Universiteit Leiden
The Netherlands

Opleiding Informatica

ECG Anomaly Detection

Using Long Short-Term Memory based Recurrent Neural Networks

Ruduan B.F. Plug

Supervisors:
Dr. W. A. Kosters
Dr. W. J. Kowalczyk

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

July 1, 2019

Abstract

Neural networks are a novel group of systems that can be used to model and encompass data sets that are nonlinear dynamic in structure, by augmenting neural networks with recurrent capability we can also model continuous dynamic processes.

One of those nonlinear dynamic processes is the measured electrical activity of the heart by ELECTROCARDIOGRAM, of which the voltage time series data may have a high variance between measures and different patients. Such variances can be explained due to the heart's condition, which may provide vital information about the patient's condition to a cardiologist.

By using state of the art LONG SHORT-TERM MEMORY neural clusters instead of the standardized perceptrons in a RECURRENT NEURAL NETWORK we can extract such information from an ECG data stream and automatically classify heart impulse sequences to detect anomalies in the heart condition.

Research Question

To which degree of accuracy can Long-Short Term Memory based Recurrent Neural Networks be utilized to detect amplitude and phase anomalies in electrocardiogram time series data?

Contents

Introduction

1

> *"The soul is the same in all living creatures, although the body of each is different."*
>
> **— Hippocrates**

Since the beginning of the 20th century the field of medicine has seen many revolutionary advancements that would change the way we look at clinical treatment and the human physiology. The perspective broadened from exclusively rehabilitation to preventive and supportive medicine, as such within the UK life expectancy increased from 48.5 years for men and 52.4 years for women to 76.0 and 80.6 years respectively at the start of the 21st century [1].

Cardiovascular diseases are one of the primary causes of death in first world countries. According to *European Heart Network* (EHN), a joint cardiological research bureau, cardiovascular diseases accounted for 3.7 million deaths within Europe in 2017. This accounts for 45% of all 2017 mortalities within Europe [2]. Currently over 85 million people, more than ten percent of the European population, are suffering from a cardiovascular condition. Early diagnosis and treatment is critical in opposing this phenomena.

1.1 Cardiac Physiology

To understand data structures pertaining to the activity of the heart and their interconnection to underlying cardiac conditions, one must first understand the physiology of the heart. The heart is a muscular organ that is at the center of the cardiovascular system and is responsible for pumping blood through the circulatory system.

A constant flow of blood is crucial in supplying organs, muscles and other tissues with oxygen and nutrients while carrying away metabolic waste. The circulatory system can be divided into two parts: the pulmonary circuit and the systemic circuit.

The pulmonary circuit is responsible for oxygenating red blood cells. Blood flows into the heart from the superior vena cava and inferior vena cava and collects into the heart's right atrium as shown in Figure 1.1. When the heart relaxes blood flows into the right ventricle and when the heart compresses the tricuspid valve closes and blood is pushed into the pulmonary artery towards the lungs where the blood gets oxygenated [3].

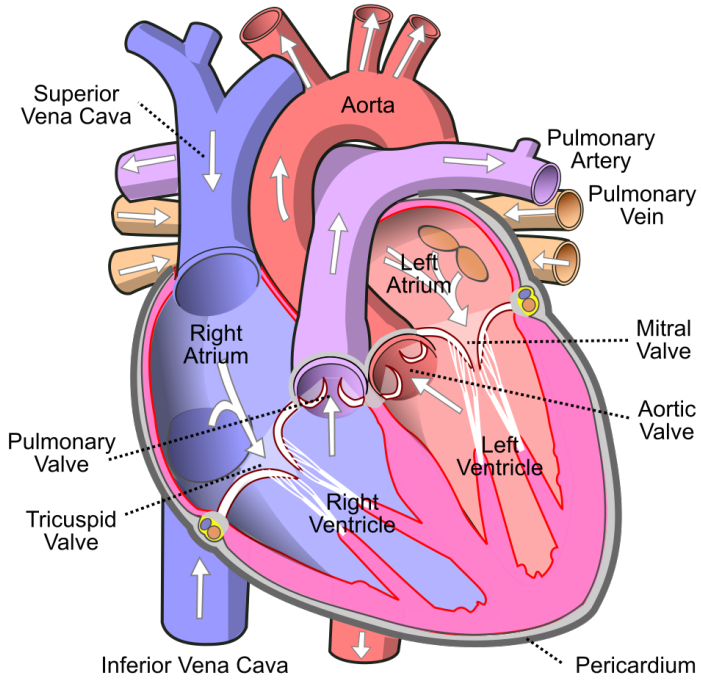


Fig. 1.1.: Circulatory structure of the heart [4].

The systematic circuit carries oxygen-rich blood from the heart towards organs and tissues throughout the body, to achieve this the left side of the heart is more muscular and is able to generate a higher blood pressure. Blood flows from the pulmonary veins back to the heart into the left atrium. During decompression of the heart blood flows from the left atrium into the left ventricle. When it compresses the mitral valve closes and blood from the left ventricle is pushed into the aorta, which is connected to the main arteries in the body.

Typically during compression the blood pressure within the aorta and left ventricle is 120 mmHg [5]. To withstand such pressure the aorta is up to 2 mm thick, which makes it the thickest artery in the human body. The immense pressure generated can easily be observed by means of touch or by listening with a stethoscope directly on the chest. However, there are more advanced and informative ways to monitor the heart.

1.2 Cardiac Electrical System

Blood flows through the circulatory system at a steady rate due to the heart muscle contracting and relaxing rhythmically. The contractions are triggered by a complex electrical system. The complete routine from the generation of an electrical signal and the resulting motion is named the cardiac cycle [6].

The cardiac cycle starts at sinoatrial (SA) node located in the right atrium as illustrated in Figure 1.2. This cluster of cells called the cardiac myocytes can generate an electrical action potential across the membrane of the node by means of rapid depolarization from -90 mV to $+10$ mV using Ca^{2+} pumps [7]. Between each cycle the myocytes repolarize by opening potassium channels resulting in exchange of K^{+} over a refractory period of approximately 300 ms. The rate at which the cardiac action occurs within the SA node is influenced by the autonomic nervous system.

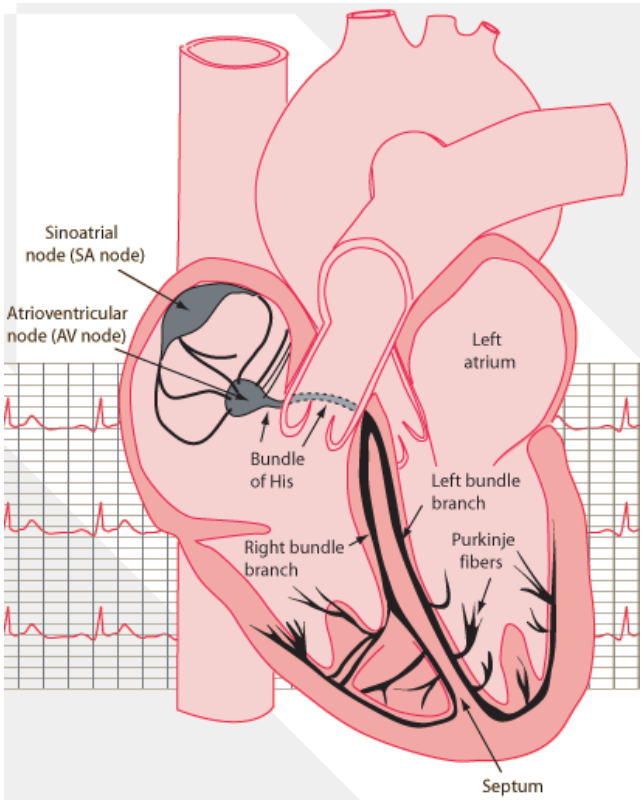


Fig. 1.2.: The heart's electrical system [8].

After an electrical potential has been generated it travels down to the atrioventricular (AV) node located near the center of the heart. Whereas the SA node generates the electrical impulse, the AV node coordinates the flow of the electrical signal within the heart.

As discussed in the previous section, to maintain proper blood flow it is essential that the atria has ejected all blood into the ventricles before they contract. As such, the atria and the ventricles should not contract simultaneously. To achieve this the AV node induces a delay of approximately 90 ms before the signal is propagated through the bundle of his towards the lower ventricles, while the atria is stimulated without additional delay. The His bundle divides into left and right bundle branches that simultaneously carry the signal to their respective ventricles. The full electrical cycle ends when the Purkinje fibers located in the lower ventricles propagate the signal throughout the ventrical myocardium [7, 9].

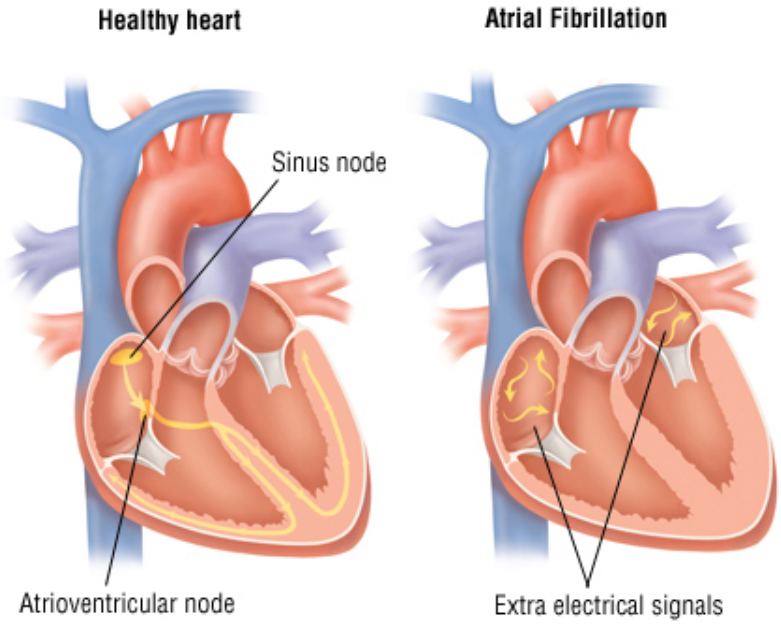


Fig. 1.3.: The difference in cardiac conductivity patterns [10].

In this research in particular we will be looking at a specific case of an anomaly named atrial fibrillation (*Afib*). Atrial refers to the upper heart chambers and fibrillation indicates the irregular contraction of muscle fibers without proper coordination [11]. In this case the atria contract irregularly and out of synch with the ventricles due to a complication within the heart's electrical system which is illustrated in Figure 1.3. Afib may lead to more serious complications such as blood clotting and is an early symptom or precursor for many life threatening cardiovascular diseases such as stroke, heart failure and myocardial infarction [12, 13].

The Electrocardiogram

2

> *"The typical heart anomaly has remained unchanged all this long time, but what was then a puzzle is now explained."*
>
> **— Willem Einthoven**

The electrocardiogram (ECG) is a composite measurement of the heart's electrical activity. This observational procedure is generally performed in a clinical environment by placing 12 electrode leads on the subject at specific positions, which measure minute changes in electrical potential of the dermal membrane [14]. The resulting data is used to monitor the status of the heart and contains information about structures in the heart due to the temporal and spatial dependency of the electrical impulse propagating step-by-step from the SA node through the cardiac nerve pathways.

The complete periodic pattern within the cardiogram is called the cardiac complex or cardiac waveform. This consists of multiple wave components, each providing information of activity in different stages of the heart's depolarization action leading to compression. Proper segmentation and analysis of these individual waveforms is required to be able to extract accurate information from the complete ECG [15].

2.1 The History

The foundation of the cardiogram starts with the discovery of the existence of bioelectricity by Luigi Galvani in 1791. Since then countless experiments have been performed to uncover the electrical propagation properties of muscle and nerve fibers [16]. A few decades later, in 1856, the periodic electrical activity was observed in the heart by Köllicker and Müller at the University of Würzburg. This experiment was performed by connecting wires from the surface of a frog's heart to it's sciatic nerve, after which a periodic current could be observed proportional to the heart activity, causing the connected leg muscle to contract synchronistically.

But it was not until 1887 that the first heart activity in a human has been recorded by Augustus D. Waller in what we now call an electrocardiogram using Gabriel Lippmann's capillary electrometer. The results however had proven to be erratic and as such the measurements were not usable to any clinical degree.

This inspired Willem Einthoven at the University of Leiden to invent the string galvanometer in 1901, which he would eventually develop together with a theoretical framework into the first accurate and clinically usable electrocardiogram device shown in Figure 2.1. For this discovery he has been awarded the Nobel Prize in 1924 [17].

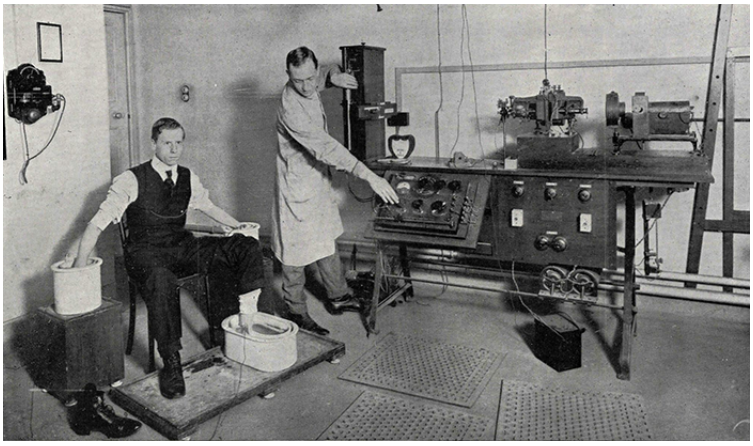


Fig. 2.1.: The String Galvanometer [18].

2.2 Clinical Measurement

The modern electrocardiogram device, also named the ECG monitor, uses the same principle as originally named Einthoven's Triangle. It consists of a baseline measurement comprised of three bipolar lead electrodes (I, II, III) attached to the arms and left leg, creating a baseline dynamic membrane potential with the heart as the center of the triangle as shown in Figure 2.2. In addition three positive unipolar electrical leads (aVR, aVL, aVF) are extracted from the same electrodes, which form the perpendicular augmented vector measurements for each limb in relation to the other leads [19].

This results in six baseline angles of measurement around the coronal plane orthogonal to the chest, 0°, 60° and 120° for leads I, II and III and 30°, −30° and 90° for leads aVR, aVL and aVF respectively for the electrical charge using the hexaxial reference system [20].

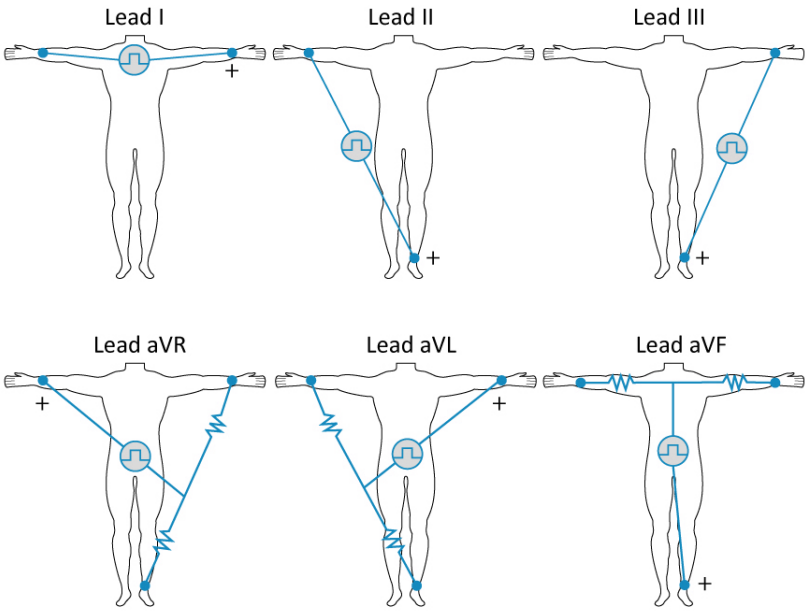


Fig. 2.2.: Limb Lead Placement.

According to Kirchhoff's law the voltages within a loop must equate to zero. Given the electrode potential Φ and voltage V we find for the Einthoven leads V_i:

$$V_I = \Phi_L - \Phi_R$$

$$V_{II} = \Phi_F - \Phi_R$$

$$V_{III} = \Phi_F - \Phi_L$$

From this the augmented leads can be derived using an equilateral triangle construction, shown in Figure 2.3, as such:

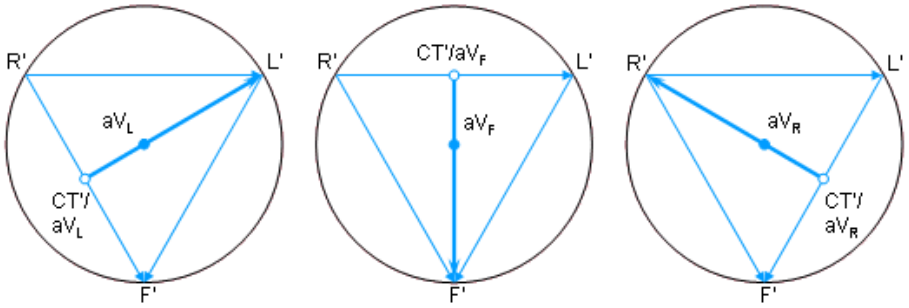


Fig. 2.3.: Electrically Equilateral Triangle Circuit.

$$\alpha V_F = \Phi_F - \frac{\Phi_R + \Phi_L}{2}$$

$$\alpha V_R = \Phi_R - \frac{\Phi_F + \Phi_L}{2}$$

$$\alpha V_L = \Phi_L - \frac{\Phi_F + \Phi_R}{2}$$

Finally, the cardiac baseline potential Φ_α, based on point measurement of Wilson's central terminal Φ_{CT}, within the triangle is given by the potential over the area of the equilateral triangle:

$$\Phi_{CT} = \frac{\Phi_F + \Phi_R + \Phi_L}{3}, \quad \Phi_\alpha = \frac{\sqrt{3}}{4}\Phi_{CT}^2$$

Now the baseline potential has been established six more electrodes C_n, as illustrated in Figure 2.4, are used to measure the heart's electrical output in the transverse plane. These are the precordial unipolar leads V_1, V_2, V_3, V_4, V_5 and V_6 that are located around the heart on the upper chest. These measure the heart's electrical output passing through the skin in detail, which can then be normalized by using Einthoven's triangle construction [21].

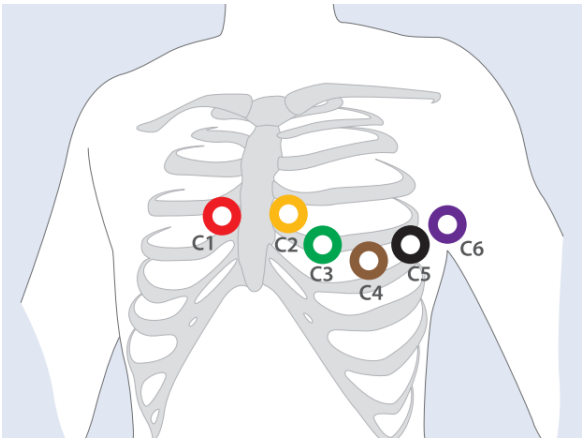


Fig. 2.4.: Precordial Electrode Placement.

The twelve transverse and coronally oriented leads measure the potential passing through the dermal membrane from different angles. The voltages generated by these leads are mapped onto a graph to display the electrical actuation over time. The standard scale is an amplitude equivalent to 1 mV and period equal to one fifth of a second relative to the reference pulse. The complex of all twelve signals, see Figure 2.5, forms the full ECG waveform.

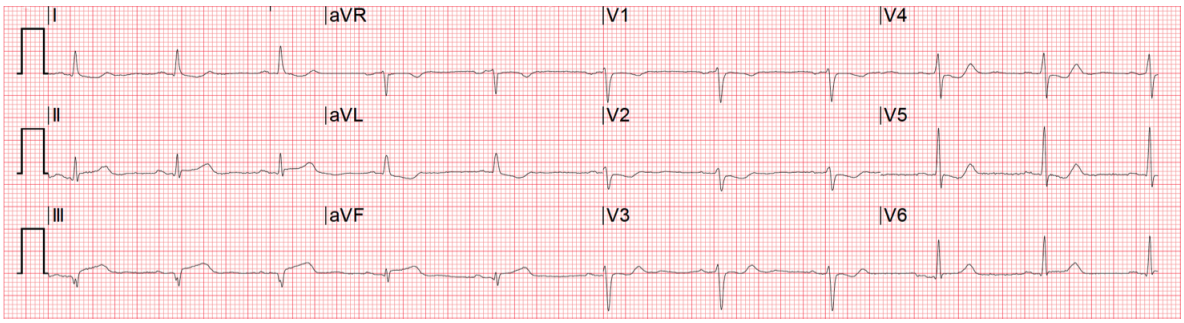


Fig. 2.5.: A standardized 12-lead ECG Recording [22].

2.3 Cardiac Waveform

As discussed in the previous section, the cardiac waveform is a composite function of the twelve leads measured from the ten electrodes attached to the subject. The main area of interest, the cardiac complex, are the three main waves we will be looking at through the process of feature extraction. These three waves contain novel informative features about the condition of the heart. In cardiology this process is called segmentation of the cardiac complex.

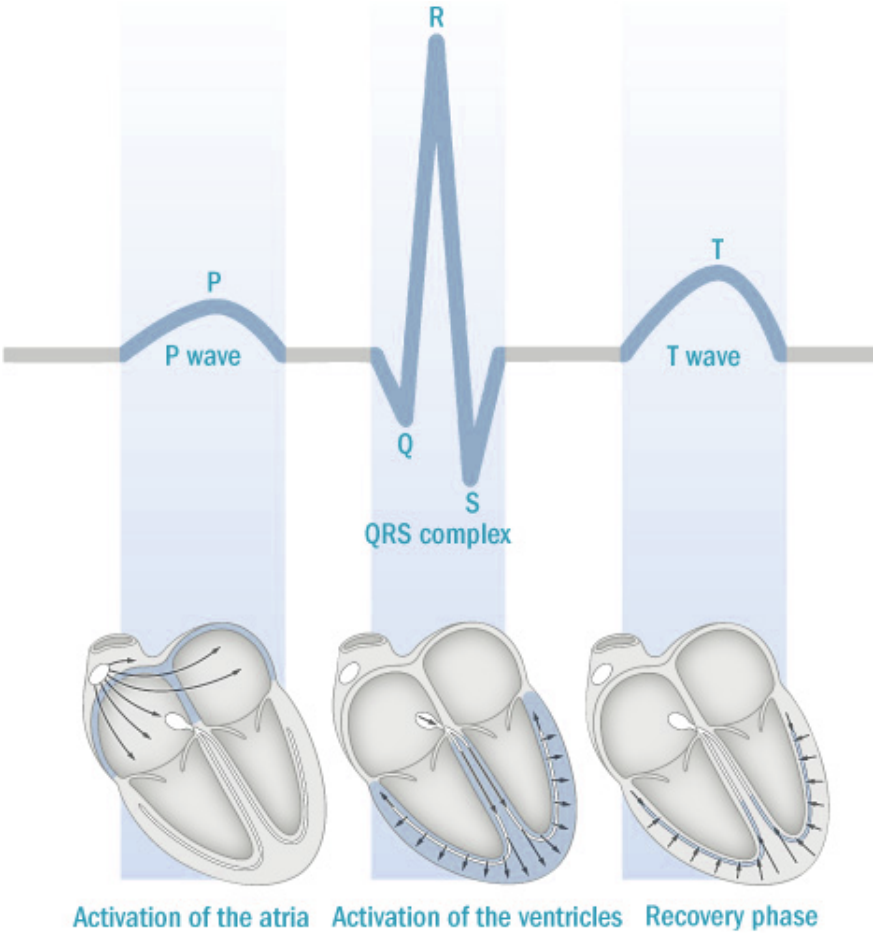


Fig. 2.6.: Segmented Cardiac Wave [23].

The cardiac complex can be segmented into distinct features P, Q, R, S and T. The resulting wave segments for the cardiac wave are the *P wave*, *QRS complex* and the *T wave* in order of occurrence with local refractory intervals PR and QT after the first two main waves [24]. Each of these segments corresponds to a physiological section of the heart during depolarization through the progress of time.

· P Wave

The P wave is the first electrical impulse at the start of the cardiac action. It has has a typical amplitude of 0.05 mV to 0.25 mV and an interval of approximately 80 ms in healthy individuals. This wave indicates the cycling of the SA node and the resulting depolarization of the atria [14].

An abnormally high or low P wave may indicate an abnormal level of K^+ ions in the blood serum, which are normally between a concentration of $3.5 \cdot 10^{-3}$ and $5.0 \cdot 10^{-3}$ mol/L. A An amplitude over 0.25 mV might suggest enlargement in the right atrium while a split P wave might indicate an irregularity in the left atrium [7].

Most notably an irregular propagation of the wave can be indicative of atrial fibrillation, as the electrical current propagates irregularly throughout the myocardium of the atrium. This is illustrated in Figure 2.7.

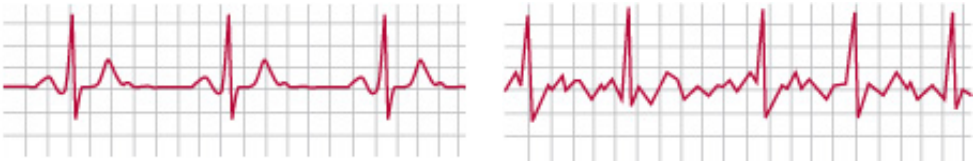


Fig. 2.7.: Healthy pattern compared to the Afib type pattern [22].

· PR Segment

The PR segment is the isoelectric period between the P wave and the QRS complex. It typically lasts 40 to 120 ms at a zero potential, indicating the delay between the contraction of the atria and the transfer of the electric signal from the AV node through the bundle of His [7].

A PR segment longer than 120 ms might indicate a first-degree atrioventricular blockage, in which the signal through the AV node is delayed for too long, which is a risk factor for Afib and other complications [12]. An abnormally short PR segment indicates that the signal has bypassed the AV node guard, also called the WPW pattern, which may pose a faiting and sudden cardiac arrest risk.

· QRS Complex

The QRS complex is the main cardiac action performed by the ventricles, a large rapid depolarization occurs in the myocardium connected to the purkinje fibers. [9] The period of this action is typically 60 ms to 100 ms with a peak potential of 3.5 mV.

The QRS complex is an essential part of the heartbeat and small differences in the complex may have serious consequences to the general health of a person. One of such abnormalities is a delayed or widened QRS wave which may be caused by blockages in the bundle of His or the Tawara branches.

Abnormalities within the QRS complex may be a precursor to fatal conditions such as a cardiac arrest or a myocardial infarction. [19] But QRS anomalies are difficult to observe because it requires high frequency data analysis and may present itself in minute differences due to the high baseline derivatives within the segment.

· ST Segment

Just like the PR segment, the ST segment is an isoelectric period between the rapid depolarization of the ventricles before the repolarization occurs. A typical ST segment lasts between 80 ms and 120 ms of baseline zero potential measured across the electrodes. Clinically the ST segment is the most important indicator within an ECG and will generally be the first point of data a cardiologist will analyze [19].

A consistent significant net positive or negative baseline current of 0.1 mV or more for over 80 ms within the the ST segment may be an indicator of a myocardial infarction and is an medical emergency. A down-sloped potential is indicative of a blockage within the coronary arteries called coronary ischemia which can cause heart failure if left untreated.

It is essential for the heart to complete its cardiac action before repolarization occurs and it requires proper blood flow within the heart to consistently and completely perform this procedure. As such the ST pattern gives critical information about the health of the heart itself.

· T Wave

The final part of the cardiac complex is the T wave. This is the segment in which the ventricles repolarize themselves for the next cardiac action to occur, it is considered to be the refractory period of the heart [3].

The T wave sees the highest variance in its duration, with a range of 100 ms to 250 ms in healthy individuals. Typically the T wave reaches no more than 1.0 mV but averages much lower than that with approximately 0.4 to 0.6 mV [14].

Despite the T wave indicating repolarization, of which the vector is opposite to the depolarization action, the measured potential is positive. This is due to the fact that the repolarization process occurs in the reverse direction of polarization, the last cardiomyocytes to depolarize will be the first to repolarize during the refractory. This process starts with the endocardium and ends with the epicardium. Since now a negative current is going away from the electrode it is registered as a positive potential because there are two negative vectors being multiplied.

Despite this a negative T wave may still occur, and if frequent this can be a symptom of an pulmonary embolism, myocardial ischemia or coronary artery disease. T waves that exceed a potential of 1.0 mV can be linked to angina or may be an early symptom of a myocardial infarction in combination with other markers [19].

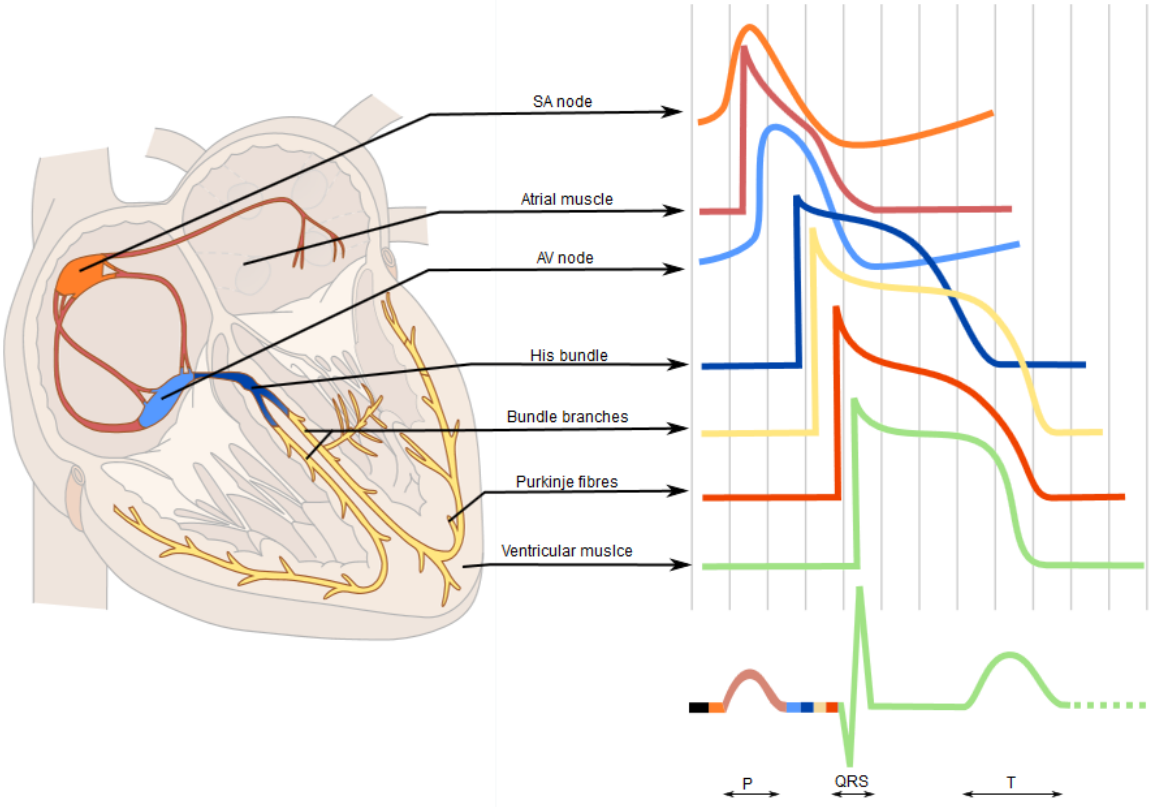


Fig. 2.8.: The complete waveform model of the heart [25].

The resulting waveform is the amalgamation of stochastic processes, shown simplified in Figure 2.8. As such it is non-trivial to derive a functional notation of the waveform. A singular waveform complex may be approximated using Fourier series, but a more sophisticated model is required to be able to make inferences about hypotheses regarding the cardiac status.

Learning Model

3

> *"For the things we have to learn before we can do them, we learn by doing them."*
>
> **— Aristotle**

In the previous chapters we have discussed the physiological structures of the heart and the resulting data that can be measured from the electrical activity. From this data we want to be able to detect anomalies and classify the data stream, in particular the myriad group of anomalies related to atrial fibrillation.

The data we retrieve from the 12-lead measurement is stochastic in nature due to the many dependencies [19]: the local electrical conductivity through the tissue, the autonomic nervous system affecting cardiomyocyte activity, epinephrine and norepinephrine hormonal levels influencing heart rate and the many biological and environmental processes related to the generation of electrical current by the myocardial fibers [7, 24]. To account for all these factors while only measuring the electrical potential we need to make use of statistical and computational methods to model and classify the cardiac waveform.

3.1 Linear Classification

To create our statistical model we first need to define the basic unit that we will use, which is the linear classifier. The linear classifier is a linear relation between two variables that splits a data set into two distinctly classified sets.

For this project in particular we will consider the *perceptron*, also called a neuron in respect to neural network architectures. This is a discrete or continuous connective unit that takes an input vector $\mathbf{x}$ where each element x_i has its own weight w_i derived from the weight vector $\mathbf{w}$ and a bias b [26].

A special property of the perceptron is the activation function, which is a function σ that maps the weighted continuous input with respective domain $x_i w_i \in (-\infty, \infty)$ to an activation value in the codomain of the activation function, which is typically $[0, 1]$ or $[-1, 1]$. The activation y is the output of the perceptron in regards to the mapping of the input, given by $y = \sigma(xw + b)$ for a single input perceptron. For the general case with n inputs the output of the perceptron unit is given by the activation function of the summed weighted input plus the bias:

$$f_{\hat{y}}(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

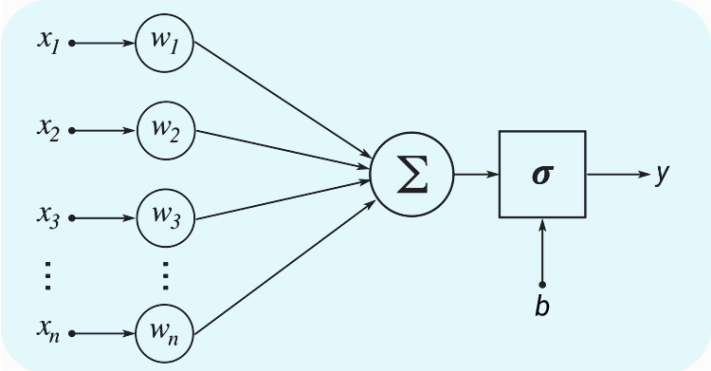


Fig. 3.1.: Perceptron Model.

There are many different kinds of activation units that can be used, and the most common ones are given in the table below. Each activation function has its own characteristics and performance levels for different kinds of classification problems [27]. A special case is the step-function activation unit, which is a discrete binary classifier.

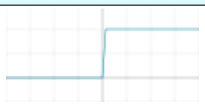
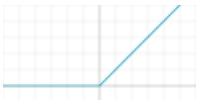
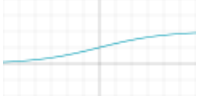
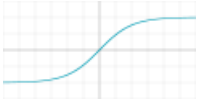
Function	Plot	Equation	Derivative	Range
Step		$\begin{cases} 0 \text{ if } x < 0 \\ 1 \text{ if } x \geq 0 \end{cases}$	$\begin{cases} 0 & \text{if } x \neq 0 \\ \text{undef. if } x = 0 \end{cases}$	$[0, 1]$
ReLU		$\begin{cases} 0 \text{ if } x < 0 \\ x \text{ if } x \geq 0 \end{cases}$	$\begin{cases} 0 \text{ if } x < 0 \\ 1 \text{ if } x > 0 \end{cases}$	$[0, \infty)$
Sigmoid		$\sigma(x) = \dfrac{1}{1 + e^{-x}}$	$\sigma(x)(1 - \sigma(x))$	$(0, 1)$
Tanh		$\tau(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	$1 - \tau(x)^2$	$(-1, 1)$

Table 3.1.: Common Activation Functions.

The weights are the main factor that determine the classification performance. The weights need to be set such that the most possible data is correctly classified, as such determining the weights is an optimization problem. In the next section we will focus on the technique that is used to optimize the weights.

The main challenge of linear classifiers is that they can only segment data through data set division with linear boundaries, creating two new subsets of data. This poses a problem when trying to classify data that is not linearly separable, such as XOR-type data. In addition, linear classification doesn't allow us to model complex polynomial or periodic functions.

3.2 Gradient Descent

To optimize the weight parameters for a perceptron we need to use an optimization technique. One of such optimization algorithms is *gradient descent*, which is based on random initialization and local error minimization through iterative weight adjustment with training samples.

Gradient descent is a first-order optimization technique, therefor the objective function must be differentiable on the optimization domain to find the gradient vector for the target variable. In this case we will be looking at the vector $\mathbf{w}$ with input $\mathbf{x}$ in the multi-variable perceptron function f_y, for which we can optimize the weight vector towards a function minimum of the error function in the convex plane for randomized parametric starting vector $\mathbf{p}$ initialized to $\mathbf{w}$ [28].

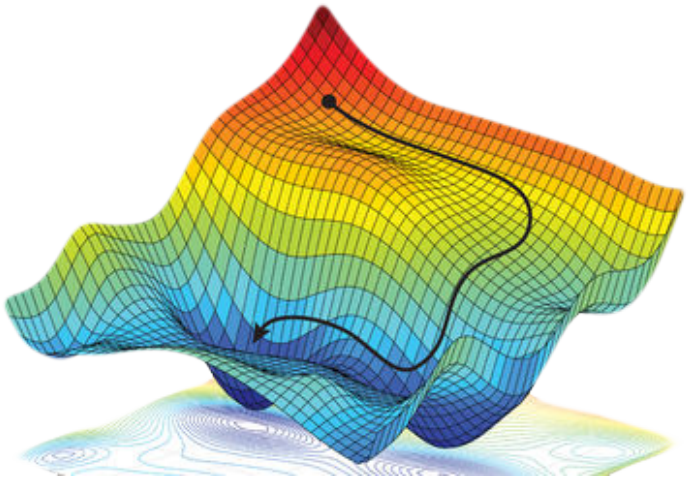


Fig. 3.2.: Stochastic Gradient Descent along a Convex Plane [29].

The process that gradient descent follows from the initial points is to find the gradient and proceed along the curve derived from the weight vector that shows the steepest possible decline in the error function output with step size η, as such we can define the update function for weight w [30]:

$$w_{t+1} \leftarrow w_t - \eta \nabla E(y_t, \hat{y}_t)$$

The error function is derived from the mean squared error in the regression technique, which is proportional to the deviation of the estimate $\hat{y}$ to the expected target value y:

$$E(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Each iteration we move η steps against the gradient until the gradient becomes zero, at which point the process converges at the local minimum. This process always converges as long as the plane for the error function is convex such that ∇E becomes zero along the path of the negative gradient such that $E(\mathbf{y}, \hat{\mathbf{y}}) \supseteq \{ \exists y, \hat{y} \mid \nabla E(y, \hat{y}) = 0 \}$. To apply this technique we need to define the multivariate gradient of the error function as the partial derivative in respect to the weight on the contour to get the weighted loss function. For a general case we use a linear hypothesis $h_\theta(x) = \theta_0 + \theta_1 x$ for the best fit to the training data where θ_0 is the bias and θ_1 the linear weight.

$$\theta_{t+1} = \theta_t - \eta \frac{\partial}{\partial \theta_t} L(\theta_t)$$

$$L(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (h_\theta(x_i) - \hat{y}_i)^2$$

$$\frac{\partial}{\partial \theta_j} L(\theta) = \frac{1}{2n} \sum_{i=1}^{n} 2 (\frac{\partial}{\partial \theta_j} h_\theta(x_i) - \hat{y}_i)^{2-1} = \frac{1}{n} \sum_{i=1}^{n} (h_\theta(x_i) - \hat{y}_i) x_j$$

To get the complete gradient descent equation for a function with j variables and n training data for hypothesis h_θ the update complete equation is as follows:

$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) x_t^\ell$$

A drawback of gradient descent is that it can very easily get stuck in compact local minima if the gradient becomes small enough. A solution to this is to use stochastic gradient descent as pictured earlier. Instead of optimizing over the complete set for every epoch, only a small random subset of the training set is used to find the lowest gradient within the set, which makes it less computationally expensive and gives it probabilistic properties which may help with global optimization through regression to the mean.

3.3 Non-Linear Classification

The linear classification given by the perceptron unit is limited in accuracy for complex data sets, as it can only accurately optimize weights for a linear functional relationship between the inputs and the output. To introduce non-linearity we build a network of perceptrons called the *multi-layer perceptron* (MLP), which is a basic class of the *feedforward neural network* (FFNN) architecture [31].

To make the non-linear data linearly separable input $x_1, \ldots, x_p$ is projected by a non-linear transformation Φ, which is learned by a cluster of perceptrons linked together. Within Φ-space the data becomes linearly separable by the final linear output regressor. This transformation Φ is formed by intermediate layers of interconnected perceptrons called the hidden layers which take data from the input layer and pass it forward towards the output layer $y_1, \ldots, y_k$. For each perceptron from the input to the output, the activation of the neurons from the previous layers forms the input of the perceptrons in the next layer, see Figure 3.3.

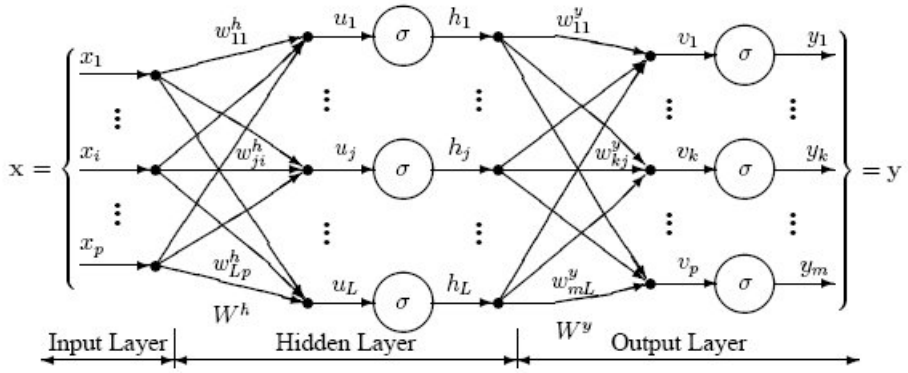


Fig. 3.3.: Multilayer Perceptron Architecture.

A single hidden layer with finite neurons is sufficient for a neural network to become a universal approximator of n-dimensional continuous functions in $\mathbb{R}^n$ space. The output vector is given for an input vector multiplied by the transpose of the weight matrix and taking the activation over the intermediate output vector [32] with n inputs and k outputs.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \sigma(\mathbf{w}^{\mathrm{T}} \circ \mathbf{x} + \mathbf{b}) = \sigma \left(\begin{bmatrix} w_{1,1} & w_{2,1} & \cdots & w_{n,1} \\ w_{1,2} & w_{2,2} & \cdots & w_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,k} & w_{2,k} & \cdots & w_{n,k} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix} \right)$$

This can be expanded to generalize over ℓ layers of perceptrons by using the output vector as the input vector of the next layer to recursively generate the entire output vector for the network.

$$\hat{\mathbf{y}}(\ell) = \begin{cases} \hat{\mathbf{y}}_1 & = \sigma(\mathbf{w}_1^{\mathrm{T}} \circ \mathbf{x} + \mathbf{b}) \\ \hat{\mathbf{y}}_\ell & = \sigma(\mathbf{w}_{\ell-1}^{\mathrm{T}} \circ \hat{\mathbf{y}}(\ell-1) + \mathbf{b}) \end{cases}$$

To optimize the weights of the entire network we perform an iterative process from the output to the input called backpropagation. We do this by performing gradient descent on the output calculated through feed forward compared to the pre-classified output in the training set by taking the Jacobian matrix of the error in respect to the weight.

$$\nabla E(\mathbf{y}, \hat{\mathbf{y}}) = \begin{bmatrix} \frac{\partial E}{\partial w_{1,1}} & \frac{\partial E}{\partial w_{1,2}} & \cdots & \frac{\partial E}{\partial w_{1,k}} \\ \frac{\partial E}{\partial w_{2,1}} & \frac{\partial E}{\partial w_{2,2}} & \cdots & \frac{\partial E}{\partial w_{2,k}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial w_{n,1}} & \frac{\partial E}{\partial w_{n,2}} & \cdots & \frac{\partial E}{\partial w_{n,k}} \end{bmatrix}$$

The chain rule gives us the weight delta using the loss function and activation function derivative:

$$\frac{\partial E}{\partial w_{n,k}} = \frac{\partial E}{\partial y_n} \frac{\partial y_n}{\partial y_{n-1}} \frac{\partial y_{n-1}}{\partial w_{n,k}} = L(y_k) \frac{\partial}{\partial x_\ell} \sigma(x_\ell) y_{n-1}$$

From this we can derive the update function of the weight matrix:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \nabla E(\mathbf{y}, \hat{\mathbf{y}})$$

3.4 Recurrent Learning

Although regular multi-layer perceptron neural networks can provide a model for non-linear equations, they have poor performance on continuous processes with stochastic elements. The first problem is that these networks cannot effectively handle variable input sizes, relying on a mapping or compression technique which makes the input lossy and reducing the accuracy as a consequence. Secondly, these networks have no sense of order in which the data is input, the time-series type data we are studying have stochastic and temporal dependencies which affect future outcomes.

These classes of problems cannot be solved without memory. To resolve this we will use a special class of neural networks called the *recurrent neural network* (RNN). Instead of learning patterns, the RNN learns sequences in a specific order by maintaining an internal state. [33] To achieve this the RNN has an internal state that is modified between each time step and the output for every t depends on not only the input x_t but also the state of the hidden layer in the previous step denoted by h_{t-1} shown in Figure 3.4.

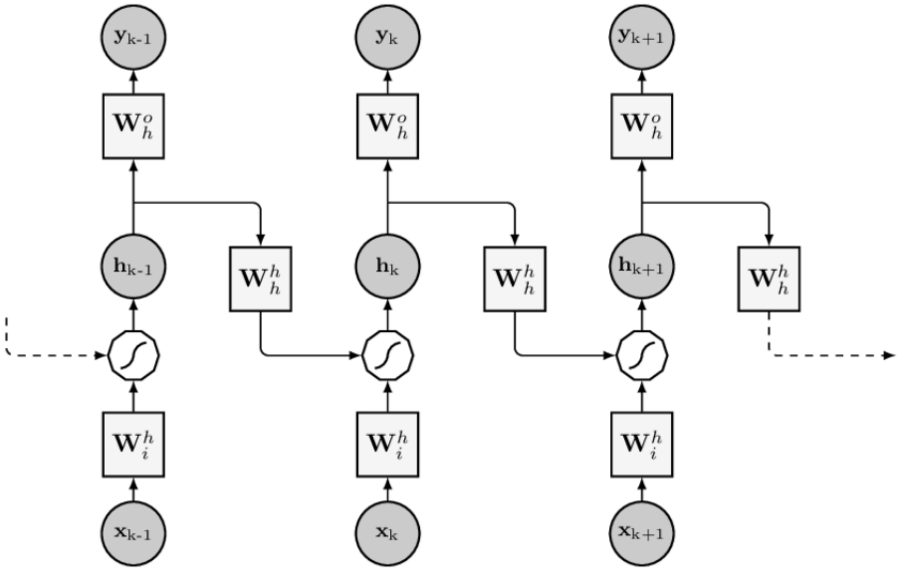


Fig. 3.4.: Unrolled RNN Representation [34].

The recursive feed forward process is very similar to the MLP, except now we also have a factor from the previous layer and three new weights that are shared across iterations [35].

$$\hat{\mathbf{y}}_t = \sigma(\mathbf{w}_y^{\mathrm{T}} \circ \mathbf{h}_t + \mathbf{b}_y) = \sigma(\mathbf{w}_y^{\mathrm{T}} \circ \sigma(\mathbf{w}_h^{\mathrm{T}} \circ \mathbf{h}_{t-1} + \mathbf{w}_x^{\mathrm{T}} \circ \mathbf{x}_t + \mathbf{b}_h) + \mathbf{b}_y)$$

To train a recurrent network we need a modified learning rule called back-propagation through time. For every time step t, we train the n weights of the previous $t-1$ steps by calculating the error over y and $\hat{y}$ for t. To perform this we sum the derivative chain rule over all previous time steps.

$$\nabla E(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k=0}^{n} \frac{\partial E_k}{\partial w} = \sum_{k=0}^{n} \sum_{i=0}^{t} \left(\frac{\partial E_k}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_i} \frac{\partial h_i}{\partial w} \right)$$

The weights are modified after each time step as usual, substituting the iterator for the temporal variable:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla E(\mathbf{y}, \hat{\mathbf{y}})$$

There are various schemes available for the recurrent hidden layer within the RNN, the most simple one being a standard perceptron. While quick to train, in most cases it does not have optimal convergence properties because of a problem called the *vanishing gradient* [36].

The vanishing gradient is a phenomena that arises during training of a neural network's weights, in particular relating to deep and recurrent neural networks. As discussed earlier both within RNN and deep MLP networks, the gradient is generated by iterating over the derivatives for each node. The further we backpropagate within the network towards the initial nodes, the more derivatives are multiplied relative to each other using the chain rule.

As a result, when the network is presented with input which has a section that has a low derivative, the activation function squashes the data down to a low factor. Which when multiplied through the layers of the network causes the derivative of the deeper layers to approach zero, reducing the overall learning rate significantly.

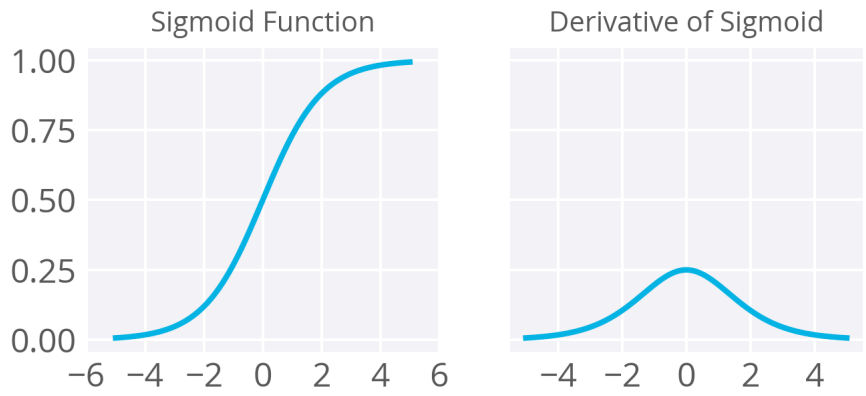


Fig. 3.5.: The logistic sigmoid function and it's derivative.

In certain types of time series data this is a particular problem because between segments of informative data, there may be sections of data with a low or near-zero derivative. When training a RNN on this type of data the learning rate will approach zero when encountering these sections of non-informative data that we would rather not train on.

A solution to this problem is to use a different kind of hidden layer architecture, one of such is the *long-short term memory* (LSTM) node [37]. The LSTM is an architecture that preserves it's learning rate by using an internal state to store informative data while discarding uninteresting data.

As such LSTM nodes can store long-term dependencies without overfitting on data that doesn't possess the desired features that we want to train on, in essence we want to train onto the dynamic changes of the data based on it's inherent dependencies rather than the scalar property itself.

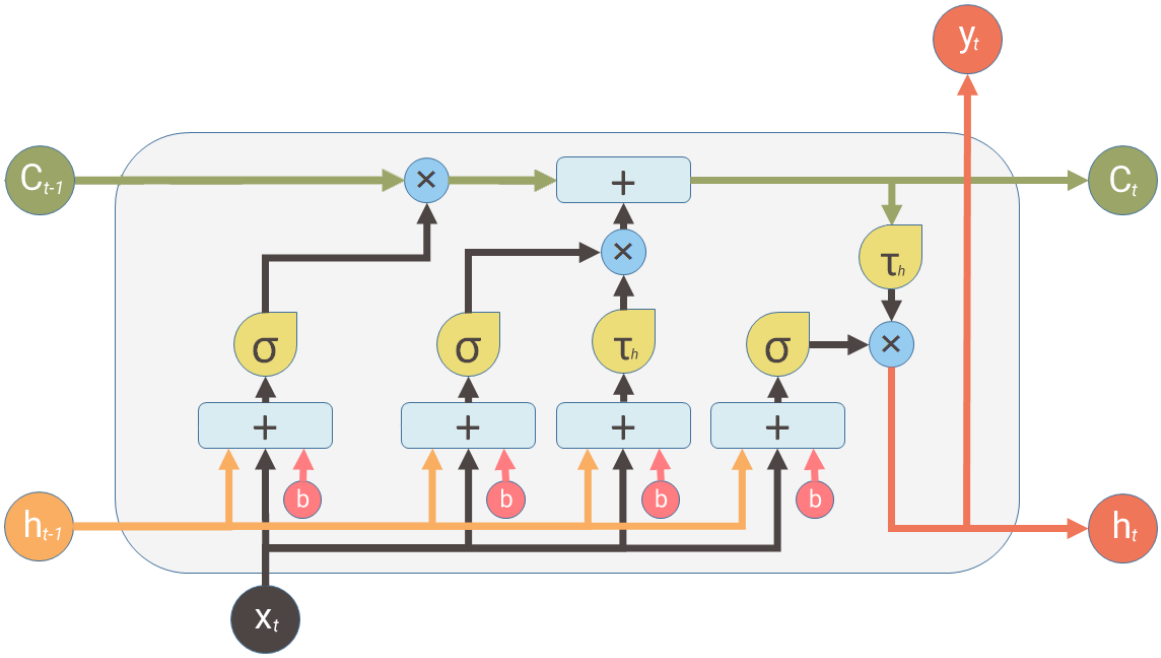


Fig. 3.6.: A schematic of the LSTM architecture [38].

The main feature of the LSTM is that it utilizes a vector $\mathbf{C}_t$ which denotes the cell state at time step t. The cell has three inputs: the previous cell state vector $\mathbf{C}_{-1}$, the previous hidden output vector $\mathbf{h}_{t-1}$ and the input vector $\mathbf{x}_t$. The outputs are the new cell state $\mathbf{C}_t$, the hidden layer output $\mathbf{h}_t$ and the net output vector $\mathbf{y}_t$, see Figure 3.6. The architecture is divided into three distinct gates that govern the internal state and all lanes are vectorized operating in parallel on Hadamard products of the inputs [37].

· **Forget Gate** — *decays cell state by factor*

$$\mathbf{f}_t = \sigma(\mathbf{w}_f \mathbf{x}_t + \mathbf{w}_h \mathbf{h}_{t-1} + \mathbf{b}_f)$$

· **Input Gate** — *adds to the active cell state*

$$\mathbf{i}_t = \sigma(\mathbf{w}_i \mathbf{x}_t + \mathbf{w}_h \mathbf{h}_{t-1} + \mathbf{b}_i)$$
$$\hat{\mathbf{c}}_t = \tau_h(\mathbf{w}_c \mathbf{x}_t + \mathbf{w}_h \mathbf{h}_{t-1} + \mathbf{b}_c)$$

· **Output Gate** — *outputs based on activation and cell state*

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \hat{\mathbf{c}}_t$$
$$\mathbf{h}_t = \sigma(\mathbf{w}_y \mathbf{x}_t + \mathbf{w}_h \mathbf{h}_{t-1} + \mathbf{b}_y) \circ \tau(\mathbf{c}_t)$$

4 Data Analysis

> *"Life is short, and art long, opportunity fleeting, experimentations perilous, and judgment difficult."*
>
> **— Hippocrates**

So far we have investigated the mechanics behind cardiac physiology and electrocardiography. We have assessed that we require a deep learning approach to generate an appropriate model and we have defined the LSTM architecture as a viable candidate to model and classify our time series type data.

In this chapter we will concretely focus on specifying, processing and exploring the data set we have at our disposal. Once we have a candidate input data set will use a recurrent neural network based on the LSTM architecture to build a model for the data. During this process we will need to fine tune the parameters of the architecture to optimize the learning rate and classification performance on our data set. Finally we will compile and report our results and compare the effectiveness of our model compared to other recurrent neural network architectures.

4.1 Data Set

The data set we will be using for this project are anonymized composite ECG recordings from the PhysioNet database, which were released for the 2017 edition of the Computers in Cardiology (CinC) conference [39].

The data set consists of 8528 recordings from unique patients, each lasting between 10 and 60 seconds. The sampling rate is a constant 300 Hz at a resolution of 16 bits, recorded as signed words. The resulting signals contain 3000 to 18000 samples with a dynamic input range of 10 mV, measuring electrical activity in the frequency range of 0.5 to 40 Hz. The only available attribute is the voltage at each time frame in millivolts.

The recorded data set has been annotated by a clinical expert brought forward by the CinC committee, resulting in the following labels: Normal (N), Atrial Fibrillation (A), Other (O) and Noisy (∼). A subset of the entries with low classification confidence was brought up for majority consensus among the expert committee. A summary of the labeled data set is given in table 4.1.

Label	**Count**	**Sample Length (10^3)**					**Variation Coefficient**
		Mean	Median	Standard Deviation	Min	Max	
N	5050	9.63	9.00	2.99	2.71	18.29	0.31
A	738	9.63	9.00	3.71	3.00	18.06	0.39
O	2456	10.32	9.00	3.52	2.74	18.26	0.34
∼	284	7.26	9.00	3.10	2.81	18.00	0.43
Total	8528	9.75	9.00	3.21	2.71	18.29	0.33

Table 4.1.: Data set statistics.

Each sample has been filtered using high-pass and low-pass filters to remove interference from unrelated muscle activity and external sources. By segmenting the data into the QRS, P and T components appropriate bandpass filters can be applied to remove noise outside the working frequency.

For the QRS segment the target frequency is 4 to 20 Hz and for the P and T segments this lowers to 0.5 and 10 Hz. Note that there is an overlap between the electrical background noise between range 0 and 0.8 Hz and the target range which means a certain degree of noise cannot be attenuated in the P and T segments between 0.5 and 0.8 Hz without presenting data loss. This presents us with the following distribution for the signals as given in Figure 4.1, which are processed using MATLAB.

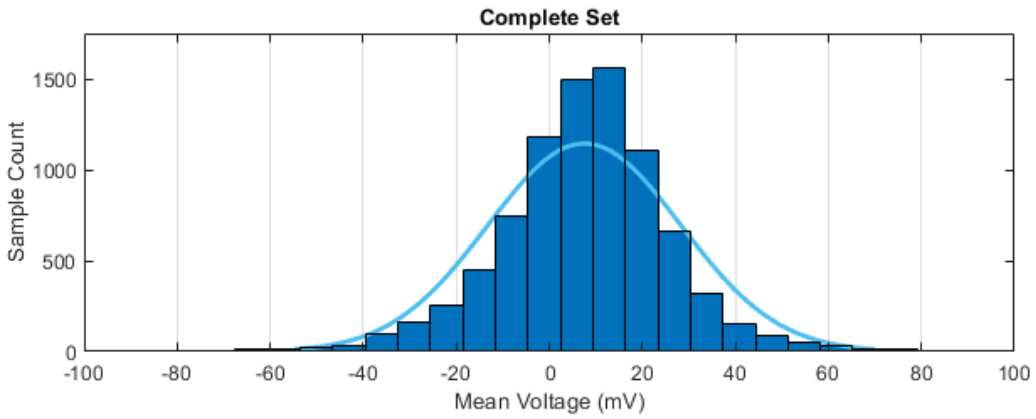


Fig. 4.1.: Distribution of the signal output.

From the processed signal set we can derive the statistics for the output data in table 4.2. Right away we can observe that the signals containing anomalies have a substantially higher standard deviation than the normal set, however these statistics alone are not sufficient to make accurate inferences about specific cases from the general population.

Label	Count	Average Voltage (mV)					Variation Coefficient
		Mean	Median	Standard Deviation	Min	Max	
N	5050	7.79	8.13	16.62	– 111.02	114.03	2.13
A	738	9.03	10.17	20.23	– 80.98	80.24	2.24
O	2456	6.91	8.75	25.62	– 437.18	182.41	3.71
∼	284	4.75	4.04	36.06	– 211.22	211.98	7.59
Total	8528	7.54	8.35	20.17	– 437.18	211.98	2.68

Table 4.2.: Processed output statistics.

Finally we can visualize the target signal classes (N, A and O) to get a better idea of the structure of the data, which will aid in the preprocessing and feature extraction processes.

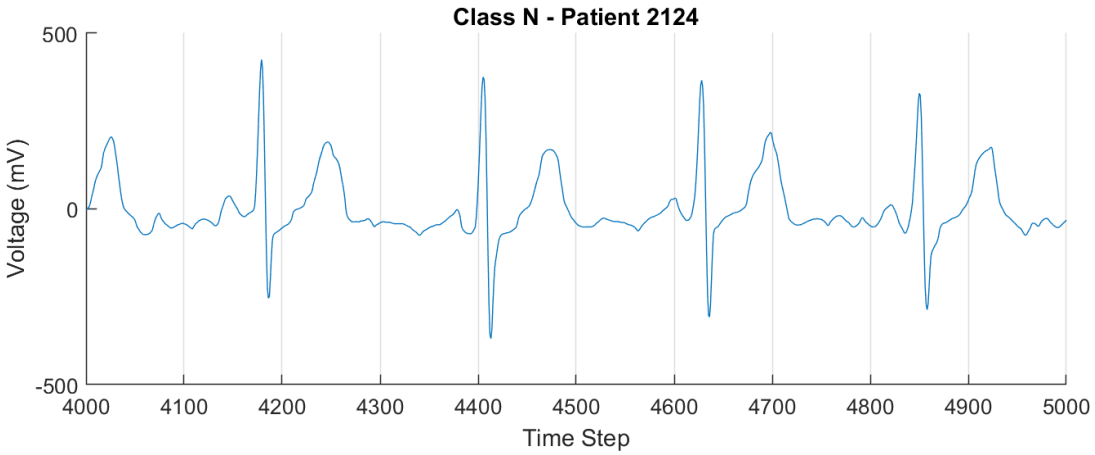


Fig. 4.2.: A normal ECG signal.

In Figure 4.2 a healthy ECG signal can be observed, this can be deduced from several key features. First of all the P, QRS and T segments are clearly present. Secondly, the interval between each QRS complex is adequately consistent. Furthermore, there is no depression indicated in the ST interval and finally the positive peak potential within the QRS complex is greater than the negative peak.

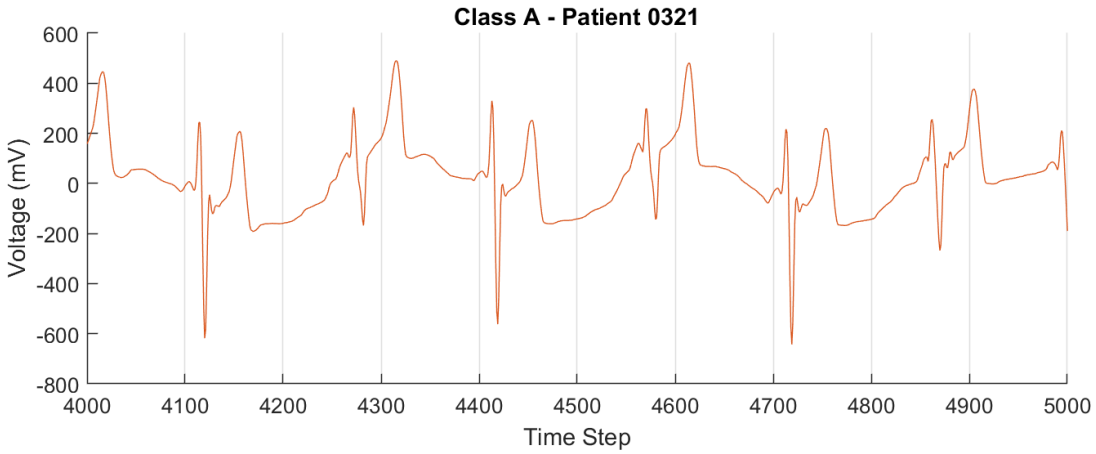


Fig. 4.3.: An ECG signal containing the Afib anomaly.

The ECG trace given by Figure 4.3 shows a very different progression from the previous signal. The most obvious clue for the Afib diagnosis is the abnormal P segment, instead of a localized discharge that diminishes before the QRS complex it keeps rising until the main cardiac action because the electric current propagates erratically through the atria.

As a result less charge passes through the AV node into the ventricular myocardium, causing a diminished QRS complex with a lower maximum positive potential than the T wave on the cardiogram.

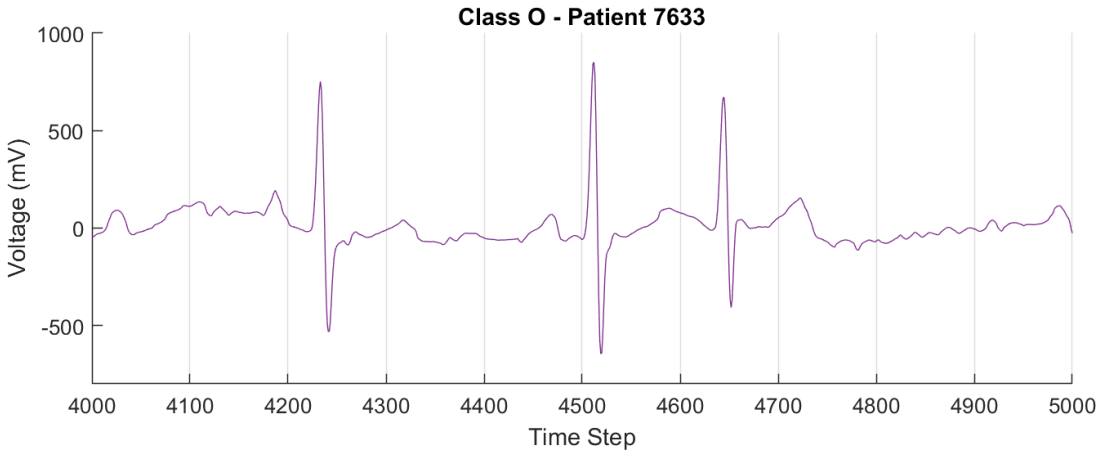


Fig. 4.4.: An ECG signal containing an anomaly other than Afib.

Finally we observe in Figure 4.4 an electrocardiogram trace from the class of all other anomalies. In this particular case the heart rate is very inconsistent and near the end of the observed segment there is an abnormal delay towards the next heartbeat.

From our statistical analysis we have observed that there is quite a large range of signal lengths. Since the derivatives for recurrent neural networks can get quite long for large samples, there is a risk for a diminishing or exploding gradient during long recurrent training. In this case it is better to use the process of *data segmentation* to split large samples up into multiple smaller samples, this not only increases our sample count but is also a requirement for proper classification if no convolution is used [40].

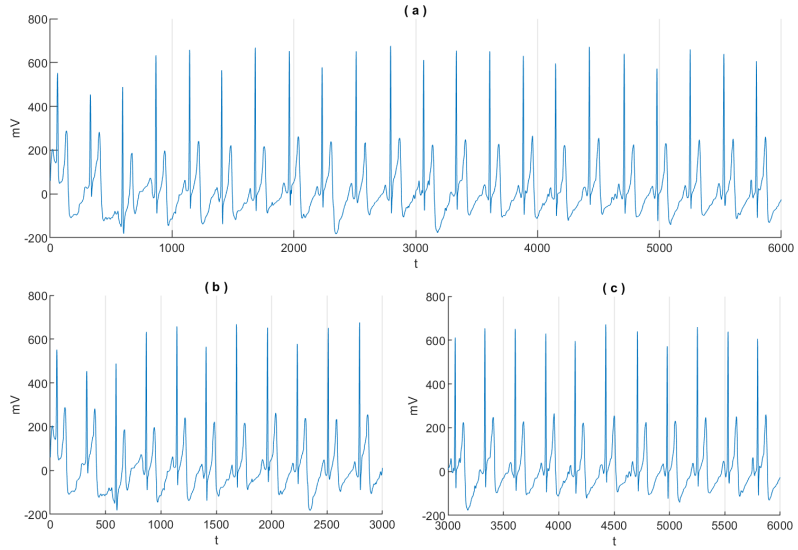


Fig. 4.5.: Partitioning signal *a* of length 6000 into *b* and *c*.

In this case we will choose to resample each unique signal to a standardized format, which is a fraction of the median sample length. This works by splitting the signals up into as many chunks as possible equal to a fraction of the median length, for example to 3000 frames (10 seconds at 300 Hz). This is shown in Figure 4.5. The data set is now partitioned in a set of normalized samples, each retaining the label of the original signal.

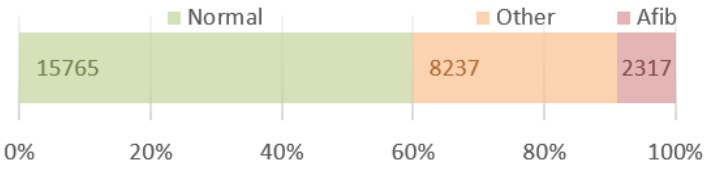


Fig. 4.6.: Distribution of the segmented data set.

A final aspect of our data set we have to analyse is the distribution of the data, which we have plotted in the Figure 4.6. As can be seen the data is quite unbalanced, which will cause the model to bias it's loss function towards the majority class. Several options are available to avoid training bias, however due to the critical nature of the data only oversampling in the training set is a viable *data augmentation* option [41].

In addition to resampling the data set, we have decided to remove the noisy signals from the data. In a 2018 revision of the labels of the original data set these examples were considered too unreliable to present a reasonable diagnosis criterion for an expert. Finally, after processing all the data we end up with the following data count in table 4.3.

Subset	Label	Count	Samples	Duration (h)
Regular	N	15 765	47 295 000	43.79
Divergent	A	2 317	6 951 000	6.44
	O	8 237	24 711 000	22.88
		26 319	78 957 000	73.11

Table 4.3.: Final dataset sample count after processing.

4.2 Model Architecture

To run our preliminary attempt to classify the data we first need to define a complete network architecture from the input handler to the output regressor. For this purpose we will be using MATLAB's Deep Learning Toolbox (DLT), which supports the vectorization technique we have defined in the previous chapter. In addition DLT supports multi-threading and GPU training by using CUDA vector instructions on the SIMD architecture which will greatly speed up the training process.

Our architectural goal consists of a series of components connected in a directed graph [42], using everything we have studied so far as building blocks. Each basic component is an abstraction of the theory we have previously covered. The components that we can now define are as follows.

· **Sequence Input Buffers**

The first step in the architecture is to take the set of input data and feed them forward through the network in a chronological sequence. This input is done by buffering a set interval of data elements equal to the width of the moving window over the data, which we call the mini-batch size for discrete sets.

We want the minibatch size to be small enough to prevent overfitting but also not too large, which causes the computational complexity to soar. In this case our data consists of n points per unique sample, as such we want a mini-batch that is divisible by this amount to make optimal use of our data. For our initial parameters we may set the input buffering size to $b = \frac{n}{k}$ points, which will give us k mini-batches for each of our samples in our data set.

· LSTM – RNN Layers

Next, the input buffer feeds the mini-batch segmented data through the LSTM-RNN consisting of input width n and $h \geq 1$ hidden units. The LSTM is structured as described in the previous chapter, with the addition of hidden state transfer across hidden units displayed Figure 4.6.

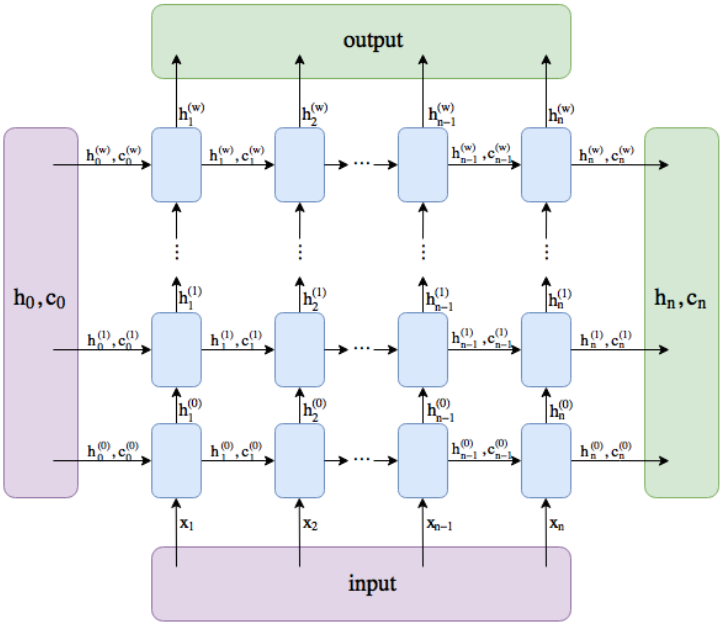


Fig. 4.7.: Fully unrolled LSTM layer with n^2 states.

These lateral state transfers use the same weight as the recurrent weights because they are technically vectorizations of the data lanes, as such the learning algorithm doesn't need to be modified. Rather, it is the vectorized data that gets transformed with each additional abstraction layer. The higher the dimensionality of the input data, the more abstractions are possible and the more informative features we may end up with without overfitting.

This results in $4h(n + h) = 4nh + 4h^2$ trainable weights, $4nh$ weights for the inputs n and feature outputs h growing linearly while the $4h^2$ hidden weights grow exponentially as also seen in Figure 4.6 when unrolling.

In our case we implement a LSTM with a n-to-1 paradigm, where only the last time step outputs a value, this value is a vector of size h as h represents the bus size that is used within the architecture. By doing this each bus lane reduces the dimensionality of it's input to a single dimension feature, giving us a total of n distinct features.

· Fully Connected Layers

Within the network the LSTM extracts the temporal features from the row vector and transforms the input down to a linearly separable column vector, now we need to reduce the column vector down to a desired set of output features.

Each output node of the final FC layer corresponds to an output class. For deep neural networks generally ReLU activations are used for the FC, which provide favourable properties such as not being as susceptible to vanishing gradient and being less computationally complex. However ReLU activation is not normalized and not differentiable at every point, so an additional layer is required.

· Soft-Max Layers

The output from the fully connected layer cannot be used straight away, first we need to normalize the outputs to a probability distribution over the target classes and provide a fully differentiable mapping. For this we use a soft-max layer, which uses the probability distribution $\text{softmax} = \frac{e^{h_i}}{\sum_{k=1}^{n} e^{h_k}}$ over the n different class label outputs.

· Majority Confidence Classifiers

Finally the resulting class label is selected by choosing the class with the highest confidence probability by using a max function over the n results of the soft-max layer. This lets us transform a numeric regressor output column vector from the soft-max layer to a sparse binary column vector which represents a single categorical classification output value.

For backpropagation training we use the target class $\mathbf{y}$ in conjunction with the output probabilities $\hat{\mathbf{y}}$, after all for *cross-entropy loss* it holds for the loss function that $L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{n} y_i \ln \hat{y}_i$. As such, backpropagation over one-hot column vectors only propagates towards the target class of the example – letting the network train each output pathway from the softmax layer towards a unique categorical class.

Furthermore, for a one-hot column vector the loss is calculated over appropriate softmax channel, all other column entries result in a zero log multiplication. This lets the network train towards the specific target class for each sample. Since we make use of a softmax distribution in the previous layer $\hat{y}$ is guaranteed to be non-zero and thus we can always backpropagate using cross-entropy.

4.3 Preliminary Model

In the previous sections we have laid out the structure of our data and the components that we can use to create an architecture to train our model. In this section we will focus to learn the data with the simplest possible LSTM architecture for categorical classification. Doing so, we can detect interesting features and resolve problems that might appear before tackling more computationally complex architectures [43].

The abstraction of our initial architecture is given in Figure 4.7, which results in the following matrix operations:

· **Input** – Input sequence data of n length column vectors and t data points as matrix: $\mathbb{R}^n \times \mathbb{R}^t$

· **Transform** – Transform input length t to h feature matrix using LSTM: $\mathbb{R}^n \times \mathbb{R}^t \rightarrow \mathbb{R}^n \times \mathbb{R}^h$

· **Transform** – Transform h temporal features to f feature matrix using FC: $\mathbb{R}^n \times \mathbb{R}^h \rightarrow \mathbb{R}^n \times \mathbb{R}^f$

· **Operation** – Map feature matrix elements using softmax function: $f : X \rightarrow Y, \quad \forall x_{i,j} \in \mathbb{R}^n \times \mathbb{R}^f$

· **Operation** – Map decimal feature matrix elements to sparse binary representation using max function on column vectors: $f : X_{10} \rightarrow Y_2, \quad \forall x_{i,j} \in \mathbb{R}^n \times \mathbb{R}^f$ where $f(x_{i,j}) = 1 \leftrightarrow \max(\mathbf{x}_j) = x_{i,j} \quad \vee \quad f(x_{i,j}) = 0 \leftrightarrow \max(\mathbf{x}_j) \neq x_{i,j}$.

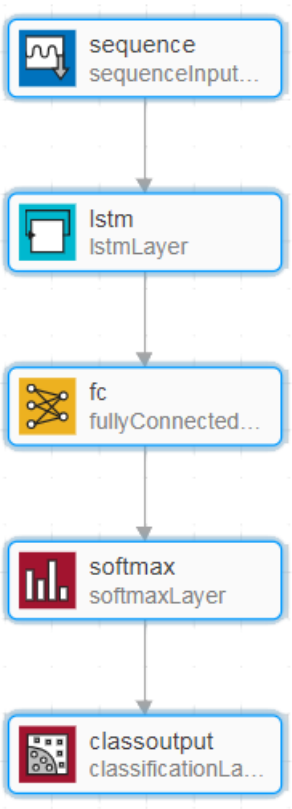


Fig. 4.8.: The simple architecture.

Using this schematic we can explore the performance measure on our model. For our initial run we will specify a mini-batch size of one hundred samples and a *segmentation ratio* of $\frac{1}{3}$ of the mean, each training session consisting of a maximum of ten epochs. To prevent overfitting early stopping [44] will be used when the value of the loss function falls below the minimum threshold of the local maximum on n consecutive validation passes. Our objective is to generate the following two models:

· **I – Composite Anomaly Detection Set** which maps the label set L to normal set $N = \{x \in L,\ L(x) = N\}$ and divergent set $D = \{x \in L,\ L(x) \neq N\}$ to learn the binary decision problem based on the composite of all anomalous feature labels.

· **II – Specific Anomaly Detection Set** which maps the label set L in a 3-way categorical split with normal set $N = \{x \in L,\ L(x) = N\}$, target set A = $N = \{x \in L,\ L(x) = A\}$ and control set $O = \{x \in L,\ L(x) \notin \{N, A\}\}$ to learn diagnosis criteria based on mutual exclusivity of disjoint label sets.

For the initial training specification we produce a cross validation sample using a partitioning ratio of 0.8, 0.1 and 0.1 for the training, testing and validation set respectively. The resulting training set contains 21 055 samples, the testing and validation sets both contain 2 632 uniquely sampled entries from our data set. After augmenting the training set by oversampling we end up with a total of 25 242 samples, 12 621 for each individual class for model I and 37 764 samples, 12 588 for each class with regards to model II.

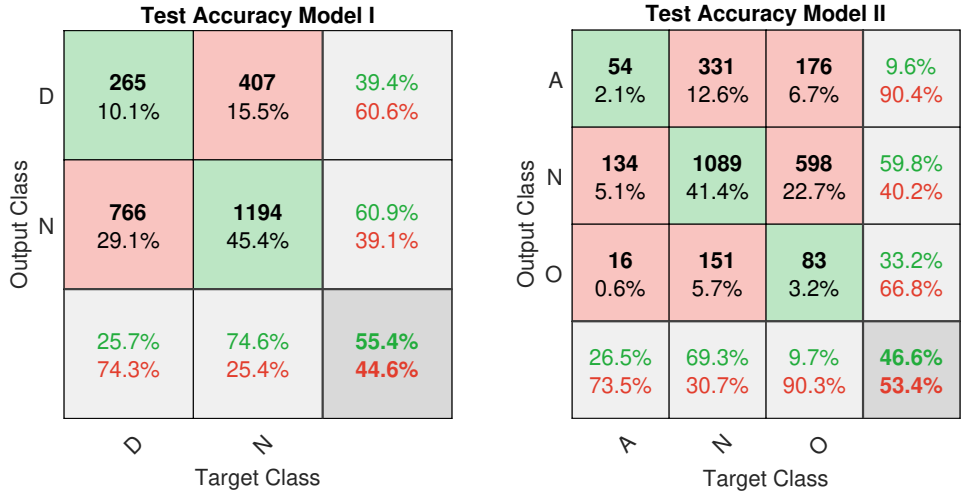


Fig. 4.9.: Test set confusion matrix for the baseline model.

The results of the initial training of each model with respect to the validation set has been included in the appendix section C. The model classification performance against the test set is given in Figure 4.9 on the previous page.

From the training results we find that both models converged based on the early stopping criteria, however both models failed to converge on a high degree of accuracy. Clearly, a single layer LSTM cannot find a proper fit on the raw signal data. Both models show great bias towards the normal class, but fail to identify the exterior classes. To improve classification performance we will use two approaches: utilize spectral and time-frequency analysis to engineer features with better convergence properties and modify the network to contain more abstraction levels to find a better fit on the data.

4.4 Signal Processing

Modelling abstract features from highly variate sensor data is challenging for a recurrent neural network, without convolutional layers to select features it will likely get stuck in a local minima and then overfit towards a poor model [43]. In our initial training this was the case, the model quickly found a local minima during it's first training epoch. Without these convolutional layers we need to guide the model in the right direction.

Such abstract features, based on mathematical frameworks instead of domain knowledge, can be derived from the data through a process called feature extraction. The resulting augmented feature sets form an intermediate form, which we can use in the training and classification process while retaining the original labels.

First we will resample the full raw signals to make them useable for spectral and time-frequency analysis. We do this by applying the *Savitzky-Golay filter* on the raw signals. The S-G filter uses a convolutional method by fitting same-order polynomials on the data points of adjacent sets using linear least squares. For a given central hinge point Y_n it follows:

$$Y_n = \sum_{i=\frac{1-m}{2}}^{\frac{m-1}{2}} C_i \, y_{n+i} = C_{\frac{1-m}{2}} y_{n-\frac{1-m}{2}} + \ldots + C_0 y_n + \ldots + C_{\frac{m-1}{2}} y_{n+\frac{m-1}{2}}$$

Where $m \geq 1,\ m_{\text{uneven}}$ is the segment width, Y_n is the n^{th} hinge point and C_i is the convolution coefficient associated to each data point y. Each coefficient within the segment is an element of the k^{th} order convolution kernel:

$$\mathbf{C} = (\mathbf{S^T S})^{-1}\mathbf{S^T},\ \mathbf{S} = \begin{pmatrix} y_{n-i}^0 & \cdots & y_{n-i}^k \\ \vdots & \ddots & \vdots \\ y_n^0 & \cdots & y_n^k \\ \vdots & \ddots & \vdots \\ y_{n+i}^0 & \cdots & y_{n+i}^k \end{pmatrix}$$

After applying the Savitzky-Golay analytic filtering function to map mean segmented samples on the entire data set using a third order kernel we get the resampling result as shown in Figure 4.10.

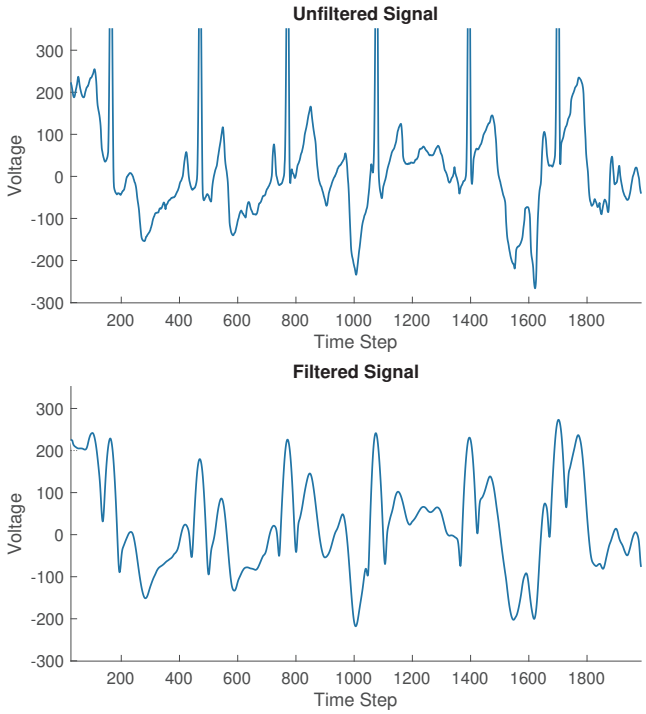


Fig. 4.10.: Sample data before and after resampling.

From the resampled signals we can start to extract new features. The first feature we would like to look at is the local entropy measure, since a higher degree of local signal entropy compared to the model fit is a primary indicator of an anomaly [45]. Using signal processing equations we can extract the *instantaneous spectral entropy* from a time series with a constant frequency using the local time-frequency power spectrum.

First we calculate the time-frequency power density of continuous signal m for each t and over all frequency channels f using the spectral density S:

$$P(t,m) = \frac{S(t,m)}{\sum_f S(t,f)}$$

From the power density of each t we can calculate the entropy as usual.

$$H(t) = -\sum_{m=1}^{N} P(t,m)\log_2 P(t,m)$$

Since we are dealing with a non-deterministic source signal, we can apply non-stationary analysis [46]. Since we know the signal was recorded at $f = 300Hz$ we can derive the *instantaneous frequency* of every t from the Fourier signal transform m. First we need to determine the analytic representation of our real-valued $m : u(\tau)$ using Hilbert transform, imparting a $\frac{\pi}{2}$ phase shift on negative frequency Fourier components and $\frac{\pi}{2}$ on positive frequency components.

$$H(u)(t) = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{u(\tau)}{t-\tau}\, d\tau$$

From there finding the instantaneous frequency for value t is straight forward:

$$f_i(t) = \frac{1}{2\pi}\frac{du(\psi)}{dt}$$

With these new feature measures we can generate feature sets with reduced time-complexity and better convergence properties on our model, reducing samples size from in excess of 4096 bytes to 512 bytes at the cost of transformation overhead. One problem with using the composite of different feature sets is that they use different measures, which can be seen in table 4.4 below, averaging over all the data in our set.

Feature	**Output Value**					**Variation Coefficient**
	Mean	Median	Standard Deviation	Min	Max	
$H(t)$	0.57	0.57	0.05	0.45	0.77	0.09
$f_i(t)$	3.36	2.97	1.51	1.25	23.55	0.45

Table 4.4.: Feature set statistics.

To regularize our feature vectors we need to apply statistical *normalization* [47], which makes the data invariant to the base unit.

$$\forall y_i \in \mathbf{y},\ z_i = \frac{y_i - \bar{\mathbf{y}}}{\sigma}$$

This gives us the normalized z-statistics for each vector element in the signal over our total feature matrix in table 4.5 below.

Feature	**z Value**					**Variation Coefficient**
	Mean	Median	Standard Deviation	Min	Max	
$H(t)$	0.00	−0.01	1.00	−2.44	4.36	∼
$f_i(t)$	0.00	−0.25	1.00	−1.40	13.40	∼

Table 4.5.: Normalized feature set statistics.

Finally to account for the disproportionate validation set loss over the training loss we modify our training rule to add the L_2 *regularization* term $\lambda \sum_{i=0}^{n} w_i^2$ to each weight modification iteration. This adds a squared penalty to the loss function, scaling greatly with the scalar weight of the individual weights, reducing the likelihood of overfitting [48].

Using a mean segmentation ratio plus L_2 regularized training on the feature set we find a significant improvement for both models as shown in Figure 4.11, the training graphs for each model can be found in appendix C.

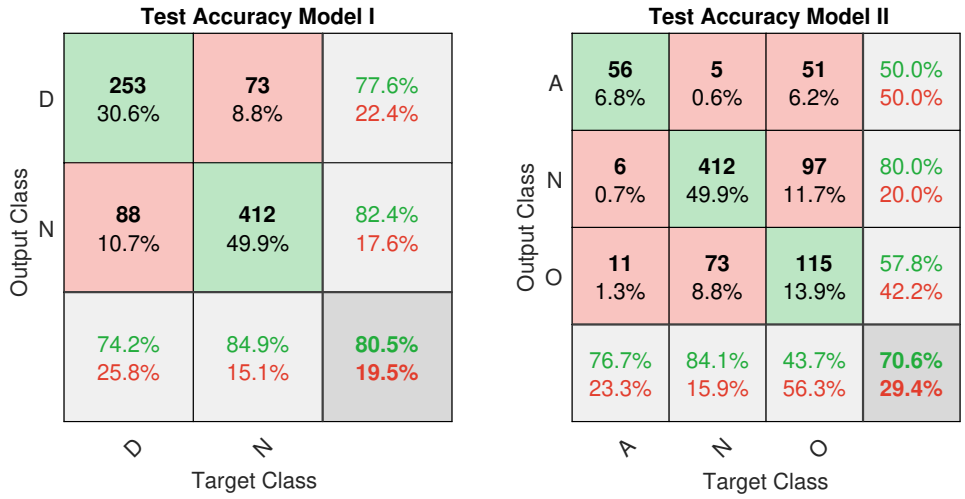


Fig. 4.11.: Test set confusion matrix after signal processing.

4.5 Deep Learning Models

So far we have studied an architecture with a single LSTM layer followed by a single fully connected layer as our trainable model. Even though this can be rather quickly trained and generalizes well, the feature complexity that can be effectively modelled is limited.

To further improve the classification accuracy we need to expand the scope of the applied deep learning paradigm from the single LSTM layer to the broader architecture. We will discuss two such ways to apply a deep learning architecture, which is represented as a tree topology in the form of a directed acyclic graph (DAG) of learning layers [49].

· Width Topology

The abstraction of our $n = 2$ width architecture is given in Figure 4.12. Within this architecture we apply a paradigm called width learning, by learning the same feature set on n parallel layers. Each layer has it's own hyperparameters. for example in our two-width network we use one LSTM with 128 hidden units and one LSTM with 64 hidden units, each initialized with semi-random weights.

At the end of each of these layers they are normalized and weighted to a specific feature count. The weighted results are passed through an activation function and added together. The result gives a weighted output of smaller layers that generalize better and bigger layers that can achieve a greater degree of fitting.

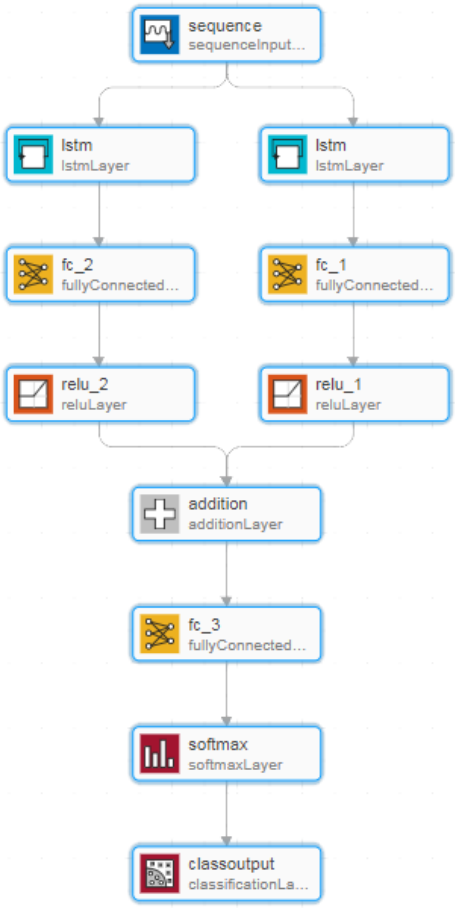


Fig. 4.12.: Width based learning topology.

Each LSTM in this DAG has its own specific temporal feature encoding, which will specifically help us find outliers in the form of anomalies – as with each additional encoding, the conjugate from each encoding will exacerbate the anomalous features. As a result a model of this complexity benefits from a smaller learning rate, each changes in weights has a greater effect on the overall network. In addition, by keeping our weights small by using L_2 and regularization a low learning rate we can prevent overfitting.

After training this new network architecture of width $n = 2$ we find a small improvement in overall accuracy compared to a $n = 1$ width network, as shown in Figure 4.13. Most notably, the false-negative rate on anomalous features has gone down significantly.

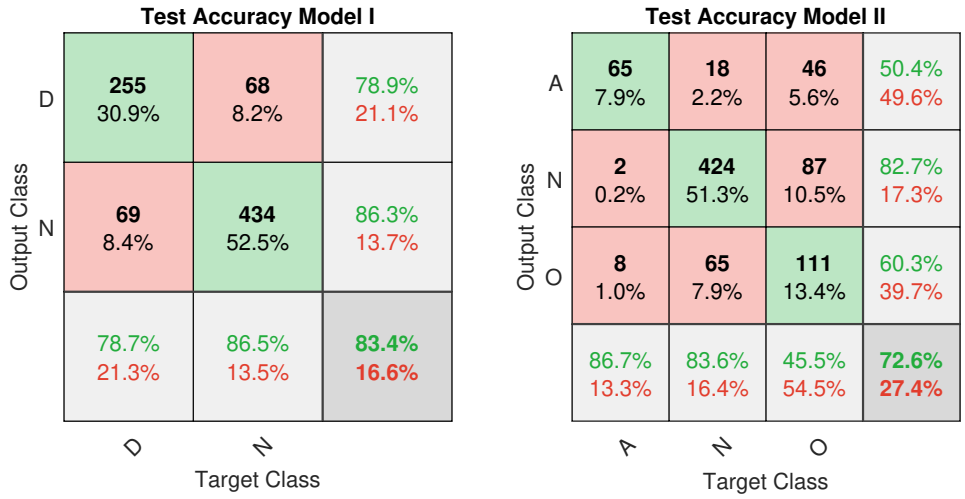


Fig. 4.13.: Test set confusion matrix using width learning.

· Depth Topology

Another way we can increase the complexity of our model is by stacking multiple LSTM layers in a series. One of the problems of temporally encoding very large series of data is that it is very prone to under- and overfitting based on the feature count. Reducing a large amount of temporal features using fully connected layers doesn't preserve the temporal properties in the reduction phase.

By using a m depth LSTM we can sample temporal features down through multiple iterations. The $m - 1$ top layers output sequence data reduced to the feature count, each reducing the amount of encoded features in downward cascading sequence.

The final layer takes the encoded input sequence that is output by the penultimate layer and then encodes it down to a single output feature for each hidden unit. Even though this output has a lower feature count, each abstract feature has been selected through the conjugation of multiple *temporal encodings* of the input features. This allows us to perform our initial encodings on high temporal feature counts without overfitting on the end result.

Our $m = 2$ depth architecture shown in Figure 4.14 uses a top layer LSTM with 128 hidden units outputting a sequence and the final layer LSTM outputs a single feature for each of the 64 hidden units.

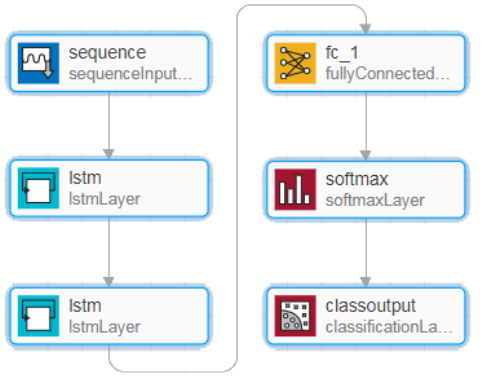


Fig. 4.14.: Depth based learning topology.

Training this model gives us the confusion matrix results on the test set as displayed in Figure 4.15. Once again overall accuracy is slightly increased, the depth learning method seems to decrease false-negatives over the categorical split to a greater degree whereas it is less effective on the binary split for anomaly detection.

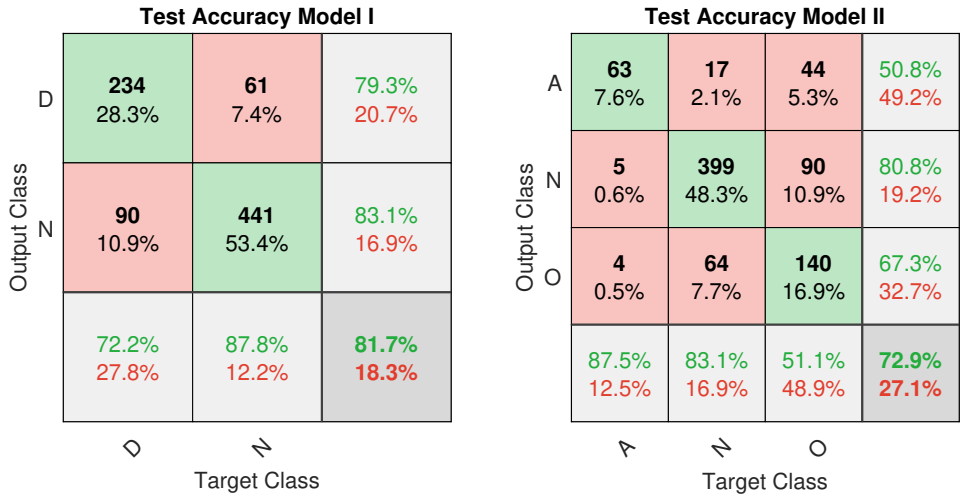


Fig. 4.15.: Test set confusion matrix using depth learning.

· Hybrid Topology

Finally, we can combine both width and depth learning in a single architecture, as given below in Figure 4.16 using $n = 3$, $m = 2$. This combines both abstraction through using different LSTM encoding resolutions laterally as well as downsampling to smaller temporal feature sets to improve generalization.

Fig. 4.16.: Hybrid learning topology.

With out final model we achieve the following metrics in Figure 4.17.

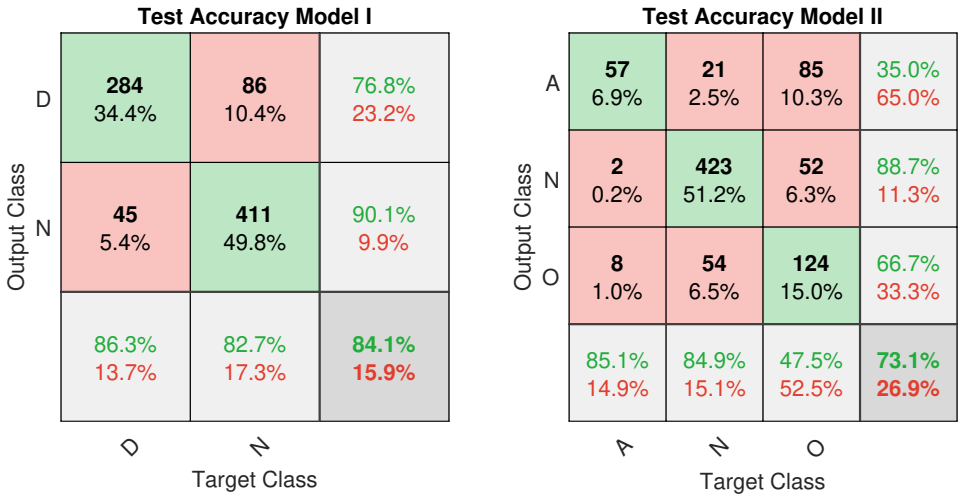


Fig. 4.17.: Test set confusion matrix using hybrid learning.

To determine the significance found between the initial simple model and the hybrid model for the augmented anomaly feature set that was deterministically derived from our ECG data set, we will be performing the McNemar's Test ($\alpha = 0.05$) on the contingency table of the predictions of these models on the test set ($n = 826$).

Assume that T_1 is the prediction result from the simple model and T_2 is the prediction result from the proposed hybrid model. Assume for our hypothesis $H_0 : T_1 = T_2$, $H_a : T_1 \neq T_2$.

Model	T1 True		T1 False	
	T2 True	T2 False	T2 True	T2 False
I	619	52	78	71
II	405	51	31	339

Table 4.6.: Test Set Cross Table.

From the table we can calculate the McNemar Chi-Square test statistic:

$$\chi^2_{\mathrm{I}} = \frac{(n_{\mathrm{T,F}} - n_{\mathrm{F,T}})^2}{n_{\mathrm{T,F}} + n_{\mathrm{F,T}}} = \frac{(52-78)^2}{52+78} = 5\frac{1}{5}$$

$$\chi^2_{\mathrm{II}} = \frac{(n_{\mathrm{T,F}} - n_{\mathrm{F,T}})^2}{n_{\mathrm{T,F}} + n_{\mathrm{F,T}}} = \frac{(51-31)^2}{51+31} = 4\frac{36}{41}$$

And we know for the degrees of freedom using T_1 and T_2 as parameters it holds $df = N - 1 = 1$, so then we can calculate the P value from the Chi-Square statistic. We will use cumulative distribution function as below to evaluate our χ^2 values in R.

$$P(\chi^2, df) = \int_0^{\chi^2} \left(\frac{x^{\frac{df-2}{2}} e^{-\frac{x}{2}}}{2^{\frac{df}{2}} \Gamma(\frac{df}{2})} \right) dx,\ \Gamma(df) = \int_0^{\infty} (x^{df-1} e^{-x}) dx$$

Evaluating the P value from the χ^2 values of each of the models gives us a residual probability of $1 - P(\chi^2_{\mathrm{I}}) = 0.0226$ and $1 - P(\chi^2_{\mathrm{II}}) = 0.0272$, which are both well below the significance level of $\alpha = 0.05$. As such we can reject H_0 and can infer that it is over two standard deviations improbable that the differences between T_1 and T_2 are due to chance, thus the improvements of our hybrid model are statistically significant.

Conclusion

5

The 2017 Computing in Cardiology (CinC) data set comprises of single dimensional time series data of electrocardiogram recordings, mapping band-passed voltage readings to static frequency time steps. The voltage range is highly variate between QRS segments while lowly variate outside of those segments whilst also retaining backward dependencies from previous segments. Those properties make it difficult to model and to make inferences over such data using traditional mathematical techniques.

In this research project we have employed a subset of deep neural networks called recurrent neural networks, with a specific focus on long short-term memory cells which have advantageous properties when regarding data where the derivative is irregular between segments. These techniques use a statistical learning based approach. Instead of modelling the signal using static analysis, the model is optimized using stochastic gradient descent with every example from our data set we present it with.

We have employed such models to perform anomaly detection on two target groups, Model I being binary anomaly detection between the normal group and all other non-normal groups. Model II trains on a three-way anomaly detection problem between a baseline group, a target group and a residual group. Initial training on the raw data gave us a poor maximum accuracy threshold compared to using a feature selection approach, see Figure 5.1.

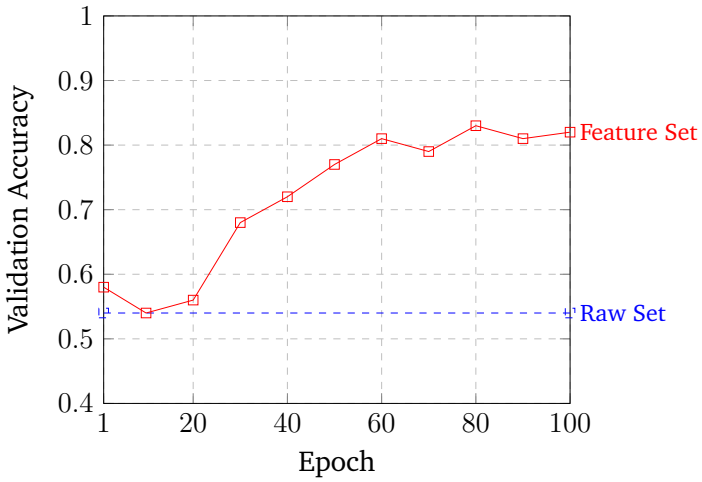


Fig. 5.1.: Sparse model I training plot for the raw data set.

By filtering the raw data and then performing spectral and time-frequency analysis on our signal to extract informative features, we managed to make the data much more suitable for temporal encoding by our LSTM. From there we generated several models, of which the performance is compiled in Figure 5.2 below.

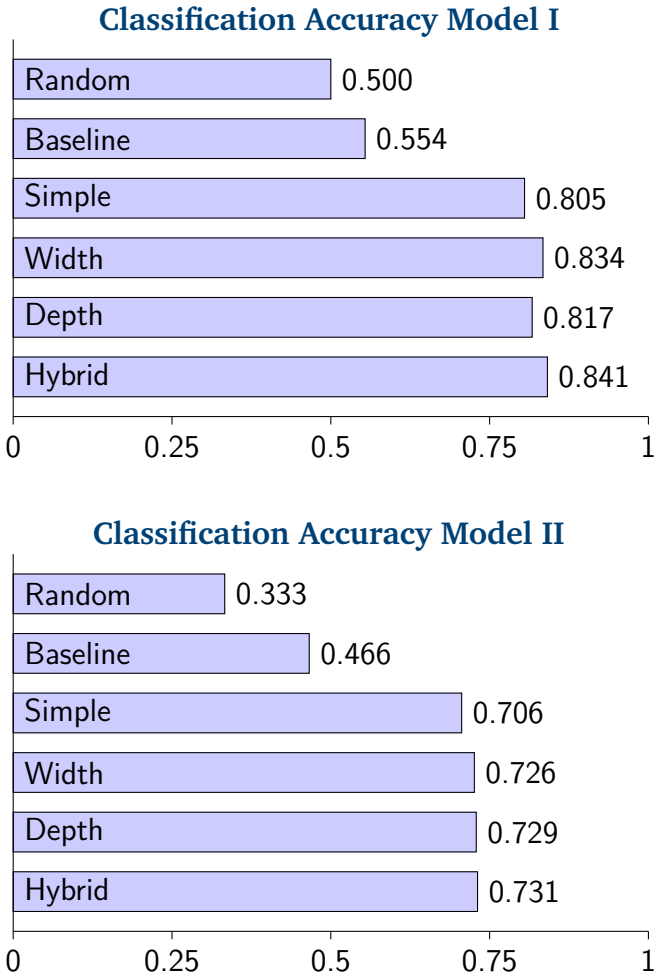


Fig. 5.2.: Final Model Performance Graph.

From our quantitative analysis and significance test we can confirm that for a binary anomaly detection, we can achieve 84.1 percent accuracy on the test data. For the more complex problem of categorical anomaly detection the final model achieves an accuracy of 73.1 percent on the test set.

Concluding, LSTMs provide a suitable technique to extract temporal features from processed ECG data. However, the measure of performance is highly dependant on the quality and informativeness of the feature extraction techniques used. Given a good feature representation, performance can be further enhanced by broadening and deepening the network.

Future Work

6

From our research we have studied and implemented a subset of the recurrent neural network class called the long short-term memory network. As a result we have found an optimized model to perform anomaly detection of a desired subset class on static ECG data recordings. However, the method we have used is still limited and can be improved in various novel ways and as a result form interesting topics for further research.

· **Real-Time Analytics** is the first topic of interest regarding our original research topic. In our current implementation an ECG recording needs to be stored on a file and is then read into the memory, from the memory the data is written to the input sequence buffer in chunks equal to the mini-batch size before being processed by the network. What we would want is for the device driver to write it's data straight to our sequence input buffer.

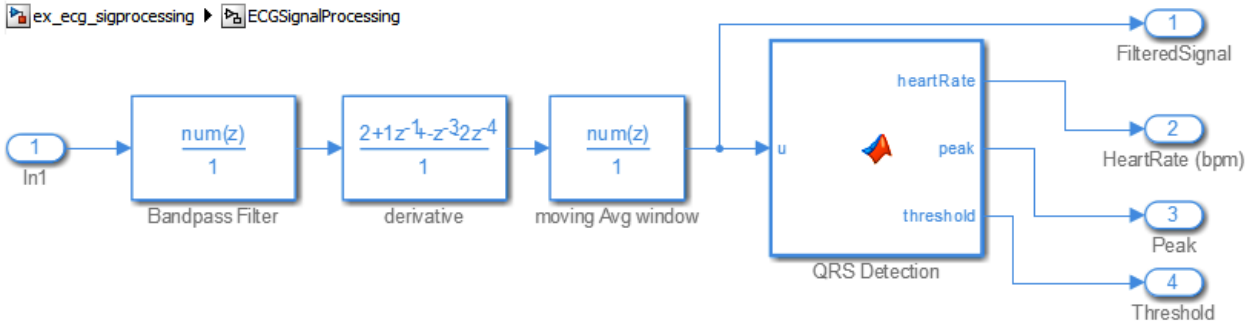


Fig. 6.1.: Signal Processing Schematic

Real-time classification introduces a broad set of challenges, which is a forth-going subject of study [50]. In addition to extracting the raw data it is common practice to produce augmented features, such as averages and peaks, which provide additional metrics. These metrics can aid in monitoring exceptions and can be used as part of the classification process, in Figure 6.1 a sample MATLAB implementation is given of direct signal processing.

Current challenges within this topic relate to automated normalization of source data, minimizing the latency between the source signal and the processed data, handling signal drops, adjustments for signal quality and determining the classification confidence at an arbitrary time frame.

· **Clinical Computing** has different goals and requirements than general purpose computing. As such it is important to cooperate with clinical experts to implement domain knowledge and requirements within a model that can be deployed within a clinical setting [51].

This results into a different goal from standard machine learning, where often the mean accuracy over a data set is to be maximized. When dealing with data-based diagnosis criteria, Type II errors are much more dangerous than Type I errors. In other words: we'd rather let a medical monitoring device be overly sensitive, causing an acceptable amount of false positives, than have it provide a false-negative which can have serious consequences. A domain expert can help shape appropriate weights for each classification to optimize for minimal Type II error count.

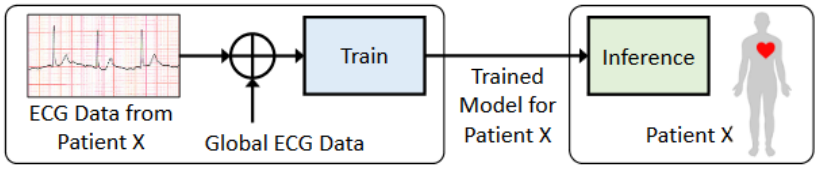


Fig. 6.2.: Augment model with Transfer Learning [52].

Using further domain knowledge can also be incorporated in the model itself by using transfer learning techniques as displayed in Figure 6.2. Since each patient has different baseline biometrics, we can augment the model by introducing expert-classified samples from the patient themselves into the model. This is especially useful for long-term monitoring.

· **Advanced Techniques** can be studied to further improve upon the architecture. One of such state-of-the-art techniques is the Temporal Convolutional Networks (TCN) [53]. We can utilize the max pooling of features from CNNs and use them as input into the temporal encoding functionality of RNNs. The main purpose of this is to reduce overfitting and automate feature extraction by pooling feature abstractions, allowing us to train on data with a higher sampling rate and dimensionality while still being able to generalize.

Another advanced technique is to use ensemble learning [54]. We can use additional biometric signals, such as EEG, to utilize their interaction features as informative data. We train different models on multiple synchronized signals towards the same output labels. By using the weighted consensus of all our models we can reduce the bias and variance in our classifications.

Appendix

A

A - Matlab Code

```
function [XData, YData] = readData()
    % Extract Data Package
    if ~exist('./data', 'dir')
        unzip('./data.zip')
    end
    cd data

    % Extract File Listing and Labels
    YFile = readtable('DATA.csv','ReadVariableNames',false);
    YFile.Properties.VariableNames = {'Filename','Label'};

    % Delete the Noisy Data
    idx = strcmp(YFile.Label,'~');
    YFile(idx,:) = [];

    % Allocate the Output Data
    YData = categorical(YFile.Label);
    XData = cell(height(YFile),1);

    % Read each Data File
    for i = 1:height(YFile)
        Buffer = load([YFile.Filename{i},'.mat']);
        XData{i} = Buffer.val;
    end

    % Return to Root
    cd ..
end
```

Listing A.1: Reading Data (modification of MathWorks implementation)

```
function [XData, YData] = segmentData(XData,YData,len)
    % Allocate the new Buffering Cell
    XBuffer = {};
    YBuffer = {};

    % Segment each Data element
    for i = 1:size(XData,1)
        % Get source Data and Label
```

```
        x = XData{i}(:);
        y = YData(i);

        % Discard Data smaller than len
        if length(x) < len
            continue;
        end

        % Segment Data in Chunks and repeat Labels
        chunks = floor(length(x)/len);
        x = x(1:chunks*len);
        y = repmat(y,[chunks,1]);

        % Buffer the Segmented Data
        XBuffer = [XBuffer; mat2cell(reshape(x,len,chunks).',ones
    (chunks,1))];
        YBuffer = [YBuffer; cellstr(y)];
    end

    % Place Buffers to Output
    XData = XBuffer;
    YData = categorical(YBuffer;
end
```

Listing A.2: Data Segmentation (modification of MathWorks implementation)

```
function [XTrain,XTest,XVal,YTrain,YTest,YVal] = partitionData(
    XData,YData,mode)

    % Modify labels to Binary Split if mode is set
    if nargin > 2 && mode
        YData = renamecats(YData,'A','D');
        YData = mergecats(YData,{'D','O'});
    end

    % Partition the Training Set
    cv = cvpartition(size(XData,1),'HoldOut',0.8);
    idx = cv.test;
    XHold = XData(~idx,:);
    XTrain  = XData(idx,:);
    YHold = YData(~idx,:);
    YTrain  = YData(idx,:);

    % Partition the Test and Validation Set
    cv = cvpartition(size(XHold,1),'HoldOut',0.5);
    idx = cv.test;
    XTest = XHold(~idx,:);
    XVal  = XHold(idx,:);
```

```
    YTest = YHold(~idx,:);
    YVal  = YHold(idx,:);
end
```

Listing A.3: Data Partitioning

```
function [XData,YData] = augmentData(XData,YData)
    % Ordered categorical counts
    stat = [cellstr(unique(YData)), ...
        num2cell(arrayfun(@(x)sum(YData==x),unique(YData)))];
    [max_val,~] = max([stat{:,2}]);

    % Determine the Underrepresentation Count
    for i = 1:size(stat,1)
        stat{i,3} = max_val - stat{i,2};
    end

    % Concatenate input Data and Labels into Composite
    XYData = [cellstr(YData),XData];

    % Iterate over all Categories
    for i = 1:size(stat,1)
        % Generate Sampling Pool for each Category
        idx = strcmp(stat{i,1}, XYData(:,1));
        pool = XYData;
        pool(any(~idx,2),:) = [];

        % Select n Samples from Pool with Replacement
        XYData(end+1:end+stat{i,3},:) = datasample(pool,stat{i
    ,3});
    end

    % Shuffle XYData
    XYData = XYData(randperm(size(XYData,1)),:);

    % Split A back into YTrainA and XTrainA
    XData = XYData(:,2);
    YData = categorical(XYData(:,1));
end
```

Listing A.4: Data Augmentation

```
function [XData] = featData(XData,order,len,freq)
    % Filter data using Savitzky-Golay
    XData = cellfun(@(x)sgolayfilt(x,order,len)',XData,'
    UniformOutput',false);

    % Extract Features x1, x2, ...
```

```
    XData = cellfun(@(x1,x2)[x1;x2], ...
    cellfun(@(x)pentropy(x,freq),XData,'UniformOutput',false),
    ...
    cellfun(@(x)instfreq(x,freq),XData,'UniformOutput',false),
    ...
    'UniformOutput',false);

    % Normalize Feature Set
    XData = cellfun(@(x)(x-mean([XData{:}],2))./std([XData
    {:}],[],2), ...
    XData,'UniformOutput',false);
end
```

Listing A.5: Feature Extraction

B - Final Model Specification

```
DAGNetwork with properties:

  Layers: [21x1 nnet.cnn.layer.Layer]
  Connections: [23x2 table]

    21x1 Layer array with Layers:

     1   'sequence_1'     Sequence Input        Sequence input with 2 dimensions
     2   'lstm_2'         LSTM                  LSTM with 64 hidden units
     3   'lstm_3'         LSTM                  LSTM with 128 hidden units
     4   'lstm_4'         LSTM                  LSTM with 64 hidden units
     5   'fc_5'           Fully Connected       16 fully connected layer
     6   'relu_6'         ReLU                  ReLU
     7   'lstm_7'         LSTM                  LSTM with 32 hidden units
     8   'fc_8'           Fully Connected       16 fully connected layer
     9   'relu_9'         ReLU                  ReLU
    10   'addition_10'    Addition              Element-wise addition of 2 inputs
    11   'relu_11'        ReLU                  ReLU
    12   'lstm_12'        LSTM                  LSTM with 64 hidden units
    13   'lstm_13'        LSTM                  LSTM with 32 hidden units
    14   'fc_14'          Fully Connected       16 fully connected layer
    15   'relu_15'        ReLU                  ReLU
    16   'addition_16'    Addition              Element-wise addition of 2 inputs
    17   'relu_17'        ReLU                  ReLU
    18   'addition_18'    Addition              Element-wise addition of 2 inputs
```

```
19   'fc_19'          Fully Connected       auto fully connected layer
20   'softmax_20'     Softmax               softmax
21   'classoutput_21' Classification Output crossentropyex


23x2 Table with Connections:

   Source          Destination
------------    ----------------

'sequence_1'       'lstm_2'
'sequence_1'       'lstm_3'
'sequence_1'       'lstm_12'
'lstm_2'           'lstm_7'
'lstm_3'           'lstm_4'
'lstm_4'           'fc_5'
'fc_5'             'relu_6'
'relu_6'           'addition_10'
'relu_6'           'addition_16'
'lstm_7'           'fc_8'
'fc_8'             'relu_9'
'relu_9'           'addition_10'
'addition_10'      'relu_11'
'relu_11'          'addition_18'
'lstm_12'          'lstm_13'
'lstm_13'          'fc_14'
'fc_14'            'relu_15'
'relu_15'          'addition_16'
'addition_16'      'relu_17'
'relu_17'          'addition_18'
'addition_18'      'fc_19'
'fc_19'            'softmax_20'
'softmax_20'       'classoutput_21'
```

C - Training Plots

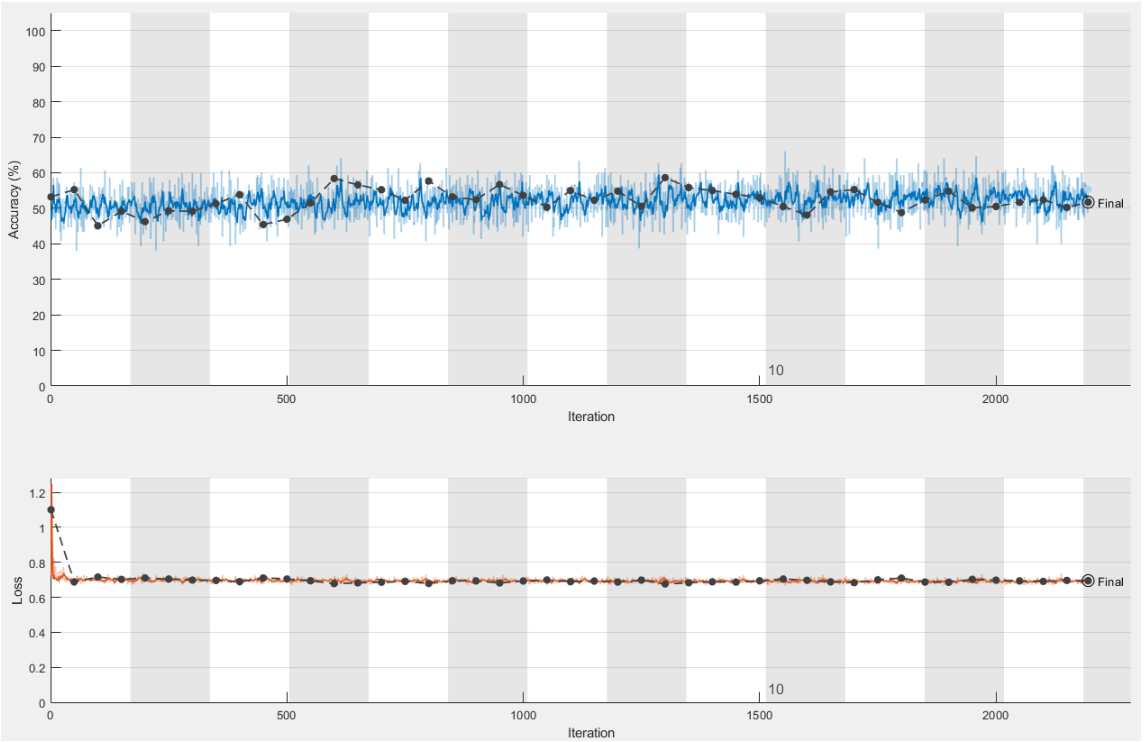


Fig. A.1.: Binary Raw Data Training

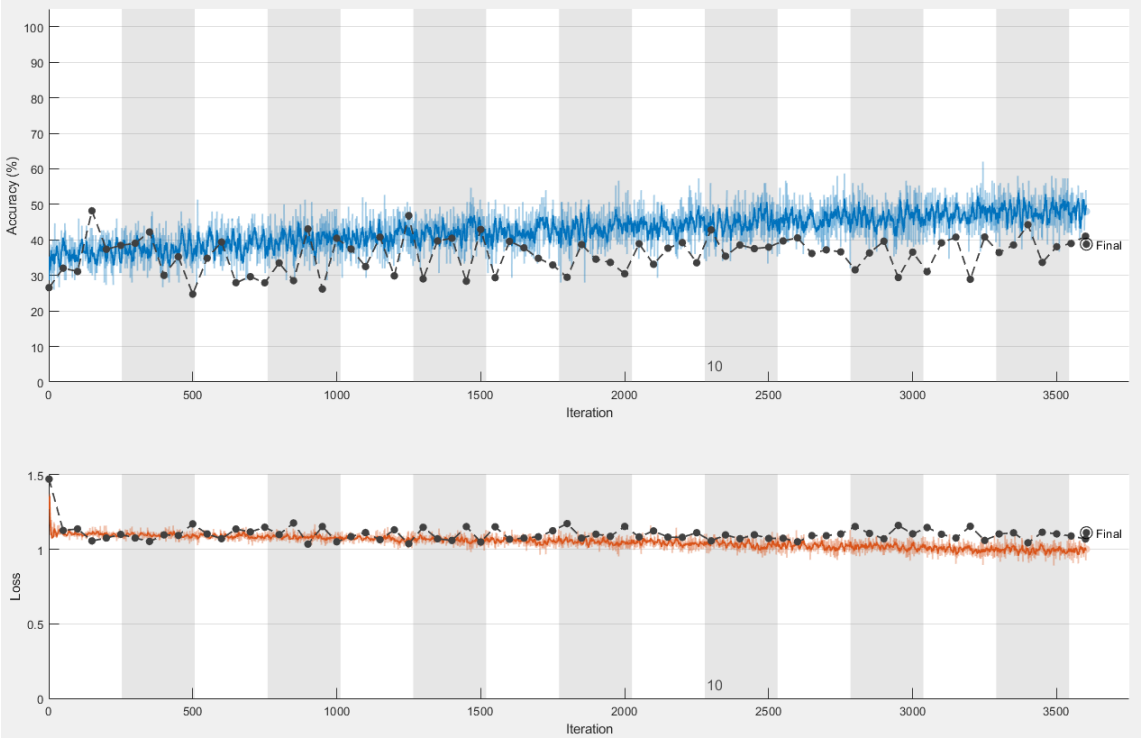


Fig. A.2.: Categorical Raw Data Training

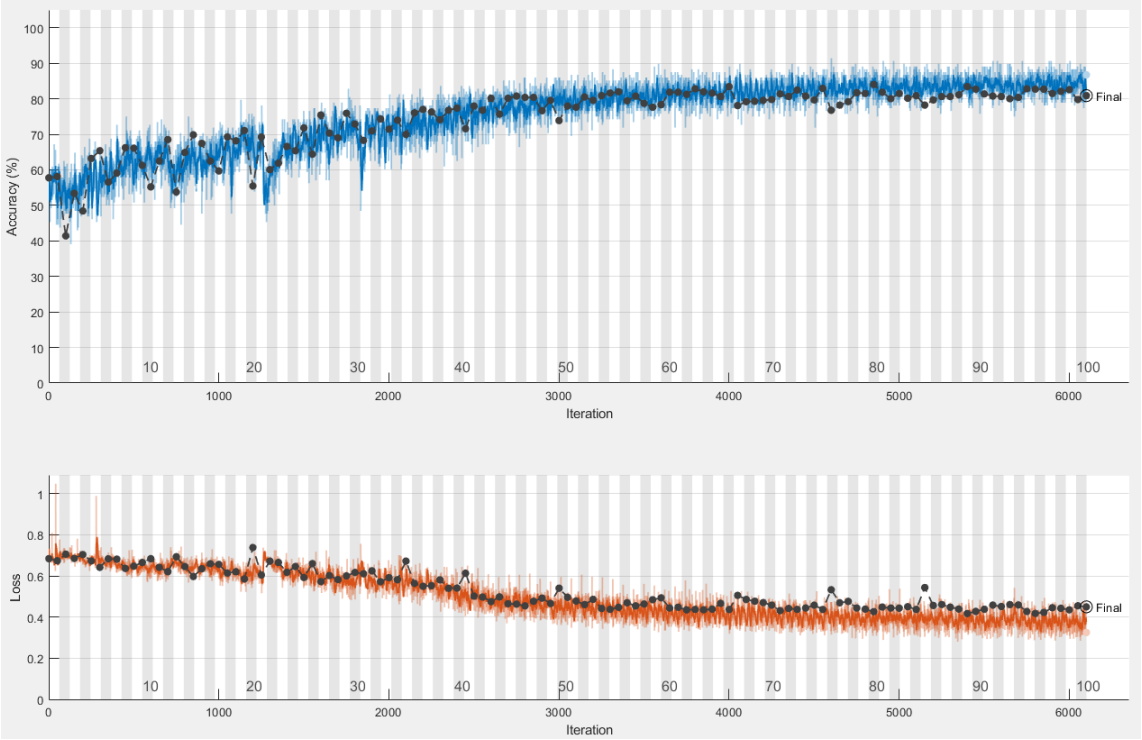


Fig. A.3.: Binary Simple Model Training

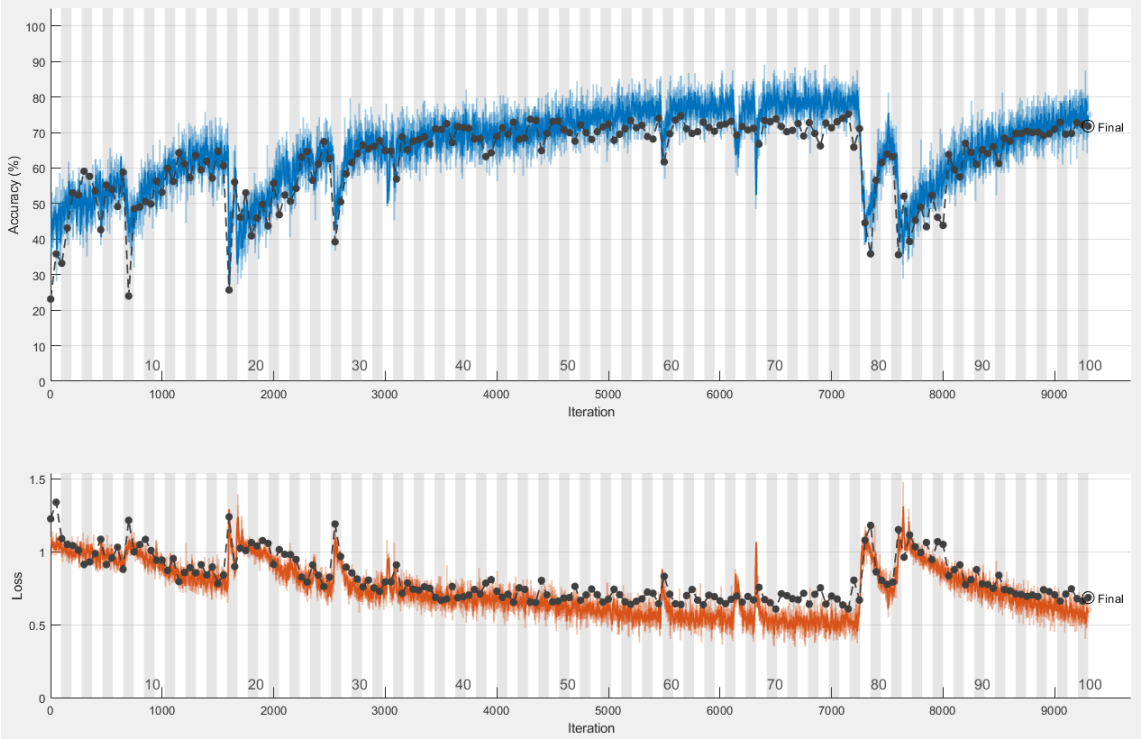


Fig. A.4.: Categorical Simple Model Training

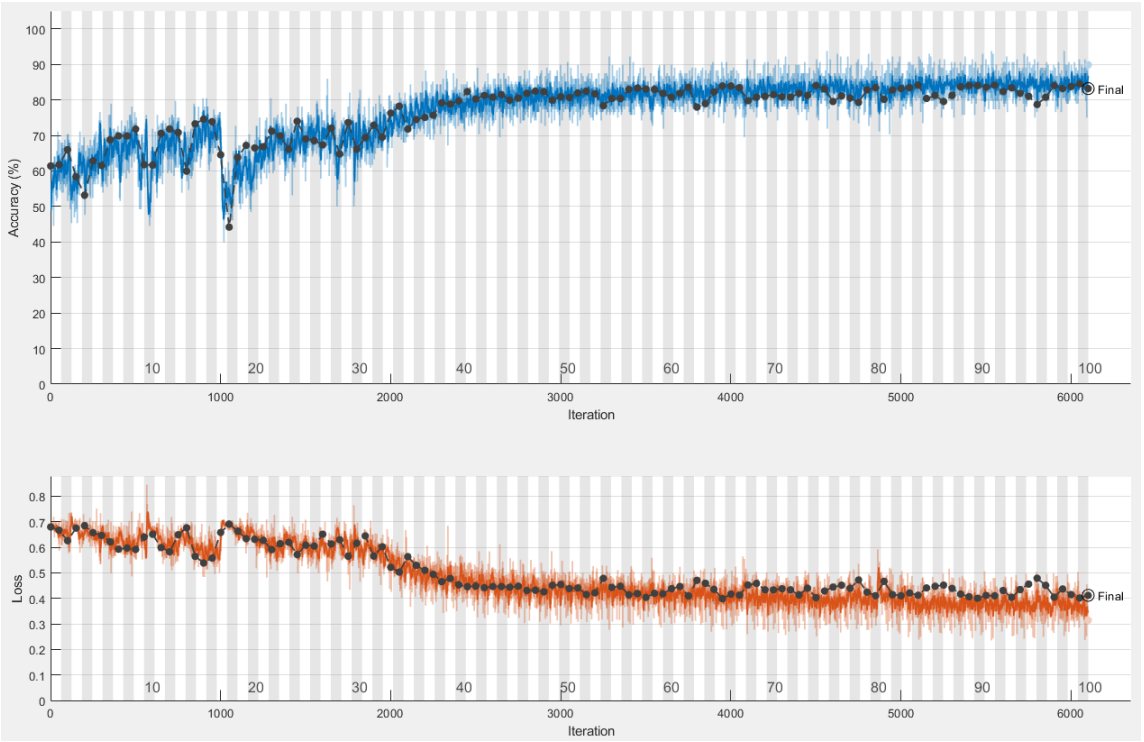


Fig. A.5.: Binary Width Model Training

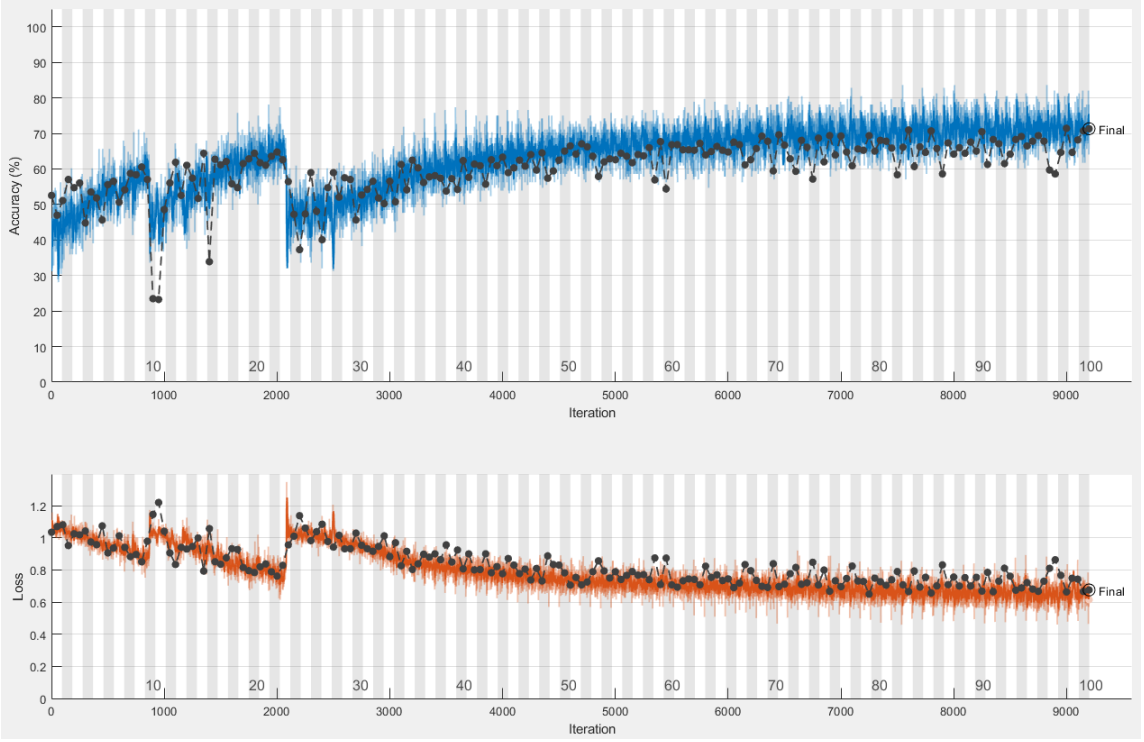


Fig. A.6.: Categorical Width Model Training

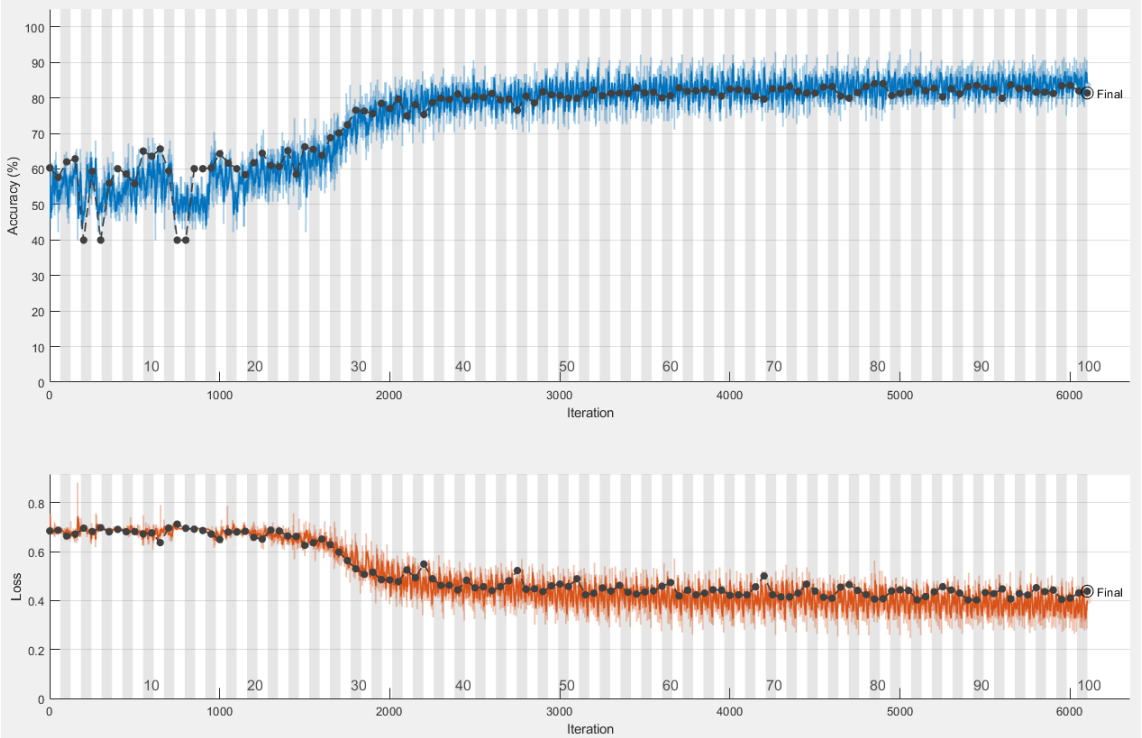


Fig. A.7.: Binary Depth Model Training

Fig. A.8.: Categorical Depth Model Training

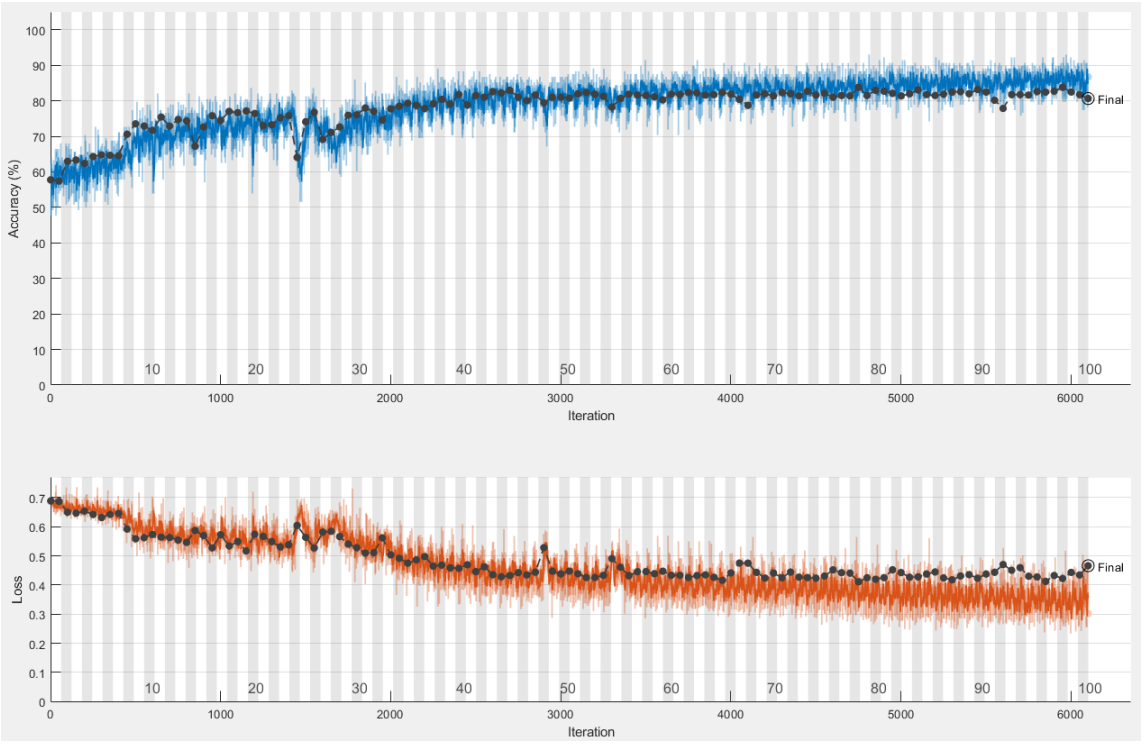


Fig. A.9.: Binary Hybrid Model Training

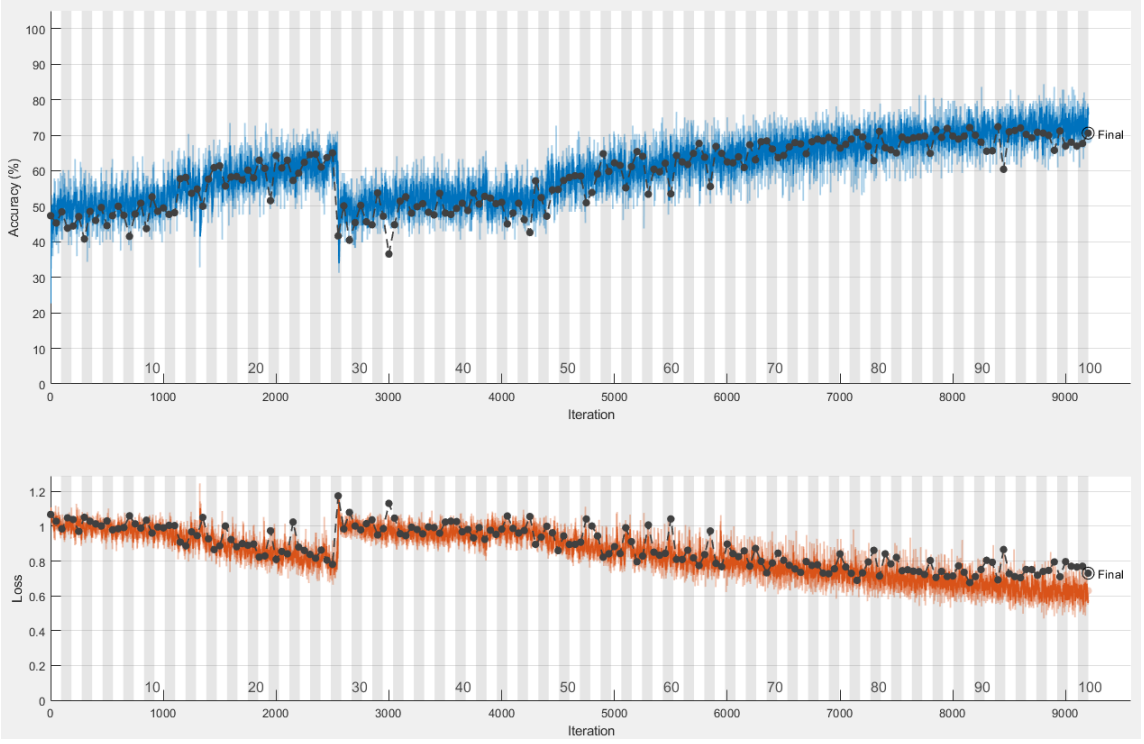


Fig. A.10.: Categorical Hybrid Model Training

References

B

Literature

[1] Office for National Statistics; UK Statistics Authority. *Decennial Life Tables*. 2015 (cit. on p. 1).

[2] European Heart Network. *European Cardiovascular Disease Statistics*. 2017 (cit. on p. 1).

[3] J. Pinnell; S. Turner; S. Howell. *Cardiac Muscle Physiology*. 2007 (cit. on pp. 2, 11).

[5] J.T. Ottesen; M.S. Olufsen; J.K. Larsen. *Applied Mathematical Models in Human Physiology*. 2004 (cit. on p. 2).

[6] H. Fukuta; W.C. Little. *The Cardiac Cycle and the Physiologic Basis of Left Ventricular Contraction, Ejection, Relaxation, and Filling*. 2008 (cit. on p. 3).

[7] N. Sperelakis; Y. Kurachi; A. Terzic; M. Cohen. *Heart Physiology and Pathophysiology 4th Edition*. Elsevier, 2000. ISBN: 9780126569759 (cit. on pp. 3, 4, 10, 13).

[9] G.A. Langer. *The Myocardium*. Academic Press, 1997. ISBN: 9780124365704 (cit. on pp. 4, 10).

[11] B. Freedman; J. Camm; H. Calkins. *Screening for Atrial Fibrillation*. Circulation Journal Vol 135 Issue 19, 2017 (cit. on p. 4).

[12] F. Violi; E.Z. Soliman; P. Pignatelli; D. Pastori. *Atrial Fibrillation and Myocardial Infarction: A Systematic Review and Appraisal of Pathophysiologic Mechanisms*. Journal of the American Heart Association Vol 5 Issue 5, 2016 (cit. on pp. 4, 10).

[13] V. Ruddox; I. Sandven; J. Munkhaugen; J. Skattebu; T. Edvardsen; J.E. Otterstad. *Atrial Fibrillation and the Risk for Myocardial Infarction, All-Cause Mortality and Heart Failure: A Systematic Review and Meta-Analysis*. European Journal of Preventive Cardiology Vol 24 Issue 14, 2017 (cit. on p. 4).

[14] T.B. Garcia. *12-Lead ECG: The Art of Interpretation*. Jones Bartlett Learning, 2001. ISBN: 0763712841 (cit. on pp. 5, 10, 11).

[15] J.P. Madeiro; P. Cortez; J.M. Filho; A.R. Brayner. *Developments and Applications for ECG Signal Processing*. Academic Press, 2018. ISBN: 9780128140352 (cit. on p. 5).

[16] M. AlGhatrif; J. Lindsay. *A Brief Review: History to Understand Fundamentals of Electrocardiography*. 2012 (cit. on p. 5).

[17] R. Zetterstorm. *Nobel Prize to Willem Einthoven in 1924 for the Discovery of the Mechanisms Underlying the Electrocardiogram*. 2009 (cit. on p. 6).

[19] A.L. Goldberger. *Goldbergers Clinical Electrocardiography 9th Edition*. Elsevier, 2006. ISBN: 9780323401692 (cit. on pp. 6, 11–13).

[20] S. Meek; F. Morris. *ABC of Clinical Electrocardiography: Leads, Rate, Rhythm, and Cardiac Axis*. 2006 (cit. on p. 6).

[21] S. Nedios; I. Romero; J.H. Gerds-Li; E. Fleck; C. Kriatselis. *Precordial Electrode Placement for Optimal ECG Monitoring*. 2014 (cit. on p. 8).

[24] I.B. Wilkinson. *Oxford Handbook of Clinical Medicine*. Oxford University Press, 2017. ISBN: 9780199689903 (cit. on pp. 9, 13).

[26] F. Rosenblatt. *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*. Psychological Review, 1958 (cit. on p. 13).

[27] A.M. Emad; A. Shenouda. *A Quantitative Comparison of Different MLP Activation Functions in Classification*. International Symposium on Neural Networks, 2006 (cit. on p. 14).

[28] A. Nur; N.H. Radzi; A.O. Ibrahim. *Artificial Neural Network Weight Optimization: A Review*. 2014 (cit. on p. 15).

[30] S. Ruder. *An Overview of Gradient Descent Optimization Algorithms*. 2017 (cit. on p. 16).

[31] P. Norvig; S.J. Russell. *Artificial Intelligence: A Modern Approach*. Pearson, 2009. ISBN: 0136042597 (cit. on p. 17).

[32] B.K. Christensen. *Matrix representation of a Neural Network*. 2003 (cit. on p. 17).

[33] L.R. Medsker. *Recurrent Neural Networks: Design and Applications*. CRC Press, 1999. ISBN: 0849371813 (cit. on p. 19).

[35] Z.C. Lipton; J. Berkowitz. *A Critical Review of Recurrent Neural Networksfor Sequence Learning*. 2015 (cit. on p. 19).

[36] R. Pascanu; T. Mikolov; Y. Bengio. *On the Difficulty of Training Recurrent Neural Networks*. 2012 (cit. on p. 20).

[37] A. Sherstinsky. *Fundamentals of Recurrent Neural Network and Long Short-Term Memory Network*. 2018 (cit. on pp. 20, 21).

[39] G. Clifford; C. Liu; B. Moody; L.H. Lehman; I. Silva; Q. Li; A. Johnson; R.G. Mark. *AF Classification from a Short Single Lead ECG Recording the PhysioNet Computing in Cardiology Challenge 2017*. Computing in Cardiology Vol 44 2017, 2017 (cit. on p. 22).

[40] J. Namikawa; J. Tani. *A Model for Learning to Segment Temporal Sequences Utilizing a Mixture of RNN Experts Together with Adaptive Variance*. 2008 (cit. on p. 26).

[41] S.C. Wong; A. Gatt; V. Stamatescu; M.D. McDonnell. *Understanding Data Augmentation for Classification when to Warp*. 2016 (cit. on p. 26).

[42] Z. Alom; T.M. Taha; C. Yakopcic. *A State of the Art Survey on Deep Learning Theory and Architectures*. 2019 (cit. on p. 27).

[43] S. Bai; J.Z. Kolter; V. Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. 2018 (cit. on pp. 30, 32).

[44] R. Caruana; S. Lawrence; L. Giles. *Overfitting in Neural Nets Backpropagation Conjugate Gradient and Early Stopping*. 2001 (cit. on p. 31).

[45] C. Kamath. *Quantification of Electrocardiogram Rhythmicity to Detect Life Threatening Cardiac Arrhythmias using Spectral Entropy*. Journal of Engineering Science and Technology, 2017 (cit. on p. 33).

[46] L. Qiu; G. Li. *Representation of ECG Signals based on the Instantaneous Frequency Estimation*. IEEE Conference on Signal Processing, 1996 (cit. on p. 34).

[47] T. Jayalakshmi; A.Santhakumaran. *Statistical Normalization and Back Propagation for Classification*. International Journal of Computer Theory and Engineering Vol 3 No 1, 2011 (cit. on p. 35).

[48] A.Y. Ng. *Feature Selection L1 vs L2 Regularization and Rotational Invariance*. Stanford University ICML 04 Proceedings, 2004 (cit. on p. 35).

[49] H. Cheng; L. Koc; J. Harmsen. *Wide and Deep Learning for Recommender Systems*. 2016 (cit. on p. 36).

[50] B. Eskofier S.Gradl; P. Kugler; C. Lohmuller. *Real-time ECG Monitoring and Arrhythmia Detection using Android-Based Mobile Devices*. 2012 (cit. on p. 43).

[51] T.H. Payne. *Introduction and Overview of Clinical Computing Systems within a Medical Center*. 2015 (cit. on p. 44).

[53] S. Bai; J.Z. Kolter; V. Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networksfor Sequence Modeling*. 2018 (cit. on p. 44).

[54] T.G. Dietterich. *Ensemble Methods in Machine Learning*. 2000 (cit. on p. 44).

Figures

[4] E. Pierce. *Diagram of the Human Heart*. Wikimedia Commons. GNU Free Documentation License. June 2, 2006. URL: https://commons.wikimedia.org/wiki/File:Diagram_of_the_human_heart_(cropped).svg (cit. on p. 2).

[8] C.R. Nave. *The Heart's Electrical Sequence*. Georgia State University. 2016. URL: http://hyperphysics.phy-astr.gsu.edu/hbase/Biology/ecg.html (cit. on p. 3).

[10] *Atrial Fibrillation*. Harvard Medical School. Dec. 2018. URL: https://www.health.harvard.edu/a_to_z/atrial-fibrillation-a-to-z (cit. on p. 4).

[18] *String Galvanometer*. London National Heart Hospital. 1916 (cit. on p. 6).

[22] E.A. Ashley; J. Niebauer. *Cardiology Explained - Conquering the ECG*. Remedica. 2004 (cit. on pp. 8, 10).

[23] *Understanding the ECG: Reading the Waves*. Harvard Medical School. Feb. 2011. URL: https://www.health.harvard.edu/heart-health/understanding-the-ecg-reading-the-waves (cit. on p. 9).

[25] *Cardiac Conduction*. CardioNetworks. Creative Commons BYNCSA 3.0 License. July 24, 2011. URL: https://en.ecgpedia.org/index.php?title=File:Conduction_ap.svg (cit. on p. 12).

[29] M. Gallagher. *Gradient Descent*. University of Queensland. 2018 (cit. on p. 15).

[34] F.M. Bianchi; E. Maiorino; M.C. Kampffmeyer; A. Rizzi; R. Jenssen. *Properties and Training in Recurrent Neural Networks*. Springer Briefs in Computer Science. Nov. 10, 2017 (cit. on p. 19).

[38] S. Yan. *Understanding LSTM and its diagrams*. ML Review. Mar. 13, 2016. URL: https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714 (cit. on p. 21).

[52] S. Saadatnejad; M. Oveisi; M. Hashemi. *LSTM Based ECG Classification for Continuous Monitoring on Personal Wearable Devices*. IEEE Journal of Biomedical and Health Informatics. 2018 (cit. on p. 44).

Index

C

List of Figures

List of Tables