UiO : Department of Informatics
University of Oslo

Detecting Anatomical Landmarks in 3D Cardiovascular Images Using Convolutional Neural Networks

Betina Høyer Wester

Master's Thesis, Spring 2020

Abstract

Medical imaging enables us to visualize the interior of the body. Traditionally, medical images have been analyzed by doctors, but lately, methods for extracting information automatically from medical images have been explored. Automatical feature extraction is time-saving and makes medical tools more accessible. Convolutional neural networks have proven to be a powerful tool in several medical image processing tasks, with the potential to exceed human performance. In this thesis, we have explored the use of convolutional neural networks for landmark detection in 3D cardiovascular images. Landmark detection in 3D images is useful to provide automatic registration between ultrasound and CT images of the same patient. A patch-based convolutional neural network was used. Two types of network architectures were tested; ResNet18 and a fully convolutional neural network. To improve the performance of the networks, multi-task learning with classification as a secondary task was combined with the landmark detection. In addition, a weighted loss function was applied. The results from the classification were later used to determine the final landmark prediction. The model providing the lowest euclidean error was ResNet18. After post-processing, the average error was 8.78 mm. One application of this method is GE Vingemed's CT-fusion tool. The results were found to provide acceptable accuracy for a semi-automatic landmark detection model.

Acknowledgements

I would like to express my very great appreciation to Eigil Samset for his valuable and constructive suggestions during the development of this thesis. I would also like to express my deep gratitude to Andrew Gilbert, for his patient guidance and useful critiques. You have taught me so much this last year. Thanks to Federico Veronesi for data and guidance in the CT-Fusion tool and to Børge Solli Andreassen for helping me with the visualization.

Finally, I wish to thank everyone who made my 5 years at the University of Oslo memorable, my friends and family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

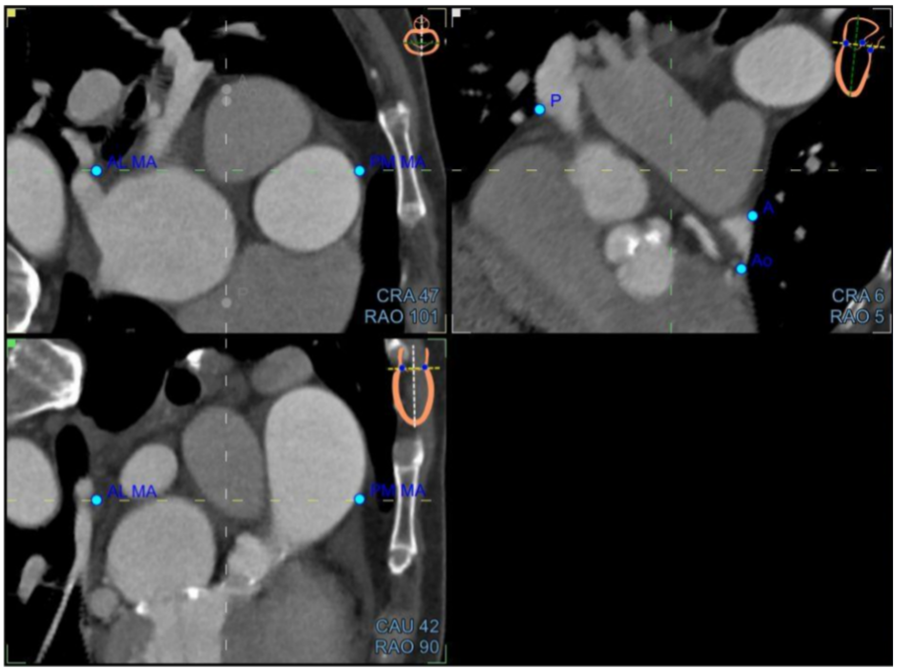
1.1 Motivation

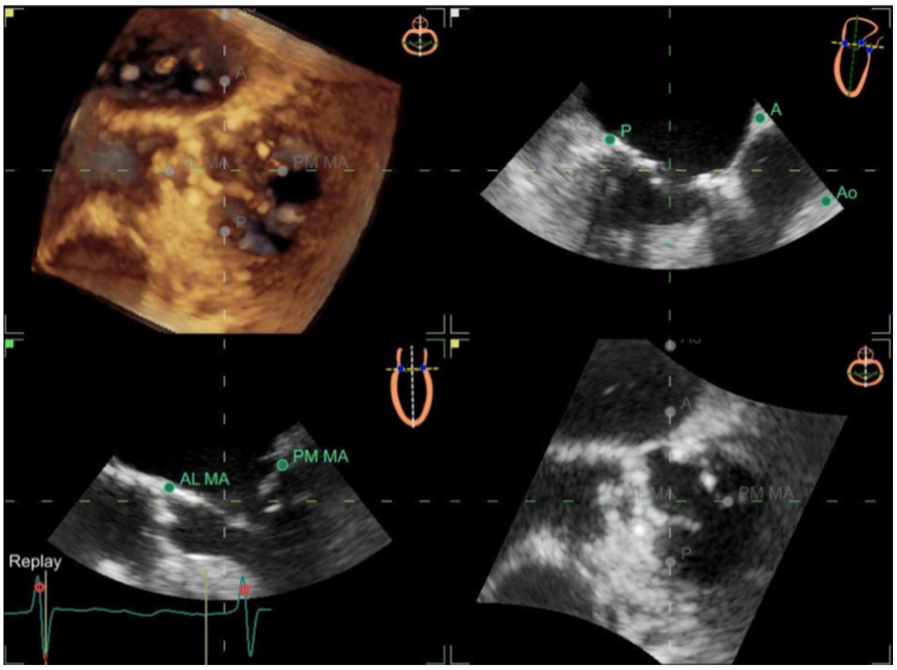
According to the World Health Organization, cardiovascular diseases are the number one cause of deaths worldwide. By detecting these kinds of diseases at an early stage, we can start treatments to prevent them from going worse [8]. With echocardiography, we can easily examine the size, structure, blood flow and movement of various parts of the heart, including the heart valves, walls and chambers [24]. Doctors can use information given from echocardiography to detect and diagnose a variety of cardiovascular diseases [33]. The heart size can indicate if a person has high blood pressure, leaky heart valves, or heart failure. The thickness of the wall can be used to discover valve diseases and congenital heart defects [19], and the movement of the heart can tell whether any of the heart valves do not open or close normally. We are also able to detect congenital heart defects, such as holes in the heart, by looking at the structure of the heart.

Convolutional neural networks have become a popular tool for image classification, segmentation, and other tasks. Using convolutional neural networks for computer vision tasks have been studied since the 1960s, and the concept of neural networks was inspired by how neurons in the cortex communicated with each other after receiving stimulation. In 1999, researches started using convolutional networks for feature-based object recognition, and the first algorithm for face recognition was introduced [31]. From 2005, the Pascal VOC project started, which provided a dataset for image classification and object detection. The intention was to establish a benchmark for investigating the performance of recognition methods [9]. As a result of this, several state-of-the-art network architectures for classification and object detection were introduced in the following years, each better than the other. In 2015, the 152 layer deep residual network was introduced, which exceeded the human level in image classification. As scientists were able to increase the accuracy of the networks, the exploration of convolutional neural networks for medical applications started. In recent years, we have learned that CNNs can be used to perform time-consuming tasks that until now have only been performed by professionals [12]. For example have methods for detecting and segmenting different types of cancer been developed. Convolutional neural networks are promising for automatic, fast and accurate medical image analysis, which have the potential to outperform experts [32].

Our goal was to use CNNs to detect six anatomical landmarks in 3D cardiovascular images. These landmarks could, in turn, be used to automate workflows related to landmark-based image registration, such as the CT-fusion tool made by GE Vingmed Ultrasound. In the CT-fusion tool, the six anatomical landmarks in the ultrasound image, together with the six corresponding landmarks in CT scans, are used to merge the ultrasound image and CT scan to one image. The new image contain more detailed information, as shown in figure 1.1.1. The CT-fusion tool can be used to combine pre-operative CT with intra-operative echo during complex cardiac interventions. Furthermore, the two imaging techniques complete each other, as CT is able to visualize structures not easily seen in echo, and echo provides real-time 3D modality that can be used to guide positioning and placement of devices during an operation. The landmark registration task is currently manual and may need to be repeated many times. The advantage of automatic landmark detection is that this is less time consuming than finding the landmarks manually, and we can be sure that the same landmarks are detected each time.

Applying convolutional neural networks on medical 3D data is challenging due to anatomical variation among patients and differences in image acquisition [21]. We will explore to what extent deep learning handles these problems and how well they will perform in our problem. We will also look at how applying multi-task learning can improve the performance of neural networks.


AL MA
PM MA
CRA 47
RAO 101
P
A
Ao
CRA 6
RAO 5
AL MA
PM MA
CAU 42
RAO 90


A
AL MA
PM MA
P
P
A
Ao
AL MA
PM MA
Replay
A
AL MA
PM MA
P

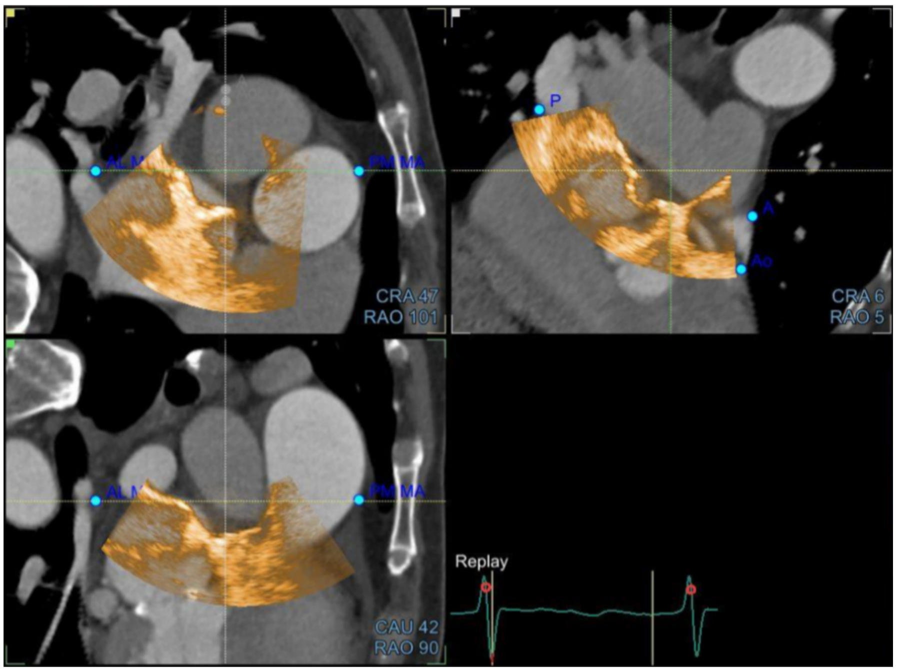


Figure 1.1.1: GE Vingmed Ultrasound CT fusion tool

1.2 Related work

In "Detecting Anatomical Landmarks From Limited Medical Imaging Data Using Two-Stage Task-Oriented Deep Neural Networks," different deep learning methods and network architectures for detecting anatomical landmarks in 3D MR images and CT scans were compared. Due to limitations in some of the methods, two data sets were used. The first data set consists of 500 MR-images of the brain labeled with 1200 anatomical landmarks, and the second dataset consists of 73 CT scans of prostate labeled with seven landmarks. Both data sets are 3D images. Samples from the data are shown in figure 1.2.2.

In the paper, six different methods for landmark detection was tested and compared. Some of the methods use whole images as input, while other methods use only parts of the images as input. The methods which use parts of the images, use extracted patches as input and are called patch-based methods. The methods that were tested are multi-atlas (MA), random forest (RF) regression, shallow convolutional network (shallow-net), U-net, and a new method proposed by the authors. MA is a patch-based method and returns displacement vectors between the extracted patch and the landmarks. RF, shallow-net and U-net use whole im-

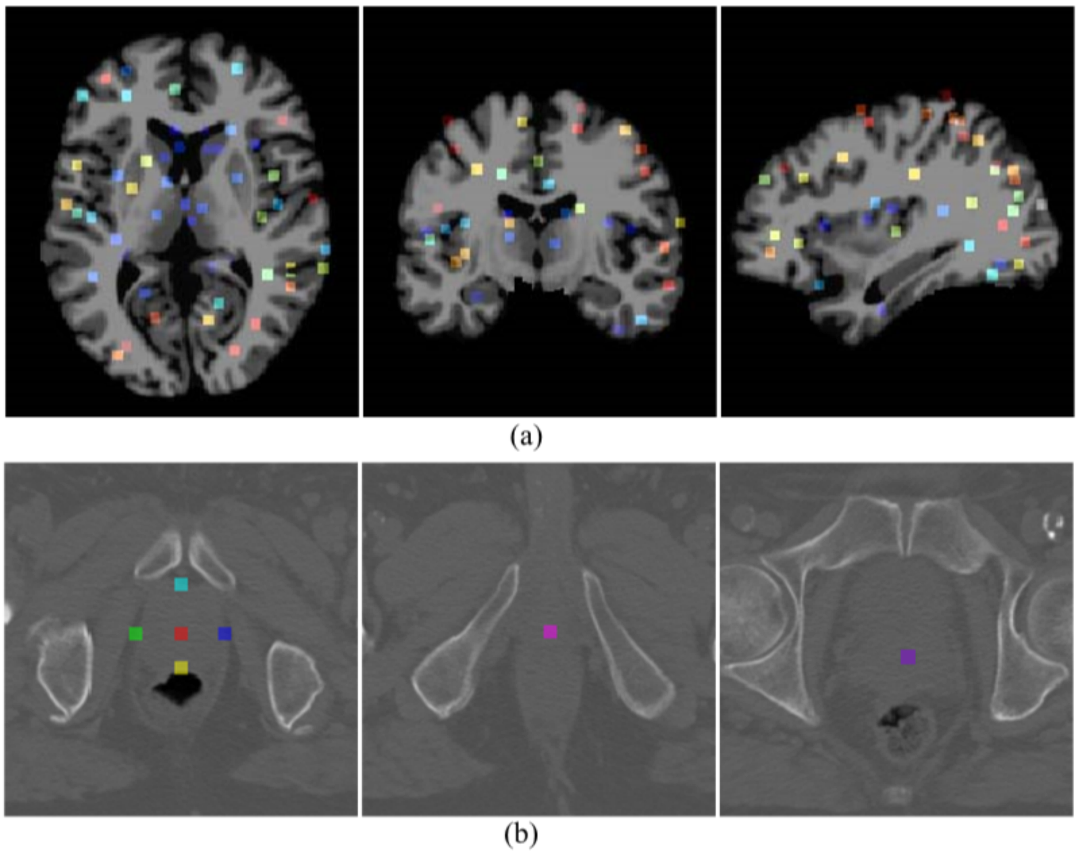


Figure 1.2.2: Anatomical landmarks from the two data sets. (a) is brain MR dataset and (b) is prostate CT dataset

ages as input. RF and shallow-net return the coordinates to the landmarks, while U-net returns one heat-map for each landmark, representing their most-likely coordinates. The method proposed by the authors is a two-staged task-oriented deep neural network (T^{2}DL). This network consists of two sub-networks. The first network (First-stage-only) is a 14-layer convolutional neural network that takes an image patch as input and returns a displacement vector between the patch and each landmark. The second part uses the same weights as the ones trained in the first stage, with seven layers added at the end. The network uses the entire 3D image as input, and the output is the coordinates to each of the landmarks. The network is illustrated in figure 1.2.3. In the patch-based methods, the final landmarks are decided by using a weighted mean from all the suggested landmarks. The results from the tests are shown in table 1.2.4. As some of the methods were not able to detect large scale landmarks, the authors created a new dataset from the brain data set, by selecting 10 random landmarks from the original set. We see

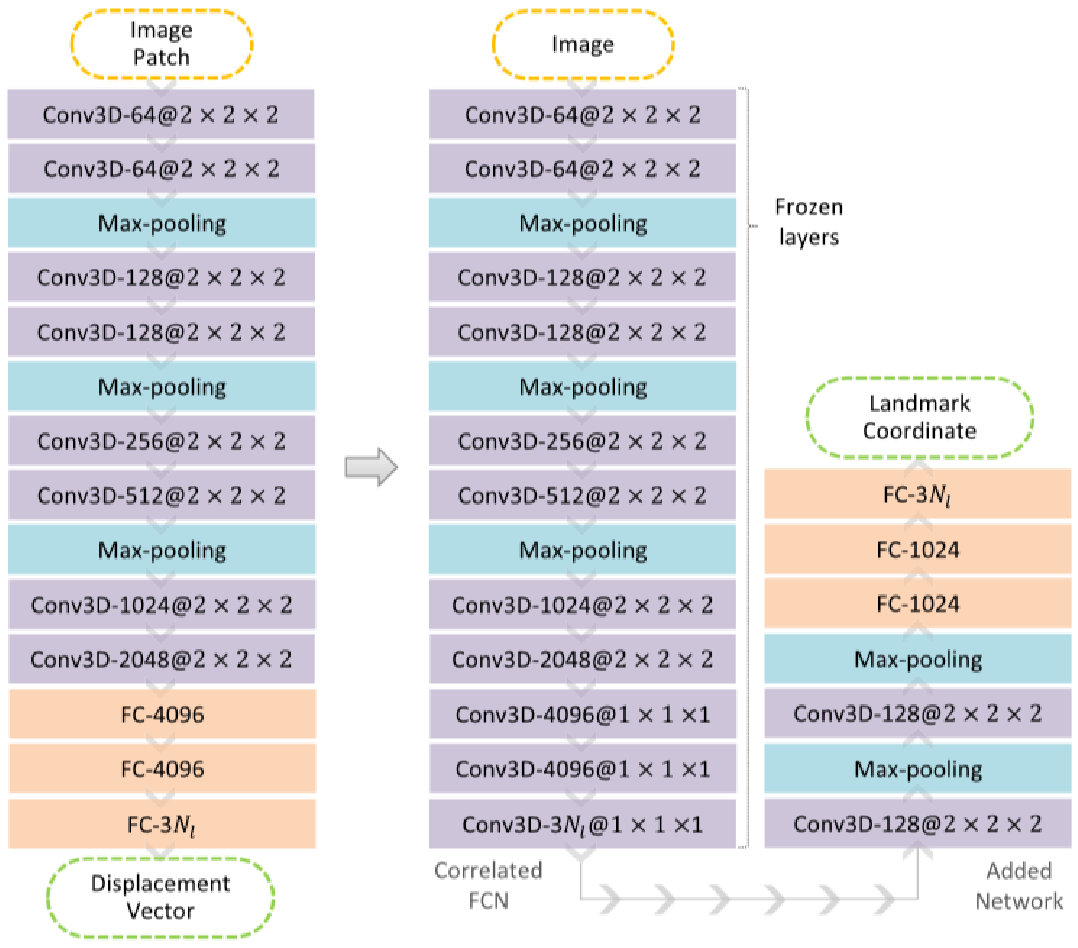
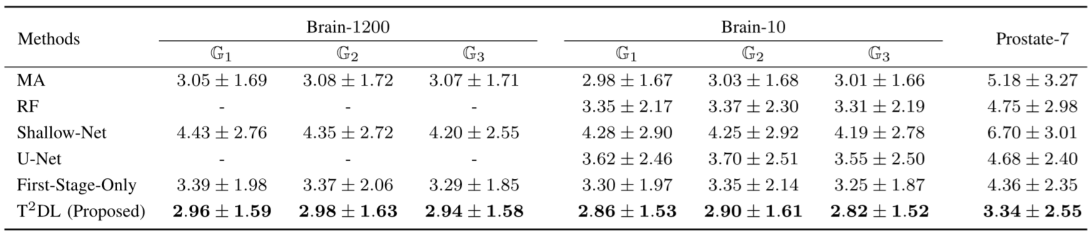


Figure 1.2.3: Illustration of the purposed network in Zhang et al. [33], where (a) is the first stage and (b) is the second stage. Conv3D represents 3D convolutional layers, FC represents fully connected layers

from the results that the landmark detection problem can be solved with various methods and that it is possible to achieve good results, although the data set is limited.

In "CNN-based Landmark Detection in Cardiac CTA Scans", the same patch-based method as in Zhang et al. was used. In this paper, the goal was also to detect anatomical landmarks in medical data. The dataset used in the paper is 198 CTA scans of the heart, labeled with six landmarks. A sample from the data is shown in figure 1.2.5. In the paper, a new method for detecting these six landmarks was proposed. This was done by using a patch-based method with a



Methods	Brain-1200			Brain-10			Prostate-7
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_3$	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_3$	
MA	3.05 ± 1.69	3.08 ± 1.72	3.07 ± 1.71	2.98 ± 1.67	3.03 ± 1.68	3.01 ± 1.66	5.18 ± 3.27
RF	-	-	-	3.35 ± 2.17	3.37 ± 2.30	3.31 ± 2.19	4.75 ± 2.98
Shallow-Net	4.43 ± 2.76	4.35 ± 2.72	4.20 ± 2.55	4.28 ± 2.90	4.25 ± 2.92	4.19 ± 2.78	6.70 ± 3.01
U-Net	-	-	-	3.62 ± 2.46	3.70 ± 2.51	3.55 ± 2.50	4.68 ± 2.40
First-Stage-Only	3.39 ± 1.98	3.37 ± 2.06	3.29 ± 1.85	3.30 ± 1.97	3.35 ± 2.14	3.25 ± 1.87	4.36 ± 2.35
T^{2}DL (Proposed)	$\mathbf{2.96 \pm 1.59}$	$\mathbf{2.98 \pm 1.63}$	$\mathbf{2.94 \pm 1.58}$	$\mathbf{2.86 \pm 1.53}$	$\mathbf{2.90 \pm 1.61}$	$\mathbf{2.82 \pm 1.52}$	$\mathbf{3.34 \pm 2.55}$

Figure 1.2.4: Landmark detection error in brain dataset and prostate dataset (mm)

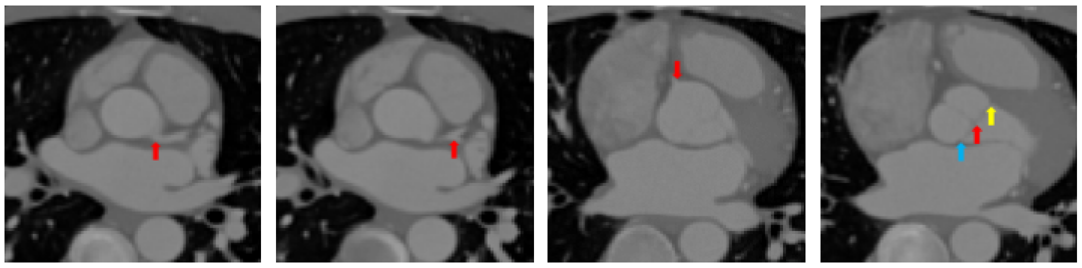


Figure 1.2.5: Samples from the dataset. The arrows are pointing at the different landmarks

fully convolutional neural network. Hence the input of the network are patches extracted from the 3D images. The network use multitask learning and combines regression and classification. As shown in figure 1.2.6, the network consists of six convolutional layers, with max-pooling layers after the three first. After these layers, the network splits into the regression part and the classification part, each consisting of a fully connected layer. The output of the network is displacement vectors between the input patch and the landmarks, and a class indicating whether the patch contains a landmark or not [21]. During training, the loss function to the network was log-transformed, so that patches far away from the landmarks had less influence on the updates of the network during backpropagation. This weighting was added because the author assumed that patches far away from the landmarks would make worse predictions than patches close to landmarks. The networks were tested with and without classification, and with and without log-transform loss function to see their effect. As seen in table 1.1, using classification and log-transform improved the results. The network was trained for 60 000 iterations, with a batch size of 25 patches. The error of each landmark is shown in table 1.2. The article shows how important it is to lower the contribution of the patches that are far away on the loss. Adding classification to the network or log-transform to the loss function or both vastly improve the final results. Using the results from the classification turned out to be useful when predicting the final landmarks from

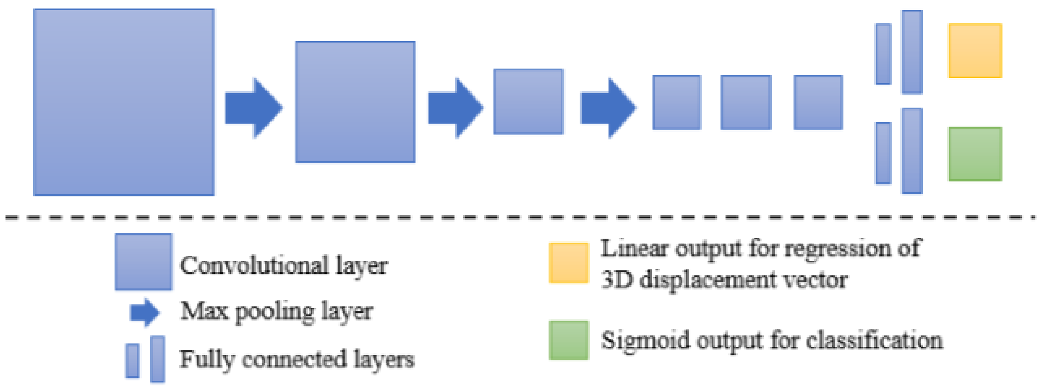

Convolutional layer
Max pooling layer
Fully connected layers
Linear output for regression of 3D displacement vector
Sigmoid output for classification

Figure 1.2.6: Fully convolutional neural network from Noothout et al.

Log-transformed	Classification	Error	Minimum	Maximum
no	no	29.07 ± 6.83	17.3	43.64
yes	no	5.57 ± 3.35	1.32	16.9
no	yes	6.33 ± 2.54	1.43	13.62
yes	yes	2.19 ± 1.97	0.63	12.72

Table 1.1: Average Euclidean distance errors with standard deviations (Error), and the minimum (Minimum) and maximum (Maximum) distance errors expressed in mm. Effect of adding classification and log-transform [21]

Landmark	Error	Minimum	Maximum
Right ostium	2.19 ± 1.97	0.63	12.72
Left ostium	2.88 ± 1.58	0.18	7.02
LM bifurcation	3.76 ± 2.58	0.59	10.83
Right aortic valve commissure	1.82 ± 0.97	0.40	4,56
non-coronary aortic valve commissure	2.10 ± 0.93	0.45	5.62
Left aortic valve commissure	1.89 ± 0.95	0.41	5.28

Table 1.2: Average Euclidean distance errors with standard deviations (Error), and the minimum (Minimum) and maximum (Maximum) distance errors expressed in mm for each landmark

all predicted displacement vectors. Using only patches classified to contain the landmarks to decide the final landmark resulted in accurate predictions.

Applying convolutional neural networks to 3D cardiovascular images has already been tested in "Mitral Annulus Segmentation Using Deep Learning in 3-D Transesophageal Echocardiography" [2]. The goal was to detect the mitral annulus. Like the papers presented earlier, the author had chosen to use parts of the 3D images to feed the model. 2D planes from the original images were extracted by rotating around the z-axis and sent through a U-net. For each slice sent through the network, the U-net returned a heat-map. The heat-map represents where the intersection between the mitral annulus and the plane is most likely to be. The mitral annulus was found by fitting a spline to the predicted points. The method is illustrated in figure 1.2.7, and the final error was only 2 mm. Hence it is possible to get good accuracy applying convolutional neural networks to ultrasound images.

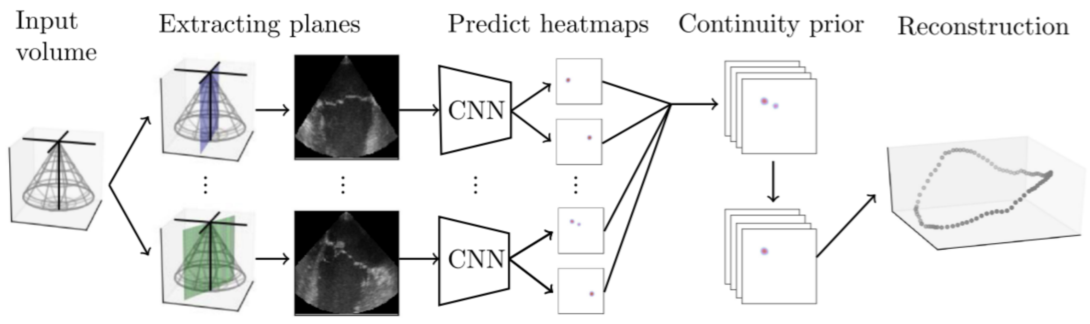


Figure 1.2.7: method used in Andreassen et al. [2]

Chapter 2

Theory

2.1 Feedforward neural networks

Given set of input values x and output values y, the purpose of a feedforward neural network is to estimate the mapping between x and y with a function $\hat{y} = f(x; \Theta)$, so that $\hat{y} \approx y$. The parameters Θ are optimized to give the best approximation of true mapping f [11]. Neural networks are created by combining multiple functions like these, where each function represent a layer:

$$f(x; \theta) = f_3(f_2(f_1(x; \Theta)))$$

f_1 represents the first layer, f_2 the second and f_3 the third. The first layer uses the data x as input, while the input of the next layers are the output from the previous layer. Thus the information is passed through each layer of the network. The first layer is called the input layer, the last layer is called the output layer, and all other layers are called hidden layers. Within each layer, a given number of *neurons* are representing a function of the form:

$$f(x) = Wx + b \tag{2.1}$$

where W is the weight and b is the bias. Hence the parameters that are to be optimized are $\Theta = (W, b)$. After each layer, a non-linear function called *activation function* is added. Without the activation function, all hidden layers collapse to one single linear mapping, from x to y. If the network becomes a linear regression model, it will to learn complex mappings from x to y. We will look more into activation functions later in this chapter. Let $a_k^{[l]}$ denote the activation of a neuron k in layer l given by

$$a_k^l = g\left(\sum_{j=1}^{n^{l-1}} w_{jk}^l a_j^{l-1} + b_k^l\right) \tag{2.2}$$

where w_{jk}^l is the weight from node j in layer l. The network is fully connected if all neurons are connected to all neurons in the previous layer as shown in figure 2.1.1.

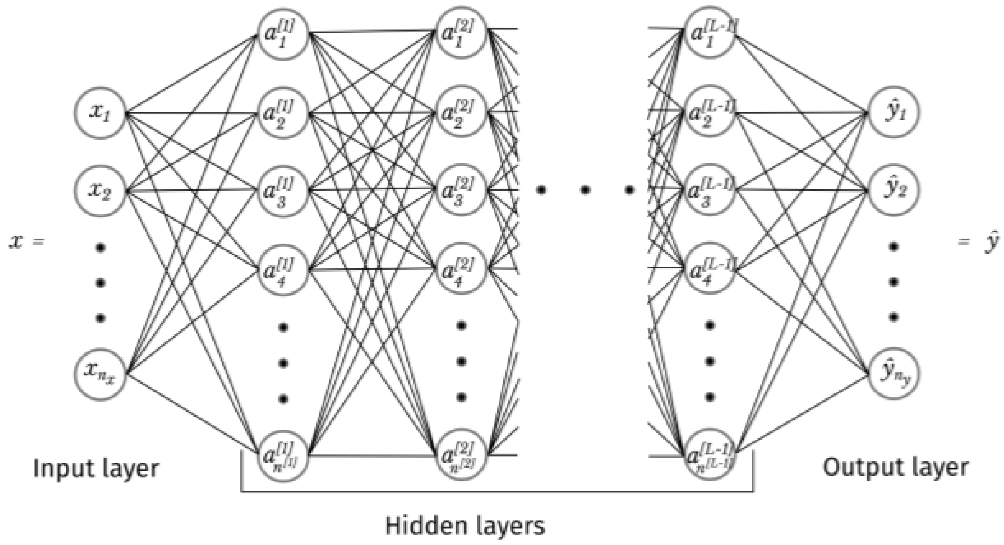


Figure 2.1.1: Visual representation of how neurons in the different layers pass information in a fully connected layer

2.2 Convolutional Neural Networks

In the previous section, we showed how a mapping between input x and output y could be represented as a neural network consisting of several layers, which again consists of neurons. Up until now, only examples where the input x is one dimensional have been considered. However, feedforward neural networks do not handle multi-dimensional data well. Take an image as an example: Before sending an image through a feedforward neural network, the image needs to be transformed into a one-dimensional vector. If the image is an RGB image with 200 pixels in height and width as input, this will result in a vector containing 200x200x3 $=$ 120 000 elements. Each element in the first layer in the feedforward neural network will contain the same amount of parameters as the size of the input. This results in millions of parameters, which are computationally expensive and maybe even impossible to compute.

To solve this, convolutional neural networks (CNN) were introduced. Cnn's have a lot in common with feedforward neural networks. They share the same structure of layers containing neurons with trainable weights, but they use convolutions instead of matrix multiplications.

A convolution can be compared to a cross-correlation. The concept of cross-correlation is to slide a filter spatially across some input data, like an image, and compute the sum of products in each position. Given a filter w with kernel size $2K + 1$ and some input data x in a point (p, q), the cross-correlation is given by:

$$z[p, q] = w * x = \sum_{r=-K}^{K} \sum_{s=-K}^{K} w[r, s]x[p + r, q + r] \tag{2.3}$$

Convolutions are the same as cross-correlation, but the filter is rotated 180 degrees. The general expression of a convolution is then:

$$z[p, q] = w * x = \sum_{r=-K}^{K} \sum_{s=-K}^{K} w[r, s]x[p - r, q - r] \tag{2.4}$$

When performing convolution on the input data, the convolution can be calculated in each pixel position, or some positions can be skipped. The spatial step between each convolution is called stride. For stride = 1, the convolution is performed in all pixel positions. As seen in figure 2.2.2, the entire filter needs to be inside the image to perform a convolution, which means that the input image shrinks linearly with the kernel size. To avoid this, the image can be broaden by adding extra pixels to the edges of our image. This is called padding.

There are three main differences between CNNs and regular neural networks. The first difference is sparse interaction. The output from a layer is only connected to a limited number of pixels from the previous layer, and not all pixels. While moving deeper into the network, neurons indirectly interact with an increasing part of the input, which eliminates the need for fully connected layers. The number of input pixels that are visible to a neuron in a CNN is called the *receptive field.* In some computer vision tasks, the size of the receptive field is crucial for good performance

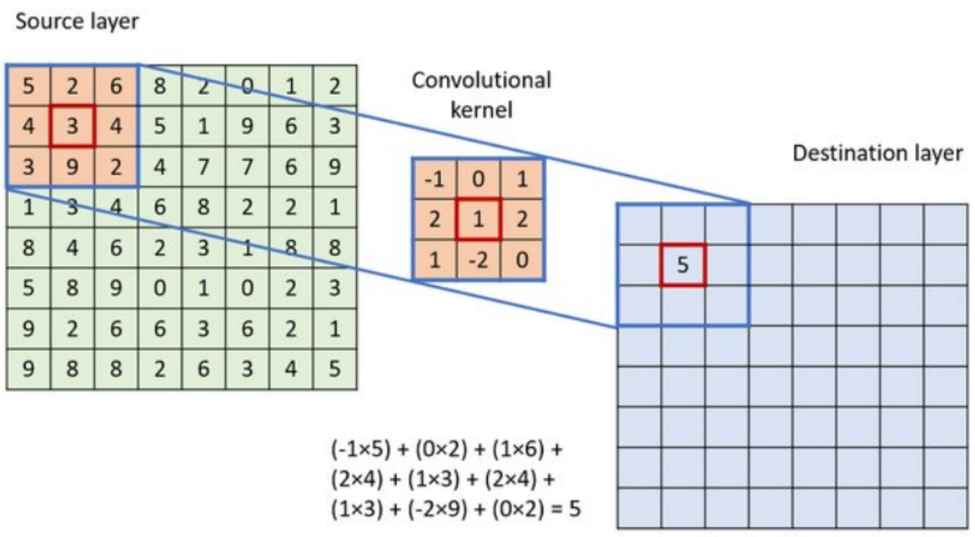


Figure 2.2.2: The result of one convolution with a 3x3 filter kernel. Illustration from [3].

.

because it helps the network to understand connections between features extracted from the first layers [18]. The receptive field increases with the depth of the networks, so a way to increase the receptive field is to have a deep network. Another solution is to increase the size of the filter kernel or to increase the stride. The second difference is that the same parameters are used for several functions in the same model. When a convolutional kernel is slid over the input data, the filter kernel remains the same the entire time. A constant filter kernel results in fewer network parameters compared to fully connected neural networks, where there is one parameter per input value. This allows the network to limit the amount of weights in each layer [10]. The third and last difference is equivariant representation, which is a result of shared parameters. This means that if the input fed to the network shift, then the output changes the same way [10].

A convolutional layer consists of n filters of equal dimensions. The depth of the input is equal to the number of layers in the previous layer, and the number of layers corresponds to the output depth of the current layer. Each neuron activation is computed from a convolution between the input and the weights in a layer.

$$y = W * x + b \tag{2.5}$$

Here x represent the input, W are the weights of the filter, and b is a bias. An example of a convolutional layer is shown in figure 2.2.3. For each layer, padding size, stride size, and kernel size are set. These parameters are set for this layer and do not change.

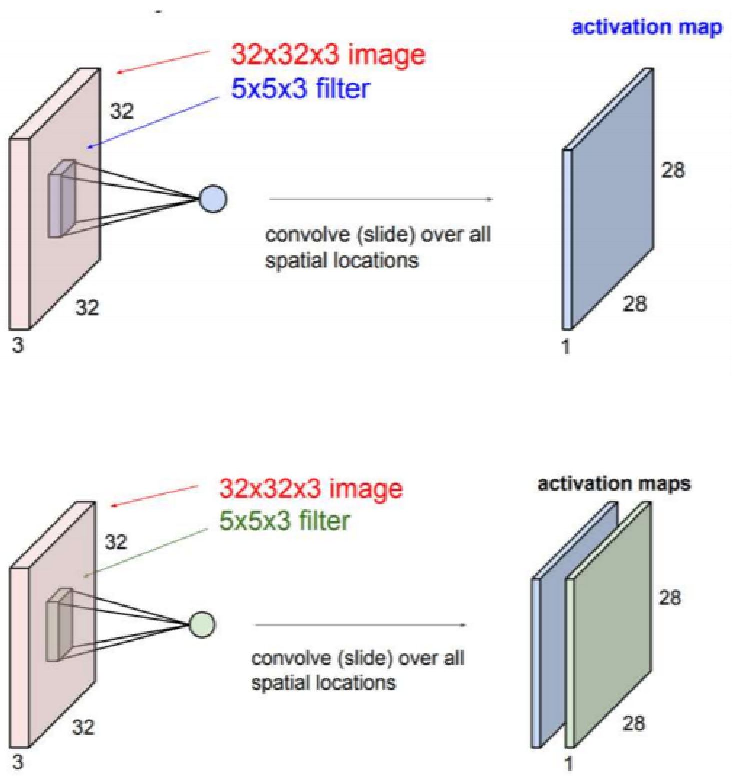


Figure 2.2.3: The result of sending an input of size 32x32x3 through a convolutional layer with two filters of size 5x5x3 with no padding and stride 1. The output has the size 28x28x2. The figure is from [6]

Convolutional networks does not only consist of convolutional layers, but are supplemented by other types of layers:

Fully connected layers

A fully connected layer means that all neurons in the current layer have a connection with all neurons in the previous layer. This layer is the same as the layers in feedforward neural networks, which is equivalent to a matrix multiplication [22]. Fully connected layers are typically used as the last layer in a convolutional neural network.

Pooling layers

Pooling layers progressively reduce the spatial size of the representation to reduce the need for memory and the number of computations of the network. It is therefore common to let the pooling layer contain a 2X2-filter with stride 2 to downsize the input with scale 2 in each dimension. Typical operators to use in pooling layers are max-, average- and the L2-operator. Since a pooling layer downsamples the input, the output can be considered a summarize of the input. Pooling layers are also a tool to control overfitting, which will be explained further in section 2.4.

2.2.1 Activation functions

An activation function is a non-linear, differentiable function which modifies the activation of neurons after a convolutional layer. The output of the activation function is sent to the next layer and used as input. As mentioned earlier, without the activation function, the network becomes a linear mapping from input to output. The nonlinearities in the activation functions are critical because they allow networks to calculate highly non-linear functions [17]. There is not a true answer to which activation function is best. It depends on the problem, and all functions have pros and cons. The activation function, which best fits a model, is found by testing different functions. Some common activation functions are shown in figure 2.2.4. In this section, exploding and vanishing gradients are mentioned. These expressions will be explained in section 2.3.2.

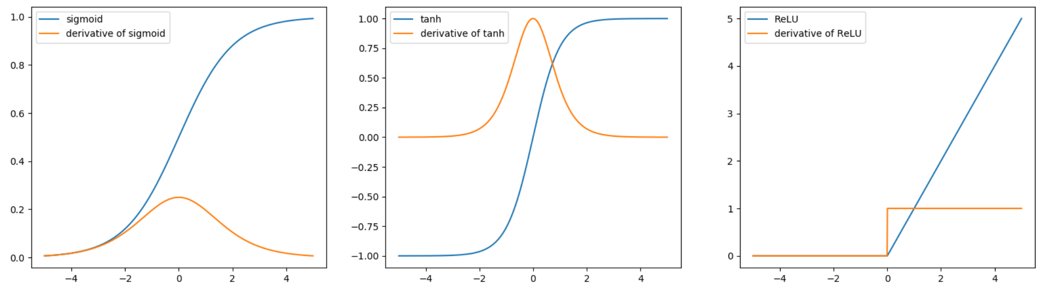


Figure 2.2.4: Some common activation functions and their derivative

Logistic Activation Function

Also known as Sigmoid function and is given by:

$$\phi(z) = \frac{1}{1+e^{-z}} \tag{2.6}$$

This means that $\phi(z) \in (0,1)$. This activation functions is usually applied when predicting probabilities, because the range of the function is the same as the probability range. If the task is to classify the input data, the logistic activation function will output the probability of the input belonging to a class, for all classes.

The logistic activation function transform a large input space inside a small input space, since $\phi(z) \in (0, 1)$. As a result of this, a large change in input will give a small change in output [30]. Moreover, the gradient of the loss function is small. Therefore, using this activation function may lead to vanishing gradients, but exploding gradients are avoided. As seen in figure 2.2.4, the derivative of the Sigmoid function is highest when the inputs are small. When the absolute value of the input increases, the derivative decrease, which can result in slow learning.

Tanh

The tanh activation function looks a lot like the logistic activation function, but the range of the Tanh activation function is wider than the logistic function. Tanh $\in (-1, 1)$. Tanh can also cause vanishing gradients, but less likely than logistic activation function. Since the derivative of tanh is steeper than the logistic activation function, the derivatives are larger.

ReLU

ReLU, or rectified linear unit, is given by:

$$\phi(z) = max(0, z) \tag{2.7}$$

hence, $\phi(z) \in [0, \infty)$. Unlike the logistic activation function, the derivative of ReLU is not small, hence vanishing gradient is avoided using this activation function. However, this activation function "stop" the training process for a given neuron if the input is negative, since negative input is set to zero. Exploding gradients are also a risk using ReLU, but this can be handled with other methods, like batch normalization. There also exists several variation of ReLU to avoid negative values being set to zero, like exponential linear unit (ELU):

$$\phi(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha(exp(z) - 1) & \text{if } z < 0 \end{cases} \tag{2.8}$$

and leaky ReLU:

$$\phi(z) = max(\alpha z, z) \tag{2.9}$$

but these are more expensive to compute compared to ReLU.

2.2.2 Building a convolutional neural network

The hidden layers are usually convolutional layers and pooling layers. The number of layers used in the network is one of the hyperparameters that need to be optimized. A general rule is that the more layers added to the network, the more complex mappings the network can represent. The different layers in a CNN extract different types of features, where some of those are visualized in figure 2.2.5. The first layers extract low-level features, like edges and lines. while moving deeper into the network, the layers detect more complex features. This is called mid-level features and high-level features. The last layer of a convolutional neural network is fully connected. The output has the same shape as the true value y.

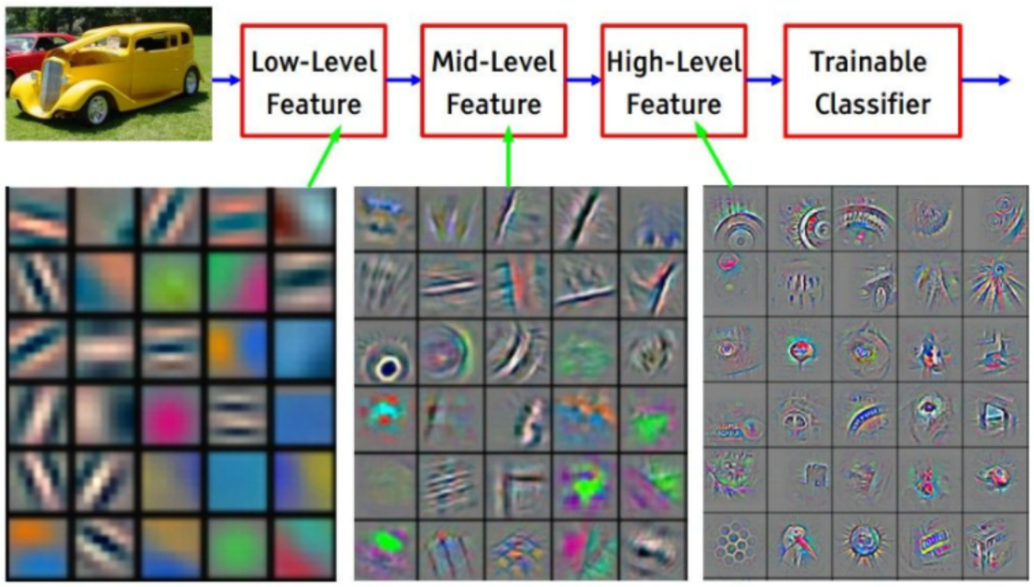


Figure 2.2.5: Different features extracted from each layer

2.2.3 Convolutional neural networks on three-dimensional data

We can perform convolution on 3D images the same way as 2D images, by using a three-dimensional kernel, as visualized in figure 2.2.6. Unlike the regular convolutional network, the filter kernels in a layer are smaller than the number of channels in the input data. This enables the filter to move in all three directions of the volumetric input data, making the output 3D as well. Hence, CNNs for volumetric data can be created the same way as for 2D data. The expanding of the filter kernels from 2D to 3D results in more parameters. In general, the more parameters the network contains, the more data is needed for training. One would assume that a CNN applied on 3D images would need more images than a CNN used on 2D images. However, since one single 3D images contain more data, this is not an issue.

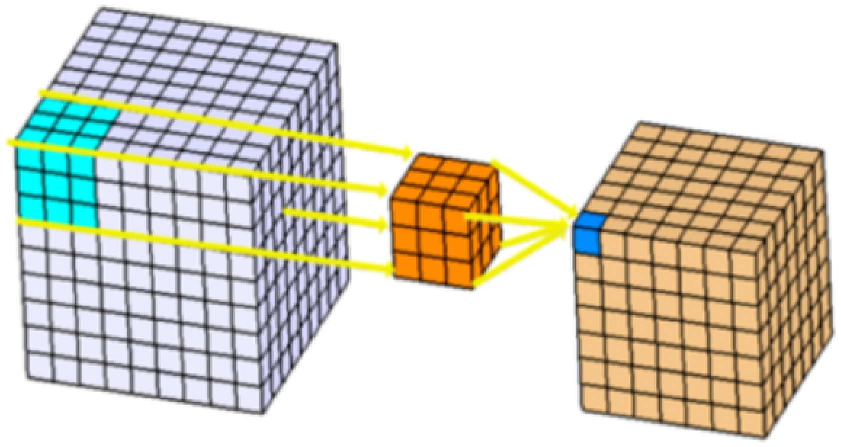


Figure 2.2.6: Illustration of a 3D convolution from "A Comprehensive Introduction to Different Types of Convolutions in Deep Learning" by Kunlun Bai.

2.2.4 Residual networks

An essential factor in the performance of a network is the number of layers. One would think that better results are achieved when more layers are used. This statement is only valid up to a certain number of layers before the accuracy of the network becomes saturated. The gradient from the loss converges to zero after several applications of the chain-rule [26]. When the gradient is close to zero, the weights of the layers will not update. When propagating through the network, the layers learn less as the gradient decrease. At best, only the last layers of our network gets updated. This event is known as the vanishing gradient problem. A solution to this is to add a so-called "shortcut connection". As shown in figure 2.2.7, the shortcut connection is an identity mapping, which skips a given number of layers, and is later added to the output of the skipped layers. By considering a neural network with input x, which is trying to learn a mapping $H(x)$, the residual, or difference, are given by:

$$R(x) = H(x) - x$$

Which gives us

$$H(x) = R(x) + x$$

The gradient can move through these shortcut connections and also reach the initial layers [26], which enable us to train deeper networks. A residual network

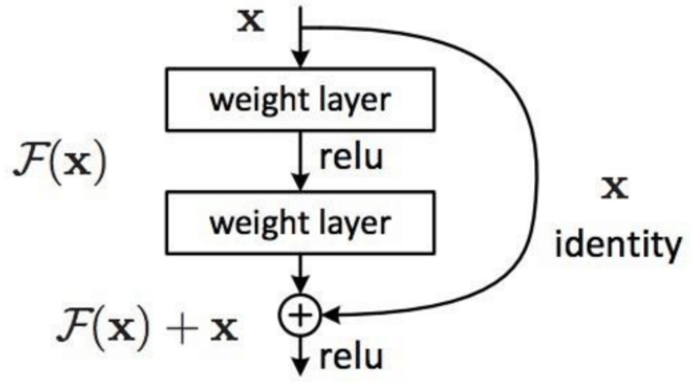

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.2.7: Illustration of a residual block with shortcut connection

consists of residual blocks put together, as shown in figure 2.2.8. The residual block in figure 2.2.7 has only two layers, but the number of layers within the block is optional.

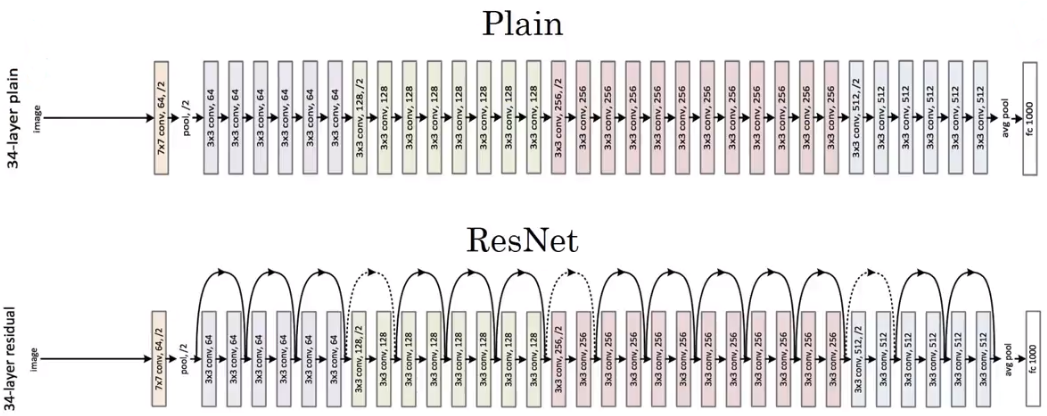


Figure 2.2.8: A plain neural network and an example of ResNet containing shortcut connection

2.2.5 Classification using convolutional neural networks

When using neural networks for classifying objects, the network returns the probabilities for the object belonging to each of the classes. The object is placed in the class with the highest probability. The last layer of a network for classification has a fully connected layer at the end, where the number of output values match the number of classes. The most common choice of activation function for classification is the Sigmoid activation function:

$$F(x_i) = \frac{1}{1 + e_i^x}$$

or the softmax activation function:

$$F(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}}$$

for $i = 1, ..., k$ classes.

Both of the functions return a probability between 0 or 1 for the object belonging to each of the classes. Choosing the activation function depends on the data. More specifically, it depends on whether an object can be classified into more than one class or not. In the case where all objects only belong to one single class, the best choice is softmax. The reason for this is that the softmax activation function forces the sum of all probabilities to be equal to one. Hence if the probability of one class increases, then the probability for the rest of the classes decreases. The function then avoid suggesting multiple classes as an option and decides on one. When using the Sigmoid activation functions, the sum over all probabilities does not necessarily add up to one. Therefore, the Sigmoid function is used if the objects fit into multiple classes.

2.3 Training a neural network

As mentioned, the purpose of a neural network is to estimate the mapping between x and y with a function $\hat{y} = f(x; \Theta)$, so that $\hat{y} \approx y$. This section will demonstrate how the parameters Θ are optimized to give the best approximation of true mapping f.

2.3.1 Loss function

We can measure the performance of a network by using a loss function. The purpose of the loss function is to measure the error between the prediction $\hat{y}$ and the true value y. Deciding which loss function to use depends on the task of the network. There are two categories of loss functions, Regression losses and

Classification losses [23]. If the network is trying to classify items on images, a good loss function can be cross-entropy loss:

$$L_{crossentropyloss} = -(ylog(\hat{y} + (1-y)log(1-\hat{y})) \tag{2.10}$$

However, if the task of the network is to predict some value, a good loss function can be a mean squared error:

$$L_{MSE} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{2.11}$$

2.3.2 Gradient descent

In calculus, the global minima is found by calculating the derivative of the function and checking the extrema, which often works when the function is convex. A function is convex if the second derivative is non-negative, which usually applies to functions with few parameters. Hence this method can not be used for non-convex functions, like the loss function to a neural network. Gradient descent is a method for minimizing functions which are too complex to minimize using calculus [20]. The gradient of a function f is a vector with the partial derivatives of f in point $x = (x_1, x2, ..., x_n)$ as components:

$$\nabla f(x) = \left[\frac{\delta f}{\delta x_1}(x), \frac{\delta f}{\delta x_2}(x), ..., \frac{\delta f}{\delta x_1}(x)\right]^T \tag{2.12}$$

The gradient tells in which direction the function decreases fastest and at which rate in each point p. If the gradient is zero, then p is a stationary point. If the gradient is non-zero, then f increases fastest at p in the same direction as the gradient. If the task is to minimize a non-convex function f, this can be done by iteratively taking small steps in the direction of the negative gradient [28]. For a starting point $x^0 = (x_1^0, x_2^0, ..., x_n^0)$, our new point $x^1 = x_1^1, x_2^1, ..., x_n^1$ is:

$$\begin{aligned} x_1^1 &= x_1^0 - \alpha\frac{\delta f}{\delta x_1}(x^0) \\ x_2^1 &= x_2^0 - \alpha\frac{\delta f}{\delta x_2}(x^0) \\ &\vdots \\ x_n^1 &= x_n^0 - \alpha\frac{\delta f}{\delta x_n}(x^0) \end{aligned}$$

where the learning rate α controls the step length. this can be generalized to:

$$x_j^i = x_j^{i-1} - \alpha\frac{\delta f}{\delta x_j}(x^{i-1}) \tag{2.13}$$

For each iteration, the point x_j^i moves closer and closer to the minimum of f, as shown in figure 2.3.9. However, there is a risk of getting stuck in a local minimum or a saddle point when minimizing using gradient descent. This can be solved using momentum.

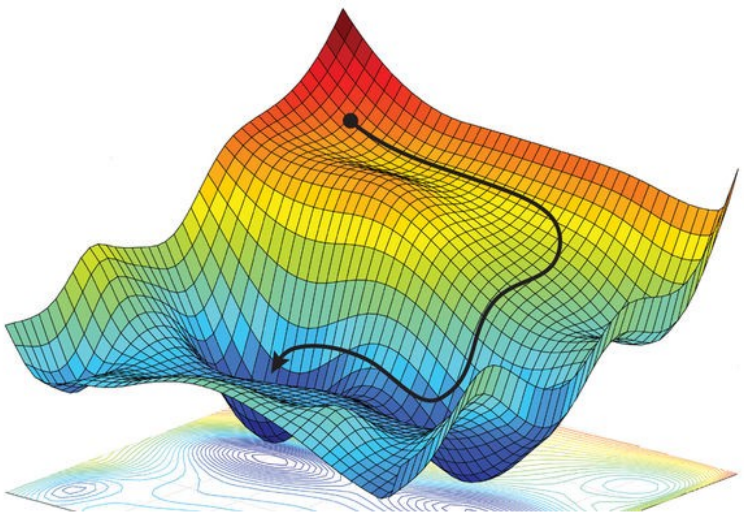


Figure 2.3.9: Finding the minimum of some function using gradient descent. Figure from [13]

2.3.3 Gradient decent with momentum

Adding momentum to gradient descent is a technique to avoid getting stuck in local minima or saddle points. When moving along the negative gradient, the step length is no longer dependent on just learning rate and size of the loss, but also the momentum caused by the steepness of the function. This can be done by using the velocity created by the slope and add this to the step length. The steeper the function is, the larger velocity is achieved, and the step length increase. Remember regular gradient descent was given by:

$$x_j^i = x_j^{i-1} - \alpha \frac{\delta f}{\delta x_j}(x^{i-1}) \tag{2.14}$$

for simplicity, x_j^i is denoted as w and x_j^{i-1} as x:

$$w = x - \lambda \delta w$$

We include momentum with the following equations:

$$v = \rho v - \lambda \delta w \tag{2.15}$$

$$w = w + v \tag{2.16}$$

Where ρ is the momentum parameter, and v is the velocity at a given point. If the network comes across a local minimum or saddle point, this is passed if the velocity is sufficiently large. The momentum is a hyperparameter that needs to be optimized as well. If the momentum is too small, the network will still get stuck in local minima and saddle points.

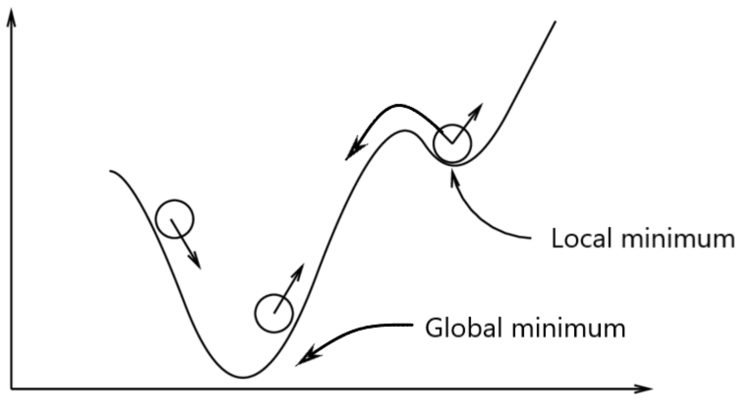


Figure 2.3.10: Momentum helps the network escape local minima.

2.3.4 Gradient descent with Nesterov momentum

Nesterov momentum differs from regular momentum because it considers previous iterations when the velocity is calculated, and the momentum is built up. Also since velocity is pushing us towards the point $w_{ahead} = w + \rho v$,the gradient is calculating in the point previous to the current point $w_{previous}$ instead of w:

$$\begin{aligned} w_{previous} &= w + \mu v \\ v &= \mu v - \lambda \delta w \\ w &= w + v \end{aligned}$$

2.3.5 Training algorithm

The first step in training a neural network is to send a batch of data through the network. Next, the predicted values provided by the network are compared with the true values, and the error between these values are computed. The total error of our network is represented with a suitable loss function L, as described in section 2.3.1. The goal is to minimize the error of the network. The error is minimized by optimizing the loss function. By using gradient descent on the loss function, all the weights in each layer are updated through the backward propagation. Let z be the weights in the final layer. The derivative of the loss function $\frac{\partial L}{\partial z}$ is calculated and used to find

$$\frac{\partial L}{\partial w} \quad \text{and} \quad \frac{\partial L}{\partial b} \tag{2.17}$$

We update w and b with the following equation:

$$w = w - \lambda \frac{\partial L}{\partial w} \tag{2.18}$$

$$b = b - \lambda \frac{\partial L}{\partial b} \tag{2.19}$$

where λ is a predefined learning rate. The algorithm starts in the last layer and work backward until the first layer is reached, updating all parameters on the way. Hence $\frac{\partial L}{\partial w^l}$ and $\frac{\partial L}{\partial b^l}$ is computed for all layers. This is calculated using the chain rule. Consider a function $y = f(x)$, where f depends on the functions $g_1, ..., g_n$, then the derivative of f is given by:

$$\frac{\delta f}{\delta x} = \sum_{i=1}^{n} \frac{\delta f}{\delta g_i} \frac{\delta g_i}{\delta x} \tag{2.20}$$

First the loss is computed:

$$\frac{\delta L}{\delta y^l}$$

in all layers $l = 1, ..., n$ and then this is used to derive

$$\frac{\delta L}{\delta w^l} \quad \text{and} \quad \frac{\delta L}{\delta b^l}$$

Sending images through the network, calculating the loss, and updating the weights with backpropagation is repeated until convergence.

There are two possible problems with using backpropagation with gradient descent, vanishing gradient, and exploding gradients. Vanishing gradients means that the gradient of the loss function converges to zero. The weight-update during backpropagation is dependent on how large the gradient of the loss is. Larger gradients result in more significant updates. We have vanishing gradients when the gradient is close to zero. The changes of the weights during backpropagation becomes small, and the network training becomes time-consuming. Exploding gradients are the opposite issue. The large gradients make extensive updates in the weights of the layers. The network will most likely not converge because the updates cause the network to "hop" over the minima.

2.4 Overfitting and underfitting

When training a neural network, two problems might stop our network from achieving good results, underfitting and overfitting. Figure 2.4.11 is used as an example. The points in the example are from a sine wave. The blue points represent samples from the training set and red from the test set. We want to estimate the points

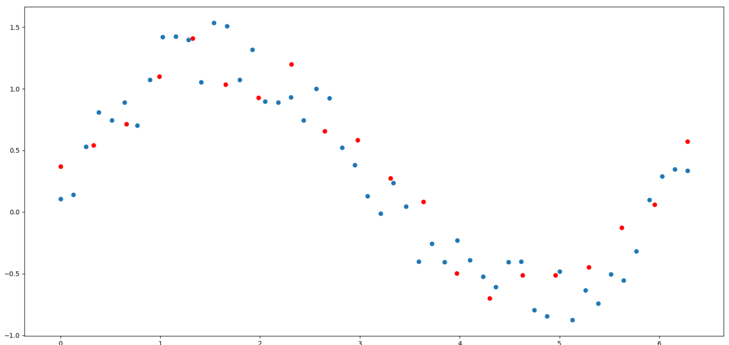
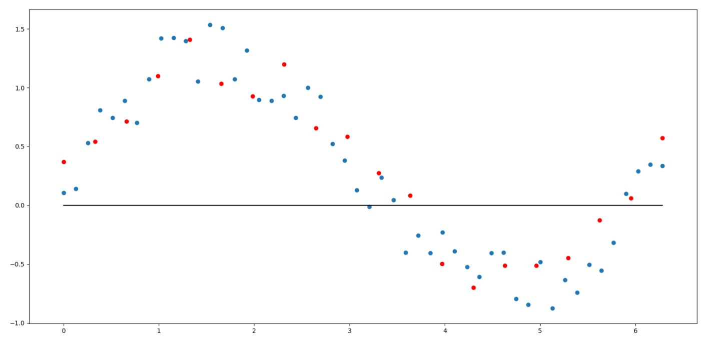


Figure 2.4.11: Points from an unknown function. The blue points represent samples from the training set and red from the test set.

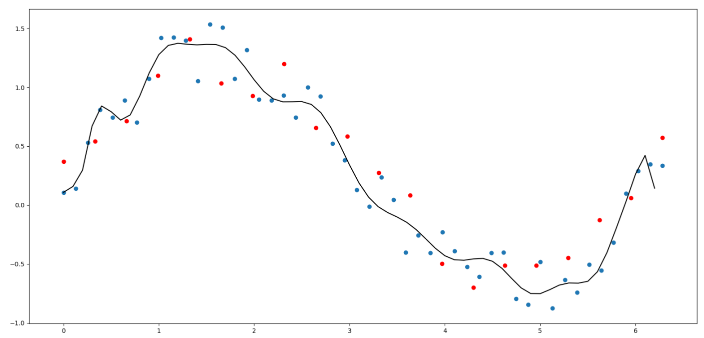
with a model such that the model not only fits the points available for training the model, but also fits unseen data points. Figure 2.4.12a shows an underfitted model. This model is too simple and will not be able to represent the variety in the data and perform badly for both the training set and the test set. An overfitted model is shown in figure 2.4.12b. This model is customized to fit only the training data, and perform poorly to unseen data.

Figure 2.4.12c shown a model that is not overfitted nor underfittet. To achieve this, some error needs to be tolerated by the model.

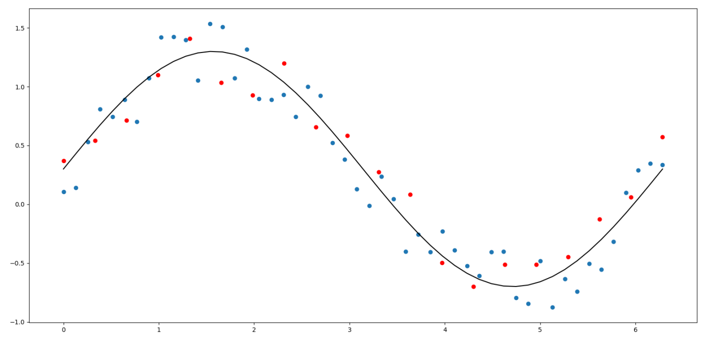
It is possible to understand whether a neural network is overfitting or underfitting by looking at the training loss and the validation loss. When training a neural network, the loss from the validation set is expected to be slightly larger than the loss of the validation set. During training, both losses should decrease at the same pace. If the validation loss begins to saturate or increase as the training loss continues to decrease, the network fits too well to the training set, and the network is overfitting. This is shown in figure 2.4.13. A model is underfitted if the overall performance is poor, and the loss does not improve during training [27].



(a) Underfitted model



(b) Overfitted model



(c) Good model. Not overfitted or underfitted

Figure 2.4.12: blue points symbolize the data used to fit a model to. The black line is the purposed model.

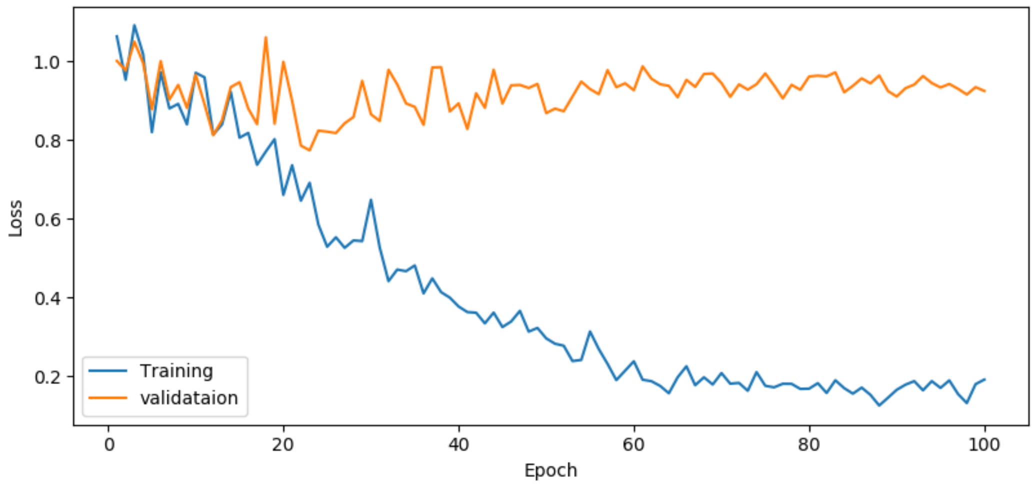


Figure 2.4.13: Training loss and validation loss when the neural network is overfitting to the training data.

There are several techniques for avoiding underfitting and overfitting:

Add more data

If possible, collect more data for the training set. Increasing the training dataset can help the network to generalize better, and therefore reduce the chance of overfitting [15].

Data augmentation

Image augmentation is a collection of methods for increasing the number of training samples. Data augmentation is a good alternative if, for some reason, it is not possible to get more data. Creating more data can be as easy as flipping an image vertically or horizontally, changing contrast and brightness, resizing images, or create new data by using smaller parts of the original data. The last technique will be further reviewed in section 3.2.

Dropout

Dropout means that the network is "dropping out units" during training [29]. In practice, this is done by setting random activations to zero and ignoring its incoming and outgoing connections during training. A fixed probability p denotes the probability that a node is staying active. During testing, all nodes stay active. When training a neural network using standard backpropagation, some neurons might learn to correct other neurons mistakes, which leads to complex co-adaptations [29]. These do not generalize well, and the network overfit to the training data. By adding random dropout, all neurons need to perform well on is own because the network can no longer rely on neurons to compensate for each other's mistakes. Besides, dropout prevents the network from becoming reliant on any single set of neurons, as they might be dropped out. Adding dropout also stops the network from relying on specific nodes; this means that "different" networks are trained each round.

L2 Regularization

The network is more likely to overfit to training data if the weights in the network become too large. An additional term is added to the loss function to prevent this. For a given loss function l and the weights w in our network, a penalty is added to the loss function:

$$l + \frac{\lambda}{2}||w||^2 \qquad \text{for } \lambda \in [0, 1]$$

where λ regularization strength. The penalty will reduce the values of the weights and therefore avoid overfitting.

Change network complexity

As seen in the example above, a too simple model will not give good predictions because it does not capture the complexity of the problem, which results in underfitting. This is solved by making the model more complex. This can be done by adding more layers to the neural network. In the opposite case, if the network is overfitting to the training data, removing some layers may help.

Early stopping

The concept of early stopping is to stop training if the model begins to overfit to the training data. During training, the validation loss is monitored and saved together with the weights of the model. If the loss increase compared to earlier loss values during training, the training is terminated, and the model with the lowest validation loss is returned. The loss will not necessarily decrease continuously, and it might sometimes increase for a short period before it begins to decrease again. A tolerance is defined for how many iterations the validation is allowed to increase before training is stopped.

2.5 Batch normalization

When training a neural network, the parameters in each layer change, which again changes the distribution of network activations. This is called *The Internal Covariate Shift* [14]. This complicates the training because the layers need to adjust to changes in the input distribution, which forces us to choose a small learning rate. A consequence of this is a slow convergence. A solution to this problem is to add *Batch normalization.* For each layer k, the batch normalization scales a d-dimensional input $x = (x^1, ..., x^d)$ to have zero mean and a variance equal to 1: [14]

$$\widehat{x^k} = \frac{x^k - \mathrm{E}[x^k]}{\sqrt{\mathrm{Var}[x^k]}} \tag{2.21}$$

Then two parameters γ and β are added. These parameters shift and scale the normalized data:

$$y^k = \gamma^k \widehat{x^k} + \beta^k$$

where

$$\gamma^k = E[x]$$
$$\beta^k = \sqrt{\mathrm{Var}[x^k]}$$

Consider a mini-batch $\mathcal{B} = \{x_1, ..., x_m\}$ of size m, the mean and variance are calculated before normalization, scale and shift:

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.22}$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \tag{2.23}$$

$$\hat{x_i} = \frac{xi - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{2.24}$$

$$y_i = \gamma\hat{x_i} + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \tag{2.25}$$

The ϵ is added to avoid zero division. In Ioffe et al. [14], it has been shown that applying batch normalization to state-of-the-art image classification model reach the same accuracy using fewer training steps, and also achieve lower error in classification tasks.

Besides making the training more effective, batch normalization also has other positive side effects, like prevent vanishing or exploding gradients and prevent the model from getting stuck in a local minimum, besides having a regularizing effect [16].

2.6 Multi-Task Learning

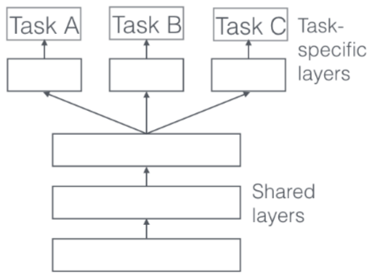
In standard machine learning, one single model are trained to perform some task. Then the hyperparameters are fine-tuned until the performance no longer increases. To further increase the performance of the network, multi-task learning can be used. This means that the model is trained to solve a similar task simultaneously with the main task. When a network is trained for only one task, the network might ignore information that can help improve the model. Multi-task learning can improve the performance of the model because it helps the model utilize this information and to generalize the problem [1]. There are two types of multi-task learning, hard parameter sharing and soft parameter sharing. Both types are shown in figure 2.6.14 In hard parameter sharing, all tasks share the first hidden layers, before the network split into task-specific output layers. In soft parameter sharing, there is one model for each task, with no shared layers, but the parameters in the models are regularized so that they are similar to each other.

Using multi-task learning has many benefits. Since the two task share layers, the model is forced to learn a more generalized representation, which reduce the chance of overfitting [1]. If data is limited or noisy, it can be difficult for å model to identify essential features. When there are two tasks, the added task supply with information about which features being most relevant, improving the feature extraction of the network.

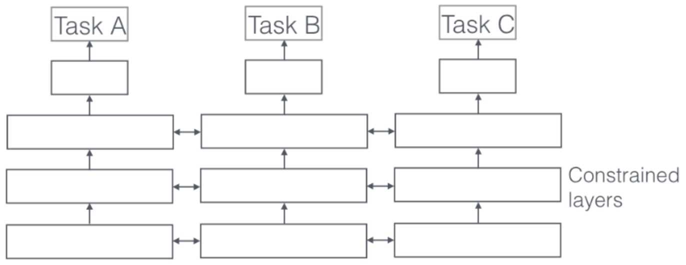
All the tasks have separate loss functions. A weighted sum over these losses is used for backpropagation.

2.7 Challenges of working with limited medical data

It is well known that training convolutional neural networks require a large amount of data. Unfortunately, there are some applications where a limited number of training data is expected, such as medical applications. There are many reasons why it is challenging to sample more data. For example, an expert is required to label the dataset. There can be regulations on the processing of personal data or not enough data subjects. Besides having fewer samples fore training, medical applications often require high precision to be accepted as a medical tool. There have



(a) Hard parameter sharing



(b) Soft parameter sharing.

Figure 2.6.14: Images from "An Overview of Multi-Task Learning in Deep Neural Networks" by Sebastian Ruder [1]

been proposed several solutions to solve the problems related to limited training data [35, 4, 12]. These methods generally combine data augmentation and multi-task learning.

2.8 Splitting dataset

Before the training of neural networks can begin, the dataset needs to be split into three separate sets, one for training, one for testing and one for validation. The training set will be used to train our model. The parameters of the model is fit by the samples in the training set. During training, the model is tested on the validation set as an unbias evaluation of the model. Hence the model is not trained on this data. The validation set is used to fine-tune the hyperparameters of the model and monitor the training process. After training the model, the test set is used as a final evaluation of the model. The results from the test set defines the final evaluation of the precision of the model.

Chapter 3

Method

A patch-based method combined with multi-task learning will be used for detecting anatomical landmarks in 3D cardiovascular images.

3.1 Data set

The data set is provided by GE Vingemed Ultrasound. It consists of 127 3D ultrasound images, each labeled with six anatomical landmarks using EchoPAC. Each image is an echocardiogram, acquired by passing the ultrasound probe into the patient's esophagus. This imaging technique gives clearer images compared to transthoracic echocardiogram.

There is a large variation in the size of each image in the dataset. Some of the images show the entire heart, as shown in figure 3.1.2, while some are close-ups of the valves. The smallest image is 6.17x6.23x4.86 cm, and the largest is 23.06x23.22x19.03 cm. By letting all images be 150x150x150 pixels, the resolution will vary from 0.29x0.29x0.23 mm^3 to 1.1x1.1x0.9 mm^2.

3.1.1 Anatomical landmarks

The six landmarks 'MA1', 'MA2', 'P', 'A', 'Coap' and 'Ao', and their position is shown in figure 3.1.2. The landmarks lie on two planes which intersect in Coap. MA1 and MA2 are placed on each side of the mitral valve. A is placed at the root of the mitral valve and the aortic valve. Ao at the opposite side of the aortic valve from A, and is P on the opposite side of the mitral valve. Coap is midpoint of MA1, MA2, P, and A. For a better understanding of the placement of the pints, see figure 3.1.1.

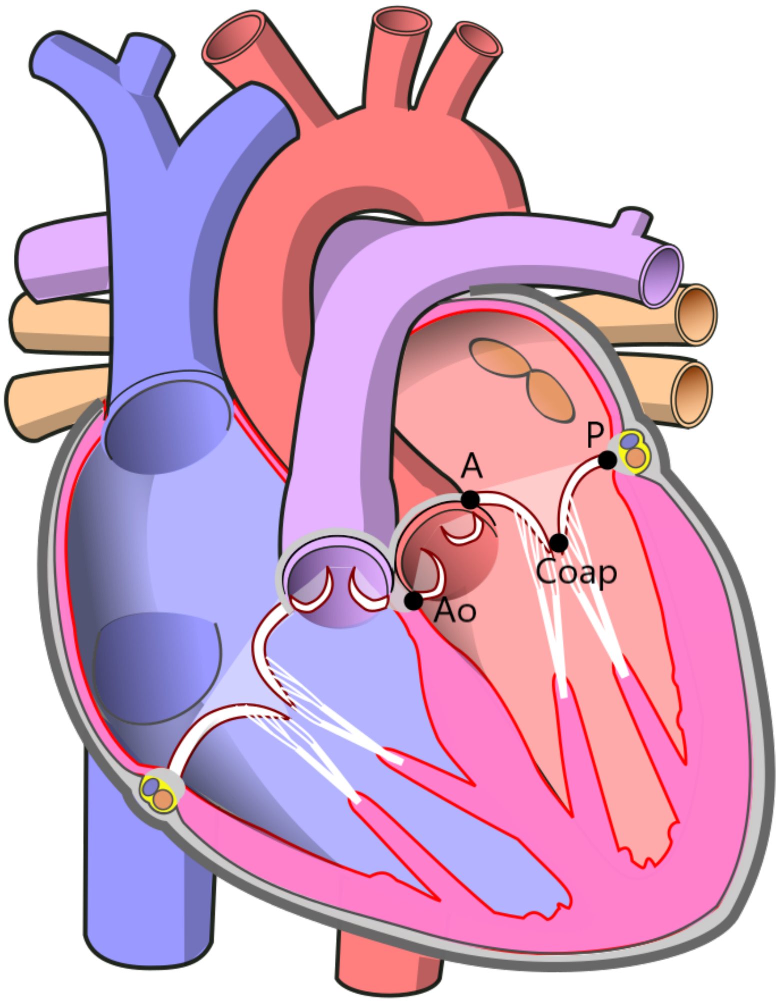


Figure 3.1.1: Anatomy of the human heart from [5].

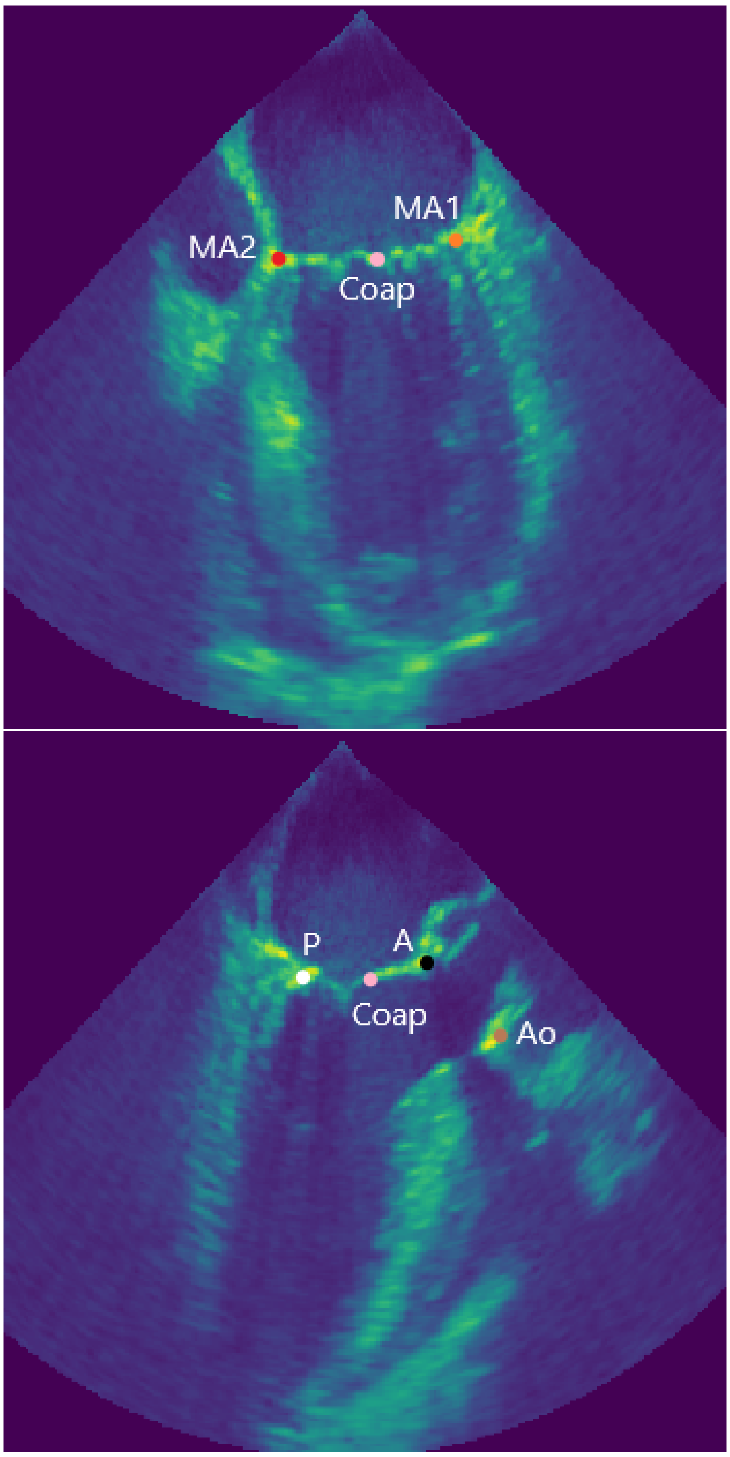


Figure 3.1.2: Red: MA2. Orange: MA1. Pink: Coap, White: P. Black: A. Brown: Ao

3.1.2 Dividing data into training, testing and validation set

Before starting training models, the dataset are split into training, validation, and test sets, as described in section 2.8. The training set is 96 images, the validation set is ten images, and the test set is 21 images. The data is divided to ensure that as much data as possible is used for training, while also having a large enough test set to reflect the variation of the data to get representative test results.

3.2 Patch based learning

Training a convolutional neural network can be challenging if the data set consists of a limited number of training subjects [35]. This can be solved using the data augmentation techniques presented in section 2.4. One of these techniques was to sample smaller patches from the original data, and use these as input to a convolutional neural network instead of the whole images. The dataset used in this thesis is relatively small (only 127 images). Using this simple approach, the dataset can be increased without needing to collect more data, and therefore avoid the problems related to limited data [33]. If the entire image is used as input, the input would be of size 150*150*150 = 3375000, hence using patch-based methods also avoid memory constraint.

Since the images in our dataset are three-dimensional, the extracted patches are also three-dimensional. Instead of training the network to predict the coordinates of a landmark, the network is trained to predict the displacement vectors between the input patch and each of the six landmarks. The outline of the patch-based method used in this thesis is shown in figure 3.2.3.

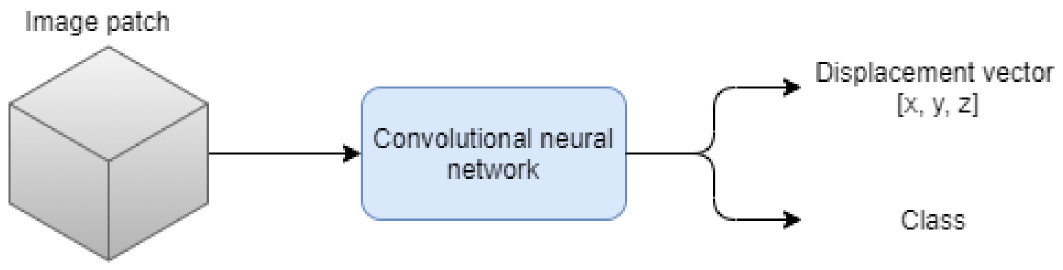


Figure 3.2.3: Outline of patch-based method

3.2.1 Prepossessing of data

The patch-based method requires preprocessing, which is patch extraction. Patches used during training are sampled random, while patches used for testing are sampled using a grid.

Random Patch Extraction

We start by extracting patches from each of the 3D images. The same number of patches is extracted from each image. All 3D images are resized to 150x150x150 pixels. Since the original images vary in size, the resized images will be at different scales, which will help make the networks invariant to scale. Two datasets with different patch sizes are generated. The first dataset has a patch size of 30x30x30 pixels, and the patch size of the second dataset is 60x60x60. The goal of the network is to find the correlation between patches and landmark locations. Moreover, the network needs to understand the location of the patch and use that information to locate the landmark.

The random patch extraction work as data augmentation to prevent overfitting. Since large parts of the original images contain no information, the extracted patches are examined with an algorithm to ensure that the patches that are sent through the network contain a sufficient amount of information. By looking at the images shown in figure 3.1.2 and 3.2.4a, we see that the image data is contained within a cone, and the areas outside this cone does not contain any information. Hence we want to avoid extracting patches from this area. By looking at the image data inside the cone, it is fair to assume that there is difficult for a network to understand the location of the patch if it only contains tissue or homogeneous noise caused by blood flow. Therefore, we want to avoid extracting patches from these parts of the image as well. The parts of the image containing the most information are the remaining areas, which includes the regions along the walls of the heart chamber and around the valves. These are the areas the patches should be extracted from.

First, a random patch is selected. Then, the pixel intensities are analyzed to ensure that the patch is extracted from the desired area. The pixel intensities range from 0 to 255. As seen on the color bar on figure 3.2.4a, the pixels can be categorize based on their intensity. The area outside the cone has a pixel intensity of approximately 0. The blood flow inside the heart chambers appear as noise with pixel intensity between 1 and 100. The intensity of the tissue is between 100 and 255.

Pixel intensity	Category
0	Area outside cone
1-100	noise created by blood flow
100-255	Tissue

We know that a patch is extracted from the area outside the cone if all pixels are 0. Likewise, if all pixels are between 1 and 100, the patch is most likely extracted from the blood pool inside a heart chamber. Patches where all pixels have an intensity less than 100 can therefore be discarded. Patches extracted from the tissue have all pixel intensities between 100 and 255, which means that these patches can also be discarded. Since the desired patches are extracted from areas around the walls and valves, the patches need to contain pixels with intensities from both tissue and blood. Informative patches are therefore extracted by setting a minimum threshold for how many pixels need to be present from each category, and only accept the patches that fulfill these demands. In figure 3.2.4a, examples of wanted patches (green squares) and unwanted patches (red squares) are shown.

patch extraction using grid search

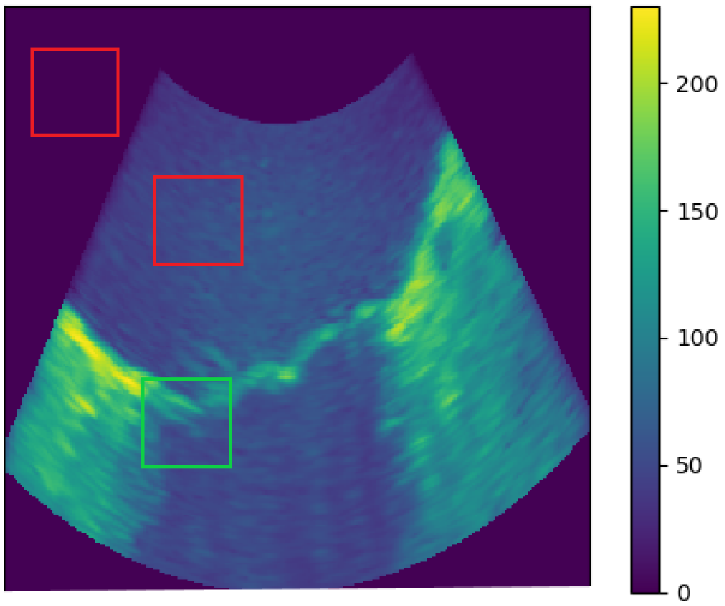
The patches that are used for testing are sampled using a grid to make it easier for others to recreate our results later. Only the patches that satisfy the same criteria as the patches selected randomly are accepted, which means that many of the extracted patches will not be used. This is to ensure that the patches are similar to the patches in the training set.

3.2.2 Find displacement vector

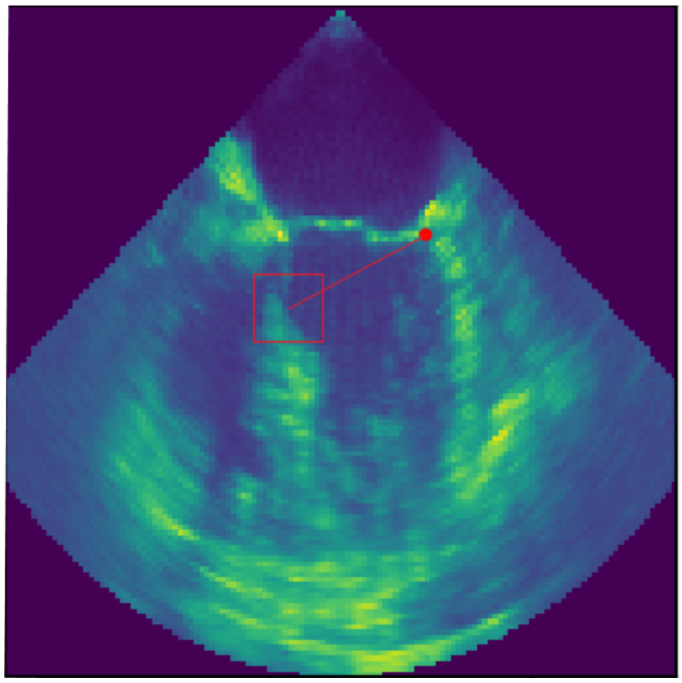
After finding an approved patch, the displacement vectors between the patch and each of the landmarks are calculated. The displacement vector is calculated using the center of the patch as the reference point. The displacement vector is given by

$$d_i = [\Delta x_i, \Delta y_i, \Delta z_i]$$

where Δx, Δy and Δz is the displacement in each axis. A displacement vector is visualized in figure 3.2.4b.



(a)



(b)

Figure 3.2.4: 2D visualization of patches (a): The green box represents a patch containing a sufficient amount of information, while the red boxes represent patches which do not. (b): The landmark MA1 and the displacement vector between these two

3.2.3 Patch classification

The purpose of the classification is to decide if an image patch contains a landmark or not. There is one class for each of the landmarks, and one class for the patches which does not contain any landmarks. Hence, the patches can be divided into seven different classes, as shown in table 3.1. As mentioned, the size of the images in the dataset varies. For large images, the landmarks are closer together. In the figures in 3.2.5, the six landmarks in the data set are visualized. In the same figures, potential patches of size 30x30x30 pixels are drawn in, with the correct ratio between image size and patch size. The figures show that a patch can belong to multiple classes. In figure 3.2.5a, both Coap and MA1 fit inside the patch, and belong to class 1 and 5. In figure 3.2.5b, three landmarks fit into one single patch. As mentioned in section 2.2.5, if the network needs to be able to classify a patch to multiple classes, a sigmoid activation function must be applied in the last layer.

class	Description
1	MA1
2	MA2
3	P
4	A
5	Coap
6	Ao
7	No landmark

Table 3.1: The different classes in the classification problem

3.3 Network architectures

To perform landmark detection, two very different architectures are tested, one state of the art network with skip connections, called ResNet18, and one more simple network which has shown to perform well on a similar task.

3.3.1 Fully convolutional neural network

The first network architecture is presented in Noothout et al. [21]. For data set with patches of size 30x30x30, the last max pool layer is removed, and for patches of size 60x60x60, the original network is used. The original network architecture are shown in figure 1.2.6 on page 8, while the network architecture used for smaller patches are shown in figure 3.3.6. The network uses multi-task learning, where the

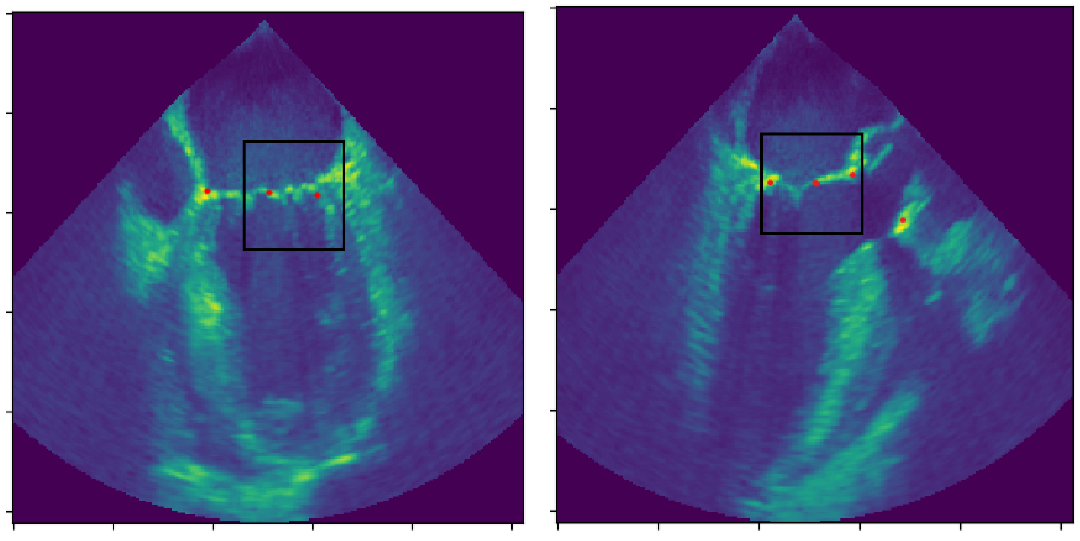

(a) (b)

Figure 3.2.5: The six landmarks are represented as red dots. (a) show MA2, Coap and MA1, and (b) show P, Coap, A and Ao. The black squares represent potential patches. A patch can contain several landmarks, and therefore belong to multiple classes

main task is to predict the six displacement vectors to each of the landmarks, and the second task is the patch classification. The two tasks use hard parameter sharing and share the first eight layers, which consists of six convolutional layers and three max pool layers. All convolutional layers have kernel size 3x3x3, stride 1x1x1, and exponential linear unit used as activation function. After this, the network split into two fully connected layers for each of the tasks. The fully connected layer used to predict the displacement vector has a linear activation function, and the fully connected layer for the classification uses a Sigmoid aviation function.

3.3.2 ResNet18

The second network used in this thesis is 3D ResNet18. The network is a residual network, as described in section 2.2.4, which consists of 18 3D convolutional layers. The architecture are visualized in figure 3.3.7. The First layer is a 3D convolutional layer with kernel size 3x7x7 and stride 1x2x2. Next, there are four different residual blocks, where each block is repeated twice. All convolutional layers in the residual blocks have kernel size of 3x3x3, stride 1x1x1, and ReLU as activation function, but the number of layers increases when move deeper into the network. 3D batch

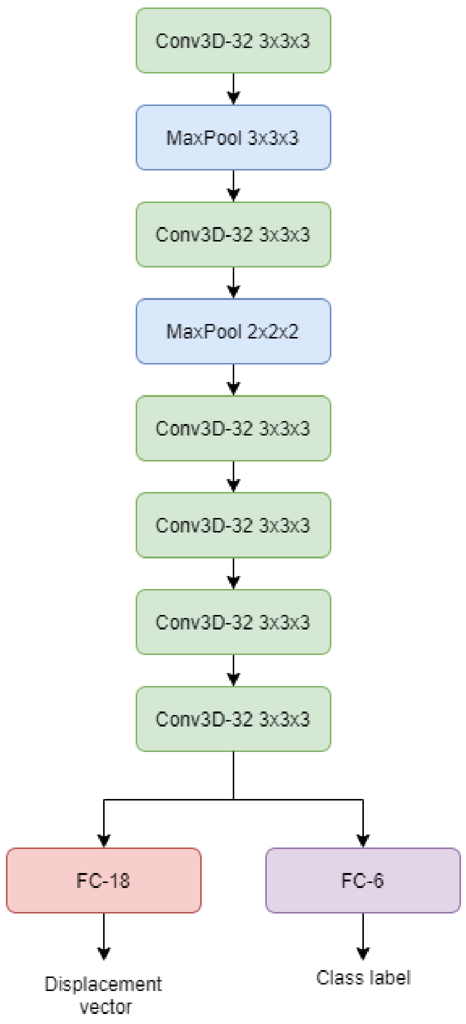
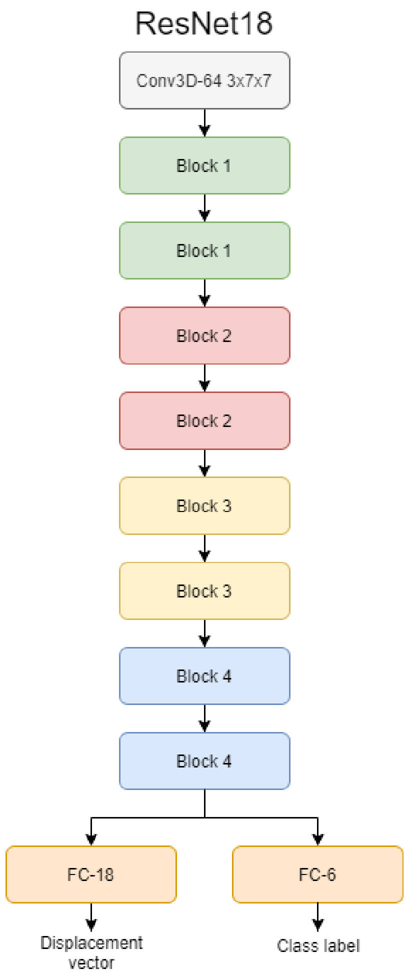


Figure 3.3.6: Fully convolutional neural network. Conv3D-xx denotes a 3D convolution where xx is the number of channels in the output. FC represents fully connected convolutional layers.

normalization is applied between each convolutional layer, and dropout is used after each residual block. As the network presented in the previous section, the original ResNet18 is modified to include multi-task learning with hard parameter sharing, by splitting the last layer of the network into two fully connected layers with the six displacement vectors and a class label as output.



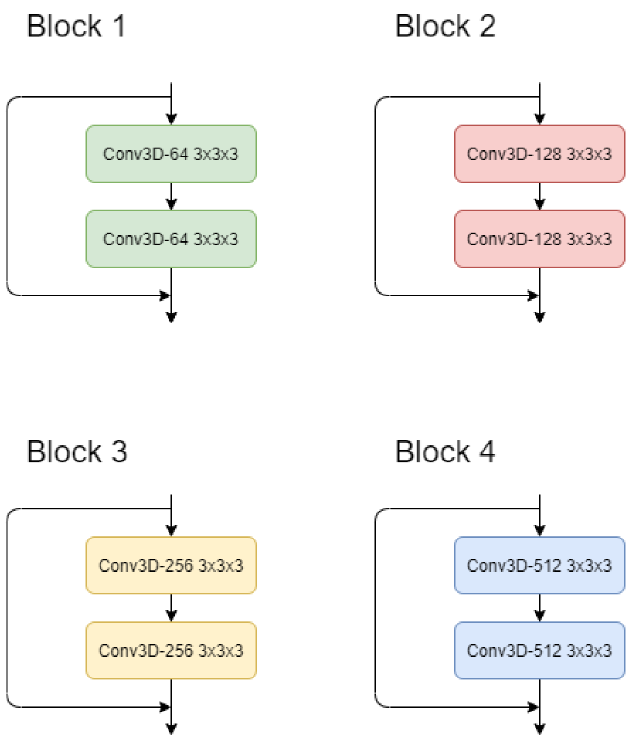


Figure 3.3.7: Illustration of layers in ResNet18. Conv3D-xxx denotes a 3D convolution where xxx is the number of channels in the output. FC represent fully connected convolutional layers.

3.4 Loss function

Since our model provides two outputs, each task has separate loss functions. These losses are added together before backpropagation.

3.4.1 Loss function for regression

From Noorhut et al. [21] and Zhang et al. [33], we know that we can expect the patches far away from a landmark to give more inaccurate prediction compare to patches close to a landmark [34]. This argues for weighting the loss function according to the distance between the patch and the landmarks for avoiding patches far away from making larger updates of the network parameters [21]. Patches with large displacement vector are given a small weight, while the ones with smaller displacement vector get a larger weight. For an image patch, the displacement vector between the patch and a landmark is given by

$$d_i = [\Delta x, \Delta y, \Delta z]$$

Let d^p be the predicted displacement vector and d^t be the true displacement vector. The weight is defined as:

$$w_i = e^{-\frac{||d^t||}{\alpha}} \tag{3.1}$$

$||d_i^t||$ is the length of the displacement vector and α is a scaling coefficient. The weight is combined with a regular Mean squared error loss as described in section 2.3.1. For a predicted displacement vector d_i^p and the true displacement vector d_i^t, the loss becomes

$$Loss = \frac{1}{6}\sum_{i=1}^{N} = w_i||d_i^t - d_i^p||_2 \tag{3.2}$$

For $i = 1, ..., 6$ landmarks.

3.4.2 Loss function for classification

One-hot encoding is used for the classifications. The output is then a vector with six values, which represent each of the landmarks. The value is 1 if the patch contains the landmark. If all indices are 0, the patch does not contain any landmark. For the classification, binary cross-entropy loss is applied:

$$l(y, \hat{y}) = L = \{l_1, ..., l_n\}^T \tag{3.3}$$

where

$$l_n = -(y * log(\hat{y} + (1 - y)log(1 - \hat{y})) \tag{3.4}$$

for each element $n = 1, ..., N$ in a batch. There is a large imbalance in the classes, because the majority of the patches will not contain any landmarks. Therefore, it is necessary to weight the cross-entropy loss accordingly, or else the network will most likely overfit to predicting that no patches contain landmarks. For each landmark $i = 1, ..., 6$, m denotes the number of patches that do not contain the given landmarks. The weight w_i is then given by:

$$w_i = \frac{m}{\text{number of patches in class i}}$$

which gives

$$W = [w_1, w_2, w_3, w_4, w_5, w_6]$$

The final loss is the binary cross-entropy loss multiplied with the weights:

$$\text{Loss} = L * W$$

3.5 Post possessing

Our model predicts displacement vectors and not final landmark, hence the output need post-processing to estimate the landmarks. The post-processing is only applied to output from networks during validation and testing, and does not impact the training process of the networks.

By sending a single image patch through the neural network, the output is six vectors and a class label. The six vectors are the displacement vector between the midpoint of the input patch and each landmark, MA1, MA2, P, A, Coap and Ao, and the class label indicates if the patch contains one of the landmarks or not. The output may look something like table 3.2 and 3.3.

	x	y	z
MA1	6.254	-2.156	1.429
MA2	6.179	-1.818	1.891
P	7.365	-2.070	1.696
A	5.029	-1.771	1.391
Coap	5.679	-2.087	1.652
Ao	3.546	-1.355	0.965

Table 3.2: Displacement vectors between the midpoint of a patch, to each of our six landmarks

.

Class	Output
Patch contain MA1	0
Patch contain MA2	0
Patch contain P	0
Patch contain A	1
Patch contain Coap	1
Patch contain Ao	0

Table 3.3: Results from classification

.

The next step is to use the displacement vectors to calculate the actual landmarks. Let us start by considering only one landmark. Given a patch p with index i, the coordinates of the midpoint of the patch is given by:

$$p_i = [x_p, y_p, z_p]$$

and the estimated displacement vector from the input patch to a given landmark in cartesian coordinates is given by:

$$d_i^p = [\Delta x_i^p, \Delta y_i^p, \Delta z_i^p]$$

At the end, use the midpoint of the patch and the displacement vector to calculate the landmark by simply adding them together.

$$l_i = p_i + d_i^p$$

This is done for all landmarks.

From each image, N patches are extracted. This means that N propositions for each of the six landmarks is obtained. These are used to estimate the final prediction. This can be done in several ways:

1. Use all N predictions and let the final prediction be the average of all predicted landmarks:

$$L = \sum_{i=1}^{N} \frac{1}{N} l_i \tag{3.5}$$

   This is shown in figure 3.5.8.

2. Use all N predictions. Instead of letting all predictions weight in equal, the prediction is weight based on its distance from the landmark, so that predicted landmarks from patches far away from a landmark get a low weight, while the patches close to the landmarks are assigned larger weights. The same weights as the ones applied to the loss function is used, but the predicted displacement vector is used instead of the true displacement vector:

$$L = \frac{\sum_i^n w_i l_i}{\sum_i^n w_i} \tag{3.6}$$

where

$$w_i = e^{-\frac{||d_i^p||}{\alpha}}$$

3. If present, only the patches containing a landmark is used to decide the prediction for that specific final landmark

4. Only use the patches predicted to contain the landmark and let the final landmark be a weighted mean of these predictions.

3.6 Evaluation of the model

The models provide two outputs, and two separate methods are needed for evaluating each of them. The method for evaluating the predicted displacement vector is the most important because that is the main task of the network. The classification also needs evaluation because the results for classification will be used during the final landmark prediction.

3.6.1 Evaluation of landmark detection

The best way to evaluate the regression, is to measure how far away our predicted points are from the true points. A suitable metric is the euclidean distance. Given the true landmark $y = [x_1, x_2, x_3]$ and predicted point $\hat{y} = [\hat{x_1}, \hat{x_2}, \hat{x_1}]$ the L2 norm is given by:

$$||y - \hat{y}||_2 = \sqrt{(x_1 - \hat{x_1})^2 + (x_2 - \hat{x_2})^2 + (x_3 - \hat{x_3})^2}$$

3.6.2 Evaluation of classification

The results from the classification can be divided into four categories:

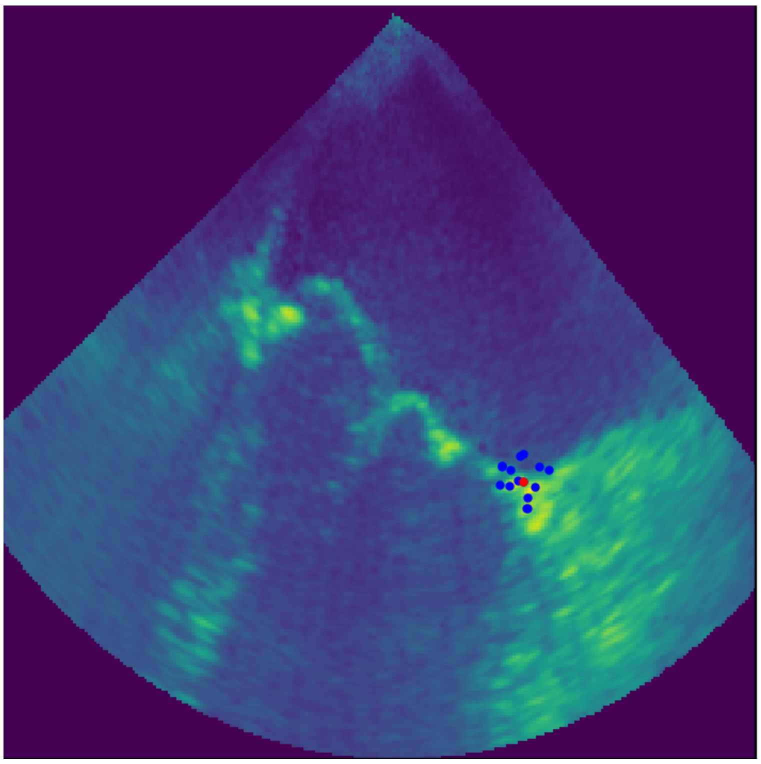


Figure 3.5.8: Example of combining multiple MA1 predictions (blue dots) into a single proposed landmark (red dot) using averaging

.

- True positive (TP): The patch contains the landmark and was predicted to contain the landmark.
- True negative (TN): The patch does not contain the landmark, and the model predicted the landmark not to contain the landmark.
- False positive (FP): The patch does not contain the landmark, but the model predicted the patch to contain the landmark.
- False negative (FN): The patch contains the landmark but was predicted not to contain the landmark.

The classification can be evaluated by looking at the precision, sensitivity and specificity of the model. The sensitivity denotes the percentage of correctly classified objects, relative to the total amount of samples in the class:

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

A sensitivity equal to 0 indicates none of the patches containing a landmark was detected, while sensitivity equal to 1 means that all were detected.

The precision denotes the percentage of patches classified to contain a landmark, actually contain the landmark:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

A high precision indicates few misclassifications to the given class. While a low precision means that many patches were classified to a class, which they do not belong to. Hence, it is possible to achieve high sensitivity and low precision at the same.

The last measurement is specificity:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

The specificity denotes the percentage of patches predicted correctly not to contain a given landmark, relative to the total number of patches not containing the landmark.

3.6.3 Evaluation of accuracy for CT-fusion tool

After landmarks in ultrasound images and CT images are detected, the images are registered in a common coordinate system by using an affine transform T. The fiducial registration error (FRE) denotes how far the paired landmarks are apart after registration:

$$FRE = ||T(l_{us}) - l_{ct}||$$

where l_{us} is the landmarks from the ultrasound images, l_{ct} is the landmarks from the CT scan and T is estimated using l_{us} and l_{ct}.

Chapter 4

Experiments and results

4.1 Classification

Before combining classification and regression in a neural network, the models performance on classification without regression is tested. Precision, sensitivity and specificity are examined for each landmark individually. The results for ResNet are shown in table 4.1 and the fully convolutional neural network are shown in table 4.2.

	MA1	MA2	P	A	Coap	Ao
Precision	0.13	0.36	0.16	0.21	0.26	0.0
Sensitivity	0.64	0.21	0.44	0.8	0.8	0.0
Specificity	0.78	0.95	0.69	0.37	0.38	0.73

Table 4.1: Results from classification using ResNet

	MA1	MA2	P	A	Coap	Ao
Precision	0.10	0.11	0.08	0.06	0.10	0.04
Sensitivity	0.46	0.08	0.45	0.78	0.90	0.50
Specificity	0.78	0.96	0.69	0.37	0.39	0.73

Table 4.2: Results for classification using FCNN

4.2 Regression

The regression is also tested alone. The results of this test provide a point of reference for evaluating whether adding classification and weighted loss improves the results. The results in table 4.3 show the average euclidean error of all displacement vectors in cm for each landmark. The average results for all landmarks are shown in table 4.4.

	MA1	MA2	P	A	Coap	Ao
ResNet	12.44	13.03	12.26	11.91	8.722	16.295
FCNN	37.03	35.80	38.25	21.80	29.30	39.51

Table 4.3: Average euclidean error for regression in mm

4.3 Combining regression with classification and weighted loss

Three tests will be performed:

1. Weighted MSE loss and excluded classification
2. Regular MSE loss and included classification
3. Weighted MSE loss and included classification

The results from each of the tests for both network architectures are shown in table 4.4.

Weighted loss	Classification	Error ResNet	error FCNN
no	no	12.443	33.615
yes	no	11.820	28.74
no	yes	11.436	24.73
yes	yes	11.073	22.10

Table 4.4: Average error for all networks. Error is denoted in mm

4.4 Predicting final landmark

In this section, only the model with the lowest error for the displacement vectors were used, which was ResNet18 with weighted loss and classification. Four techniques for deciding final landmarks were tested:

1. Use all N predictions and let the final prediction be the average of all predicted landmarks.

2. Let final landmark be a weighted mean from all N predictions. Patches close to landmarks are weighted heavier.

3. Only the patches predicted to contain the landmark are used to decide the final landmark.

4. Only the patches predicted to contain the landmark are used and the final landmark are a weighted mean of these predictions.

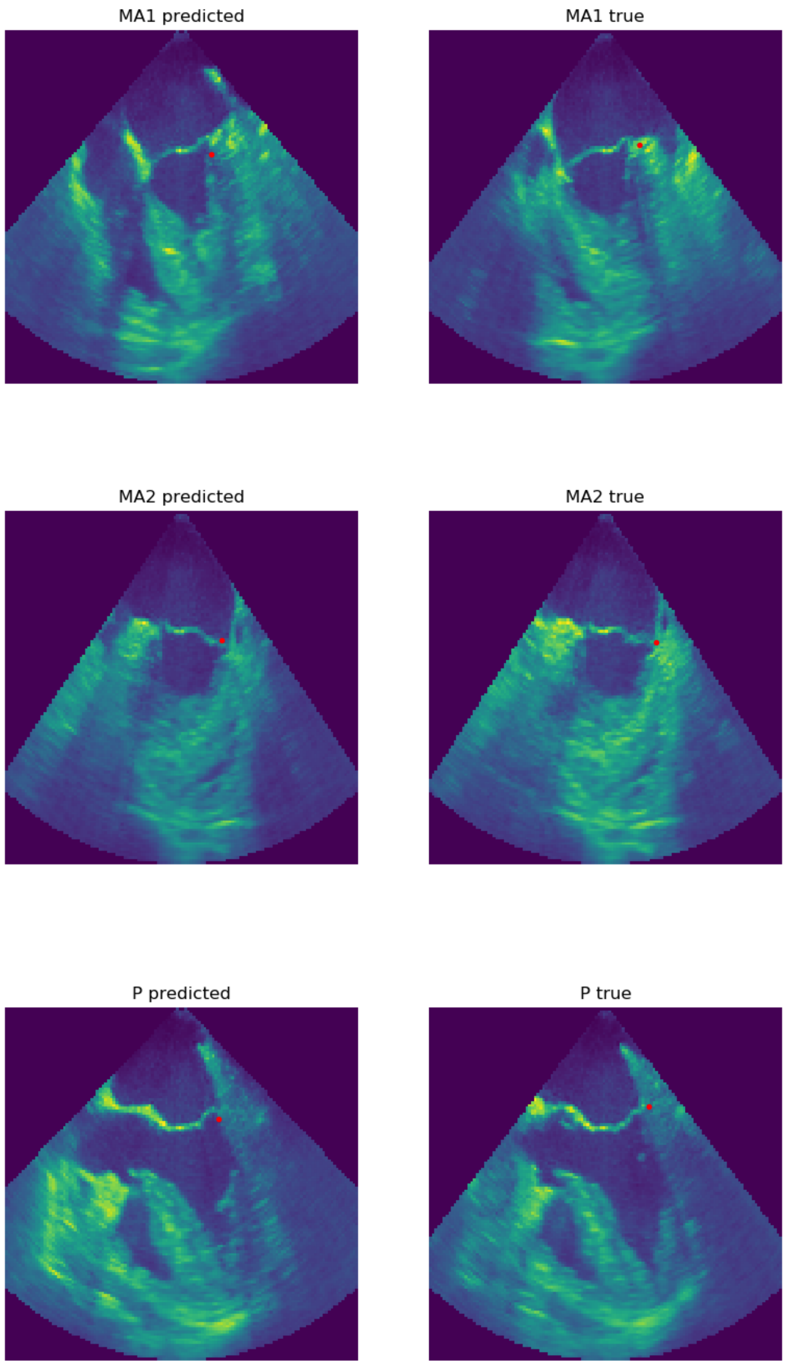
Weight	Classification	Mean	Median	Min	Max
no	no	9.12	8.18	1.21	33.77
yes	no	8.96	8.10	1.14	33.91
no	yes	8.88	7.60	0.81	30.08
yes	yes	8.78	7.48	0.79	26.77

Table 4.5: Results of final landmark detection. Error is denoted in mm

4.5 Visualization of final prediction

The image below is a sample from the test set, which represents the mean error. The best and worst prediction is shown in table B.

Sample	MA1	MA2	P	A	Coap	Ao
3	8.5	6.36	8.53	8.83	5.75	9.18


MA1 predicted
MA1 true
MA2 predicted
MA2 true
P predicted
P true

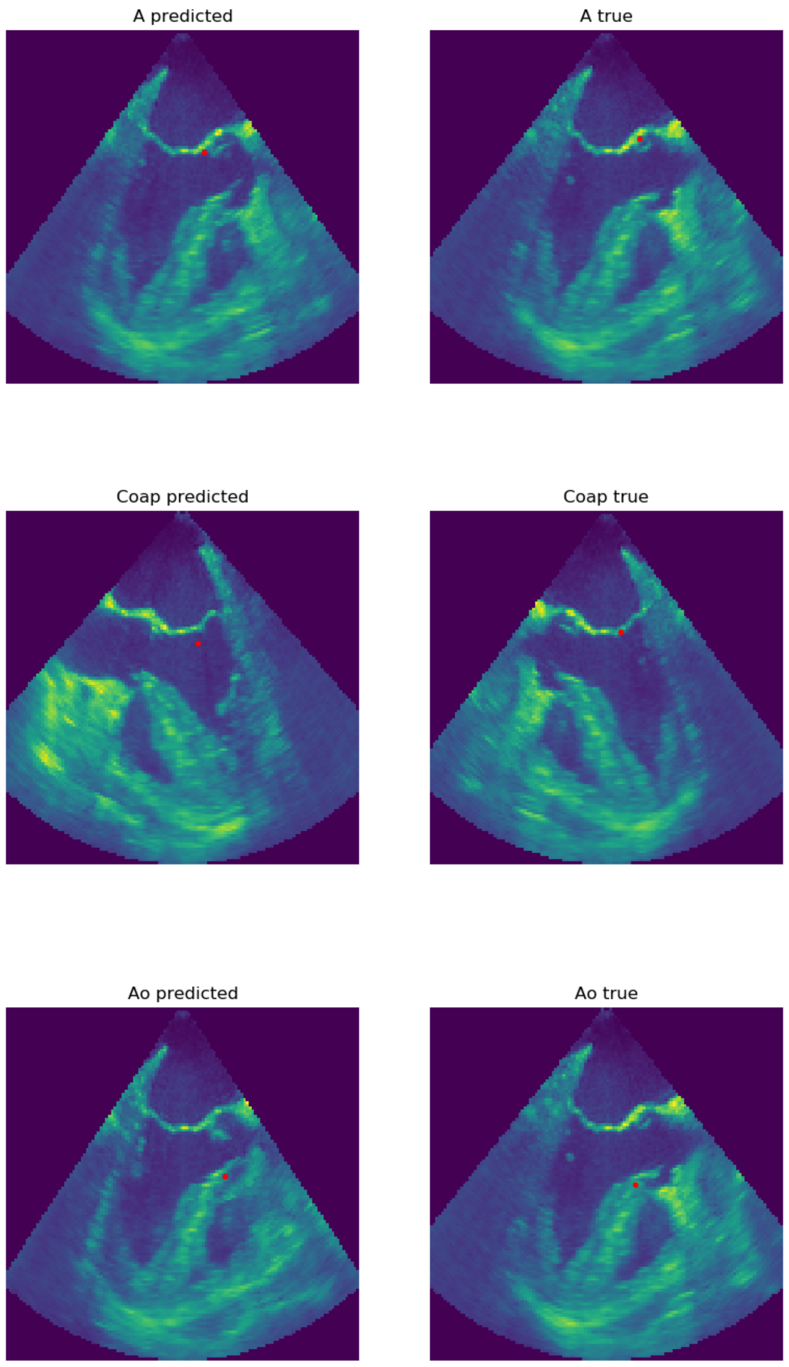


Figure 4.5.2: Image from test set which represent average error

4.6 Effect of patch size

ResNet18 is tested on two different data sets to see if patch size affect the performance of the network. The results for each dataset is shown in table 4.6.

Patch size	MA1	MA2	P	A	Coap	Ao
30x30x30	15.13	14.99	15.43	13.47	10.88	18.49
60x60x60	11.61	11.10	11.79	10.02	7.66	14.23

Table 4.6: Error for each data set. The error is denoted in mm

4.7 Application accuracy

An extra data set consisting of four ultrasound images with matching CT scans were used to evaluate registration accuracy using the identified landmarks. These images are labeled with 5 landmarks. Hence the 'Coap' landmark is excluded. The error of the predicted landmarks in the new data set is shown in table 4.7. The fiducial registration error for each sample is shown in table 4.8. The fiducial registration error for the true landmarks are also included in the same table.

sample	MA1	MA2	P	A	Ao
1	8.342	7.621	8.383	9.362	13.68
2	12.50	10.30	12.65	21.92	40.22
3	4.030	4.279	3.282	4.389	14.83
4	2.666	3.251	4.696	5.587	7.795

Table 4.7: Error data set 2 in mm

Sample	1	2	3	4	Average
FRE predicted	1.21	6.17	2.66	2.24	3.11
FRE true	0.46	1.86	1.11	1.65	1.27

Table 4.8: Fiducial registration error for predicted and true landmarks. The error is denoted in mm.

Chapter 5

Discussion

We have shown that convolutional neural networks can be used to predict the location of anatomical landmarks in 3D ultrasound images. The best performing model was ResNet18, with an average error of 11.073 mm. After applying post-processing to the displacement vectors to predict the final landmark, the average error was 8.78 mm. The purpose of the detected landmarks was to align ultrasound images with CT scans. The fiducial registration error was 3.11 mm, 1.84 mm more than the fiducial registration error for the true landmarks. As shown in figure 5.0.1, the accuracy of the predictions vary. Therefore, the method is not sufficient enough for automatic landmark detection. However, the predictions can be used as suggestions in a semi-automatic landmark detection application. Although the predictions from semi-automatic model might need corrections, it is easier and more efficient than finding the landmarks manually.

In this chapter, we will discuss how different design choices, such as patch size and network architecture, affect the final results, what kind of limitation we are facing, and how our model can be improved as further work.

5.1 Patch size

To examine how the patch size affects the training, two datasets where created. The first dataset was created by extracting 1000 patches of size 30x30x30 pixels from each image, while the second data set was created by extracting 125 patches

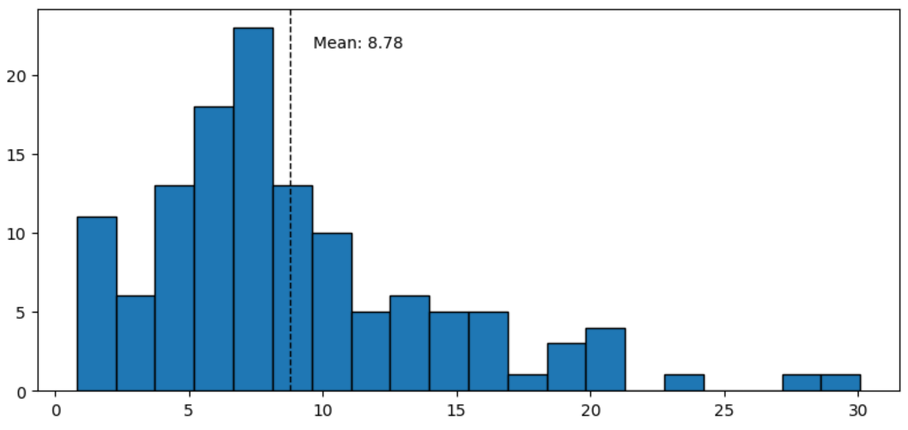


Figure 5.0.1: Error distribution for test set

.

of size 60x60x60. Due to memory constraints, it was not possible to create larger datasets than this. The different patch sizes did not have a significant impact on the final results, but the results provided by the dataset with patches of size 60x60x60 were slightly better. The accuracy of the classification was also better with the larger patches. This is probably because the larger patches better capture the structures of the heart.

5.2 Using convolutional neural networks for detecting anatomical landmarks

Four different tests were performed for using convolutional neural networks for detecting anatomical landmarks. The first tests used "regular" convolutional neural networks for regression without classification or weighted loss to establish a baseline. ResNet and a fully convolutional neural network (FCNN) were used. ResNet performed better than FCNN.

In the second test, the regular MSE loss was replaced with a weighted MSE loss to avoid inaccurate predictions from patches far away from the landmarks to make significant updates on the weights. Weighting the MSE loss decreased the error for both network architectures. This was expected, as this technique has shown to lower errors in similar applications [33, 21]

The third test used multi-task learning by combining regression and classification. A regular MSE loss were used for the regression. This technique decreased the error slightly more than the weighted loss did. Before training the networks for regression, the networks were also trained for classification alone, without achieving any impressive results. Although the accuracy of the classification was still low after applying multi-task learning, combining classification and regression improved the performance of both tasks.

The last test used multi-task learning combined with the weighted loss function. Of all the models, this model performed the best. This method also gave the best results in Noorhut et al.

The ResNet architecture generally performed better in all tasks, which indicates that the model benefits from having more hidden layers and skip connection. In Noorhut et al., they achieved excellent results using the fully connected neural network. Although our problem is quite similar to the problem in this paper, the fully convolutional neural network performed poorly on our data set. Unlike the results in the paper, adding classification or weighted loss to the fully convolutional neural network did not have a significant improvement in the results. A possible explanation for this is the size of their dataset. In the paper, they extracted 10 000 patches of size 72x72x72 pixels from each image. Hence extracting more patches might have improved the performance of the fully convolutional neural network. More reasons for why our results are worse than the results in Noorhut et al. will be discussed in section 5.4.

5.2.1 Deciding final landmark

After finding the model with the lowest average error for the predicted displacement vectors, four methods for deciding the final landmark were tested. Predicting the final landmark from an average point of all patches performed the worst of all three methods. Figure 5.2.2 show the error made by the network against the distance between the patch and an arbitrary landmark. The figure shows that the error increases approximately linearly with the distance, and that predictions from patches far away is increasing the error of the final prediction. This lead us to the next method, which was to let the final landmark be a weighted mean of the predicted points. Weighting patches according to their distance from the landmarks decreased the error.

The third method was to let the landmark be a weighted mean of the predictions from patches containing the landmark. Since the model predicts whether a patch contains a landmark or not, the patches labeled to contain the landmark was used. This reduced the error even further. The last method, which was to weight the predictions classified to contain the landmarks, resulted in the lowest error.

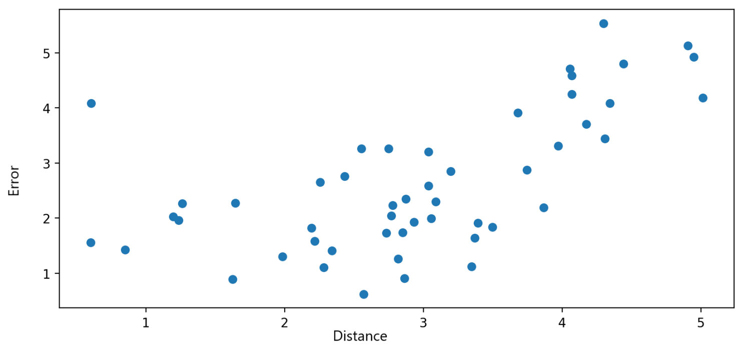


Figure 5.2.2: Error of predicted displacement vectors plotted against the distance between the landmark and the center of the patch

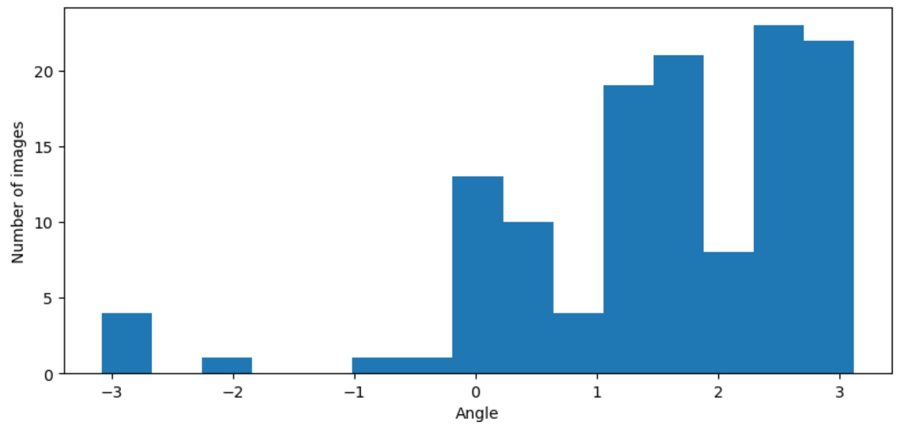
.

5.3 Errors made by the network

To better understand how the model can be improved, the errors made by the network are examined.

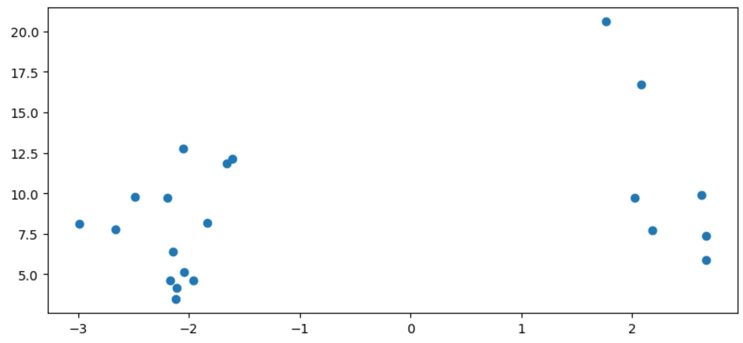
5.3.1 Error in regression

In table B.5 in the appendix, the error for all landmarks in all images are listed. Many of the predictions are close to the true landmark, with some exceptions. In order to better understand what kind of errors the network is making, the displacement vectors are examined more thoroughly. Figure 5.3.4 is a scatter plot of true landmark (black dot) and predicted landmarks (colored dots) from two predictions with small errors. The colour of the predicted landmarks in figure 5.3.4a and 5.3.4b indicates the distance between the patch and the landmark. In a majority of the predictions, patches agree on where the landmark should be, like shown in figure 5.3.4a. In the cases where the predictions are more scattered, the predictions from patches close to the landmark give the most accurate predictions, as shown in figure 5.3.4b.



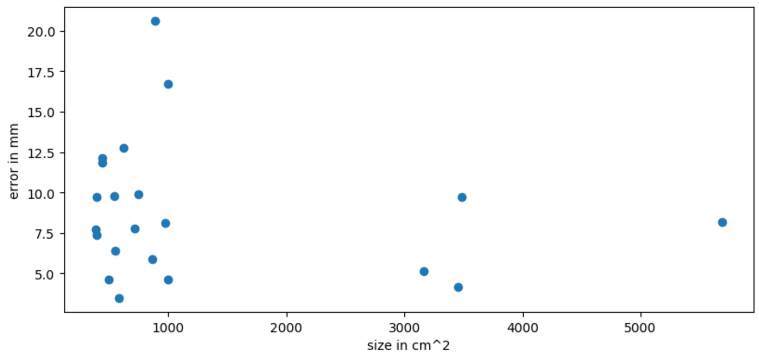
(a) rotations of the different images denoted in radian

.



(b) error plotted against angle of image

.



(c) error plotted against size of image

.

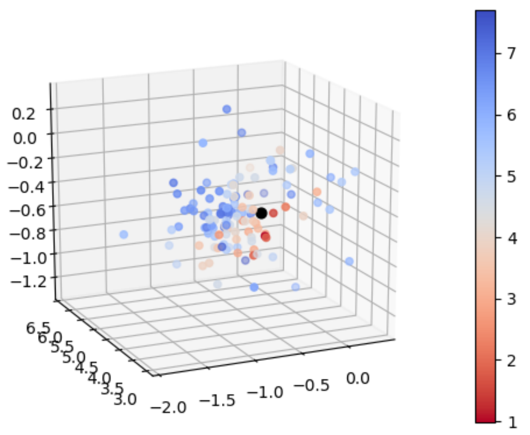
Figure 5.3.3: data set statistics

Figure 5.3.6 shows all predictions of four landmarks in the test image with the worst predictions. From the scatterplot, it is clear to see that the predictions have a low variance. In these cases, the patches close to the landmarks do not necessarily give more accurate predictions than patches further away.

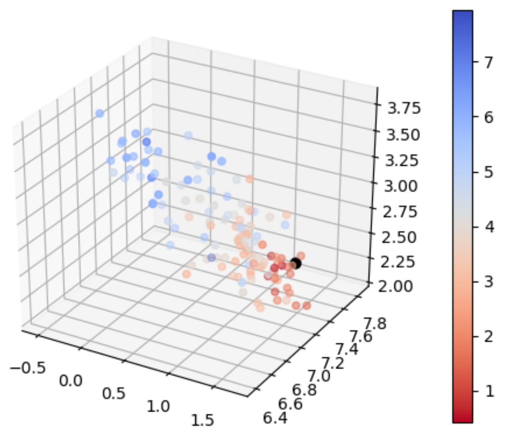
As mentioned, there is a significant variation in the images in the dataset; some of the images are large and show the entire heart, and some images are smaller and show only the valves. The size of the images in the dataset varies from $187cm^3$ to $10192cm^3$, and 75% of the images are less than $2000cm^3$. Hence the smaller images are more represented in the training set. The images also have different orientations. The angle of the two planes containing A, P, Coap and Ao are shown in the histogram in figure 5.3.3a. From the histogram, we see that a majority of the images have the same rotation. To determine whether this affects the results, the average error for each image is plotted against image size and angle. As seen in figure 5.3.3, there is no significant correlation between error an image size, nor angle.

Figure 5.0.1 shows the error distribution in the test set. The mean is marked with a dotted line. The figure shows that the error distribution is right-skewed. The median error was only 7.48, and 61.9 % of the landmarks were less the mean error. Hence the mean error is increased by several outliers. The two images with the largest error in the test set are shown in figure 2.4.2 and 2.4.3 in appendix B.4. A "shadow" is visible at the top of the cone on both images. Moreover, this does not seem to appear on any of the other images in the test set. This artifact might explain why these two images have unusually large errors. Excluding these from the data set would reduce the error by 1 mm. The patch selection algorithm should eliminate patches containing a shadow, as the pixel intensities are zero in these areas. The shadows are close to the valves. Hence the patch selection algorithm discards patches close to the valves and landmarks, while patches further away are approved. Since these patches make poorer predictions, the error is large for landmarks from these images.

Although a predicted landmark has low error, it does not necessarily mean that the prediction is good enough for the application. By looking at the predicted landmarks, it is clear that some of the predicted landmarks are inside the heart chamber, where there is no tissue. This is seen in, for example, the prediction of coap in figure 4.5.2. The error of the prediction is only 5.75 mm, which is below average. Handling points outside tissue is difficult for the CT fusion tool, and these predictions will need corrections from humans.

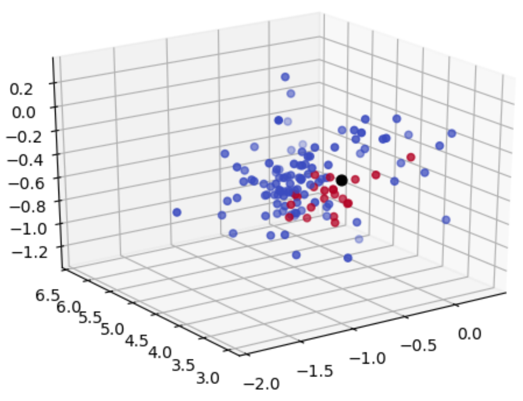


(a)

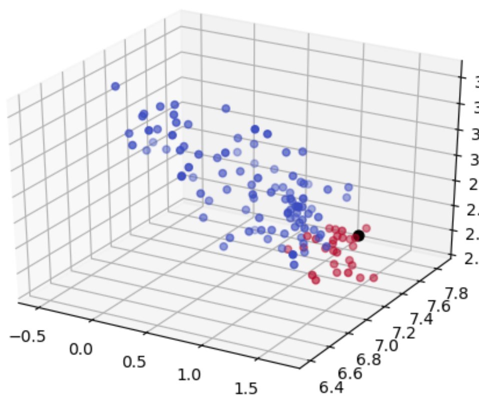


(b)

Figure 5.3.4: Predicted landmarks (colored dots) and true landmark (black dot) for two landmarks in the in the test set. The color of the prediction indicate distance between patch and landmark.

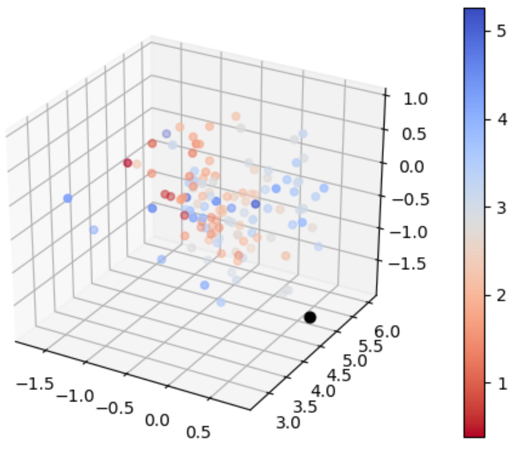


(a)

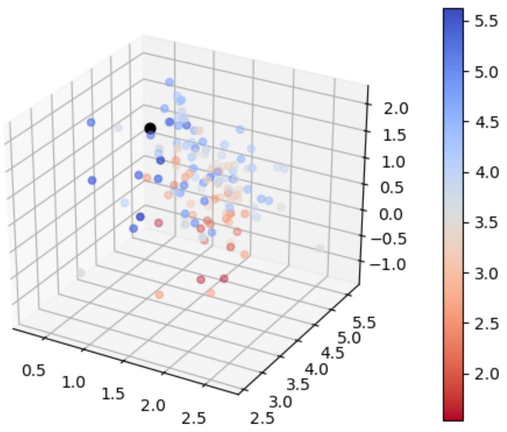


(b)

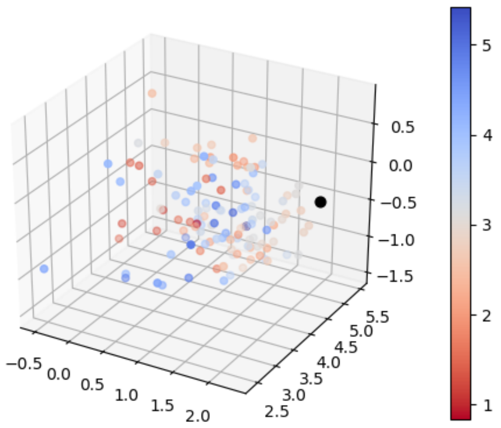
Figure 5.3.5: predicted landmarks (blue and red dots) and true landmark (black dot) for the same landmarks as in figure 5.3.4. Red dots are patches predicted to contain landmark and blue dots are patches predicted to not contain landmark.



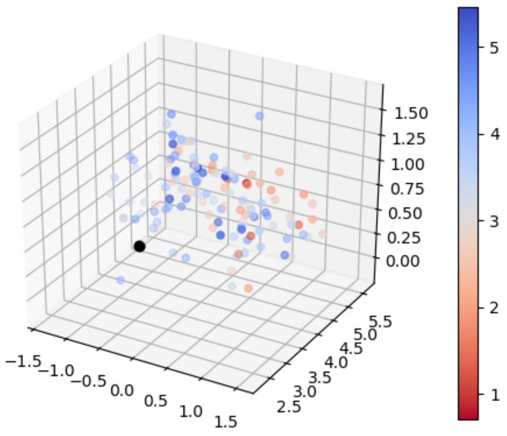
(a) MA1



(b) MA2

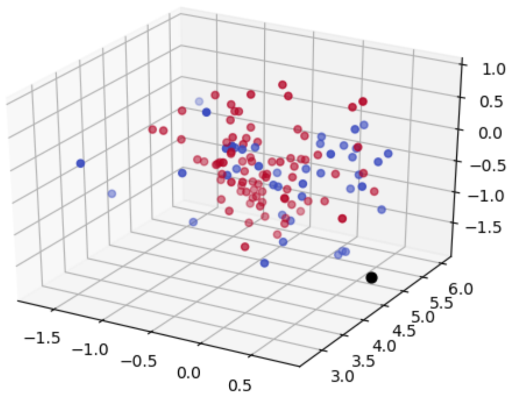


(c) P

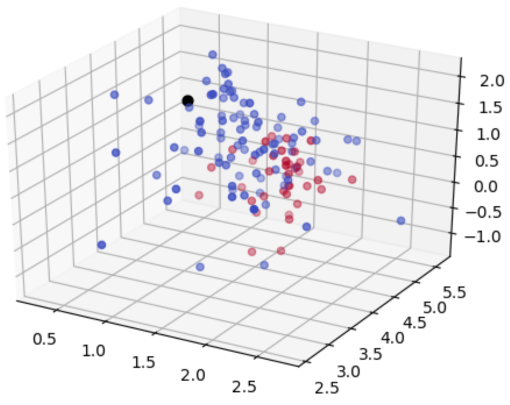


(d) a

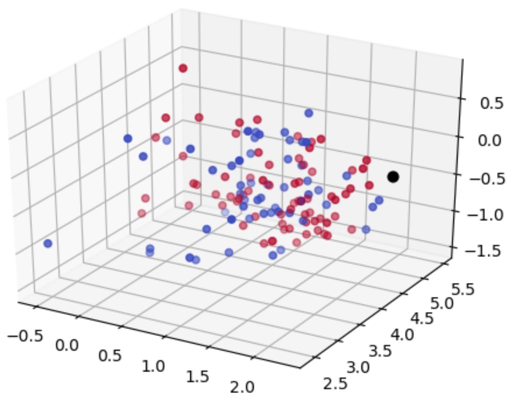
Figure 5.3.6: Predicted landmarks (colored dots) and true landmark (black dot). The colour of the prediction indicate distance between patch and landmark. The predictions are from image number 20 in the testset.



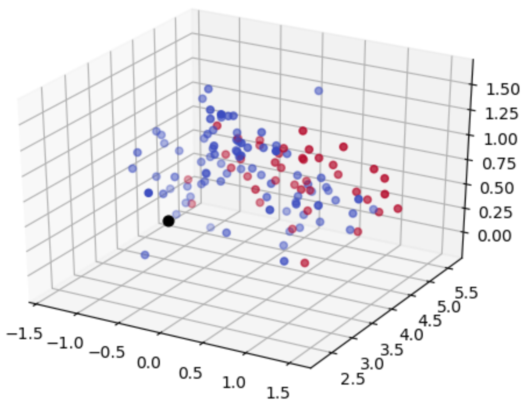
(a) MA1



(b) MA2



(c) P



(d) A

Figure 5.3.7: Predicted landmarks (colored) and true landmark (black dot). Red dots are patches predicted to contain landmark and blue dots are patches predicted to not contain landmark. The predictions are the same as the predictions shown in figure 5.3.6

5.3.2 Error in classification

Achieving high accuracy in the classification of patches turned out to be a difficult task. Before combining classification and regression, the performance of the classification was tested alone. Combining regression and classification improved the performance of the classification as well as the regression. Although using multi-task learning increased the performance, the precision and specificity remain low. Hence many of the patches without landmarks are classified to contain one and patches containing landmarks are assigned to the wrong class. From the scatter-plot in figure 5.3.5b and 5.3.7, we see that the majority of the wrong predictions are patches which are close to the landmarks. Increasing the threshold for classification from 0.5 to 0.7 had a positive effect. However, increasing the threshold further eliminated too many of the predictions.

5.4 Limitations

In the process of creating a model for automatic landmark detection in 3D cardiovascular images, limitations which affect the model's ability to produce precise predictions were discovered:

5.4.1 Human error

A random selection of 19 images from the dataset has been selected and relabeled with the six landmarks a second time. The differences between the true landmarks represent the human error. The average human error for the six landmarks is shown in table 5.1. The difference between the two rounds of labeling are visualized in figure 5.4.8. The human performance provides a measurement of how large error is expected from the model. For the dataset used in this thesis, the average human error is 4.422 mm. It is challenging to train a model to have a smaller error than humans relabeling data. To exceed the human error, additional methods must be applied, like having the dataset relabeled by multiple experts.

Error	MA1	MA2	P	A	Coap	Ao
Mean	4.358	5.090	4.692	4.0176	4.111	4.267
Max	9.029	8.588	11.36	6.341	8.350	9.584
Min	0.402	1.683	0.686	1.351	0.986	1.898

Table 5.1: average, minimum and maximum human error for all six landmarks denoted in mm

Human error also affects the accuracy of the classification. Since the position of the landmark vary, which patches containing a landmark may also vary. Hence the large human error indicates that our training set contains misclassifications. This might explain the low classification specificity and why the network tends to classify patches close to landmarks to contain the landmark.

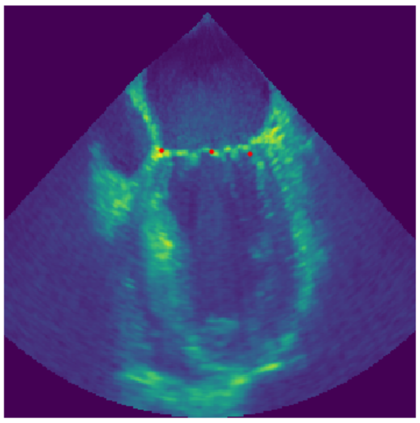
5.4.2 Working with ultrasound images

Comparing the results with Noorhut et al., The error from our networks was larger, although the tasks were similar. The different data might be a reason for this. For the method in Norrhut et al., they used CTA scans. CTA scans provide images that are more detailed than ultrasound images. Ultrasound images also contain more noise compared to the CTA scans. The noise in ultrasound images obscures the structures of the heart, which made it hard for even humans to detect the landmarks, as shown in section 5.4.1.

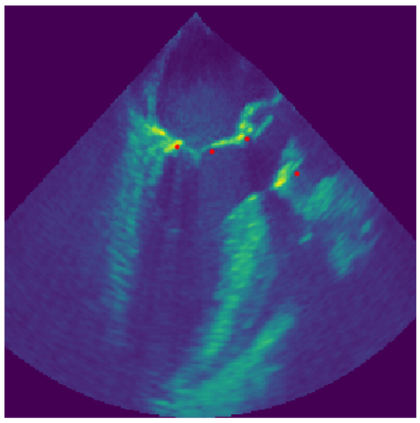
The effect of the image quality has also been studied in "Understanding how image quality affects deep neural networks" [7] by Samuel Dodge and Lina Karam. They examined how different types of noise affected the results of classification. Several network architectures were tested. In the paper, they discovered that all convolutional neural network architectures were sensitive to noise, and adding noise to images decreased the accuracy. When looking at the response of the different layers for a regular image and compared it to responses of the same image with Gaussian noise added, they discovered that small changes in the response in the first layers caused significant changes in deeper layers, causing the errors. Assuming that these results also apply to other tasks, like regression in our case, it is fair to believe that the noise in the ultrasound images might limit the results and explain why our overall results are worse than the results in Noorhut et al. However, in Andreassen et al., they achieved an error of only 2 mm in landmark detection in 3D ultrasound images. Hence, improving our model and obtain a lower error should be possible.

5.5 Further work

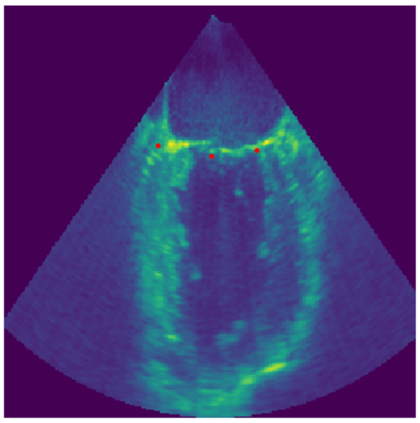
There exist many methods for using convolutional neural networks for detecting anatomical landmarks. We have only investigated some of them in this thesis, and will review some methods that might have improved the results.



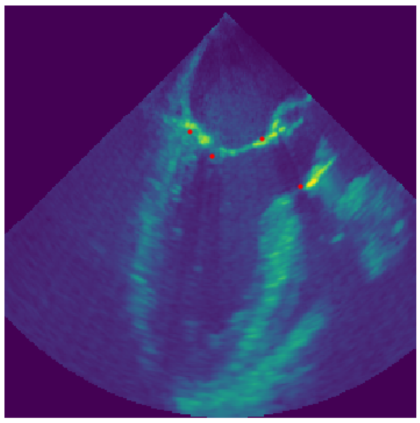
(a)



(b)



(c)



(d)

Figure 5.4.8: Two different labeling of the same image. a and b are from the original dataset, while c and d are from the relabeled images. Error for the different points: MA1: 6.03 mm, MA2 7.05 mm, P: 11.36 mm, A: 3.41 mm, Coap: 8.35 mm, Ao: 5.95 mm

5.5.1 improving the dataset

Due to memory constraints, we were only able to extract a limited number of patches from each image. As the performance of a convolutional neural network depends on having enough data for training, increasing the number of extracted patches would probably improved performance.

5.5.2 Improving the model

In this thesis, only patch-based methods have been used, and network architectures that use whole images as input has not been considered. A downside with using a patch-based method is that these methods can be more time-consuming. First, patches need to be extracted from the image. Then all patches must be sent through the network before the final predicted landmark is calculated. In "Detecting Anatomical Landmarks From Limited Medical Imaging Data Using Two-Stage Task-Oriented Deep Neural Networks", they suggested using U-net with whole images as input. U-net has proven to achieve high accuracy in several computer vision task with 3D medical data and limited training samples. This network is most commonly used for image segmentation, but have also been tested for landmark detection. The network first downsample the data using pooling layers, and afterward upsampled again to the original size, which gives the network the u-shape, as shown in figure 5.5.9. Instead of returning landmarks in the form of coordinates, the network returns one heatmap for each point with the same size as input. The values in the heatmap represent the probability that the corresponding pixel in the input image is a landmark.

Another drawback of using a patch-based method is that these networks can only model local information, such as the correlation between patches and landmarks inside the patch [33]. When using the whole image as input and combining this with a network where the deep layers have a large receptive field, like U-net, the network can capture more global information. A drawback of using networks with whole images as input, such as U-net, is that these networks usually need more layers to get a large field of view. This results in many weights that need training. It should also be mentioned that the paper using this method had more training data available; hence using this method might require more data.

In terms of applying multi-task learning to improve the performance of the network, only patch classification has been used as a secondary task. Several other tasks could have been used instead of classification. Other tasks are for example to predict the position of the input patch or to estimate the two planes the landmarks were extracted from. Another alternative is to add these tasks as third and fourth tasks.

A last adjustment that could have improved the performance is to use more information from the data other than only the images. This information could, for example, be the coordinates to the patches or size of the image. This would have been added to the input as a fourth dimension. The additional information could have helped the network to understand the input data better. Other information that could have been utilized is the geometric relation between the landmarks. The geometric relations could have been used in the post possessing. As the landmarks belong to two planes, the network could have used the predicted landmarks to estimate these two planes, and later project the predicted landmarks down on these planes.

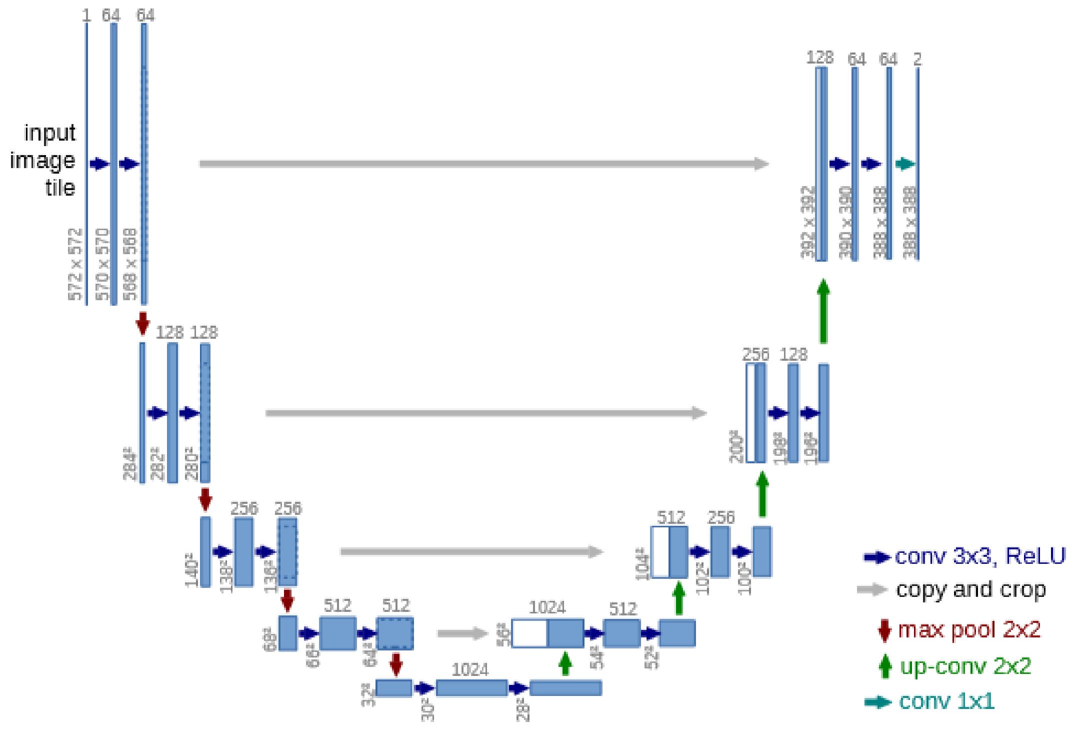


Figure 5.5.9: Original U-net from Ronneberg et al. [25]

.

5.5.3 Improving the loss function

For the regression, only mean squared error was used as loss function. Mean square error is sensitive to outliers, So an alternative loss function could have been L1 loss:

$$L1(x, y) = |x - y|$$

Another good alternative to mean squared error is smooth L1 loss, which is a combination of L1 loss and mean squared error.

$$\text{smooth L1 loss}(x, y) = \begin{cases} 0.5(x-y)^2 & \text{if } |x-y| < 1 \\ |x-y| - 0.5 & \text{otherwise} \end{cases}$$

This loss function is better to use if the correct values have a large absolute value. The squared error is used when the absolute error is less than one and an absolute error otherwise, the smooth L1 loss is more likely to avoid exploding gradients.

5.5.4 Improving methodology for evaluating the model

From the human error in section 5.4.1, we know that the coordinates of the true landmarks are highly inaccurate. As shown in table 5.1, the coordinates of a landmark could potentially vary up to 1 cm. Hence evaluating the model by using Euclidean distance might not be the best option. An alternative methodology for evaluating the model is the percentage of correct keypoints. The method finds the percentage of predicted landmarks within a reasonable distance from the true landmark. Using this method requires a threshold for how large the distance between true landmark and predicted landmark can become, for the prediction to be good enough for the application. If a prediction is within this threshold, the prediction is considered correct. The model is evaluated based on the percentage of correct keypoints (PCK):

$$\text{PCK} = \frac{\#\text{ Correct predictions}}{\#\text{ predictions}}$$

The challenge of using this method is to decide the threshold for labeling the prediction as correct or not. For this application, the threshold is unknown, and this threshold might also vary for each landmark, but an acceptable value to use could be the maximum human error, which is 11.36 mm.

For evaluating the accuracy of the CT-fusion tool, only the fiducial registration error is used. However, the target registration error (TRE) should also have been included to assess the application accuracy. TRE denotes how far a paired points without a landmark are apart after registration. TRE is given by the same equation as FRE:

$$TRE = ||T(l_{us}) - l_{ct}||$$

where l_{us} is landmarks from the ultrasound image, and l_{ct} is landmarks from the CT scan, and T is not estimated using l_{us} and l_{ct}. Hence, an additional set of points is needed.

5.5.5 Improving post-processing

Some of the landmarks are predicted to be in areas where there is no tissue. This can be avoided by using information from patches and the belonging prediction during post-processing. Landmark predictions outside tissue can be automatically identified by looking at the pixel intensities surrounding the predicted coordinate. If the pixel intensities are low, the predicted landmark is in the blood pool. If the predicted landmark is identified to be outside the tissue, the landmark can be re-predicted. A solution for re-predictions and avoiding predictions outside tissue could be to eliminate all predictions which point outside tissue.

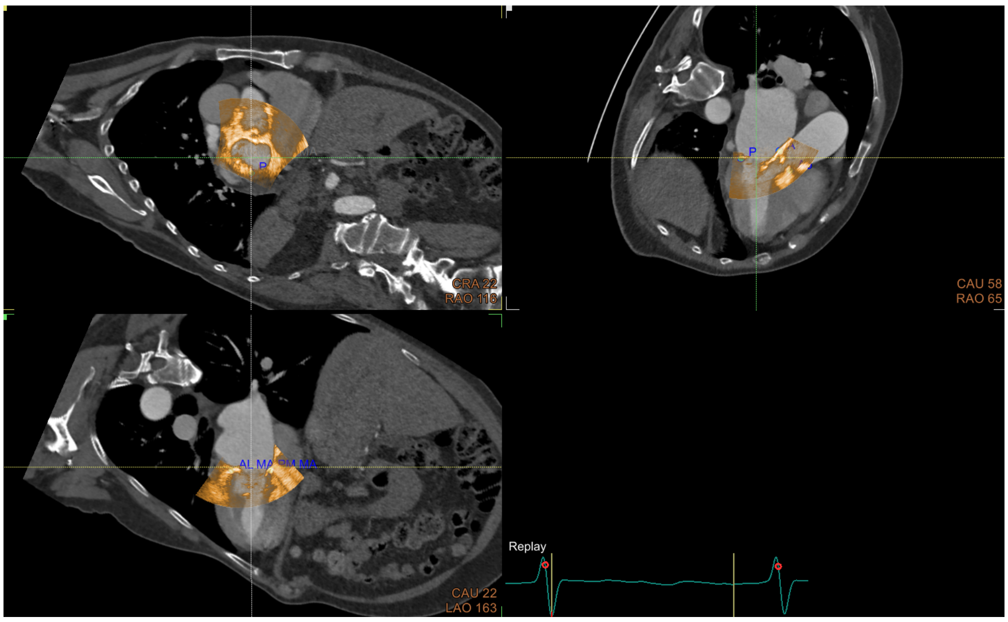
5.6 CT fusion tool

The purpose of the six landmarks was to align the ultrasound image with a CT scan. Many of the predictions are close to the true landmarks, and the model can be used for semi-automatic landmark detection. In practice, a semi-automatic model means that the predictions are suggestions to the user. The landmarks will need corrections for the tool to function optimally. However, this is less time-consuming than finding the landmarks manually. In addition, the fiducial error is low, with an average error of only 3.11 mm, which is only 1.84 mm worse than the true error.

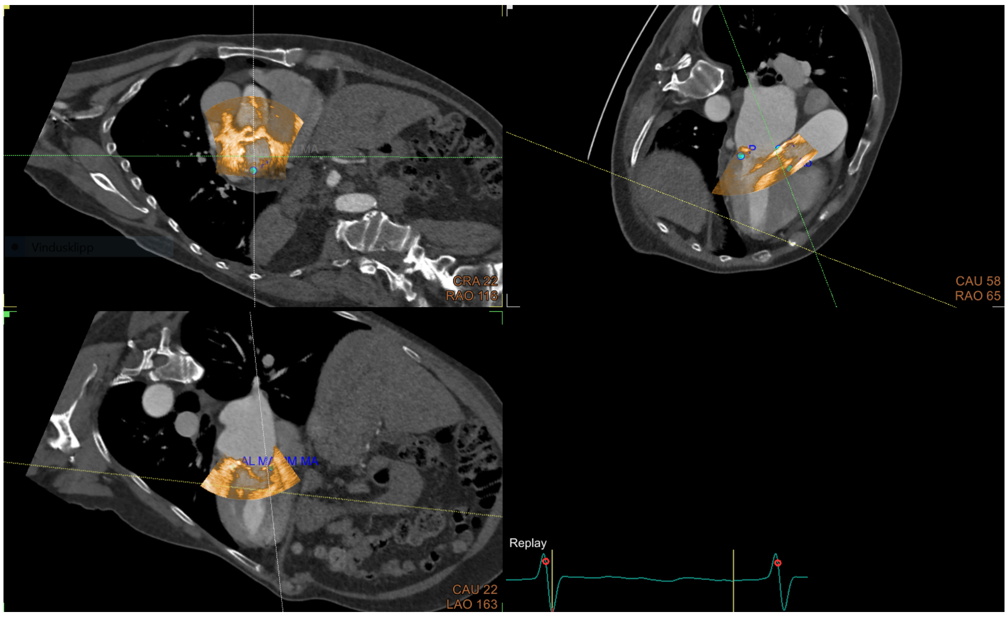
Sample 1 from the new dataset was used to test the CT-fusion tool. The error for each of the predicted landmark in this sample was:

sample	MA1	MA2	P	A	Ao
1	8.342 mm	7.621 mm	8.383 mm	9.362 mm	13.68 mm

The error of the landmarks in sample 1 was close to the average error for the test set. The fiducial registration error for the predicted landmarks was 1.21 mm and 0.46 mm for the true. A visualization of the true and predicted landmarks in the CT-fusion tool are shown in 5.6.10. The difference between the images in the CT-fusion tool is small. Hence he CT fusion tool works with predictions from our model.



(a) True



(b) Predicted

Figure 5.6.10: CT-fusion tool for predicted and true landmarks

Chapter 6

Conclusion

Through this project, we have shown that convolutional neural networks can be used for detecting anatomical landmarks in 3D cardiovascular images. This was done by using a patch-based method. Two network architectures were tested, ResNet and a fully convolutional neural network. Although the fully connected neural network had proven to perform well in similar applications, ResNet provided the best results. For the patch-based method, two datasets with different patch sizes were used. Results showed that the different patch sizes affected the error of the displacement vectors, and patches of size 60x60x60 pixels performed the best.

Two techniques were tested for improving the performance of the models. The first technique was to include multi-task learning with hard parameter sharing and classification as a secondary task. The second was to weight the loss function. Each of the techniques improved the performance of the network, but combining the two led to the lowest error. The average error for the displacement vectors were then 11.073 mm.

After finding the displacement vectors created by the model, we experimented with different methods for predicting the final landmark. As predictions from patches far away from the landmarks were unreliable, only using the predictions that were close to the landmarks resulted in the lowest error. The average error of the final prediction was 8.78 mm. The error limits our model to a semi-automatic landmark detection solution for the CT fusion tool. Furthermore, expanding the post-processing to correct coordinates which misses on tissue would positionally improve the predictions without needing to train a new model.

Several limitations for this application were discovered. The first limitation was the small dataset. Applying more techniques for data augmentation is yet to be tested. The second limitation is the noise in the ultrasound images obscures the structures, which makes it hard for both networks and humans to detect the landmarks. We discovered that the human error was 4.422 mm. Achieving a smaller error than this is not possible with the current data set.

Appendix

Appendix A

Results from regression

Test 1: Regression with weighted MSE loss

	MA1	MA2	P	A	Coap	Ao
ResNet	11.55	12.30	11.88	10.94	8.78	15.47
FCNN	31.76	26.21	28.80	27.14	22.95	35.53

Table A.1: Results for regression with weighted MSE loss. Average euclidean error in mm.

Test 2: Regression with regular MSE loss and classification

	MA1	MA2	P	A	Coap	Ao
ResNet	11.67	11.60	11.95	10.57	8.01	14.81
FCNN	24.41	25.04	24.71	24.54	20.53	29.16

Table A.2: Results for regression with regular MSE loss and classification. Average euclidean error in mm.

Test 3: Regression with weighted MSE loss and classification

	MA1	MA2	P	A	Coap	Ao
ResNet	11.61	11.10	11.79	10.02	7.66	14.23
FCNN	20.67	22.37	23.24	20.69	18.87	26.75

Table A.3: Results for regression with weighted MSE loss and classification. Average euclidean error in mm.

Appendix B

Results from landmark detection

B.1 Final landmark detection

Mean of all predictions

	MA1	MA2	P	A	Coap	Ao
Mean	9.815	8.900	9.912	8.233	5.848	12.053
Min	1.507	1.210	3.319	1.493	1.337	2.851
Max	19.542	26.354	19.157	20.771	11.849	33.768

Table B.1: Mean of all predictions. Error in mm

Weighted mean of all predictions

	MA1	MA2	P	A	Coap	Ao
mean	9.734	8.914	9.884	8.153	5.740	11.790
Min	1.747	1.143	2.885	1.753	1.211	3.022
Max	19.741	26.186	19.458	20.82	11.080	33.909

Table B.2: Weighted mean of all predictions. Error in mm

Predict using patches containing the landmark

	MA1	MA2	P	A	Coap	Ao
Mean	9.724	9.139	9.844	7.915	5.301	11.373
Min	2.021	1.666	1.489	2.987	0.811	2.094
Max	20.676	23.549	20.160	20.128	10.702	30.084

Table B.3: Using classification for deciding final landmark. Error in mm

Weighted mean of patches containing the landmark

	MA1	MA2	P	A	Coap	Ao
Mean	9.655	9.070	9.742	7.844	5.261	11.105
Min	2.119	1.612	1.182	2.908	0.792	2.077
Max	20.845	23.118	19.014	19.014	10.665	26.776

Table B.4: Weighted mean of patches containing the landmark. Error in mm

B.2 Results for all images

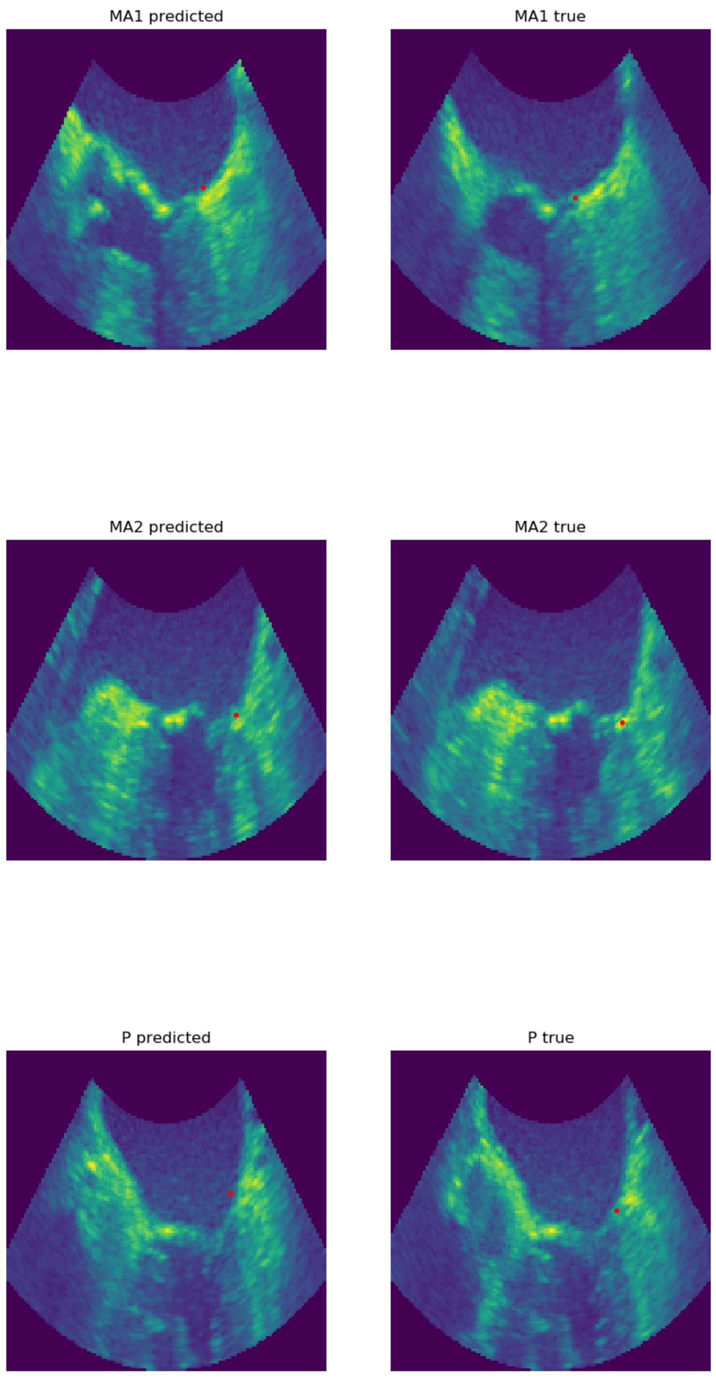
Image	MA1	MA2	P	A	Coap	Ao
1	5.38	5.21	1.18	7.3	1.59	3.81
2	8.3	7.02	6.52	11.06	5.58	20.44
3	8.5	6.36	8.53	8.83	5.75	9.18
4	18.08	19.39	15.91	16.21	4.69	26.78
5	7.77	2.21	9.75	7.21	10.66	11.19
6	14.76	15.3	18.73	8.44	9.98	7.71
7	15.77	7.96	14.05	6.8	7.51	5.83
8	7.99	1.61	7.46	7.57	1.59	12.03
9	10.5	13.64	12.54	7.05	5.73	7.92
10	7.65	7.02	3.7	2.91	2.68	6.31
11	8.58	12.16	7.1	12.88	2.04	16.02
12	10.85	11.43	16.51	5.16	10.5	16.4
13	4.59	2.82	3.8	4.7	0.79	4.81
14	6.9	5.26	5.66	7.23	3.82	5.59
15	6.77	10.01	7.12	6.62	4.0	11.98
16	8.92	3.86	3.08	5.53	2.11	4.68
17	11.12	14.71	18.46	3.13	10.44	13.57
18	12.15	6.26	8.22	5.94	6.15	7.31
19	5.24	5.49	9.28	4.04	1.72	2.08
20	20.85	23.12	19.01	19.01	8.84	26.43
21	2.12	9.61	7.99	7.1	4.32	13.17

Table B.5: Results for all images in the test set

B.3 Best results

The image with the best predictions is shown below. the error is:

Image	MA1	MA2	P	A	Coap	Ao
13	4.59	2.82	3.8	4.7	0.79	4.81


MA1 predicted
MA1 true
MA2 predicted
MA2 true
P predicted
P true

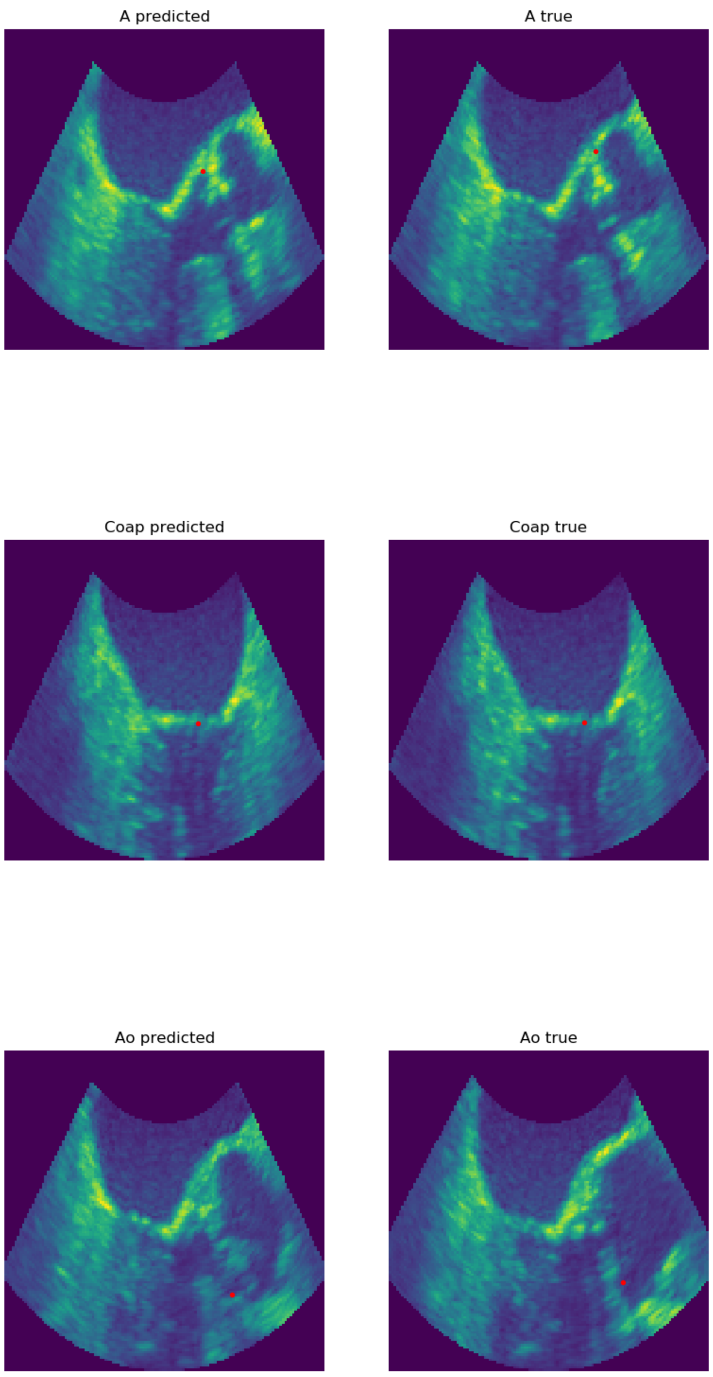
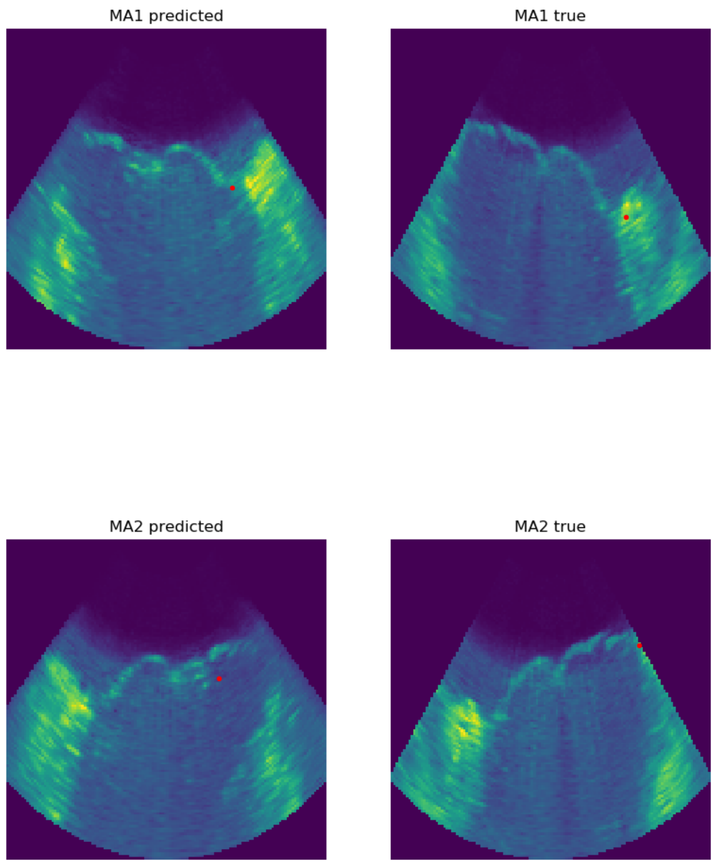


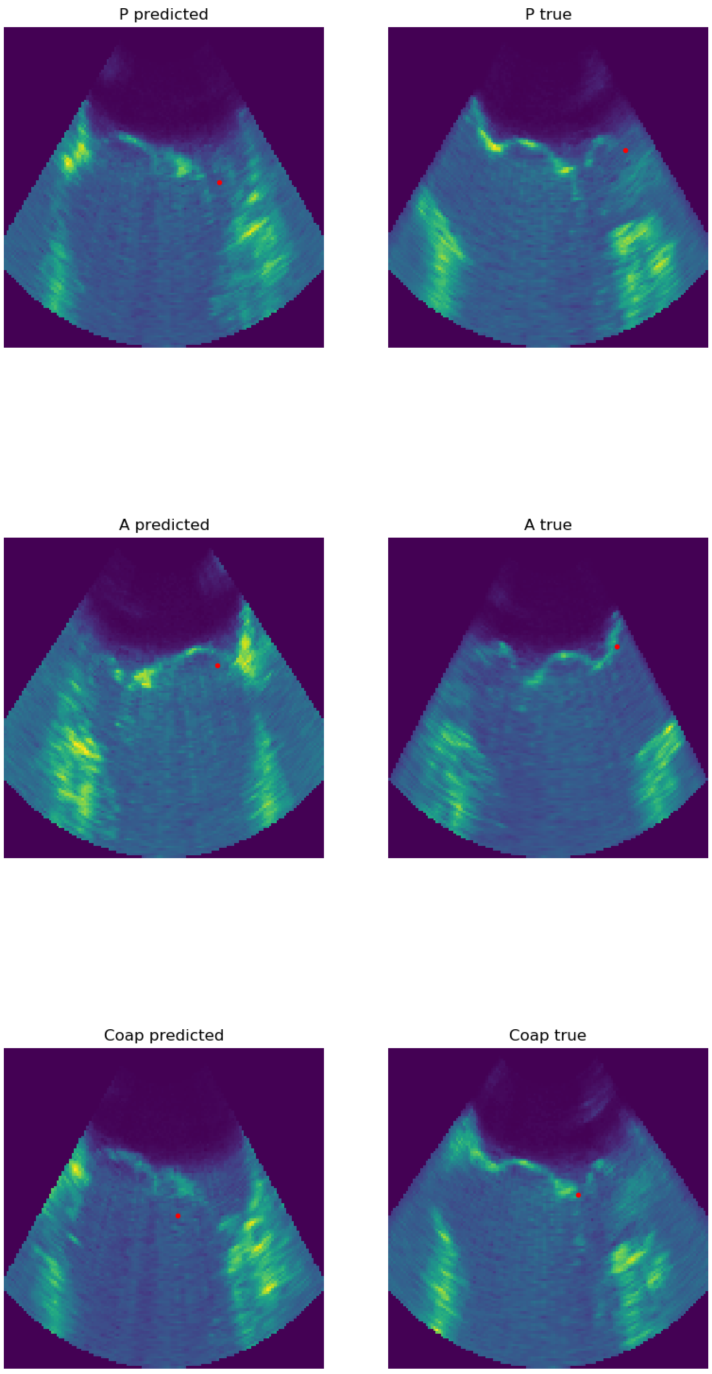
Figure 2.3.1: Best prediction

B.4 Worst results

From the test set, there are two images with error wich pointed out as particularly large. These images and the error of the predictions is shown below:

Image	MA1	MA2	P	A	Coap	Ao
20	20.85	23.12	19.01	19.01	8.84	26.43




P predicted
P true
A predicted
A true
Coap predicted
Coap true

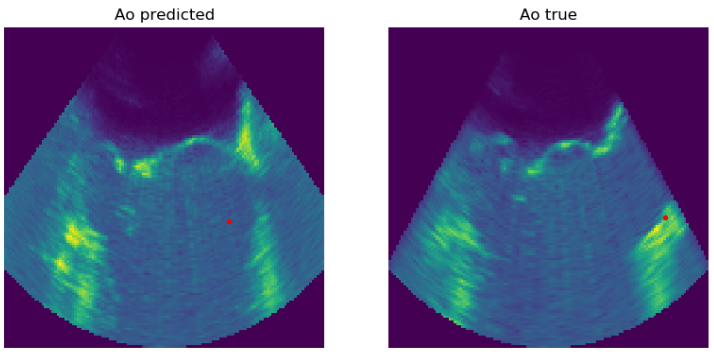
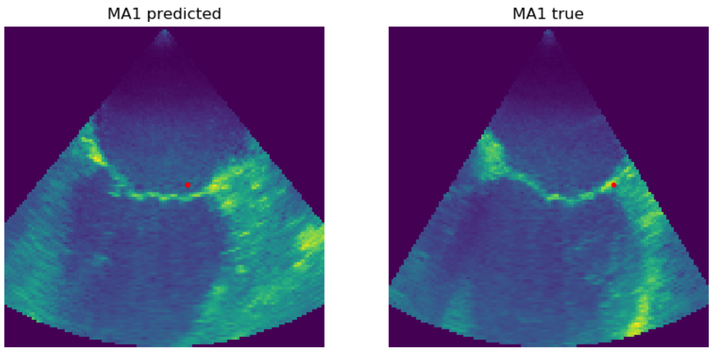
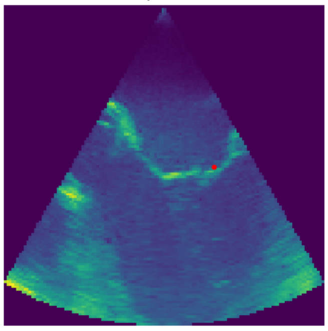
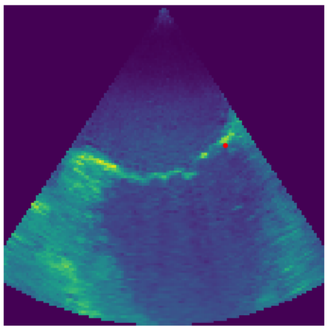


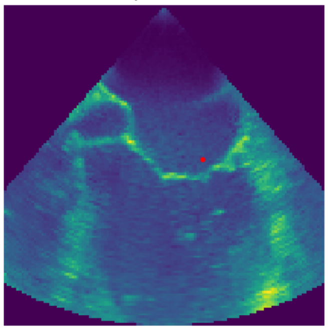
Figure 2.4.2: Worst prediction 1

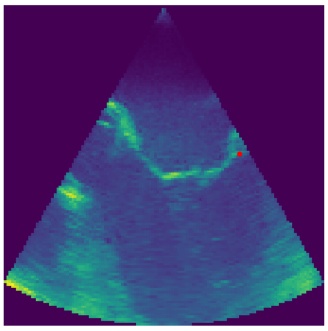
Image	MA1	MA2	P	A	Coap	Ao
4	18.08	19.39	15.91	16.21	4.69	26.78

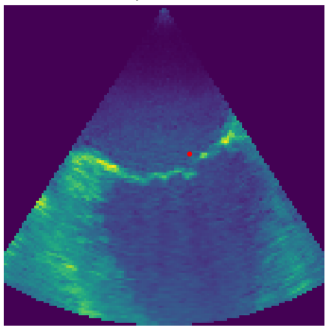


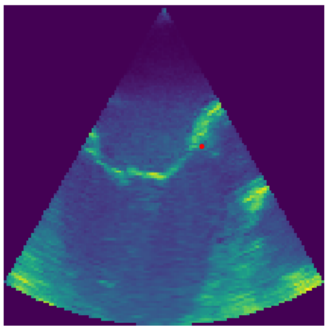

MA2 predicted


MA2 true


P predicted


P true


A predicted


A true

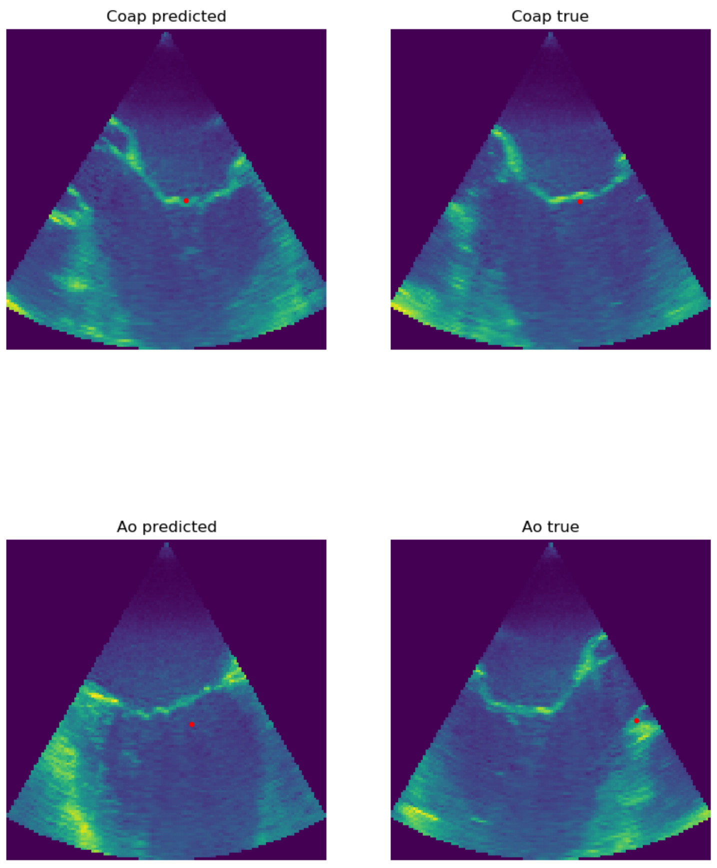


Figure 2.4.3: Worst prediction 2

Appendix C

Results classification

Test 2: Regression with regular MSE loss and classification

	MA1	MA2	P	A	Coap	Ao
Precision	0.3964	0.1	0.2833	0.2833	0.5	0.1
Sensitivity	0.1393	0.0065	0.0494	0.0496	0.0937	0.0166
Specificity	0.9928	1.0	0.9943	0.9980	0.9980	1.0

Table C.1: Results of classification using ResNet

Test 3: Regression with weighted MSE loss and classification

	MA1	MA2	P	A	Coap	Ao
Precision	0.354	0.631	0.420	0.415	0.710	0.485
Sensitivity	0.988	0.884	0.944	0.994	0.890	0.929
Specificity	0.770	0.827	0.728	0.838	0.720	0.866

Table C.2: Results of classification using ResNet

Bibliography

[1] *An Overview of Multi-Task Learning for Deep Learning*. en. https://ruder.io/multi-task/. Library Catalog: ruder.io. May 2017.

[2] B. S. Andreassen et al. “Mitral Annulus Segmentation Using Deep Learning in 3-D Transesophageal Echocardiography”. In: *IEEE Journal of Biomedical and Health Informatics* 24.4 (2020), pp. 994–1003.

[3] *Best Practice Guide - Deep Learning*. https://www.researchgate.net/publication/332190148_Best_Practice_Guide_-_Deep_Learning/references.

[4] Özgün Çiçek et al. “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation”. In: *arXiv:1606.06650 [cs]* (June 2016). arXiv: `1606.06650 [cs]`.

[5] Wikimedia Commons. *File:Diagram of the human heart (cropped).svg — Wikimedia Commons, the free media repository*. [Online; accessed 6-June-2020]. 2020. URL: `https://commons.wikimedia.org/w/index.php?title=File:Diagram_of_the_human_heart_(cropped).svg&oldid=415703023`.

[6] *CS231n Convolutional Neural Networks for Visual Recognition*. http://cs231n.github.io/convolutional-networks/.

[7] S. Dodge and L. Karam. “Understanding how image quality affects deep neural networks”. In: *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*. 2016, pp. 1–6.

[8] *Early Detection of Cardiovascular Disease - the Future of Cardiology?* https://www.escardio.org/Journals/E-Journal-of-Cardiology-Practice/Volume-4/vol4n19-Title-Early-detection-of-cardiovascular-disease-the-future-of-cardi.

[9] Mark Everingham et al.
"The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338.
ISSN: 0920-5691, 1573-1405. DOI: `10.1007/s11263-009-0275-4`.
URL: `http://link.springer.com/10.1007/s11263-009-0275-4` (visited on 05/27/2020).

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* `http://www.deeplearningbook.org`. MIT Press, 2016.

[11] Tushar Gupta. *Deep Learning: Feedforward Neural Network*. en.
https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7.
Dec. 2018.

[12] Le Hou et al. "Patch-Based Convolutional Neural Network for Whole Slide Tissue Image Classification".
In: *Proceedings. IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016 (2016), pp. 2424–2433. ISSN: 1063-6919.
DOI: `10.1109/CVPR.2016.266`.

[13] Matthew HutsonMay. 3, 2018, and 11:15 Am.
AI Researchers Allege That Machine Learning Is Alchemy. May 3, 2018.
URL: `https://www.sciencemag.org/news/2018/05/ai-researchers-allege-machine-learning-alchemy` (visited on 06/09/2020).

[14] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift".
In: *arXiv:1502.03167 [cs]* (Mar. 2015). arXiv: `1502.03167 [cs]`.

[15] Trond Linjordet and Krisztian Balog.
"Impact of Training Dataset Size on Neural Answer Selection Models".
In: *arXiv:1901.10496 [cs]* (Jan. 2019). arXiv: `1901.10496 [cs]`.

[16] Ping Luo et al.
"Towards Understanding Regularization in Batch Normalization".
In: *arXiv:1809.00846 [cs, stat]* (Apr. 2019).
arXiv: `1809.00846 [cs, stat]`.

[17] Wenjie Luo et al. "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks". In: *arXiv:1701.04128 [cs]* (Jan. 2017).
arXiv: `1701.04128 [cs]`.

[18] Wenjie Luo et al. "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks". en. In: (), p. 9.

[19] Kameswari Maganti et al.
"Valvular Heart Disease: Diagnosis and Management".
In: *Mayo Clinic Proceedings* 85.5 (May 2010), pp. 483–500.
ISSN: 0025-6196. DOI: 10.4065/mcp.2009.0706.

[20] Michael Nielsen. "Neural Networks and Deep Learning". en. In: (), p. 224.

[21] Julia M. H. Noothout et al.
CNN-based Landmark Detection in Cardiac CTA Scans. 2018.
arXiv: 1804.04963 [cs.CV].

[22] Keiron O'Shea and Ryan Nash.
"An Introduction to Convolutional Neural Networks".
In: *arXiv:1511.08458 [cs]* (Nov. 2015). arXiv: 1511.08458 [cs].

[23] Ravindra Parmar. *Common Loss Functions in Machine Learning.* en.
https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23.
Sept. 2018.

[24] Priscilla J. Peters and Sean Reinhardt.
"The Echocardiographic Evaluation of Intracardiac Masses: A Review". eng.
In: *Journal of the American Society of Echocardiography: Official Publication of the American Society of Echocardiography* 19.2 (Feb. 2006), pp. 230–240. ISSN: 1097-6795. DOI: 10.1016/j.echo.2005.10.015.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox.
"U-Net: Convolutional Networks for Biomedical Image Segmentation".
In: *arXiv:1505.04597 [cs]* (May 2015). arXiv: 1505.04597 [cs].

[26] Pablo Ruiz. *Understanding and Visualizing ResNets.* en.
https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8.
Apr. 2019.

[27] Rutger Ruizendaal.
Deep Learning #3: More on CNNs & Handling Overfitting. en.
https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting-2bd5d99abe5d. Library Catalog: towardsdatascience.com.
Oct. 2018.

[28] Yang S. *An Introduction to Gradient Descent.* en.
https://towardsdatascience.com/an-introduction-to-gradient-descent-c9cca5739307. Library Catalog: towardsdatascience.com.
Nov. 2019.

[29] Nitish Srivastava et al.
"Dropout: A Simple Way to Prevent Neural Networks from Overfitting".
In: *The Journal of Machine Learning Research* 15.1 (Jan. 2014),
pp. 1929–1958. ISSN: 1532-4435.

[30] *The Vanishing Gradient Problem - Towards Data Science.*
https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484.

[31] P. Viola and M. Jones.
"Rapid Object Detection Using a Boosted Cascade of Simple Features".
In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.*
2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. Vol. 1.
Kauai, HI, USA: IEEE Comput. Soc, 2001, pp. I-511-I–518.
ISBN: 978-0-7695-1272-3. DOI: `10.1109/CVPR.2001.990517`.
URL: `http://ieeexplore.ieee.org/document/990517/` (visited on 05/27/2020).

[32] Samir S. Yadav and Shivajirao M. Jadhav. "Deep Convolutional Neural Network Based Medical Image Classification for Disease Diagnosis".
In: *Journal of Big Data* 6.1 (Dec. 17, 2019), p. 113. ISSN: 2196-1115.
DOI: `10.1186/s40537-019-0276-2`.
URL: `https://doi.org/10.1186/s40537-019-0276-2`.

[33] J. Zhang, M. Liu, and D. Shen.
"Detecting Anatomical Landmarks From Limited Medical Imaging Data Using Two-Stage Task-Oriented Deep Neural Networks". In: *IEEE Transactions on Image Processing* 26.10 (Oct. 2017), pp. 4753–4764.
ISSN: 1057-7149. DOI: `10.1109/TIP.2017.2721106`.

[34] Jun Zhang et al. "Automatic Craniomaxillofacial Landmark Digitization via Segmentation-Guided Partially-Joint Regression Forest Model and Multiscale Statistical Features". In: *IEEE Transactions on Biomedical Engineering* 63.9 (Sept. 2016), pp. 1820–1829. ISSN: 1558-2531.
DOI: `10.1109/TBME.2015.2503421`.

[35] Yefeng Zheng et al. "3D Deep Learning for Efficient and Robust Landmark Detection in Volumetric Data". en. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015.*
Ed. by Nassir Navab et al. Vol. 9349.
Series Title: Lecture Notes in Computer Science.
Cham: Springer International Publishing, 2015, pp. 565–572.

ISBN: 978-3-319-24552-2 978-3-319-24553-9.
DOI: 10.1007/978-3-319-24553-9_69.