Classification of cardiac dysfunctions using a 3-axis accelerometer and deep learning architectures

Jonas S. Waaler

Thesis submitted for the degree of
Master in Nanoelectronics and Robotics: Robotics and Intelligent Systems

60 credits

Institute for informatics

Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2019

Classification of cardiac dysfunctions using a 3-axis accelerometer and deep learning architectures

Jonas S. Waaler

Classification of cardiac dysfunctions using a 3-axis accelerometer and deep learning architectures

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

In collaboration with

The Intervention Centre
Oslo University Hospital, Rikshospitalet
Faculty of Medicine
University of Oslo,
Oslo, Norway

Acknowledgements

This thesis was carried out autumn 2017 to 2019 in a collaboration between the research group Robotics and Intelligent Systems (ROBIN) and The Invervention Centre at the Oslo University Hospital.

Firstly, I would like to thank my supervisor, Adjunct Professor Ole Jakob Elle, for exceptional supervision and advisement. I would also like to thank co-supervisor, Associate Professor Kyrre Glette, for weekly discussions and invaluable input. Especially, I would like to thank them for being patient and supportive during and after my treatment at the hospital.

In addition, I would like to thank PhD fellow Magnus Krogh and Anesthesiologist Dr. Per Steinar Halvorsen, for sharing their outstanding knowledge and pointing me in interesting directions. Lastly, I would like to thank my partner, Hanna Furuseth, for making my English readable, and my fellow students at ROBIN for fruitful discussions and knowledge sharing.

Abstract

This thesis explores the possibility of using a 3-axis accelerometer and deep learning architectures to classify cardiac dysfunctions. The thesis is built upon previous research performed at The Intervention Centre at Oslo University Hospital, where they have developed a novel technique for detecting cardiac heart dysfunctions using a 3-axis accelerometer. The purpose of this thesis is to examine if deep learning architectures can be used to automatically detect cardiac dysfunctions using the 3-axis accelerometer.

Three experiments have been conducted. The first experiment explores what deep learning architectures that works best for extracting features from the accelerometer data, as well as building a bridge between deep learning and previous research performed at The Intervention Centre. The second experiment is performed to see if deep learning can be applied to classify heart functions using 3-axis accelerometer data. The last experiments is conducted to investigate wheter further enhancement of the classifier is possible by converting the accelerometer data into the image domain.

The conclusion is that deep learning is an excellent tool for classifying cardiac heart dysfunctions, and can also be used as a tool for extracting clinical knowledge.

Contents

List of Figures

List of Tables

Abbreviations

DL Deep Learning

DNN Deep Neural Network

ECG Electrocardiography

ACC Accelerometer

MSE Mean Square Error

PCC Pearson Correlation Coefficient

APCC Absolute Pearson Correlation Coefficient

FFT Fast Fourier Transform

MFFT Mean Square Error Fast Fourier Transform

SD Standard Deviation

IVC The Interventional Centre

OUH Oslo University Hospital

Chapter 1

Introduction

24 hours a day it ensures that you are offered nutrition, nutrition you can't do without, nutrition you need to survive. The human heart is the most important organ in our body. The importance of a healthy heart is self-explanatory, as if it fails, we fail. Luckeli, the human body is constructed in such a way that it protects vital organs, such as the heart. The heart is hidden inside our body. However, this makes the heart hard to reach, hard to study, thus hard to understand. Developing methods for looking inside a human body has been a priority for both clinics and engineers throughout time, since the early days of Röntgen's x-ray discovery, to more recent years where the spin angular momentum of protons is used to study the inside of the human body (MRI). The latter techniques are brilliant methods for understanding the body, providing instant snapshots of the visceral organs. Continuous monitoring, on the other hand, provides additional challenges, as the instrument monitoring the area of interest must be continuously present. Methods such as electrocardiography (ECG) have been developed for measuring the heart's voltage over time, thus providing a non invasive monitoring technique measuring the heart's current state. Given the fact that ECG measures a very specific function of the heart, other crucial features may not be present in the signal. This motivates the development of new and improved methods. Ultrasound, in its various forms, has become a leading heart monitoring technique as it can perform continuous monitoring of the heart, and provides 2d, or even 3d images using Doppler based methods. However, as these techniques do not involve instruments attached directly to the heart, there is a risk of some information being lost or distorted.

1.1 Motivation

In 2016, approximately 60 000 Norwegians either consulted their doctor or was hospitalized due to ischemic heart decease [1]. Blood clot, or narrow

blood vessels (occlusion) is usually the cause of ischemia, which means that there is a lack of oxygen in one or more organs. The Interventional Centre (IVS) at Oslo University Hospital (OUS) has developed a novel technique for early detection of cardiac ischemia using a 3-axis accelerometer [2]. The sensor is attached to the apex (bottom) of the heart, measuring the heart's acceleration in three directions [3]. The research revealed that changes in the heart's acceleration is prominent during occlusion [4]. The intended goal of this research, is to develop a continuous monitoring device that can detect complications during and after a cardiac surgery [5]. Therefor, an autonomous detection algorithm is desired, as stated by Halvorsen et al.:

"Echocardiography is an excellent tool to assess intraoperative ischemia, but the method is cumbersome and not routinely applied for continuous monitoring after surgery. For this period, the accelerometer may offer an important advantage relative to commonly used monitoring modalities. The development of an algorithm for automatic analysis of the accelerometer signals and further miniaturization of the device will be necessary for clinical implementation." [5]

1.2 Goal

The previous research on this topic performed at IVS, has mainly focused on analytical and manual methods for extracting information about the accelerometer signal. Deep Learning (DL) is a stochastic based branch within machine learning, currently emerging as the leading research field in terms of results and popularity, especially considering image classification and segmentation [6, 7, 8, 9, 10, 11]. This thesis will introduce DL architectures to time-dependent data, such as accelerometer data and electrocardiographic data, and among other things, explore whether DL can be used as a tool for extracting relevant features in the accelerometer signal. Hopefully, deep learning techniques might provide some new insight into the area, and support previous research done in the same field. So, in this context, the thesis will consist of three experiments. The first experiment is an attempt to build a bridge between the previous research performed at IVS, and DL architectures, where the hypothesis developed at IVS - that accelerometer data adds diagnostic value to a monitoring situation - will be investigated. The second experiment will be based on the observations made from the first experiment, exploring the possibilities of classifying multiple cardiac functions and dysfunctions using DL networks trained on accelerometer data. The third experiment will be a further study, where the accelerometer data will be transformed into a higher dimensional space by using continuous wavelet transform, to investigate if this further improves the models. In parallel to the mentioned experiments, domain knowledge obtained during

the experiments will be extracted and discussed, and hopefully add additional clinical and technological insight. A visualization of the relations between the experiments are shown in figure 1.1.

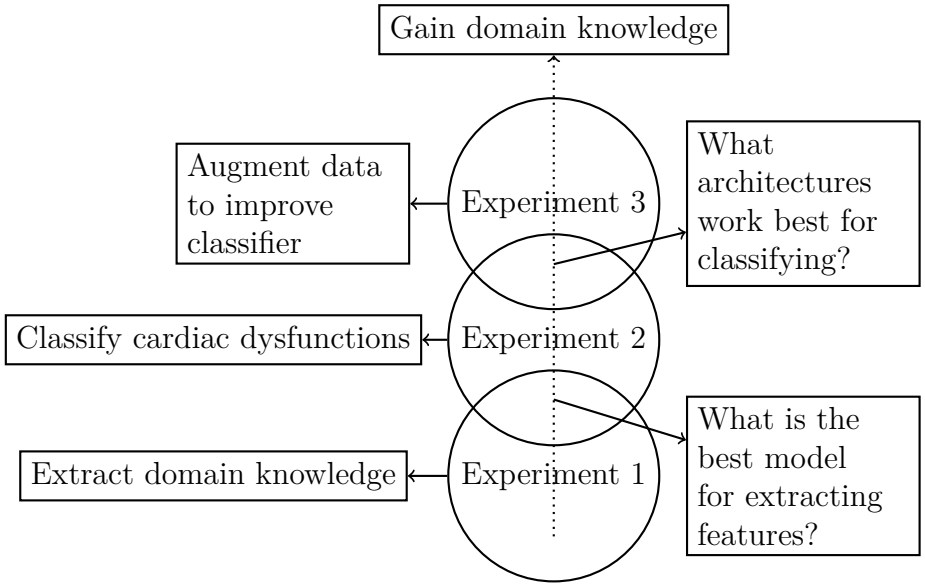


Figure 1.1: A visualization of how the experiments are related, and the questions they will address.

1.3 Outline

Chapter 1. Introduction

An introduction, the motivation and the goal of the thesis are presented.

Chapter 2. Background

Theoretical background for the thesis, and the novel technique developed at the Intervention Centre at the Oslo University Hospital, are introduced in this chapter.

Chapter 3. Data preprocessing

Preparations done on the data previous to all the experiments are presented.

Chapter 4. Experiment 1: Domain Analysis

The first experiment, where a comparison between the accelerometer data and the electrocardiographic data, is conducted. The experiment is investigating the need for accelerometer data over electrocardiographic data, in a deep learning context. Also, the best deep learning architectures will be used for experiment two and three.

Chapter 5. Experiment 2: Classifying Cardiac Heart Dysfunctions

The experiment explores if deep learning can bu used to classify heart dysfunctions, using accelerometer data as input.

Chapter 6. Experiment 3: Image Classification

The last experiment, where wavelet transform is applied to the accelerometer data before training the networks. The goal is to increase the dimension of the data, thus providing the model with more features it can discriminate between.

Chapter 7. Discussion

A discussion of the observations and the results is carried out. In addition, a comparison with similar studies done on the same data is presented here.

Chapter 8. Conclusion

Conclusions drawn from the experiments and the discussion wraps up the thesis in this final chapter. Possible future work is also discussed here.

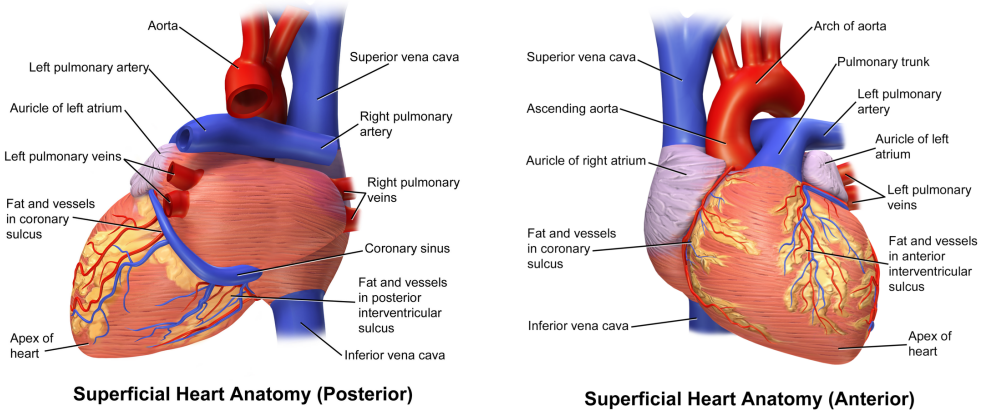
Chapter 2

Background

How does the human heart move? What is an accelerometer (ACC)? What is electrocardiography (ECG), and what features are present in the ACC and ECG signals? What is deep learning, and why is it relevant? To answer such questions, this chapter will consist of an introduction of relevant topics which form the background for the experiments described in the upcoming chapters. The first part of this chapter will consist of an introduction of basic anatomy and physiology of the human heart, followed by a review of the relevant clinical diagnostic methods and heart conditions. Next, the novel technique developed at IVS at OUH will be presented. The last part of this chapter will be an introduction to machine learning which eventually leads to a review of deep learning architectures and research relevant for applying such techniques to ECG and ACC signals.

2.1 The Human Heart

The human heart is the engine behind the circulatory system, responsible for pumping blood throughout the body's vital organs. It is approximately as big as a fist, and located in the middle part of the chest, just behind the sternum (breastbone) and above the diaphragm (muscle between the stomach and the chest). The heart consists of four chambers, the upper right and left *atrium*, and the lower right and left *ventricle*, and at the bottom, the heart ends up in a blunt tip (*apex*), pointing to the left side of the body. The heart is constructed in such a way that the blood can flow in only one direction between the chambers. This is due to the fact that the valves are constructed in "v-shapes", allowing flow in only one direction. When the valves close, they produce an audible sound, making the famous "lub-dub" noise. Figure 2.1 shows the superficial anatomy of the heart, including exterior veins and arteries. [12]



(a) (b)

Figure 2.1: The coronary circulation of the human heart [13, 14].

2.1.1 The circulatory system

The circulatory system is driven by the heart, and transports blood containing oxygen and nutrition to the body. The nutrition mostly consists of fat, carbohydrates and proteins. It also transports white blood cells and several antibodies that will protect the body against infections. Waste material is transported to the liver and kidneys and carbon dioxide to the lungs. The circulatory system stabilizes the inner environment as regards of PH, ions, flow rate, and osmotic pressure. Figure 2.2 is a simplified illustration showing the blood flow direction of the circulatory system. When the lungs are being ventilated by breathing, gas exchanges between *alveolars* and *lung capillaries*, replacing CO_2 with O_2. The newly filtrated blood is

transported to the *left atrium*, pumped to the *left ventricle* through the *mitral valve*, and later pumped out through the *aortic valve* to the *aorta* which supplies the organs with blood. The organs' waste is transported back through the *vena cava*, passing the *tricuspid valve* from the right atrium to the right ventricle, and pumped back to the lungs through the *pulmonary valve*. If the heart fails, vital organs naturally start to fail. [15]

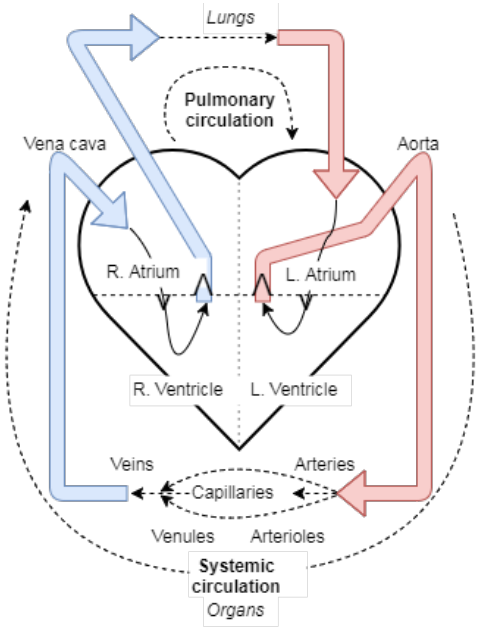


Figure 2.2: A simplified figure of the circular system. Red indicates arteries and blue indicates veins. The blood flow direction is illustrated by the arrows. Right is denoted as R. and left as L.

2.1.2 Cardiac motion

The heart's cycle consists of two phases, *systole* and *diastole*. In a normal heart, the muscles surrounding the chambers contract and relax in a rhythmic manner. Normally, this frequency is about 60-80 beats per minute (B.P.M.). However, during physical stress, the frequency can increase to 200 B.P.M. [15]. The period when the heart is contracting is called systole, followed by a resting period called diastole. During diastole, the left atrium and ventricle are filled with blood originating from the lungs. While the mitral valve is open, allowing blood to flow between the left chambers, the aortic valve is closed, preventing blood from flowing out through the aorta. At the end of the diastole, the atrium contracts, and blood flows to the ventricle. After the contraction, the pressure in the atrium drops, resulting in a short period of time where blood flows back to the atrium closing the mitral valve. The right atrium and ventricle function in a similar manner as the left chambers, and contract and relax approximately at the same time [15]. After relaxing,

the heart is contracting. This period is called systole. During systole, the ventricles contract, increasing the pressure until the aortic valve and the pulmonary valve opens and supplies the body and lungs with blood. Wigger's Diagram, shown in figure 2.3, illustrates the periodic cycle of the heart, and demonstrates the systole and diastole period. The figure shows the sound produced by the heart (phonocardiogram), the voltage propagating from the heart (electrocardiogram, introduced in 2.1.4), the pressure of the chambers, as well as the ventricular volume.

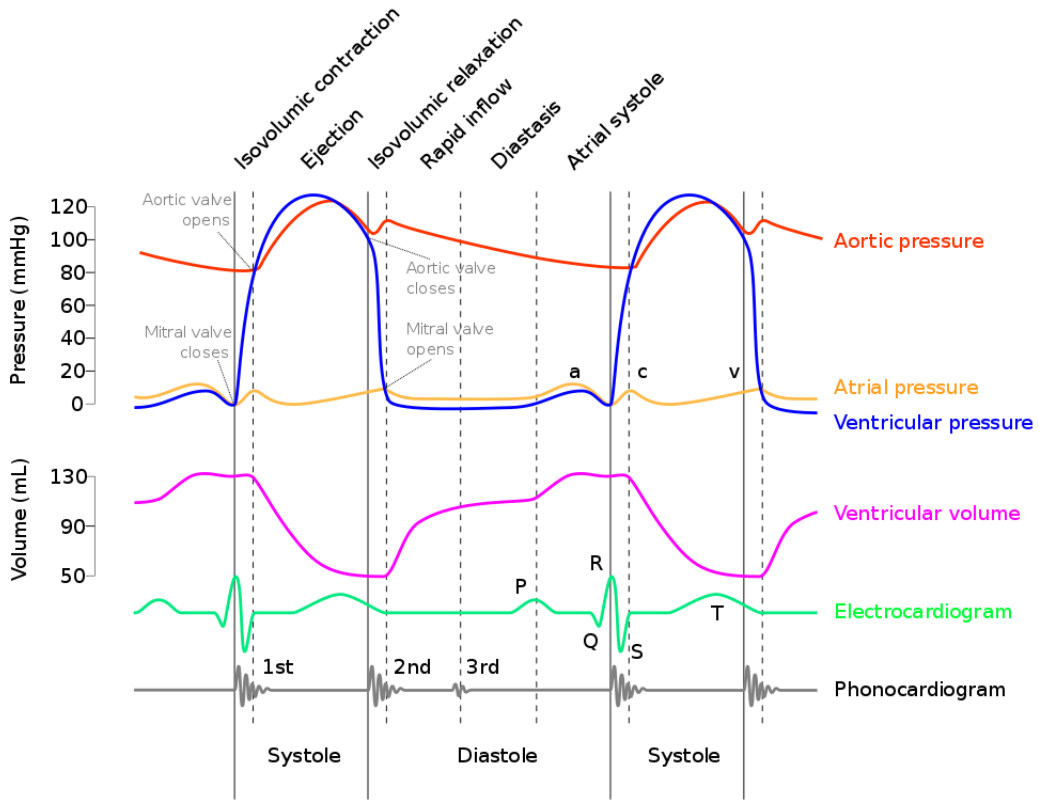


Figure 2.3: Wigger's Diagram [16] showing the heart's periodic states. The green line illustrates the equivalent electrocardiogram signal during the cycle.

Figure 2.4 shows a three axes decomposition of the heart's movements. The motion axes are the radial, longitudinal and the circumferential motions. This way of describing the motion is called elliptic motion. In this thesis, the circumferential motion will sometimes be referred to as the Y-axis, the longitudinal motion the X-axis, and the radial motion the Z-axis. This is due to the placement of the accelerometer sensor introduced in section 2.2.

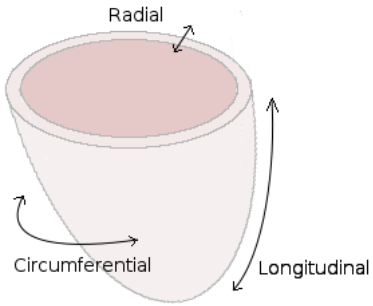


Figure 2.4: Figure demonstrating three axes relevant to the heart's motion. The radial motion, sensitive to the wall's change in thickness, the longitudinal motion, sensitive to the end of the systole, and the circumferential motion, sensitive to the beginning of ventricular systole.

The isovolumetric relaxation (IVR) phase, seen in figure 2.3, is an important state of the heart's movement when it experiences ischemia.

2.1.3 Cardiac conditions

The following drugs and interventions, forcing specific cardiac conditions, are relevant for this thesis.

Fluid loading

Fluid loading enhances preload. Preload is the cardiac myocytes initial stretching prior to contraction. Preload can also be increased by reducing the heart rate, which increases the time it takes to fill the ventricles [17].

β-blockers

The drug Esmolol, is a type of β-blockers commonly used during operation to prevent tachycardia, meaning that the heart rate is increasing above normal. Low heart rate could be a sign of low contractility. [18]

Adrenaline

The hormone epinephrine, also called adrenaline, forces high contractility and can be used as a stimulant if the myocardial contraction stops (cardiac arrest).

Nitroprusside

Nitroprusside (or Nipride) is used to lower blood pressure, and sometimes during surgery to decrease bleeding. Nitroprusside decreases afterload.

Myocardial ischemia

Ischemia refers to the lack of blood supply to tissues. Myocardial ischemia, or cardiac ischemia, occurs when the blood flow to the heart is reduced. Cardiac ischemia may be caused by a blockage in, or a shortage of, the coronary arteries. When cardiac ischemia occurs, the myocardia might be damaged, which can lead to irregular heart rhythms, heart attack or heart failure. [19]

2.1.4 Monitoring the heart

Electrocardiography

Electrocardiography (ECG) is the process of monitoring the heart's change in biopotentials originating from the *Sinoatrial node*[1] [20, p. 41]. It is a *non-invasive* procedure, meaning that no surgical intervention is needed. The procedure is done by attaching electrodes to the human skin at several standard positions. The cardiac biopotentials are measured by calculating the difference between positive electrodes and reference electrodes [20, p. 42]. The change in cardiac biopotential is measured by the electrodes, and is displayed over time as a graph, shown in figure 2.3, and specific events, or features, in the heart's cycle become prominent, known as the P, QRS and T events, shown in figure 2.1.

State	Biopotential	Physiology
P	Atrial depolarization	Contraction
QRS	Ventricular depolarization	
T	Ventricle repolarization	Relaxation

Table 2.1: The P, QRS and T events explaining the ECG signal in figure 2.3 [21].

[1]The Sinoatrial node is a group of cells in the right atrium spontaneously creating electrical impulses acting as the heart's pacemaker.

2.2 Detecting cardiac dysfunctions

Monitoring the heart for detecting cardiac dysfunctions has proven to be a difficult task. One of the most commonly used continuous monitoring techniques is the ECG. By investigating the ST-T segment of the ECG cycle, a trained clinician might detect cardiac abnormalities such as ischemia. In the context of detecting ischemia, algorithms such as Hidden Markov Models [22], neural networks [23, 24], genetic algorithms [25], and more [26], have been applied by analyzing the ST-T segment in the ECG signal. However, the ST-T segment in the ECG cycle has proven to be inconsistent [27], and it requires a trained clinician to observe the signal for manual detection of the abnormalities. This motivates the need for better continuous monitoring techniques. Strain and tissue velocity monitoring by the use of echocardiography and doppler has proven to be an excellent tool for detecting dysfunctions, as it can estimate the heart's contractility [28]. This method still requires a trained clinician to hold the probe and analyze the image during monitoring.

In 2005 Elle et al. developed a novel technique at The Intervention Centre at the Oslo University Hospital for detecting cardiac ischemia using a 3-axis accelerometer attached to the heart [2]. By attaching a 3-axis accelerometer to the heart of a pig, changes in the acceleration were prominent during occlusion of the left anterior descending artery (LAD), as seen in figure 2.6. This led to the conclusion that early recognition of ischemia should be possible using a 3-axis accelerometer, and facilitated further studies.

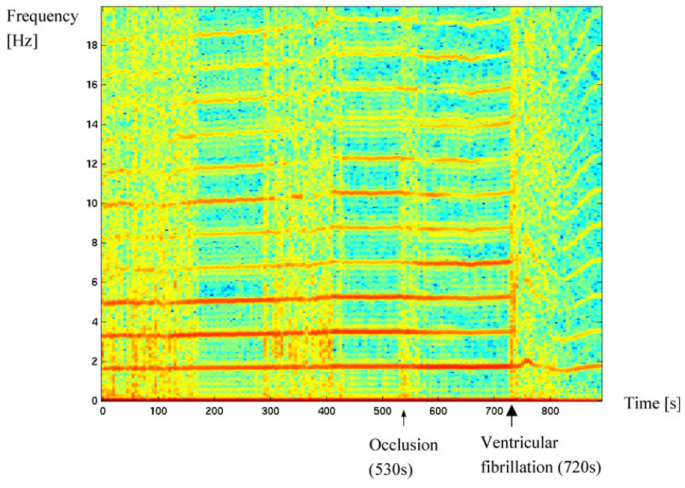


Figure 2.5: Spectrogram showing the change in frequency power before and after occlusion of the LAD. Image from [2].

Halvorsen et al. did further research on pigs, and found that the novel technique could contribute in differentiating between ischemia and other cardiac

functions, and that the most prominent change in the accelerometer data during ischemia was the circumferential (Y-axis) movement [3]. They discussed that *".. compared with echocardiographic methods, the main advantage for the accelerometer is that it may enable operator independent continuous real-time monitoring of myocardial ischemia..."*. In a later study, Halvorsen et al. investigated the possebilities of continuous monitoring of surgical patients using the 3-axis accelerometer, and found that during ischemia, *"...a typical ischemic velocity pattern was observed, with a decreasing early peak systolic velocity and an increase in OVR velocity ..."* [5]. These studies laid the foundation for years of research on the field, and was later confirmed [29, 4, 30, 31, 32, 33, 34] and also simulated [35]. The most recent study shows that the gravity factor can be estimated by adding a gyro to the readings, and *"... subtracted from the measured acceleration signal, leaving a very good estimate of the pure acceleration caused by cardiac motion..."* [34].

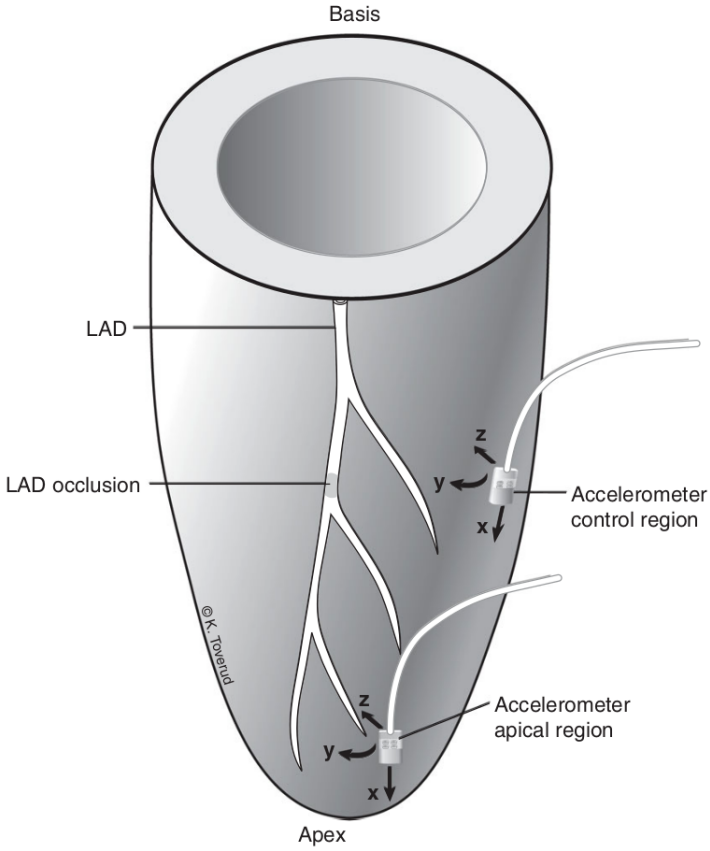


Figure 2.6: The placement of the sensors used in the experiments performed by Halvorsen et al., described in [3]. The accelerometer sensor attached closest to the apex is the sensor of interest, where the Y-axis denotes the circumferential, the X-axis the longitudinal, and the Z-axis the radial movement.

2.3 Regression

In regression analysis, the goal is to find a relationship between one or more independent variables, and a dependent variable. The independent variables are the explanatory variables (X), where as the dependent variable is the response to the explanatory variables (Y). In linear regression, the goal is to estimate the underlying linear function that describes the most likely ouput ($\hat{Y}$) given an input (X), as shown in equation 2.1, where b is an offset, and w^T the *influence* coefficients denoting the partial derivatives of the dependent variable with respect to the independent variable. The regressor line is calculated by minimizing the sum of squared distances between the points defined by the dependent and independent variables', and the linear regressor line. This form for estimation is called a least square estimate, and it is extensively used to measure the error of regressors, where the mean square error, shown in equation 2.2, is the most commonly used.

$$\hat{Y} = w^T X + b \tag{2.1}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.2}$$

Least square estimates have some limitations. For example, they are optimal if the errors are *serially uncorrelated*, meaning that the errors cannot be described as a repeatable function, and *homoscedastic*, meaning that all the independent variables have the same finit variance. This makes least square error estimates sensitive to outliers in the data set, meaning that single observations that do not follow the same patterns as the other observations might affect the regressor negatively. However, by using robust regressors, one might overcome the limitations of least square estimations. For example, the Huber loss (L_δ) is a robost loss function that is less sensitive to outliers in the data as it contains a penalty for large values. The Huber loss is quadratic for small error values, and linear for large values, as shown in equation 2.3, where δ is the slope of the linear part. Using such robust loss functions, one might be able to make use of noisy datasets containing a large number of outliers.

$$L_\delta(Y, \hat{Y}) = \begin{cases} \frac{1}{2}(Y - \hat{Y})^2 & for\ |Y - \hat{Y}| \leq \delta \\ \delta|Y - \hat{Y}| - \frac{1}{2}\delta^2 & otherwise \end{cases} \tag{2.3}$$

Regressors have become more and more powerful, as one can argue that deep learning architectures are essentially just more complex regressors. A traditional neural network, introduced in the following section, will perform linear regression if it contains only linear activation functions. When replacing the last activation with a sigmoid function, it performs logistic

regression (binary classifier). When replacing it with a softmax function, it becomes a multiclass logistic regressor (multiclass classifier).

2.4 Deep Learning

The term Deep Learning (DL) is not well defined, as it is vaguely described in most cases. However, DL can be seen as a technique applied where several high-level features are needed to describe classes. This often results in large networks consisting of multiple hidden layers with several activation functions - deep neural networks (DNN). In a phrase like "a deep network", "deep" can be interpreted as meaning large, that is, a network consisting of a large amount of hidden layers. As a consequence of the size of DNN, usually no feature extraction is done explicitly, as it is often desired to let the network learn all the features by itself. The recent research applying deep learning, has heavily focused on image-related tasks, such as image classification, segmentation, and similar [6, 7, 8, 9, 10, 11]. For time-series signals, such as accelerometer data and ECG data, there is little research done on applying similar machine learning architectures used to solve image-related tasks.

2.4.1 Traditional neural networks

Neural networks are inspired by how the human brain allegedly works, by how it makes decisions, and how it learns. In the human brain there are millions of brain cells connected to each other by nerves, or synapses. Although the brain's architecture is complex and it is difficult to get a detailed picture of the building blocks, researchers have a general idea of how signals propagate in the brain. The amount of joint signal strength arriving at each neuron determines whether or not the neurons should fire signals to the next intermediate nerve, and so on. When new knowledge is obtained, the neurons can either be weakened or strengthened, or new connections can be built. In spite of the complexity, by numerical simulation of simplified neurons, synapses and signal flow, artificial humanlike features can be simulated. In this synthesized brain, each neuron (h) is a product of the connected nerves, or weights (w), and the input (x), added toghether with a bias (b). The final output of the neuron is calculated using an activation function (F), introduced in 2.4.4.

$$h = F(\sum_{i}^{n} x_i w_i + b) \tag{2.4}$$

2.4.2 Convolutional Neural Network

Convolutional Neural Network (CNN) is inspired by biological processes where reception fields react only to a limited field of view. A convolutional layer consists of multiple small filter kernels that convolve (slide) over all spatial locations of the input and produce activation maps. This makes it so that even though the filter kernels are small, they extend to the full depth of the input volume. During forward pass, the dot product is computed between the filter and the input, producing the activation map, as demonstrated in figure 2.7. The network "learns" filters that will activate when key features are "discovered" in some spatial position of the input [36].

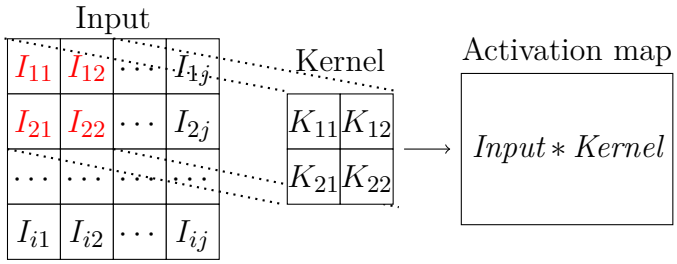


Figure 2.7: Figure showing a convolutional layer producing an activation map (without padding). The kernel in this examples is a 2×2 kernel, that will convolve over all input elements, producing the activation map. For example, the first element in the activation map (produced by the red elements in the input, and the kernel) will be $I_{11}K_{11} + I_{12}K_{12} + \cdots + I_{22}K_{22}$.

2.4.3 Training a network

The goal when training a neural network is to minimize an error function, or cost function, measuring the difference between prediction ($\hat{Y}$) and expected prediction (Y). There exists multiple error functions, such as the MSE, or the Huber loss, introduced in section 2.3. However, when dealing with prediction of multiple classes, the cross-entropy loss with the softmax function is extensively used. Cross-entropy is a metric of distances between two distributions, and is defined as shown in equation 2.5. The softmax function, defined in 2.6, takes an input vector a, and calculates a N dimensional vector that adds up to 1.

$$H(y, \hat{y}) = -\sum_{k}^{K} Y_k log(\hat{Y_k}) \tag{2.5}$$

$$S_k = \frac{e^{a_k}}{\sum_{n=1}^{N} e^{a_j}} \qquad \forall k \in 1...N \tag{2.6}$$

During training, after each prediction, the weights in the network are adjusted to minimize the loss function. To achieve this, the partial derivatives are calculated, starting at the end of the network, going "backwards" all the way to the input, and the weights are adjusted accordingly. This process is called backpropagation. Decreasing the loss function by taking negative steps guided by the gradient is called gradient descent. *Adam* is an optimized gradient descent method, that estimates first and second order moments of the gradient, resulting in a faster convergence of the loss function [37]. *Adam* is extensively used when training DNN's.

2.4.4 Deep learning building blocks

One of the biggest problems when training deep neural networks is the danger of over fitting the network to the training data. Since the network has so many hidden layers, it has enough parameters to memorize all the training data, eliminating the need for extracting features. To avoid the problem of over fitting, various techniques have been developed. These techniques are called regularization techniques. When constructing a network, the regularization techniques can be seen as building blocks, or "layers", within the network. Typically, a network is built by several different types of layers, where convolutional layers, and traditional neural network layers (also called dense layers) usually are followed by several "layers" serving different purposes. Such a composition of multiple layers of various sorts, can be referred to as a model.

Activations

Activation maps and neurons are generally followed by some activation. Historically, the sigmoid function has been a popular choice, due to its similarities to a step function, with its output being between 0 and 1, as shown in figure 2.8. The most important difference between a sigmoid function and a step function is that the sigmoid function provides a smooth, manageable derivative for all inputs. However, when the input is saturated ($-5 \ll x \gg 5$), the derivative of the sigmoid function is small, and does not reflect the change needed for adjusting a network. This can be solved by either using batch normalization, introduced in the following section, or by using other activation functions, such as variations of the *ReLU* function.

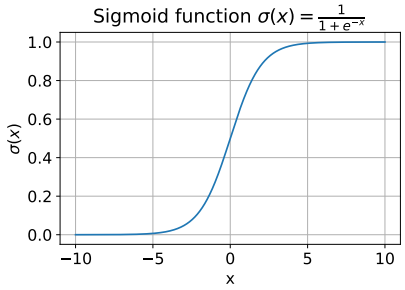


Figure 2.8

The currently most used activation function is the Rectified Linear Unit function ($ReLU$) defined in equation 2.7 [38]. $ReLU$ is linear for input bigger than or equal to zero, and zero if the input is less than zero.

$$ReLU(x) = \begin{cases} x, & if\ x \geq 0 \\ 0, & else \end{cases} \tag{2.7}$$

Sometimes variations of the $ReLU$ are used, such as Leaky $ReLU$. The only difference between Leaky $ReLU$ and $ReLU$ is that for Leaky $ReLU$, $y = \frac{x}{a}$ if $x < 0$, where a is a fixed parameter. Leaky $ReLU$ can be used when the network experiencse a lot of dead activations, or zero activations, for any input. If the network experiences a lot of "exploding gradients", a clipped $ReLU$ can be used. The clipped $ReLU$ works by having an upper limit of activation, where values above a certain parameter will output a constant. The linear activation function, or the identity function, defined in equation 2.8, is occasionally used as the last activation function. The identity function can for example be used when the target output of the network is a n sized vector.

$$f(x) = x \tag{2.8}$$

Batch normalization

When the distribution of the input training data is changing a lot (high variance), and the outputs distribution is stationary, the network is said to experience covariance shift. By giving the network a mini batch of input - more samples for each backpropagation, the gradients become more specified towards all the training samples and eliminates some of the covariance. However, despite using mini-batches, the training process can still be slow, as the network can experience some of the weights becoming very high, or low. In a sigmoidic setting, this results in small gradients (as the gradient of sigmoid for high values is low), thus resulting in slow convergence. By

pushing the input batches to a joint, similar distribution, one might overcome these issues. In fact, by normalizing over all the batches, learning the normalization parameters, convergence becomes faster as one training sample is set in perspective of the entire training data set. This process is called batch normalization [39], and today it is considered a matter of course when training deep networks as it both speeds up the training process, and works as a regularization technique.

Dropout

By dropping nodes during training, that is, ignoring connections between neurons, with a probability p, only a "thin" portion of the network is trained, averaging the model closer to a generalization. This regularization technique is called dropout [40].

Max pooling

Another regularization technique is max pooling. Max pooling downsamples an input by applying a max filter to regions of the input, where the max value within the region is selected. Usually, these regions are not overlapping, meaning that the step of the max filtering process (stride) is equal to the size of the filter. For example, a 2×2 max pooling filter applied to a 4×4 with stride 2, produces an 2×2 output, as shown in figure 2.9.

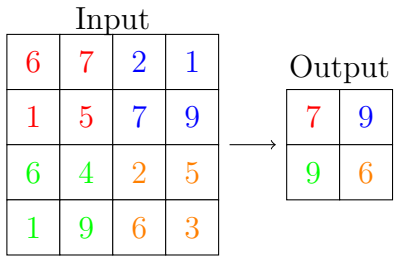


Figure 2.9: An example of max pooling applied to input of size 4×4 with stride 2, producing an output of size 2×2

2.4.5 Recurrent neural networks

Traditional neural networks are particularly good at learning hierarchical dependencies in data, both when dealing with supervised and unsupervised learning. However, after an input has been processed, the current states that led to the output are forgotten. This makes these networks incapable of modeling sequential and time dependencies. In other words, traditional

neural networks do not have sequential memories. Recurrent neural networks are constructed to have such memories, making them able to produce continuous outputs based on previously states from an given context size. [41]

2.5 Time-Series Signals

One of the main challenges when analyzing time-series signals is the amount of noise, and the fact that the signals often include redundant dimensions not needed for the task at hand. Usually, some of these problems are partially solved by doing some preprocessing of the data, using techniques as filtering or dimension reduction. However, this comes with a cost, as important features may be lost in the process, making a classification task more difficult. In addition, frequency, variance and mean can change over time, as the time-series signal can be non-stationary, making the classification task even harder. These problems are amplified by the fact that it is not guaranteed that the sensors used to measure the process capture the relevant or the most essential dimensions needed for describing the target. [42]

In complex image classification tasks, such as image segmentation or classification, one of the main problems is making the model invariant to noise, such as translation, rotation, scale and more. Deep learning has proven to be an excellent tool for solving such tasks. By using deep learning architectures, like CNNs, classifiers are able to efficiently extract high-level features invariant to noise and classify with high accuracy. When using such deep learning architectures, it has proven to be sufficient not to do any preprocessing of the data, thus using raw image data as input. [43]

However, when dealing with time-series signals, applying traditional neural network architectures can be problematic as this would require some sort of selection, or buffering, of the signal, as the model's perception of time is dependent on seeing the data in the context of time. For example, if a classifier were to classify the next word in a sentence, the classification could be dependent on the previous word or the previous sentences, the past couple of pages, or even the last chapters. So ideally, the classifier should not lose the perception of time, keep the dimension, and be able to extract infinite long-time dependencies to determine the current output.

2.6 Transfer learning

When training with a specific type of input (*feature space*) on deep neural networks, e.g. images, different networks tend to learn similar features, especially in the first hidden layers, regardless of cost function, natural image dataset and target. These first layer features are often referred to as *general*. When doing transfer learning, the goal is to utilize such first layer features from a trained network, called the *base* network, when training a new network (*target* network). This can be done by copying the n first layers from the base network to the target network and initialize the remaining layers as usual. The target network is then trained towards the target task in both, or one of the two following ways: By backpropagating the errors to the copied features (*fine-tuning*), or by *freezing* (not backpropagating the error through) the n copied layer features. The latter option is often preferred if the dataset of the target network is small, due to its capability to avoid overfitting. This process tends to work if the features are general, meaning suitable for both base and target tasks [44].

2.6.1 Autoencoder

The goal of an autoencoder is to compress data, yet keeping the information describing the data. The autoencoder will implicitly extract the most descriptive components of the data when compressing it, and it is therefore a commonly used component in creative architectures within DL. The idea behind an autoencoder is relatively simple - push raw data (X) through a shrinking pipeline (q) and make the autoencoder network learn what features to remove and what to keep. Push the shrunken data (z) through an increasing sized pipeline (p), and make the autoencoder network learn what features it needs to add to reproduce the raw input, as shown in figure 2.10.

Autoencoders can, for example, be used as a denoiseing tool. This is done by training an autoencoder on noisy input, predicting an output not containing noise [45].

2.7 Signal processing theory

In order to perform and understand the following experiments, some basic signal processing theory should be presented. Re-sampling of data could be a useful tool to either compress, or enlarge data. Re-sampling is equivalent of changing the sampling frequency (Fs) of the signal, that is, lowering or increasing the number of samples that represent the time dependent signal.

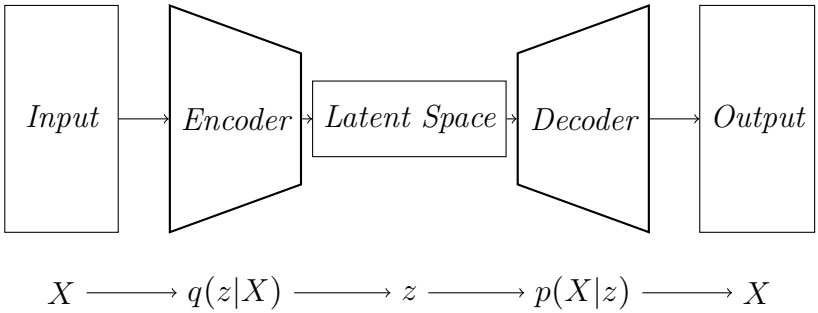


Figure 2.10: Figure demonstrating the autoencoder architecture. The autoencoder is provided an input (X) of a high dimensional space. The encoder (q) compresses the input to a smaller dimensional space, called the latent space (z). The decoder (p) decodes the smaller dimensional space and reproduces the input.

There are several ways to achieve re-sampling. The simplest re-sampling method is linear interpolation. Linear interpolation assumes there is a linear relationship between the samples representing the signal. For example, if one was to double the number of samples, one would just linearly add samples in between existing samples. When going the other way around - re-sampling from a sample rate that is higher than the target sampling rate - information in the frequency domain is lost. In figure 2.11 the Nyquist frequency is demonstrated. To be able to sample one period of a sinus signal, at least two sampling points are needed. In figure 2.12 a demonstration of the loss of information in the high frequencies due to resampling is shown.

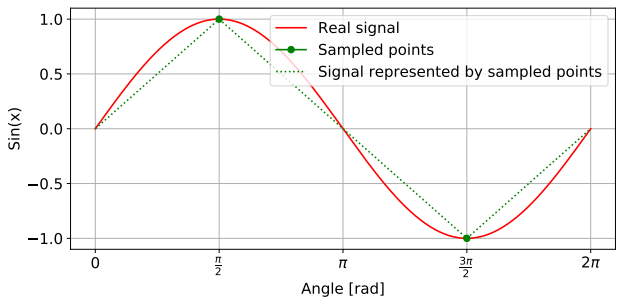


Figure 2.11: Plot demonstrating the Nyquist frequency. One period of a sinus signal (solid) is sampled by two points representing a sampled signal (dotted).

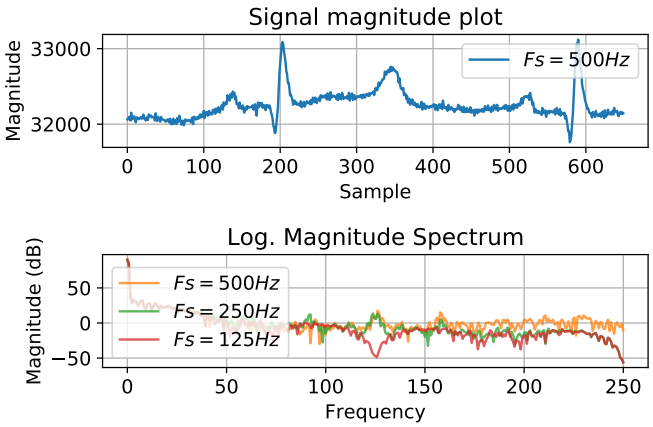
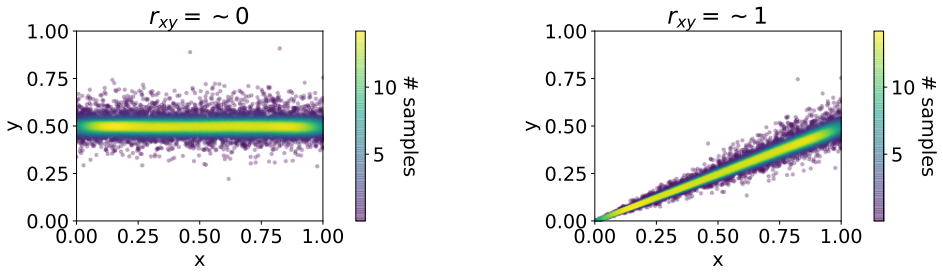


Figure 2.12: Plot demonstrating the loss of information when changing the sampling frequency.

2.7.1 Pearson Correlation Coefficient

The Pearson Correlation Coefficient, or the Pearson product-moment correlation coefficient, is a metric used for calculating the linear relation between two signals. The Pearson correlation coefficient (r) for two vectors (x and y) is calculated as shown in equation 2.9, where $\bar{x}$ and $\bar{y}$ denote the sample mean ($\frac{1}{n}\sum_i^n x_i$) for the respective vectors [46]. r_{xy}, as of equation 2.9, is the centered and standardized sum of the cross-product of two variables, and will not have bigger or smaller values than ± 1, where $r_{xy} = 1$ corresponds to absolute linear relationship between x and y, and $r_{xy} = -1$ corresponds to absolute negative linear relationship between x and y [47]. $r_{xy} = 0$ denotes undefined, or no linear relationship between the two of them. Examples of $r_{xy} = 0$ and $r_{xy} = 1$ are shown in figure 2.13.

$$\rho_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{(x_i - \bar{x})^2(y_i - \bar{y})^2}} \tag{2.9}$$



(a) (b)

Figure 2.13: (a) demonstrates extreme case of pearson correlation coeffectient r of x and y, where $r_{xy} = \sim 0$. (b) Demonstrates $r_{xy} = \sim 1$.

2.7.2 Fourier and continuous wavelet transform

Fourier transform represents a signal as a sum of sinuses, oscillating for an infinite amount of time. This is why a Fourier transform does not contain information about space and time. A wavelet, e.g. the Ricker function, is a signal that exists for a given amount of time. A wavelet function has zero mean and can be scaled in time. If a wavelet is scaled by a factor of two, its base frequency is cut in half. When convolving a wavelet over a signal, it can be seen as a bandpass process, where the scaling of the wavelet determines the band. A large scale factor gives a low frequency range, and a small scale factor gives a small frequency range. In continuous wavelet transform, wavelets of different scaling are convolved over the signal of interest. The coefficients are stored for each time step, and the final result is a two-dimensional plot of the convolutional coefficients.

2.8 Statistics

In statistics, there are four outcomes of a classifier. True positive (TP), false positive (FP), true negative (TN) and false negative (FN). A true positive means that a class was correctly classified as the correct class, and a true negative means that a class was correctly classified as another class. The false positive and the false negative refer to a classifier doing the positive predictions wrong. Precision and recall are usually used as metrics when evaluating classifiers. Precision measures how likely a prediction done by a model is to be correct. Precision is calculated by dividing the true positive by the true negative plus the false positive, as shown in equation 2.10. Recall measures how many of the target class it was able to recover. Recall is calculated by dividing the true positive by the true positive plus the false negative, as shown in equation 2.11. When classifying medical diseases, recall

is usually the most important metric, as it refers to how many cases of the disease it was able to "see". If a classifier wrongly predicts a disease (FP), it may only result in one too many doctor visits. But if it fails to detect the disease (FN), this could be even more dangerous.

$$Precision = \frac{TP}{TP + FP} \tag{2.10}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.11}$$

The F1 score is a combination of the precision and recall, where it measures the test's accuracy. It is the harmonic mean of precision and recall, as shown in equation 2.12. The F1 score can be used as an overall metric of a classifier's performance.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.12}$$

When dealing with multiple classes, a confusion matrix is often used. This matrix is an overview of the predictions done by the classifier, where the rows represent the actual classes, and the columns the predicted classes. Therefore, having the population in the diagonal refers to a classifier doing correct predictions.

Chapter 3

Data preprocessing

This chapter will consist of an explanation of what has been done to the data prior to any of the following experiments, and an introduction to the constructed data sets. Also, a noise removal algorithm is proposed.

3.1 Available data

The available data consist of recordings from two experiments, named AP and MKCM, conducted at the IVC at the OUH. Both experiments were performed on several pigs. The experiments produced recordings of an accelerometer (ACC) measuring the heart's acceleration, and recordings of an ECG, measuring the biopotential of the heart. For the AP study, these recordings were recorded during injection of several substances, such as esmolol/β-blockers, adrenaline, nitroprusside (nipride) and salt water. In addition, various interventions blocking some of the heart's blood vessels (occlusion) were recorded. During and after the interventions, clinicians observed and analyzed the data in order to select 10 seconds consisting of the maximum effect for each intervention. The MKCM study contains data where the pigs were injected with dobutamine (adrenaline), and recordings of occlusions. As a basis for the experiments explained in the following chapters, the observations and choices done by the clinicians are assumed to be correct. The data provided are both the classified data, and the unclassified data. The unclassified data will be utilized in experiment 1 as there is no need for the classes.

The AP study consists of recordings from 10 pigs, and the MKCM study consists of recordings from 11 pigs, resulting in data from 21 pigs total. Some of the recordings are recorded at 500 samples per second (Hz), some at 250 Hz, and the MKCM study is mostly recorded at 650 Hz. The raw data from these studies was provided to be used in the following experiments.

3.2 Data sets

To prevent the need for a clinician continuously monitoringing the accelerometer data, an autonomous algorithm analyzing the data in real time is desired. Given that the data is time-dependent, segmenting the raw data to appropriate input tensors[1] to such an algorithm is not trivial. One approach is to divide the data into tensors by features, e.g. the R-peak feature. When the algorithm is presented with such data, the algorithm will analyze one heart cycle at a time. Given that the whole heart beat cycle will be visible to the algorithm, all the features in the cycle will be present. This might make a classification task trivial. However, methods for keeping information about time, yet keeping the same numerical size for each tensor, must be applied. Another way is to divide the data into tensors by a set number of samples, e.g. 500 samples. By using this "sliding window" method, the

[1] An input tensor, or training tensor, is a tensor consisting of a set amount of time steps multiplied by the number of dimensions it contains. E.g. an accelerometer tensor consists of 3 dimensions multiplied with the number of time steps.

features of the data can occur at arbitrary samples, thus demanding the algorithm to be more complex. Also, given the fact that all the features of one heart beat cycle may not be visible to the algorithm, the algorithm could be dependent on the previous tensor. However, if the algorithm can solve such problems, dividing the data into windows will have a great advantage over using feature divided data, as a feature detection is not needed prior to the division, thus preventing additional errors. Inspired by this, all the data sets are time feature independent, following the "sliding window" idea, and set to be one second long. During that second, one or more heart cycles are highly likely to be present, as later demonstrated in figure 5.2. In figure 3.1, it is demonstrated how the R-peak feature will occur at different indexes for each training tensor. This way, the algorithm will learn to extract features regardless of spatial positions (time). Four data sets were constructed, as described in the sections below.

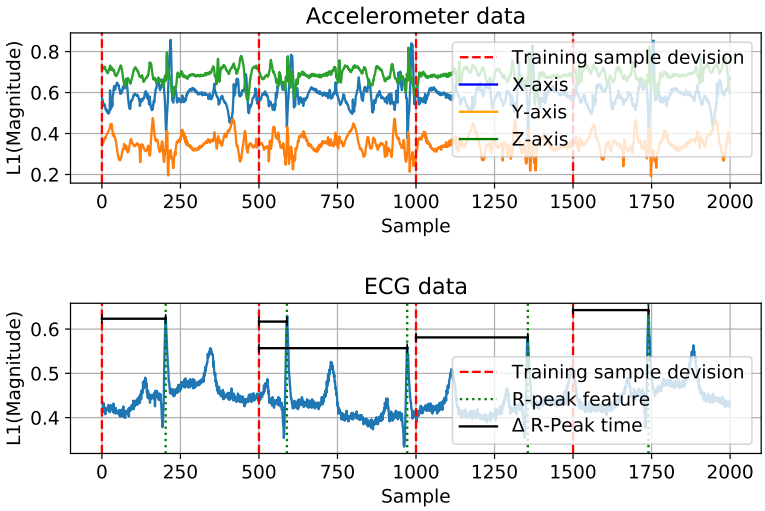


Figure 3.1: Plot demonstrating the R-peak feature occurring at arbitrary samples given a tensor division of 500 samples (1 second) for each training tensor. Features might appear several times for each training tensor, as seen in the second training vector in the ECG data plot.

3.2.1 Regressor data set

For the regressor data set, the data from the AP study was used[2]. The AP study contained a large amount of unclassified data. The data set was divided into 48030 training tensors. Each training tensor consists of $500 \times N_{dim}$ samples, where N_{dim} is the number of dimensions. 33% of the training tensors

[2]The MKCM data was available after the experiment 1 was done, and therefore not used.

was subtracted from the training set to be used as validation data, dividing the data into 32180 training tensors, and 15850 validation tensors[3] .

3.2.2 Classification data sets

Two classification data sets were constructed. One consisting of data from three classes, and one consisting of data from six classes. The six classes classification data set contains data only from the AP study, as this is the only data set containing six classes. For the six classes data set, 60 data tensors from each pig was available, adding up to a total of 600 tensors for the entire data set. Subtracting two test pigs results in 540 training tensors, 80 tensors per class. In a deep learning context, this is not much, and some augmentation of the data was required. This process is introduced and described in chapter 5.

The three classes data set contains data from both the AP and the MKMC study. The MKMC study contained longer recordings of each class. Using both data from both studies added up to a total of 1675 training tensors.

3.3 Normalization and noise removal

All the data is resampled to 500 Hz using linear interpolation. The mean of each axis in the ACC data are subtracted for each tensor, followed by a normalization to keep the data in the range from 0 to 1. The normalization is done after removal of unambiguous noise[4]. This noise is removed by using an outlier noise reduction algorithm. The algorithm works by first defining a noise coefficient (C_n). An upper and lower threshold are calculated by respectively adding and subtracting the standard deviation of the entire training set, multiplied by the noise coefficient. For each training tensor (x) in the training set (A), the training tensor is either removed or kept, as shown in equation 3.1.

[3]Validation tensors are used in the evaluation *during* the training process, not to be confused with test tensors that are used in the evaluation of the model *after* the training process.

[4]The *unambiguous noise* are elements with magnitude many times above or below the average peaks.

$$
\begin{aligned}
f(A_x) = \forall x \in A \begin{cases} keep\ x, & if\ \mu_A + \sigma_A \cdot C_n \leq \max(x_1, \ldots, x_n) \\ keep\ x, & if\ \mu_A - \sigma_A \cdot C_n \geq \min(x_1, \ldots, x_n) \\ delete\ x, & otherwise \end{cases} \\
\mu_A = \frac{1}{N_A} \sum_{i}^{N_A} A \\
N_A = length\ of\ A \\
\sigma_A = \sqrt{\frac{1}{N_A} \sum_{i}^{N_A} (|A - \mu_A|^2)}
\end{aligned}
\tag{3.1}
$$

By using the outlier noise reduction algorithm, there is no need to set separate thresholds for separate training domains or data sets, as the thresholds calculated by the algorithm will scale accordingly. In figure 3.2 thresholds produced by the algorithm are shown as a result of the input signal y, where the noise reduction coefficient (C_n) is equal to two.

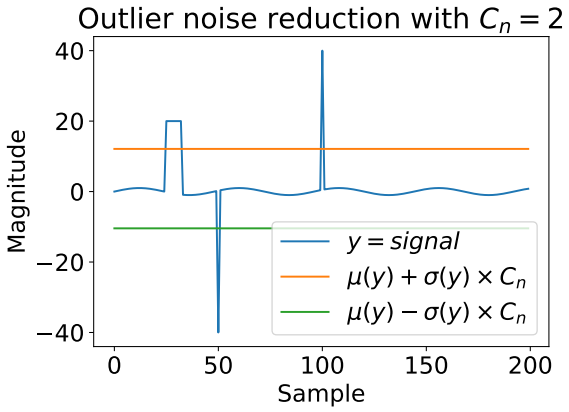


Figure 3.2: Plot demonstrating the outlier noise reduction algorithms thresholds (green and orange), where the noise coefficient (C_n) is equal to two. Tensors that contain samples above (the orange) or below (the green) the thresholds will be removed.

3.4 Software and hardware

The programming language used is Python. Python is a popular high-level programming language, and in this case it is used as an open source alternative to MatLab. The reason for using Python is the broad range of

packages available to work with Python. One of these packages is Keras[5], which is a high-level wrapper for frameworks used to train neural networks. Among other frameworks, Keras supports TensorFlow, which is Google's framework for training DNNs on NVidias' GPUs. Keras with TensorFlow utilizing two NVidia 1080 GTX GPU's with 8 GB of ram each is used in this thesis. In addition, the following open source Python packages are used and worth mentioning:

- **NumPy** for scientific computation with Python[6]
- **SciPy** containing various science, mathematics and engineering tools[7]
- **BioSPPy** for bio signal processing[8]
- **Pandas** for high-performance data structures[9]

Take notice of that when hyperparameter values are not mentioned, they are set to be default values.

[5]www.keras.io - accessed February 15, 2019
[6]www.numpy.org - accessed February 15, 2019
[7]www.numpy.org - accessed February 15, 2019
[8]www.pypi.org/project/biosppy/ - accessed February 15, 2019
[9]pandas.pydata.org - accessed February 15, 2019

Chapter 4

Experiment 1: Domain Analysis

In the context of combining and comparing DL architectures and results with the previous research performed at IVS, one might ask if ACC data is needed in the first place. It would, by all means, be better to use non-invasive sensors, such as the ECG, if the ECG could provide the same supportive and diagnostic value as the accelerometer. Therefore, this chapter will consist of a study where the goal is to compare the ACC domain and the ECG domain in a DL context. This will be done by making several DNN architectures tasked to solve the regression problem introduced in this chapter. In addition, this experiment will be a preparatory work for experiment two and three, as the best performing architectures from this first experiment will be utilized in the other two.

4.1 Idea

Given an input signal (X) represented in a specific domain (α), an output signal ($\hat{Y}$) represented in another domain (β), and a neural network that transforms the input signal to the output signal ($F_{\alpha\to\beta}(X^\alpha) = \hat{Y}^\beta$), the more features the input-signal contains compared to the output-signal, the better the transformation should be, provided that both signals represent the same real world features. If a neural network is trained to predict an ECG signal given ACC data as input, it should do better than another neural network trained to predict an ACC signal given ECG as input. In general, the probability of a neural network transforming the input (X) to the real output (Y), is higher going from α to β than vice versa, if α contains more features than β, as shown in equation 4.1.

$$P(\hat{Y}^\beta = Y|F_{\alpha\to\beta}(X^\alpha)) > P(\hat{Y}^\alpha = Y|(F_{\beta\to\alpha}(X^\beta)) \tag{4.1}$$

This is equivalent to saying that the error ($\mathcal{E}$) of a neural network transforming from α to β is smaller than the error of a network transforming from β to α, as shown in equation 4.2.

$$\mathcal{E}(\hat{Y}^\beta, Y^\beta) = E^\beta < \mathcal{E}(\hat{Y}^\alpha, Y^\alpha) = E^\alpha \tag{4.2}$$

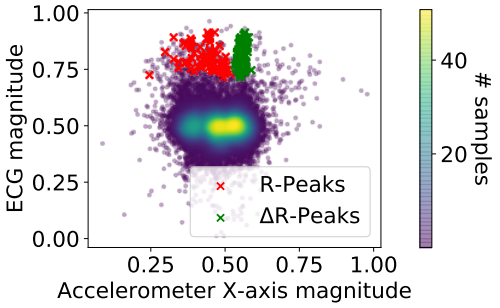
Despite the fact that the ECG signal contains less numerical information than an ACC signal (in fact, three times less), the difference in size might not be of importance when converting between the domains. ACC and ECG measures separate features, ECG the change in biopotial, ACC the movement of the heart. To reinforce this claim with an analogy, one might think of the ACC and ECG data as two pictures where the ECG picture has fewer pixels than the ACC picture. Despite the fact that the ECG "picture" has a lower resolution, it might be portraying a more relevant area.

4.2 Analyzing the data

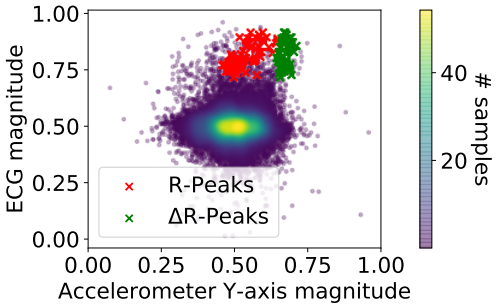
The data used for this experiment is both the classified data and the unclassified data from the AP study, as the classes are not needed for making the converter regressors. The AP study data is used as it contains the most heart conditions. Prior to making the regressors, one might ask what features the regressors need to learn when transforming from ACC data to ECG data, and vice versa. In general, the regressors need to learn the correlation between the magnitude of the ACC data and the magnitude of the ECG data. In figure 4.0, these relations are shown as scatter plots, combined with a representation of the density of the relations (color intensity), and a R-Peak feature correlation demonstration (red and green). To create the R-Peak feature demonstration, a R-Peak detector algorithm from the python

library *bioSPPy* has been applied to the ECG data. This algorithm returns the index of the R-Peak features in the ECG data. The ECG R-Peak elements and the corresponding elements in the ACC data are plotted (red). As the R-peak feature of the biopotential is occurring before it is translated to movement in the heart muscle, a lag tolerance in the ACC data is added in the R-Peak feature demonstration (green), selecting the max of the ±10 surrounding elements of the R-Peak feature element in the ACC data.

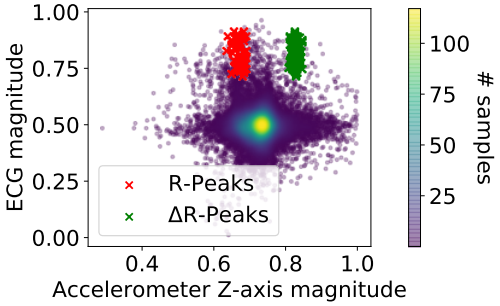
In addition to providing an estimated visual representation of the distribution of the entire dataset, figure 4.0 elucidates the lack of linear correlation and shows that prominent features in one domain might not be equally visible in the other domain. For example, as shown in figures 4.0a and 4.0c, the R-peak feature demonstration shows that the maximum value in the ECG domain does not nessesarily relate to the maximum value in the ACC domain. This reveals some of the complexity of doing the transformation, as one can easily determine the R-peak feature in the ECG domain, whereas no patterns in the magnitude of the ACC data are seen. The magnitude is around its average during these R-Peak features. Also, as seen in figure 4.0c, a wide range of ACC values corresponds to a smaller range of ECG values, and vice versa (as the figure takes form as a + sign). Lastly, figure 4.0 reveals that there is a bigger spread in the majority of the samples (bigger spread of brighter color) in the ACC domain than in the ECG domain.



(a)



(b)



(c)

Figure 4.0: (a) The direct correlation between the X-axis of the ACC data and the ECG data of 30000 (1 minute) randomly selected data samples from the entire data set. The brightness of the colors represent the density of the correlation, calculated using the Kernel-Density Estimation algorithm with Gaussian kernels. The brighter the color, the higher the density. The red points (R-Peaks) are 500 R-Peak features selected from the ECG data using the *bioSPPy* python library. The green points (ΔR-Peaks) are the same features, but with the max element within ±10 samples for each R-Peak feature in the ACC domain. Noise has been removed by using the noise reduction algorithm described in equation 3.1. (b) And (c) as for (a), but with ACC-axis Y and Z, respectively. If the ECG and the ACC domain had been equal, the plotted relations would have been linear.

4.3 Model validation and evaluation

To secure accurate and general measures of the regressors' performances, each regressor is trained on data from all the pigs, except from two that was separated to be used as test data.

4.3.1 Metrics

The measure of "how good a model is" is quite complicated, as there is no perfect way of quantifying the similarity between two signals. For this reason, the joint results of multiple metrics will be used as the quantification of the regressors' performance. The metrics used are shown in table 4.1. The evaluation process of a regressor is done by doing predictions for all tensors in the test data and using the metrics to measure the equalities between the true output (Y) and predicted output ($\hat{Y}$).

Metric	**Measures**
Mean Square Error (MSE)	Squared mean distance
Absolute Pearson Correlation Coefficient (APCC)	Absolute Linear similarity
MSE of Fast Fourier Transform (MFFT)	Squared mean frequency distance

Table 4.1: Metrics used for measuring similarities between tensors, ranking the regressors.

To measure the equality of two ACC signals, the dimensions in both ACC signals will be compared to each other, giving three equality measures for one ACC prediction. The mean of all three equality measures will be the final equality estimator for each metric comparing the ACC regressors.

4.3.2 Baseline

All regressors will be compared to three baseline regressors (B) for each domain transformation, one statistical and two random. Each baseline regressor works by outputting a single output ($\hat{Y}_B$) for all inputs (X). The statistical baseline regressor (μ) works by always outputting the average of all the training data in respect to its target domain, as shown in figure 4.1c and 4.1d. The μ baseline looks random due to the fact that the training tensors are divided by samples and not features. The two random regressors work in a similar way as the μ baseline. One of them always outputs samples drawn from the uniform distribution ($\mathcal{U}$), as shown in figure 4.1a, and one

always outputs samples drawn from the Gaussian normal distribution ($\mathcal{N}$), as shown in figure 4.1b.

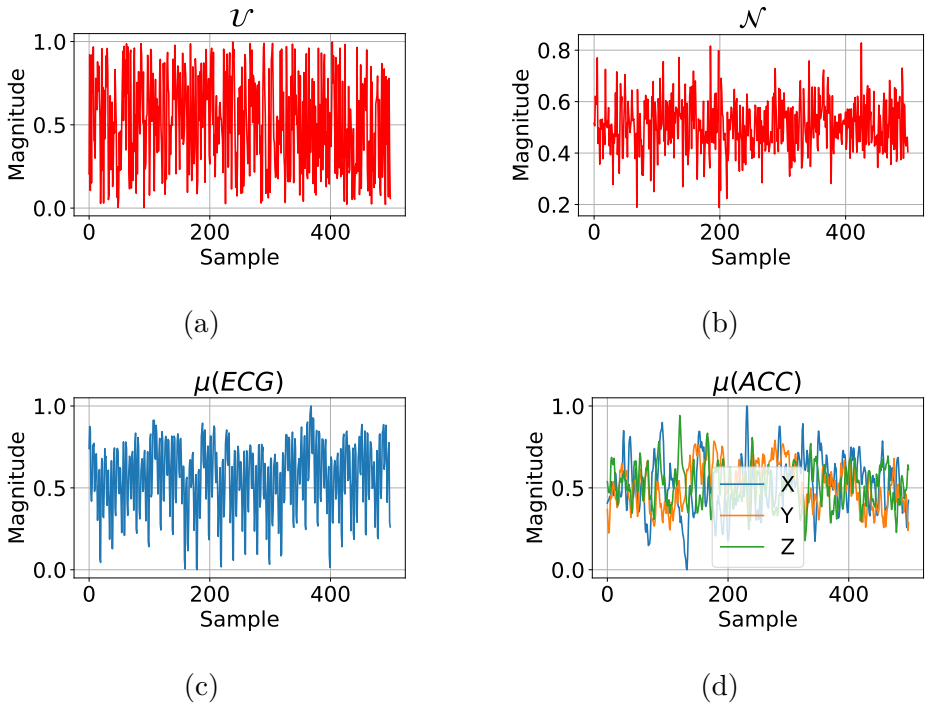
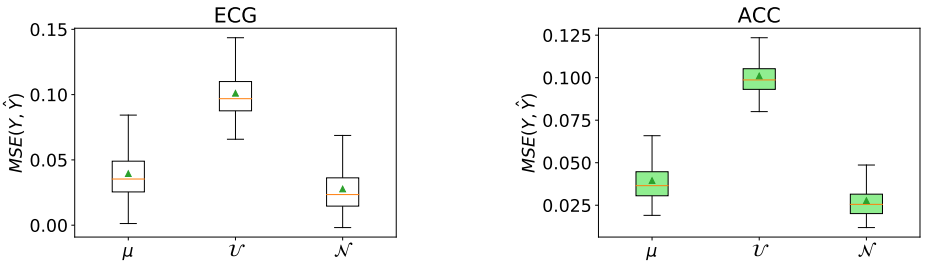


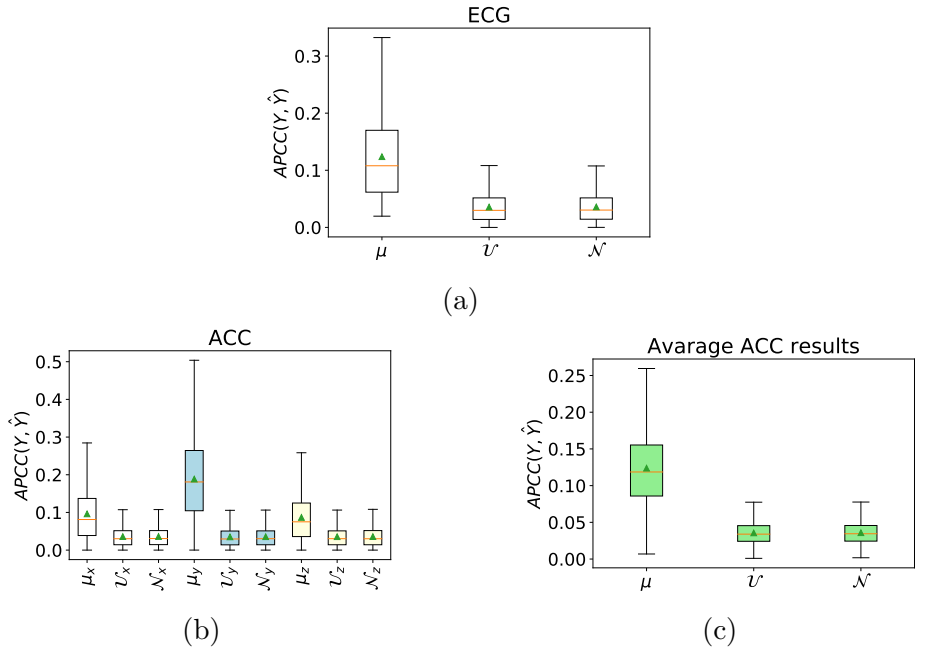
Figure 4.1: (a) Shows 500 samples drawn from the uniform distribution, and (b) 500 samples drawn from the Gaussian normal distribution with mean = 0.5 and standard deviation = 0.1. (c) Is the average of all the ECG training data. (d) Shows the average of all ACC training data. (a), (b), (c) and (d) are the baselines output tensors acting as baseline regressors (B).

In figure 4.2, the MSE of the baseline regressors ($\hat{Y}_B$) and the test data (Y) are plotted as box plots. In this figure, it becomes clear that the Gaussian noise regressor, $\mathcal{N}$, has a lower distance to the test data than the random based regressor, using the MSE as metric.



(a) (b)

Figure 4.2: (a) Shows a box plot of the MSE function metric of all true outputs and the baselines, given ECG as output. (b) As for (a), but with ACC as output. The closer MSE is to 0, the more equal the signals are.



(a) (b) (c)

Figure 4.3: (a) Is a box plot of the APCC function metric of all true outputs and the baselines, given ECG as output. (b) As for (a), but with all ACC dimensions as output. (c) Is the mean over all ACC axes plotted in (b). $NCC(Y, \hat{Y}) = 1$ equals complete linear simularity.

When using the APCC metric, both random based regressors perform equally well, as shown in figure 4.3. As the MSE metric yields a more divided performance for the random based regressors (figure 4.2), the APCC produces a better metric for measuring the "distance from noise". The Gaussian based regressor ($\mathcal{N}$) naturally performs better using MSE than the uniform based regressor ($\mathcal{U}$). $\mathcal{N}$ has a higher chance of outputting the average of the real output, than $\mathcal{U}$, making the distance smaller over all for $\mathcal{N}$. Furthermore, the APPC performance for both of the random based regressors is close to

0, as the Pearson Correlation Coefficient is undefined when the slope of the correlations is 0 (in the case of $\mathcal{U}$), and when there is no linear correlation (in the case of $\mathcal{N}$), resulting in an equal performance for each random based regressor using the APCC metric.

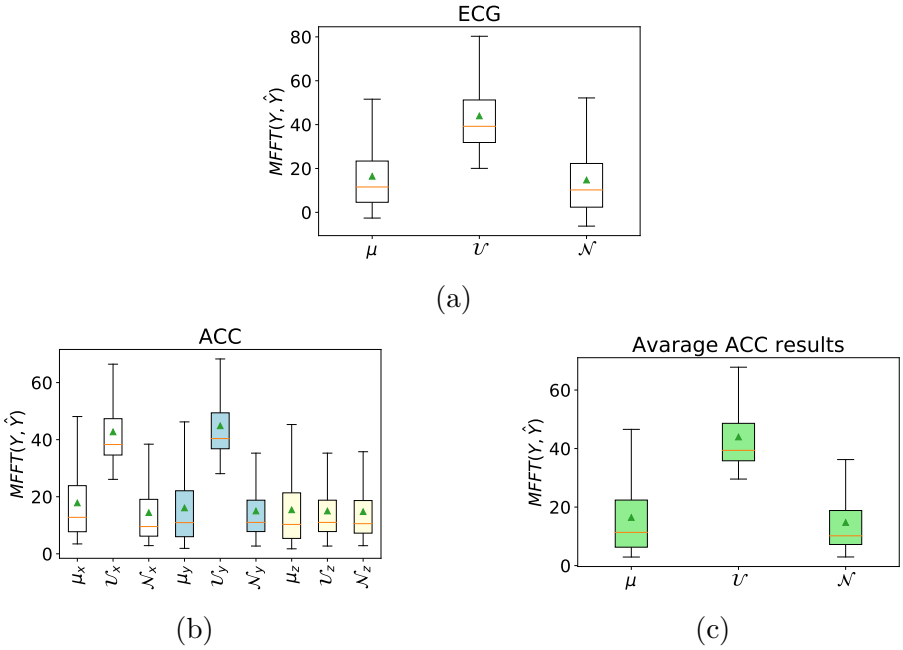


Figure 4.4: (a) Is a box plot of the FFT function metric of all true outputs and the baselines, given ECG as output. (b) As for (a), but with all ACC dimensions as output. (c) is the mean over all ACC axes plotted in (b). $MFFT(Y, \hat{Y}) = 0$ equals no distance between friequencies in Y and $\hat{Y}$.

In figure 4.4 the mean square error of the difference in the frequency domain for each regressor is shown. The distance between the baseline models and the trained models will be used as a measure when comparing the trained models.

4.4 Implementation

For each data set, four networks were trained, each solving the regression problem going from one domain to another ($\alpha \rightarrow \beta$), and vice versa ($\beta \rightarrow \alpha$). During training, for each epoch, the weights are stored if the validation error is better than the previous best validation error. This way, the regressors can be allowed to over-fit, as the best states are always kept. The optimizer used during training was the *ADAM* optimizer. When training the ECG domain to the ACC domain regressor, three parallel networks were trained,

each network responsible for predicting one dimension of the accelerometer signal.

4.4.1 Dense neural network - D.1

The first network architecture ($D.1$) is a simple dense neural network consisting of two hidden layers (H^1 and H^2), each with 24 nodes. The input tensor is a matrix consisting of M time-steps, and n features. For example, an accelerometer input tensor has 500 time steps, and 3 features. An ECG input tensor has 500 time steps, and one feature. Each neuron is activated using the leaky *ReLU* activation function, except for the output neurons that are activated using the linear function. Following the two hidden layers, the resulting matrix is flattened, resulting in a $M \times 24$ long vector. This vector is fed to the last dense layer, creating the output, as shown in figure 4.5. During training, the ADAM optimizer was used, and dropout with a 50% chance was added after each hidden layer. The loss function used is the Huber loss function, as the data set is not validated, and may contain many outliers.

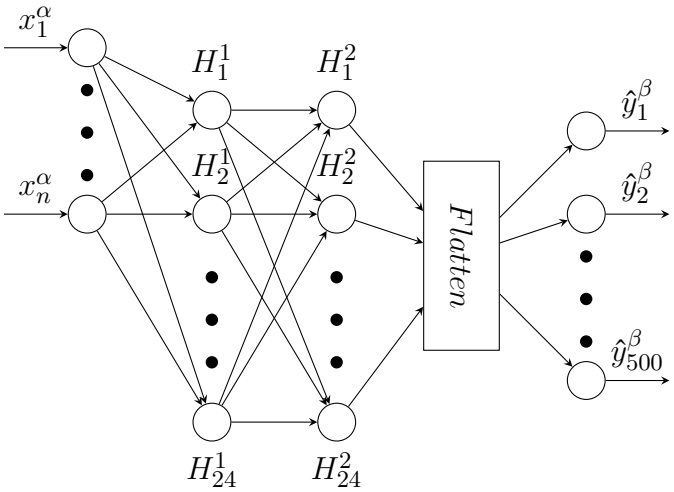


Figure 4.5: Graph demonstrating the D.1 network transforming from domain α to β. The hidden layer consists of 2×24 hidden nodes with the *leaky ReLU* activation function with the fixed parameter a beeing 1×10^{-8}. The output tensor is the product of a Dense network with 500 neurons all activated by the linear activation function.

4.4.2 Dense modular neural network - D.2

The second network architecture ($D.2$) is a modular network inspired by the residual network [9], constructed with a specified number of modules,

as shown in figure 4.6. It contains "skip connections" where each module is a product of the input, creating a "gradient highway" enabling the network to converge faster [9]. Each module consists of n nodes, activated using the *ReLU* activation function. The output tensor is produced in the same manner as in $D.1$ - it is the product of n^β neurons activated by a linear activation function. Here, the traditional *ReLU* and the MSE loss are used, as all modules have a skip connection, and are therefore less affected by possible dead neurons.

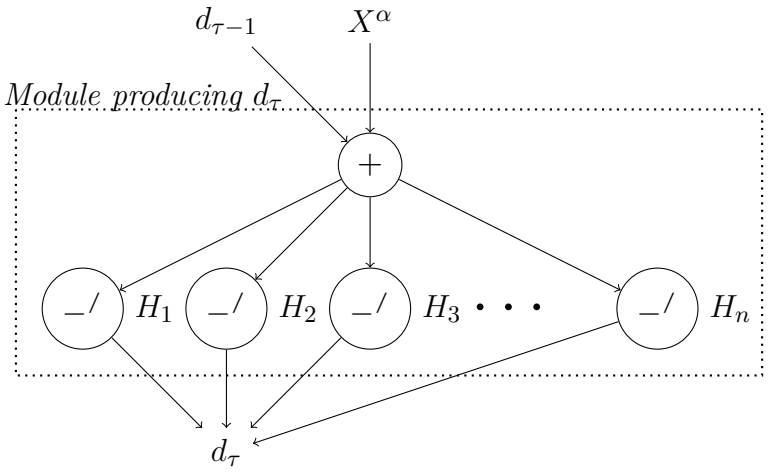


Figure 4.6: Graph demonstrating a D.2 network module. A network consists of several modules, all adding the original raw input (X^α) to the previous layer's activation ($d_{\tau-1}$). The hidden layer in the module consists of n nodes with the *ReLU* activation function. The output tensor (not shown in the graph) is the product of a Dense network consisting of n^β neurons, all activated by the linear activation function as per D.1

4.4.3 Convolutional neural network - C.1

The third network ($C.1$) is a network consisting of convolutional blocks, as well as multiple max pooling blocks, acting as regularization blocks. Each convolutional layer is followed by a batch normalization layer, a *ReLU* activation layer, and lastly a max pooling layer, as shown in figure 4.7. Due to the fact that C.1 contains batch normalization layers the MSE loss is used.

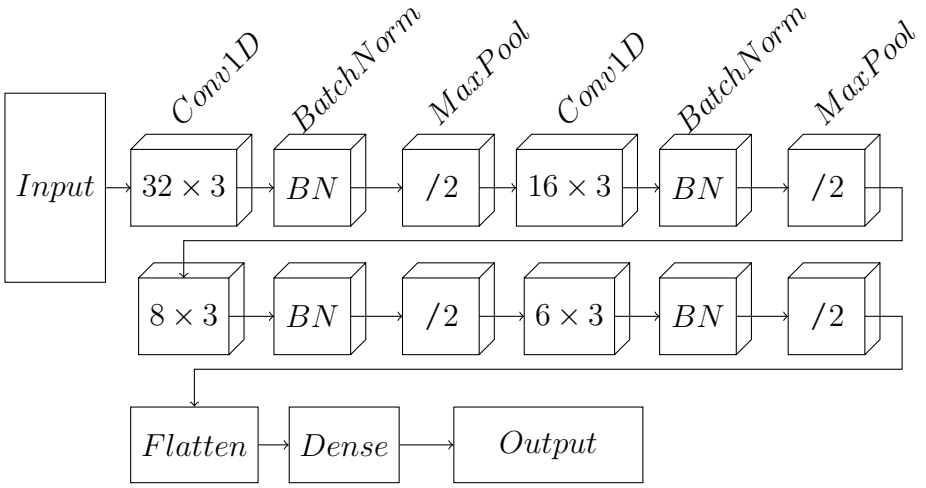


Figure 4.7: Figure demonstrating the C.1 regressor network. Each batch normalization layer is followed by a $ReLU$ activation block, not drawn in the figure. The notation of the convolutional blocks is: number of filters times the size of the kernel, respectively.

4.4.4 Auto-encoded weights network - C.AE

The second convolutional neural network architecture, the C.AE network, is a network based on an autoencoder (introduced in 2.6.1). The C.AE network training process differs from the other networks, as some "pre-training" is performed, and the domain transformation is solved in the z domain. First, two autoencoders are trained, one for each domain (α and β). If these networks are trained successfully, both networks will be able to transform their domains into a lower dimensional space, a "compressed" latent space (z). Each autoencoder consists of an encoder (q) and a decoder (p), where the encoder maps from the original input domain (X) to the latent space (z), and the decoder from the latent space to a reconstructed input ($\hat{X}$). Instead of using the original domains as targets, the latent spaces representing the targets will be used to train a third network ($\mathcal{F}$). By combining the two autoencoders, using the encoder from one network and the decoder from the other network, and by adding a third network in between ($\mathcal{F}(z)$), $\mathcal{F}(z)$ can be forced to learn and do the domain transformation in a lower dimensional space by freezing the weights. The network's $C.AE$ architecture is shown in figure 4.8, and its output ($\hat{Y}$) in equation 4.3.

$$\hat{Y}^{\beta} = p^{\beta}(\mathcal{F}_{z^{\alpha} \to z^{\beta}}(q^{\alpha}(X^{\alpha}))) \tag{4.3}$$

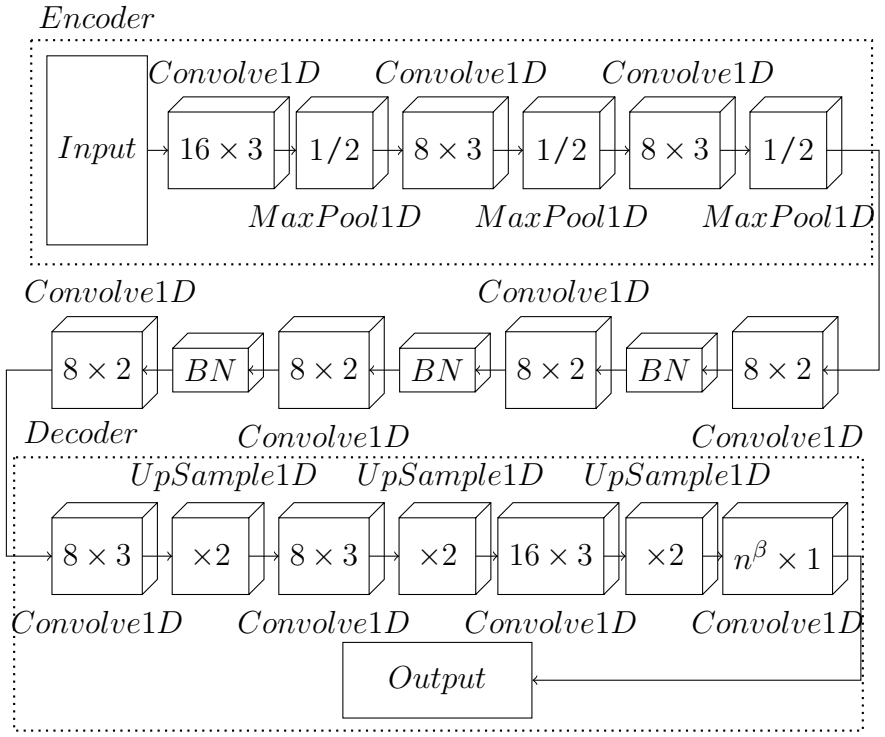


Figure 4.8: Graph demonstrating the autoencoded regressor network $D.AE$. Each $Convolve1D$ layer consists of N filters with M size, denoted in the convolutional blocks as $N \times M$. In the last convolutional layer in the decoder, the number of layers chosen is based on the dimensional size (n) of the target domain (β). For example, if the target domain β is the accelerometer domain, $n^\beta = 3$. After each $Convolve1D$ layer in the encoder and the decoder, an activation using the $ReLU$ activation function is performed. Between the encoder and the decoder, each batch normalization layer (BN) is followed by the same activation.

4.5 Results

To do the final evaluation, the models were asked to predict the test tensors from the two test pigs. For each model the training loss and some examples are presented, and a final comparison between the models and their scores with respect to the baselines is done in section 4.6. The training process was stopped after 64 epochs for each regressor[1].

[1]Setting the epoch above 64 was attempted for several regressors. However, this did not improve the performances. To save time, each regressor was set to train for 64 epochs.

4.5.1 The D.1 regressor

The losses during training of the D.1 regressors are shown in figure 4.10. The training converges after about 50 epochs for both regressors. Examples of predictions done by the D.1 regressor are shown in figure 4.9. In figure 4.9a the ECG signal predicted when given ACC as input is demonstrated. It clearly shows that the regressor struggles to reproduce the ECG signal. However, take notice of that the regressor manages to predict some of the heart's movement during isovolumic relaxation (around 400 samples) in the Z-axis of the ACC domain, as shown in figure 4.9d. The models' performances with respect to the metrics are shown and discussed in section 4.6.

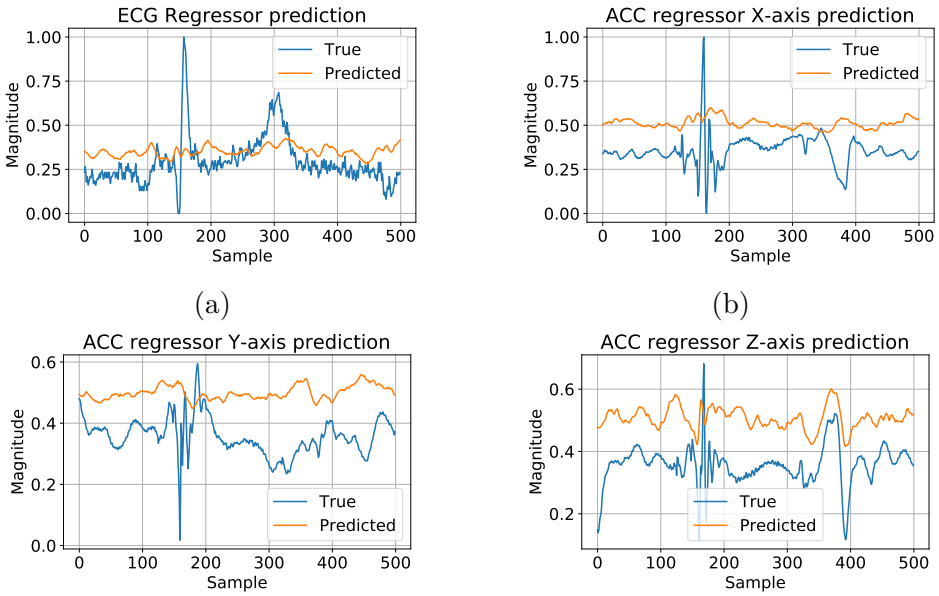
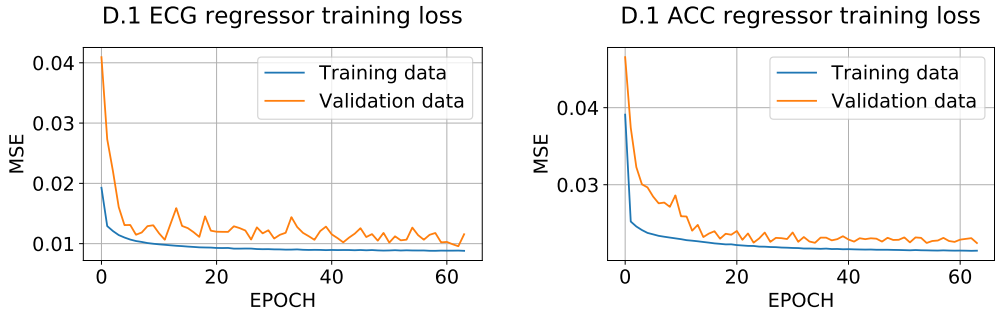


Figure 4.9: (a) Example of an ECG prediction given ACC as input, using the D.1 regressor. (b), (c), And (d) as for (a), but given ECG as input, and ACC as prediction domain.

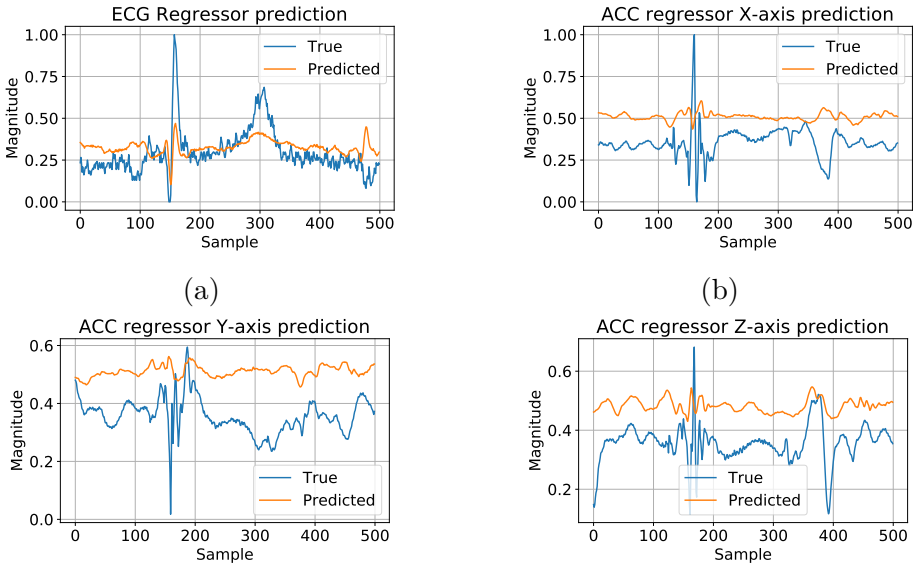


(a) (b)

Figure 4.10: The loss for the regressors during training. (a) Shows the loss for the D.1 regressor converting from ACC to ECG, and (b) the regressor converting from ECG to ACC.

4.5.2 The D.2 regressor

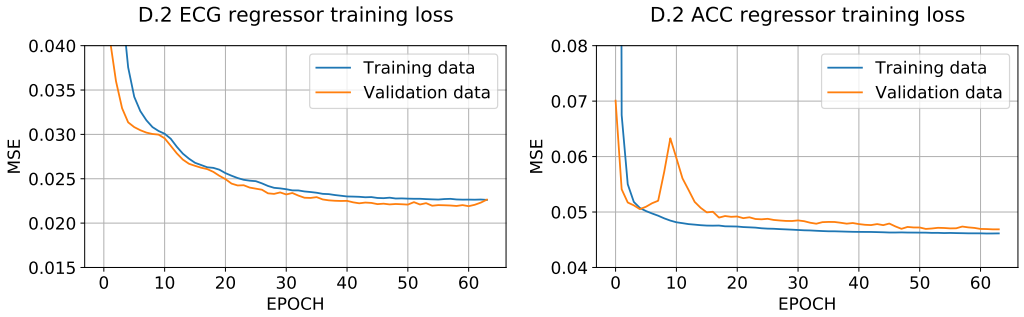
Examples of predictions done by the D.2 regressor are shown in figure 4.11. As seen in figure 4.11a, the D.2 regressor performs more convincingly than the D.1 regressor when predicting an ECG signal, given ACC as input. The losses during the training process are shown in figure 4.12. The models' performances with respect to the metrics are shown and discussed in section 4.6.



(a) (b)

(c) (d)

Figure 4.11: (a) Example of an ECG prediction given ACC as input, using the D.1 regressor. (b) As for (a), but given ECG as input, and ACC as prediction domain.

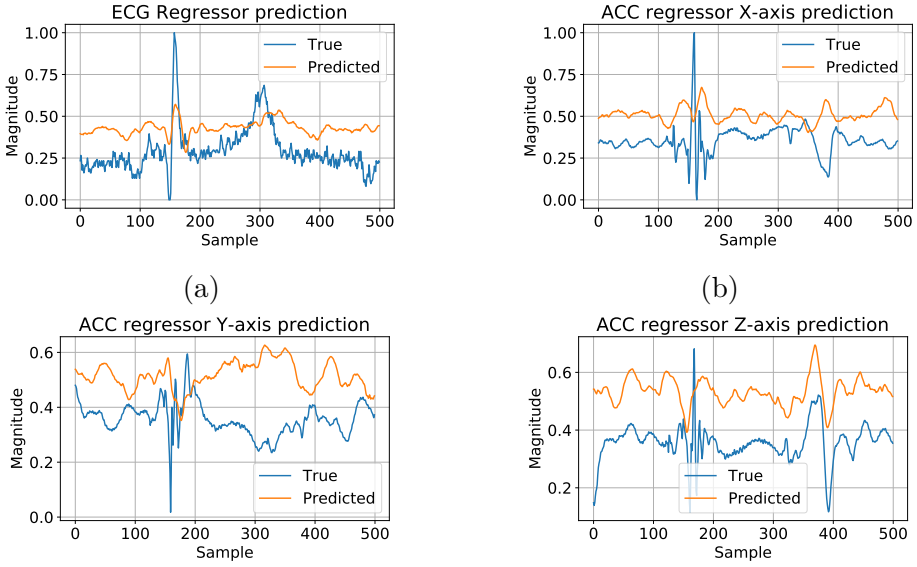


(a) (b)

Figure 4.12: (a) The MSE loss of the ECG regressor during training. (b) As for (a), but with the MSE loss of the ACC regressor.

4.5.3 The C.1 regressor

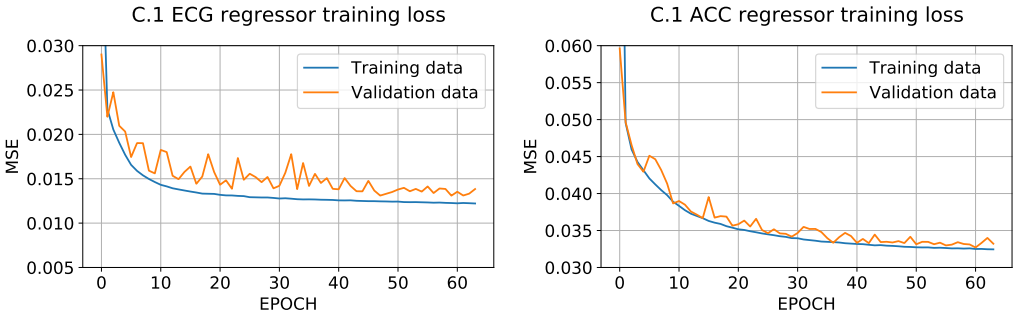
Figure 4.13 shows examples of predictions done by the C.1 regressor. The loss during training is shown in figure 4.14b. As seen in figure 4.13, the C.1 regressor relatively successfully converts from the ECG domain to the ACC domain. The ECG regressor clearly discovers the R-peak feature. The isovolumic contraction and relaxation are clearly visible in all dimensions in the ACC and ECG predictions. The models' performances with respect to the metrics are shown and discussed in section 4.6.



(a) (b)

(c) (d)

Figure 4.13: (a) Example of an ECG prediction given ACC as input, using the C.1 regressor. (b) As for (a), but given ECG as input, and ACC as prediction domain.

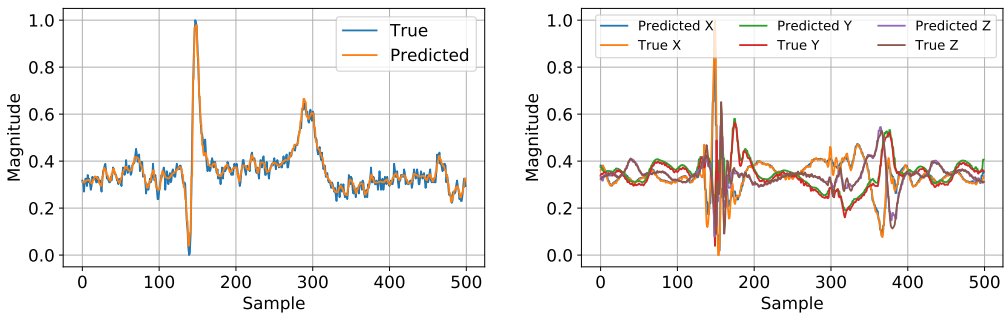


(a) (b)

Figure 4.14: (a) the MSE loss of the ECG regressor during training. (b) as for (a), but with the MSE loss of the ACC regressor.

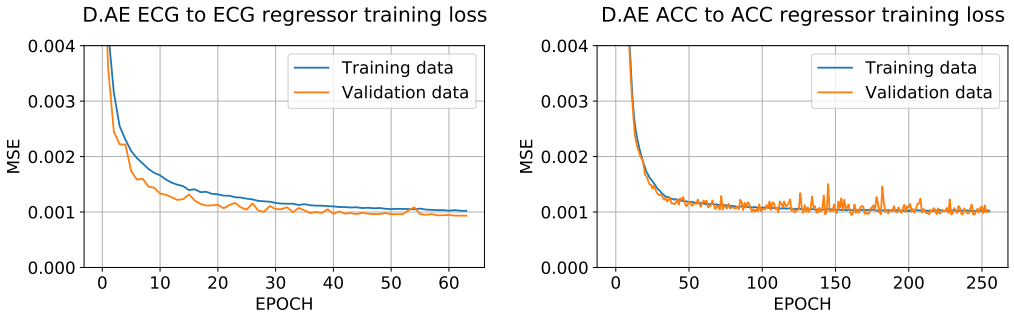
4.5.4 The C.AE regressor

Initially, two regressor were trained, one transforming ECG to a latent space (z_{ECG}) and back again to the original input ECG, and one transforming ACC to a latent space (z_{ACC}) and back again to the ACC domain. These first autoencoder networks performed well, as seen in figure 4.15. The loss of the training process of these autoencoders can be seen in figure 4.16.



(a) (b)

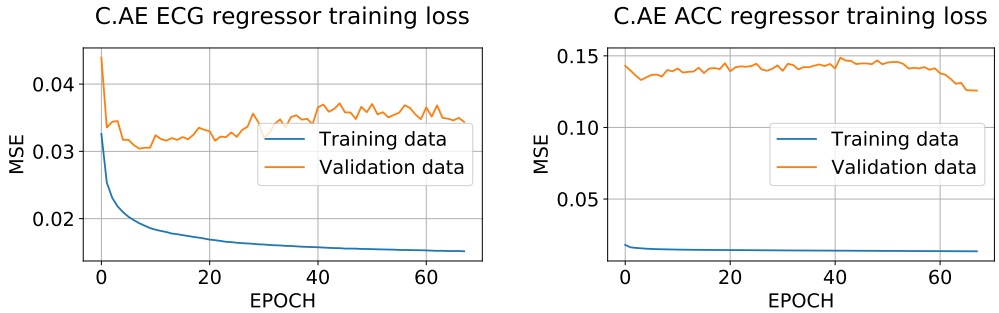
Figure 4.15: (a) Shows an example of the autoencoder compressing and recreating an ECG signal. (b) Shows an example of the autoencoder compressing and recreating an ACC signal.



(a) (b)

Figure 4.16: (a) Shows the loss for the ECG autoencoder, and (b) the loss for the ACC autoencoder.

The trained encoder from the ECG autoencoder was then followed by a new network, which again was followed by the ACC decoder from the ACC autoencoder, as introduced in section 4.4.4. The same was done for the ACC encoder and ECG decoder. The weights from the previously trained autoencoders were freezed, and the middle network was trained. The losses of the training process where the regressors transformed between the z domains are shown in figure 4.17.

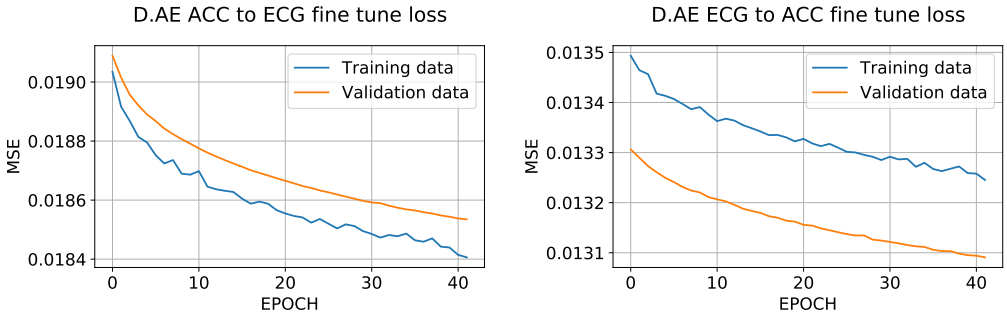


(a) (b)

Figure 4.17: (a) The MSE loss of the ECG regressor during training. (b) As for (a), but with the MSE loss of the ACC regressor.

Lastly, a fine-tuning of the networks was conducted, "un-freezing" the weights. When fine-tuning the networks, the learning rate was changed from 1 to 1×10^{-6}, and a decay[2] of 1×10^{-5} was added, forcing the $C.AE$ network to keep the weights around the values it had already learned. The fine-tuning training process was set to 40 epochs, to preserve the previously learned knowledge. The losses during fine-tuning are shown in figure 4.18.

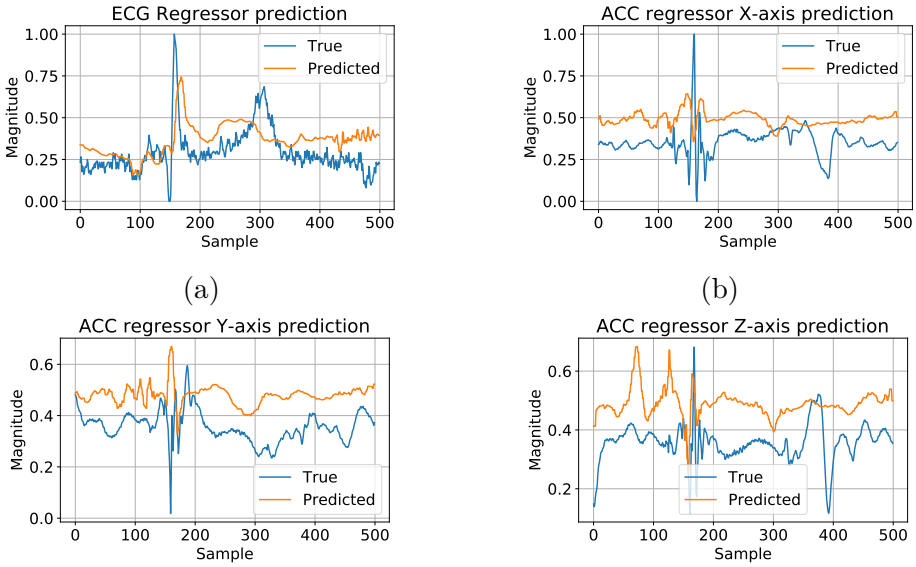
[2]A decay denotes how much the learning rate should decrease for each epoch.



(a) (b)

Figure 4.18: (a) The MSE loss of the ECG regressor during training when fine-tuning the network. (b) As for (a), but with the MSE loss of the ACC regressor.

Examples of predictions done by the C.AE regressors are shown in figure 4.19.



(a) (b)

(c) (d)

Figure 4.19: (a) Example of an ECG prediction given ACC as input, using the C.AE regressor. (b) As for (a), but given ECG as input, and ACC as prediction domain.

4.6 Metric results and observations

An overview of the performances of all the classifiers can be seen in figure 4.20. In this figure, the regressors' performances are plotted as box plots,

showing the output of all the metrics introduced in section 4.3.1. On the right hand side in each sub-plot the baseline regressors performance, introduced in section 4.3.2, are shown. Take notice of that all the regressors performed better than uniform noise baseline ($\mathcal{U}$). In sub-figure 4.20a and 4.20e, showing the ECG regressors' performances using the MSE and MFFT as metrics, all constructed regressors perform better than the baseline regressors. This in contrast to the ACC regressors, where the Gaussion noise regressor ($\mathcal{N}$) outperformed some of the constructed regressors, as seen in sub-figure 4.20b and 4.20f. When using the APPC as a metric, both the ECG and the ACC regressors outperformed the baselines.

The overall difference when converting between the two domains is smaller than expected. However, there is a slight difference in performance, as all the ECG regressors performed marginally better than the baseline regressors than the ACC regressors did. The results will be further discussed in chapter 7.

Examples of 5 seconds of prediction done by the regressors can be seen in appendix A.

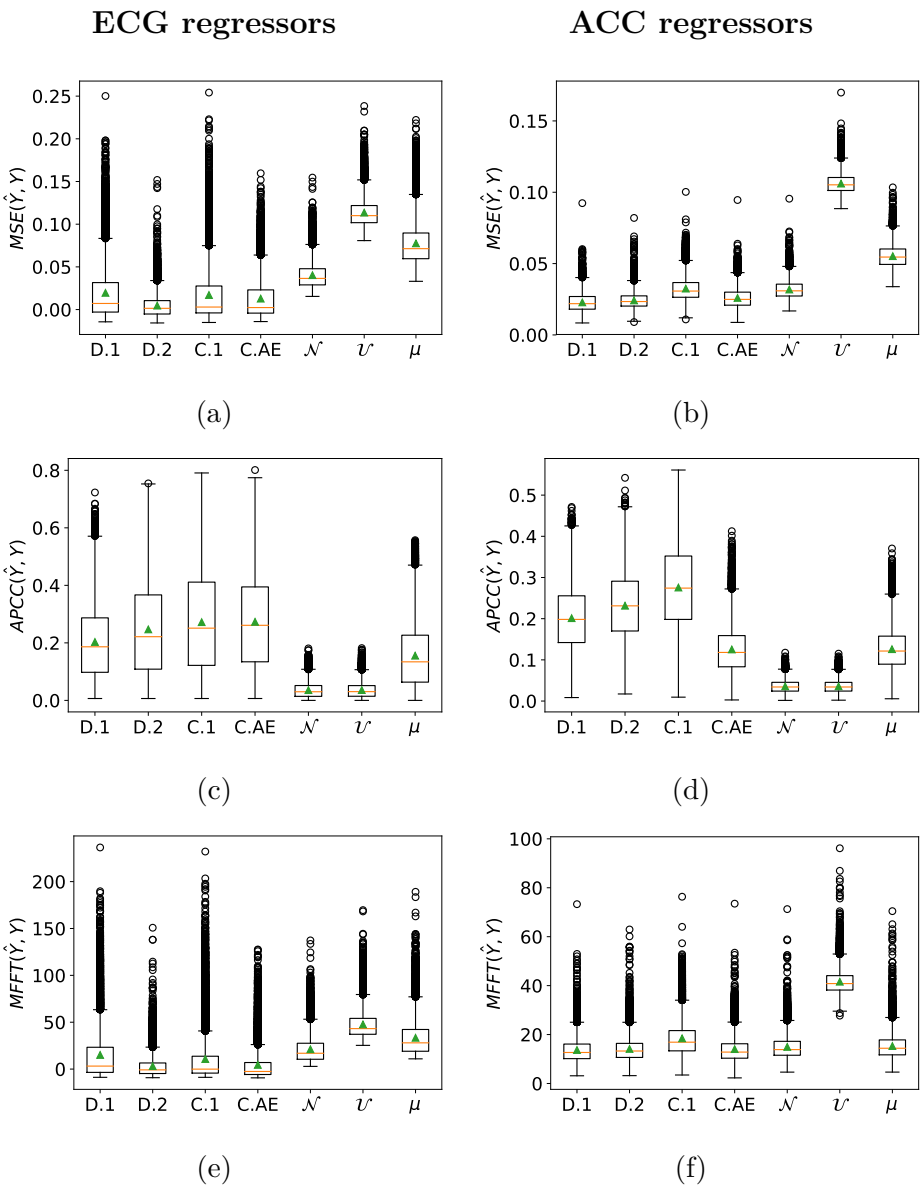


Figure 4.20: An overview of all the regressors' performances with respect to the metrics. The MSE, APCC and MFFT have been calculated for all predictions done by each regressor, and capture the errors between the predicted values ($\hat{Y}$) and the test target values (Y). The baseline regressors' scores are also plotted as a reference. (a) , (c) And (e) show the ECG regressors' scores, transforming from the ACC to the ECG domain. (b), (d) And (f) show the ACC regressors' scores, transforming from the ECG to the ACC domain. (a) And (b) show the performance using MSE as metric, (c) and (d) the performance using APCC as metric, and (e) and (f) show the performance using the MFFT metric.

Chapter 5

Experiment 2: Classifying Cardiac Heart Dysfunctions

The following chapter will introduce the classifier tasked to solve both a 3 class classification task, as well as a 6 class classification task. The model's architecture and results will be presented and compared.

5.1 Idea

The intended goal is to detect ischemia in real time using the 3-axis accelerometer attached to the heart. The data provided are data from two experiments (the AP and MKCM study), in total recordings from 21 pigs. From the AP study there are data from six heart conditions, and from the MKCM study there are data from three conditions. Ischemia, baseline and adrenaline are present in both data sets, but in the AP study, in addition to occlusion and adrenaline - nipride, fluid, and β-blockers are present. These interventions act as the targets (classes) during training and evaluation of the classifiers. Given that the two studies contain a different amounts of heart conditions, a two-part study will be conducted, where the first part asseses the possibilety of classifying three classes (adrenaline, ischemia and baseline), and the second part the possibilety of classifying additional three more classes (nipride, fluid, and β-blockers).

5.2 Analyzing the data

To get an understanding of what the classifiers need to learn, a short analysis of the data will be conducted. For this analysis, the data has been segmented into R-peak tensors, as two sequential R-peaks denote one heart cycle. In medical terms, one heart cycle starts when the hearts contracts (the P wave in the ECG signal). However, the most prominent feature of an ECG signal is the R-peak feature. Therefore, the R-peak feature will be used as the division decider. For detecting the indexes of the R-peaks, the *BioSPPy* Python library has been used. *BioSPPy* sometimes fails to detect the R-peaks, but for simplicity's sake, only R-peaks detected by the library have been used[1]. Table 5.1 shows the standard variation (SD) of the magnitude of each class from both the AP and the MKCM study.

Figure 5.1 is a 2 dimensional representation of the mean acceleration for each class. This is made by first calculating the mean over the dimensions for all tensors in the accelerometer data, producing one vector representing the x, y and z axis for each tensor. Then, the mean vector for each class is calculated, interpolated to 500 samples, and plotted. As several studies indicate [30, 5], the biggest difference between baseline and occlusion can be seen in the isovolumic relaxation phase (post systolic, around 300 samples

[1]When the difference between the R-peaks detected was too long (> 1500 samples), or too short (< 50 samples), the cycles were removed. The downside of relying on *BioSPPy* R-peak detection, is that when the ECG signal is very noisy, the library fails to detect the R-peaks, resulting in less data for the analyzis. As table 5.1 demonstrates, there is an unbalanced relation between the number of heart cycles for each class. However, the number of heart cycles per class is considered sufficient to gain an overall impression of the classes.

Class	$SD(Magnitude)[\times 10^{-2}]$	Number of cycles
Baseline	7.21	258
Occlusion	7.01	790
Adrenaline	8.81	192
β-blockers	5.49	69
Fluid	5.17	80
Nipride	4.73	85

Table 5.1: Standard Deviation (SD) of magnitude and cycles per class.

in figure 5.1), and in the early systolic phase (about the first 100 samples in figure 5.1). Figure 5.1 indicates that there is a noticeable visible difference between the classes, suggesting that deep learning networks should be able to discriminate between the classes.

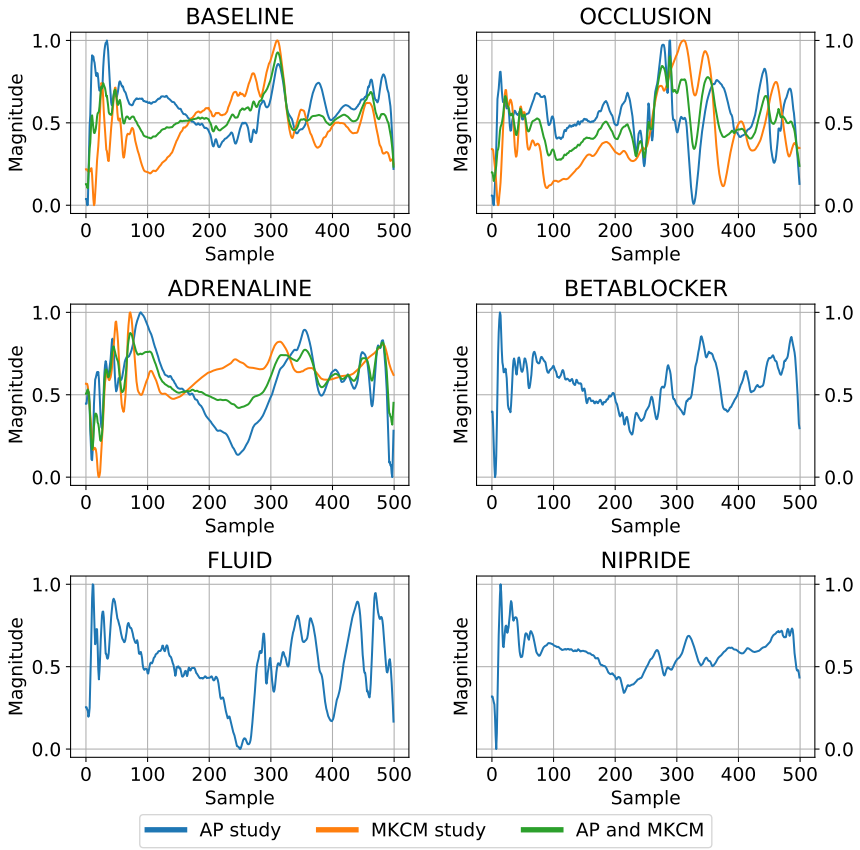
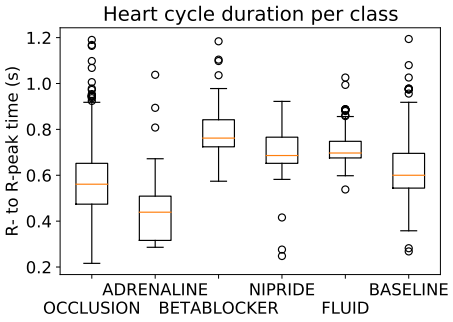


Figure 5.1: In this figure, each class tensor is reduced from 3 axes to 1 by calculating the mean over its axes. The mean for each class is then calculated and plotted for the AP (blue) and MKCM (orange) studies, and the mean for both studies (green).

On the other hand, in figure 5.1, the adrenaline does not stick out as a big of an abnormality as one might suspect. However, the classes are interpolated to 500 samples, from R- to R-peak, so there is no information about time. This motivates figure 5.2, which shows a boxplot of the R- to R-peak times for each class (left), and a table (right) of the mean (μ), and the standard deviation (SD) of the R- to R-peak times. In this figure, it becomes clear that time is an important feature and should be preserved and given to the classifier. For example, there is a big difference in R- to R-peak time between adrenaline and the other classes. For this reason, the data are not divided by R-peak features prior to training the classifiers. Also, a majority of the heart cycles in every class last for less the 1 second. This motivates splitting the training tensors into 500 samples (1 second).



(a)

Class	$\mu[\times 10^{-2} S]$	$SD[\times 10^{-2} S]$
Baseline	61.6	14.8
Occlusion	56.7	12.6
Adrenaline	44.3	11.5
β-blockers	78.8	12.0
Fluid	72.4	8.91
Nipride	70.6	11.7

(b)

Figure 5.2: (a) Is a box plot of R- to R-peak time in secounds (S) for classes from both the AP and MP study. (b) Is a table showing the mean (μ) and the standard diviation (SD) in secounds (S) for each classes R- to R-peak time.

5.2.1 Augmenting the data

One of the challenges with this study is that there is not much training data. When dealing with image classification, a lack of training data can sometimes be solved by augmenting the data - by, for example, mirroring and flipping the images, rotating, scaling and cropping them. When using augmenting methods, the amount of data increases.

Such augmentation methods can be applied to the accelerometer data. In particular, rotation about the accelerometer's axes could be a good augmentation technique. Firstly, rotation appears naturally in the data, from pig to pig, as the accelerometer is attached by hand by the clinics, and natural changes in its global rotation will occur. Also, one could argue that invariance to such rotations is a good feature for a classifier, as a real-world

implementation would require it to handle such displacement. For this experiment, the accelerometer data have been augmented, by rotating the data around a set of vectors[2], e.g $\vec{u} = (1, 0, 0)$, as demonstrated in figure 5.3. For all the training tensors, the data have been rotated 6 times around six vectors, by changing the rotation angle by $\frac{\pi}{4}$ for each rotation, multiplying the amount of data by $6 \times 6 = 36$. The counterclockwise rotation about a given vector ($\vec{k}$) by θ radians is done by dotting the vector of interest ($\vec{x}$) by a rotation matrix ($\mathbf{R}$), calculated using the Euler-Rodrigues formula [48], shown in equation 5.1.

$$\mathbf{R} = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2(bc - ad) & 2(bd + ac) \\ 2(bc + ad) & a^2 + c^2 - b^2 - d^2 & 2(cd - ab) \\ 2(bd - ac) & 2(cd + ab) & a^2 + d^2 - b^2 - c^2 \end{bmatrix}$$

$$\begin{aligned} a &= \cos\frac{\theta}{2} \\ b &= k_x \sin\frac{\theta}{2} \\ c &= k_y \sin\frac{\theta}{2} \\ d &= k_z \sin\frac{\theta}{2} \end{aligned} \tag{5.1}$$

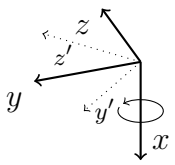


Figure 5.3: An example showing a rotation around the x-axis for vector (x, y, z), producing new vector (x, y', z').

Test, validation, and training data are separated prior to augmentation, as shown in figure 5.4. After the data has been augmented, for each tensor in the data set, the mean of each dimension (x, y and z) has been subtracted. In appendix B a comparison between a model trained on augmented data versus a model trained on data not augmented is carried out, and shows that models trained on augmented data perform better than models not trained on augmented data.

[2]The vectors are $\vec{u}_1 = (1, 0, 0)$, $\vec{u}_2 = (0, 1, 0)$, $\vec{u}_3 = (0, 0, 1)$, $\vec{u}_4 = (1, 1, 0)$, $\vec{u}_5 = (0, 1, 1)$, $\vec{u}_6 = (1, 0, 1)$

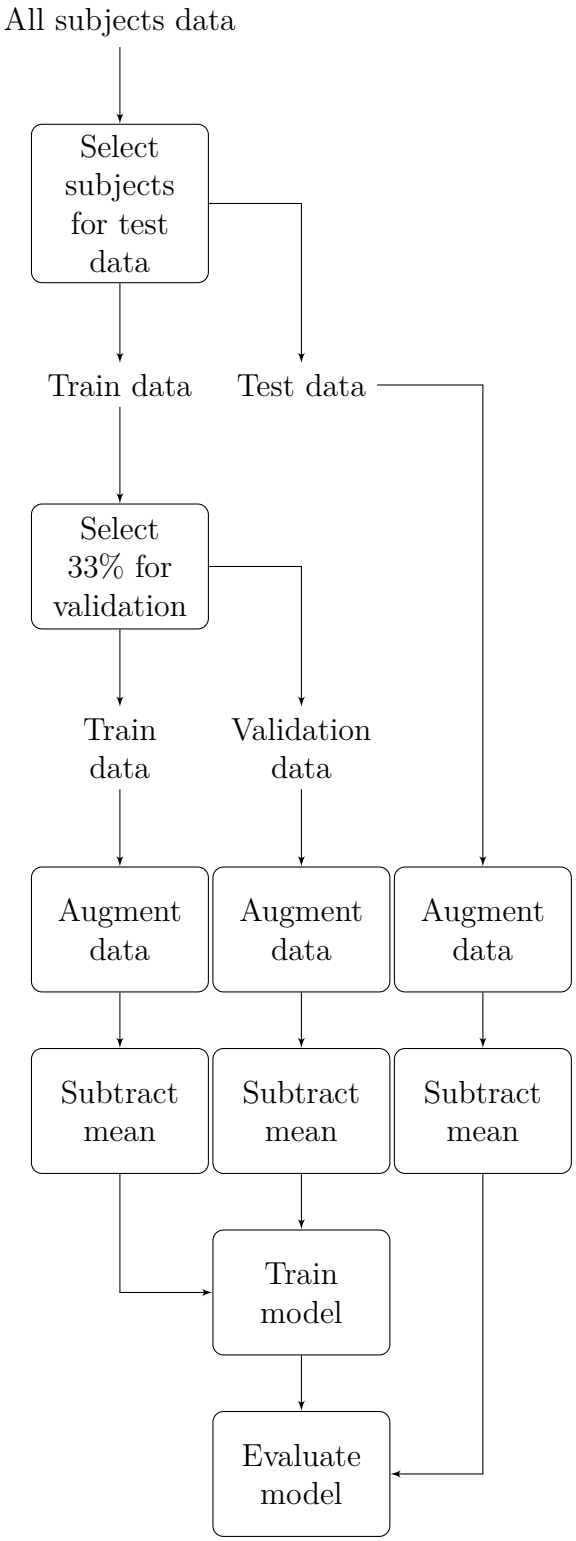


Figure 5.4: Float diagram demonstrating the process of training a model when augmenting data. First, subjects for evaluation are selected. Then 33% of the training data are selected as validation data, before all the data is augmented. The mean for each axes are subtracted, the model trained, and finally, the evaluation is done using the test data.

5.3 Domain knowledge extraction

To investigate whether the classifiers can bring additional clinical knowledge to the field, several analyzation techniques will be carried out. The following analysis of the classifiers will be carried out. This will be done to get a deeper understanding of the data, and what the classifiers consider as important features.

5.3.1 Context removal analysis

In DL, the term *occlusion* is used as a name for a method that evaluates what features in the input data a DNN focuses on. In this thesis, to differentiate between medical occlusion and DL occlusion, DL occlusion will be named *context removal*. Context removal is done by removing some information in the input data, feeding it to a trained model, and see what this does to the model's confidence [49]. The idea is that if features that are considered important to the classifier are removed from the input data, the classifier's confidence will decrease. This way, by iteratively removing different small bits of information in the input data, and by recording the scores of the classifications for each iteration, it is possible to quantify what features that are considered important to the classifier. A float diagram of this process can be seen in figure 5.5. The time context removal process is described in the following section.

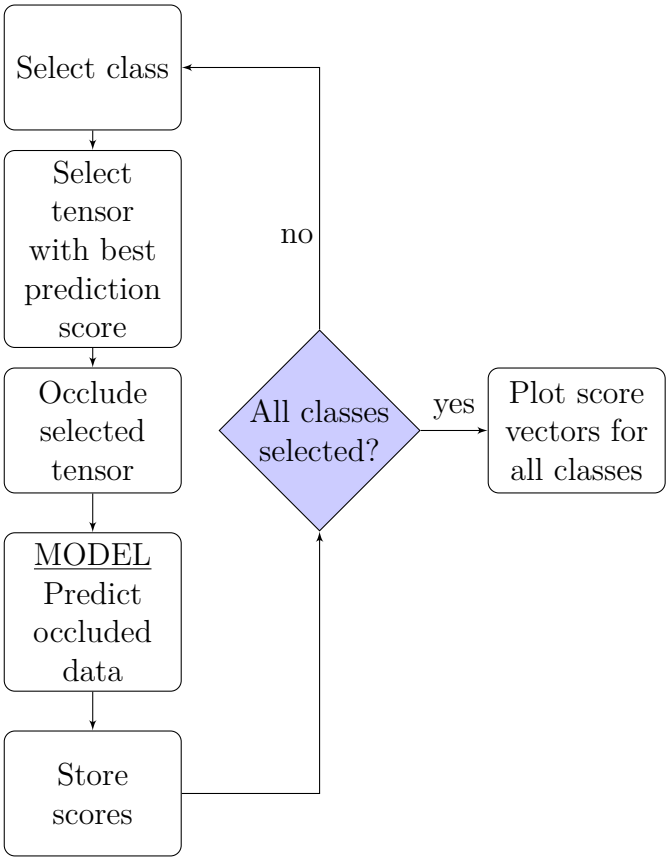


Figure 5.5: Float diagram demonstrating the context removal process. The class of interest is selected, and the input tensor that produced the best score for the class of interest is selected. The selected tensor is then occluded multiple times, where different information in the signal are removed. Following this, the model is tasked to predict the data with removed context, and store the produced scores. This is repeated for all classes, and finally the scores for each class is plotted for visual inspection.

Time feature context removal

Time feature context removal is done by iteratively selecting parts of the input tensor's magnitude to be zero. In figure 5.6, examples of five magnitude context removals are plotted, where the context removal window is set to be 100 samples long. The context removal window denotes how many of the samples that will be set to zero for each iteration. An entire time feature context removal process is iteratively done over all samples for the data, so that all samples have been set to zero at least once.

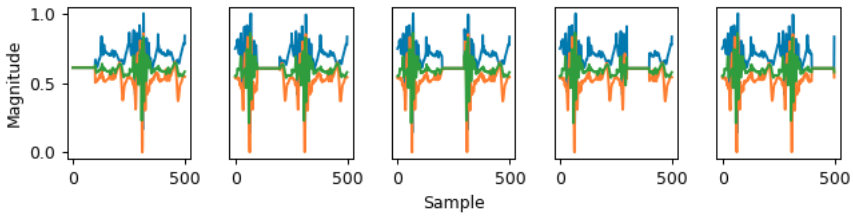


Figure 5.6: Plot showing examples of five context removals of time magnitude data for different time periods. In the plots above, the context removal window is set to be 100 samples long.

5.3.2 Maximum activation analysis

The input that maximizes the models confidence for each class will be computed and shown. The process works by finding the changes needed in the input to get a max activation of the class' output. The computation is done as shown in equation 5.2, using the Keras-vis library [50].

$$\frac{\partial LinearLoss}{\partial input} \tag{5.2}$$

5.4 Implementation

Softmax and cross-entropy loss were used, and the classifiers were optimized using the ADAM optimizer. Softmax is used in the last layer, together with cross entropy loss. After the classifier was trained, it was evaluated by calculating the precisions, recalls and F1 scores for the test data set.

5.4.1 3 class classification

For the 3 class classification, data from both the MKCM and AP study were used. 4 pigs were extracted to be used as the test data set. The classes that the classifier was tasked to classify are:

- Baseline
- Adrenaline
- Occlusion

5.4.2 6 class classification

For the 6 class classification, data from only the AP study was used. 2 pigs were extracted to be used as the test data set. Each classifier was trained to classify 6 classes. The classes that the classifier was tasked to classify are:

- Baseline
- Adrenaline
- β-blockers
- Nipride
- Fluid Loading
- Ischemia

5.4.3 The classifier

Variations of all the networks introduced in the previous experiment were tested. The best performing models were the convolutional models, as the APPC metric in figure 4.20c indicated. For this experiment, a variation of the C.1 network was developed where the kernel sizes and the number of filters were modified[3]. The bigger the filters, the more time context a convolutional kernel can "see". The classifier consists of three convolutional layers, each followed by a batch normalization layer, an activation layer and a max pooling layer. The last part of the classifier is two dense layers with activation, with a dropout layer in between. The activation used is the Leaky ReLU activation function. The dropout factor is set to 0.5, giving the previous weights a 50% chance of being 0 during training.

[3]Just using the "raw" D.1 network achieved a F1 score of 0.71 when classifying six classes.

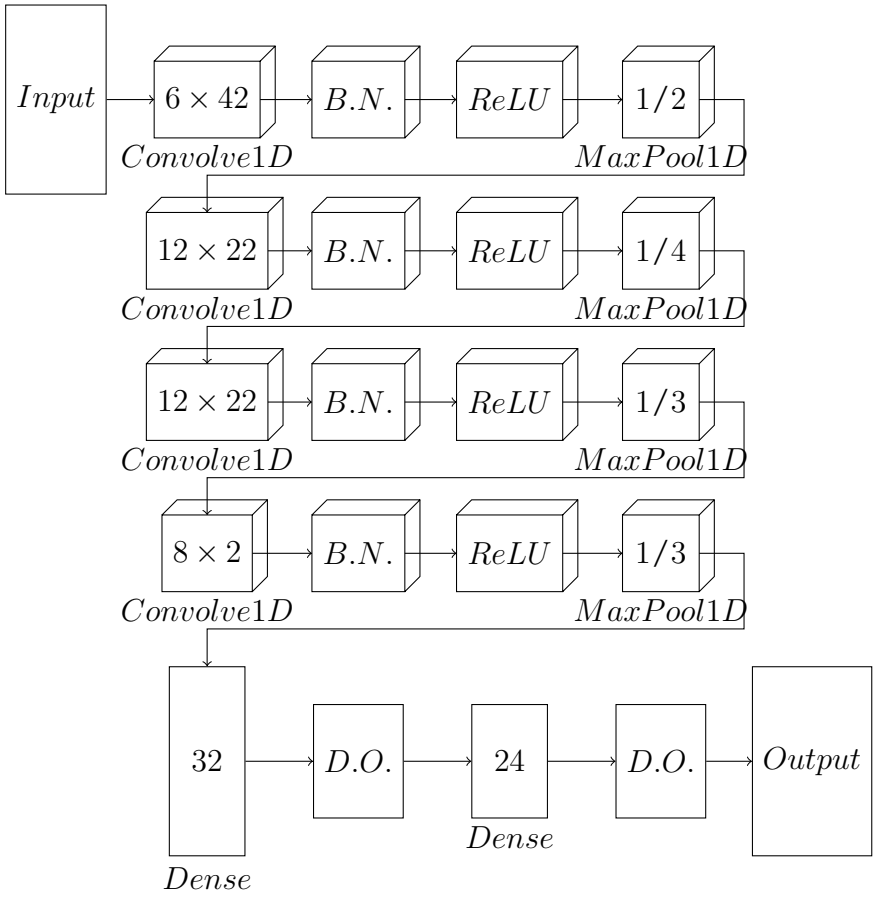


Figure 5.7: Graph demonstrating the classifiers's network. The $B.N.$ block denotes a batch normalization layer, and the $D.O.$ block a Dropout layer.

5.5 Results

5.5.1 3 class classification

The classifier achieved a total accuracy of 77%, and an average F1 score of 0.78, when classifying three classes. The confusion matrix is shown in 5.3 and 5.8, and the precision, recall and F1 score are shown in table 5.2.

Class	Precision	Recall	F1 score
Adrenaline	0.984	1.00	0.99
Baseline	0.641	0.77	0.70
Occlusion	0.777	0.64	0.70
MEAN	0.80	0.80	0.78

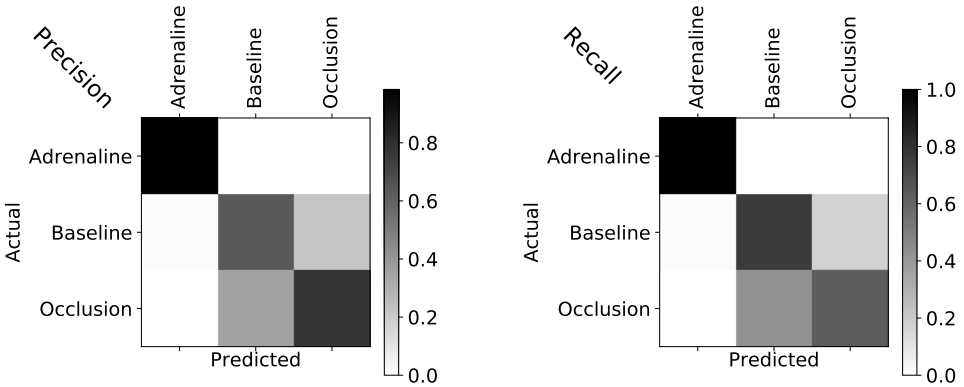
Table 5.2: The precision, recall and F1 score when classifying 3 classes.

Predicted →	Adrenaline	Baseline	Occlusion
Actual ↓			
Adrenaline	**1332**	0	0
Baseline	21	**1338**	380
Occlusion	1	748	**1323**

Total accuracy: 0.77

Table 5.3: Confusion matrix showing the predictions done by the classifier.

In figure 5.12 three context removal experiments are shown. Sub-figure 5.12a shows that the most important features for the classifier when classifying baseline are at approximately 230 samples. This is about the isovolumic relaxation period. When classifying occlusion, the classifier focuses on about the same period, as seen in sub-figure 5.12b. This indicates that the isovolumic relaxation period is an important feature when discriminating between the two classes. The adrenaline class did not lose any confidence when occluding the image, as seen in figure 5.9c.



(a) (b)

Figure 5.8: (a) Is the confusion matrix when classifying three classes. It is normalized with trespect to its rows/actual values, showing the precision over its diagonal. (b) As for a, but normalized in respect to its columns/predicted values, showing the recall over its diagonal. The values in the normalized confusion matrices (a) and (b) are plotted as gray colors.

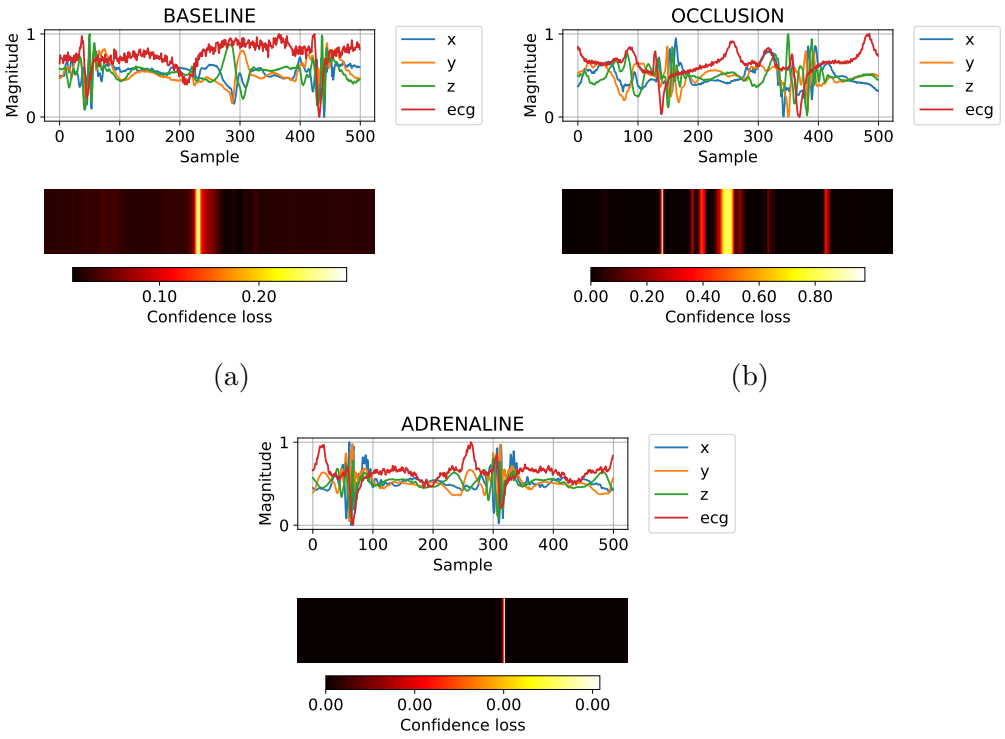
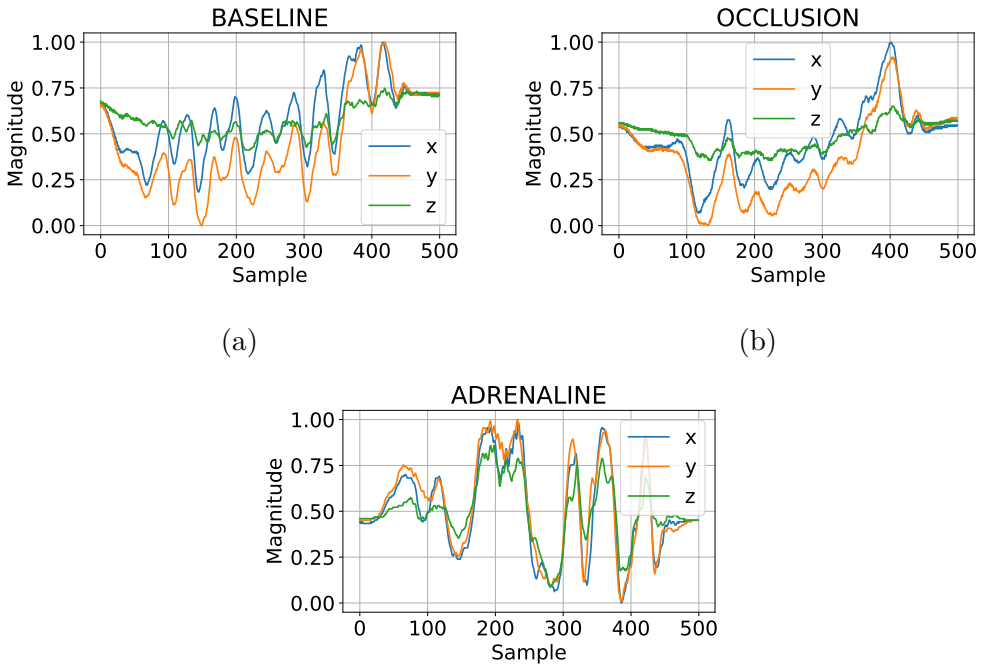


Figure 5.9: Figure demonstrating what features that are considered important to the classifier when discriminating between the classes. The top figure in (a) shows the magnitude of the ACC input tensor (x,y and z) in addition to the corresponding ECG signal. The figure below is an illustration of the confidence that is lost when removing the features in the above figure. (b) And (c) as of (a), but with different classes.

The maximum activation input tesor for each class is shown in figure 5.10. The constructed baseline tensor is similar to the occlusion tensor. However, the occlusions' class has a noticeable peak at the end (around 400 samples in figure 5.10b) in contrast to the baseline's. Take notice of that these constructed inputs do not look like ACC inputs as they are products of convolutions, and the model is trained on data not divided by features.



(a) (b)

(c)

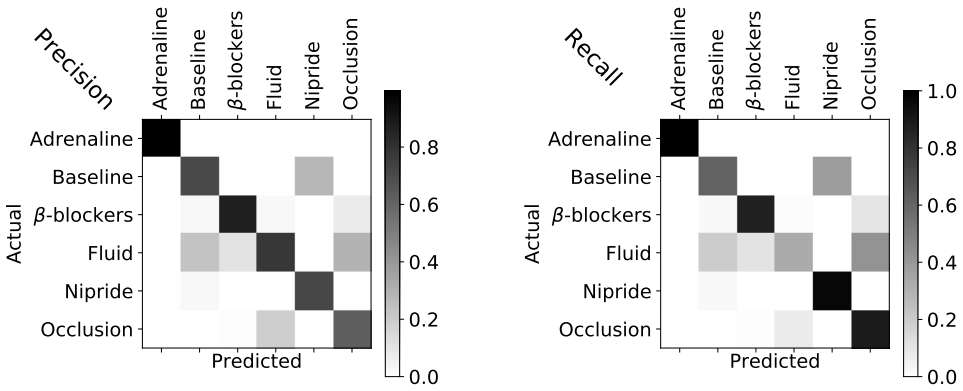
Figure 5.10: (a) Shows what input that maximizes the confidence for the class baseline. (b) and (c) As of (a), but with the different classes.

5.5.2 6 class classification

The classifier had an average accuracy of 77% and an average F1 score of 0.76 when classifying 6 classes. Figure 5.7 and 5.11 shows the confusion matrix for the classifier, where the diagonal line shows the number of correct classifications for each class. The fluid class is the class the classifier struggles the most to identify, with a recall of only 0.331.

Predicted →	Adrenaline	Baseline	β-blockers	Fuid	Nipride	Occlusion
Actual ↓						
Adrenaline	**740**	0	0	0	0	0
Baseline	0	**451**	4	0	285	0
β-blockers	0	19	**644**	9	0	68
Fluid	0	148	81	**245**	0	266
Nipride	3	18	0	0	**719**	0
Occlusion	0	0	8	60	0	**561**

Table 5.4: Confusion matrix of predictions done by the classifier.



(a) (b)

Figure 5.11: (a) Is the confusion matrix normalized with trespect to its rows/actual values, showing the precision over its diagonal. (b) as for a, but normalized in respect to its columns/predicted values, showing the recall over its diagonal. The values in the normalized confusion matrices (a) and (b) are plotted as gray colors.

Class	Precision	Recall	F1 score
Adrenaline	0.996	1.00	0.998
Baseline	0.709	0.609	0.656
β-blockers	0.874	0.870	0.872
Fluid	0.780	0.331	0.465
Nipride	0.716	0.972	0.825
Occlusion	0.627	0.892	0.736
MEAN	0.784	0.779	0.759

Table 5.5: The precision, recall and F1 score when classifying 6 classes

Figure 5.12 shows six context removal experiments. In sub-figure 5.12f it becomes clear that the classifier consider the same period as in the 3 class classification to be the most important (in this case around 100 samples). However, it does not seem to notice the same feature occurring at approximately 430 samples. This could be due to the fact that the isovolumic relaxation period must be seen in the context of another period.

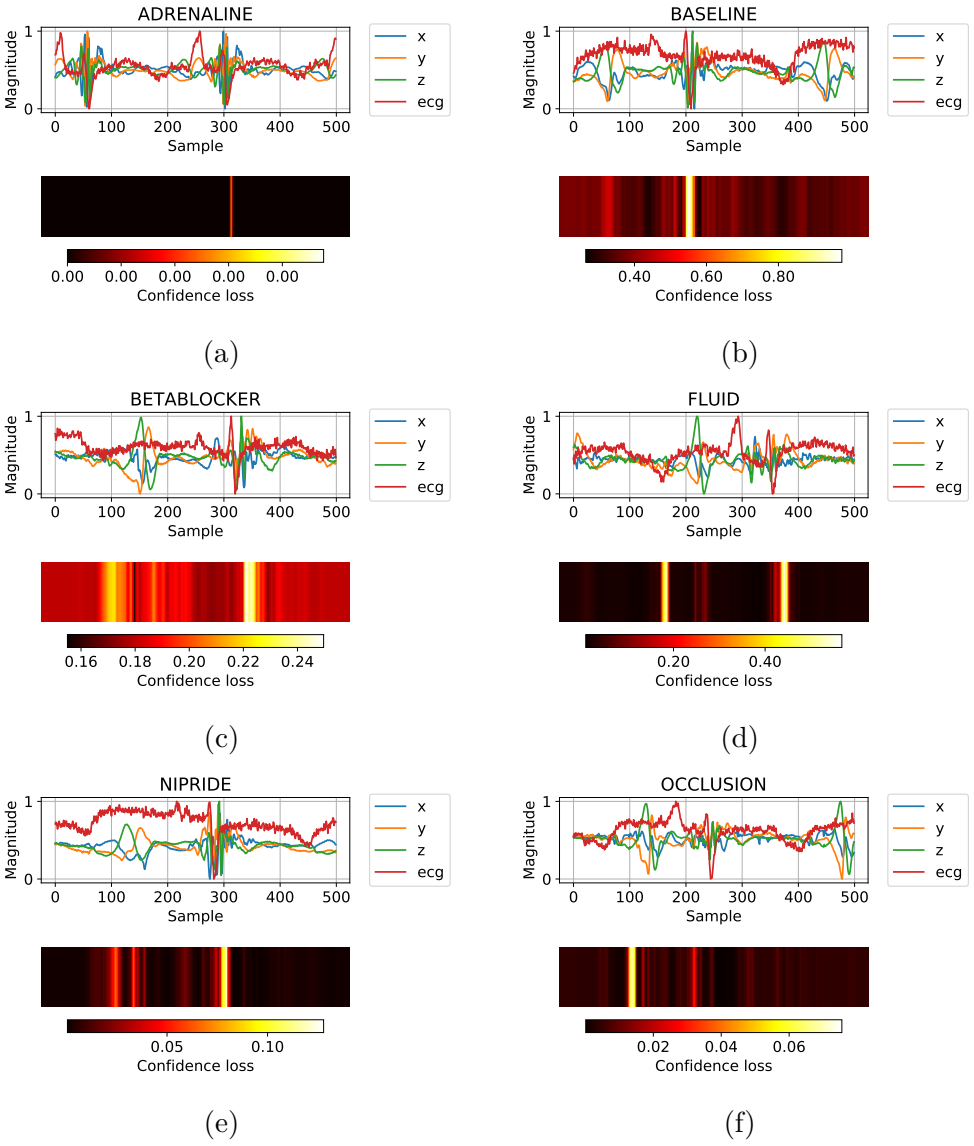


Figure 5.12: Figure demonstrating what features that are considered important to the classifier when discriminating between the classes. The top figure in (a) shows the magnitude of the ACC input tensor (x,y and z) in addition to the corresponding ECG signal. The figure below is an illustration of the confidence that is lost when removing the features in the above figure. (b) And (c) as of (a), but with different classes.

In figure 5.13 the inputs producing the maximum activation for the classes are shown. Here, the occlusion and baseline classes are not as similar as the three class maximum activation inputs seen in 5.10. The occlusion max activation input, seen in figure 5.13b, contains more information in the higher frequencies than the baseline maximum activation input.

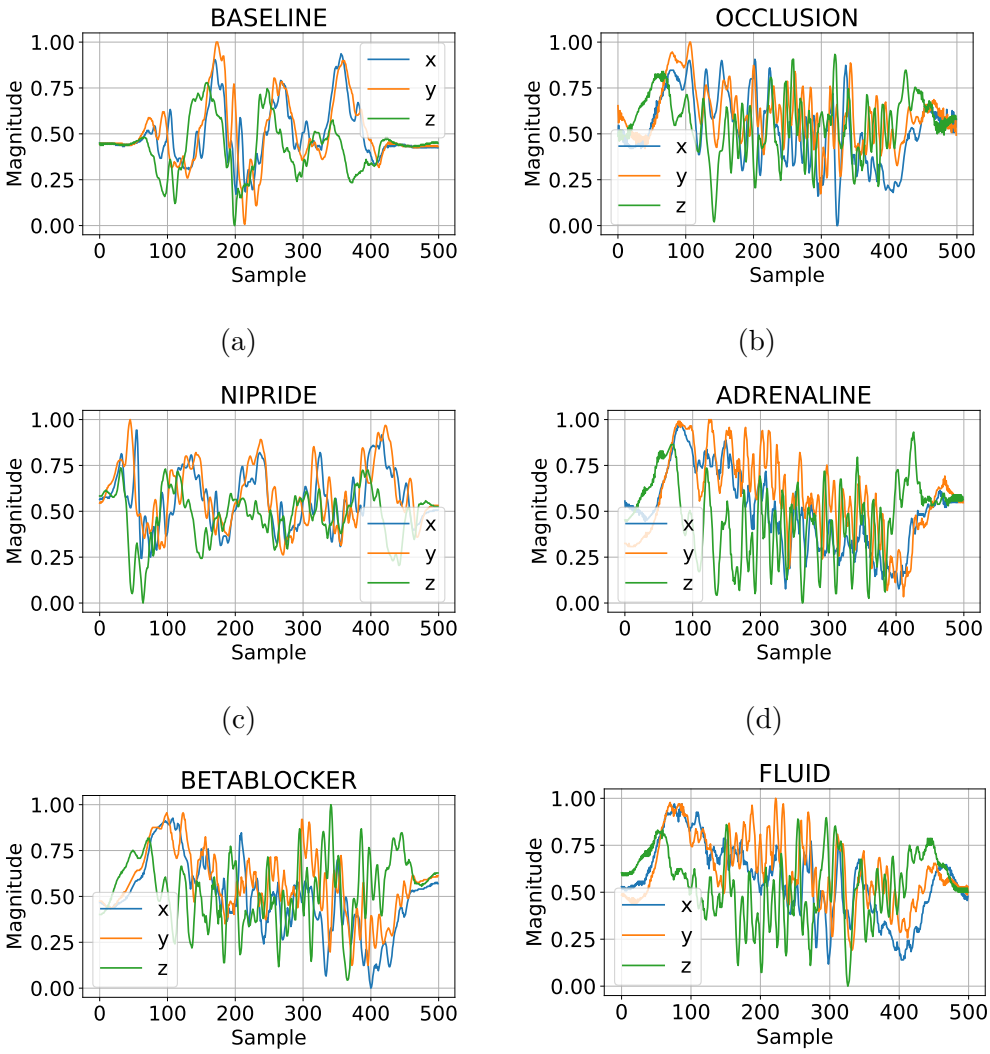


Figure 5.13: (a) Shows what input that maximizes the confidence for the class baseline. (b), (c), (d), (e) And (f) as of (a), but with the different classes.

Chapter 6

Experiment 3: Image classification

This chapter will introduce the final experiment. The accelerometer domain will be transferred into an image domain by using the continuous wavelet transform. The transformed accelerometer data will be used when training and testing a DNN. This is done to see if such transformation can improve the classification task described in experiment 2.

6.1 Idea

The idea behind the following experiment is to transform the ACC data into a higher dimensional space before classification. This is done to provide additional features to the classifier. This might make the classifier able to extract even higher level features, which again might lead to a better classification.

6.2 Preparations

Before training the classifier, all the ACC input tensors was transformed using the *SciPy*'s continuous wavelet transform (cwt) function. The wavelet used was the Ricker wavelet, with widths extending from 1 to 50. The Ricker wavelet amplitude at time t with peak frequency f is defined in equation 6.1.

$$A(f,t) = (1 - 2\pi^2 f^2 t^2)e^{-\pi^2 f^2 t^2} \tag{6.1}$$

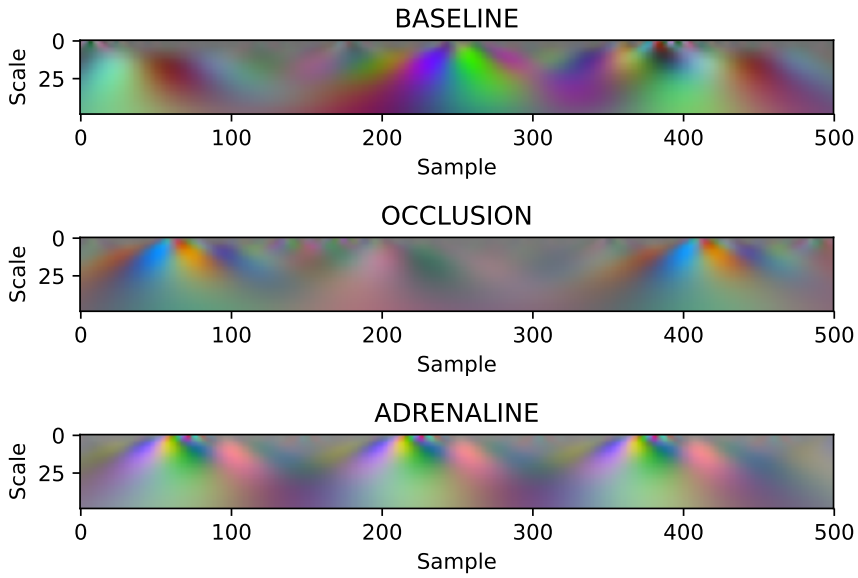


Figure 6.1: Examples of training images where the accelerometer data has been transformed using the continuous wavelet transform. Each dimension in the accelerometer data is responsible for its own RGB channel in the image.

Each dimension in the ACC data was transformed using the ctw, followed by a normalization. The final training images were produced by setting each dimension to its own image channel. The X-axis the R channel, the Y-axis the G channel, and the Z-axis the B channel, resulting in images as demonstrated in figure 6.1.

6.3 Implementation

In figure 6.2 the network used to classify the ACC cwt data is demonstrated. It has few kernels, and only three hidden layers. However it consists of multiple regularization layers to prevent overfitting to individual pigs. The *ADAM* optimizer was used, as well as softmax with cross entropy loss.

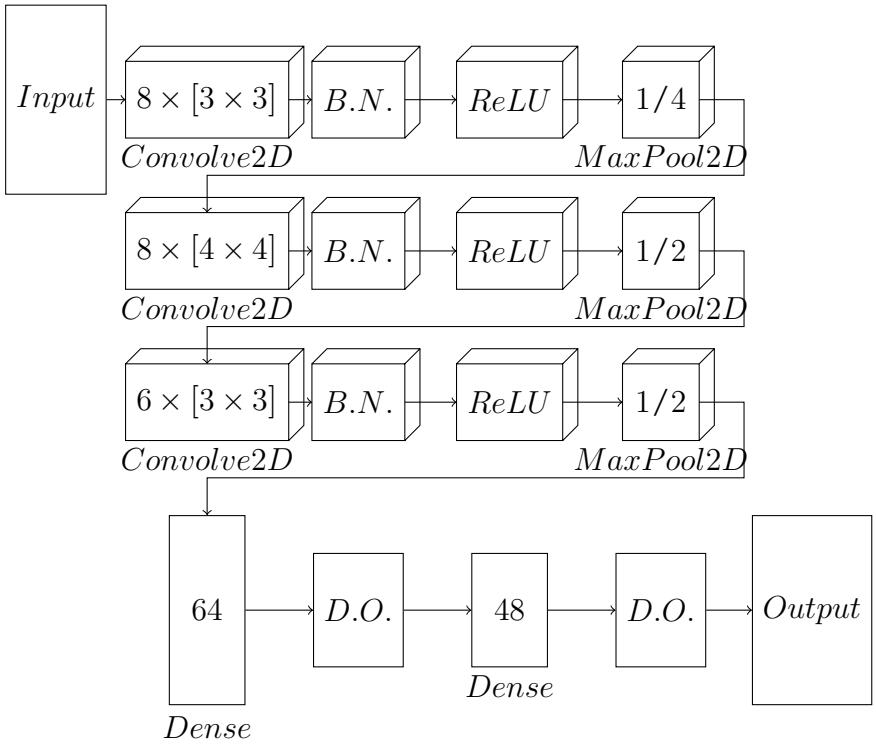


Figure 6.2: Graph demonstrating the image classifier network. The *B.N.* block denotes a batch normalization layer, and the *D.O.* block a Dropout layer. After each dense layer there is an a *ReLU* activation not shown in the figure.

6.4 Results

The classifier classifying the ACC cwt images achieved an average accuracy of 79%, with an F1 score of 0.80. The precision, recall and the F1 from the classifier are shown in table 6.1, and its confusion matrix in table 6.2 and 6.3.

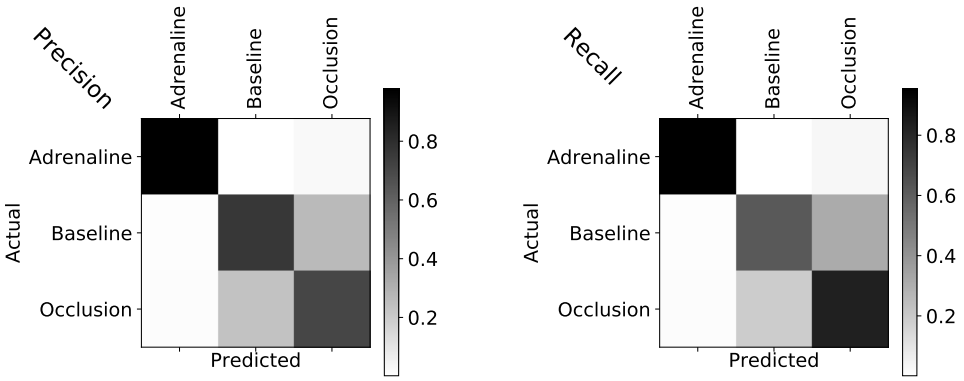
Class	Precision	Recall	F1 score
Adrenaline	0.795	0.950	0.962
Baseline	0.763	0.619	0.683
Occlusion	0.707	0.831	0.764
MEAN	0.815	0.799	0.803

Table 6.1: Precision, recall and F1 score when classifying 3 classes.

Predicted →	Adrenaline	Baseline	Occlusion
Actual ↓			
Adrenaline	**1266**	2	64
Baseline	15	**1076**	648
Occlusion	18	333	**1721**

Total accuracy: 0.79

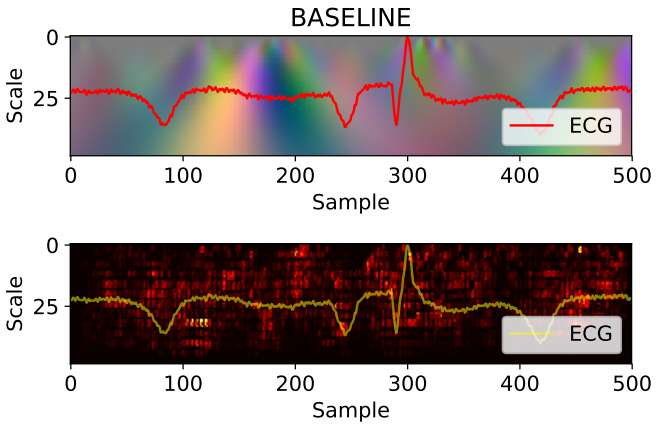
Table 6.2: Confusion matrix showing the predictions done by the classifier.



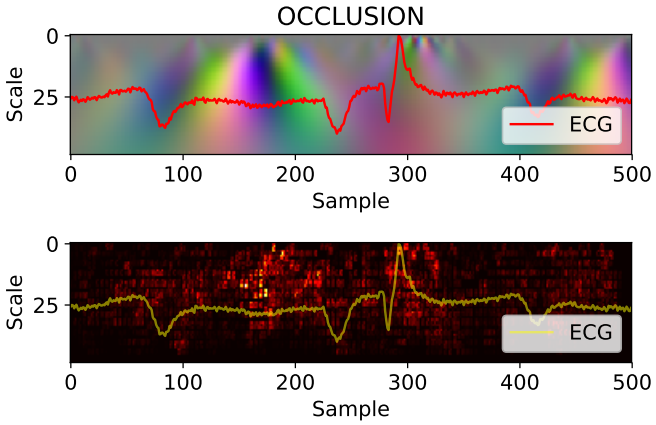
(a) (b)

Figure 6.3: (a) Is the confusion matrix of the image classifier classifying three classes. It is normalized with trespect to its rows/actual values, showing the precision over its diagonal. (b) as for a, but normalized in respect to its columns/predicted values, showing the recall over its diagonal. The values in the normalized confusion matrices (a) and (b) are plotted as gray colors.

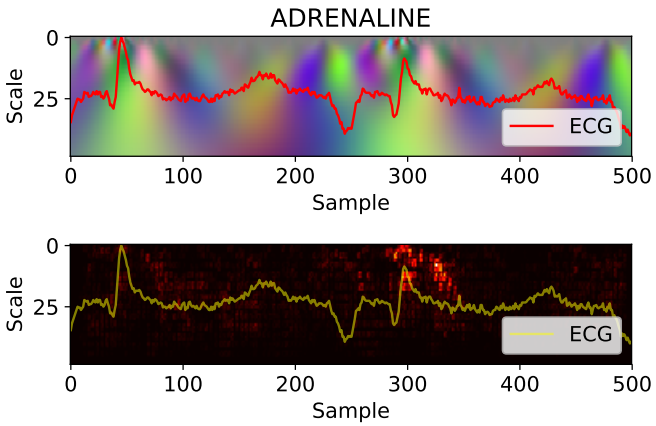
In figure 6.4 three context removal experiments are shown. In sub-figure 6.4c it becoms clear that the isovolumic relaxation phase is still the area of interest for a DL classifier. Also, the frequency range from 10 to 20 Hz is of interest for the classifier (seen as the scale range from 12.5 to 25). The same frequencies was discovered to change during occlusion by Elle et al. in 2005 [2].



(a)



(b)



(c)

Figure 6.4: (a) is a context removal experiment done on an ACC cwt input. The top image is the input image, as well as the corresponding ECG signal. The bottom image is a heat map of pixels important for the classifier for discriminating between baseline and the other classes. The brighter the color, the more important the pixle.

Chapter 7

Discussion

This chapter will consist of a comparison between this thesis' results and previous research done on the same topic. This will be followed by a general discussion of the limitations and the findings of the experiments.

7.1 Previous results

Stumpf did a master's thesis on this topic, exploring the possibility of using AdaBoost (ensemble learning), decision trees and support vector machines to classify cardiac dysfunctions using a 3-axis accelerometer [51]. Stumpf trained the classifiers to discriminate between at most five classes: baseline, adrenaline, esmolol, ischemia and fluid loading, and achieved 54% average prediction accuracy. By training a binary classifier, baseline versus ischemia, Stumpf managed to get a precision and recall of 87% and 94% and a total accuracy of 90%. The results from this thesis hvae not reached similar numbers, however, no binary classifier was trained. Stumpf also created a three class classifier, discriminating between baseline, adrenaline and ischemia and reached an average accuracy of 74%. The classifier introduced in section 5.4.3 managed to discriminate between the same three classes with 77% accuracy, with an average F1 score of 0.78. When predicting six classes, the model managed to discriminate between the same classes with the same accuracy, but with a lower F1 score of 0.75.

The main difference between the work done by Stumpf and the work done in this thesis is that Stumpf did feature extraction prior to classification. In addition, when using raw data, the data was divided by features (R-peak features). No such feature extraction or feature division is done in this thesis. Also, Stumpf did cross validation, validating over the entire dataset. No such validation is done in this thesis, as several pigs were separated prior to training the classifiers. These were used as test data.

7.2 General discussion

Experiment 1

The regressors created in experiment 1 performed surprisingly well, with several features being present in the predictions. Features were visible when transforming between the accelerometer data and the ECG data, and this demonstrated that such regressors should be further investigated. The metrics used showed that most of the regressors did better than noise. This indicates that the regressor models used in this thesis are able to extract features in both the accelerometer domain, and the ECG domain. Most surprisingly, the regressor transforming from the ECG data to the accelerometer data managed to reproduce noticeable features, despite the fact that the metrics did not reflect it. Especially, the isovolumic contraction and relaxation in the accelerometer signal made by the C.1 regressor was visible in the predictions.

No filtering of the ECG signal prior to the training was done. Some of

the ECG tensors contained a lot of noise, especially in the high frequencies. A low pass filtering of the signal would probably have increased the quality of the regressors' performance, and this should be done if these techniques are investigated further.

Experiment 2

During the construction of the network, several combinations of deep learning building blocks were made. These are not discussed in the thesis. However, balancing between a complex network that has the ability to extract the necessary high-level features, and a network that does not over-fit to the individual pigs after a few epochs, was a difficult task. However, the networks constructed coped with that balance in a decent manner. The three class classifier achieved a F1 score of 0.78. This is not satisfactory. However, there exists an infinite number of combinations for constructing DNNs, and further development, parameter tuning etc., would be required to find the one that performs the best. The six class classifier achieved a F1 score of 0.75. For a six class classifier this is pretty high, but in a medical setting, it is not satisfactory.

The model trained to classify 6 classes achieved a higher recall when predicting occlusion than the model trained to classify 3 classes. This could be due to the difference in quality between the two data sets used. The network might have learned to discriminate between the data sets, and it may therefore have been left with too few parameters to do the final classification. The max activation for the occlusion input from the six class classifier, seen in figure 5.13, contains more information in the higher frequencies than the baseline maximum activation input. In the three class classifier's maximum inputs, the baseline and occlusion max activation inputs are more similar, as seen in figure 5.13. This indicates that the six class classifier managed to extract more features than the three class classifier, and could explain why the six class classifier has a higher recall than the three class classifier.

Experiment 3

When transforming the ACC data using the continuous wavelet function, the three class classifier improved slightly. The recall of occlusion was improved from 0.64 to 0.83. It achieved a F1 score of 0.80, slightly outperforming the previous classifier that had a F1 score of 0.78. The most interesting results achieved in this experiment, was that the frequencies used by the classifier became visible when the context removal experiments were conducted. The experiment revealed that the classifier focused on the frequencies between 10 and 20 Hz during the isovolumic relaxation phase

when classifying occlusion. In addition, the context removal experiment demonstrated that the classifier was most interested in the features occuring during the isovolumic contraction when classifying the adrenaline class.

7.3 Limitations

Prior to the training the classifiers, random pigs were selected for the test process. Ideally, k-cross validation should be used for testing the models. However, given the fact that the networks' training time was long, k-cross validation has not been done. The downside of not using k-cross validation is that the pigs used for test data might have been simple subjects, and therefore no proofs of general models are provided. K-cross validation of the results would be preferred and should be done in future work. However, the achieved results do provide an indicator of the capability of such models - if the test pigs selected for testing are a good representation of the population, results from this thesis are promising.

Chapter 8

Conclusion

This chapter concludes this master's thesis. Firstly, a summary of the experiments will be given, followed by a general discussion. Lastly, further work is suggested.

8.0.1 Summary of experiments

Experiment 1

Experiment 1 demonstrated that making regressors able to convert between the accelerometer and the ECG domain was difficult. However, the regressors managed to recover some of the features in both domains. The regressors converting from the accelerometer domain to the ECG domain managed to reproduce most of the R-peaks and some T-segments. The regressors converting from the ECG domain to the accelerometer domain managed to reproduce some of the movement during isovolumic contraction and relaxation. The most satisfying regressor with respect to the absolute Pearson Correlation Coefficient metric was the C.1 regressor. The C.1 regressor consisted of convolutional layers, indicating that convolutional layers should be used for experiment two and three. Also, the regressors demonstrated impressive results when predicting accelerometer signals given ECG as input. However, neither of the regressors performed at a satisfactory level.

Experiment 2

Experiment two showed that convolutional neural networks are excellent tools for classifying cardiac heart dysfunctions, achieving an accuracy of 77% when classifying six cardiac functions. The experiment also demonstrated how a deep network can be analyzed to extract domain knowledge, as context removal experiments elucidated some important features in the accelerometer signal. Some of these features have previously been pointed out as important by researchers at the Intervention Centre at the Oslo University Hospital.

Experiment 3

Experiment 3 demonstrated that accelerometer data transformed using the continuous wavelet transform can be used as input to the classifier classifying cardiac heart dysfunctions. However, when using the transformed accelerometer domain, the classifier did not outperform the classifiers from experiment two with a huge factor. However, domain knowledge was gained, as the frequencies considered important by the classifier were highlighted.

8.1 Conclusion and further work

This thesis has investigated whether deep learning architectures can be used to classify cardiac heart dysfunctions by using data from a 3-axis accelerometer attached to the heart. The experiments conducted demonstrated that deep learning architectures are excellent tools for classifying cardiac heart dysfunctions using motion data from a 3-axis accelerometer sensor attached to the heart. The intended goal was to create a classifier that could discriminate between different heart conditions, and to explore which deep learning architectures that performed best. The best performing models in this thesis were the networks using convolutional layers, discriminating between six classes with an accuracy of 77% and a F1 score of 0.75. The optimal deep neural network architecture for classifying cardiac dysfunctions is still to be found, but this work indicates that deep neural networks can be applied to solve such classification tasks. The findings in this master's thesis can be further investigated in collaboration with clinicians.

8.1.1 Further work

To avoid cracking nuts with a sledge hammer, the networks utilized in this thesis have been simple networks using dense and convolutional layers. In a deep learning context they are quite traditional. It would be interesting to utilize other techniques, such as recurrent neural networks and more complex models, to see if this could further improve the performance. For example, it could be interesting to train an autoencoder on accelerometer data and use the z-space[1] learned by the autoencoder to train a recurrent neural network to predict "the future"[53]. For example, one could feed four time steps to the recurrent neural network, and then ask the recurrent neural network to predict the next time step. If the recurrent neural network successfully predicts the future, one could feed the predicted future to a classifier discriminating between heart conditions. Suddenly, one can "model the world"[54] and predict the state of the future.

On the other hand, the bigger and more complex the model is, the more data is required. Another natural follow up to this thesis would be to tune the networks used in this thesis, by adjusting the network size, filter size, learning rate, connections, and other similar hyper parameters, to find an optimal combination of deep learning building blocks and parameters for

[1]The motivation for transforming to the z-space is that a recurrent neural network, for example a LSTM network [52], has few trainable parametres. By compressing the data to a smaller space, the RNN network hopefully doesn't have to handle noise etc., as discussed in section 2.6.1.

classifying cardiac dysfunctions. Such tuning is a search problem that can be solved by, for example, utilizing evolutionary algorithms. By evolving neural networks (neuroevolution), interesting constructions specified to the task at hand can be discovered. For example, by using existing neuroevolution techniques, Miikkulainen et al. managed to evolve a recurrent neural network to create a "double-skip" connection between recurrent layers (LSTM) [55], resulting in a 5% improvement over the vanilla LSTM.

Bibliography

[1] I. Ariansem, R. Selmer, S. Graff-Iversen, G. M. Egeland, and S. Sakshaug. *Folkehelserapporten, Hjerte- og karsykdommer i Norge*. 2014. URL: `https://www.fhi.no/nettpub/hin/ikke-smittsomme/Hjerte-kar/`.

[2] O. J. Elle, S. Halvorsen, M. G. Gulbrandsen, L. Aurdal, A. Bakken, E. Stamset, H. Dugstad, and E. Fosse. "Early recognition of regional cardiac ischemia using a 3-axis accelerometer sensor". In: *Physiological Measurement* 26 (2005), pp. 429–440.

[3] P. S. Halvorsen, L. Fleischer, A Espinoza, O. Elle, L Hoff, H Skulstad, T Edvardsen, and E Fosse. "Detection of myocardial ischaemia by epicardial accelerometers in the pig". In: *British journal of anaesthesia* 102.1 (2008), pp. 29–37.

[4] P. S. Halvorsen, E. W. Remme, A. Espinoza, H. Skulstad, R. Lundblad, J. Bergsland, L. Hoff, K. Imenes, T. Edvardsen, O. J. Elle, and E. Fosse. "Automatic real-time detection of myocardial ischemia by epicardial accelerometer". In: *The Journal of Thoracic and Cardiovascular Surgery* 139.4 (2010), pp. 1026–1032.

[5] P. S. Halvorsen, A. Espinoza, L. A. Felischer, O. J. Elle, L. Hoff, R. Lundblad, H. Skulstad, T. Edvardsen, H. Ihlen, and E. Fosse. "Feasibility of a three-axis epicardial accelerometer in detecting myocardial ischemia in cardiac surgical patients". In: *The Journal of Thoracic and Cardiovascular Surgery* 136.6 (2008), pp. 1496–1502.

[6] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[7] V. Badrinarayanan, A. Kendall, and R. Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 12 (2017), pp. 2481–2495.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[9] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[10] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. "Aggregated residual transformations for deep neural networks". In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE. 2017, pp. 5987–5995.

[11] S. Ren, K. He, R. Girshick, and J. Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[12] Wikipedia. *Heart*. 2019. URL: https://en.wikipedia.org/wiki/Heart.

[13] B. staff. *Medical gallery of Blausen Medical 2014, WikiJournal of Medicine 1 (2). DOI:10.15347/wjm/2014.010. ISSN 2002-4436*. 2014. URL: https://commons.wikimedia.org/w/index.php?curid=30111372.

[14] I. D.v. O. Blausen Medical Communications. *Superficial Heart Anatomy (Anterior View)*. 2013. URL: https://commons.wikimedia.org/w/index.php?curid=26986380.

[15] H. Arnesen. *Hjertet*. 2016. URL: https://sml.snl.no/hjerte.

[16] Wikipedia. *Wiggers Diagram 2*. 2016. URL: https://commons.wikimedia.org/wiki/File:Wiggers_Diagram_2.svg.

[17] R. E. Klabunde. *Cardiac Preload*. 2015. URL: http://www.cvphysiology.com/CardiacFunction/CF007.

[18] Wikipedia. *Beta blocker*. 2019. URL: https://en.wikipedia.org/wiki/Beta_blocker.

[19] Harald Arnesen. *iskemi*. 2018. URL: https://sml.snl.no/iskemi.

[20] G. D. Baura. *Medical Device Technologies: A Systems Based Overview Using Engineering Standards*. Academic Press, 2011.

[21] H. Arnesen. *Elektrokardiografi*. 2016. URL: https://sml.snl.no/elektrokardiografi.

[22] P.-C. Chang, J.-J. Lin, J.-C. Hsieh, and J. Weng. "Myocardial infarction classification with multi-lead ECG using hidden Markov models and Gaussian mixture models". In: *Applied Soft Computing* 12.10 (2012), pp. 3165–3175.

[23] R Silipo, A Taddei, and C Marchesi. "Continuous monitoring and detection of ST-T changes in ischemic patients". In: *Computers in Cardiology 1994*. IEEE. 1994, pp. 225–228.

[24] A. Gharaviri, M. Teshnehlab, and H. Moghaddam. “Ischemia detection via ECG using ANFIS”. In: *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE. 2008, pp. 1163–1166.

[25] Y. Goletsis, C. Papaloukas, D. I. Fotiadis, A. Likas, and L. K. Michalis. “Automated ischemic beat classification using genetic algorithms and multicriteria decision analysis”. In: *IEEE transactions on Biomedical Engineering* 51.10 (2004), pp. 1717–1725.

[26] Y. Fakhri, M. Sejersten, M. M. Schoos, J. Melgaard, C. Graff, G. S. Wagner, P. Clemmensen, and J. Kastrup. “Algorithm for the automatic computation of the modified Anderson–Wilkins acuteness score of ischemia from the pre-hospital ECG in ST-segment elevation myocardial infarction”. In: *Journal of electrocardiology* 50.1 (2017), pp. 97–101.

[27] G. Crescenzi, T. Bove, F. Pappalardo, A. M. Scandroglio, G. Landoni, G. Aletti, A. Zangrillo, and O. Alfieri. “Clinical significance of a new Q wave after cardiac surgery”. In: *European journal of cardio-thoracic surgery* 25.6 (2004), pp. 1001–1005.

[28] T. Edvardsen, H. Skulstad, S. Aakhus, S. Urheim, and H. Ihlen. “Regional myocardial systolic function during acute myocardial ischemia assessed by strain Doppler echocardiography”. In: *Journal of the American College of Cardiology* 37.3 (2001), pp. 726–730.

[29] E. W. Remme, L. Hoff, P. S. Halvorsen, E. Nærum, H. Skulstad, L. A. Flaischer, O. J. Elle, and E. Fosse. “Validation of cardiac accelerometer sensor measurements”. In: *Physiological Measurement* 30 (2009), pp. 1429–1444.

[30] S. Hyler, A. Espinoza, H. Skulstad, E. Fosse, and P. S. Halvorsen. “Left ventricular function can be continuously monitored with an epicardially attached accelerometer sensor”. In: *European Journal of Cardio-Thoracic Surgery* 46.2 (2014), pp. 313–320.

[31] F. Tjulkins, A. T. T. Nguyen, E. Andreassen, N. Hoivik, K. Aasmundtveit, L. Hoff, O. J. Grymyr, P. S. Halvorsen, and K. Imenes. “MEMS-based implantable heart monitoring system with integrated pacing function”. In: *Electronic Components and Technology Conference (ECTC), 2014 IEEE 64th*. IEEE. 2014, pp. 139–144.

[32] O.-J.H. N. Grymyr, E. W. Remme, A. Espinoza, H. Skulstad, O. J. Elle, E. Fosse, and P. S. Halvorsen. “Assesment of 3D motion increases the applicability of accelerometers for monitoring left ventricular function”. In: *Interactive CardioVascular and Thoracic Surgery Advance Access* (2014), pp. 1–9.

[33] O.-J.H. N. Grymyr, A.-T. T. Nguyen, F. Tjulkins, A. Espionaza, E. W. Remme, H. Skulstad, E. Fosse, K. Imenes, and P. S. Halvorsen. "Continuous monitoring of cardiac function by 3-dimensional accelerometers in a closed-chest pig model". In: *Interactive CardioVascular and Thoracic Surgery Advance Access* (2015), pp. 1–10.

[34] M. R. Krogh, G. M. Nghiem, P. S. Halvorsen, O. J. Elle, O.-J. Grymyr, L. Hoff, and E. W. Remme. "Gravity Compensation Method for Combined Accelerometer and Gyro Sensors Used in Cardiac Motion Measurements". In: *Annals of biomedical engineering* 45.5 (2017), pp. 1292–1304.

[35] E. W. Remme, L. Hoff, P. S. Halvorsen, A. Opdahl, E. Fosse, and O. J. Elle. "Simulation model of cardiac three dimensional accelerometer measurements". In: *Medical engineering & physics* 34.7 (2012), pp. 990–998.

[36] B. Nusantrara. *Convolutional Neural Network*. 2017. URL: http://socs.binus.ac.id/2017/02/27/convolutional-neural-network/.

[37] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[38] B. Xu, N. Wang, T. Chen, and M. Li. "Empirical evaluation of rectified activations in convolutional network". In: *arXiv preprint arXiv:1505.00853* (2015).

[39] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: (2015), pp. 448–456.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[41] Z. C. Lipton. "A Critical Review of Recurrent Neural Networks for Sequence Learning". In: *CoRR* abs/1506.0 (2015). URL: http://arxiv.org/abs/1506.00019.

[42] M. Längkvist, L. Karlsson, and A. Loutfi. "A review of unsupervised feature learning and deep learning for time-series modeling". In: *Pattern Recognition Letters* 42 (June 2014), pp. 11–24. ISSN: 0167-8655. URL: https://www.sciencedirect.com/science/article/pii/S0167865514000221#t0005.

[43] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 0028-0836. URL: http://www.nature.com/articles/nature14539.

[44] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[45] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research* 11.Dec (2010), pp. 3371–3408.

[46] Pearson correlation coefficient. *Pearson correlation coefficient — Wikipedia, The Free Encyclopedia*. 2019. URL: `https://en.wikipedia.org/wiki/Pearson_correlation_coefficient`.

[47] J. Lee Rodgers and W. A. Nicewander. "Thirteen ways to look at the correlation coefficient". In: *The American Statistician* 42.1 (1988), pp. 59–66.

[48] J. S. Dai. "Euler–Rodrigues formula variations, quaternion conjugation and intrinsic connections". In: *Mechanism and Machine Theory* 92 (2015), pp. 144–152.

[49] M. D. Zeiler and R. Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[50] R. Kotikalapudi and contributors. *keras-vis*. `https://github.com/raghakot/keras-vis`. 2017.

[51] O. K. Stumpf. "Using motion data to detect cardiac dysfunctions". PhD thesis. University of Oslo, Oslo University Hospital - The Intervention Centre, 2016.

[52] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[53] A. Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[54] D. Ha and J. Schmidhuber. "World Models". In: *arXiv preprint arXiv:1803.10122* (2018).

[55] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, et al. "Evolving deep neural networks". In: *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 2019, pp. 293–312.

Appendices

Appendix A

Examples of predictions done by regressors

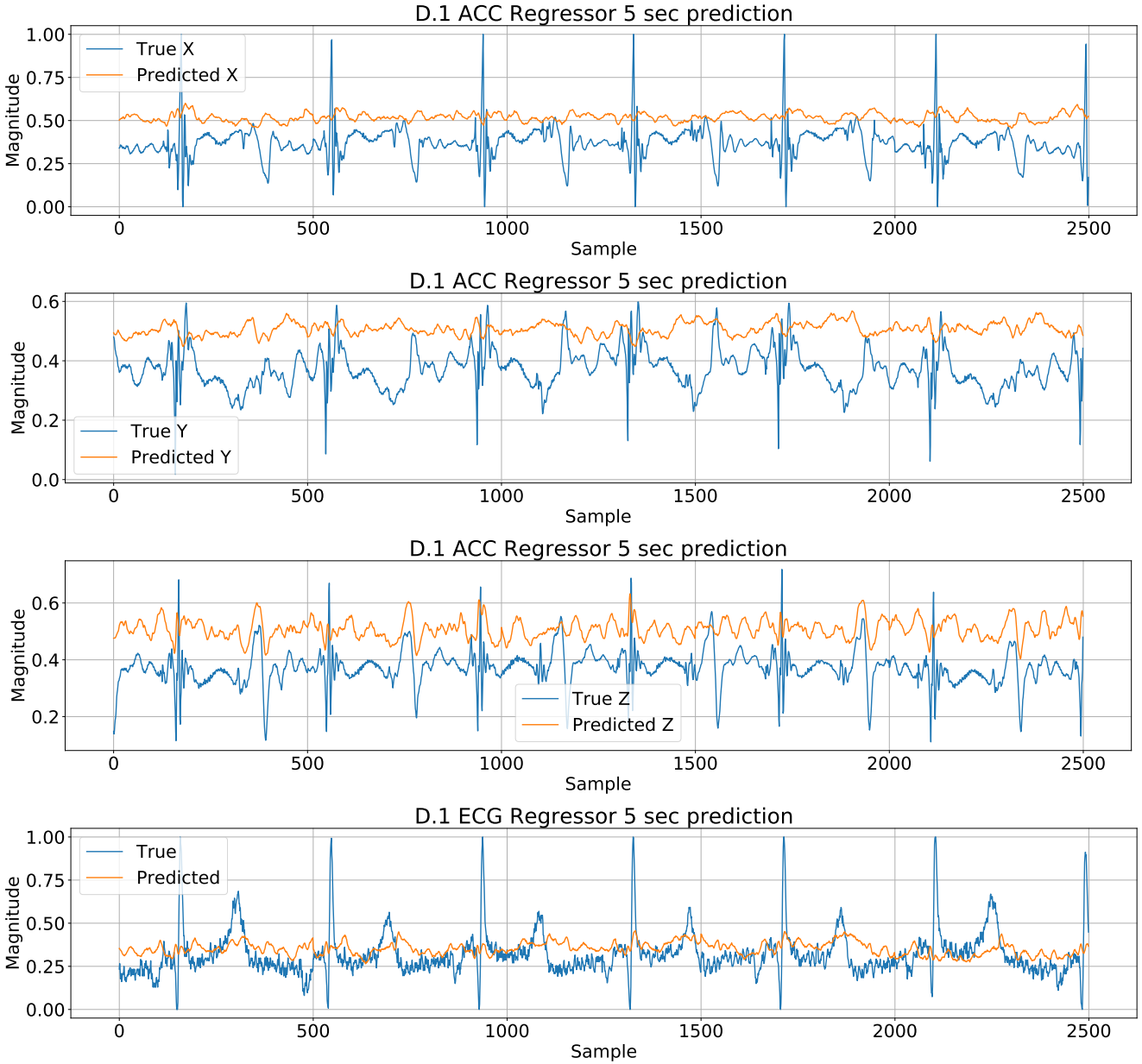


Figure A.1: 5 secounds of predictions done by the D.1 regressor

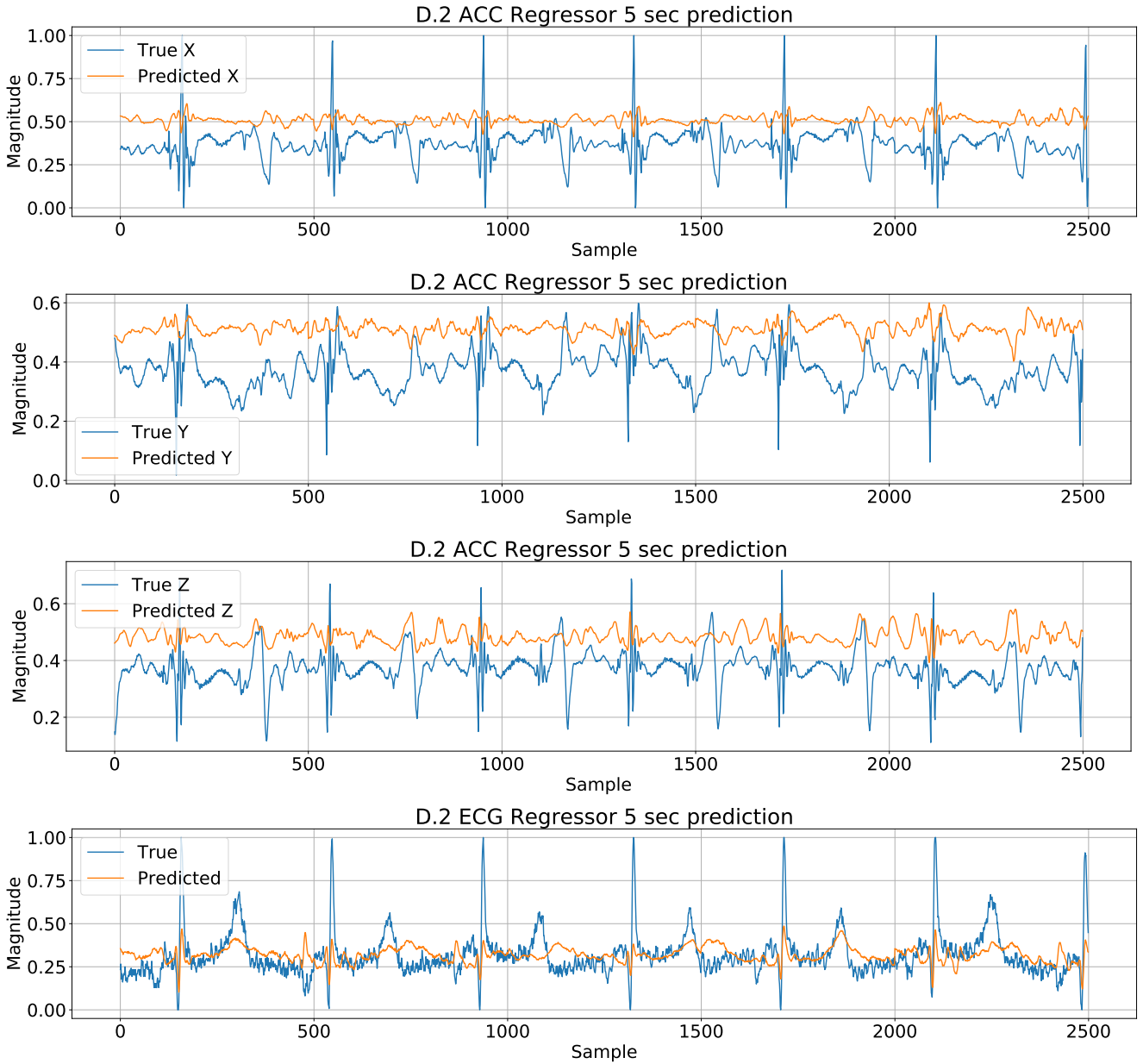


Figure A.2: 5 secounds of predictions done by the D.2 regressor

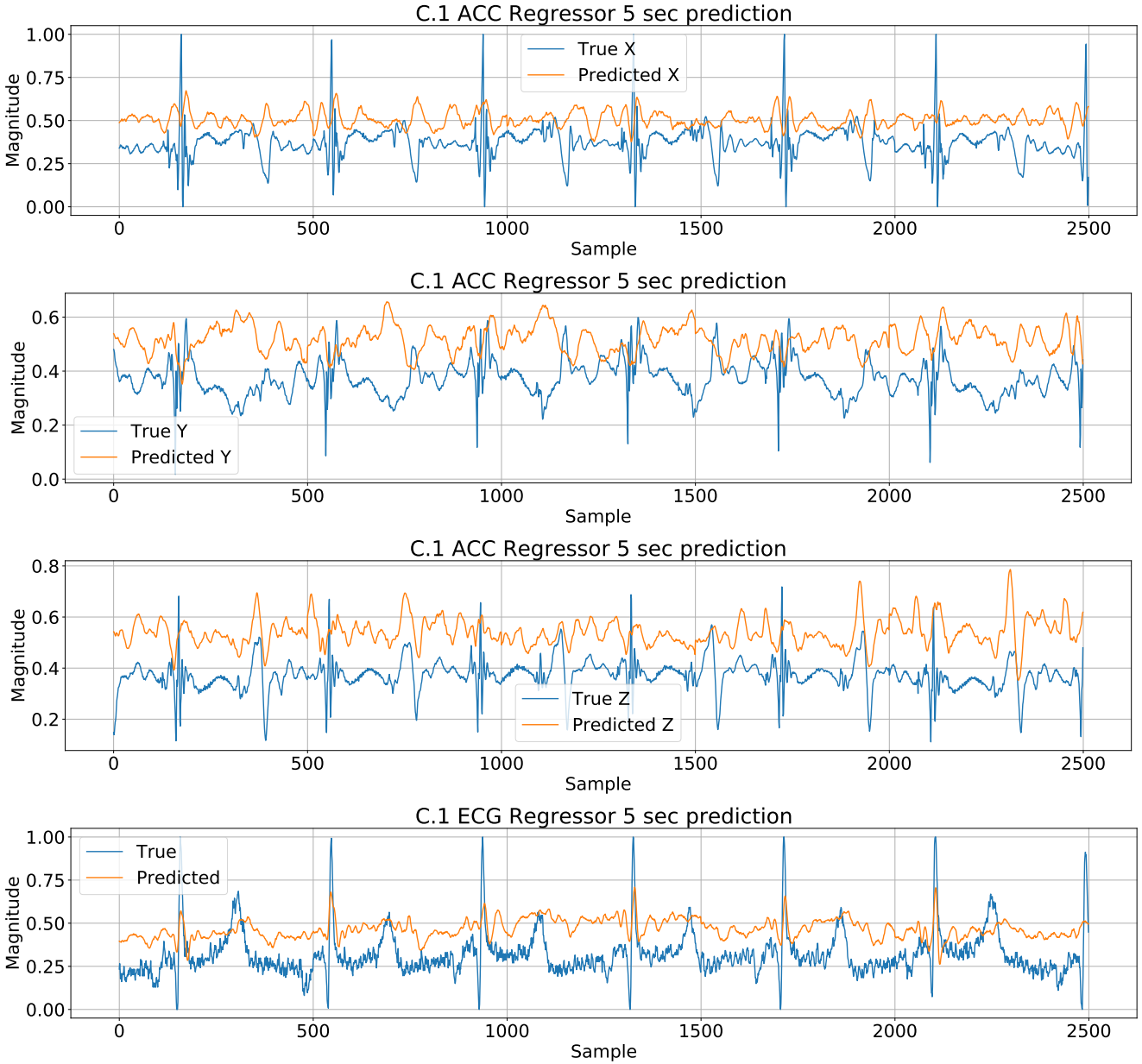


Figure A.3: 5 secounds of predictions done by the C.1 regressor

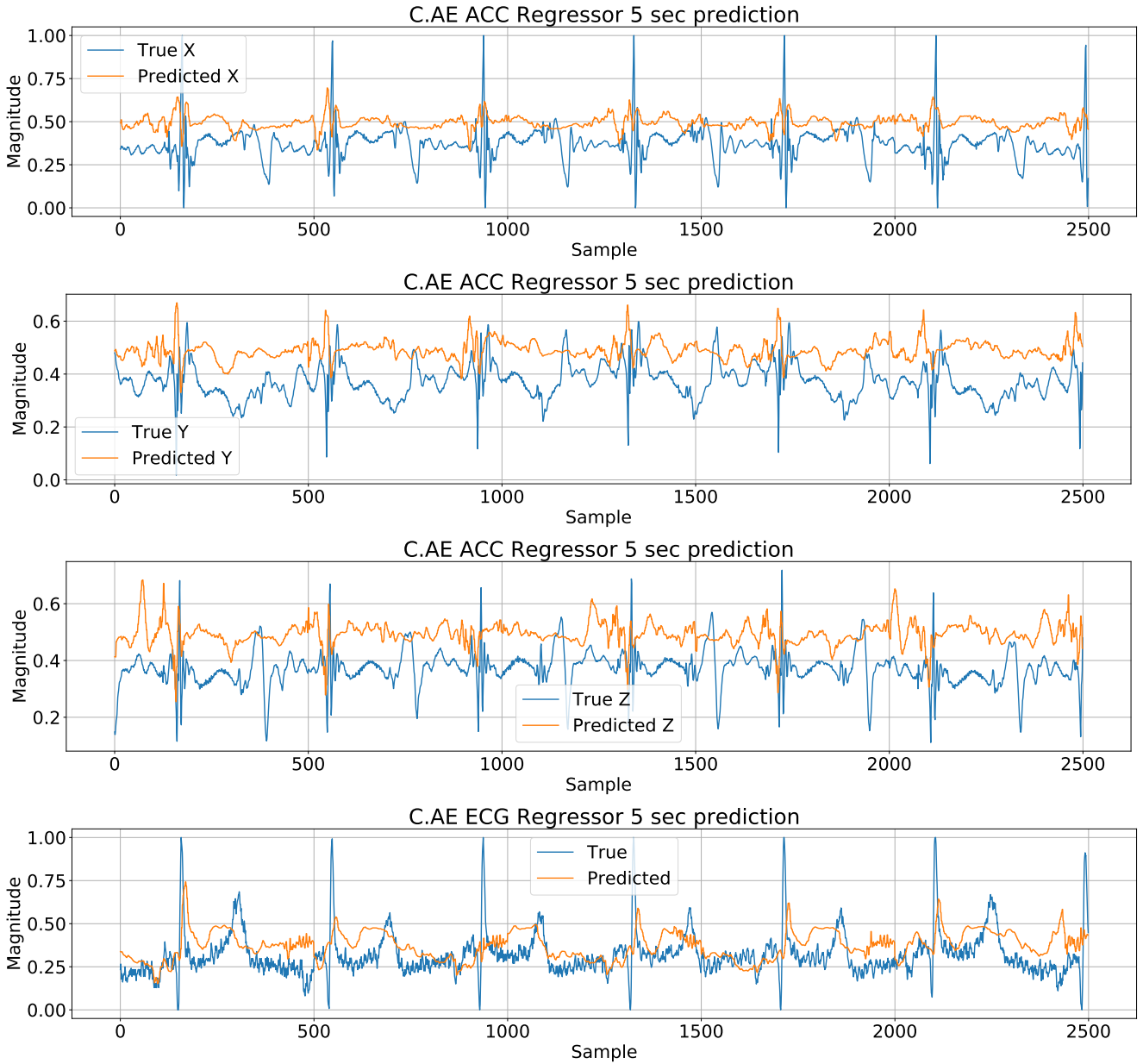


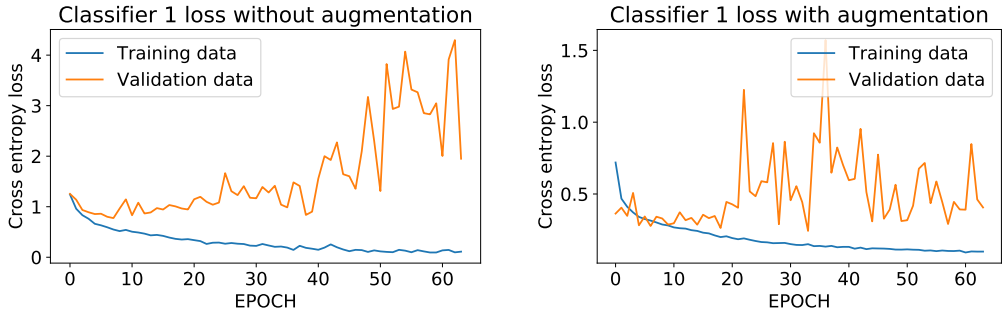
Figure A.4: 5 secounds of predictions done by the C.AE regressor

Appendix B

Augmented data review

The classifier network, described in 5.4.3, was used for this review, tasked to classify 3 classes. Two models were trained. The first one (**AUG 1**) was trained on data not augmented, and the second one (**AUG 2**) was trained on the same data, but augmented as described in 5.2.1. Bot trained models were evaluated using the same *not augmented* test data.

In figure B.1 the loss for both models is shown. Both models over-fit quickly, however, the **AUG 1** model over-fits after about 9 epochs, whereas the **AUG 2** model over-fits after approximately 19 epochs. Furthermore, the average loss for the validation data for **AUG 2** is lower than for the **AUG 1** model. Tables B.1, B.2, B.3, B.3, and figures B.1 and B.2, clearly demonstrate that using augmented data outperforms using data not augmented. Take notice of how the pig selected for test data for this review is unbalanced, as there exist more examples of the occlusion class than of the other classes.



(a) (b)

Figure B.1: (a) shows the loss during training for the classifier trained data not augmented. (b) As for (a), but shows the model trained on augmented data.

Performance without augmentation

Class	Precision	Recall	F1 score
Adrenaline	0.471	0.615	0.533
Baseline	0.275	0.440	0.338
Occlusion	0.824	0.568	0.627
MEAN	0.523	0.540	0.515

Table B.1: Precision, recall and F1 score from the *Classifier 1*'s performance **not** trained on augmented data.

Performance with augmentation

Class	Precision	Recall	F1 score
Adrenaline	0.963	1.000	0.981
Baseline	0.850	0.680	0.756
Occlusion	0.910	0.959	0.934
MEAN	0.908	0.878	0.890

Table B.2: Precision, recall and F1 score from the *Classifier 1*'s performance trained on augmented data.

Performance without augmentation

Predicted →	Adrenaline	Baseline	Occlusion
Actual ↓			
Adrenaline	**16**	4	6
Baseline	11	**11**	3
Occlusion	7	25	**42**

Total accuracy: 0.552

Table B.3: Confusion matrix showing the predictions done by the classifier trained on the data not augmented.

Performance with augmentation

Predicted →	Adrenaline	Baseline	Occlusion
Actual ↓			
Adrenaline	**26**	0	0
Baseline	1	**17**	7
Occlusion	0	3	**71**

Total accuracy: 0.912

Table B.4: Confusion matrix showing the predictions done by the classifier trained on the augmented data.

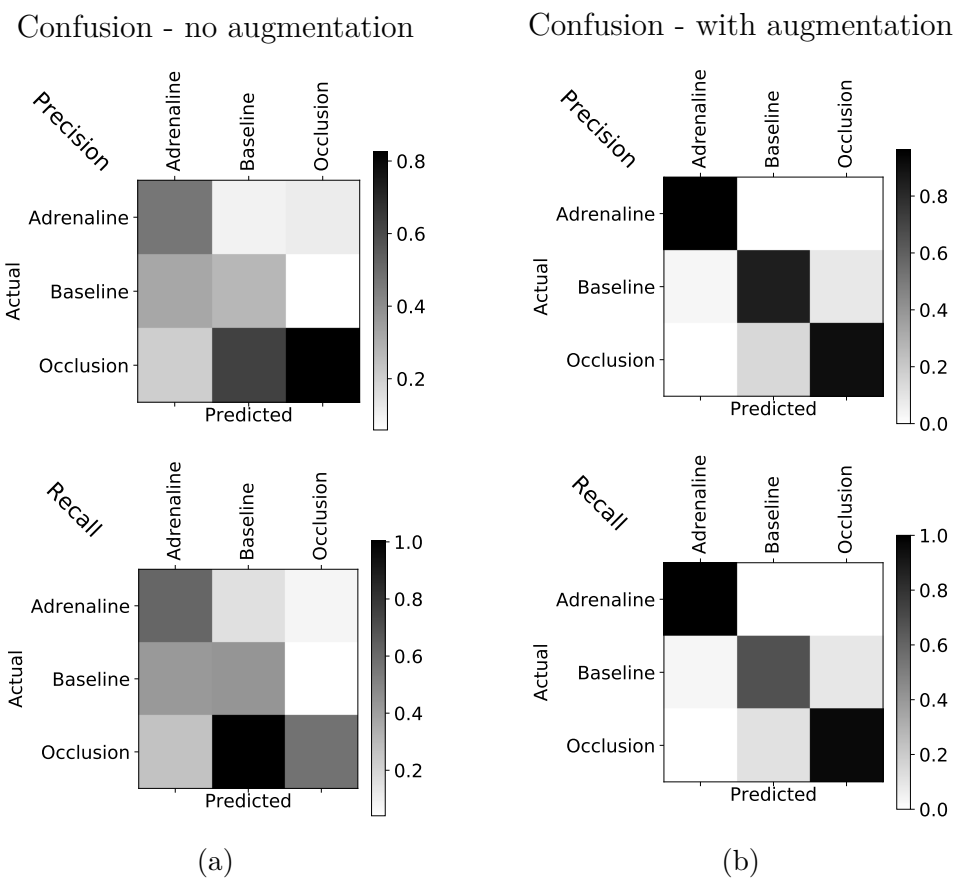


Figure B.2: (a) Shows the normalized confusion matrix for the classifier trained on data not augmented. The diagonal on the top figures represents the precision, and the diagonal on the bottom figures represent the recall. (b) As for (a), but shows the classifier trained on augmented data.