Using LSTM network to detect R-peaks from noisy ECG signals

Master's Thesis
University of Turku
Department of Future Technologies
Health Technology
August 2020
Juho Laitala

UNIVERSITY OF TURKU
Department of Future Technologies

JUHO LAITALA: Using LSTM network to detect R-peaks from noisy ECG signals

Master's Thesis, 67 p., 9 app. p.
Health Technology
August 2020

Electrocardiogram (ECG) is one of the most important signals that can be measured from the human body. It contains lots of important information from the function of the heart, which can be utilized e.g. in medical diagnosis. Also, one of the vital signs, the heart rate can be derived from the ECG. Because of these reasons, ECG is extensively used by researchers and medical professionals. However, usage of ECG is not only limited into medical field as it is often collected e.g. during sport activity to track the heart rate. Utilization of the ECG is becoming even more widespread, today some of the newest smart watches have capability to measure it.

Detection of QRS complexes or R-peaks from the ECG signal is a prerequisite for heart rate calculation. Over time, numerous rule-based algorithms have been proposed for the task. Many of them work well when ECG signal has good quality, but their performance can drop in the presence of noise. Recently Laitala et al. [2] proposed robust R-peak detection algorithm that is based on Long Short-Term Memory (LSTM) network. The work of Laitala et al. is extended in this thesis. More detailed description is given from the LSTM based R-peak detection algorithm. It is also evaluated with new additional dataset and more QRS detection algorithms are used as reference.

Results are in line with the original work, LSTM based detector has the best general performance from the all of the evaluated detectors. However, results are not so striking as before. One reference detector bested the LSTM based detector with the new dataset. The major strength of the LSTM based detector is its robustness to noise. Another strong point is its ability to do sample precise R-peak detection. Majority of the reference detectors lack this ability and they often induce a lag to their predictions.

Keywords: ECG, LSTM, QRS, R-peak

Acknowledgements

This thesis expands the original work made by me and my co-authors [2]. Hence, I would like to express my gratitude to all of my co-authors as they helped me to create the solid foundations where it was easy to build on.

Contents

Abbreviations

Term	Abbreviation
Artificial Neural Network	ANN
Cross-Entropy	CE
Electrocardiogram	ECG
Glasgow University Database	GUDB
Heart Rate Variability	HRV
Independent and Identically Distributed	IID
Internet of Things	IoT
Left Arm	LA
Left Leg	LL
Long Short-Term Memory	LSTM
Multilayer Perceptron	MLP
Recurrent Neural Network	RNN
Right Arm	RA
Right Leg	RL
Root Mean Square of the Successive Differences	RMSSD
Signal-To-Noise Ratio	SNR
Squared Error	SE
University of California, Irvine	UCI
Wavelet transform	WT
Wilson's Central Terminal	WCT

Chapter 1

Introduction

1.1 Overview

Healthy living is important for everyone. Lots of resources have been used to develop methods that can assess our health. Many of these methods are based on signals that can be continuously measured and monitored from our bodies. These signals are known as biosignals and they can be e.g. electrical, acoustic, mechanical or chemical in nature. Our body emits these different signals constantly. For example, electrical current, sound and vibrations are produced on every heartbeat.

One of the most widely used biosignals is ECG (Fig. 1.1). It measures the electrical current that is produced by our most important organ, the heart. ECG is a graph where electrical activity (measured usually as mV) varies as a function of time. Voltage variations represent different electrical actions of the heart that are produced during the cardiac cycles (heartbeats).

ECG has long history, first ECG from human was recorded already in 1887 by Augustus Waller [1]. Since then the popularity of ECG has steadily increased. It is not anymore utilized solely by researchers or medical professionals as it can benefit others as well. Today ECG is widely utilized e.g. in chest straps that allow us to record our heart rate during sport activities. Also, in the recent years, the capability

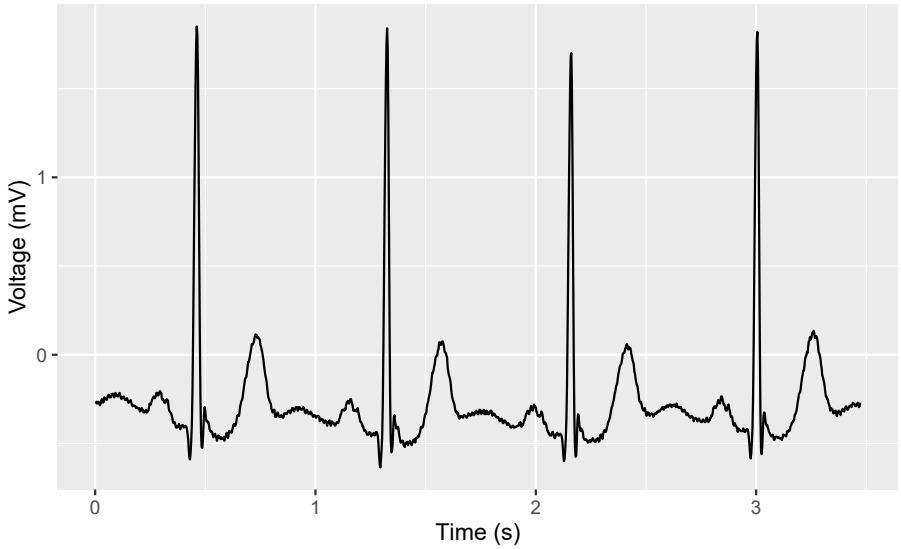

Figure 1.1: Example of typical ECG signal.

to measure ECG has been added to some small wearable devices like smart watches. Thus, development of the health technology may allow even wider adaptation of the ECG in the future.

ECG signal carries a lot of useful information from the heart and its function. Medical professionals can diagnose different heart diseases by studying the ECG waveform. Shapes and time intervals between the different ECG components give clues from the existence of different heart conditions. The most widely extracted information from the ECG is the heart rate which measures the number of heartbeats per minute. To calculate heart rate, it is necessary to identify locations of the individual heartbeats from the ECG. Heartbeat locations are usually determined by measuring the QRS complex or R-peak locations from the ECG. QRS complex is composed from three graphical deflections termed as Q, R and S waves. R wave is central part of the QRS complex, and its maximum is referred as R-peak. QRS complexes and R-peaks within them are the most easily distinguishable features of the ECG. Therefore, they are usually selected as markers of the individual heartbeats.

Over the years numerous algorithms have been proposed for QRS or R-peak detection. Majority of the proposed methods have been rule-based algorithms. They work generally very well on the regular ECG, but they can fail if ECG signal is noisy. Re-

cently Laitala et al. [2] proposed a robust R-peak detection algorithm that is based on LSTM network. The LSTM based detector showed good performance even on noisy ECG signals. However, one of the limitations of the work of Laitala et al. [2] was the small test set size (seven subjects). My goal in this thesis is to address these shortcomings and provide broader in-depth analysis of the LSTM based detector. Therefore, the research question remains largely the same as in the original publication: Is it possible to utilize modern deep learning methodologies for robust QRS or R-peak detection?

This thesis examines the question in a structured manner. First chapter is introductory in nature. Necessary background information from the human heart, ECG, existing QRS or R-peak detection algorithms and neural networks are provided. Second chapter explains the materials and methodologies used in this thesis. Development process, structure and working principles of the LSTM based detector are described in detail. Also used datasets and evaluation procedures are described in the second chapter. Third chapter focused totally on evaluation; LSTM based detector is evaluated with the reference detectors. Fourth and final chapter concludes the work.

1.2 Human heart

Circulatory (cardiovascular) system is the transportation system of the body. Its major purpose is to transport oxygen and different nutrients in and metabolic waste out from the body tissues [3]. Heart is the driving force of the circulatory system, it maintains circulation by constantly pumping blood. Heart is located almost middle of the chest, being behind and slightly left to the sternum (breastbone, Fig. 1.2). It is roughly a cone shaped muscular organ whose tip is pointing downward left and forward. Heart has typically the size of a fist and its weight is usually in the range of 255-340 grams [4].

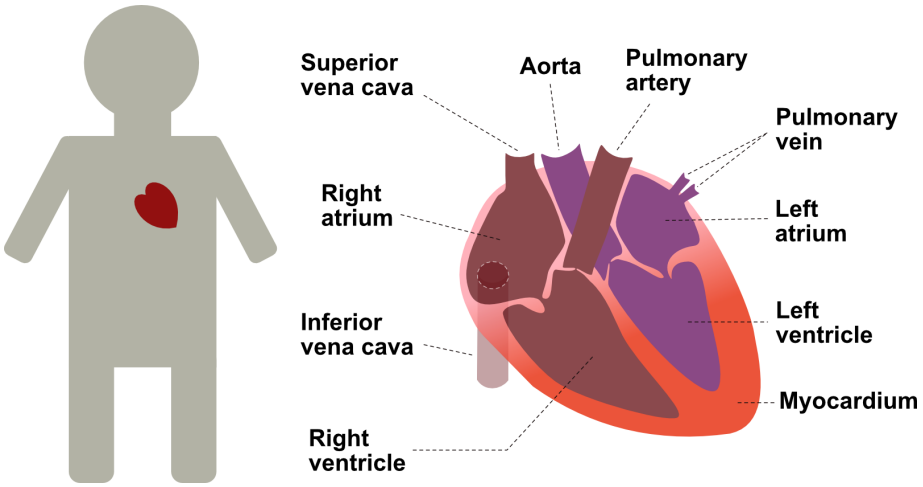

Figure 1.2: Location of the heart and its internal structure.

1.2.1 Structure of the heart

Heart is a hollow organ that is composed of four muscular chambers, left and right atria and corresponding ventricles. The outermost layer of the heart is called pericardium, it is a two-layer sac that surrounds the heart and roots of the great vessels. The space between the sacs layers has pericardial fluid which acts as an lubricant and reduces the friction during the heartbeats [3].

The thin lining in the interior surface of the chambers is called endocardium, it is composed just from single layer of endothelial cells [5]. The muscle tissue between pericardium and endocardium is termed the myocardium. It represents the largest portion of the heart's wall and it is the muscle that contracts in every heartbeat. The thickness of the chamber wall is determined by the amount of high-pressure work that the chamber is responsible. Atria have relatively light weight duties as they just collect blood for the ventricles, therefore their walls are thin when compared to ventricles [4]. Left ventricle has to pump blood against pressure that is seven times higher than the pressure is for the right ventricle. Hence its wall is much thicker (approx. 12 mm) than the wall of the right ventricle (approx. 5 mm) [3].

1.2.2 Electrical activity of the heart

Heart needs electrical stimulus to contract and pump blood. Creation and transmission of electrical impulses happens in cellular level when cardiac cells depolarize and repolarize. Depolarization-repolarization cycle can be summarized after Coviello [4] as follows: At rest, cardiac cells are polarized and there is no electrical activity. Cardiac cells have negative charges inside of them which is known as resting potential. This charge is due to different concentrations of ions (e.g. sodium, potassium) at the inside and outside of the cardiac cell (ions are separated by the cell membrane). After stimulus, ions exchange happens cross the cell membrane which leads to cell depolarization, or action potential. Fully depolarized cell then returns to its resting state in a process that is known as repolarization. Some of the cardiac cells have ability to initiate an impulse spontaneously (pacemaker cells) while others receive the impulse from neighboring cells by conduction.

Electrical impulses resulting from depolarization and repolarization flow through the heart via electrical conduction system (Fig. 1.3). When heart functions normally, each heartbeat initiates from the sinoatrial node which is known as the heart's natural pacemaker. Sinoatrial node is about 1 cm long strip of specialized cardiac muscle fibers (pacemaker cells) in the upper right wall of the right atrium [3]. After electrical impulse has been generated by the sinoatrial node, it travels first through the both atriums. In the right atrium impulse travels via anterior, middle and posterior internodal tracts while conduction to the left atria happens through Bachmann's bundle of nerves [4]. Electrical impulse goes to the atrioventicular node at the base of the right atria before it reaches the ventricles. There impulse is delayed by 0.04 seconds which prevents ventricles to contract too quickly and allows ventricles to fill up with blood when the atria contracts [4]. After delay, electrical impulse leaves the atrioventicular node and continues its journey towards the tip of the heart. It travels through bundle of his to the right and left bundle branches. From the branches the electrical impulse finally goes via Purkinje fibers into the endocardium and deep into the myocardium. This makes both ventricles to contract simultaneously.

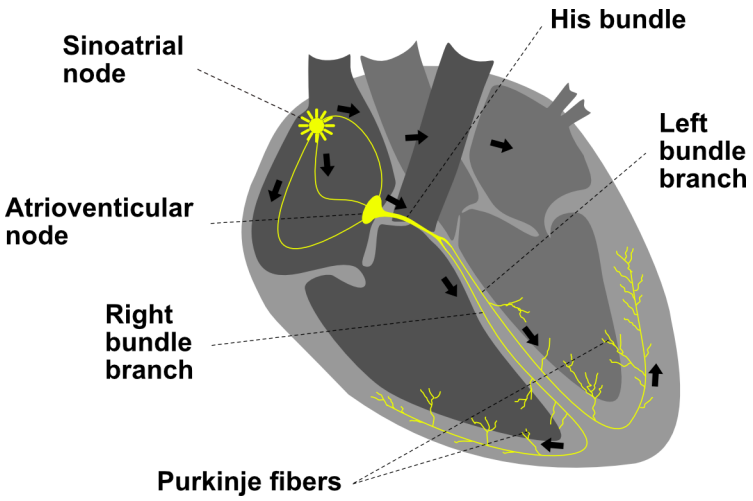

Figure 1.3: Simplified illustration of the electrical conduction system of the heart

1.2.3 Function of the heart

Contractions of the heart push blood in motion through the circulatory system. Circulation of blood can be summarized as follows: At first, deoxygenated blood enters to the right atrium via superior and inferior vena cava. Then right atria pushes blood into the right ventricle. When right ventricle contracts, it pushes blood through the pulmonary arteries into the lungs. In lungs, carbon dioxide is released from the blood while oxygen is absorbed. Oxygen-rich blood travels back to the heart, to the left atrium via pulmonary veins. Then left atria pushes blood into the left ventricle which in turn pumps the blood through the whole body.

Heart is partly controlled by the sympathetic and parasympathetic branches of the autonomous nervous system. Sympathetic nerves accelerate the heart while parasympathetic nerves slow it down. However, neural connections are not necessary because the heart can function autonomically, even right after the heart transplantation without any neural connections [3].

1.3 Electrocardiogram

1.3.1 ECG Measurement

ECG is measured from the surface of the skin by using electrodes (Fig. 1.4). Two electrodes form an electrode pair that is known as lead and the electrical potential difference between the pair is known as lead voltage. Multiple different electrode configurations can be used to obtain ECG. According to Webb [6] the simplest geometrical recording configuration uses three active electrodes with one ground electrode. In this configuration the active electrodes are placed on the right arm (RA), left arm (LA) and left leg (LL) while the ground electrode is placed on the right leg (RL). Three different leads can be derived from the active electrodes. These leads are also known as limb leads I-III and they are defined as:

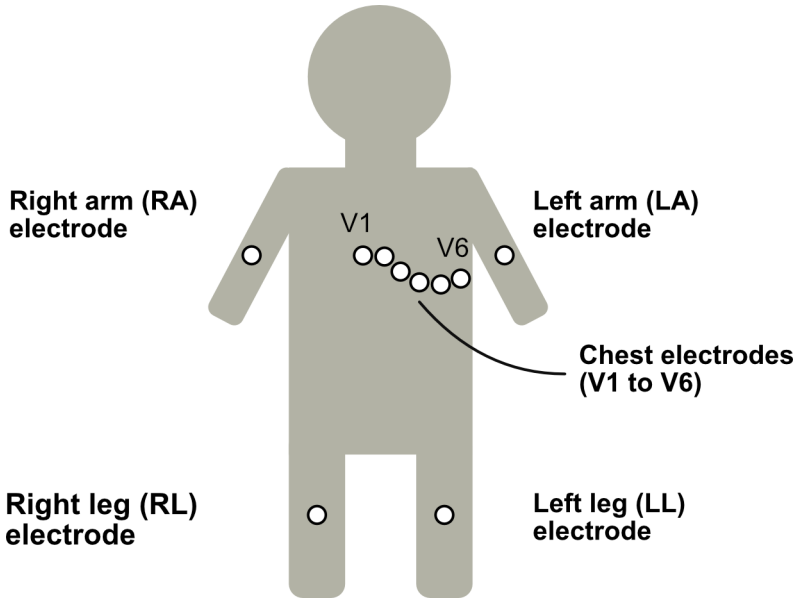

Figure 1.4: Placement of electrodes.

$$\text{Lead I} = \text{LA} + \text{RA}$$

$$\text{Lead II} = \text{LL} + \text{RA}$$

$$\text{Lead III} = \text{LL} + \text{LA}$$

These leads are referred as bipolar, which means that one electrode is positive pole while the other acts as negative reference [5]. They share a relationship that is known as Einthoven's law which can be expressed as:

$$\text{Lead II} = \text{Lead I} + \text{Lead III}$$

This means that three limb leads contain only two pieces of independent information [6], i.e. if two leads are known then third can be calculated. In addition to limb leads I-III there is also three augmented limb leads that are known as aVR, aVL and aVF. These leads are referred as unipolar, which means that there is positive pole but no single negative pole. Negative references for unipolar leads are derived by averaging other limb electrodes [5]. In the case of augmented limb leads aVR, aVL and aVF negative references are Goldberg's central terminals of (LA + LL)/2, (RA + LL)/2 and (RA + LA)/2 respectively [6]. These leads are then defined as:

$$\text{aVR} = -\frac{\text{Lead I} + \text{Lead II}}{2}$$

$$\text{aVL} = \text{Lead I} - \frac{\text{Lead II}}{2}$$

$$\text{aVF} = \text{Lead II} - \frac{\text{Lead I}}{2}$$

Together these six limb leads (I, II, III, aVR, aVL and aVF) give information from the heart's frontal plane, which is a vertical cut through the heart [4]. However, these leads alone do not yet give complete picture from the electrical activity of the heart. Information from the perpendicular plane is also needed. It can be collected by using six chest (precordial) electrodes V1, V2, V3, V4, V5 and V6. These electrodes give information from the heart's horizontal plane, which is a transverse cut through the heart [4]. These leads are unipolar in nature and their negative reference is the Wilson's central terminal (WCT), which is produced by averaging measurements from the three limb electrodes [6]. WCT is defined as:

$$\text{WCT} = \frac{\text{LA} + \text{RA} + \text{LL}}{3}$$

By using all of the leads (six limb and six chest leads) complete ECG, also known as 12-lead ECG can be obtained. It provides 12 different views of the heart which is very useful in medical diagnosis.

1.3.2 ECG Interpretation

The direction of the electrical current determines the magnitude and direction of the deflection (upwards or downwards) for waveform in ECG. Relationship between electrical force and ECG waveform can be summarized after Lilly [5] by following four points:

1. Electrical forces towards the positive electrode of a lead produces positive deflection.
2. Electrical force heading away from the positive electrode of a lead produces negative deflection.
3. Magnitude of the deflection is related to the angle between electrical force and lead axis, magnitude of deflection is highest when they are parallel.
4. If electrical force is perpendicular to the lead axis, then lead does not register any activity (flat line on the ECG).

P-wave, QRS complex and T-wave are the major components of typical ECG (Fig. 1.5). Together they represent one normal heartbeat. Each heartbeat begins with P-wave which represents depolarization of the atria. After P-wave comes QRS complex which represents depolarization of the ventricles. Last component of a heartbeat is T-wave which represents repolarization of the ventricles. Besides these main components, there are also other features that are commonly utilized by the medical professionals e.g. time intervals between the different components.

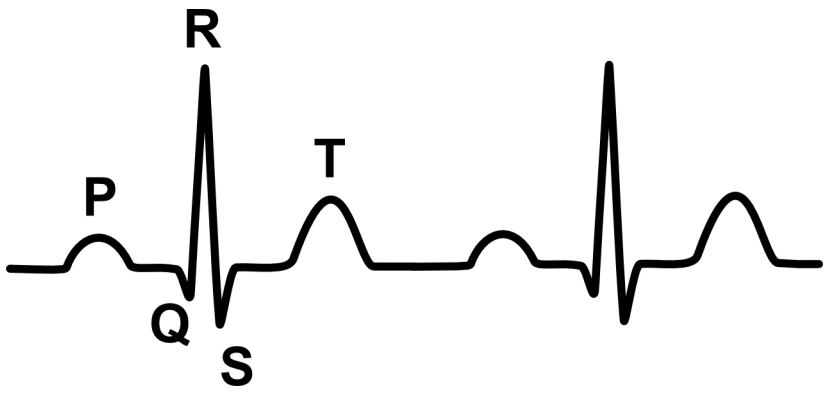

Figure 1.5: Schematic illustration of normal ECG.

1.3.3 ECG Noise sources

ECG signals might contain different types of noises that hinder the use of ECG (Fig. 1.6). Moody et al. [7] classified noise according to its frequency-domain characteristics to following four categories:

1. Baseline wander
2. Electrode motion artifact
3. Muscle noise
4. Power line interference

Baseline wander is low-frequency signal produced by motions of the subject or leads. Motion can happen e.g. when patient breaths. This adds sinusoidal component which has the frequency of the respiration (0.15 to 0.3 Hz) to the ECG [8]. Electrode motion artifacts are produced when mechanical forces act on electrodes frequently. According to the Moody et al. [7] this noise type is most challenging as it can resemble the features of the ECG signal. Muscle noise, also known as electromyographic interference is produced when muscles (other than heart) contract. Muscle contractions produce signals that last for approximately 50 ms and whose frequency content ranges from DC to 10 kHz. Power line interference is sinusoidal interference whose frequency (50 or 60 Hz) depends on the country. In addition to these four noises there can be also electrode pop or contact noise which occurs when electrode loses

the contact to the skin [6]. As a result of electrode disconnection, signal baseline shifts sharply.

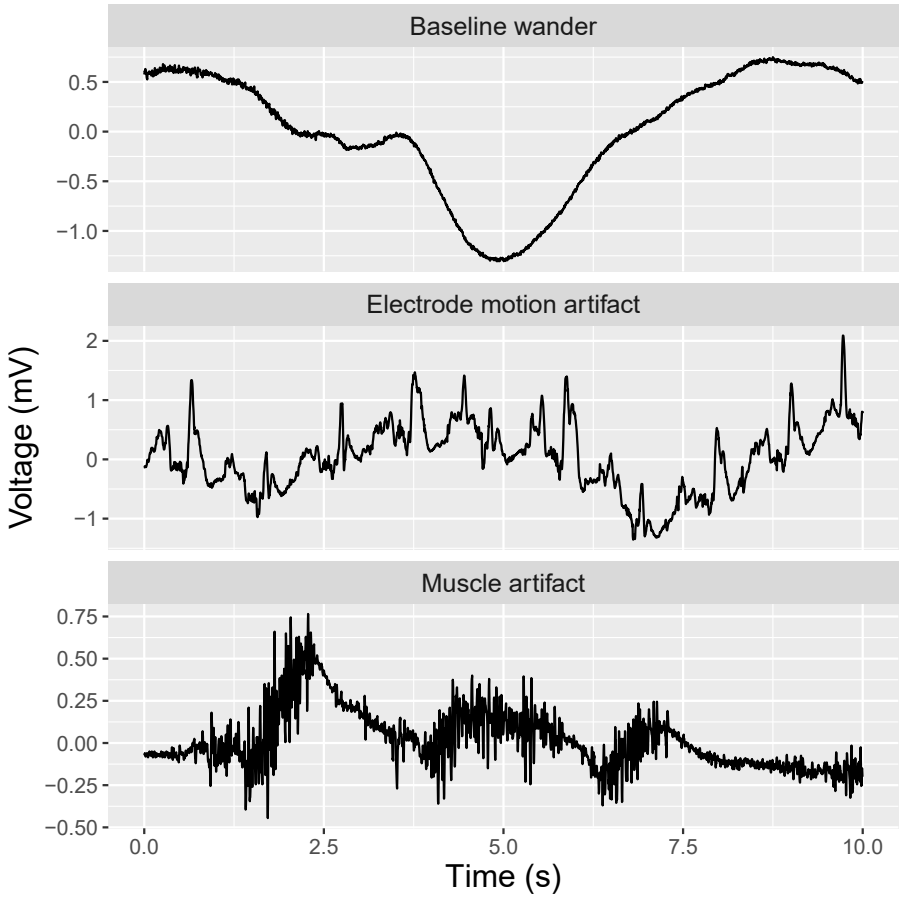

Figure 1.6: Examples of common noise types in ECG. Noise examples are from MIT-BIH Noise Stress Test Database [9].

1.4 QRS detectors

Detection of QRS complexes or R-peaks from ECG signal is an old problem and many different algorithms have been proposed as an solution. In 2002 Köhler et al. [10] reviewed common detectors and divided them into four different groups that are based on either:

1. Signal derivatives and digital filters

2. Wavelets
3. Neural networks
4. Additional approaches

Methods in the first group are commonly composed from separate preprocessing stage and decision stages (Fig. 1.7). The purpose of preprocessing stage is to make decision stage easier by enhancing the input ECG. This is usually achieved by the usage of bandpass filter that attenuates noise and other ECG components that might hamper the QRS detection. One of the most well-known algorithms belonging into first group is the Pan-Tompkins algorithm [11]. It uses the bandpass filter with a passband of 5-15 Hz as a first step. Additional measures of the preprocessing stage are the derivative filter, squaring and 150 ms moving average filter. Purpose of derivative filter is to provide QRS complex slope information while the squaring makes all points positive and enhances QRS complexes. Moving average filter acts as final preprocessing step, its purpose is to give information of QRS complex duration. After preprocessing steps, the local peaks (fiducial marks) are detected from the preprocessed signal. Each fiducial mark is considered as candidate value for an QRS complex. The final QRS complexes are retrieved by comparing amplitude of each peak to the adaptive threshold that contains information form previously detected QRS complexes and noise level.

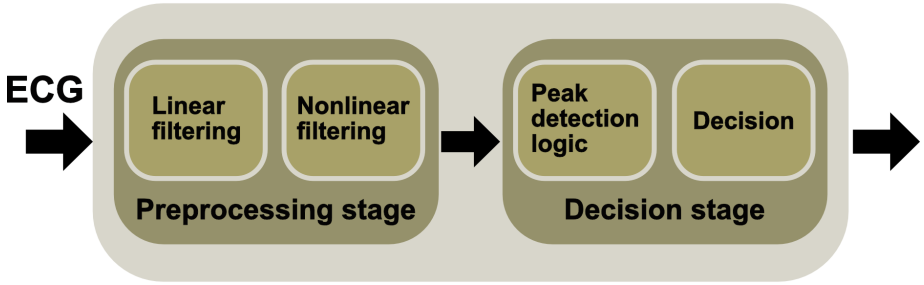

Figure 1.7: Detectors commonly have separate preprocessing and decision stages. After decision stage, locations of detected QRS complexes or R-peaks are returned. Modified after Köhler et al. [10].

Wavelet transform (WT) is a widely used tool in the field of signal processing. Similarly to Fourier transform, WT can give information from the frequency charac-

teristics of the signal. In contrast to Fourier transformation which uses various stretched and infinite length sinusoids, WT uses shorter waveforms (wavelets) with finite lengths [12]. Wavelets can be generated from single mother wavelet $\Psi(t)$ by scaling (a) and translation (b):

$$\Psi_{a,b}(t) = \frac{1}{\sqrt{2}}\Psi\left(\frac{t-b}{a}\right) \tag{1.1}$$

Similarly to short-time Fourier transform, WT is not just limited to frequency analysis and it can also provide valuable time information. While the short-time Fourier transform produces time-frequency representation, WT produces time-scale representation [10]. Generally, the WT of a function *f(t)* can be expressed by the following equation:

$$Wf(a,b) = \int_{-\infty}^{\infty} f(t)\Psi^*_{a,b}(t)dt \tag{1.2}$$

Where the * means complex conjugate. Many of the proposed wavelet-based approaches (e.g. [13]) are based on singularity detection algorithm of Mallat and Hwang [14]. Some of the methods (e.g. [15]) utilize WT in the preprocessing stage to remove noise and enhance QRS complexes.

Artificial neural networks (ANNs) have been also proposed as solution to the QRS or R-peak detection problem. According to the 2002 review of Köhler et al. [10], the most common neural network architectures proposed as solution were mainly multilayer perceptron (MLP), learning vector quantization or radial basis function networks. In the recent years, researchers have focused into more advanced neural network architectures like convolutional neural networks or LSTM networks [2], [16]–[18]. ANNs are able to map inputs into good decision outcomes by automatically identifying and extracting relevant patterns from the input data [19]. Thus, ANNs offer a more data-driven approach than traditional methods that are hand crafted solutions which rely to different digital signal processing techniques. Traditional methods can work very well when ECG quality is good, but their performance can

drop when the ECG signal contains noise and artifacts. ANNs have potential to cope even with noisy ECG signals, and thus, different ANN architectures have been proposed as solution to the problem of noisy ECG [2], [18], [20].

Numerous alternative solutions that belong to the group 4 have also been proposed. Following Köhler et al. [10] these methods can be divided into following subgroups that are based either: adaptive filters, hidden Markov models, mathematical morphology, matched filters, genetic algorithms, Hilbert transform, length and energy transforms, syntactic methods, maximum a posteriori estimation and zero crossings counts. Also, common machine learning methods like support vector machine [21] or k-nearest neighbor [22] have been proposed.

1.5 Artificial neural networks

Artificial neural network (ANN) is a machine learning method that is based on the connectionism. The key concept behind the connectionism is the idea that intelligent behavior can be achieved when numerous simple processing units (i.e. neurons) work together [19]. ANNs are able to create distributed representations from concepts, meaning that network of neurons can represent concept by pattern of activity [23]. Right now ANNs are a very hot topic. This is due the success achieved in the field of deep learning during the last decade. Deep learning is a subfield of machine learning that uses a large (deep) and often very complex ANN as a machine learning model. Deep ANNs have achieved remarkable results e.g. in the fields of image recognition [24], speech recognition [25], language translation [26] and many other fields.

1.5.1 Single-layer neural networks

Theory behind ANN started to form already in 1943 when Warren McCulloch and Walter Pits tried to understand how biological brain works [27]. They presented simple computational model describing neural events and their relations by the means of propositional logic. Their model is based on artificial neurons that are also known

as Threshold Logic Units. In essence, first artificial neurons were simple logic gates with binary input(s) and output. They produced outputs (fired) only if their inputs reached some predefined threshold value. Linking artificial neurons together as network could have allowed to perform more complex computations.

The next major step was taken in 1958 when Frank Rosenblatt introduced the perceptron model (Fig. 1.8) [28]. It's more flexible than McCulloc-Pits Threshold Logic Unit as neuron inputs are not just binary values anymore. Input vector $\mathbf{x}$ is multiplied by learnable weight coefficients $\mathbf{w}$. Thus, neuron input z then becomes a weighted sum of input values:

$$z = w_1x_1 + w_2x_2 + ... + w_nx_n = \sum_{i=1}^{n} w_ix_i = \mathbf{w}^T\mathbf{x} \tag{1.3}$$

This linear combination of input values (z) is then fed to the step function where it is compared to the threshold value θ. Step function is also known as activation function, because it determines if neuron activates (fires) or not. Heaviside or sign is commonly used as step function [29]:

$$heaviside(z) = \begin{cases} 1 & z \geq \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{1.4}$$

$$sgn(z) = \begin{cases} 1 & z \geq \theta, \\ -1 & \text{otherwise.} \end{cases} \tag{1.5}$$

For both functions, θ can be moved to the left side of the inequation and it can be defined as a weight $w_0 = -\theta$ for input x_0. The additional input feature x_0 outputs 1 all the time ($x_0 = 1$) and it is known as the bias neuron. After rearrangement of terms, the inequation becomes:

$$z = w_0x_0 + w_1x_1 + w_2x_2... + w_nx_n \geq 0 \tag{1.6}$$

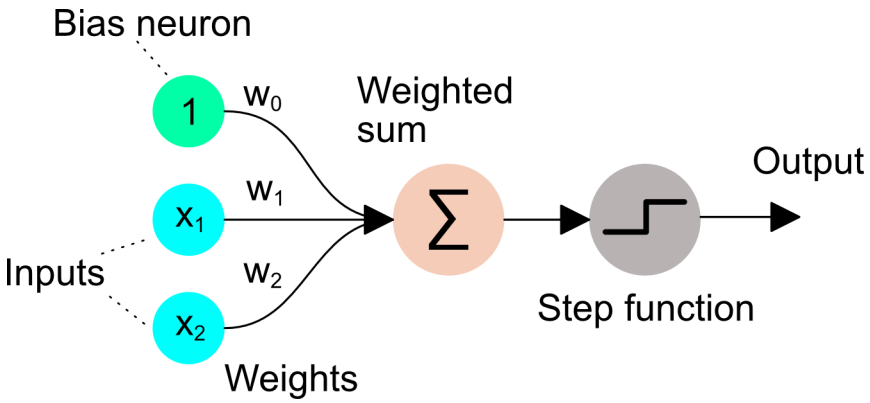

Figure 1.8: Perceptron architecture with two inputs and one output.

Before training, learnable weight coefficients ($\mathbf{w}$) are usually initialized to small random numbers. During training, weight coefficients are updated by using perceptron learning rule:

$$\mathbf{w}_{j+1} = \mathbf{w}_j + \alpha e \mathbf{x}^{(i)} \tag{1.7}$$

Where $\mathbf{w}_{j+i}$ is the new weight vector, $\mathbf{w}_j$ refers to current weight vector, α is a learning rate, e is error and $\mathbf{x}^{(i)}$ refers to ith training example. Error e is the difference between the true class label $y^{(i)}$ and predicted class label $\hat{y}^{(i)}$ of the ith training example:

$$e = (y^{(i)} - \hat{y}^{(i)}) \tag{1.8}$$

In overall, working principles of the perceptron algorithm can be summarized as follows:

1. Initialize random weights $\mathbf{w}$.
2. For each training example $\mathbf{x}^{(i)}$ in the dataset:
    a. Apply step function to the dot product $\mathbf{w}^T\mathbf{x}$ to get $\hat{y}^{(i)}$.
    b. Calculate error e between $y^{(i)}$ and $\hat{y}^{(i)}$.
    c. Multiply input features $\mathbf{x}^{(i)}$ with learning rate α and error e.

d. Add product of previous step (c) to the weight vector **w** to update weights.

In 1969 Minsky and Papert [30] highlighted the limitations of perceptrons in their book *Perceptrons*. One major limitation is the incapability to solve exclusive-or classification problem. Thus, perceptons have same limitations as other linear classifiers, they work well only for linearly separable data.

1.5.2 Multi-layer neural networks

Multi-layer network is composed of input layer, output layer and one or more hidden layers (Fig. 1.9). Size of input layer must equal to number of features while size of output layer depends on a given task e.g. one neuron in binary classification and three or more neurons in a multiclass classification. There are no strict rules for the number or sizes of the hidden layers. If all neurons of the layer are connected to the all neurons in the previous layer, then layer is considered to be fully connected or dense [29]. If network has more than one hidden layer, it is also known as deep ANN [31]. If input signal flows only forward in the network (i.e. no recurrent connections), then architecture represents a feedforward neural network.

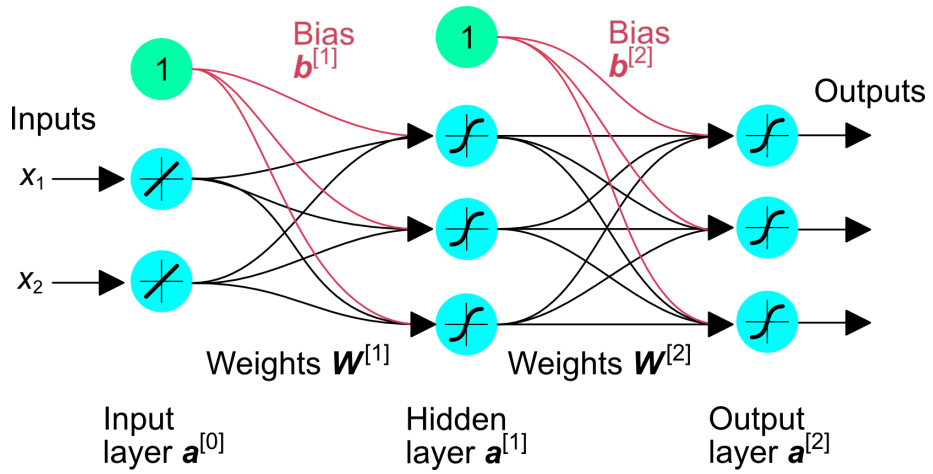

Figure 1.9: Multi-layer network with one hidden layer. Network takes two inputs and produces three outputs.

It is possible to solve exclusive-or classification problem by stacking perceptrons to form a multi-layer architecture that is known as multilayer perceptron (MLP). In

fact, a neural network with just one hidden layer can approximate any continuous function to any degree of accuracy, given that it has enough nodes [32]. This property is known as universal approximation theorem [33], [34]. Large enough neural network might be able to represent any continuous function, however, it is not guaranteed that the learning algorithm can learn that function [35]. For example, learning algorithm might overfit and choose a wrong function or it might not be able to find the parameters that correspond to the target function.

I describe simple multi-layered network architecture and working principles of it by using roughly similar mathematical notations as Andrew Ng and Kian Katanforoosh used in CS229 lecture notes [36]. Outputs of the different layers are denoted as $\mathbf{a}^{[l]}$. Letter $\mathbf{a}$ is used as it refers to the "activation" value of the neuron. Letter l in superscript brackets foo$^{[l]}$ refers to the layer index (zero-indexing), e.g. $\mathbf{a}^{[0]}$ would refer to the activations of the input layer and $\mathbf{a}^{[1]}$ activations of the first hidden layer. Activation value of the individual neuron j in layer l can be referred by using subscript j e.g. $\mathbf{a}_j^{[1]}$ would refer to the activation of jth unit in first hidden layer. Thus, output of the layer l is a vector that contains the activation values of the neurons in that particular layer:

$$\mathbf{a}^{[l]} = \begin{bmatrix} a_1^{[l]} \\ a_2^{[l]} \\ \vdots \\ a_j^{[l]} \end{bmatrix} \tag{1.9}$$

Activations of the first layer of the network (input layer $\mathbf{a}^{[0]}$) are equal to the input values (features). Neurons of the input layer are passthrough neurons that just pass the signal through [29]. For the first hidden layer, neuron inputs are linear combinations of input features with additional bias values. Thus, the input of the first neuron in the first hidden layer is:

$$z_1^{[1]} = \mathbf{W}_1^{[1]}\mathbf{x} + \mathbf{b}_1^{[1]} \tag{1.10}$$

Where W is a weight matrix and b is a bias term. W_1 refers to first row of the weight matrix while b_1 is a first component of a bias vector. Output or activation for this particular neuron can be calculated applying activation function g to the neuron input z. This can be expressed as:

$$a_1^{[1]} = g(z_1^{[1]}) \tag{1.11}$$

Non-linear activation function is needed to solve complex problems. It must be differentiable to allow the use of gradient-based approach to learn the weights that are connected to the neuron [31]. Some of the popular activation functions are (Fig. 1.10):

$$sigmoid(z) = \frac{1}{1 + e^{-z}} \tag{1.12}$$

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{1.13}$$

$$ReLU(z) = max(z, 0) \tag{1.14}$$

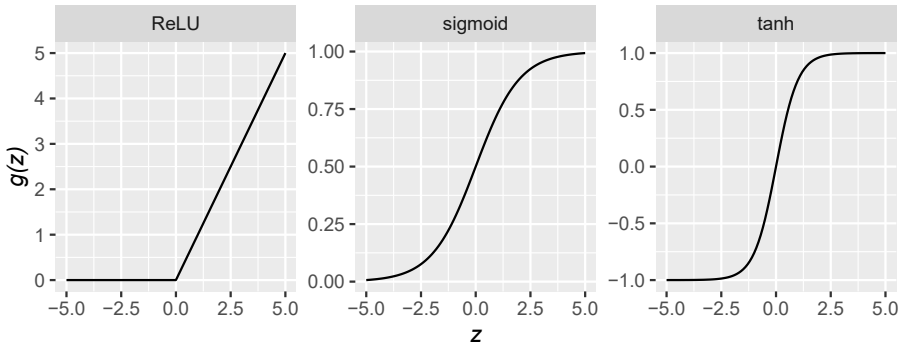

Figure 1.10: Popular activation funtions

To summarize, activations of the first hidden layer $\mathbf{a}^{[1]}$ can be calculated in two steps by fist calculating the layer input $\mathbf{z}$ and then applying the activation function g to every component of $\mathbf{z}$:

$$\text{(step 1)} \quad \mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]} \tag{1.15}$$

$$\text{(step 2)} \quad \mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]}) \tag{1.16}$$

To calculate activations of the subsequent hidden layer (or any other layer l), similar steps can be taken. In general, activations of the previous layer *l-1* are used to calculate activations of the current layer l:

$$\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]}) = g^{[l]}(\mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}) \tag{1.17}$$

Training multi-layered neural networks was not possible until 1986 when David Rumelhart, Geoffrey Hinton and Ronald Williams introduced backpropagation training algorithm [37]. In summary, backpropagation algorithm has two phases, the forward phase and the backward phase. Before forward phase, weight and bias parameters of a network need to be initialized. For example, small random values that are normally distributed around zero ($\mathcal{N}(0,\ 0.1)$) can be used as initial parameter values [36].

In forward phase, the input signal is first propagated forward from the input layer to the output layer to compute the network prediction. Then loss function $\mathcal{L}$ is used to measure how well prediction matches the target value (label). Loss function compares prediction (activation of the output layer) to target and returns a scalar value which represents a difference between the two. For example, Squared Error (SE) can be used in regression tasks and Cross-Entropy (CE) can be used in classification tasks as a loss function:

$$SE(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2 \tag{1.18}$$

$$CE(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log (1 - \hat{y})) \tag{1.19}$$

In backward phase error signal (loss) is propagated backwards from the output layer to input layer. This is done by calculating the gradient of the loss function with respect to ANN parameters (weights and biases). I demonstrate this whole process with simple ANN (Fig. 1.11) that can be used for regression task. To recap, squared error loss for this example ANN when single training example $\mathbf{x}^{(i)}$ is used can be expressed as a nested function:

$$\mathcal{L} = \frac{1}{2}(g^{[2]}(\mathbf{w}^{[2]^T}\, g^{[1]}(\underbrace{\mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}}_{\mathbf{z}^{[1]}}) + b^{[2]}) - y)^2 \tag{1.20}$$

$\mathbf{a}^{[1]}$

$z^{[2]}$

$\hat{y}=a^{[2]}$

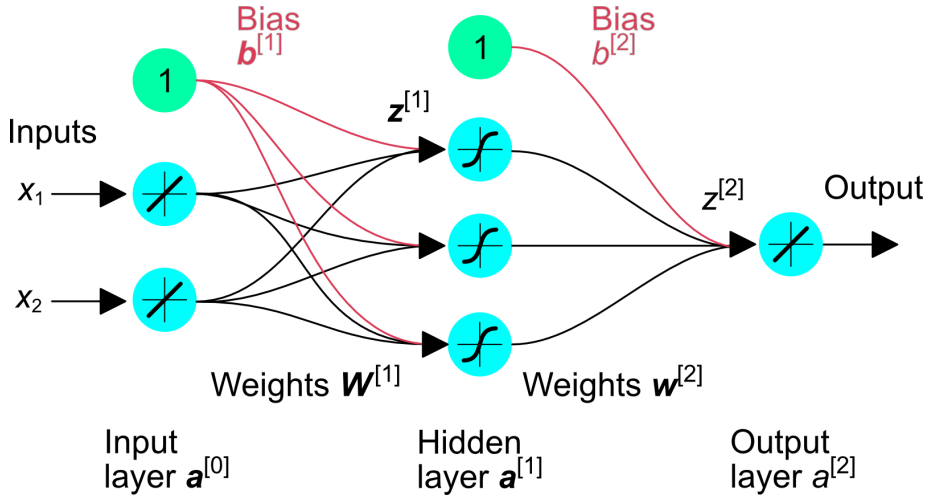

Figure 1.11: Simple ANN architecture for regression. ANN uses sigmoid activation in the hidden layer and linear activation in the output layer

Gradients with respect to different ANN parameters can be calculated by utilizing the chain rule. For the example ANN, calculations are done with the following

equations:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}^{[2]}} = \frac{\partial \mathcal{L}}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial \mathbf{w}^{[2]}} = (a^{[2]} - y)\mathbf{a}^{[1]} \tag{1.21}$$

$$\frac{\partial \mathcal{L}}{\partial b^{[2]}} = \frac{\partial \mathcal{L}}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial b^{[2]}} = (a^{[2]} - y) \tag{1.22}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1]}} = \frac{\partial \mathcal{L}}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial \mathbf{a}^{[1]}} \frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}} \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}} = (a^{[2]} - y)\mathbf{w}^{[2]} \odot g'(\mathbf{z}^{[1]})\mathbf{x}^T \tag{1.23}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1]}} = \frac{\partial \mathcal{L}}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial \mathbf{a}^{[1]}} \frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}} \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{b}^{[1]}} = (a^{[2]} - y)\mathbf{w}^{[2]} \odot g'(\mathbf{z}^{[1]}) \tag{1.24}$$

Where $\odot$ indicates element-wise multiplication. Note how right sides of the equations share common terms. In practice, there is no redundant calculations and each term is calculated only once. As network is iterated backwards, terms calculated for layer l are reused for the calculations of layer *l-1*.

After gradients of loss with respect to different network parameters are calculated, they can be used to update the weights to decrease the loss. Update of parameters is done by using the gradient descent method. This means that network parameters are updated by moving them by small steps in the opposite direction of gradient. The size of the update step is proportional to the gradient and also controlled by the learning rate (α). This process is illustrated in Fig. 1.12. Parameter updates for layer l of the ANN can be done with following rules:

$$\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[l]}} \tag{1.25}$$

$$\mathbf{b}^{[l]} = \mathbf{b}^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[l]}} \tag{1.26}$$

In practice, optimization of the ANN is more complex than shown in Fig. 1.12.

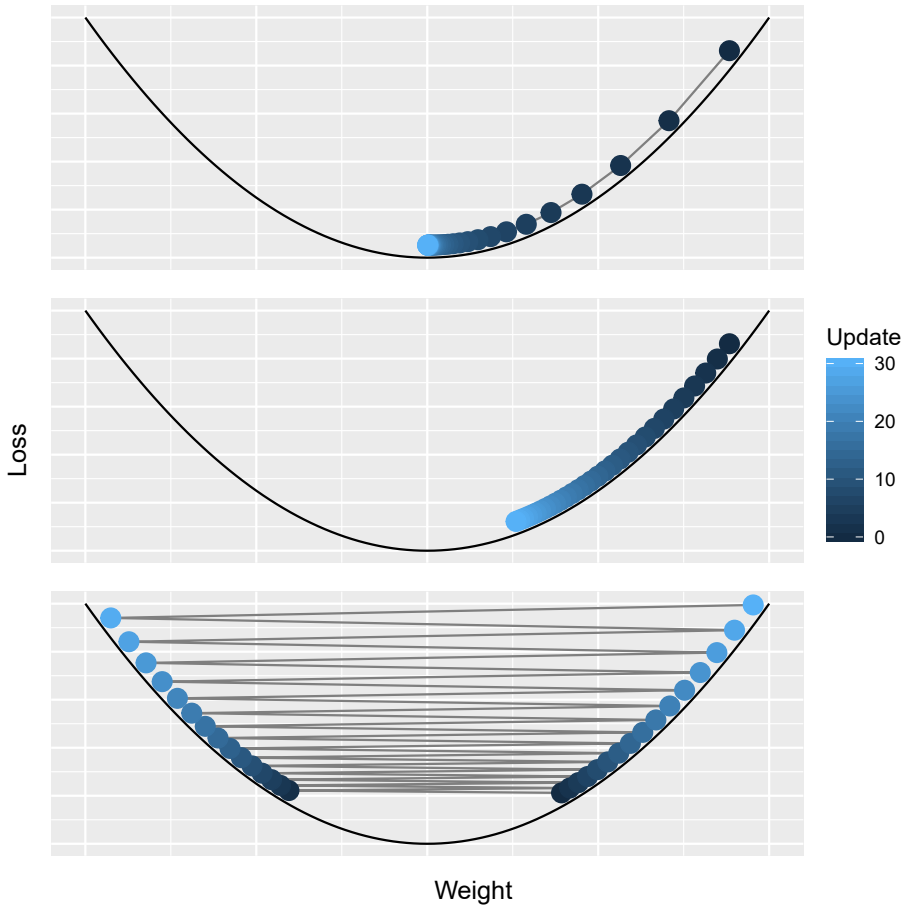

Figure 1.12: Simplified illustration of the gradient descent method and effect of learning rate with the model that contains only one parameter. In topmost figure learning rate has a good value and gradient descent converges to global minimum. Note how update steps are largest at the beginning when derivatives are also larger. Figure in the middle shows what happens when too small learning rate is used. In this case algorithm does not converge because learning happens too slow. In the other hand, too high learning rate might lead to oscillating behavior around the minimum and algorithm might even diverge from the minimum. This is illustrated at the bottommost figure.

Instead of just one global minimum, multiple local minima might be present. This could be seen a problem for a gradient-based methods as they head to the nearest local minimum and might get stuck there. If the local minimum has high loss, this leads to poor results. However, large neural networks rarely get stuck to local minimum and even if it happens, the local minimum is often as good solution as the global minimum [29]. Also, in high dimensional spaces saddle points are more common zero gradient points than local minima [35]. Therefore, learning process can still be hindered by saddle points, plateaus or other flat regions.

Up until now I have described method that uses only single training example at time to optimize the ANN. This method is known as stochastic gradient descent. In batch gradient descent, optimization is done by considering all training examples (m) and the goal is to minimize cost function J:

$$J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)} \tag{1.27}$$

Gradients calculated just from $\mathcal{L}^{(i)}$ might be noisy and thus, not as accurate than gradients calculated from J [36]. However, in the case of large datasets, using batch gradient descent might not be computationally feasible. Therefore, a method that is known as mini-batch gradient descent is usually used. It falls between the stochastic and batch gradient descent as it uses user defined amount (B) training examples. The cost function J_{mb} for mini-batch gradient descent is expressed as:

$$J_{mb} = \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}^{(i)} \tag{1.28}$$

1.5.3 Recurrent neural networks

Traditionally data is treated as a set of independent observations, i.e. training examples in data are independent and identically distributed (IID). In the case of sequential data IID assumption does not hold as the data is intrinsically ordered, i.e. samples in the sequence are related to each other. Time-series data (e.g. ECG,

stock prices) is a one common example of sequential data where samples are collected with respect to time dimension. However, other types of sequences also exist (e.g. text). Sequential datasets have one extra dimension when compared to traditional two-dimensional datasets, where rows represent observations and columns represent features. For example, in the case of ECG first dimension could represent the patient, second dimension the time and third dimension the features (lead voltages). I will refer to this extra time dimension with superscript of angle brackets. For example $(\mathbf{x}^{(i)<1>}, \mathbf{x}^{(i)<2>}, ..., \mathbf{x}^{(i)<T>})$ would refer to i*th* training example which is a time-series of length T where superscript angle brackets denote each individual time step of the time-series.

Recurrent neural networks (RNNs) are a subclass of ANNs that are specialized in sequence modelling. In theory, RNNs can utilize entire history of the input sequence when they predict output [38]. This is possible due to the recurrent connections that act as 'memory' of the previous inputs, and thus, previous inputs can affect to current output of the network. One of the key advantages of RNNs is the ability to process sequences with variable lengths. This gives more flexibility when compared to other ANN architectures where input and output dimensions are fixed and they need to be known beforehand [39]. This flexibility is due to the parameter sharing, i.e. same parameters are used in the all time steps of the input sequence [35].

RNNs can be used for many different use cases. For example, it is possible to produce prediction to every time step of the input sequence or just one prediction that describes the whole sequence. RNN can also produce an output sequence from an input that is not a sequence itself. Example of this is the image captioning, where combination of convolutional and recurrent neural network is used to produce sequence of text (caption) for a input image [40]. One more popular use is encoder-decoder model used in language translation. In this architecture, encoder codes input sequence (e.g. sentence in English) into the vector representation which is then decoded back to sequence (e.g. sentence in other language) [29].

Structure of simple RNN with just one hidden layer $\mathbf{h}$ (Fig. 1.13) is quite similar

to feedforward network with one hidden layer. It has the weights $\mathbf{W}_{xh}$ that connect inputs $\mathbf{x}$ to hidden layer and weights $\mathbf{W}_{ho}$ that connect hidden layer to output layer $\mathbf{o}$. Similarly there is also bias vectors $\mathbf{b}_h$ and $\mathbf{b}_o$ that connect to hidden layer and output layer respectively. However, besides these parameters RNNs also have a weight matrix $\mathbf{W}_{hh}$ that connects the hidden layer output from previous time step *t-1* to current time step *t*. This is the recurrent connection that passes information from the previous time steps. At the beginning of the sequence, there is no previous time steps and thus, $\mathbf{W}_{hh}$ is initialized to small random values or zeros [31]. Note that with weight matrices I used the subscript where the first letter denotes the origin layer and second letter denotes the target layer. I also do not use letter $\mathbf{a}$ with superscript to refer different layers anymore, layers $\mathbf{a}^{[0]}$, $\mathbf{a}^{[1]}$ and $\mathbf{a}^{[2]}$ are referred with letters $\mathbf{x}$, $\mathbf{h}$ and $\mathbf{o}$ respectively. This simplifies notation and removes need for multiple superscripts.

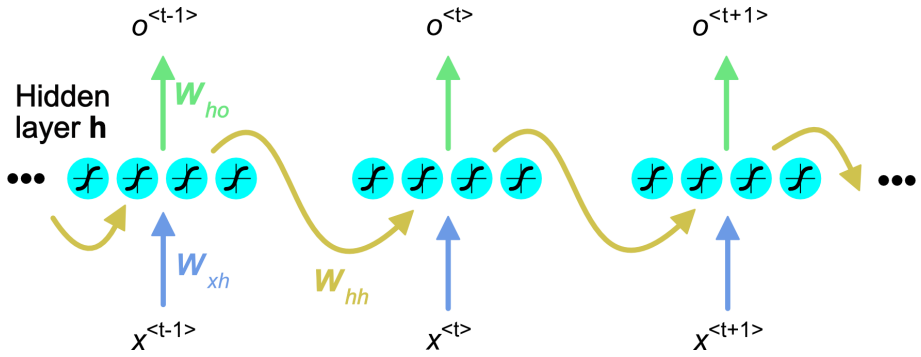

Figure 1.13: Simple recurrent neural network. Arrows indicate weight matrices (full connections) between the layers. Same weight matrices are repeatedly used in each time step. For the sake of simplicity, bias vectors are not shown.

Like before, I explain steps used in training RNN just in the case of singe training example $\mathbf{x}^{(i)}$ Training algorithm for RNNs is called backpropagation through time [41]. In this approach both input signal and error signal (gradients) flow through different time steps, but otherwise it is like regular backpropagation. At first I describe the forward phase. Input $\mathbf{z}^{<t>}$ to the hidden layer for a time step t is calculated by adding linear combination of inputs to the bias vector and to the linear combination of activations from the previous time step *t-1*:

$$\mathbf{z}^{<t>} = \mathbf{W}_{xh}\mathbf{x}^{<t>} + \mathbf{W}_{hh}\mathbf{h}^{<t-1>} + \mathbf{b}_h \tag{1.29}$$

Like before, activations of the hidden layer $\mathbf{h}^{<t>}$ are simply calculated by applying activation function g (usually *tanh*) to every component of the layer input $\mathbf{z}^{<t>}$:

$$\mathbf{h}^{<t>} = g_h(\mathbf{z}^{<t>}) \tag{1.30}$$

Output for the time step t is then calculated as:

$$\mathbf{o}^{<t>} = g_o(\mathbf{W}_{ho}\mathbf{h}^{<t>} + \mathbf{b}_o) \tag{1.31}$$

Because the whole sequence acts as training example, the loss is calculated by summing up the losses from different time steps of the sequence:

$$\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}^{<t>} \tag{1.32}$$

Gradients of loss with respect to different RNN parameters are calculated similarly by summing up the gradients at every time step. For example, gradient with respect to $\mathbf{W}_{hh}$ would be calculated as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{<t>}}{\partial \mathbf{W}_{hh}} \tag{1.33}$$

Gradient of loss with respect to $\mathbf{W}_{hh}$ for time step t is calculated by utilizing the chain rule:

$$\frac{\partial \mathcal{L}^{<t>}}{\partial \mathbf{W}_{hh}} = \sum_{k=1}^{t} \frac{\partial \mathcal{L}^{<t>}}{\partial \mathbf{o}^{<t>}} \frac{\partial \mathbf{o}^{<t>}}{\partial \mathbf{h}^{<t>}} \frac{\partial \mathbf{h}^{<t>}}{\partial \mathbf{h}^{<k>}} \frac{\partial \mathbf{h}^{<k>}}{\partial \mathbf{W}_{hh}} \tag{1.34}$$

Note that chain rule is also used to term $\frac{\partial \mathbf{h}^{<t>}}{\partial \mathbf{h}^{<k>}}$. This can result long chains if the sequence is long, e.g. if *t=50* and *k=1*:

$$\frac{\partial \mathbf{h}^{<50>}}{\partial \mathbf{h}^{<1>}} = \frac{\partial \mathbf{h}^{<50>}}{\partial \mathbf{h}^{<49>}} \frac{\partial \mathbf{h}^{<49>}}{\partial \mathbf{h}^{<48>}} \cdots \frac{\partial \mathbf{h}^{<2>}}{\partial \mathbf{h}^{<1>}} \tag{1.35}$$

Multiplicative factor of the term $\frac{\partial \mathbf{h}^{<t>}}{\partial \mathbf{h}^{<k>}}$ can make gradients either vanish or explode [31]. Gradients vanish or explode because RNNs have to propagate gradients backwards over long sequences. If small values are repeatedly used in matrix multiplication, gradient will shrink and eventually it will vanish. This prevents states that are too far from the current time step to contribute gradient computation [42]. Therefore, RNNs gradually forget the first inputs of the sequence and are not able to learn the long-term dependencies. Vice versa, usage of too large values in matrix multiplication leads to exploding gradients. Problems with unstable gradients are not just restricted to RNNs but any deep ANN might suffer from them [29].

RNNs can have more complex and powerful architectures than described earlier. Like with feedforward networks, adding depth to RNN network can increase its performance. This have been experimentally shown e.g. by Graves et al. [43]. In multi-layer architecture, the output sequence produced by one layer is fed as input sequence to the layer above it. It can be thought that the lower layers extract representations from the raw input that are more usable at the higher layers [35].

Standard RNNs process input only in chronological order. This means that state at time t contains information only from previous time steps. In some cases this is not enough and it is beneficial also to know the future context. Bidirectional recurrent neural networks [44] can utilize information both from past and the future. Bidirectional RNN is a combination of two RNNs that are connected to same output layer. One RNN iterates the input sequence in chronological order while the other iterates it in reversed order. Therefore, output layer is able to produce presentation that depends on both past and future context.

Like mentioned before, RNNs can be thought to possess some sort of 'memory'. The part of the network which acts as memory i.e. contains information from previous time steps is known as memory cell [29]. In a simple RNN the hidden layer $\mathbf{h}^{<t>}$ acts as memory cell. More complex and powerful RNN memory cells have been

also proposed. The Long Short-Term Memory (LSTM) cell proposed by Hochreiter and Schmidhuber in 1997 [45] is one of the most popular RNN variants (Fig. 1.14). Unlike regular RNN, LSTM is able to learn long-term dependencies. This is possible due to separate cell state $\mathbf{c}^{<t>}$ where information can be stored over longer time spans. Flow of information within the LSTM cell is controlled by forget, input and output gates that are element-wise multiplication operations. Fully connected layers act as gate controllers, they are defined in respective order as follows:

$$\mathbf{f}^{<t>} = \sigma(\mathbf{W}_{xf}\mathbf{x}^{<t>} + \mathbf{W}_{hf}\mathbf{h}^{<t-1>} + \mathbf{b}_f) \tag{1.36}$$

$$\mathbf{i}^{<t>} = \sigma(\mathbf{W}_{xi}\mathbf{x}^{<t>} + \mathbf{W}_{hi}\mathbf{h}^{<t-1>} + \mathbf{b}_i) \tag{1.37}$$

$$\mathbf{o}^{<t>} = \sigma(\mathbf{W}_{xo}\mathbf{x}^{<t>} + \mathbf{W}_{ho}\mathbf{h}^{<t-1>} + \mathbf{b}_o) \tag{1.38}$$

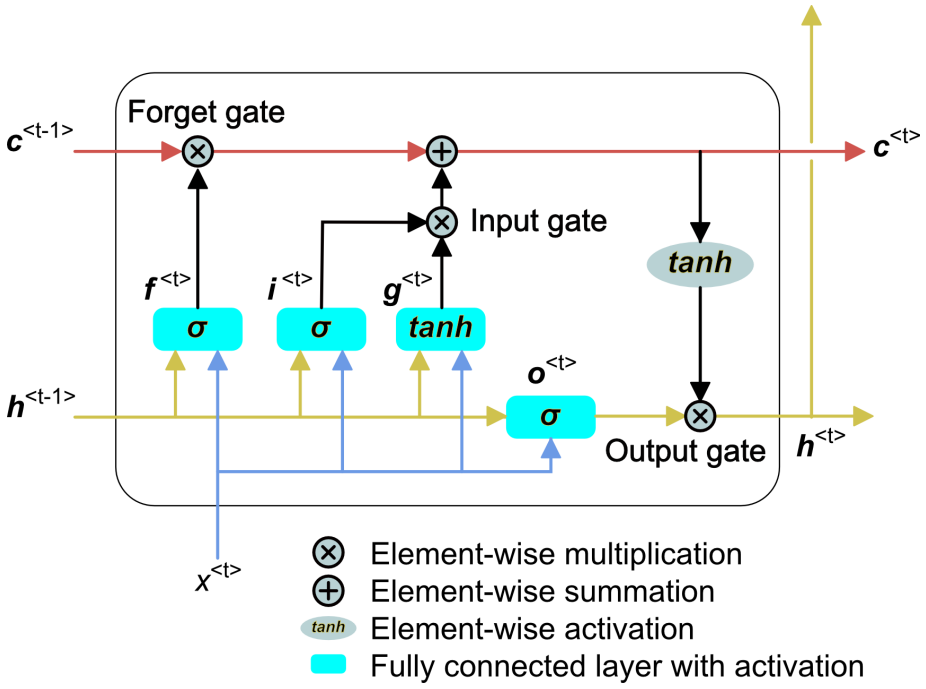

Figure 1.14: Structure of the LSTM cell. Modified after Raschka and Mirjalili [31].

Since the weights parameterizing the gates are learned during training, it is impos-

sible to know exactly what operations these gates actually do [46]. However, I will describe next the roles and functions that are commonly assigned to these gates. All gate controllers use sigmoid activation (σ), which produces outputs that range from 0 to 1. This means that e.g. the forget gate can completely erase the cell state $\mathbf{c}^{<t-1>}$ if gate controller $\mathbf{f}^{<t>}$ outputs just zeroes. Hence, it can be thought that the forget gate controls what information from $\mathbf{c}^{<t-1>}$ can pass thought and what needs to be erased. Similarly, the input gate controls what information from the main layer $\mathbf{g}^{<t>}$ is saved to cell state $\mathbf{c}^{<t>}$. Main layer $\mathbf{g}^{<t>}$ is analogous to the hidden layer $\mathbf{h}^{<t>}$ of a simple RNN, and thus, its role is to analyze current input $\mathbf{x}^{<t>}$ with previous hidden state $\mathbf{h}^{<t-1>}$ [29]. Main layer $\mathbf{g}^{<t>}$ is defined as:

$$\mathbf{g}^{<t>} = tanh(\mathbf{W}_{xg}\mathbf{x}^{<t>} + \mathbf{W}_{hg}\mathbf{h}^{<t-1>} + \mathbf{b}_g) \tag{1.39}$$

Gate controllers $\mathbf{f}^{<t>}$ and $\mathbf{i}^{<t>}$ are used to calculate cell state $\mathbf{c}^{<t>}$ as follows:

$$\mathbf{c}^{<t>} = \mathbf{f}^{<t>} \odot \mathbf{c}^{<t-1>} + \mathbf{i}^{<t>} \odot \mathbf{g}^{<t>} \tag{1.40}$$

The role of output gate is little different than other gates as it does not affect to the cell state $\mathbf{c}^{<t>}$ but rather just determines what is read from it and what parts of the cell state $\mathbf{c}^{<t>}$ are outputted as hidden state $\mathbf{h}^{<t>}$. While $\mathbf{c}^{<t>}$ can be thought to represent long-term state, $\mathbf{h}^{<t>}$ can be thought as more short-term state [29]. Before output gate, cell state $\mathbf{c}^{<t>}$ is scaled by applying *tanh* activation element-wise and thus, hidden state $\mathbf{h}^{<t>}$ is calculated as follows:

$$\mathbf{h}^{<t>} = \mathbf{o}^{<t>} \odot tanh(\mathbf{c}^{<t>}) \tag{1.41}$$

Chapter 2

Material and methods

2.1 Data sets

Four different datasets were used in this work, MIT-BIH Arrhythmia database [47], [48], MIT-BIH Noise Stress Test database [7], [48], Glasgow University Database (GUDB) [49] and small dataset of seven subjects [50]. Former two were used for model training while latter two were used for evaluation. All of the data sets except the small one are publicly available.

2.1.1 Training data sets

MIT-BIH Arrhythmia database was the first generally available database for arrhythmia detector evaluation. It has been widely used to benchmark R-peak or QRS detection algorithms and for basic research of cardiac dynamics. Database has a total number of 48 half-hour ECG records from 47 subjects. Each record is two channel ambulatory ECG record that has been digitized at 360 Hz. For most of the records one channel is modified limb lead II (electrodes are on the torso) while the other is V1.

MIT-BIH Noise Stress Test database contains three half-hour recordings of noise: baseline wander, muscle artifact and electrode motion artifact. These noise records

have been constructed by recording noise with electrode placement where ECG is not visible and then concatenating segments of similar noise type.

2.1.2 Evaluation data sets

GUDB contains ECG records from 25 subjects that are performing different tasks (sitting, solving maths test, walking, operating a hand bike and jogging) for duration of 120 seconds. ECG signals have been recorded using an Attys Bluetooth data acquisition board that has a sampling rate of 250 Hz. All ECG records in the database have been annotated at sample precision. Two different kind of setups, loose cables (standard Einthoven leads I, II, III) and chest strap have been used to record ECG for each task. In this work, chest strap and Einthoven lead II from the loose cables setup are used. Therefore, a total number of 250 (25x5x2) different ECG records are available with these conditions. However, 21 of the records have no annotations, and thus, only 229 ECG records are available for evaluation.

The small dataset is a part of larger database that is collected by University of California, Irvine researchers. From now on this dataset is referred as UCI dataset. One channel ECG signals in the UCI dataset have been recorded from postoperative patients during re-examination by using portable biopotential acquisition device [51]. In total, these seven records have 103 minutes of Lead I ECG that has been sampled at 500 Hz. Nearly 19 minutes of ECG recordings are very noisy. Annotations for UCI data set were created by fellow researcher who used threshold-based automatic R-peak detection and manual correction of detection errors.

To evaluate how detectors perform at different degrees of noise, one ECG record (No. 1) from the UCI dataset was selected and different degrees of Gaussian noise were added to it by a fellow researcher (Fig. 2.1). Noisy ECG examples were generated with a signal-to-noise ratios (SNRs) of 20, 10, 5, 1, 0.5, 0.4, 0.3, 0.2 and 0.1. Sliding 1-second window was used and Gaussian noise was added by controlled linear SNR=P_{signal}/P_{Noise}. The detrended raw ECG samples within the window were the signal base and P_{Signal} was their total power.

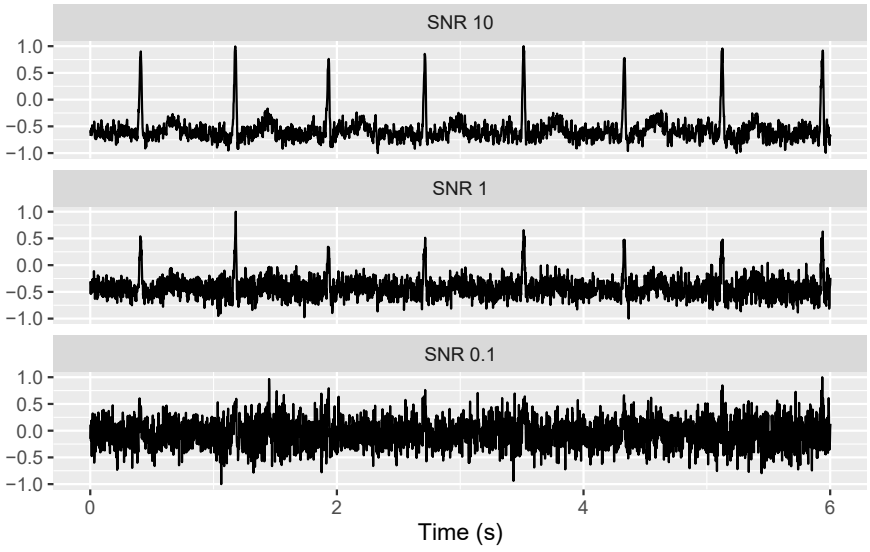

Figure 2.1: Examples of ECG record with three different degrees of gaussian noise. Signals have been normalized to range [-1,1].

2.2 LSTM based method for R-peak detection

2.2.1 Overview of the Development process

LSTM based method for R-peak detection was developed in two separate stages (Fig. 2.2). At first stage, LSTM model was trained by using publicly available MIT-BIH arrhythmia and MIT-BIH Noise stress test databases. Training data was created by using data augmentation where ECG signals were mixed with typical ECG noise sources. Separate generator function was developed to do this task. In a second stage, two functions, a wrapper function and a filtering function were created. Former uses trained model to detect R-peak locations while the latter removes unnaturally closely occurring R-peaks by utilizing model predictions and distances between successive R-peaks.

2.2.2 Data augmentation

Lots of noisy ECG signals are needed as training data if the goal is to train the network to detect R-peaks from noisy data. This means that there are two problems,

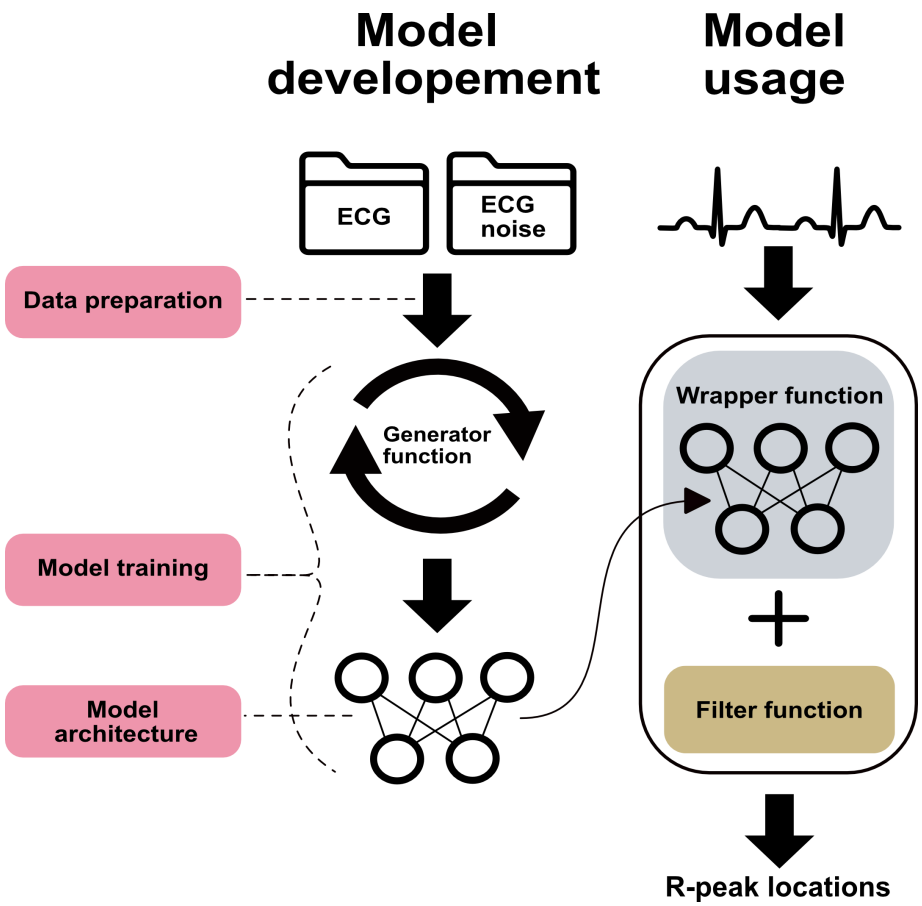

Figure 2.2: Overview of the development process [2].

the need of noisy training data and labeling the training data. If the ECG signal is saturated with complex noise it might not be possible to label R-peak locations with high confidence. To solve both problems, training data can be augmented, i.e. noisy ECG signal can be simulated. Training examples are created by adding variable amounts of different noises to noise free ECG signals. This way large amounts of diverse training examples can be created, and R-peak locations can be known even in the presence of complex noise. Each training example is constructed by the generator function roughly by the following five steps (Fig. 2.3):

1. Select randomly one ECG record from the database.
2. Select randomly 1000 sample window from the ECG record.
3. Make sure that all beats in the window are normal type.
4. Normalize ECG window to [-1, 1] range and create labels for it

5. Generate noise and add it to the selected window.

From these the steps 1-4 are trivial but step 5 contains more details, and thus, needs a more detailed description. In step 5 randomization is also used in noise generation. At first, 1000 sample windows are selected randomly from the both noise sources (baseline wander, muscle artifact). Then the selected noise windows are multiplied by random number. Random multipliers are drawn from uniform distributions of (0,10) for baseline wander and (0,5) for muscle artifact. Multiplication is done to add variation to the magnitude of the noise. The next step is to choose the noise type that is added from the following three categories: Baseline wander, muscle artifact or combination of these two. After selection, 60 Hz sine wave that simulates power line interference is added to the noise. Sine wave is also multiplied by random number from uniform distribution of (0, 0.5) to alter its magnitude. After noise is create it is added to the ECG signal which has been normalized to [-1, 1] range. After addition the noisy ECG signal is normalized again to the [-1, 1] range.

Labeling scheme used in step 4 is binary, each time step of the window is labeled either as zero or one (Fig. 2.4). Zeros indicate time steps without R-peaks while ones correspond to time steps where R-peak is present. Ones were added also two time steps before and after the the R-peak. This makes labels slightly more balanced and helps with model training.

Generator function generates high amounts of differing training examples (Fig. 2.5). They can be almost noise free or saturated with complex noise or anything between these two extremes.

2.2.3 Model architecture and training

Model was built and trained with TensorFlow deep learning framework (2.0 RC) [52] by using high-level Keras API [53]. Sequential model architecture is shown in (Fig. 2.6). Model has sequence of two bidirectional LSTM layers and one dense layer. Binary cross-entropy was used as loss function and Adam [54] was used as

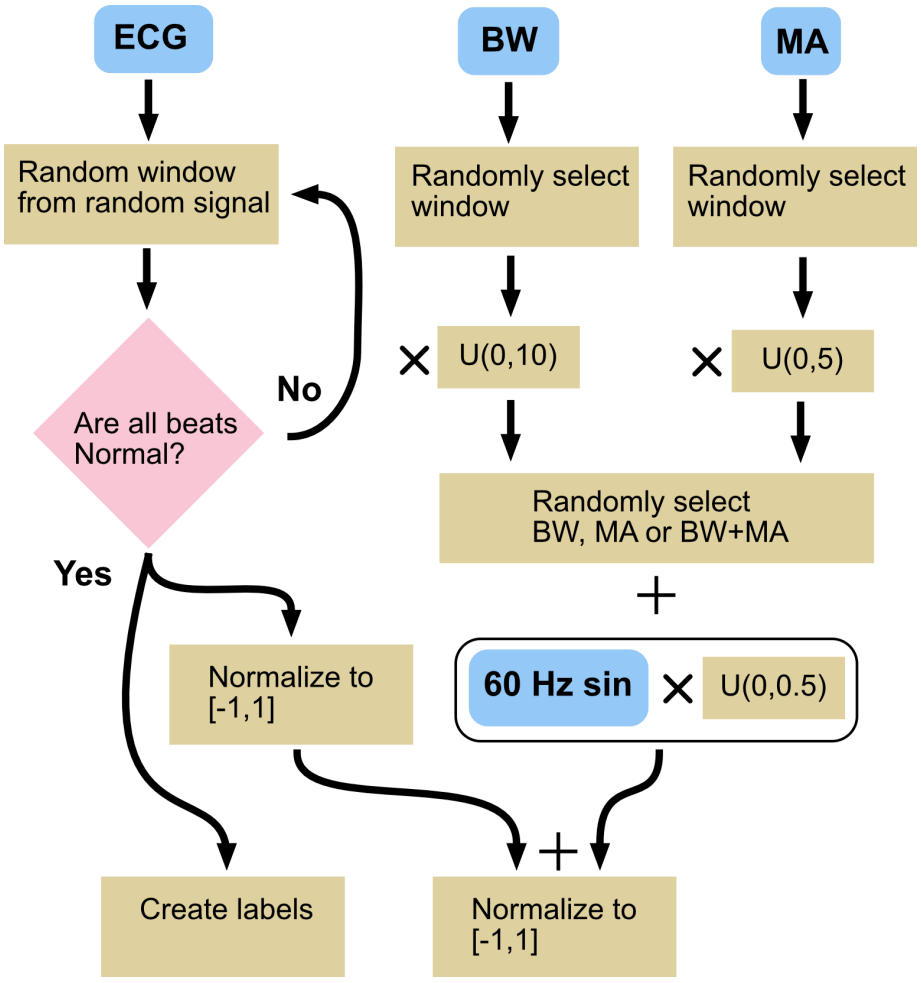

Figure 2.3: Schematic illustration of the generator function. Graph illustrates how single training instance is created. BW:baseline wander, MA:muscle artifact. U means uniform distribution where multiplier is randomly drawn [2].

optimizer. Model was trained a total number of 150 epochs, with 40 steps per epoch and batch size being 256. With aforementioned settings, a total number of 1 536 000 (150x40x256) different training examples were generated and used for model training.

2.2.4 Model usage

When model is used to make predictions, it expects its inputs to be in a similar form that was used in training phase. This means that the input ECG needs to be reshaped into 3D tensor where first axis is the batch dimension, second axis

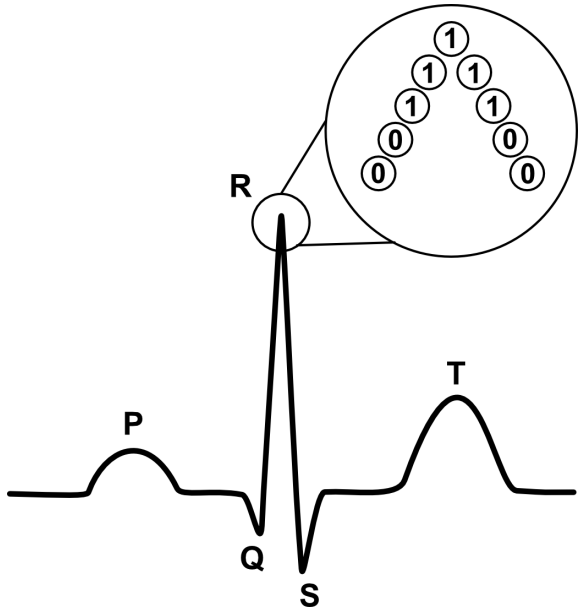

Figure 2.4: Illustration of the used labeling scheme. Each R-peak is marked by 5 ones while rest of the time steps are zeroes [2].

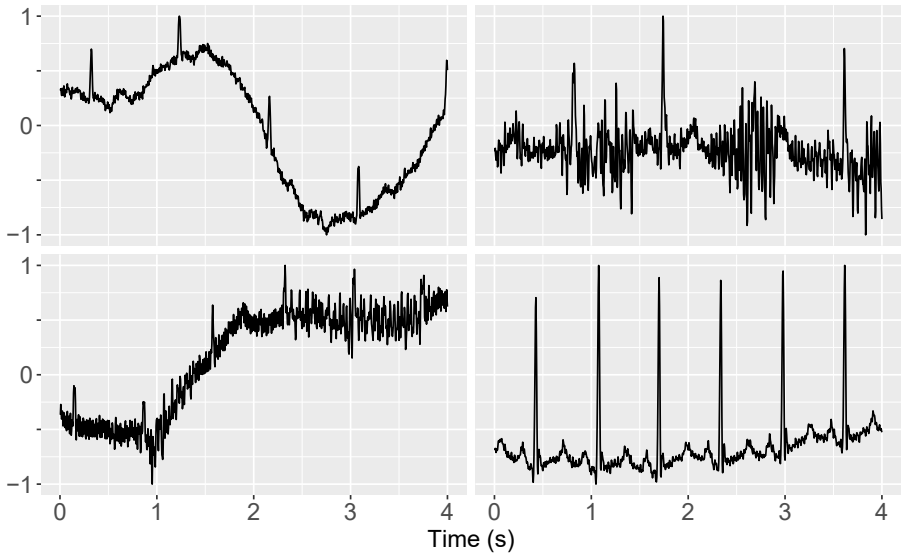

Figure 2.5: Training examples produced by the data augmentation process (generator function). Training examples can be almost noise free (lower right) or contain variable amounts of noise from different sources.

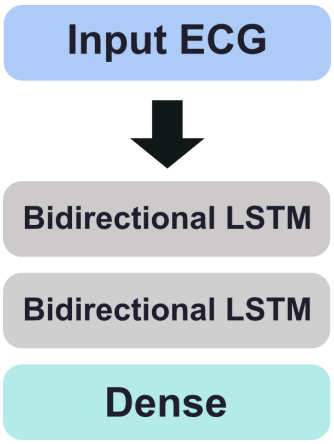

Figure 2.6: Schematic illustration of the used network architecture [2].

represents time steps and third axis corresponds to features. Outputs of the LSTM model are also in the same form as the inputs. Because of this, "wrapper code" was developed which processes the input ECG to suitable form and extracts R-peak locations from the model outputs. The code was developed by using object oriented paradigm and its structure is illustrated in Fig. 2.7.

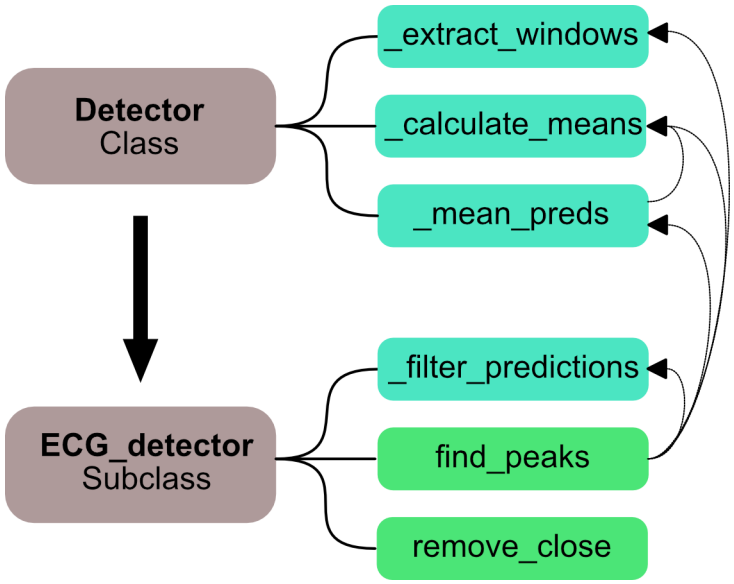

Figure 2.7: Structure of wrapper code. Private methods are as mint green while public methods are as green. Small arrows show if a method utilizes other method.

Class `ECG_detector` was designed to process model inputs and outputs. It is inherited from the `Detector` class that contains more general methods that are not restricted just to ECG signals. These methods could be used in future e.g. when

processing RR interval time series. Hence, this kind of design allows future extensibility.

The `Detector` class contains methods `_extract_windows`, `_calculate_means` and `_mean_preds` that are intended for private use. Method `_extract_windows` divides input signal into overlapping windows. Methods `_calculate_means` and `_mean_preds` calculate mean values from overlapping predictions. R-peak locations are extracted by private `_filter_predictions` method. The whole peak detection algorithm is executed by calling public `find_peaks` method. It uses previously defined methods to carry out subtasks in needed order to process model inputs and outputs. In addition, there is public `remove_close` method whose purpose is to filter out unnaturally closely occurring R-peaks.

The work done by `find_peaks` method can be roughly summarized into eight phases:

1. If needed, resample ECG signal to 250 Hz.
2. Extract overlapping segments (windows) from the ECG signal.
3. Use LSTM model to make predictions.
4. Calculate average values for overlapping predictions.
5. Extract R-peak locations from predicted probabilities.
6. Resample R-peak locations to original sampling frequency
7. Use original ECG signal to correct R-peak locations.
8. If duplicate R-peaks are present, remove them.

Phases 1 and 2 cover the input preprocessing tasks. The purpose of first step is quite clear, the input ECG needs to have same sampling frequency which was used during model training. In a second phase, input ECG is split into overlapping windows. Splitting is done by sliding the 1000 time step window by user defined step (stride). The stride value can be 100, 200, 250 or 500 time steps that corresponds to 10, 5, 4 or 2 time overlap respectively. This increases computational cost but at the same time it allows model to see time steps at different context and thus, improves R-peak detection. To achieve same amount of overlap for each time step, padding is used at

the both ends of signal. Median value of 1000 closest time steps is used as padding value. After splitting, extracted ECG segments are normalized to range of [-1, 1] and data is reshaped into 3D tensor that can be fed to the model. Model makes predictions for every ECG segment in the 3D tensor (Fig. 2.8) and then returns 3D tensor that contains predictions.

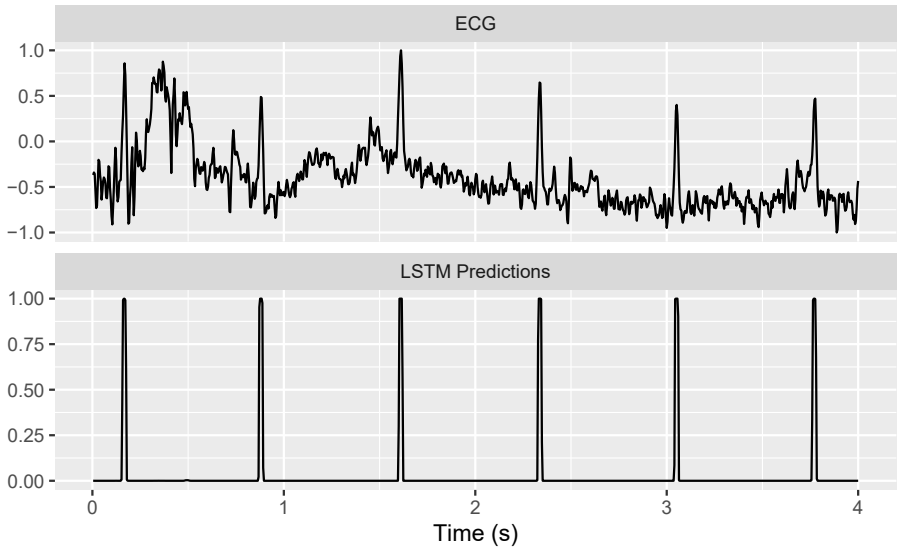

Figure 2.8: Normalized ECG segment (above) and corresponding predictions made by the LSTM model (below).

After LSTM model has been used for prediction in phase 3, phases 4-7 are executed to extract R-peak locations from the model predictions. At first, averages are calculated from overlapping predictions in phase 4. Then R-peak locations can be extracted from average predictions in phase 5. This is done by using probability threshold which determines the time steps that belong to R-peaks. Only time steps whose average prediction is above user defined probability threshold are kept. This doesn't yet give exact R-peak locations as time steps belonging to R-peaks occur in groups i.e. one R-peak is described by multiple consecutive (e.g. 5) time steps. As R-peaks are "peaks" it is natural to assign exact R-peak location to the corresponding local maximum of the ECG signal. This is done by moving each time steps to the local maximum that is within five time steps. If five or more time steps end to same location (e.g. local maximum) then that location is assigned as an R-peak.

Upon this point all of the work has happened with respect to ECG signal that has been resampled to 250 Hz in phase 1. To get R-peak locations in original sampling frequency, phase 6 is needed. R-peak locations can differ slightly in the two different sampling frequencies, therefore correction to local maxima with respect to original signal is done in phase 7. In some very rare cases there can occur duplicate R-peaks (i.e. two R-peaks in same location). Duplicates are removed in phase 8 if they occur.

2.2.5 Filtering false positives

After all R-peaks are extracted from model predictions, some unnaturally close R-peaks (false positives) might still occur. These can be e.g. noise peaks or pronounced T-waves that have been falsely detected as R-peaks. To get rid of these false positives, separate filtering function `remove_close` (Fig 2.9) was developed. It works very well when the aim is to maximize recall and low user defined probability threshold is used in phase 5. This produces high number of false positives, but most of them can be removed by filtering function. Filtering function works by removing R-peaks that are closer than user defined threshold distance (e.g. 200 ms). It returns array of R-peak locations where all R-peaks are at least threshold distance away from each other.

Filtering function uses average probabilities calculated in phase 4 as an aid to do the filtering. At first, set of R-peaks that occur within threshold distance are determined and this set is removed from set of all R-peaks. Then, these removed R-peaks are iterated over so that in each iteration R-peak with highest probability value is selected. Selected R-peak is then compared against the set of approved R-peaks. If distance of selected R-peak is higher than threshold distance to all R-peaks in the set of approved R-peaks, it is added to the set of approved R-peaks. If this is not the case, the selected R-peak is thrown away. Iteration continues until the set of removed R-peaks is empty. This simple algorithm is surprisingly efficient in the case where heart rate does not vary too much. Obvious drawback of this method is the fixed threshold distance value. With long records where heart rate is not stationary,

more advanced method with adaptive threshold distance is needed.

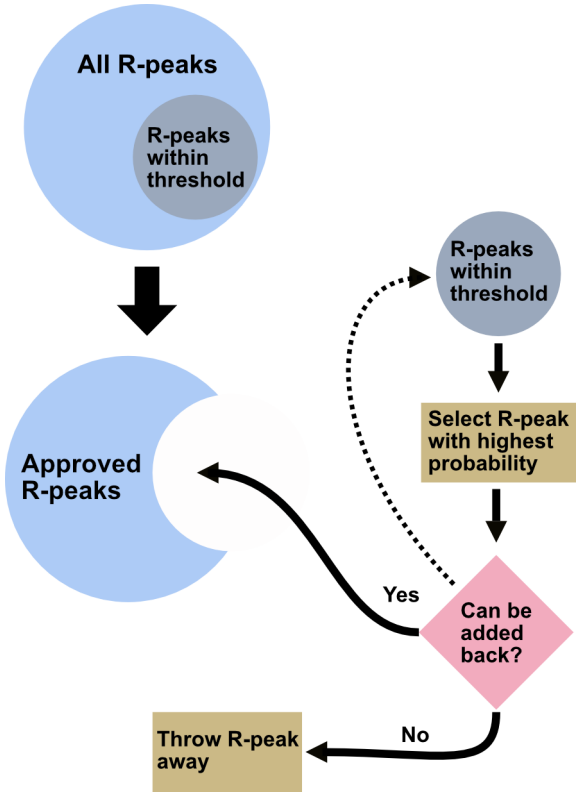

Figure 2.9: Working principles of the filtering function.

2.3 Evaluated QRS detectors

All reference algorithms are from the `py-ecg-detectors` (v1.0.2) [55] package that has been developed conjunction with the GUDB. Package contains implementations for seven QRS detection algorithms. These algorithms are: Pan-Tompkins [11], Hamilton [56], Christov [57], Elgendi [58], Kalidas [15], Engzee [59] with modifications by Lourenco et al. [60] and matched filter [49]. However, only six algorithms were used as one algorithm (matched filter) did not work properly. All reference algorithms were run with default settings.

LSTM based method was run by using 0.05 as user defined probability threshold, 250 as stride value and 300 ms as a threshold distance for filtering function. Except for threshold distance, these values are same that was used in publication of Laitala

et al. [2]. This time 50 ms smaller threshold distance is used with GUDB as some of the activities are very physical in nature and as a result heart rate can be much faster. With UCI dataset, the threshold distance is not altered.

2.4 Evaluation process

Common feature for majority of the detectors is the delay that they induce to the input ECG. Thus, predicted R-peak timestamps often occur later than the ground truth R-peak timestamps. This delay needs to be considered during evaluation and some tolerance needs to be allowed when true positives are determined. Friesen et al. [8] allowed delay of 88 ms during their evaluation of different detectors. Same 88 ms time tolerance is used in this work. However, in this work time tolerance of 88 ms is not restricted just to delay but it covers both preceding and following 88 ms with respect to true R-peak (Fig. 2.10). This difference in approach is due the used evaluation algorithm (`compare_annotations` function from the `wfdb` package was used).

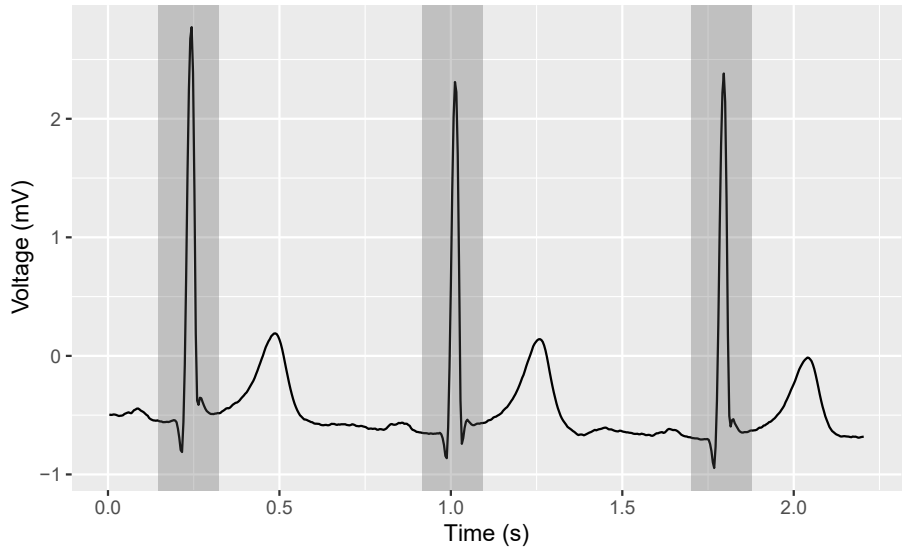

Figure 2.10: Tolerance distance (88 ms) before and after an R-peak is shaded. If predicted R-peak falls within this region it is considered as true positive

Precision, Recall and F1 score were used as performance metrics to evaluate different

detectors. These metrics are defined in equations 2.1-2.3. All of the used performance metrics are based on number of true positives, false positives and false negatives. True positives were already defined as a predictions that occur within 88 ms from the ground truth R-peaks. Predictions that fall outside of this threshold distance are considered as false positives. False negatives are ground truth R-peaks that do not have prediction within the threshold distance.

Precision is a measure of quality; in this work it describes how often the predicted R-peak is really the R-peak. Recall is a measure of quantity, in this work it tells how many R-peaks from all of the available R-peaks algorithm found. These metrics need to be used always together as it is very easy to maximize one of them in isolation but not both at the same time. Information from precision and recall can be summarized by calculating F1 score, which is the harmonic mean of the precision and recall.

$$\text{precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.1}$$

$$\text{recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.2}$$

$$\text{F1 score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.3}$$

However, these metrics alone do not necessarily give good enough picture of the performance as all true positives might not be equally important. This can be especially true from the point of heart rate variability (HRV) analysis. HRV means the variation of time intervals between consecutive heartbeats (R-peaks). These intervals are known as RR intervals (Fig. 2.11) or NN (normal-to-normal) intervals if only heart beats originating from sinus node depolarization are considered. HRV gives information from overall cardiac health and functioning of the autonomous nervous system [61]. By allowing tolerance when classifying correctly detected R-peaks also uncertainty is allowed. For example, allowing the 88 ms tolerance before

and after each R-peak means that in extreme cases the time interval between two consecutive R-peaks can be 176 ms longer or shorter than the ground truth interval. This uncertainty can clearly hamper the heart rate variability analysis.

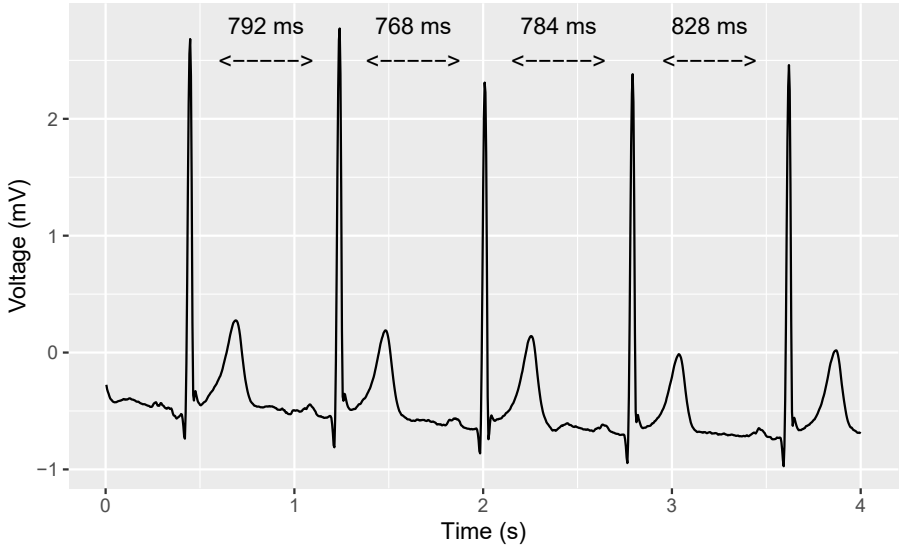

Figure 2.11: Illustration of RR intervals between consecutive R-peaks. In this case all all of the RR intervals are also NN intervals as all heart beats are normal.

To get an idea how well different QRS detection algorithms perform on the point of HRV analysis, distances from true positives to ground truth R-peaks are examined. This gives information how different algorithms behave and how suitable they are from the point of HRV analysis. Performance of the detectors in Time-domain HRV analysis is also briefly evaluated. Time-domain analysis of HRV focuses to quantify the amount of variability in monitoring periods that may range from ~2 min to 24 h [62]. One of the popular time-domain HRV metrics is the root mean square successive difference of intervals (RMSSD, equation 2.4), which gives information of vagal effect in HRV [62].

$$\text{RMSSD} = \sqrt{\frac{\sum_{i=1}^{N-1}(RR_i - RR_{i+1})^2}{N-1}} \tag{2.4}$$

RMSSD is calculated on RR intervals that are based on predictions (R-peaks) made by each detector. This RMSSD is then compared to RMSSD that is calculated

from RR intervals that are based on ground truth R-peaks. Only the ECG records corresponding the activities of sitting and maths are used in RMSSD evaluation. This is in line with the usual HRV analysis practice which is carried out under physiologically stable conditions. Because of this exclusion, only 99 ECG records are used for RMSSD evaluation.

UCI dataset is used for more traditional style evaluation of the QRS detectors as only precision, recall and F1 scores are calculated for different QRS detectors. Similar evaluation with the UCI dataset was already done by Laitala et al. [2]. However, in this evaluation more reference QRS detectors and different implementations from the same QRS detection algorithms are used. The tolerance distance (88 ms) is also stricter than the tolerance distance (100 ms) that was used by Laitala et al. [2]. Because sample precise annotation has been the focus with the GUDB, it is used to more detailed analysis. Besides the standard performance metrics, GUDB is also used to examine true positive distances and performances of the QRS detectors in time-domain HRV analysis.

Because focus of this work is to re-evaluate the performance of the LSTM based detector, a short section is also devoted to error analysis of this method. In this section the most significant types of failures and reasons behind them are examined briefly.

Chapter 3

Evaluation

3.1 Evaluation with the UCI dataset

Table 3.1 shows precision, recall and F1 scores for the detectors on the UCI dataset. It is clear that the LSTM based detector has the best performance across the whole dataset. All reference detectors show much worse performances and their performance vary a lot between different subjects. Results are very similar to the results obtained by Laitala et al. [2]. However, reference detectors (BioSPPy package [63]) used by Laitala et al. [2] gave slightly better performances than the reference QRS detectors that are used now.

Fig. 3.1 shows how different detectors perform at the different SNR ratios. It is again evident that the LSTM based detector has the best performance in all noise levels. Performance of the reference detectors decays very fast as SNR ratio decreases, while the LSTM based detector shows relatively good results even at the lowest SNR ratios. This is in line with the results of Laitala et al. [2]. However, current reference detectors show much worse results at low SNR ratios than the reference detectors used by Laitala et al. [2]. Interestingly, reference detectors could be also grouped into two different groups based on their performance with different noise levels. Performances of Hamilton and Elgendi decay much slower than other reference detectors as the SNR decreases. Similar behavior could be seen also in

the evaluation of Laitala et al. [2] where the Hamilton had clearly much better performance at low SNR ratios than rest of the reference detectors.

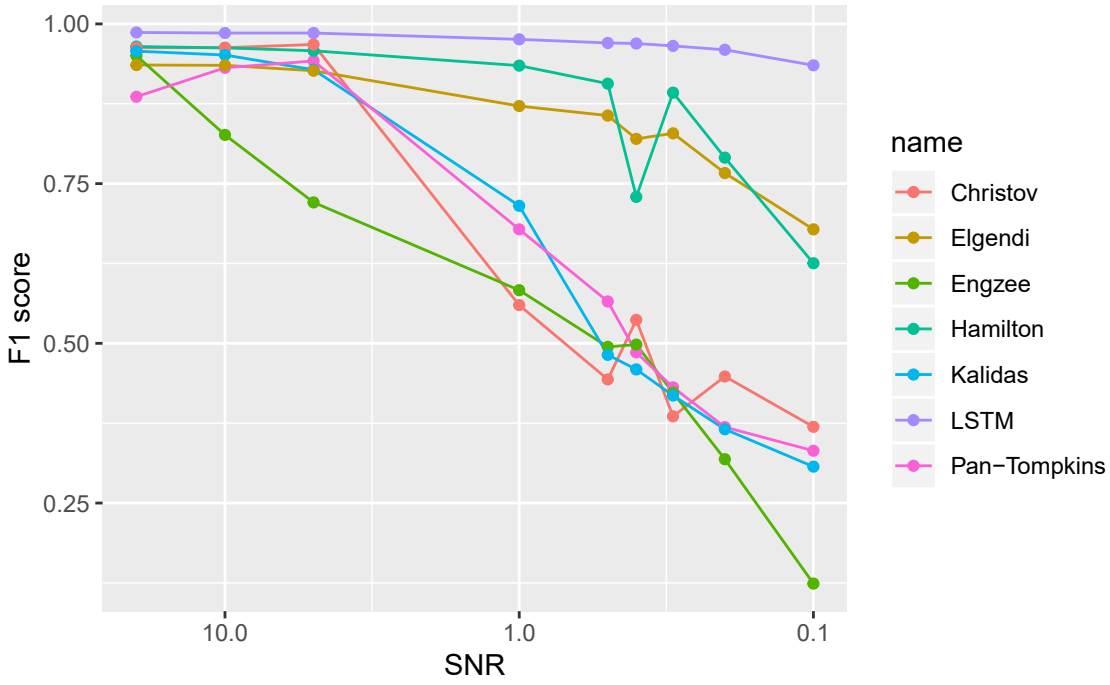

Figure 3.1: F1 scores of the detectors at different SNR ratios

Table 3.1: Performance of the detectors. The best scores for each metric within the same ECG record (in each row) are highlighted with bold font.

	LSTM			Pan-Tompkins			Hamilton			Christov			Engzee			Kalidas			Elgendi		
Subject	preci.	recall	F1	preci.	recall	F1	preci.	recall	F1	preci.	recall	F1	preci.	recall	F1	preci.	recall	F1	preci.	recall	F1
1	**0.991**	**0.99**	**0.991**	0.769	0.789	0.779	0.967	0.978	0.972	0.970	0.972	0.971	0.976	0.932	0.953	0.964	0.978	0.971	0.943	0.948	0.946
2	**1**	**1**	**1**	0.821	0.836	0.829	0.982	0.993	0.987	0.996	0.997	0.996	0.979	0.886	0.930	0.911	0.955	0.933	0.981	0.987	0.984
3	**0.995**	**0.997**	**0.996**	0.879	0.767	0.819	0.987	0.843	0.910	0.991	0.843	0.911	0.967	0.691	0.806	0.878	0.786	0.830	0.944	0.969	0.956
4	**0.998**	**0.999**	**0.998**	0.921	0.951	0.936	0.963	0.984	0.974	0.985	0.994	0.990	0.978	0.838	0.903	0.936	0.973	0.954	0.942	0.956	0.949
5	**0.995**	**0.987**	**0.991**	0.862	0.188	0.309	0.970	0.979	0.975	0.983	0.924	0.953	0.988	0.944	0.965	0.957	0.205	0.338	0.988	0.978	0.983
6	**0.999**	**0.999**	**0.999**	0.582	0.590	0.586	0.986	0.993	0.989	0.986	0.993	0.989	0.994	0.985	0.989	0.959	0.976	0.968	0.993	0.998	0.995
7	**0.995**	**0.995**	**0.995**	0.905	0.490	0.635	0.958	0.986	0.971	0.829	0.987	0.901	0.982	0.676	0.801	0.917	0.494	0.642	0.969	0.979	0.974

3.2 Evaluation with the GUDB

Precision and recall were calculated for each of the 229 ECG records. These records can be grouped by type of recording device and activity to ten different groups. Information from the performance metrics is summarized by the used recording setup and activity in the Fig. 3.2. Tables of F1 scores that cover all subjects, detectors and test setups (activity, recording device) are at appendix A.

Three detectors, Kalidas, LSTM based and Engeldi perform noticeably better than rest of the detectors. These detectors have very high average precision and recall within each group and also standard deviations of the metrics are negligible. From these three Kalidas has the best performance while LSTM based and Engeldi share the second place. LSTM based detector has the most notable drop in the performance with loose cables setup during jogging.

Rest of the detectors do not perform as well but they show interesting characteristics. For example, Christov has very high recall within each group but it has notably lower precision with high standard deviation. For Engzee this is reversed, i.e. it has generally high precision but recall is always lower with higher standard deviation. Hamilton does not perform as good with loose cables setup than it does with the chest strap setup. For Engzee and Pan-Tompkins this is the other way around as they show better performance with loose cables setup. Pan-Tompkins has clearly the worst performance from all of the detectors. Its behavior is also rather strange, as it shows better performance during jogging than more sedentary activities.

3.2.1 Examining true positive distances

Fig. 3.3 shows how true positive distances to the corresponding R-peaks vary between different detectors. From the methods, Engzee and LSTM based seem to be the most robust ones as most of the true positives are on point with the ground truth R-peaks. Rest of the algorithms are not so consistent as true positive distances to the R-peaks vary much more. Engeldi seems to be especially inconsistent as it shows

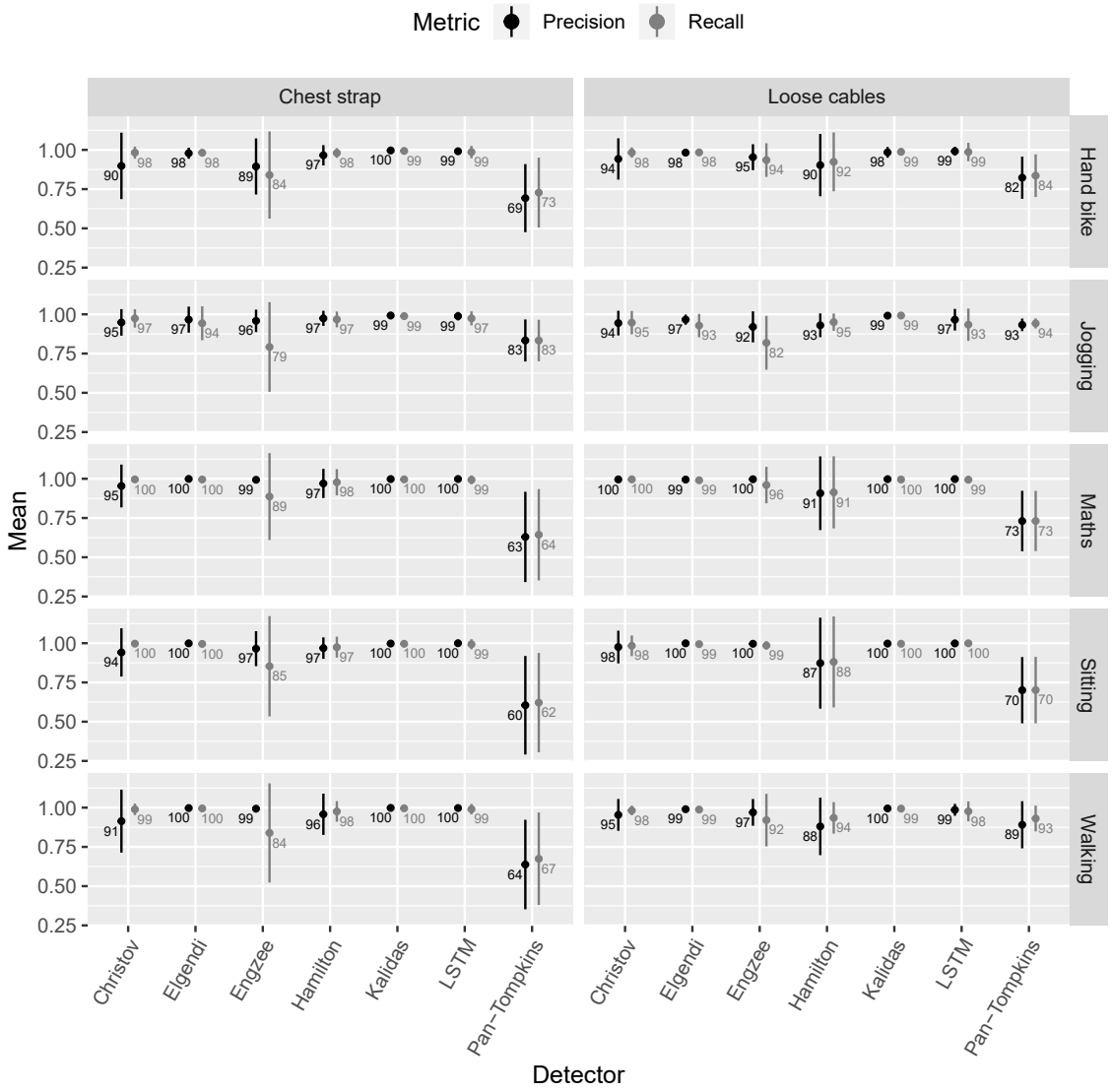

Figure 3.2: Precision and recall of different detectors for all different test setups (recording device and activity type combinations). Dots represent mean values and vertical bars standard deviations of the metrics within each combination.

large amounts of both low and high true positive distances.

However, Fig. 3.3 was made by considering true positives from all of the 229 ECG records. From the point of the HRV analysis true positives do not need to be precisely at the same position as R-peaks but rather their distance (bias) to the R-peaks should be constant within each record. Fig. 3.4 shows distributions for standard deviations of true positive distances that are calculated for each individual ECG record. Engzee, LSTM based detector and Kalidas stand out with high number of low standard deviations, i.e. true positive distances for these methods are relatively consistent within ECG records. Rest of the methods are not so consistent.

3.2.2 Performance in time-domain HRV analysis

According to [64] normal value for RMSSD is 27 ms (mean) with standard deviation of 12 ms. Fig. 3.5 shows how RMSSD values based on predictions differ from ground truth RMSSD. Values calculated on predictions of Pan-Tompkins and Hamilton differ notably from the ground truth RMSSD for majority of the records. Difference to the ground truth RMSSD is also usually very large, being usually more than two standard deviations away. Engzee and Christov produce results that are generally quite close to the ground truth but for several ECG records these methods fail completely. LSTM based detector shows similar behavior to Engzee and Christov. However, RMSSD values calculated from the predictions of the LSTM based detector are generally much more precise as they are usually almost equal to the ground truth. Kalidas behaves similarly to LSTM based detector but it shows small differences for higher number of records. Engeldi performs worse than Kalidas as it shows differences for higher number of records and the differences are also larger.

To summarize this information, absolute values are taken from the differences (errors) and then both mean and median are calculated for each method. Table 3.2 shows the mean and median absolute errors between predicted RMSSD and ground truth RMSSD for different methods. Kalidas shows smallest mean absolute error while the LSTM based detector has the smallest median absolute error.

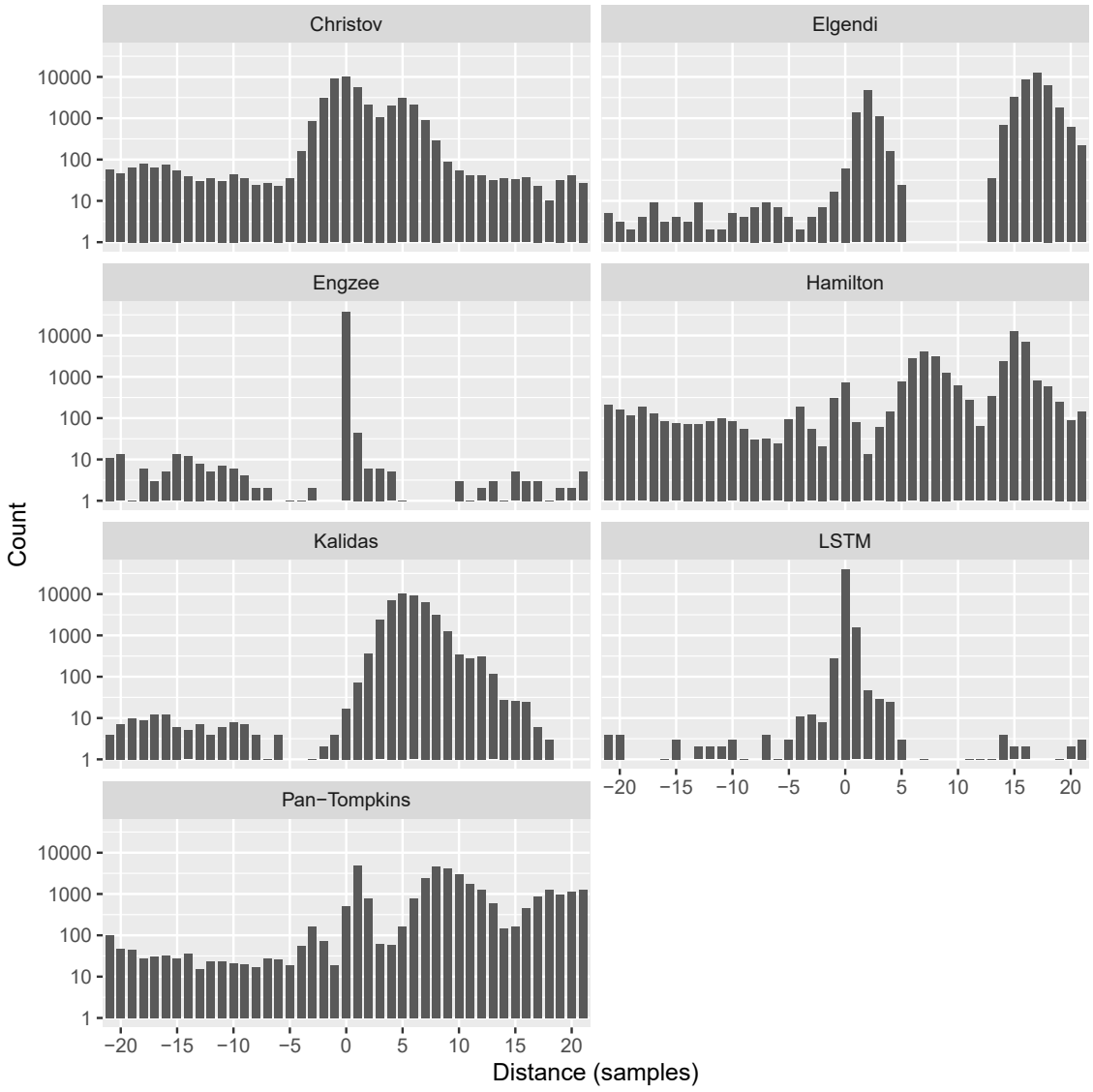

Figure 3.3: Distance of true positive to the R-peak. Results from all of 229 ECG records are combined.

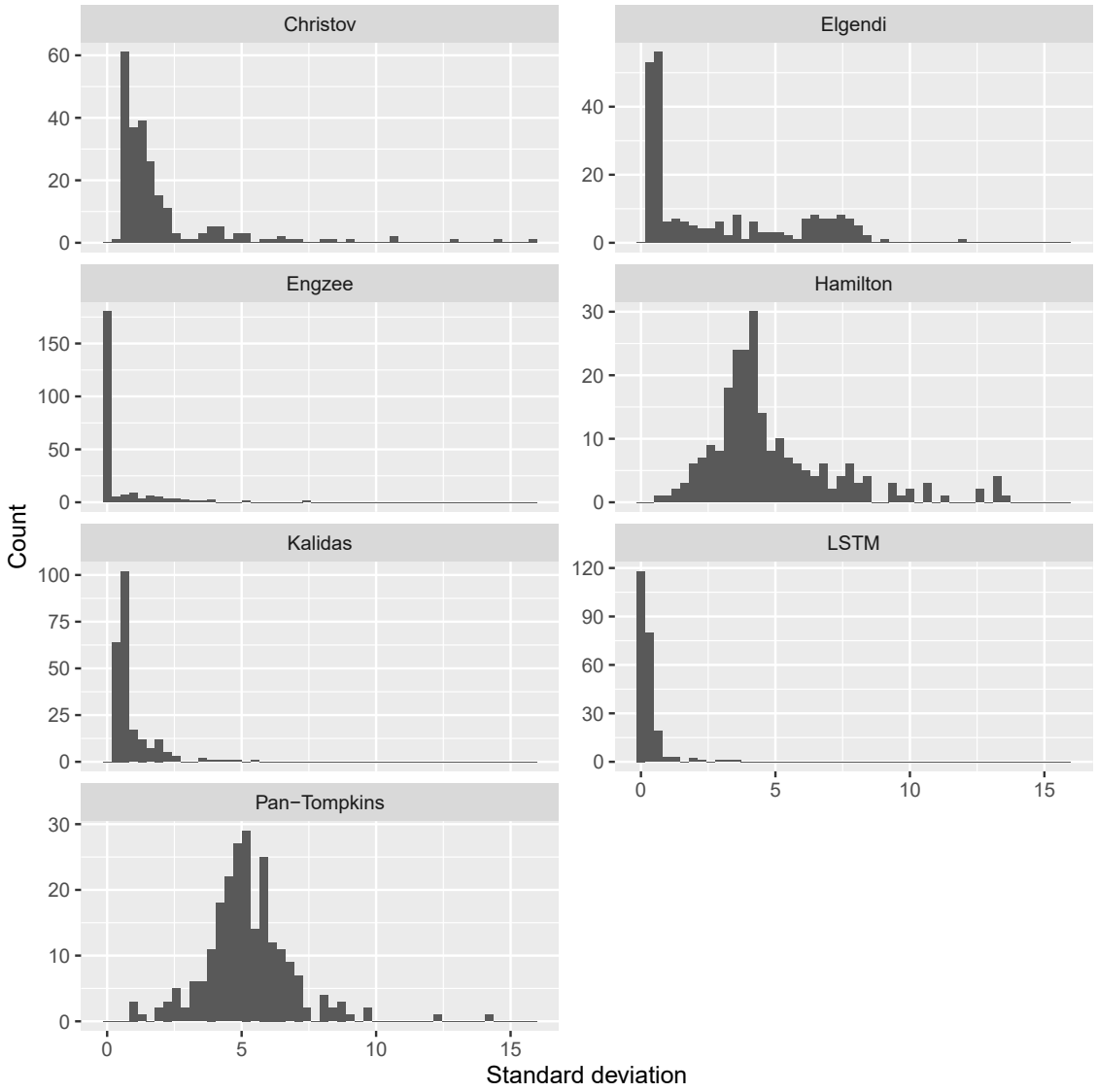

Figure 3.4: Histograms of standard deviations of true positive to R-peak distances within each record. High counts of low standard deviations indicate good performance.

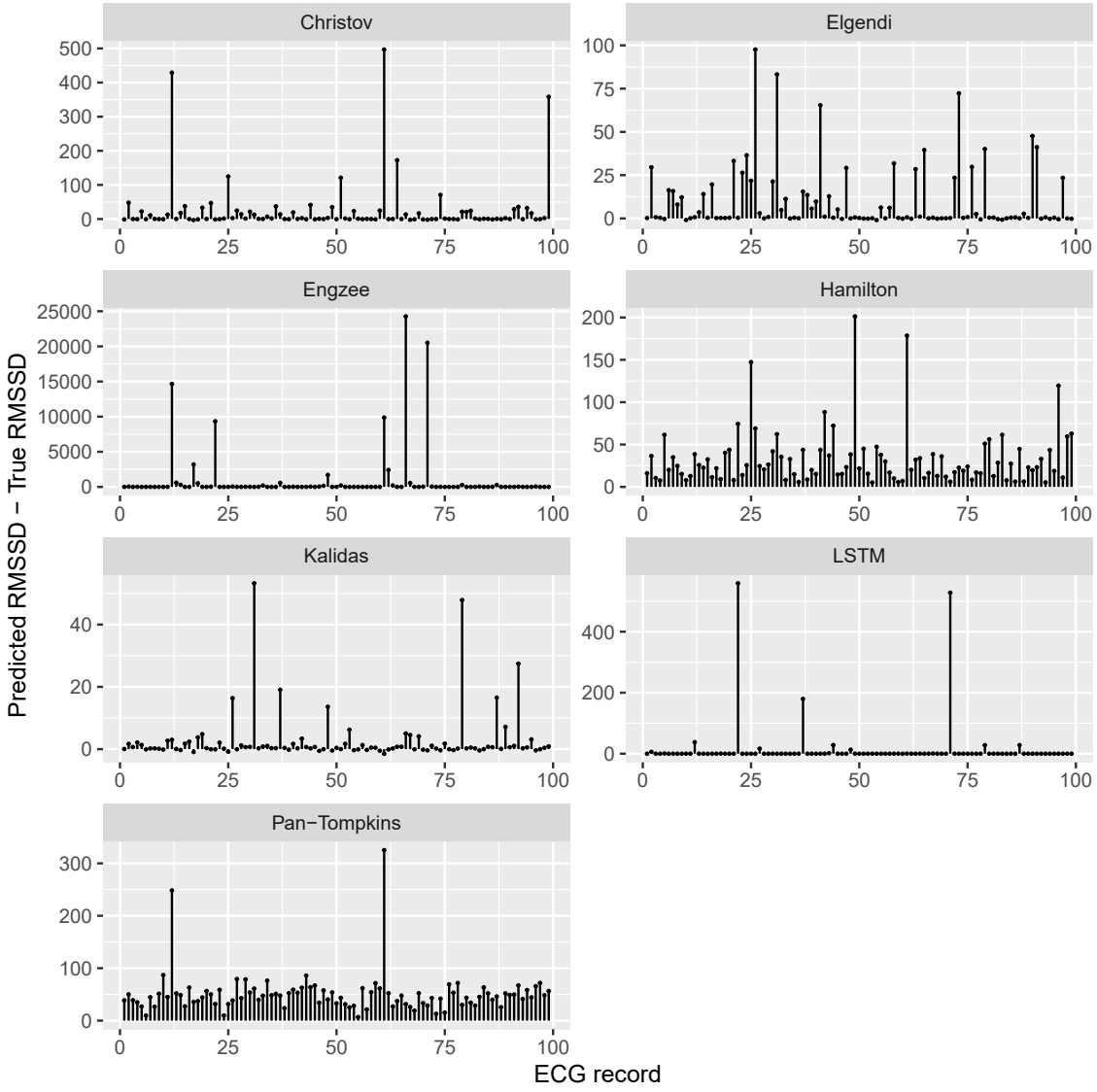

Figure 3.5: True RMSSD substracted from predicted RMSSD (y-axis) for activities sitting and maths (99 ECG records, x-axis)

Table 3.2: Mean and median absolute errors between predicted RMSSD and true RMSSD.

	Mean AE	Median AE
LSTM	14.48	0.05
Pan-Tompkins	50.62	47.53
Hamilton	32.48	23.12
Christov	25.66	1.32
Engzee	913.91	1.15
Kalidas	2.91	0.52
Elgendi	10.43	0.67

3.3 Error analysis of the LSTM based detector

LSTM based detector had excellent performance across the whole UCI dataset. However, it has still some weak points as it does not perform well in the vicinity of sharp base line changes that are caused by the electrode contact artifacts (Fig. 3.6).

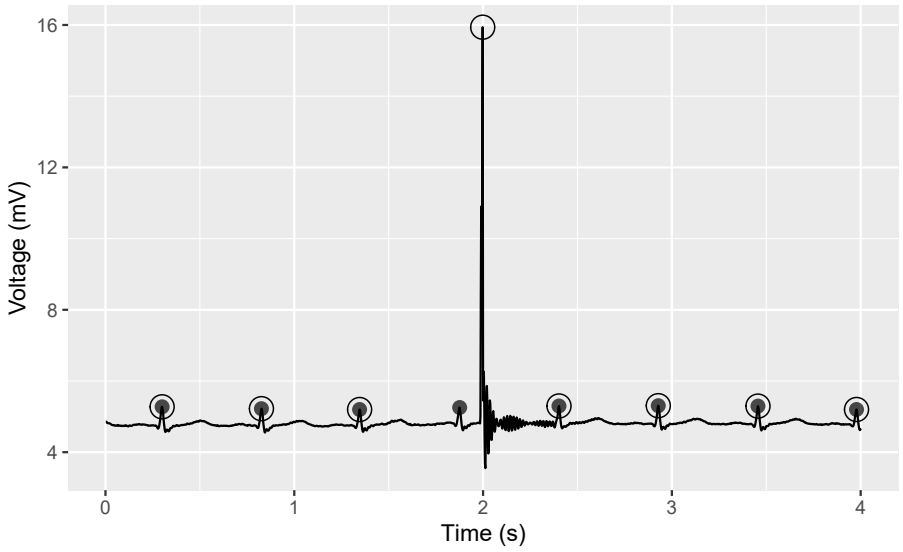

Figure 3.6: LSTM based detector fails in the vicinity of strong electrode contact artifact. Hollow circles indicate predicted R-peaks while grey dots mark the true R-peaks. ECG from the subject 5 (UCI dataset).

LSTM based detector had lowest precision and recall for the ECG records from the GUDB that were recorded during jogging by using loose cables setup. F1 scores for different detectors for this particular scenario are presented on table 3.3. Table has

only 9 subjects from 25, as there were no annotations for rest of the subjects. This is most likely due to very noisy ECG records that were produced in this setup.

From the table it is evident that the performance of LSTM based detector is quite good except for subjects 7 and 12. Without these records LSTM based detector would have performed almost as well as Kalidas, which is clearly the best one. By examining the detection failures in (Fig. 3.7) one can see that the failures are due to the very challenging ECG signal. Noise peaks have stronger amplitude than real R-peaks and they seem to occur at constant frequency. They are probably electrode motion artifacts that can be produced e.g. during running, when mechanical forces act on electrodes as a frequent manner.

Table 3.3: F1 scores, jogging with loose cables setup. The best F1 score within the same ECG record (in each row) are highlighted with bold font.

Subject	LSTM	Pan-Tompkins	Hamilton	Christov	Engzee	Kalidas	Elgendi
0	**0.99**	0.91	0.83	0.94	0.85	0.97	0.93
6	**1**	0.96	0.98	0.99	0.99	**1**	0.99
7	0.75	0.86	0.86	0.75	0.71	**0.97**	0.85
8	0.98	0.98	0.99	0.95	0.69	**1**	0.87
12	0.87	0.97	**1**	0.98	0.79	**1**	0.96
13	**1**	0.94	0.99	**1**	**1**	**1**	0.99
17	**1**	0.94	0.99	0.97	0.94	**1**	0.98
18	**1**	0.93	0.88	0.99	0.99	**1**	0.99
24	0.95	0.96	0.93	0.93	0.76	**0.99**	0.95

The artifact types seen on Figs 3.7 and 3.6 were not present on the ECG that was used to train the LSTM network. This largely explains the detection errors made by the LSTM based detector.

3.4 Discussion

LSTM based detector has the best performance with the UCI dataset and also very good performance with the GUDB. This makes it much more robust than reference detectors whose performance varies between the two datasets. Evaluation with dif-

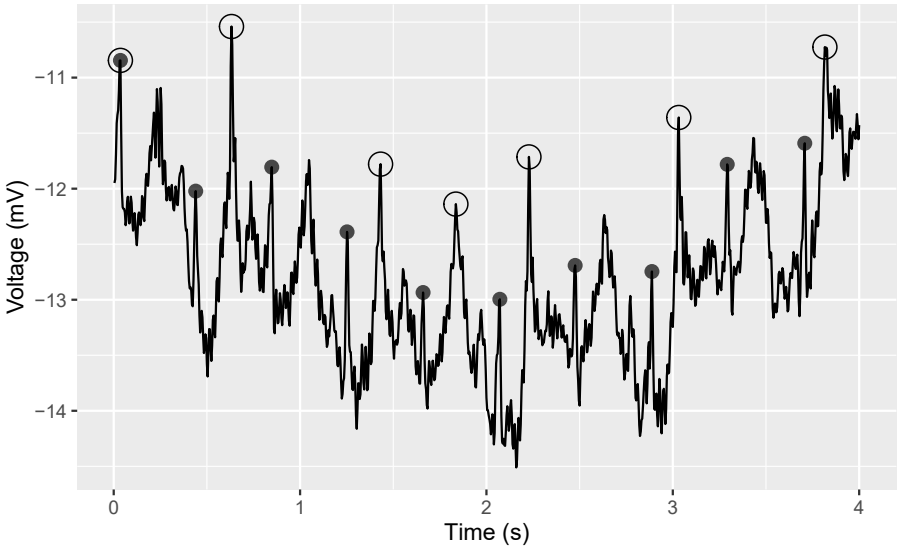

Figure 3.7: Failure of the LSTM based detector. Hollow circles indicate predicted R-peaks while grey dots mark the true R-peaks. ECG from Subject 7, jogging with loose cables setup (GUDB).

ferent degrees of Gaussian noise also emphasized the robustness of the LSTM based detector. Its performance in the lowest SNR ratios was substantially above the reference detectors.

There is only few records from the GUDB where LSTM based method showed some degree of failure. Without these, the performance of the LSTM based QRS detector would have been nearly flawless. Performance with the UCI data set was nearly flawless as LSTM based QRS detector showed only few detection errors. In all cases, failures are probably explainable by the noise types that were not used in the training, and thus, are unknown to the LSTM network.

Besides robustness to the noise, the obvious strong points of the LSTM based detector is its capability for sample precise detection. The predictions are usually at point with the ground truth R-peaks and there is no lag that is often present with reference detectors. From the reference detectors only the Engzee shows similar characteristics as its predictions match to the ground truth R-peaks. However, the overall performance of Engzee is not very good as it has often low recall. Performance of the LSTM based detector is also consistent as there are only small number

of records where distances from true positives to ground truth R-peaks vary notably. These factors make LSTM based detector a very good option for HRV analysis where precise identification of R-peak locations can be beneficial. This was validated with RMSSD evaluation, where LSTM based detector showed substantial failures only for few ECG records.

Error analysis showed that LSTM based detector fails on very challenging ECG signals. Failures occurred with electrode contact artifacts or with electrode motion artifacts. In former, the ECG contains sharp and intense baseline changes, while in the latter case ECG signal contains noise peaks that occur in the frequent manner. These challenging noise types were not used during the training phase, and therefore, LSTM model is unable to handle them. On the positive side, this gives straightforward direction for the model improvement. Using more noise sources during the training phase is most likely the easiest way to improve performance of the LSTM based detector.

From all of the detectors, Kalidas had best overall performance with the GUDB dataset. However, its performance was not nearly as good with the UCI dataset. One explanation could be the different nature of the data sets. For example, UCI dataset seems to have more high frequency noise (e.g. electrical interference, muscle artifacts) than GUDB dataset. Other possible explanation could be overfitting to GUDB dataset. Porr and Howell [49] made changes to the original algorithm of Kalidas to increase its performance:

> "In the original paper, a moving average was performed on the squared signal, however, it was found that using a bandpass filter instead significantly increased sensitivity and accuracy."

This kind of optimization could have caused overfitting against the GUDB dataset, which in turn can decrease the generalization performance.

Chapter 4

Conclusions

ECG is important biosignal that gives information from the function of the heart. It has a long history in the field of medicine where it is widely used by health professionals for diagnostic purposes. ECG has been utilized also by the regular people and athletes e.g. to track heart rate during the training. It has become more popular also with small wearable devices, like smart watches. For example, Apple Watch Series 5 and Withings Move ECG have the capability to record ECG.

Precise detection of QRS or R-peaks is requirement for accurate tracing of heart rate and HRV. Over the years, numerous methods have been proposed for the task. Most of the proposed methods are are traditional rule-based algorithms. They often work well when the ECG has good quality, but they can fail when the ECG is saturated with complex noise. ECG records can become easily contaminated with different noises. For example, electrodes can have bad contact to the skin, muscle artifacts can be induced during physical activity and electrical interference can occur in the vicinity of electrical devices. These kinds of noises can easily creep into the ECG e.g. during intense sport activity or long 24-h ambulatory ECG recordings.

Laitala et al. [2] proposed LSTM based detector for solution to noisy ECG signals. This thesis expanded the work of Laitala et al. [2] by giving more detailed explanation of the method and more comprehensive performance evaluation. The results are in the line with the previous work, the LSTM based detector showed the best overall

performance. This is not surprising, during the current decade the ANNs have shown their worth in many different fields, often resulting to better outcomes than can be achieved with traditional methods. However, this time results were not so one sided as in the evaluation of Laitala et al. [2] as one of the reference detectors (Kalidas) showed slightly better overall performance with the GUDB dataset. But when evaluated with the UCI dataset, all reference detectors showed much lower performances than the LSTM based detector.

Major strengths of the LSTM based detector are its capability to process very noisy ECG signals and sample precise detection of R-peaks. These are valuable features for any detector. No separate preprocessing steps are required from the user as majority of the work is done by the LSTM model. This makes method very straightforward and easy to use. Because of the sample precise R-peak detection, it is possible to calculate very accurate heart rate and HRV values. Although results of the LSTM based detector were very good, there is still some room for improvement. The most straightforward way to improve method would be to use more different noise sources (e.g. electrode motion and contact noise) during the training phase. Also the `remove_close` filtering function should be developed further so that it can work better in the case where heart rate varies a lot. This could be done by replacing the fixed threshold distance that is currently used with more advanced advanced adaptive threshold distance.

The major downside of the LSTM based detector is its high computational costs. The reference algorithms are very lightweight, and they can be run on small embedded devices. This is not true for LSTM based detector which requires more computational resources. However, today Internet of Things (IoT) has a big role and IoT based solutions are developing fast. IoT could offer a solution for computational requirements, processing could be performed in the Fog or Cloud layers where more computational capacity is available.

References

[1] A. D. Waller, "A demonstration on man of electromotive changes accompanying the heart's beat," *The Journal of physiology*, vol. 8, no. 5, p. 229, 1887.

[2] J. Laitala *et al.*, "Robust ECG R-peak detection using LSTM," in *Proceedings of the 35th annual ACM symposium on applied computing*, 2020, pp. 1104–1111.

[3] E. Hiltunen *et al.*, *Galenos: Johdanto lääketieteen opintoihin*. WSOYpro, 2010.

[4] J. S. Coviello, *ECG interpretation made incredibly easy!* Wolters Kluwer, 2016.

[5] L. S. L. MD, *Pathophysiology of Heart Disease: A Collaborative Project of Medical Students and Faculty*, Fifth, North American edition. Baltimore, MD: LWW, 2010.

[6] A. G. Webb, *Principles of biomedical instrumentation*. Cambridge University Press, 2018.

[7] G. B. Moody, W. E. Muldrow, and R. G. Mark, "The MIT-BIH noise stress test database," in *Computers in cardiology*, 1984, pp. 381–384.

[8] G. M. Friesen, T. C. Jannett, M. A. Jadallah, S. L. Yates, S. R. Quint, and H. T. Nagle, "A comparison of the noise sensitivity of nine QRS detection algorithms," *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 1, pp. 85–98, Jan. 1990.

[9] G. B. Moody, W. E. Muldrow, and R. G. Mark, "A noise stress test for arrhythmia detectors," *Computers in cardiology*, vol. 11, no. 3, pp. 381–384, 1984.

[10] B.-U. Köhler, C. Hennig, and R. Orglmeister, "The principles of software QRS detection," *IEEE engineering in medicine and biology magazine: the quarterly magazine of the Engineering in Medicine & Biology Society*, vol. 21, no. 1, pp. 42–57.

[11] J. Pan and W. J. Tompkins, "A Real-Time QRS Detection Algorithm," *IEEE Transactions on Biomedical Engineering*, vols. BME-32, no. 3, pp. 230–236, Mar. 1985.

[12] R. G. Lyons and D. L. Fugal, *The essential guide to digital signal processing*. Pearson Education, 2014.

[13] C. Li, C. Zheng, and C. Tai, "Detection of ECG characteristic points using wavelet transforms," *IEEE Transactions on Biomedical Engineering*, vol. 42, no. 1, pp. 21–28, Jan. 1995.

[14] S. Mallat and W. L. Hwang, "Singularity detection and processing with wavelets," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 617–643, Mar. 1992.

[15] V. Kalidas and L. Tamil, "Real-time QRS detector using Stationary Wavelet Transform for Automated ECG Analysis," in *2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE)*, 2017, pp. 457–461.

[16] M. Šarlija, F. Jurišić, and S. Popović, "A convolutional neural network based approach to QRS detection," in *Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis*, 2017, pp. 121–125.

[17] Y. Xiang, Z. Lin, and J. Meng, "Automatic QRS complex detection using two-level convolutional neural network," *BioMedical Engineering OnLine*, vol. 17, Jan. 2018.

[18] B. Yuen, X. Dong, and T. Lu, "Inter-Patient CNN-LSTM for QRS Complex Detection in Noisy ECG Signals," *IEEE Access*, vol. 7, pp. 169359–169370, 2019.

[19] J. D. Kelleher, *Deep learning*. MIT Press, 2019.

[20] S. Parsons and J. Huizinga, "Robust and fast heart rate variability analysis of long and noisy electrocardiograms using neural networks and images," *arXiv:1902.06151 [q-bio]*, Feb. 2019.

[21] S. S. Mehta and N. S. Lingayat, "Detection of QRS complexes in electrocardiogram using support vector machine," *Journal of Medical Engineering & Technology*,

vol. 32, no. 3, pp. 206–215.

[22] I. Saini, D. Singh, and A. Khosla, "QRS detection using K-Nearest Neighbor algorithm (KNN) and evaluation on standard ECG databases," *Journal of Advanced Research*, vol. 4, no. 4, pp. 331–344, Jul. 2013.

[23] G. E. Hinton and others, "Learning distributed representations of concepts," in *Proceedings of the eighth annual conference of the cognitive science society*, 1986, pp. 1–12.

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

[25] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, 2013, pp. 6645–6649.

[26] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[27] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[28] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[29] A. Géron, *Hands-on machine learning with scikit-learn, keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2019.

[30] M. L. Minsky and S. Papert, *Perceptrons: An introduction to computational geometry.* MIT Press, 1969.

[31] S. Raschka and V. Mirjalili, *Python machine learning: Machine learning and deep learning with python, scikit-learn, and TensorFlow 2, 3rd edition.* Packt Publishing, 2019.

[32] P. Wilmott, *Machine learning: An applied mathematics introduction*. Panda Ohana Publishing, 2019.

[33] G. Cybenko, "Approximations by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 183–192, 1989.

[34] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[36] A. Ng and K. Katanforoosh, "CS229 lecture notes deep learning," 2018.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[38] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, vol. 385. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[39] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *arXiv:1409.3215 [cs]*, Dec. 2014.

[40] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: A Neural Image Caption Generator," *arXiv:1411.4555 [cs]*, Apr. 2015.

[41] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[42] G. Chen, "A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation," *arXiv:1610.02583 [cs]*, Jan. 2018.

[43] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," *arXiv:1303.5778 [cs]*, Mar. 2013.

[44] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[46] F. Chollet, *Deep learning with python*, 1st ed. USA: Manning Publications Co., 2017.

[47] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH Arrhythmia Database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, May 2001.

[48] A. L. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220.

[49] B. Porr and L. Howell, "R-peak detector stress test with a new noisy ECG database reveals significant performance differences amongst popular detectors," *bioRxiv*, Aug. 2019.

[50] E. K. Naeini *et al.*, "Prospective study evaluating a pain assessment tool in a postoperative environment: Protocol for algorithm testing and enhancement," *JMIR Research Protocols*, vol. 9, no. 7, p. e17783, 2020.

[51] V. K. Sarker, M. Jiang, T. N. Gia, A. Anzanpour, A. M. Rahmani, and P. Liljeberg, "Portable multipurpose bio-signal acquisition and wireless streaming device for wearables," in *Proceedings of IEEE sensors applications symposium*, 2017.

[52] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[53] F. Chollet and others, "Keras," 2015.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[55] L. Howell and B. Porr, "Popular ECG R peak detectors written in python." Zenodo, Aug-2019.

[56] P. Hamilton, "Open source ECG analysis," in *Computers in Cardiology*, 2002, pp. 101–104.

[57] I. I. Christov, "Real time electrocardiogram QRS detection using combined adaptive threshold," *BioMedical Engineering OnLine*, vol. 3, no. 1, p. 28, Aug. 2004.

[58] M. Elgendi, M. Jonkman, and F. D. Boer, "Frequency bands effects on QRS detection," in *BIOSIGNALS 2010 - Proceedings of the 3rd International Conference on Bio-inpsired Systems and Signal Processing*, 2010, pp. 428–431.

[59] W. Engelse and C. Zeelenberg, "A single scan algorithm for qrs-detection and feature extraction," *Computers in cardiology*, vol. 6, no. 1979, pp. 37–42, 1979.

[60] A. Lourenço, H. Silva, P. Leite, R. Lourenço, and A. L. N. Fred, "Real Time Electrocardiogram Segmentation for Finger based ECG Biometrics," in *BIOSIGNALS*, 2012.

[61] U. Rajendra Acharya, K. Paul Joseph, N. Kannathal, C. M. Lim, and J. S. Suri, "Heart rate variability: A review," *Medical and Biological Engineering and Computing*, vol. 44, no. 12, pp. 1031–1051, Dec. 2006.

[62] F. Shaffer and J. P. Ginsberg, "An Overview of Heart Rate Variability Metrics and Norms," *Frontiers in Public Health*, vol. 5, Sep. 2017.

[63] C. Carreiras *et al.*, "BioSPPy: Biosignal processing in Python," 2015.

[64] Electrophysiology Task Force, "Heart Rate Variability," *Circulation*, vol. 93, no. 5, pp. 1043–1065, Mar. 1996.

Appendix F1 Scores

Table 4.1: F1 scores, jogging with chest strap setup

Subject	LSTM	Pan-Tompkins	Hamilton	Christov	Engzee	Kalidas	Elgendi
0	**1**	0.84	**1**	**1**	**1**	**1**	**1**
1	0.85	0.9	**0.99**	0.87	0.76	**0.99**	0.94
2	**1**	0.81	**1**	**1**	0.99	**1**	**1**
3	**1**	0.87	**1**	**1**	0.99	**1**	**1**
4	0.99	0.91	0.99	0.99	0.99	**1**	**1**
5	**1**	0.76	**1**	**1**	**1**	**1**	0.99
6	**1**	0.95	0.99	0.99	**1**	**1**	**1**
7	**0.91**	0.72	0.81	0.72	0.89	0.9	0.59
8	**1**	0.92	**1**	**1**	0.99	**1**	0.94
9	**1**	0.81	**1**	**1**	**1**	**1**	**1**
10	**0.95**	0.83	0.93	0.87	0.92	0.94	0.77
11	0.99	0.92	**1**	0.89	0.56	**1**	**1**
12	**1**	0.85	0.99	0.99	0.83	**1**	0.99
13	**1**	0.84	0.99	**1**	0.77	**1**	0.98
15	0.98	0.91	0.99	0.92	0.97	**1**	0.9
16	**1**	0.65	0.92	**1**	0.07	0.99	0.99
17	0.98	0.31	0.87	**1**	0.14	**1**	**1**
18	**1**	0.92	0.96	0.99	0.81	0.99	0.99
19	**1**	0.87	0.99	0.98	0.94	0.99	0.99
20	**1**	0.9	**1**	**1**	0.97	**1**	**1**
21	0.98	0.81	0.99	0.99	0.57	**1**	0.99
22	**1**	0.83	**1**	**1**	0.95	**1**	0.99
23	0.95	0.89	0.93	0.89	0.94	**0.99**	0.85
24	0.98	0.96	0.97	0.97	0.91	**1**	**1**

Table 4.2: F1 scores, hand bike with loose cables setup

Subject	LSTM	Pan-Tompkins	Hamilton	Christov	Engzee	Kalidas	Elgendi
0	**1**	0.88	0.8	0.98	0.81	0.88	0.94
1	**1**	0.64	0.99	0.99	0.99	0.99	0.99
2	**1**	0.74	0.56	0.99	0.93	0.99	0.94
3	**1**	0.86	**1**	**1**	**1**	**1**	0.99
5	**0.98**	0.86	0.89	0.93	0.82	0.92	0.94
6	**1**	0.88	0.87	0.69	0.83	0.98	0.97
7	**1**	0.87	**1**	**1**	**1**	**1**	**1**
8	**1**	0.72	0.99	**1**	0.94	**1**	**1**
9	**1**	0.89	**1**	0.99	0.99	0.99	**1**
10	**1**	0.94	0.99	0.99	0.99	0.99	0.98
11	**1**	0.96	0.99	0.99	**1**	**1**	0.99
12	0.78	0.87	**0.97**	0.89	0.7	0.96	0.96
13	**1**	0.89	0.99	0.99	**1**	**1**	**1**
14	**1**	0.73	**1**	**1**	**1**	**1**	0.99
15	**1**	0.89	0.98	**1**	0.99	0.99	**1**
16	**1**	0.4	0.98	0.99	**1**	**1**	0.99
17	**1**	0.6	0.99	0.99	0.99	**1**	**1**
18	**1**	0.8	0.99	**1**	**1**	**1**	0.98
19	**1**	0.9	0.13	**1**	0.99	0.99	**1**
20	**1**	0.98	0.97	**1**	0.99	**1**	**1**
21	**0.99**	0.85	0.89	0.64	0.98	**0.99**	0.97
22	**1**	0.99	**1**	0.98	0.95	**1**	0.99
23	**0.98**	0.93	0.96	**0.98**	0.73	0.97	**0.98**
24	**1**	0.82	0.99	0.99	**1**	**1**	**1**

Table 4.3: F1 scores, hand bike with chest strap setup

Subject	LSTM	Pan-Tompkins	Hamilton	Christov	Engzee	Kalidas	Elgendi
0	0.99	0.75	0.99	0.99	**1**	**1**	**1**
1	**0.99**	0.89	**0.99**	0.73	0.81	**0.99**	**0.99**
3	**1**	0.69	0.97	**1**	**1**	**1**	0.99
4	**1**	0.93	**1**	**1**	0.99	**1**	0.99
5	**1**	0.88	0.96	**1**	0.99	0.99	0.99
6	**1**	0.89	0.98	0.99	0.99	0.99	**1**
7	**1**	0.95	0.99	**1**	**1**	**1**	**1**
8	**1**	0.65	**1**	**1**	**1**	**1**	**1**
9	**1**	0.38	0.97	**1**	**1**	0.99	**1**
10	**1**	0.53	0.98	0.99	**1**	**1**	0.99
11	0.93	0.63	0.98	0.66	0.58	**0.99**	0.98
12	**1**	0.87	**1**	**1**	0.98	**1**	**1**
13	**1**	0.84	0.96	0.99	0.92	0.99	0.99
14	0.95	0.79	0.89	0.63	0.64	**0.97**	0.9
15	**1**	0.67	0.99	**1**	**1**	0.99	**1**
16	**1**	0.45	0.98	**1**	0.04	**1**	0.99
17	**1**	0.09	0.98	**1**	0.66	**1**	**1**
18	0.98	0.68	**1**	0.4	0.66	0.98	0.95
19	0.98	0.81	0.77	0.86	0.8	**1**	0.92
20	**1**	0.61	0.99	**1**	0.96	**1**	**1**
21	0.9	0.55	**1**	0.99	0.09	**1**	0.99
22	**1**	0.5	0.99	**1**	0.99	**1**	**1**
23	0.99	0.95	0.99	0.99	0.93	**1**	0.92
24	**1**	0.96	0.99	0.99	0.67	**1**	0.97

Table 4.4: F1 scores, maths with loose cables setup

Subject	LSTM	Pan-Tompkins	Hamilton	Christov	Engzee	Kalidas	Elgendi
0	**1**	0.89	0.93	0.99	0.99	0.99	0.95
1	**1**	0.63	**1**	**1**	0.99	**1**	0.99
2	**1**	0.61	0.97	**1**	0.99	0.99	**1**
3	**1**	0.72	0.99	**1**	0.99	**1**	**1**
5	**1**	0.82	0.96	0.99	**1**	**1**	0.99
6	**1**	0.8	0.97	0.99	**1**	0.98	0.96
7	**1**	0.97	0.99	**1**	0.99	**1**	**1**
8	**1**	0.66	**1**	**1**	0.99	**1**	**1**
9	**1**	0.79	**1**	**1**	**1**	**1**	**1**
10	**1**	0.76	**1**	0.99	0.99	**1**	**1**
11	**1**	0.29	0.99	**1**	0.99	**1**	**1**
12	0.92	0.98	**1**	**1**	0.84	0.99	0.99
13	**1**	0.98	0.99	**1**	**1**	**1**	0.99
14	**1**	0.36	0.99	**1**	0.99	**1**	**1**
15	**1**	0.71	**1**	0.99	0.99	0.99	**1**
16	**1**	0.48	0.98	**1**	**1**	**1**	0.98
17	**1**	0.46	0.76	0.99	**1**	0.99	**1**
18	**1**	0.64	**1**	0.99	**1**	**1**	**1**
19	0.99	0.89	0.23	0.99	**1**	**1**	**1**
20	**1**	0.78	0.99	**1**	**1**	**1**	**1**
21	**1**	0.66	**1**	**1**	0.99	**1**	0.99
22	**1**	0.8	0.99	**1**	0.99	**1**	**1**
23	**0.99**	0.98	**0.99**	**0.99**	0.65	**0.99**	**0.99**
24	**1**	0.87	0.12	**1**	**1**	**1**	0.99

Table 4.5: F1 scores, maths with chest strap setup

Subject	LSTM	Pan-Tompkins	Hamilton	Christov	Engzee	Kalidas	Elgendi
0	**1**	0.51	0.99	**1**	0.99	**1**	**1**
1	**1**	0.76	0.99	0.67	**1**	**1**	**1**
2	**1**	0.45	**1**	**1**	**1**	**1**	**1**
3	**1**	0.54	**1**	**1**	**1**	**1**	**1**
4	**1**	0.99	0.98	**1**	**1**	0.99	**1**
5	**1**	**1**	**1**	**1**	0.99	**1**	**1**
6	**1**	0.9	0.99	0.97	0.99	**1**	**1**
7	**1**	0.98	**1**	**1**	**1**	**1**	**1**
8	**1**	0.54	**1**	**1**	**1**	**1**	**1**
9	**1**	0.37	**1**	**1**	**1**	**1**	**1**
10	**1**	0.18	**1**	**1**	**1**	0.99	**1**
11	0.97	0.66	0.99	0.66	0.14	0.99	**1**
12	**1**	0.86	0.99	**1**	0.87	**1**	0.99
13	**1**	0.85	0.98	**1**	0.98	**1**	**1**
14	**1**	0.95	0.99	**1**	**1**	0.99	**1**
15	**1**	0.44	**1**	**1**	**1**	0.99	**1**
16	**1**	0.37	0.99	**1**	0.62	**1**	**1**
17	**1**	0.16	**1**	**1**	0.97	0.99	**1**
18	**1**	0.94	0.96	**1**	**1**	0.99	**1**
19	**1**	0.57	0.98	**1**	**1**	**1**	**1**
20	**1**	0.3	**1**	0.98	0.99	**1**	0.99
21	0.93	0.4	0.98	**1**	0.05	**1**	0.99
22	**1**	0.25	0.99	**1**	**1**	0.99	**1**
23	**1**	0.99	**1**	0.99	**1**	**1**	**1**
24	**1**	0.9	0.55	0.96	0.99	**1**	0.99

Table 4.6: F1 scores, sitting with loose cables setup

Subject	LSTM	Pan-Tompkins	Hamilton	Christov	Engzee	Kalidas	Elgendi
0	**1**	0.96	0.98	0.99	**1**	0.99	**1**
1	**1**	0.48	**1**	**1**	**1**	**1**	0.99
2	**1**	0.52	0.98	**1**	**1**	**1**	0.99
3	**1**	0.53	0.98	**1**	**1**	**1**	**1**
4	0.99	**1**	0.25	0.99	0.93	0.99	0.99
5	**1**	0.66	0.87	**1**	**1**	**1**	**1**
6	**1**	0.83	**1**	**1**	**1**	**1**	**1**
7	**1**	0.97	**1**	0.99	**1**	**1**	**1**
8	**1**	0.54	0.98	**1**	0.99	**1**	0.99
9	**1**	0.78	**1**	0.99	**1**	**1**	**1**
10	**1**	0.78	0.99	**1**	0.99	**1**	**1**
11	**1**	0.28	0.98	**1**	**1**	**1**	**1**
12	0.99	**1**	**1**	**1**	0.96	0.99	**1**
13	**1**	0.85	**1**	**1**	**1**	**1**	**1**
14	**1**	0.28	**1**	0.99	0.99	0.99	**1**
15	**1**	0.8	0.99	**1**	**1**	**1**	0.99
16	**1**	0.55	0.99	**1**	**1**	**1**	**1**
17	**1**	0.71	**1**	**1**	**1**	**1**	**1**
18	**1**	0.6	**1**	**1**	**1**	**1**	**1**
19	**1**	0.85	**1**	0.99	0.99	0.99	**1**
20	**1**	0.85	0.07	**1**	**1**	0.99	**1**
21	**1**	0.53	0.88	**1**	0.99	**1**	**1**
22	**1**	0.51	**1**	**1**	0.99	**1**	**1**
23	**1**	0.95	0.94	**1**	0.97	**1**	**1**
24	**1**	0.73	0.04	0.56	0.99	**1**	**1**

Table 4.7: F1 scores, sitting with chest strap setup

Subject	LSTM	Pan-Tompkins	Hamilton	Christov	Engzee	Kalidas	Elgendi
0	**1**	0.44	0.98	**1**	**1**	**1**	**1**
1	**1**	0.93	**1**	0.75	0.69	**1**	0.99
2	**1**	0.61	0.99	0.99	**1**	0.99	**1**
3	**1**	0.54	0.99	**1**	**1**	0.99	**1**
4	**1**	0.99	**1**	**1**	0.99	0.99	**1**
5	**1**	**1**	0.97	0.99	0.99	**1**	**1**
6	**1**	0.95	0.99	**1**	**1**	0.99	**1**
7	**1**	0.93	**1**	**1**	**1**	**1**	0.99
8	**1**	0.43	**1**	**1**	**1**	**1**	**1**
9	**1**	0.35	**1**	**1**	0.99	**1**	**1**
10	**1**	0.2	0.99	**1**	0.99	**1**	**1**
11	**1**	0.59	0.95	0.66	0.21	0.99	**1**
12	**1**	0.83	0.99	**1**	0.59	**1**	**1**
13	**1**	0.88	0.95	**1**	0.96	**1**	**1**
14	**1**	0.85	0.99	0.67	**1**	**1**	**1**
15	**1**	0.25	0.99	**1**	**1**	**1**	**1**
16	**1**	0.18	0.98	**1**	0.03	0.99	**1**
17	**1**	0.08	0.91	**1**	0.95	0.99	**1**
18	**1**	0.98	0.98	0.99	**1**	**1**	**1**
19	**1**	0.58	0.99	**1**	**1**	0.99	**1**
20	**1**	0.28	0.98	**1**	0.99	**1**	**1**
21	0.91	0.42	0.99	**1**	0.11	**1**	**1**
22	**1**	0.19	**1**	**1**	0.99	**1**	**1**
23	**1**	**1**	**1**	0.99	**1**	**1**	0.99
24	**1**	0.81	0.67	0.99	0.99	**1**	**1**

Table 4.8: F1 scores, walking with loose cables setup

Subject	LSTM	Pan-Tompkins	Hamilton	Christov	Engzee	Kalidas	Elgendi
0	**1**	0.92	0.82	**1**	0.99	0.97	0.98
1	**1**	0.97	0.99	0.99	**1**	**1**	**1**
3	**1**	0.96	0.99	0.99	**1**	**1**	**1**
4	0.92	0.77	**0.99**	0.9	0.7	**0.99**	**0.99**
5	0.99	0.96	0.97	0.99	**1**	**1**	0.99
6	**1**	0.79	0.97	**1**	**1**	**1**	**1**
7	**1**	0.98	0.8	0.99	**1**	**1**	**1**
8	**1**	**1**	**1**	**1**	0.94	**1**	0.99
9	**1**	0.67	0.67	0.99	**1**	**1**	0.99
10	0.99	0.98	0.97	0.96	**1**	0.99	0.97
11	0.99	0.69	0.58	0.82	0.77	**1**	0.95
12	0.88	0.99	**1**	**1**	0.83	**1**	**1**
13	**1**	0.96	0.99	**1**	0.99	**1**	**1**
14	**1**	0.98	0.98	**1**	**1**	**1**	**1**
15	0.99	0.99	0.98	**1**	**1**	0.99	**1**
16	**1**	0.98	0.9	0.99	**1**	0.99	0.98
17	**1**	0.97	0.96	**1**	**1**	**1**	**1**
18	**1**	0.92	0.91	0.99	**1**	0.99	**1**
19	**1**	0.56	0.41	0.97	0.99	**1**	**1**
20	**0.98**	0.92	0.96	0.96	0.96	**0.98**	0.96
21	**1**	0.99	0.99	0.98	0.97	0.99	0.99
22	**1**	0.97	0.99	0.99	0.95	**1**	**1**
23	0.79	0.88	0.87	0.68	0.44	**0.98**	0.96
24	**1**	0.97	0.99	0.99	0.99	**1**	**1**

Table 4.9: F1 scores, walking with chest strap setup

Subject	LSTM	Pan-Tompkins	Hamilton	Christov	Engzee	Kalidas	Elgendi
0	**1**	0.62	0.99	**1**	**1**	**1**	**1**
1	**1**	0.85	0.99	0.67	**1**	**1**	**1**
2	**1**	0.84	**1**	**1**	**1**	**1**	**1**
3	**1**	0.57	**1**	**1**	**1**	**1**	**1**
4	**1**	**1**	0.99	0.99	**1**	0.99	**1**
5	**1**	0.98	0.99	0.99	**1**	0.99	0.99
6	**1**	0.89	0.99	0.99	0.99	**1**	**1**
7	**1**	**1**	**1**	**1**	**1**	**1**	**1**
8	**1**	0.62	**1**	**1**	0.99	**1**	**1**
9	**1**	0.33	**1**	**1**	**1**	**1**	**1**
10	**1**	0.31	0.99	**1**	**1**	0.99	**1**
11	0.95	0.67	0.99	0.66	0.04	**1**	**1**
12	**1**	0.87	**1**	**1**	0.68	**1**	**1**
13	**1**	0.76	**1**	**1**	0.94	**1**	**1**
14	0.98	0.65	0.48	0.72	0.97	**0.99**	0.97
15	**1**	0.43	**1**	**1**	**1**	0.99	0.99
16	**1**	0.29	0.99	**1**	0.17	0.99	0.99
17	**1**	0.11	0.96	**1**	0.68	0.99	**1**
18	**1**	0.96	0.99	0.46	0.98	**1**	**1**
19	**1**	0.41	0.98	**1**	**1**	**1**	**1**
20	**1**	0.52	0.97	0.99	0.98	**1**	**1**
21	0.91	0.48	**1**	**1**	0.21	**1**	0.99
22	**1**	0.15	0.99	**1**	**1**	**1**	**1**
23	**1**	0.99	**1**	**1**	**1**	**1**	**1**
24	**1**	0.97	0.81	0.99	**1**	**1**	**1**