really sensitive to initial parameters and learning rate. For a greater part of the early experimentation, it was hard to make both the nnFormer and swin transformer converge at all. During the experiments detailed in Section 3.3.5, the transformer models failed to converge until the learning rate came close to 0.0001. Even though better parameters were found, the transformer models seemed to stop improving earlier in training than the CNNs. The convolutional neural networks were more forgiving in terms of parameters and converged within a larger interval of tested learning rates.

As mentioned in Section 3.3.1, the author initially wanted to train using 16-FP half-precision values to utilise an increased batch size. It is relevant to Research Question 4 that due to the increased memory requirements of using the whole volume as input, batch sizes were lower than they would be in a corresponding two-dimensional case. An increased batch size would lead to faster training epochs and possibly better generalisation. However, this caused too many numerical instabilities. Adding gradient clipping to prevent exploding gradients were tested. However, eventually, the gradients would turn into NaN-values. Training with 32-FP full precision limited the batch size to 32 for the CNNs, and 16 and 8 for the nnFormer and swin transformer, respectively.

5.6 Evaluating Final Performance

The evaluation results of the final models are presented in Table 4.7. From the data, it is clear that the best-performing architecture on average is ResNet-50. This could be expected, seeing as ResNet is one of the most widely used backbones for computer vision tasks. Swin Transformer performs slightly better than nnFormer, which in turn performs better than HeartNet. This means that all proposed methods of this study perform better than the baseline from the preliminary project.

Research Question 1 will have to be answered based on what criteria there are for what is considered sufficient accuracy. If the criterion is for the method to

perform better than the baseline model HeartNet, then all these networks could be said to fulfil the research question. If the task is to approximately locate the aortic valve, then this is also a success, as the network is able to locate the aortic valve in all recordings of the test set. However, if the margin of error is less than 15mm, then none of these methods can be used as a fully automated localisation method. As these models perform bounding box detection, as opposed to single coordinate localisation, it is difficult to compare them directly to results from the literature. However, when Wester [43] was able to detect anatomical landmarks with a mean error of 8.78mm, it was concluded that this was not sufficient for fully automated image registration. That might mean that the methods proposed in this thesis are not sufficiently accurate for registration purposes.

From the experiments conducted with the available data set of this study, it is clear that ResNet outperforms both transformer-based architectures. As stated earlier, this might be a result of insufficient data augmentation or a lacking data set. Based on the evidence presented in this study alone, one can answer Research Question 2, that CNNs still seem to perform the best for applications with limited data sets. However, given the right circumstance, one might be able to make transformer architectures perform comparatively to CNNs.

5.6.1 Inference Time

As an answer to Research Question 4, it is found that using the entire 3D volume as input, also means that the pre-processing pipeline has to handle more data. By virtue of this, the pre-processing also takes more time, lowering the inference rate of the models. If inference rate is a significant criterion for the current application, then it might be better to use other forms of input representations, such as proposed by Wester [43] or Taskén [44].

Inference time can also be used to discuss Research Question 2. Because of the increased complexity of the transformer models compared to the CNNs, they all suffer from slower inference rates. If inference rate is a relevant metric for the wanted application, then the choice of architecture is in favour of a CNN based model.

5.6.2 Error distribution of test set

Averages might not always give a good evaluation of model performance. An alternate visualisation is the error distribution across the test set, as shown in Figure 5.1 and Figure 5.2, for the single frame dataset and the full data set respectively.

From Figure 5.1 it is clear that performance varies a lot across different recordings. The outliers are found to be recordings of bad image quality or artefacts. Figure 5.2 shows that there are also variations in performances within different frames of the same recording. Generally, it seems as if the middle frames of the recordings give worse results than the first and last frames. This might be because

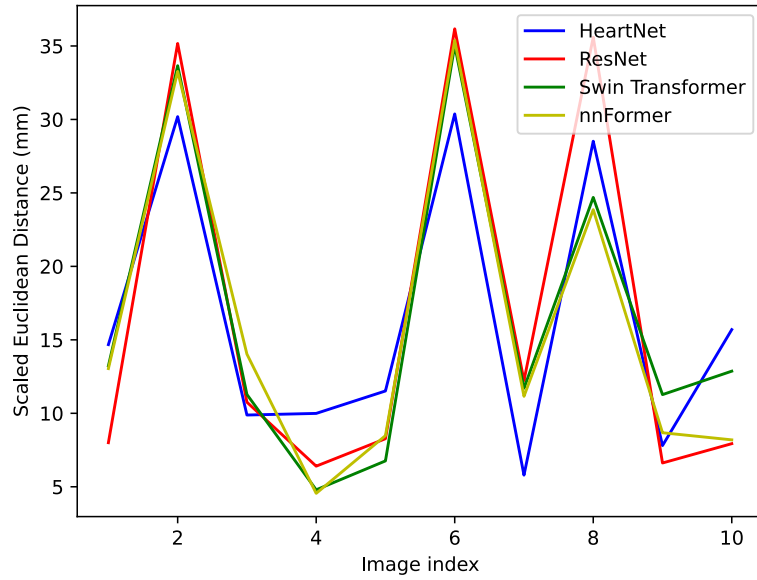


Figure 5.1: Final scores of all models, on all first frames of the test set. The metric used is scaled euclidean distance.

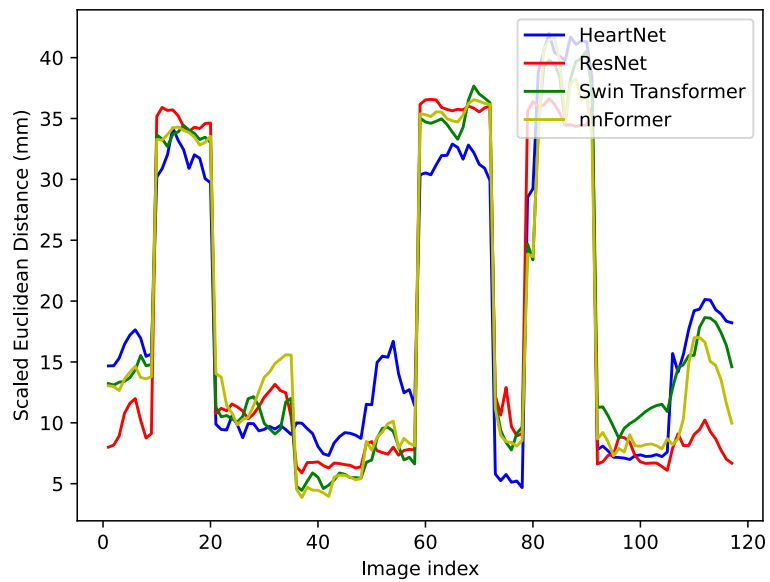


Figure 5.2: Final scores of all models, on all frames of the full test set. The metric used is scaled euclidean distance.

when the aortic valve is in the open position, it is at a greater elevation than in the closed position and, therefore, harder to see.

5.7 Limitations

The previous sections present some of the limits of the thesis. The most important of them have to do with data. It can be argued that the size of the available data set is a significant limitation. All models, but especially the transformers, should perform better with a larger data set. It would also be optimal to not discard any of the initial data set because of quality issues.

The annotations might also pose as a limitation. Their accuracy is limited by the author not being a medical professional and the fact that they are axis-aligned bounding boxes. The annotations for one frame also count for an entire recording.

Using the whole 3D volume as input also puts a hard limit on batch size and input size, because of memory requirements. Both larger input- and batch size might have led to better performance.

Time is also a limiting factor. Due to the rapid pace of machine learning research, there are a plethora of machine learning methods to evaluate. However, because of the limited time allotted for this thesis, the author was only able to evaluate one representative model from each ANN type. This limits the scope and the study's ability to make general conclusions regarding CNNs against transformers.

A general attribute of newer machine learning methods is that they have increased parameter counts compared to older methods. Both transformer-based methods proposed in this study are magnitudes larger than ResNet, which is an older method. Even though a powerful graphics processing unit (GPU) was made available to the author for the period of the thesis work, most novel machine learning models are trained using multi-GPU clusters in the cloud. With increased computing power, it would be easier to evaluate more models or train larger variants of the proposed ones.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, three deep learning methods for automated detection of the aortic valve in 3D transesophageal echocardiography volumes are presented. The proposed models represent CNNs, transformers, and a hybrid of the two. The application of the transformer and hybrid architecture on TEE data is novel. The different methods are 3D variants of ResNet-50, Swin transformer, and nnFormer respectively.

To answer Research Question 2 and Research Question 3, the proposed methods are evaluated against one another and a baseline method from the author's preliminary project. All three methods outperform the preliminary project method; however, ResNet-50 is the most accurate. The mean errors of HeartNet, ResNet, Swin Transformer and the nnFormer on the test set were 18.65mm, 17.40mm, 18.15mm and 18.52mm respectively. The inference times, given in the same order, were 9.75, 7.18, 7.08 and 7.17 frames per second. Answering Research Question 1, the obtained models might provide sufficient accuracy depending on the use case of the localiser. Due to the wide bounding box annotations, as discussed in Section 5.2, real-life network performance may be better than reported numerically.

It is found that the transformer and hybrid architectures are generally more sensitive to the learning rate, while ResNet is more resilient to different hyperparameters and, therefore, easier to train.

Per Research Question 4, all proposed methods use the whole 3D volume as input, as opposed to previous methods, which often are patch-based or two-dimensional. The study proves that this is possible. However, this increases hardware requirements and lowers the batch size and resolution which models can be trained with. Inference times are also slower because of applying pre-processing on more image data.

It is concluded that for this specific use case, and considering the limitations of the study, a convolutional neural network, represented by ResNet-50, would provide the best alternative as a landmark localiser.

Depending on the use case, the overall research goal of the study is considered to be met. However, more research could be done to improve the obtained results and compare the different network types on a deeper level.

6.2 Future Work

Recent innovations within CNNs would be a valuable topic for further research. Since ResNet is a widely used but older architecture, detection accuracy could most likely be increased by utilising newer CNNs.

Alternate input representations could also be a research topic. Using the entire 3D volume gives the network much information. However, there might be smarter and less demanding representations that would enable a larger batch size and faster inference time. Faster inference times would be particularly beneficial for intraoperative use cases. A possible candidate could be to represent the input data as three orthogonal slices, as done by Huang *et al.* [62].

In this study, Smooth L1 loss was found to be a well-performing loss function for bounding box detection. However, research into other alternatives might lead to better training performance. An example of this is to utilise IoU as a loss metric [63].

To improve accuracy of the currently proposed methods, it might be beneficial to experiment with alternate detection heads or even change the type of detection from bounding boxes to single coordinate landmarks. Since much previous work has been done within single coordinate localisation, this would give grounds for better evaluations against existing work.

Since transformers have shown exemplary performance in other tasks, they should still be researched as alternatives to CNNs. It would be relevant to research different approaches to data augmentation in more depth to see if this could better facilitate transformer training. Larger transformer models are also often pre-trained utilising self-supervised learning [64]. A self-supervised approach could help in solving the data scarcity problem for this application.

Bibliography

- [1] W. H. Organization, *Cardiovascular diseases (cvds)*, 2021. [Online]. Available: [https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-\(cvds\)](https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)) (visited on 11/06/2022).
- [2] F. Guarracino and R. Baldassari, 'Transesophageal echocardiography in the or and icu,' *Minerva Anestesiologica*, vol. 75, pp. 518–529, 9 2009.
- [3] A. F. Zierer, J. Kammler, H. Blessberger, M. Ay and C. Steinwender, 'First in man: Off-pump transapical transcatheter aortic valve implantation and mitral valve repair,' *The Annals of Thoracic Surgery*, vol. 107, no. 4, e249–e250, 2019, ISSN: 0003-4975. DOI: <https://doi.org/10.1016/j.athoracsur.2018.07.062>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0003497518313109>.
- [4] K. He, X. Zhang, S. Ren and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, 2015. DOI: 10.48550/ARXIV.1502.01852. [Online]. Available: <https://arxiv.org/abs/1502.01852>.
- [5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2020. DOI: 10.48550/ARXIV.2010.11929. [Online]. Available: <https://arxiv.org/abs/2010.11929>.
- [6] Wapcaplet, *Diagram of the human heart*, Distributed on wikimedia commons under the CC BY-SA 3.0 lisenche, 2006. [Online]. Available: [https://en.wikipedia.org/wiki/File:Diagram_of_the_human_heart_\(cropped\).svg#file](https://en.wikipedia.org/wiki/File:Diagram_of_the_human_heart_(cropped).svg#file) (visited on 16/04/2022).
- [7] J. Pollock and A. Makaryus, *Physiology, Cardiac Cycle*. StatPearls Publishing, 2022. [Online]. Available: <https://www.ncbi.nlm.nih.gov/books/NBK459327/>.
- [8] adh30, *Wiggers diagram*, Distributed on wikimedia commons under the CC BY-SA 4.0 lisenche. adh30 revised work by DanielChangMD who revised original work of DestinyQx; Redrawn as SVG by xavax, 2016. [Online]. Available: https://commons.wikimedia.org/wiki/File:Wiggers_Diagram_2.svg (visited on 14/05/2022).

- [9] C. Kenny and M. Monaghan, 'How to assess aortic annular size before transcatheter aortic valve implantation (tavi): The role of echocardiography compared with other imaging modalities,' *Heart*, vol. 101, no. 9, pp. 727–736, 2015, ISSN: 1355-6037. DOI: 10.1136/heartjnl-2013-304689. eprint: <https://heart.bmj.com/content/101/9/727.full.pdf>. [Online]. Available: <https://heart.bmj.com/content/101/9/727>.
- [10] S. Y. Ho, 'Structure and anatomy of the aortic root,' *European Journal of Echocardiography*, vol. 10, no. 1, pp. i3–i10, Jan. 2009, ISSN: 1525-2167. DOI: 10.1093/ejehocard/jen243. [Online]. Available: <https://doi.org/10.1093/ejehocard/jen243>.
- [11] P. Nagpal, M. D. Agrawal, S. S. Saboo, S. Hedgire, S. Priya and M. L. Steigner, 'Imaging of the aortic root on high-pitch non-gated and ecg-gated ct: Awareness is the key,' *Insights into Imaging*, p. 51, 2020. DOI: 10.1186/s13244-020-00855-w. [Online]. Available: <https://insightsimaging.springeropen.com/articles/10.1186/s13244-020-00855-w>.
- [12] V. Chan and A. Perlas, 'Basics of ultrasound imaging,' in *Atlas of Ultrasound-Guided Procedures in Interventional Pain Management*, S. N. Narouze, Ed. New York, NY: Springer New York, 2011, pp. 13–19, ISBN: 978-1-4419-1681-5. DOI: 10.1007/978-1-4419-1681-5_2. [Online]. Available: https://doi.org/10.1007/978-1-4419-1681-5_2.
- [13] U. H. N. 2020, *Tee: 3d standard views web tool*, 2020. [Online]. Available: https://pie.med.utoronto.ca/TEE/TEE_content/assets/applications/standardViewsHTML5/TEE-HTML5-3D/index.html (visited on 28/10/2021).
- [14] National Heart Lung and Blood Institute (NIH), *Transesophageal echo*, Distributed on wikimedia commons in the public domain., 2013. [Online]. Available: https://commons.wikimedia.org/wiki/File:Transesophageal_echo.jpg (visited on 18/05/2022).
- [15] R. T. Hahn, T. Abraham, M. S. Adams, C. J. Bruce, K. E. Glas, R. M. Lang, S. T. Reeves, J. S. Shanewise, S. C. Siu, W. Stewart and M. H. Picard, 'Guidelines for performing a comprehensive transesophageal echocardiographic examination: Recommendations from the american society of echocardiography and the society of cardiovascular anesthesiologists,' *Journal of the American Society of Echocardiography*, vol. 26, pp. 921–964, 9 Sep. 2013, ISSN: 08947317. DOI: 10.1016/j.echo.2013.07.009.
- [16] F. Rosenblatt, 'The perceptron: A probabilistic model for information storage and organization in the brain.,' *Psychological review*, vol. 65, pp. 386–408, 1958.
- [17] W. S. McCulloch and W. Pitts, 'A logical calculus of the ideas immanent in nervous activity,' *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943. DOI: 10.1007/BF02478259. [Online]. Available: <https://link.springer.com/article/10.1007/BF02478259>.

- [18] Chrislb, *Diagram of an artificial neuron*, Distributed on wikimedia commons under the CC BY-SA 3.0 licence, 2005. [Online]. Available: https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_english.png (visited on 27/05/2022).
- [19] L. Vu-Quoc, *Neuron and myelinated axon, with signal flow from inputs at dendrites to outputs at axon terminals*, Distributed on wikimedia commons under the CC BY-SA 3.0 licence, 2018. [Online]. Available: <https://commons.wikimedia.org/wiki/File:Neuron3.svg> (visited on 27/05/2022).
- [20] Offnfopt, *Illustration of an artificial feed-forward neural network*, Distributed on wikimedia commons under the CC BY-SA 3.0 licence, 2015. [Online]. Available: https://commons.wikimedia.org/wiki/File:Multi-Layer_Neural_Network-Vector-Blank.svg (visited on 27/05/2022).
- [21] V. Nair and G. E. Hinton, 'Rectified linear units improve restricted boltzmann machines,' in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel: Omnipress, 2010, pp. 807–814, ISBN: 9781605589077.
- [22] K. Jarrett, K. Kavukcuoglu, M. Ranzato and Y. LeCun, 'What is the best multi-stage architecture for object recognition?' In *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 2146–2153. DOI: 10.1109/ICCV.2009.5459469.
- [23] P. Ramachandran, B. Zoph and Q. V. Le, 'Searching for activation functions,' *CoRR*, vol. abs/1710.05941, 2017. arXiv: 1710.05941. [Online]. Available: <http://arxiv.org/abs/1710.05941>.
- [24] A. Amini, *Gradient descent*, Originally published under MIT license by A. Amini; Adapted by M. Atarod for use in Science Magazine. [Online]. Available: <https://www.science.org/content/article/ai-researchers-allege-machine-learning-alchemy> (visited on 28/05/2022).
- [25] L. A. Cauchy, 'Méthode générale pour la résolution des systèmes d'équations simultanées,' *Compte Rendu à l'Académie des Sciences*, vol. 25, pp. 536–538, 1847.
- [26] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors,' *Nature*, vol. 323, pp. 533–536, 1986. DOI: 10.1038/323533a0.
- [27] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].
- [28] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. DOI: 10.48550/ARXIV.1502.03167. [Online]. Available: <https://arxiv.org/abs/1502.03167>.
- [29] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, 'Gradient-based learning applied to document recognition,' *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

- [30] A. Krizhevsky, I. Sutskever and G. E. Hinton, ‘Imagenet classification with deep convolutional neural networks,’ in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS’12, Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.
- [31] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama and J. Sakuma, ‘Malware analysis of imaged binary samples by convolutional neural network with attention mechanism,’ Mar. 2018, pp. 127–134. DOI: 10.1145/3176258.3176335.
- [32] A. L. Maas, A. Y. Hannum and A. Y. Ng, ‘Rectifier nonlinearities improve neural network acoustic models,’ in *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.
- [33] F-F Li, J. Wu and R. Gao, *Cs231n: Deep learning for computer vision course notes: Convolutional neural networks for visual recognition*. 2022. [Online]. Available: <https://cs231n.github.io/convolutional-networks/> (visited on 31/05/2022).
- [34] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385. [Online]. Available: <https://arxiv.org/abs/1512.03385>.
- [35] K. He and J. Sun, *Convolutional neural networks at constrained time cost*, 2014. DOI: 10.48550/ARXIV.1412.1710. [Online]. Available: <https://arxiv.org/abs/1412.1710>.
- [36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Attention is all you need*, 2017. DOI: 10.48550/ARXIV.1706.03762. [Online]. Available: <https://arxiv.org/abs/1706.03762>.
- [37] Y. Liu, Y. Zhang, Y. Wang, F. Hou, J. Yuan, J. Tian, Y. Zhang, Z. Shi, J. Fan and Z. He, *A survey of visual transformers*, 2021. arXiv: 2111.06091 [cs.CV].
- [38] D. Hendrycks and K. Gimpel, *Gaussian error linear units (gelus)*, 2016. DOI: 10.48550/ARXIV.1606.08415. [Online]. Available: <https://arxiv.org/abs/1606.08415>.
- [39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ‘ImageNet: A Large-Scale Hierarchical Image Database,’ in *CVPR09*, 2009.
- [40] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, ‘Microsoft COCO: common objects in context,’ *CoRR*, vol. abs/1405.0312, 2014. arXiv: 1405.0312. [Online]. Available: <http://arxiv.org/abs/1405.0312>.
- [41] X. Shen, ‘A survey of object classification and detection based on 2d/3d data,’ *CoRR*, vol. abs/1905.12683, 2019. arXiv: 1905.12683. [Online]. Available: <http://arxiv.org/abs/1905.12683>.

- [42] J. M. H. Noothout, B. D. de Vos, J. M. Wolterink, T. Leiner and I. Isgum, 'Cnn-based landmark detection in cardiac CTA scans,' *CoRR*, vol. abs/1804.04963, 2018. arXiv: 1804.04963. [Online]. Available: <http://arxiv.org/abs/1804.04963>.
- [43] B. H. Wester, 'Detecting anatomical landmarks in 3d cardiovascular images using convolutional neural networks,' M.S. thesis, Department of Informatics, University of Oslo, Oslo, 2020.
- [44] A. A. Taskén, 'Automated segmental cardiac monitoring by advanced computerized artificial intelligence on intra-operative three-dimensional ultrasound recordings,' M.S. thesis, Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, Trondheim, 2021.
- [45] T. Nordal, 'Automatic detection of mitral annular plane systolic excursion from transesophageal echocardiography using deep learning,' M.S. thesis, Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, Trondheim, 2019.
- [46] M. H. bin Ahmad Nizar, C. K. Chan, A. K. M. Yusof, A. Khalil and K. W. Lai, 'Detection of aortic valve from echocardiography in real-time using convolutional neural network,' in *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, 2018, pp. 91–95. DOI: 10.1109/IECBES.2018.8626735.
- [47] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, 'SSD: Single shot MultiBox detector,' in *Computer Vision – ECCV 2016*, Springer International Publishing, 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2. [Online]. Available: https://doi.org/10.1007%2F978-3-319-46448-0_2.
- [48] R. Girshick, J. Donahue, T. Darrell and J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, 2013. DOI: 10.48550/ARXIV.1311.2524. [Online]. Available: <https://arxiv.org/abs/1311.2524>.
- [49] O. Ronneberger, P. Fischer and T. Brox, 'U-net: Convolutional networks for biomedical image segmentation,' *CoRR*, vol. abs/1505.04597, 2015. arXiv: 1505.04597. [Online]. Available: <http://arxiv.org/abs/1505.04597>.
- [50] A. Hatamizadeh, Y. Tang, V. Nath, D. Yang, A. Myronenko, B. Landman, H. Roth and D. Xu, *Unetr: Transformers for 3d medical image segmentation*, 2021. DOI: 10.48550/ARXIV.2103.10504. [Online]. Available: <https://arxiv.org/abs/2103.10504>.
- [51] A. Hatamizadeh, V. Nath, Y. Tang, D. Yang, H. Roth and D. Xu, *Swin unetr: Swin transformers for semantic segmentation of brain tumors in mri images*, 2022. DOI: 10.48550/ARXIV.2201.01266. [Online]. Available: <https://arxiv.org/abs/2201.01266>.

- [52] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin and B. Guo, *Swin transformer: Hierarchical vision transformer using shifted windows*, 2021. DOI: 10.48550/ARXIV.2103.14030. [Online]. Available: <https://arxiv.org/abs/2103.14030>.
- [53] H.-Y. Zhou, J. Guo, Y. Zhang, L. Yu, L. Wang and Y. Yu, *Nnformer: Interleaved transformer for volumetric segmentation*, 2021. DOI: 10.48550/ARXIV.2109.03201. [Online]. Available: <https://arxiv.org/abs/2109.03201>.
- [54] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015. arXiv: 1409.1556 [cs.CV].
- [55] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin and H. Hu, *Video swin transformer*, 2021. DOI: 10.48550/ARXIV.2106.13230. [Online]. Available: <https://arxiv.org/abs/2106.13230>.
- [56] R. B. Girshick, 'Fast R-CNN,' *CoRR*, vol. abs/1504.08083, 2015. arXiv: 1504.08083. [Online]. Available: <http://arxiv.org/abs/1504.08083>.
- [57] I. Loshchilov and F. Hutter, 'Fixing weight decay regularization in adam,' *CoRR*, vol. abs/1711.05101, 2017. arXiv: 1711.05101. [Online]. Available: <http://arxiv.org/abs/1711.05101>.
- [58] I. Loshchilov and F. Hutter, *Sgdr: Stochastic gradient descent with warm restarts*, 2016. DOI: 10.48550/ARXIV.1608.03983. [Online]. Available: <https://arxiv.org/abs/1608.03983>.
- [59] P. Xu, D. Kumar, W. Yang, W. Zi, K. Tang, C. Huang, J. C. K. Cheung, S. J. D. Prince and Y. Cao, *Optimizing deeper transformers on small datasets*, 2020. DOI: 10.48550/ARXIV.2012.15355. [Online]. Available: <https://arxiv.org/abs/2012.15355>.
- [60] B. Landman, Z. Xu, J. E. Igelsias, M. Styner, T. R. Langerak and A. Klein, *2015 MICCAI Multi-Atlas Labeling Beyond the Cranial Vault – Workshop and Challenge*. 2015. DOI: 10.7303/syn3193805. [Online]. Available: <https://www.synapse.org/#!Synapse:syn3193805/wiki/217789>.
- [61] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman and A. Zisserman, *The kinetics human action video dataset*, 2017. DOI: 10.48550/ARXIV.1705.06950. [Online]. Available: <https://arxiv.org/abs/1705.06950>.
- [62] Y. Huang, J. He, X. Wu, X. Zhao and J. Wu, 'Tracking 3d ultrasound anatomical landmarks via three orthogonal plane-based scale discriminative correlation filter network,' *Medical Physics*, vol. 48, 5 May 2021, ISSN: 0094-2405. DOI: 10.1002/mp.14798.
- [63] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai and R. Yang, 'Iou loss for 2d/3d object detection,' in *2019 International Conference on 3D Vision (3DV)*, 2019, pp. 85–94. DOI: 10.1109/3DV.2019.00019.

- [64] Y. Tang, D. Yang, W. Li, H. Roth, B. Landman, D. Xu, V. Nath and A. Hatamizadeh, *Self-supervised pre-training of swin transformers for 3d medical image analysis*, 2021. DOI: 10.48550/ARXIV.2111.14791. [Online]. Available: <https://arxiv.org/abs/2111.14791>.

Appendix A

Additional Material

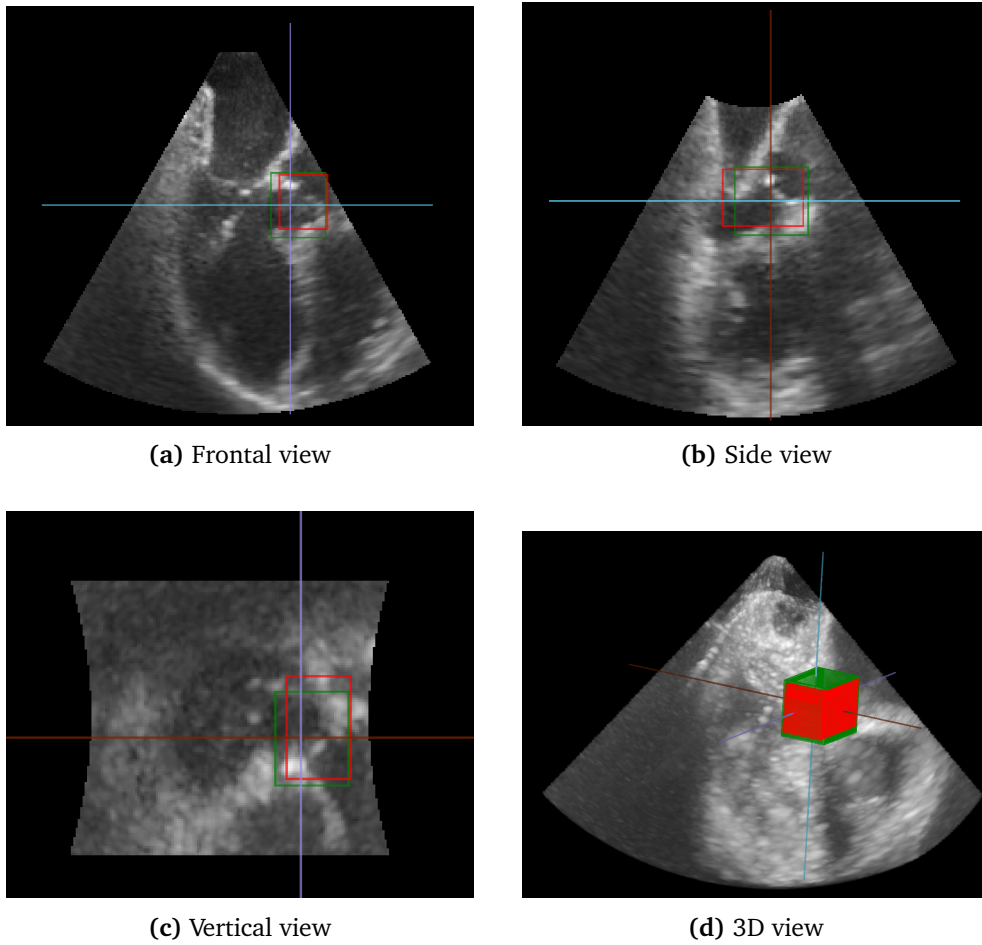


Figure A.1: Visualisation of the best prediction overall. Made by nnFormer.

