Can we Teach a Machine to be a Cardiologist?

by

Sonia L. Kleinig

Thesis submitted for the degree of

Bachelor of Engineering (Honours)

in

Electrical and Electronic Engineering

University of Adelaide

2021

This page is intentionally blank.

Statement of Originality

I declare that all material in this thesis is my own work, except where there is clear acknowledgement and reference to the work of other. I have read the University Policy Statement on Academic Honesty and Assessment Obligations.

This work contains no material that has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published written by another person, except where due reference has been made in the text.

I give permission for this work to be reproduced and submitted to other academic staff for the purposes of assessment and to be copied, submitted and retained by the University's plagiarism detection software provider for the purposes of electronic checking of plagiarism.

Signed

4 June 2021

Date

Acknowledgements

I would like to acknowledge the effort my supervisors, Professor Derek Abbott and Mr Mohsen Dorraki, for their constant suggestions and feedback throughout the semester. I would also like to acknowledge my project partner, Hien Long Nguyen, for his efforts and assistance as we cooperated on many aspects of this project.

I would also like to thank all my lecturers, tutors and teachers, both past and present, for helping me get to where I am today.

Finally, on a more personal note, I would like to thank my friends and family for the constant support and encouragement.

Abstract

Electrocardiograms (ECGs) are recordings of the electrical activity of the heart, and play an important role in the diagnosis of many cardiac abnormalities. Recently, there has been an interest in finding methods of classifying ECGs using machine learning (ML) techniques. Two major steps are involved with this: pre-processing and classification. Pre-processing techniques, including bandpass filtering and wavelet transforms, are used to reduce noise and extract relevant features from the signal. Then classification is able to be done using a range of techniques including SVM and CNN. First, a set of signals with known classification are used to train the ML, and then another set of test signals are used to examine the accuracy of the classifier. This thesis begins with a literature review of previous works in this and related areas. Then, these techniques are examined by making modifications to an existing classifier. So far, a classifier using wavelet de-noising and an SVM has been modified to fit with data collected from the PhysioNet Database [2]. Although this classifier is able to achieve an accuracy of almost 98% with the example data, only accuracies of 70% or less have been achieved with the collected data. Further modification and optimisation of parameters over the remainder of the project should see improvements in this area. As well as viewing ECGs in the time domain, time-frequency plots known as scalograms have been created. Another classifier which may be able to classify the ECGs from their scalograms by utilising a CNN has been identified, although analysis of this, and other classifiers, are yet to be completed.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation and Significance

Biomedical signals, such as electrocardiograms (ECGs), can reveal unseen details about a patient's health. Hence, it is critical to use these signal to quickly and accurately detect abnormalities to facilitate timely treatment.

According to the World Health Organisation (WHO), cardiovascular disease (CVD) continues to be the leading cause of death globally [1]. CVDs can take many forms, and can exist for a number of reasons including behavioural factors, and underlying health conditions [1]. Since most CVDs can be prevented by managing behavioural risk factors, including smoking and unhealthy diet [1], identifying and addressing them is important.

CVDs generally have some impact on the rhythm of the heart which can be identified from an ECG recording, making them a useful tool in identifying heart diseases. Hence, analysing these quickly and correctly is important, and has led to an interest in using machine learning (ML) techniques to identify heart abnormalities, and even classify the type of abnormality.

This project examines the possibility of using standard ML techniques, programmed in either MATLAB or Python, to identify various CVDs. Although this project has no official sponsor, this and similar work could prove invaluable in the biomedical field.

1.2. Project Aims

The aim of this project, *Can we Teach a Machine to be a Cardiologist?*, is to explore various machine learning techniques to determine whether they can be used to teach a machine to correctly classify CVDs. The project will involve developing a ML algorithm which extracts the relevant features of an ECG, learns which features correspond to which condition, and then accurately classifies other ECG signals according to these feaures.

An exploration of different techniques will reveal which processes are most able to be used to achieve this goal. The effectiveness of each will be compared, and these results will be compared to the results recorded in the literature, with the aim of achieving an outcome which is comparable to those.

1.3. Project Scope

The project will identify a number of ML techniques which can be used to classify signals and examine as many as possible in the given timeframe. These techniques will be chosen based on their effectiveness, how easily they can be understood, and whether existing coded examples can be accessed and modified.

This project requires the analysis of ECG recordings. Thankfully, numerous databases with an extensive collection of ECG recordings are available online, so no experiments will need to be conducted to collect data. The PhysioNet database [2] has been used for all work so far.

1.4. Budget

The budget allocated for this project is equal to $250 per student, meaning the total budget is $500 for this project. Data, such as ECG recordings, are available for free online, and all required software and journal articles are available through the University already. So, none of these items are required expenses.

As the MATLAB computations can be quite intensive, the possibility of purchasing more RAM for a PC is being investigated. If this is approved, the cost will be approximately $200, leaving $300 unspent. Table 1 summarises the current financial situation of the project.

Table 1: Summary of Budget

Expense	Amount ($)	Total ($)
Project Budget	+500	500
Total Income		500
Data (ECG Recordings)	0	0
Software (MATLAB/ Python)	0	0
Journal Articles	0	0
RAM	-200 (approx.)	200
Total Expenses		200
Budget Remaining		300

At this stage, it is worth ensuring excess budget in case expenses are revealed at a later point.

1.5. Background Information

This thesis will discuss a number of technical topics. These come under three broad subjects: ECG analysis, pre-processing methods, and machine learning. These topics will be briefly explained here, and further information can be found in the sources referenced.

1.5.1. Electrocardiogram Analysis

In the human body, the contraction of muscles is associated with changes in the membrane potential of cells, i.e. depolarisation [3]. ECGs measure this electrical activity in relation to the heart. These measurements are obtained by placing electrodes on the patient's torso and measuring the electrical activity produced.

Any irregularity in the ECG waveform could be indicative of a CVD or other abnormality. The challenge lies in identifying these abnormalities, particularly since ECG recordings naturally vary person to person [4], and the abnormalities can be very subtle. Hence, it is important to find a way to accurately identify irregularities and classify the signal accordingly.

Figure 1 shows an idealised ECG signal. A number of points on the ECG are of particular importance. Namely, the P-wave, the QRS complex, the T-wave and the interval between subsequent R-peaks (RR interval).

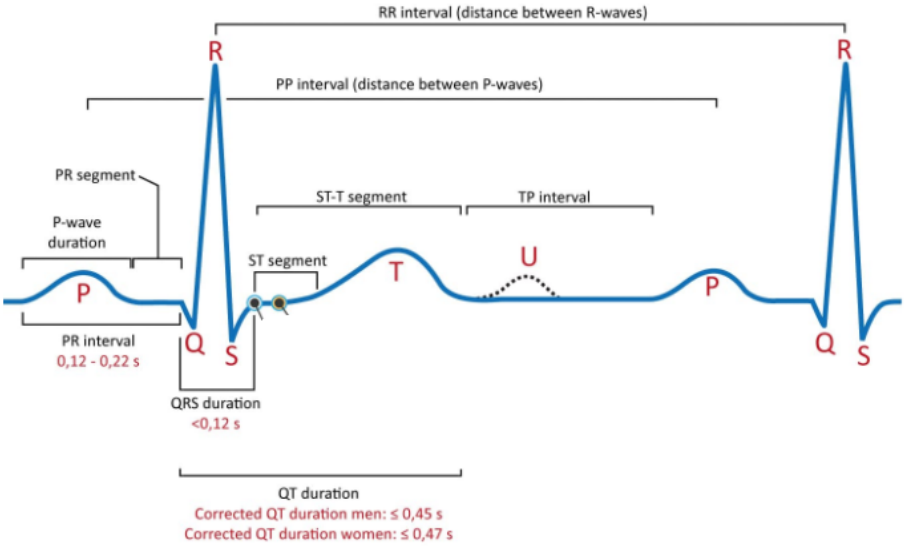


Figure 1: Simplified ECG signal [5]

The P-wave corresponds to the contraction of the two smaller chambers of the heart, the atria. The QRS complex following represents the contraction of the two larger chambers of the heart, the ventricles. This is the contraction that pushes the blood out

of the heart and around the body. The T-wave then represents the repolarisation (return to resting state) of the ventricles (note the repolarisation of the atria is hidden in the QRS complex) [3]. Finally, the RR interval represents the length of time between subsequent heart beats.

Quick analysis of the RR interval can reveal whether or not a patient's heart is beating in a regular rhythm and may point out an arrhythmia if not. Conversely, it is more difficult to determine whether the pattern of P-waves, QRS complex and T-waves are abnormal. This is made difficult by normal variability of ECG features, both within and between patients [4]. Furthermore, electrical activity of other muscles must be taken into consideration when analysing an ECG recording.

The databased used here (the PhysioNet 2017 Computing in Cardiology Database [2]) contained four types of signal: normal, atrial fibrillation (AF), other arrhythmia, and noisy. Hence, it was important to gain a quick understanding of each.

Normal signals have the characteristic waveform as in Figure 1, although this does have variations from patient to patient. Each feature should have a duration within a specified range, and the RR interval should be fairly constant.

AF is an abnormal condition in which the regular atrial activity has been replaced with fast and disorderly tremor waves [6]. The normal P-waves often disappear, and the distance between R-peaks varies. The incidence of AF increases with age, and can be characterised by palpitations, shortness of breath and chest pain.

Figure 2 compares an ECG with AF with a normal ECG rhythm.

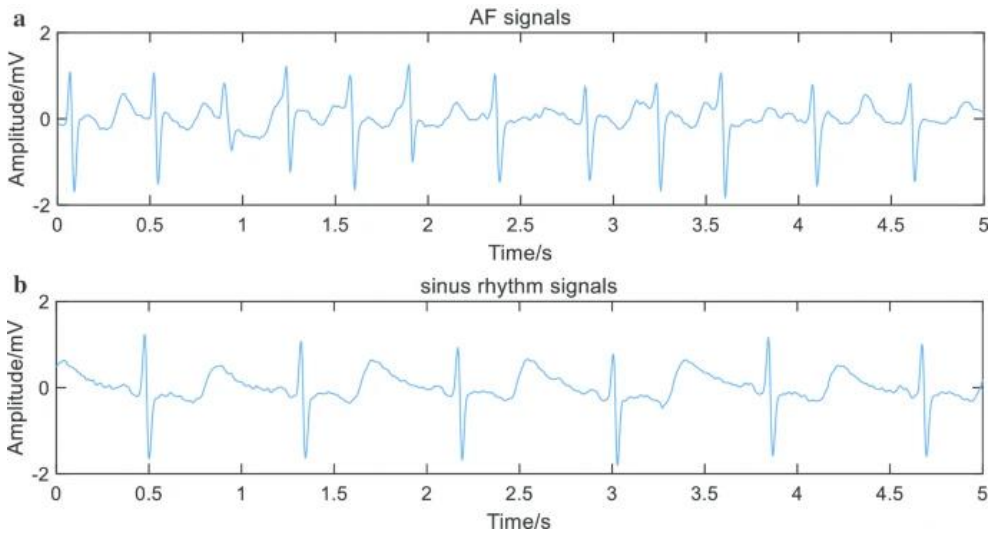


Figure 2: (a) Atrial fibrillation ECG waveform in comparison to (b) normal ECG [6]

Other arrhythmia describes any other abnormal condition. This can include congestive heart failure, a range of blockages, or any other arrhythmia. Noisy signals are classified as signals which contained too much noise to be accurately classified into another class.

1.5.2. Pre-Processing Techniques

Pre-processing of a signal is required prior to analysis to remove artifacts and noise which may impede the classifier. The important features of an ECG are relatively low-frequency (0.5-30 Hz) [6], so much of the high frequency content of the signal is noise which can be removed with filtering [7].

The ECG is unable to distinguish heart activity from other electrical activity in the chest, so other muscle contraction may also be recorded. These are known as artifacts, and may include any muscle movement or slow oscillations from breathing, for example, at the time of ECG recording.

A number of pre-processing techniques exist, including bandpass filtering and wavelet denoising. A brief introduction to wavelets is included here.

Wavelets form an orthonormal basis. This means they can be used to apply a wavelet transform to a time-domain signal to transform it into the wavelet domain in much the same way a set of sinusoids can be used to transform a time-domain signal into the frequency domain, as in the Fourier transform (FT). However, the FT provides only globally averaged information, meaning transient and location-specific features are often lost [3]. Wavelet transforms, on the other hand, allow for time and frequency analysis of a signal simultaneously, which allows transient and intermittent components to be localised.

It is possible to obtain a similar result by utilising a short time Fourier transform (STFT) [8]. This computes the FT of a signal in smaller time windows and can be used to plot an image known as a spectrogram (see Figure 3). However, wavelet transforms do a better job of this since they apply a window of varying length to the signal [3] [8]. This allows the transform to adapt based on the frequency components of the signal, which is much more difficult to do with FTs. The time-frequency plot produced by the wavelet transform is called a scalogram, and an example is shown in

Figure 4. Notice that it is similar to the STFT spectrogram, but provides more information, particularly at higher frequencies.

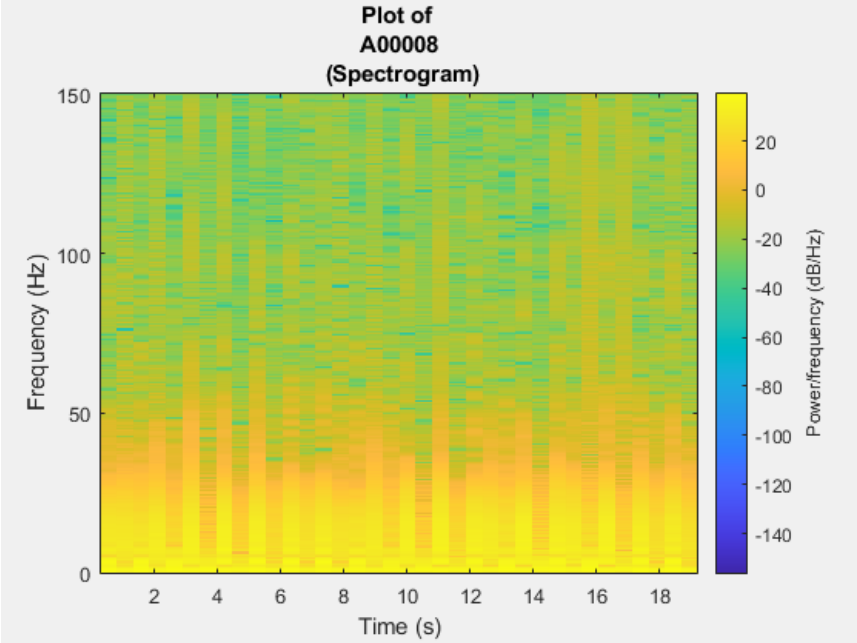


Figure 3: Example spectrogram

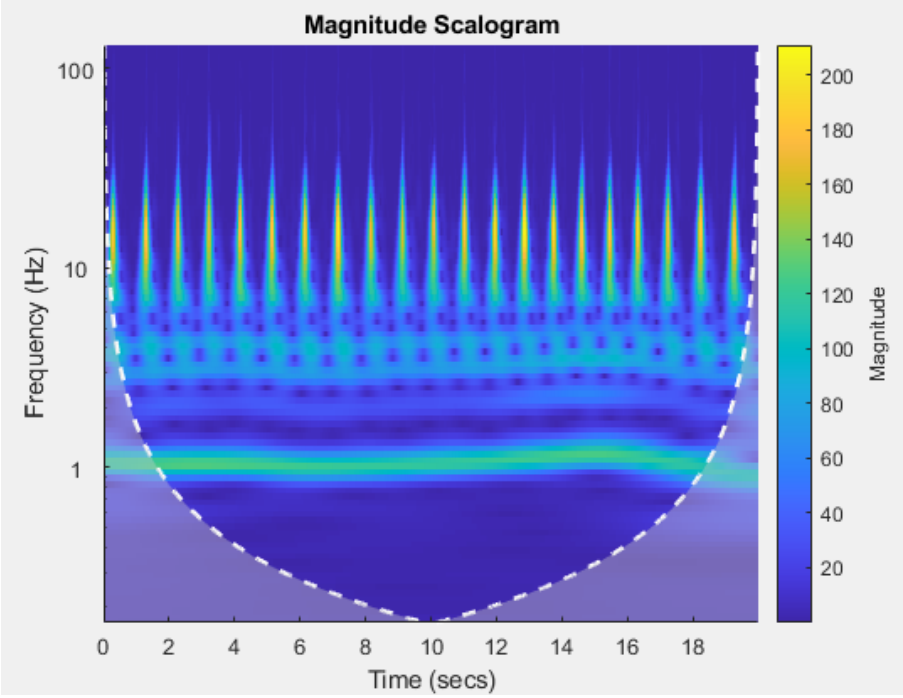


Figure 4: Example scalogram

Furthermore, a number of different wavelets may be used. The most popular of these are shown in Figure 5. The type of wavelet used for a given application can be chosen to best match the signal being analysed. Notice that some of these wavelets have a similar shape to the ECG waveform, and prove an invaluable tool in this case.

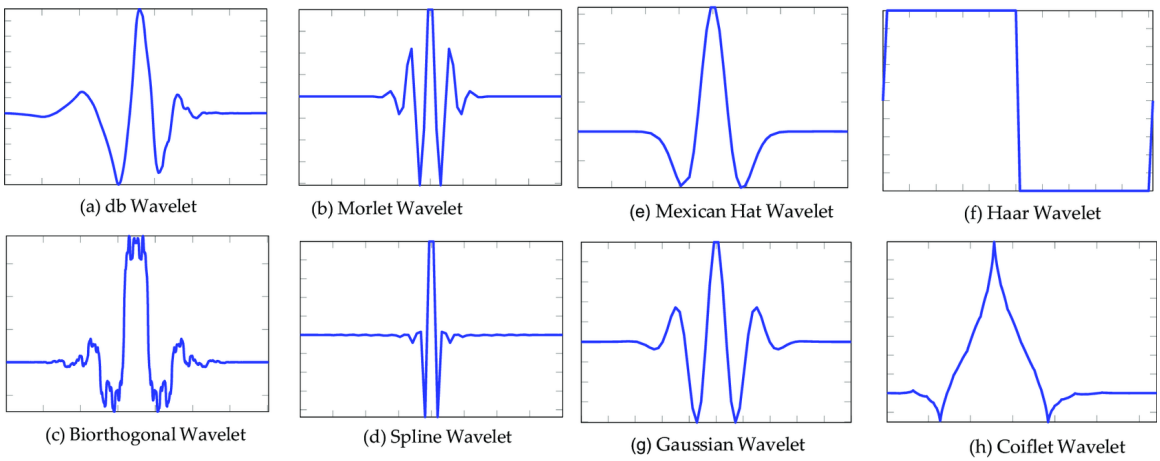


Figure 5: Common wavelets [9]

1.5.3. Machine Learning Techniques

ML is an application of artificial intelligence in which algorithms parse data, learn from it, and apply what they've learned to make an informed decision [10].

First, a few definitions are important to understand. Before it can be used as a classifier, ML algorithms must be trained in how they should classify that data. As such, a set of data must be divided into a 'training set' and a 'test set'. The training set and the correct labels for each piece of data is given to the machine to teach it what each classification means. Then the test set is provided to verify how well the machine has learnt these classifications. The machine's classifications can be compared to the actual label for each data to calculate the precision, recall and F1-score for the algorithm.

The following ML techniques have been identified:

- Support vector machine (SVM);
- Artificial neural network (ANN); and
- Convolutional neural network (CNN).

A basic description of each of these techniques is included here for readers who have not encountered these terms before.

An SVM is a supervised machine learning algorithm which can be used to assign labels to data, based on the value of a number of features it possesses. Each data item is plotted in n-dimensional space, where 'n' is the number of features under consideration [11]. Then, the SVM draws a line, or in higher-order space a

hyperplane, which best separates the data in the training set into its known categories. The test set is then plotted in this n-dimensional space, and classified according to which side of the hyperplane it falls on. Figure 6 illustrates a simple 2D example of this concept, in which the solid line shows the plane between classes (red and blue in this simple case).

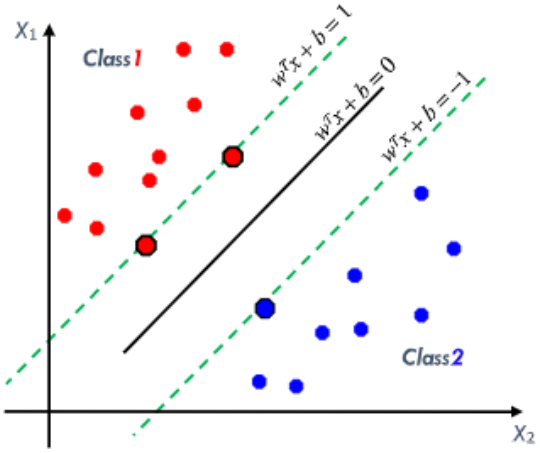


Figure 6: 2-dimensional example SVM [12]

An ANN is capable of extracting complex and non-linear relationships between features of a set of data [12]. They are constructed to simulate neurons in a biological nervous system, as depicted in Figure 7. It's comprised of many interconnected units, whereby the network function is largely determined by the connections, and each connection is a certain nonlinear function. The weight of each connection determines its contribution, and these weights can be adjusted through training, either from outside information or in response to the inputs [13]. The network is built directly form experimental data and the ANN's self-organising capabilities, and does not require prior assumptions [13].

Building on from ANNs, CNNs add some processing stages to the input of the neural network. They are especially helpful for classifying images, such as handwritten symbols as shown in Figure 8. The convolution layer extracts features, and the pooling layer reduces the size of these convolved features to decrease computational power [14]. These two layers enable the model to understand the features. Multiple convolution and pooling layers can be used to extract higher-level features than possible with a single layer. Finally, a fully-connected layer is used to classify the images, and this is generally a regular ANN [14].

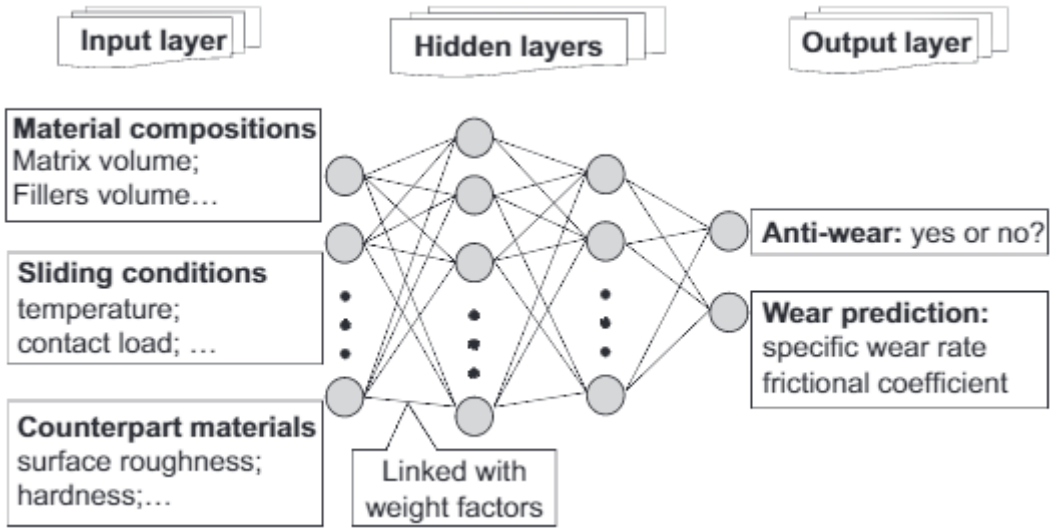


Figure 7: Example of an ANN for correlating certain properties with various parameters [13]

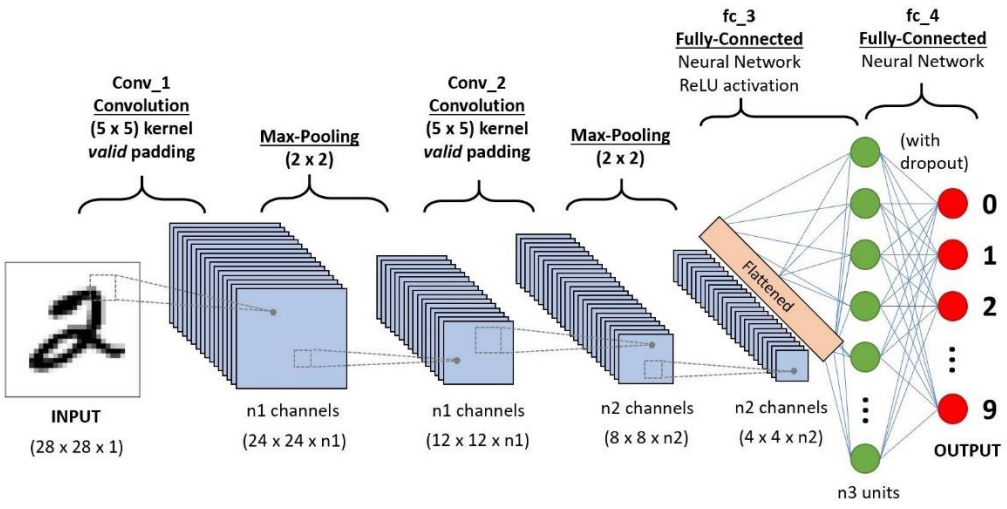


Figure 8: CNN example [14]

ANNs and CNNs both come under the category of deep learning. This is a subfield of machine learning in which algorithms are structured in layers to create the neural network. This makes it possible for the machine to learn through its own method of computing [14]. Conversely, basic ML requires human guidance to improve the process of its classifications, such as with the SVM.

1.6. Technical Challenges

A number of technical challenges were identified. Perhaps the biggest, is the use of programming software. Both members have a basic understanding of MATLAB from previous years, but will need to continue to develop this knowledge in order to be able to complete the project. An alternative which may be more effective at processing data is Python. Neither member has used Python previously, however it was decided to attempt to learn this language and use MATLAB as a backup if this proves unsuccessful. Hence, although it is preferable to process large amounts of data using Python, completing all programming in MATLAB is acceptable.

The project involves finding ML algorithms which can successfully classify ECG recordings, although neither member has previously worked in ML. This challenge has been mitigated by beginning research into ML concepts early, and by following coded MATLAB examples to get a deeper understanding of how these techniques work.

Finally, ML algorithms can become quite complex to run, and as such the computational power and time available may present a challenge to this project. For example, a reasonably powerful computer (generally more than a laptop) is required to have enough processing power and RAM to run more complex algorithms. As such, alternatives such as using Adapt through the University and purchasing more RAM have been considered.

2. Literature Review

2.1. Introduction

From preliminary research, it was decided that the project would be split into two main objectives, namely pre-processing and classification. Hence, the literature review has a similar structure. The focus was directed at papers which applied these processes to ECG analysis, however papers which used these techniques for a different purpose were considered in some cases, especially where that application was also in the biomedical field.

It was suggested that wavelet denoising be used in the pre-processing stage, and that a support vector machine (SVM) be used as a classifier. Hence the literature review began with a focus on methods which use these techniques, or which compare these techniques to others. Other approaches have also been compared.

The literature review will cover the following topics:

a. Available pre-processing techniques;

b. ECG features and feature extraction techniques;

c. The effectiveness of SVM classifiers;

d. The effectiveness of CNNs as classifiers; and,

e. Other classification methods and a comparison of their effectiveness.

The findings for each of these points are discussed in the subsections of Section 2.2, and conclusions in relation to this project are drawn in Section 2.3.

2.2. Findings

The findings of the literature review can be divided into the three stages of pre-processing, feature extraction, and classification. Although each stage will be discussed individually, it should be noted that most papers use one or more techniques in each stage. The effectiveness of each classification method is summarised in Table 2.

2.2.1. Pre-Processing Methods

Pre-processing is an important first step in classifying the ECG signals [7,15,16]. It removes undesirable features including noise, baseline wander, motion artifacts and other interruptions [15].

2.2.1.1. Noise Removal

Noise removal can be done in a number of different ways. It can be as simple as using a bandpass filter, as in [6,17,18]. Wang et al. [18] used a number of different filters to pre-process the ECG recordings. These were a 50 Hz notch filter to remove the mains power hum, a 30Hz low-pass filter to remove high-frequency noise, and a 0.1 Hz Chebyshev high-pass filter to remove low-frequency noise and artifacts.

Hu et al. [6] used a similar filtering process, with a digital FIR bandpass filter which had cut-off frequencies at 0.5 Hz and 30 Hz. The 30 Hz cut-off frequency could eliminate some electrical activity from the muscles, as well as the powerline interference. The 0.5 Hz cut-off was chosen to remove low-frequency artefacts due to respiration, electrode movement, and other low-frequency noise [6].

Alternatively, wavelet denoising can be used. The wavelet decomposition of a noisy signal concentrates the signal information across a few wavelet coefficients, without modifying the random distribution of the noise [9]. This meant denoising could be achieved by thresholding the wavelet coefficients. Another advantage of the WT is that it gave a time-variant decomposition, making it possible to choose different filtering settings for different time windows [9].

Adaptive filtering methods can be developed from Least Mean Square (LMS) and modified LMS-type algorithms [19]. The concept of these algorithms is to estimate signals which are corrupted with additive noise by minimising the mean squared error between the noisy ECG signal and a reference signal containing noise correlated to that in the ECG [19]. Venkatesan et al. [15] proposed a delayed error normalised LMS, which they find to be superior to both normalised LMS and delayed normalised LMS. Similarly, Khiter et al. [19] proposed a self-correcting leaky normalised LMS (SC-LNLMS), which applied multiple iterations of an adaptive noise canceller to further reduce signal noise.

2.2.1.2. Other Pre-Processing Steps

Pre-processing can also include other steps, such as segmenting ECG signals [6,15,18,20,21,22,23], and heartbeat normalisation [22]. The length of signal segment can vary, with some methods using long signals (i.e. >30 seconds) [15,23], and some used 10 second intervals [18,20,21,22], or even less [6].

Multiple pre-processing steps could be combined to create a more robust pre-processing method. Zhao et al. [22] used an FIR highpass filter to remove baseline drift and other low frequency noise. The signal was then decomposed into three levels by a biorthogonal spline wavelet, and a threshold was selected to remove the unwanted frequency components. The heartbeat of each signal was normalised to 75 bpm to remove the effect of natural variations in heartrate. Finally, the quality of the signal was measured [22], to verify only clean signals were passed to the next stage of the classifier.

2.2.2. Feature Extraction

Feature extraction involves the analysis of raw data to extract relevant features. These features may then be used to classify a signal into a class with similar features, and distinguish it from classes with different features. For ECG signals, a number of different features are of interest. These may exist as either time-domain or frequency-domain features, or even time-frequency features.

2.2.2.1. Time-Domain Feature Extraction

The detection of R peaks and/or RR intervals was a popular time domain feature to extract [6,17,24,25]. It was also reasonably common to detect the whole QRS complex [21,26], and some studies extended this to more features including the QT-interval and P-wave duration [21]. The time between subsequent R-peaks corresponds to one heartbeat, so if the time between two R-peaks was inconsistent, this could indicate an abnormality [6].

Wang et al. [24] extracted the RR intervals (time between subsequent R-peaks) from their ECG data. They then compared subsequent RR intervals and the ratio between these. This data was one of two sets of data used to train and test their CNN (the other being spectrograms). Hu et al. [6] also examined the difference in RR intervals, although just for AF and normal ECGs. Their results showed that AF signals had an

RR interval half the length of a normal signal, and there was greater variance between the RR intervals.

Bae et al. [25] discussed how a pair of derivative filters could be used to detect R-peaks and the QRS complex. They also demonstrated a noise detection algorithm which could be used to exclude contaminated R-peaks. It was suggested that developing a technique to measure the reliability of detected R-peaks and calculated RR intervals may be just as important as developing a QRS algorithm with a higher detection rate [25]. The application of this research was for real-time detection of arrhythmias, in which case it was especially important to remove the unreliable RR intervals [25].

Numerous papers report on the use of a K-Nearest Neighbour (KNN) algorithm to detect R-peaks [17,21,26]. KNN is a type of instance-based learning in which an object is assigned to the class of its k nearest neighbours, where k is an integer. For the case where k = 1, the object is assigned the same class as its single nearest neighbour [17].

Using KNN to detect either R-peaks or the QRS-complex have proven successful. He et al. [17] achieved a peak detection accuracy of 99.43%, Saini et al. [26] achieved QRS detection rates of up to 99.89%, and in a different paper [21] accomplished results of 100% for detecting QRS duration, heart rate, QT-interval, P-wave duration and PR-intervals.

The use of a method called the Pan-Tompkins algorithm has also been used to identify the QRS complex from ECG signals [4,20]. In this method, the signal is filtered and differentiated to remove noise and suppress the lower frequency components of the P and T-waves. Squaring further enhances the high frequency components, and a moving window integration extracts the slope of the R-wave [4]. Further threshold adjustment may be done to improve sensitivity.

Finally, it is also possible to detect the RR intervals using wavelets. For example, Venkatesan et al. [15] uses a Coiflet wavelet to detect the RR interval and possible R-peaks. Although wavelets are commonly used to extract features, they are mostly used to extract frequency or time-frequency information instead [4,8,24,27].

2.2.2.2. Frequency-Domain Feature Extraction

The main features of an ECG signal are contained within the frequencies of about 0.5 Hz to 30 Hz [6,18], with components outside of this range largely corresponding to noise.

Hu et al. [6] demonstrated that the maximum amplitude frequency component of an ECG may be an important feature. The maximum amplitude frequency component was consistently close to 1 Hz for normal signals, and much more volatile (ranging from 2 to 8 Hz) for ECG recordings with AF [6]. This study did only analyse two conditions, with no mention of whether or not this feature can be generalised to other abnormal conditions.

ECG signals are non-stationary data, and as such, their instantaneous frequency changes with time [16]. This means their properties can't be fully described just by using frequency-domain information [16]. Therefore, a means of combining this frequency information with time-domain information is required.

2.2.2.3. Time-Frequency Analysis

Spectrograms and scalograms are powerful tools in the analysis of ECG recordings [3,16,24,27]. They demonstrate how the frequency content of a non-stationary signal varies with time (refer back to Section 1.5.2 for further background information). Spectrograms and scalograms are important for this application as they can be saved as images for input in classifiers, such as the CNN [16,24].

Huang et al. [16] plotted spectrograms as 256×256 pixel images before using them to successfully train and test a CNN. These images were created using a STFT over 10 seconds of a recorded signal. Rashed-Al-Mahfuz et al. [27] produced scalograms of segments of ECG signal as input to a VGG16-based CNN. The results were compared to ones obtained using a Hilbert-Huang Transform (HHT), in which case the scalogram overperformed the HHT in all cases. Wang et al. [24] also used the WT to produce scalograms of ECG signals, to use as one input to their CNN (along with RR interval information).

Even without plotting a spectrogram, the WT can be used to decompose a signal into a series of wavelet coefficients [34]. Emanet [34] converted each signal into 265 wavelet coefficients, and MathWorks [23] selected 190 features. The idea behind this

is to represent the signals as a smaller number of data points, and for the case of [23] this meant decreasing the points down from 65536 points to just 190.

2.2.3. Classification with Machine Learning

Machine learning techniques can be used to classify ECG signals, based on a number of learned features. Preliminary research identified a number of possible classification methods, namely:

a. CNNs [16,24,27,28,29];

b. SVMs [15,23,30,31];

c. k-Nearest neighbour algorithm [22,32,33];

d. Random Forest algorithm [31,34,35,36]; and

e. Decision trees [6].

This review focusses on the two most relevant, those being CNNs and SVMs, with the other techniques being discussed more briefly.

2.2.3.1. Convolutional Neural Networks

CNNs are a type of deep learning model commonly used in image and data analysis, as well as disease classification [29]. See section 1.5.3 for further background information.

Huang et al. [16] reported an average accuracy of 99.00% with their 2D-CNN. This classifier used three layers of convolution and pooling. For comparison, they demonstrated a 1D-CNN which produced an accuracy of 90% when supplied with a similar sized test set. Wang et al. [24] also used a CNN with three convolution and pooling layers. Although they produced a high accuracy of 98.74%, the positive predictive value (PPV), sensitivity (SE) and F1-scores of their method were lower, with 70.75%, 67.47%, and 68.76%, respectively [24].

Rashed-Al-Mahfuz et al. [27] used a *VGG16* architecture, which consists of a CNN with 16 layers in order to classify an input image. They found accuracy was improved when a CWT scalogram was used instead of HHT spectrum. The accuracy of this classifier was also dependent on the number of classes to be distinguished between. When two, three or four classes were being distinguished the classifier could achieve a 100% accuracy, but had a lower accuracy of 99.9% when six classes were used [27].

Dokur et al. [28] considered the accuracy of both 1D-CNN classification, and the classification of ECG images using CNN which had been trained with Walsh functions. Walsh functions form an orthonormal basis (like trigonometric functions in Fourier analysis), but have a number of advantages including the ability to easily expand the number of learned classes [28]. The results found this method achieved an accuracy of 99.45% for the classification of 1D ECG signals, and a 98.7% accuracy for 2D ECG images.

Lih et al. [29] made use of a model called Long Short-Term Memory (LSTM) to improve the results obtained from their CNN. Although the training of this approach was time-consuming and required a sizeable amount of data, the system was able to achieve a high classification accuracy (97.33%) despite using signals with noise [29].

It was recommended that a pre-trained model with high-performance in a related task be used to reduce computational complexity [27]. Parts of the classifier can then be modified as needed to improve its performance.

2.2.3.2. Support Vector Machine

The SVM is a widely adopted pattern recognition, object identification, and image classification technique [15]. Venkatesan et al. [15] used an SVM classifier to sort ECG recordings into a normal and abnormal set based on a range of time-domain and frequency-domain features. This achieved an accuracy of 96% [15]. An example MATLAB program written by MathWorks [23] also obtained an accuracy of 97.96% with an SVM.

Zhang et al. [31] tested a couple different SVMs, including a kernel SVM (KSVM) and least-squares SVM (LS-SVM). Their results found the traditional KSVM to have the worst results, and the LS-SVM to be the most effective of the methods compared, with an accuracy of over 92%.

Li et al. [30] extended the idea of the SVM by experimenting with different ways in which it could be optimised. Particle swarm algorithm (PSO), genetic algorithm (GA) and the grid search algorithm (GS) were each used to optimise the SVM which was used to classify between six ECG beat types. The results for each were high, with average SE, specificities (SP), PPVs and accuracies each well above 95% [30] (see Table 2 for specific values).

2.2.3.3. Other Classification Methods

KNN algorithms can also be used to classify ECG recordings. Bouaziz et al. [32] used a KNN algorithm to classify 5 different types of ECG signal with a 98.71% accuracy. By using a fuzzy KNN, Castillo et al. [33] was able to obtain results of 95.33% for 5 signal classes, and raised this to 98% by combining the output of the KNN classifier with two other ANN classifiers in a 'fuzzy interference system'. Zhao et al. [22] achieved a 95% accuracy with a KNN network, after subjecting the signals to a robust pre-processing method (involving noise elimination, heartbeat normalisation and quality measurement).

Castillo et al. [22] investigated the effect of the integer k on the effectiveness of the classifier. Larger k values reduced the effect of noise, however they also blurred the boundaries between classes [22]. For this reason, Castillo et al. [22] found k = 1 to be the optimal solution, even though k = 1 and k = 3 produced similar results. Other studies found k = 3 was the best choice for their methods [32,33]. Furthermore, selecting k as an odd number was advised, since it prevents the issue of tied votes [32].

Random Forest algorithms (RaF) are comprised of recursively built classification trees, each of which casts a unit vote to determine the classification of input data [34]. The classification is the class which wins the most votes out of the entire forest. RaFs are resistant to noise, and not subject to overfitting, giving them good performance on a number of practical problems [34].

However, the reported results from RaF classification are mixed. Some studies found good results [34,35,36], and others found poorer results [31]. Emanet [34] claimed RaF to be fast, have excellent performance and no cross validation, making them useful for long-term ECG beat classification. Conversely, Zhang et al. [31] noted that RaF generalise poorly, making them far less effective than other methods, such as an SVM.

Yet more methods are mentioned in the literature. Hu et al. [6] used a decision tree algorithm to classify between AF and normal ECGs with high success. Celin and Vasanth [7] mentioned the use of an Adaptive Boosting algorithm, ANN and a Naïve Bayes classifier, and Jambukia et al. [4] briefly reviewed a number of neural networks.

2.2.4. Comparison of Classification Methods

Table 2 summarises the results published in the literature for a range of classification methods. This table has been sorted by year, simply to make it easier to display cases where one study tested multiple methods.

Table 2: Comparison of Results in the Literature

Researcher	Number of Features	Processing and Feature Extraction	Classifier Method	Performance Measures	Average Performance
M. Rashed-Al-Mahfuz et al. (2021) [27]	5	CWT and RR intervals	CNN	Accuracy SE SP AUC	99.90% 99.90% 99.98% 99.94%
	2 3 4 5	HHT and RR intervals	CNN	Accuracy Accuracy Accuracy Accuracy	95.75% 89% 81.38% 72.70%
T. Wang et al. (2021) [24]	5	CWT	CNN	Accuracy PPV SE F1-Score	98.74% 70.75% 67.47% 68.76%
Z. Dokur and T. Olmez (2020) [28]	11	-	CNN trained with Walsh Functions	Success	80%-100%
Y. Hu et al. (2020) [6]	2	signal splitting, bandpass filtering	Decision Tree	Accuracy Sensitivity Specificity	98.9% 97.93% 99.63%
H. Li et al. (2020) [30]	6	Wavelet packet decomposition	PSO-SVM	Accuracy SE SP PPV	97.78% 97.78% 99.63% 97.87%
			GA-SVM	Accuracy SE SP PPV	98.33% 98.33% 99.72% 98.42%
			GS-SVM	Accuracy SE SP PPV	98.89% 98.89% 99.81% 98.92%
O.S. Lih et al. (2020) [29]	4	--	CNN-LSTM	Accuracy SE SP PPV	98.51% 99.30% 97.89% 97.33%
J. Huang et al. (2019) [16]	5	Spectrogram	2D-CNN	Accuracy	99.00%
	5	Spectrogram	1D-CNN	Accuracy	90.93%
Y. Zhang et al. (2019) [31]	2	Various feature extraction methods	KSVM	Accuracy	89-92%
			LS-SVM	Accuracy	91-92%
			RaF	Accuracy	89-91%

F. Bouaziz et al. (2018) [32]	5	Wavelet denoising, DWT	KNN	Accuracy	98.71%
S. Celin and K. Vasanth (2018) [7]	2	Bandpass filtering	SVM	Accuracy SE SP	87.5% 75% 100%
	2	Bandpass filtering	Adaptive Booster	Accuracy	93%
	2	Bandpass filtering	ANN	Accuracy	94%
	2	Bandpass filtering	Naïve Bayes	Accuracy	99.7%
C. Venkatesan et al. (2018) [15]	2	Delayed error normalised LMS, and DWT	SVM	Accuracy	96%
M. Kropf et al. (2017) [36]	4	QRS detection, and other time-domain features	RaF	F1-score	81%
R. Mahajan et al. (2017) [35]	4	Genetic Algorithm	RaF	Accuracy	82.7%
Z. Zhao et al. (2013) [22]	-	highpass filter, wavelet thresholding, heartbeat normalisation, quality measure	KNN	Accuracy	95%
O. Castillo et al. (2012) [33]	5	Bandpass filter, segmentation, heartbeat normalisation	Fuzzy KNN	Classification rate	95.33%
N. Emanet (2009) [34]	5	DWT	RaF	Accuracy	99.8%
MathWorks [23]	3	Wavelet decomposition	SVM	Accuracy	97.96%

2.3. Review Conclusions

A number of conclusions can be drawn from the review at this stage. These encompass how to progress with all of pre-processing, feature extraction and classification of ECG signals.

First, in terms of pre-processing, it is important to do, however more research is needed to decide the best way to go about this. Bandpass filtering seems easy to implement, and has been used to some success, but is not the best method of removing noise from a complex signal like the ECG. More research is needed into how wavelet denoising and LMS filtering can be implemented, and the effectiveness of each.

The length of signal segments to be analysed does not seem to play a major role, since different methods have reported success with >30 seconds, 10 seconds and even

shorter segments. However, since some arrhythmias, like AF, are easily distinguished with variations in RR interval distances, it may be best to use longer signals to capture more of this information.

Second, feature extraction can be completed in a number of different ways, for both features in the time and the frequency domains. For some arrhythmias, it may be sufficient to extract only time- or frequency-domain features, but with the growing popularity of image-based classification methods (like the CNN), it is worth considering the time-frequency information available in spectrograms.

Finally, ECG classification can utilise a broad range of techniques, and a wide range of modifications to each of these techniques. SVMs have been identified as a suitable starting point, and further research has begun on these. However, CNNs look promising in the literature. These classifiers also have the advantage of being able to classify signals from the spectrogram images, which are easy to obtain with MATLAB. Hence, further research into these, and also their computational processing requirements, will be completed.

The literature boasts highly accurate results, with a range of processing, feature extraction and classification methods. Hence, it may be difficult to narrow down a "one best" classification algorithm.

It is worth mentioning that although most papers will quote the accuracy of their classification method, it is rarely suitable to summarise the results with this single metric. The precision and recall (or similarly sensitivity and specificity) give more insight into the actual effectiveness of a classifier. For example, a classifier which classifies all signals as normal will have a good recall for this, but a poor precision since it also classified many signals incorrectly. Suppose this classifier was used with a data set of 80% normal signals and 20% abnormal. Here, classifying all signals as normal would give a high accuracy, even though common sense tells us this classifier is no use at all. Hence papers which only quote an accuracy value may not be providing enough information about their results.

Note that the comparisons made in Table 2 do not take into account all information possible. For example, computational power and time have not been considered here,

although they are very real constraints for this project. More research into these, and other potential issues will be investigated as required.

In summary, based on the findings of the literature review, the project should progress in the following manner:

1. Begin by applying the raw PhysioNet data [2] to an existing processor which makes use of a SVM [23];
2. Pre-process the data using existing or simple (i.e. low-pass filtering) pre-processing techniques, and apply this data to the existing processor;
3. Experiment using other techniques to process and classify the data, such as a CNN;
4. Compare the results produced from each type to the existing classifier [23]; and
5. Develop a “best” combination of pre-processing and classification methods.

The effectiveness of the “best” pre-processing and classifier combination will be compared to that of results generated by running existing classifiers, and with those quoted in the literature, to determine the final conclusions of the project.

3. Method

The project so far, has taken the following form:

1. Gained a basic understanding of heart disease and ECGs;
2. Performed preliminary research on ECG analysis using ML techniques;
3. Wrote a one-page review of the topic to consolidate understanding;
4. Decided to focus on wavelet de-noising and SVM classification;
5. Identified a database and an existing SVM classifier, and examined the effectiveness of the classifier without pre-processing;
6. Identified other tools, such as spectrograms, and gained an understanding of how these could help with the project; and
7. Identified possible pre-processing techniques.

The following sub-sections will discuss how each of these steps was carried out, and any major challenges involved. See Section 4: Results for the results developed, and Section 5: Discussion for a discussion on the results and method.

At this time, the project is incomplete. The following steps identify plans for future development:

8. Apply the pre-processed data to the classifier, and compare the results to the un-processed data;
9. Explore other processing and classification techniques, such as CNN, and analyse accordingly;
10. Develop a “best” methodology for classifying ECG recordings; and
11. Compare the “best” methodology to reported results.

Meanwhile, a thesis was developed over the course of the project and an interim seminar has been presented, however discussion of these deliverables will not be discussed in this section.

3.1. Heart Disease and ECG Understanding

A basic understanding of the human body, and in particular the cardiovascular system, was previously obtained during courses undertaken for Biomedical Engineering. However, heart disease had not yet been analysed. It was important to gain a basic understanding in this area to give insight into the variety of problems which could be identified in the ECG recordings.

At this time, PhysioNet [2] was identified as a suitable database from which to obtain ECG recordings. The database contained 8528 ECG recordings classified into four groups: normal rhythm, atrial fibrillation (AF), other rhythm, and noisy signal. Hence, both normal rhythm and AF were researched further. A summary of these results is included in the background information (Section 1.5.1).

3.2. Preliminary Research on ML Techniques

ML techniques had not been learnt previously, so it was important to gain a rudimentary understanding of these. To avoid ML techniques which would not be applicable to the project, the search was limited to ML techniques which had already been applied to ECG analysis. A summary of these results was used to write the background information (Section 1.5.3).

This led to the identification of SVM classification. This classification method had also been recommended by the Supervisors, so a decision was made to focus on it. CNNs were also identified as another possible classification tool. Although this has not been examined further as of yet, plans to do so are in place for Semester 2.

3.3. One-Page Review

A one-page review was developed to succinctly consolidate and demonstrate understanding of the topics covered at that time. Although lacking much technical detail, this review identified a number of good references and acted as a starting point for the literature review.

A copy of the one-page review is included in Appendix A.

3.4. Wavelet De-Noising and SVM Classifier

Wavelet de-noising had been recommended as an appropriate pre-processing method due to the similarity in shape between the wavelet and the ECG. Research into wavelets was completed to develop an understanding of how they could be used.

Using an SVM classification method was also recommended. Hence, research into how the SVM works was conducted.

3.5. Existing SVM Classifier in MATLAB

An example MATLAB script which used both a wavelet de-noising and an SVM classifier was identified. This code was run without change to replicate the results provided by the example. This was deemed to be a suitable benchmark to compare future results to.

The data collected from the PhysioNet database [2] was then processed to be applied to this classifier. Processing involved ensuring all signals were of the same length and required data (i.e. the signals and their classification) were stored in the correct data type. The example code also had to be modified extensively to be compatible with our data. These required changes were:

- Changing of the classes from the data used to our data classes;
- Adding of additional variables to compensate for the change in signal classes; and,
- Modification of window length given our data contained fewer data points.

Other changes may need to be made to improve the accuracy of the classifier. At this stage these changes have been identified, but not implemented:

- Altering the features and number of features to extract; and,
- Altering the parameters used for the SVM.

Further discussion on how these changes were made is included below, and the results produced are included in the Results section. A copy of the original code can be found here: https://au.mathworks.com/help/wavelet/ug/ecg-classification-using-wavelet-features.html [23], and the modified code used to collect results is included in Appendix B1 Modified MathWorks Example.

3.5.1. Preparing the Data

The data downloaded from the PhysioNet database [2] comprised of a folder of individual MATLAB vectors. Each vector represented one ECG recording. These recordings had a consistent sampling frequency (300 Hz [2]), but did not have a consistent length. As such, the first step in preparing the data was to select a data length. A number of options were examined:

1. Truncating each ECG recording to the length of the shortest recording (2712 samples ≈ 9 seconds, in this case);
2. Selecting a length and truncating all recordings longer than this while removing all recordings shorter than this:
    i. Length of 3000 samples, equivalent to 10 seconds;
    ii. Length of 6000 samples, equivalent to 20 seconds;
    iii. Length of 9000 samples, equivalent to 30 seconds; and,
3. Duplicating the recording as many times as required to ensure each signal has the same number of points as those in the example code (i.e. 65536 data points).

Other studies had used 10 seconds of ECG recording (or split recordings into 10 second blocks) [18,20,21,22], so using 3000 samples was chosen for this reason. Over half of the signals had 9000 data points, so both 6000 and 9000 samples could be used without significantly reducing the size of the data set. It was theorised that longer signals would result in a better classification, hence the range of signal lenths.

The PhysioNet data [2] contained four different ECG classifications. These were normal rhythm (N), atrial fibrillation (A), other arrhythmia (O), and noisy recording (~). After some experimentation, the number of different classes was also altered to investigate how this would affect the results:

4. All four data classes were included;
5. The noisy data class was removed, the other three were included; and,
6. The noisy data was removed and the AF and other arrythmia classes were combined into a single ‘abnormal’ class.

This would allow investigation as to whether the number of classes the SVM had to discriminate between had an impact on the results it was able to produce.

In each case the ECG recordings and their labels were saved in a data structure which matched that in the example. This was done to reduce the number of modifications which had to be made to the example code.

It was also discovered that the way the example code separates the data into a training set and a test set requires that the data be sorted in order of its class (i.e. all normal signals first, then all AF signals, etc.). Hence, after the other processing steps mentioned above, the data was sorted according to its classification.

3.5.2. Modifying the Example Classes

The example code contained ECG recordings of three different types: normal sinus rhythm (NSR), arrhythmia (ARR) and congestive heart failure (CHF), whereas the data downloaded from PhysioNet [2] contained four different types. Due to this, there were a number of cases where the example code classes had to be changed from {ARR, CHF, NSR} to {A, O, N, ~}.

Other points in the code required additional variables to be added to enable the fourth class to also be identified. This also required altering matrix indices which used to go up to 3, but now needed to include 4.

3.5.3. Magic Numbers

The example code contained a number of parameters which were very specifically chosen to fit with the data, yet were hard-coded. The most obvious of these (since it produced an error, whereas the other magic numbers didn't) was the window length used for feature extraction. This was hard-coded as 8192 samples, or one eighth the length of the supplied signals. This was modified to a formula which found the (rounded-down) length of an eighth of the length of an ECG data signal being processed.

Other magic numbers are included in the code, such as the autoregressive model order, and the polynomial order of the SVM classifier. These numbers do not throw an error when the code is executed, however they may not be the optimal parameters for the data used. Further research will determine the meaning of each of these parameters, and which values may be optimal.

3.6. Spectrograms

Spectrograms were also identified as a possible tool to help with the classification of ECG signals. Research was done to identify the information which could be gleaned from the spectrogram, and how to apply these to an ECG signal.

The spectrogram function in MATLAB was used to plot the spectrograms of the ECG signals. These were then examined for patterns manually. Research into CNNs suggest they are powerful tools for classifying images. Hence further research will be needed to learn how to transform the MATLAB results into images which can be fed to a CNN for classification.

3.7. Data Pre-Processing

Preliminary work on pre-processing the data was started. One method of pre-processing which was identified was the Pan-Tompkins algorithm [4,20]. This involves the following steps:

1. Removal of DC offset;
2. Band pass filtering to reduce noise from the ECG signal;
3. Differentiation to find high slopes which usually identify R-peaks and suppresses low frequency components of the P and T waves;
4. Squaring to further enhance high frequency components;
5. Averaging the signal with an averaging function; and,
6. Moving window integration to extract the slope of the R wave.

Each of these steps were completed using MATLAB. At each step the results produced were compared to those in [20] to verify the code was achieving the correct outcome. This process is described in more detail below.

The first step was to remove the DC offset of the ECG by subtracting the mean of all data points from each point. The next step was to apply a bandpass filter to the data to remove noisy high-frequency components, and low frequency artifacts (such as breathing) from the ECG recordings. In theory, this would make it easier to classify the ECG signals and the spectrogram images easier to examine.

The signal then needed to be differentiated to find the high slopes which are indicative of the R-peak on the ECG waveform. This was done in MATLAB using the **gradient** function. This was then squared to make these high frequency changes even more noticeable, and ensure the whole signal was positive.

The averaging function was realised using a moving window filter. A window size of 15 was found to be sufficient. This function smoothed the spikes in the signal into gentle peaks. The signal integration was also completed with a moving window. In this case a window of width 30 was moved across the signal. The result was further smoothing of the signal peaks.

The results of this process were then able to be analysed.

3.8. Identification of Other Classifiers

The aims of this project include comparing classification methods to find a "best" way to classify ECG signals. Much work is still to be done in this area, but one classifier which utilises a CNN has been identified [37]. This classifier is programmed in MATLAB, and in the example, it is used to learn how to identify handwritten digits. It may be possible to modify this classifier to classify our ECG data based on their spectrograms.

4. Results

4.1. MATLAB SVM Classifier Example

Without altering any parameters or data, the identified MATLAB classifier example produced a test accuracy of 97.96% [23]. This classifier was given 162 ECG signals, of which 70% were used to train the machine, and the other 30% were used for testing. The example contained three different types of signal: NSR, ARR and CHF. In total, 96 of these recordings were from people with ARR, 30 were from people with CHF, and 36 recordings were from people with NSR.

The features of each signal were extracted using wavelets. This decreased each signal from 65536 data-points in the time-domain to just 190 features [23]. This made the data quicker and easier to process. There are differences in each feature between the classes. For example, Figure 9 shows the variance in the second-lowest frequency wavelet sub-band [23]. While no one feature alone is enough to separate all classes, the idea is to have a rich enough set of features to make this process accurate.

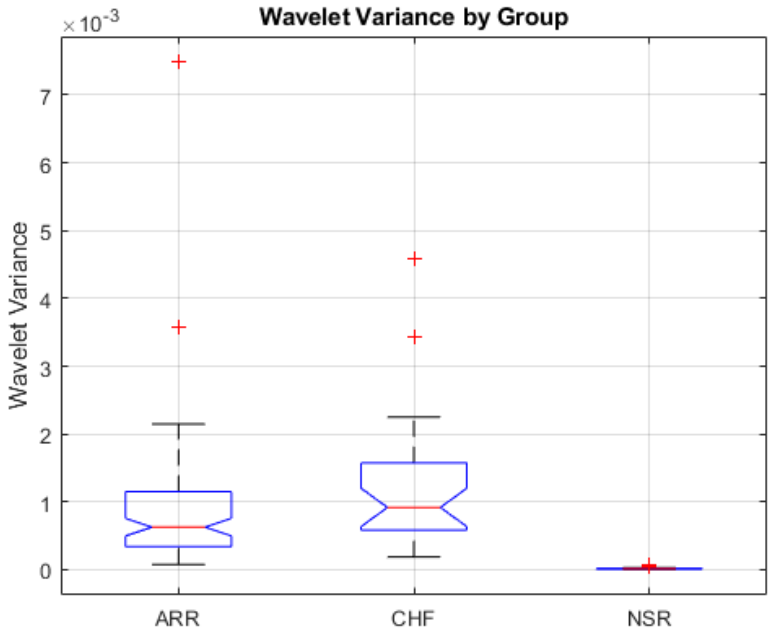


Figure 9: Wavelet Variance by Group in MATLAB Example [23]

When the example code was run exactly as-is, the results in Table 3 were produced. These results show how well the trained classifier sorted the testing set of data. Precision is defined as the number of correct positives divided by the number of positive results, i.e. the proportion of data records assigned to a label that actually belong to that label [23]. Conversely, recall is defined as the number of correct labels

divided by the number of labels for a given class, i.e. the proportion of all records belonging to a class that got labelled as that class [23]. The F1-score is the harmonic mean of precision and recall, so it can better summarise the classifier performance. The results show all ARR signals were correctly identified, and no other signals were accidentally assigned this label, although one CHF signal was misclassified as an NSR signal.

Table 3: MATLAB Example Classifier Results [23]

	Precision	Recall	F1_Score
ARR	100	100	100
CHF	100	88.889	94.118
NSR	91.667	100	95.652

These results were considered a good baseline for further study since they used the classification methods recommended, and although quite good there was still room for improvement.

4.2. MATLAB Classifier Modification

After verifying the example code, the next stage of analysis was to fit the results collected from the PhysioNet database [2] to the classifier. In essence, this required selecting a time interval (or equivalently number of data points) and truncating or extending the data to fit this, deciding how many classes of signal to classify for, and then altering the example code to fit the data prepared. See the 23Method Section for a more detailed discussion of the steps taken.

As outlined in the Method, a number of different cases have been input to the example MATLAB SVM classifier. Table 4 summarises these results, and they are described in more detail in the following sub-sections.

Table 4: Summary of Results from the Example MATLAB SVM Classifier

Name	Samples per Signal	Number of Classes	Number of Signals					Accuracy
			Total	A	O	N	~	
MATLAB Example Data	65536	3	162	-	-	-	-	97.96%
Truncated to Shortest ECG Recording	2712	4	8527	738	2456	5049	284	62.98%
Truncated to 3000 Samples	3000	4	8511	737	2452	5039	283	62.10%
Truncated to 6000 Samples	6000	4	7911	654	2348	4741	168	66.02%
6000 samples, Three Classes	6000	3	7743	654	2348	4741	0	66.88%
6000 samples, Two Classes	6000	2	7743	0	3002	4741	0	70.34%
Truncated to 9000 Samples	9000	3	7415	625	2262	4528	0	65.87%
9000 samples, All Classes	9000	4	7560	625	2262	4528	145	65.06%
Extended to 65536 Samples	65536	4	8527	738	2456	5049	284	63.78%

Note that as execution time, and CPU and memory requirements were not recorded as all cases were able to run successfully on the PC (even while other processes were running, such as a web browser and text document). However, it should be noted that data sets with more samples took longer to process than ones with less samples, and that memory requirements were also higher for the longer signals.

4.2.1. Truncated to Shortest ECG Recording

The overall accuracy of the classifier when ECG recordings were truncated to the shortest signal was about 63%. Table 5 shows the precision, accuracy and F1-score for each class.

Table 5: Modified Classifier Results for Recordings Truncated to Shortest Signal

	Precision	Recall	F1_Score
A	48.78	9.0498	15.267
O	46.039	29.172	35.714
N	67.095	89.505	76.697
~	68.966	23.529	35.088

4.2.2. Truncated to 3000 Samples

Previous studies had suggested 10 seconds of ECG recording is enough to develop an accurate classifier [18,20,21,22]. The sampling rate of the data used was 300 Hz [2], so 10 seconds of data corresponded to 3000 samples. This was not especially different to the number of samples used when all signals were truncated to the shortest recording, so the similar accuracy of 62% was expected. Table 6 shows the precision, recall and F1-scores for each class.

Table 6: Modified Classifier Results for Recordings Truncated to 3000 Samples

	Precision	Recall	F1_Score
A	40	9.0498	14.76
O	41.86	24.457	30.875
N	66.931	89.418	76.557
~	62.963	40	48.921

4.2.3. Truncated to 6000 Samples

Processing on data recordings truncated to 6000 samples was chosen to investigate the impact of different numbers of classes since it was both more accurate than using fewer samples, and quicker to process than using a greater number of samples.

4.2.3.1. Classification of all Four Classes

The accuracy of classifying all four classes using 6000 samples was 66%. The precision, recall and F1-scores are detailed in Table 7.

Table 7: Modified Classifier Results for Recordings Truncated to 6000 Samples in 4 Classes

	Precision	Recall	F1_Score
A	59.184	14.796	23.673
O	51.812	34.517	41.432
N	69.912	89.873	78.646
~	61.538	32	42.105

It was thought that the number of classes, and indeed nature of the class of noisy signal, may have a negative impact on the results. Hence, this data was modified to include fewer different classes.

4.2.3.2. Classification of Three Classes

Since the noisy signals were too noisy to be classified into one of the other three classes, it was decided that these should be removed from the data to be classified. These signals were simply removed omitted from the list of recordings.

The accuracy of this was just shy of 67%, and so was not a notable improvement over the four-class case. The precision, recall and F1-scores are shown in Table 8.

Table 8: Modified Classifier Results for Recordings Truncated to 6000 Samples in 3 Classes

	Precision	Recall	F1_Score
A	54.545	15.306	23.904
O	53.881	33.523	41.331
N	70.366	90.506	79.176

4.2.3.3. Classification of Two Classes

The results with three classes were not much better than the results with four classes, so it was decided to test the results with just two classes: normal and abnormal. To reduce coding changes these were labelled as ‘N’ (normal) and ‘O’ (abnormal). To do this, the noisy recordings were removed, and all ‘A’ class signals were relabelled as ‘O’ signals. The result being the ‘O’ class contained all abnormal signals and the ‘N’ class contained all normal signals.

This produced and accuracy of just over 70%, which is an increase over using three or four classes, but still lower than expected. Table 9 shows the precision, recall and F1-scores of both signal classes.

Table 9: Modified Classifier Results for Recordings Truncated to 6000 Samples in 2 Classes

	Precision	Recall	F1_Score
O	70.152	40.954	51.717
N	70.395	88.959	78.596

4.2.4. Truncated to 9000 Samples

The low increase in the accuracy produced by 6000 samples prompted the testing of another signal length, 9000 samples (or 30 seconds). Approximately 75% of all the signals in the PhysioNet database [2] had at least 9000 samples, so truncating to this length did not reduce the size of the dataset much.

4.2.4.1. Classification of all Four Classes

Again, classification wit four classes was done. This produced an accuracy of 65%, which was actually lower than the results obtained with the signals containing 6000 samples. Detailed results are shown in Table 10.

Table 10: Modified Classifier Results for Recordings Truncated to 9000 Samples

	Precision	Recall	F1_Score
A	44.776	16.043	23.622
O	49.438	32.401	39.146
N	69.896	89.249	78.396
~	61.905	30.233	40.625

4.2.4.2. Classification of Three Classes

Similarly, the noisy signals were omitted to bring the number of classes down to 3. The results can be seen in Table 11, and the overall accuracy was 65.9%. This was not a notable improvement, and was worse than the results obtained for 3 classes with 6000 samples.

Table 11: Modified Classifier Results for Recordings Truncated to 9000 Samples in 3 Classes

	Precision	Recall	F1_Score
A	43.284	15.508	22.835
O	51.136	33.137	40.214
N	70.53	89.175	78.764

4.2.5. Extended to 65536 Samples

The example data contained 65536 samples. Hence, it seemed reasonable to see if this length of data had any bearing on the accuracy of the results produced. The data was extended by adding copies of the recording to the end of it until the data was the required number of samples.

The accuracy of this was 63.8%, and the precision, recall and F1-scores can be found in Table 12.

Table 12: Modified Classifier Results for Recordings Extended to 65536 Samples

	Precision	Recall	F1_Score
A	47.368	16.29	24.242
O	52.247	25.237	34.035
N	66.618	90.627	76.79
~	56.923	43.529	49.333

This approach was completed first since it did not require as many changes to the original code. Once a greater understanding of the code was developed, the other approaches mentioned were done.

4.3. Modified Pan-Tompkins Algorithm

The results of using the modified Pan-Tompkins Algorithm to process one signal is shown in Figure 10.

Panel (a) shows the original ECG signal with the DC component removed. This means the baseline of the signal rests at 0.

Panel (b) shows the ECG after filtering with a bandpass filter. This has minimal effect on the appearance of the signal since there was little noise in the original signal.

Panel (c) shows the signal after derivative filtering. This makes the large peaks (R-peaks of the ECG) stand out, and hides the smaller peaks due to the P and T waves.

Panel (d) shows the signal after squaring. This further highlights the large peaks of the R waves.

Panel (e) shows the signal after averaging. This converts the signal into a series of peaks corresponding to the R-peaks.

Panel (f) shows the signal after integration which smooths these peaks.

Figure 11 shows how the signal produced at the end of this sequence corresponds to the recorded ECG signal. Notice that each of the peaks corresponds to a QRS complex in the signal and that other peaks (even the noise after the first beat) are not counted. Hence, this algorithm is able to identify the R-peaks in an ECG recording. This can be a useful first step in classifying whether or not a signal is abnormal since irregular heartbeats often signify an abnormal condition.

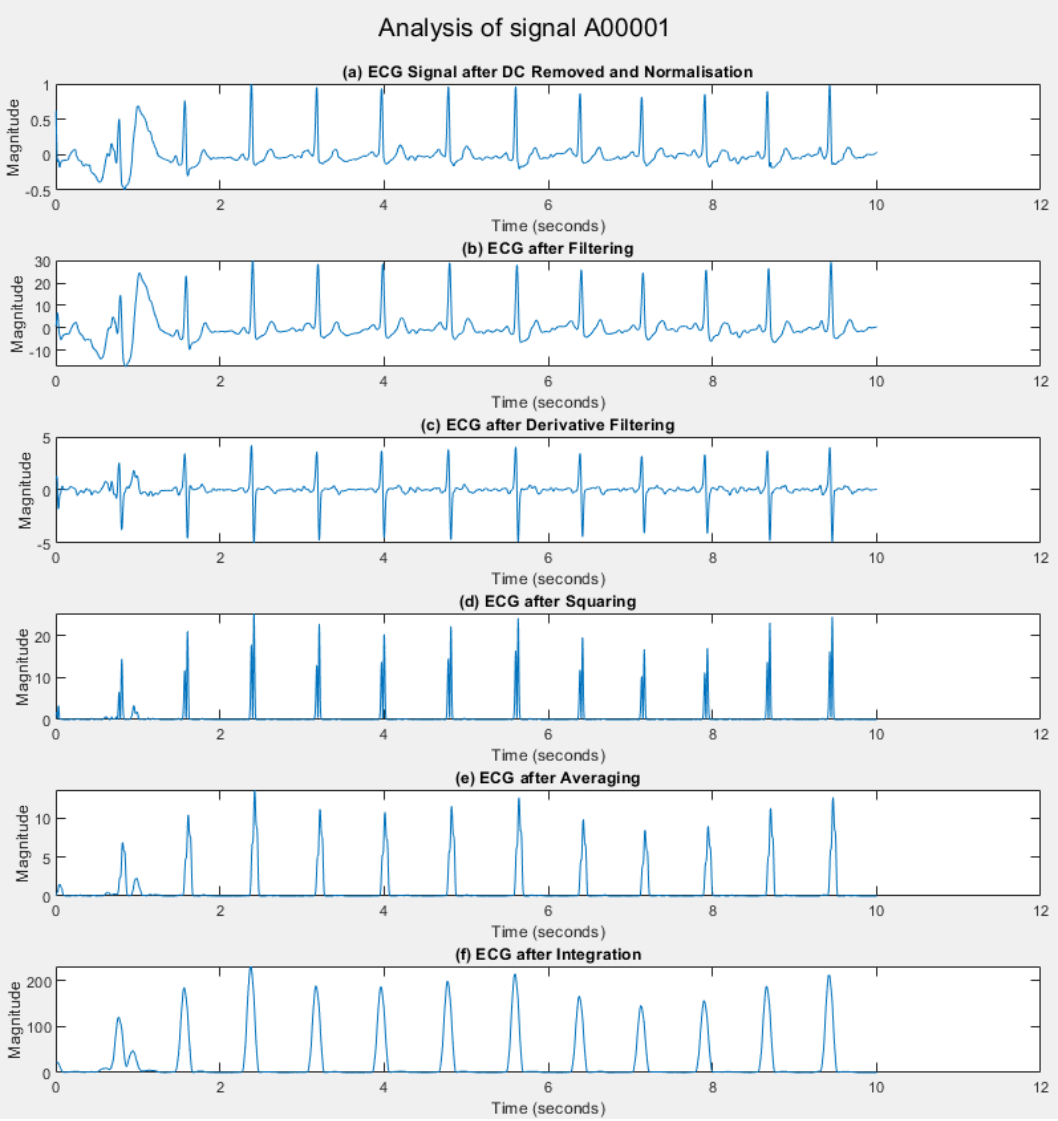


Figure 10: Modified Pan-Tompkins Algorithm Stages

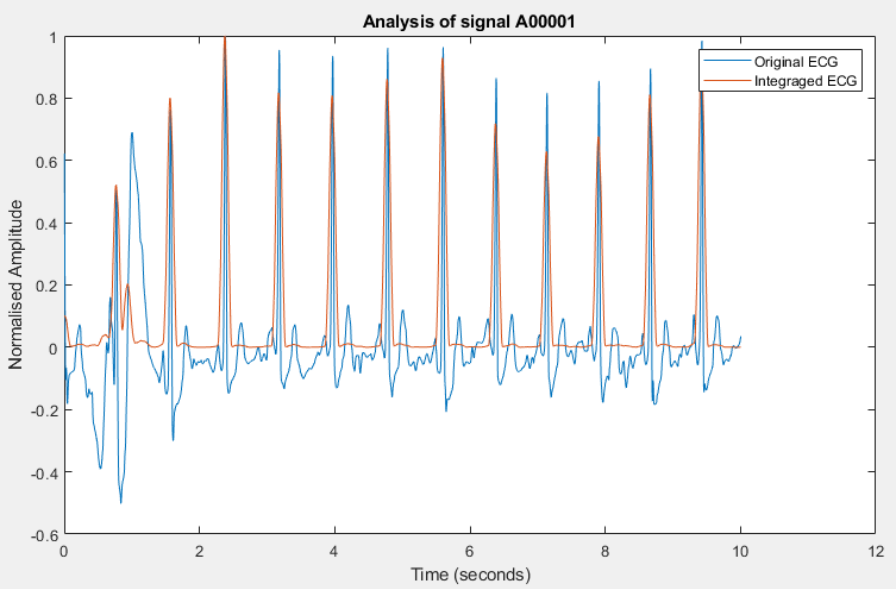


Figure 11: Identified R-Peaks of ECG Signal

5. Discussion

This section focuses on the results obtained by modifying the MATLAB SVM Classifier [23], as well as a quick discussion of the results from the modified Pan-Tompkins algorithm.

5.1. MATLAB Wavelet Denoising and SVM Classification

Table 4 shows the classifier was not very accurate in classifying our data.

It was expected that truncating all signals to the smallest recording length and truncating all signals to 10 seconds (3000 samples) would yield similar results, since the number of samples per signal was similar in each case. The accuracy was indeed less than 1% different for each case, with truncating to the shortest being slightly more accurate. It was thought that this was due to having a greater number of signals, rather than due to the length of the data.

Using the longer data length of 20 seconds (6000 samples) achieved results which were slightly better than with 10 seconds of recorded data. This suggests that the greater information stored in longer recordings may be helpful in achieving more accurate results. However, extending the data to 30 seconds (9000 samples) decreased the classifier accuracy when compared to the 6000 samples case. More research is required to understand why our results show these counterintuitive differences, and how the code can be further modified to produce better results.

Furthermore, extending the recordings to the same length as the example data proved to be no more effective at accurately classifying the data than truncating each recording to the shortest length. Meanwhile, this still took much longer to collate and classify than the shorter signal lengths did.

It should be noted that there may be errors with this method of data extending which could be improved upon. For example, the signals used in the Example Code [23] do not specify the sampling frequency of the signals. The databases this data was collected from has a few different sampling frequencies (i.e. one is sampled at 360 Hz, and the other at 250 Hz). It is entirely possible that these differences are hard-coded into the Example Code and have not been altered to fit our data.

The number of classes the classifier must discriminate between also seems to have an impact on the accuracy of the classifier. Our results found the classifier was 4% more accurate when distinguishing between 2 classes as opposed to 4 (for the 6000 samples case). However, as this result is small, it is uncertain whether it is indeed better or is simply due to chance. Further investigation into the reasons behind the high inaccuracies achieved is needed first.

This inaccuracy of the classifier may have much to do with the nature of the modifications made to fit the classifier to our data. Many of the parameters in the original code were selected based on published research but were not meant to be an exhaustive or optimised list of features [23]. Additionally, since our data differs to that in the Example Code, many parameters and variables have had to be modified to fit our data. In many cases these modifications have been made ad hoc to compensate for errors, rather than having been chosen carefully. Hence, further research will be needed to learn how these parameters may be altered to best suit our data.

It has also been suggested that it would be beneficial to ensure there are an equal number of each signal class to be examined. There are two ways this could be done:

1. Use all signals of the smallest class, and only use this number of recordings for each of the other classes; or,
2. Duplicate signals in the smaller classes so each class can have the same number of signals.

The impacts of these options have not yet been analysed, however duplicating the signals may not be a reliable means of testing a classifier. If one copy of a recording ends up in the training set and the other in the test set, the classifier will classify it correctly, since its set of features will exactly match the features of a signal the ML was trained with. Hence this solution is likely to produce unreliably high results. On the other hand, reducing the dataset may make it more difficult to properly train the ML.

5.2. Modified Pan-Tompkins Algorithm

This algorithm was quite accurate in being able to identify the R-peaks of the ECG recordings. Unfortunately, the usefulness of this result is not yet known. This did prove helpful in creating a suitable time window with which to create spectrograms of the signal. However, in this case it is better to perform a wavelet transform on the data to produce a scalogram than it is to produce a spectrogram with STFT, so this may not be needed.

There is a possibility of making use of features extracted by this algorithm (i.e. RR intervals, and variance between these intervals), in conjunction with other features obtained by the WT or scalogram. This is similar to the method used by Wang et al. [24], which produced an accuracy of greater than 98%. Signals with abnormally short RR intervals, or with inconsistent RR intervals could be quickly deemed abnormal, hopefully reducing the number of these which are mistakenly classified as normal rhythms.

6. Conclusions (Preliminary)

So, can we teach a machine to be a cardiologist? From the results gathered so far, training ML to accurately classify ECG signals is promising. Although our results are not as favourable as those in the literature, hopefully with more research effort a similarly reliable classifier can be achieved.

There has been success in modifying an existing MATLAB SVM classifier to analyse the data obtained from the PhysioNet Database [2]. Although the accuracy of this classifier when analysing these signals is low, it may be improved by selecting more suitable parameters and a more optimised set of features. Pre-processing is yet to be applied to these signals also, and this should lead to a better classification accuracy.

A range of other methods have also been identified which offer promising classification results. A MATLAB CNN classifier [37] has been identified and will be modified to classify the ECG scalogram data over the coming weeks.

The implementation of a modified Pan-Tompkins algorithm has also been successfully produced and applied to the ECG signals. The possibility of using this data as alongside other features is currently being investigated.

7. Project Status

At the time of writing, the project has made good progress, however there is still much work to be done. So far, the achievements made include:

- A literature review of ML methods for pre-processing, extracting features of, and classifying ECG recordings;
- Successful implementation of a modified Pan-Tompkins algorithm;
- Identification of an existing MATLAB classifier, which uses wavelet denoising and an SVM;
- Modification of this classifier to analyse data collected from the PhysioNet database; and,
- Preliminary exploration of other techniques for both pre-processing and classification of ECG signals.

The final goal is to design a highly accurate classifier to identify abnormalities in the ECG signals. A few specific steps have been identified for the following months:

- More rigorous modification of parameters in the existing MATLAB classifier to produce more accurate results;
- Implementation of pre-processing steps to improve the accuracy of the classifier; and,
- Analysis of other existing classification methods, in particular to begin with the example CNN classifier.

This is not a complete list of further requirements, as additional steps may be added based on project developments. A critical comparison of the performance of the methods investigated will be completed prior to project close.

Along with these further points it should be noted that the literature review will continue to be developed as further research is conducted.

8. Definitions

Term	Meaning
PhysioNet	Database which the examined ECG recordings were downloaded from
P-wave	ECG feature corresponding to contraction of the atria (refer to Figure 1)
QRS Complex	ECG feature corresponding to the contraction of the ventricles (refer to Figure 1)
R-peak	The characteristic, (usually) highest peak on an ECG waveform (refer to Figure 1)
RR interval	The time between subsequent R-peaks in an ECG recording
T-wave	ECG feature corresponding to repolarisation of the ventricles (refer to Figure 1)
VGG16	A convolutional neural network architecture with 16 layers

9. Abbreviations

Abbreviation	Meaning
AF	Atrial Fibrillation
ANN	Artificial Neural Network
ARR	Arrhythmia
AUC	Area Under Region of Convergence Curve
bpm	Beats per minute
CHF	Congestive Heart Failure
CNN	Convolutional Neural Network
CVD	Cardiovascular Disease
CWT	Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
ECG	Electrocardiogram
FIR	Finite Impulse Response
FT	Fourier Transform
GA	Genetic Algorithm
GS	Grid Search Algorithm
HHT	Hilbert-Huang Transform
KSVM	Kernel Support Vector Machine
LMS	Least Mean Squares
LS-SVM	Least Squares Support Vector Machine
LSTM	Long-Short Term Memory
ML	Machine Learning
NSR	Normal Sinus Rhythm
PPV	Positive Predictive Value
PSO	Particle Swarm Optimisation
RaF	Random Forest
SC-LNLMS	Self-Correcting Leaky Normalised Least Mean Squares
SE	Sensitivity
SP	Specificity
STFT	Short Time Fourier Transform
SVM	Support Vector Machine
WHO	World Health Organisation

10. References

[1] W.H.O., 2017, *Cardiovascular Diseases (CVDs)*, Accessed: 24 May 2021, [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[2] G. Clifford, C. Liu, B. Moody, L. Lehman, I. Silva, A. Johnson, R. Mark; 2017, *AF Classification from a Short Single Lead ECG Recording – The PhysioNet Computing in Cardiology Challenge 2017*, Accessed: 8 March 2021, [Online], Available: https://www.physionet.org/content/challenge-2017/1.0.0/

[3] P.S. Addison, *Wavelet Transforms and the ECG: a Review*, in Physiological Measurement, vol. 26, 2005; [Online], Available: https://iopscience.iop.org/article/10.1088/0967-3334/26/5/R01/pdf

[4] S.H. Jambukia, V.K. Dabhi, H.B. Prajapati, *Classification of ECG Signals using Machine Learning Techniques: a Survey*, IEEE, 2015; [Online], Available: https://ieeexplore.ieee.org/abstract/document/7164783

[5] ecgwaves.com, *Clinical ECG Interpretation*, 2021, Accessed: 18 May 2021, [Online], Available: https://ecgwaves.com/course/the-ecg-book/

[6] Y. Hu, Y. Zhao, J. Liu, J. Pang, C. Zhang, P. Li, *An Effective Frequency-Domain Feature of Atrial Fibrillation Based on Time-Frequency Analysis*, in BMC Medical Informatics and Decision Making, vol. 20, 2020; [Online], Available: https://link.springer.com/article/10.1186/s12911-020-01337-1

[7] S. Celin, K. Vasanth, *ECG Signal Classification Using Various Machine Learning Techniques*, in Journal of Medical Systems, vol. 42, 2018; [Online], Available: https://link.springer.com/article/10.1007/s10916-018-1083-6

[8] S.R. Messer, J. Agzarian, D. Abbott; 2001; *Optimal Wavelet Denoising for Phonocardiograms*, in Microelectronics Journal, vol. 32, issue 12, 2001, pp 931-941; [Online], Available: https://www.sciencedirect.com/science/article/pii/S0026269201000957

[9] O. Faust, U.R. Acharya, H. Adeli, A. Adeli; 2015, *Wavelet-Based EEG Processing for Computer-Aided Seizure Detection and Epilepsy Diagnosis*, in Seizure, vol. 26, 2015, pp 56-64; [Online], Available: https://www.sciencedirect.com/science/article/pii/S1059131115000138

[10] B. Grossfeld, *Deep Learning Vs Machine Learning: a Simple Way to Learn the Difference*, 7 Apr 2021, Accessed: 24 May 2021, [Online], Available: https://www.zendesk.com/blog/machine-learning-and-deep-learning/

[11] W.S. Noble, *What is a Support Vector Machine?*, in Nature Biotechnology, vol. 24, 2006; [Online], Available: https://www.ifi.uzh.ch/dam/jcr:00000000-7f84-9c3b-ffff-ffffc550ec57/what_is_a_support_vector_machine.pdf

[12] R. Gholami, N. Fakhari, *Support Vector Machine: Principles, Parameters, and Applications*, in Handbook of Neural Computation, 2017, pp 515-535; [Online], Available: https://www.sciencedirect.com/science/article/pii/B9780128113189000272

[13] L. Chang, Z. Zhang, L. Ye, D. Friedrich, *Synergistic Effects of Nanoparticles and Traditional Tribofillers on Sliding Wear of Polymeric Hybrid Composites*, in Tribology of Polymeric Nanocomposites, 2nd ed., 2013, pp 49-89; [Online], Available: https://www.sciencedirect.com/science/article/pii/B9780444594556000039

[14] S. Saha, *A Comprehensive Guide to Convolutional Neural Networks – the ELI5 Way*, 16 Dec 2018, Accessed: 24 May 2021, [Online], Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[15] C. Venkatesan, et al.; *ECG Signal Preprocessing and SVM Classifier-Based Abnormality Detection in Remote Healthcare Applications*; IEEE, 2018; Accessed 20 March 2021; [Online] DOI: 10.1109/ACCESS.2018.2794346

[16] J. Huang, B. Chen, B. Yao, W. He, *ECG Arrhythmia Classification Using STFT-Based Spectrogram and Convolutional Neural Networks*, in IEEE Access, vol. 7, 2019; [Online]. Available: https://ieeexplore.ieee.org/document/8759878

[17] R. He, K. Wang, Q. Li, Y. Yuan, N. Zhao, Y. Liu, H. Zhang; *A Novel Method for the Detection of R-Peaks in ECG Based on K-Nearest Neighbours and Particle Swarm Optimisation*, in EURASIP Journal on Advances in Signal Processing, 2017; [Online]. Available: https://link.springer.com/article/10.1186/s13634-017-0519-3

[18] J. Wang, P. Wang, S. Wang, *Automated Detection of Atrial Fibrillation in ECG Signals Based on Wavelet Packet Transform and Correlation Function of Random Process*, in Biomedical Signal Processing and Control, vol. 55, 2020; [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1746809419302435

[19] A. Khiter, A.B.H. Adamou Mitiche, L. Mitiche, *Muscle Noise Cancellation from ECG Signal Using Self Correcting Leaky Normalised Least Mean Square Adaptive Filter under Varied Step Size and Leakage Coefficient*, in Traitement du Signal, vol. 37, no. 2, 2020, pp. 263-269; [Online], Available: https://doi.org/10.18280/ts.370212

[20] Y. Palaniappan, V.A. Vishanth, N. Santhosh, R. Karthika, M. Ganesan; 2020, *R-Peak Detection Using Altered Pan-Tompkins Algorithm*, IEEE, 2020; [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9182298

[21] I. Saini, D. Singh, A. Khosla, *Delineation of ECG Wave Components Using K-Nearest Neighbour (KNN) Algorithm: ECG Wave Delineation Using KNN*, IEEE, 2013, [Online], Available: https://ieeexplore.ieee.org/document/6614392

[22] Z. Zhao, L. Yang, D. Chen, Y. Luo, *A Human ECG Identification System Based on Ensemble Empirical Mode Decomposition*, in Sensors, vol 13, issue 5, 2013, pages 6832-6864, [Online], Available: https://doi.org/10.3390/s130506832

[23] MathWorks, 2018, *Signal Classification Using Wavelet-Based Features and Support Vector Machines*, Accessed: April 2021, [Online]. Available: https://au.mathworks.com/help/wavelet/ug/ecg-classification-using-wavelet-features.html

[24] T. Wang, C. Lu, Y. Sun, M. Yang, C. Liu, C. Ou, *Automatic ECG Classification Using Continuous Wavelet Transform and Convolutional Neural Network*; in Entropy, vol. 23, issue 1, 2021; [Online]. Available: https://www.mdpi.com/1099-4300/23/1/119/htm

[25] T.W. Bae, K.K. Kwon, *Efficient Real-Time R and QRS Detection Method Using a Pair of Derivative Filters and Max Filter for Portable ECG Device*, in Applied Sciences, vol. 9, issue 19, 2019; [Online], Available: https://www.mdpi.com/2076-3417/9/19/4128/htm

[26] I. Saini, D. Singh, A. Khosla, *QRS Detection Using K-Nearest Neighbour Algorithm (KNN) and Evaluation on Standard ECG Databases*, in Journal of Advanced Research, vol. 4, issue 4, 2013, pp 331-344; [Online], Available: https://www.sciencedirect.com/science/article/pii/S209012321200046X

[27] M. Rashed-Al-Mahfuz, M.A. Moni, P. Lio, S.M.S. Islam, S. Berkovsky, M. Khushi, J.M.W. Quinn, *Deep Convolutional Neural Networks Based ECG Beats Classification to Diagnose Cardiovascular Conditions*, in Biomedical Engineering Letters, vol 11, 2021, pp 147-162; [Online], Available: https://link.springer.com/article/10.1007/s13534-021-00185-w

[28] Z. Dokur, T. Olmez, *Heartbeat Classification by using a Convolutional Neural Network Trained with Walsh Functions*, in Neural Computing and Applications, vol. 32, 2020, pp 12515-12534; [Online]. Available: https://link.springer.com/article/10.1007%2Fs00521-020-04709-w

[29] O.S. Lih, et al., *Comprehensive Electrocardiographic Diagnosis Based on Deep Learning*, in Artificial Intelligence in Medicine, vol. 103, 2020; [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0933365719309030

[30] H. Li, et al., *Arrhythmia Classification Algorithm Based on Multi-Feature and Multi-Type Optimised SVM*, in the American Scientific Research Journal for Engineering, Technology and Sciences (ASRJETS), vol. 63, No 1, 2020, pp 72-86; [Online]. Available: https://asrjetsjournal.org/index.php/American_Scientific_Journal/article/view/5509/2046

[31] Y. Zhang, S. Wei, L. Zhang, C. Liu, *Comparing the Performance of Random Forest, SVM and Their Variants for ECG Quality Assessment Combined with Nonlinear Features*, in Journal of Medical and Biological Engineering, vol. 39, 2019, pp 381-392. [Online], Available: https://link.springer.com/article/10.1007/s40846-018-0411-0

[32] F. Bouaziz, D. Boutana, H. Oulhadj, *Diagnostic of ECG Arrhythmia using Wavelet Analysis and K-Nearest Neighbour Algorithm*, IEEE, 2018; [Online]. Available: https://ieeexplore.ieee.org/document/8652020

[33] O. Castillo, P. Melin, E. Ramirez, J. Soria, *Hybrid Intelligent System for Cardiac Arrhythmia Classification with Fuzzy K-Nearest Neighbors and Neural Networks Combined with a Fuzzy System*, in Expert Systems with Applications, vol. 39, issue 3, 2012, pp 2947-2955. [Online], Available: https://www.sciencedirect.com/science/article/pii/S0957417411012875

[34] N. Emanet, *ECG Beat Classification by Using Discrete Wavelet Transform and Random Forest Algorithm*, IEEE, 2009, [Online]. DOI: 10.1109/ICSCCW.2009.5379457

[35] R. Mahajan, R. Kamaleswaran, J.A. Howe, O. Akbilgic, *Cardiac Rhythm Classification from a Short Single Lead ECG Recording via Random Forest*, in 2017 Computing in Cardiology, 2017, [Online], DOI: 10.22489/CinC.2017.179-403

[36] M. Kropf, D. Hayn, G. Schreier, *ECG Classification Based on Time and Frequency Domain Features using Random Forests*, in 2017 Computing in Cardiology, 2017, [Online], DOI: 10.22489/CinC.2017.168-168

[37] MathWorks, 2012, *Create Simple Deep Learning Network for Classification*, Accessed 25 May 2021, [Online]. Available: https://au.mathworks.com/help/deeplearning/ug/create-simple-deep-learning-network-for-classification.html

Appendix A. One-Page Review

(Copied exactly as-is, including the Reference List.)

Can we Teach a Machine to be a Cardiologist?

Medical equipment, such as electrocardiograms (ECG), play a pivotal role in the diagnosis of a patient. In particular they make it possible for medical professionals to determine heart abnormalities and administer the correct treatment [1]. Heart disease continues to be a leading cause of death [2], so identifying and treating these diseases early is critical.

ECGs measure the electrical activity of the heart, which is then plotted as a waveform. Any irregularity in the plotted waveform can be indicative of an abnormality [3], so they are a useful tool for medical professionals in assessing patient health. An example of an ECG signal, including relevant points and intervals and their definitions can be found in Figure 1 at the end of this document.

Classifying ECGs is a challenging process for a number of reasons. Namely, normal ECGs may differ between individuals, one disease may have dissimilar signs on different patients, and two distinct diseases may have a similar effect on a normal ECG [3].

Recently, there has been an interest in employing machine learning (ML) in the medical field [1] [2] [4], such as by analysing the features of ECG to detect abnormalities. ML techniques could make it possible to diagnose patients more precisely than when done manually [3].

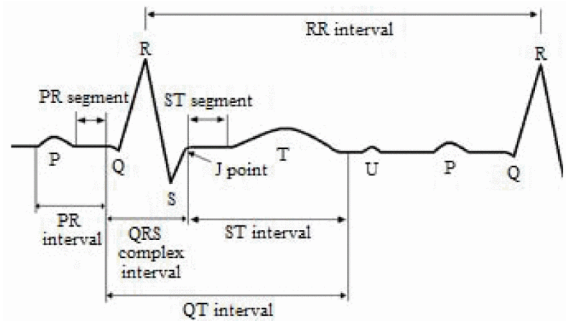
Prior to analysing the ECG for features, it is important to complete some pre-processing on the signal. This is done to remove baseline wander, motion artifacts, and other interruptions present in the collected results [6]. These noise removal techniques can be done with simple filtering techniques such as low pass filtering and Butterworth filters [7], or they can be based on adaptive filtering methods such as wavelet transforms, discrete Fourier transform (DFT) and the Pan-Tompkins algorithm [3] [6]. Adaptive filtering can produce much better results than the simple filtering methods.

Now it is possible to extract the relevant features of the ECG. In the time domain this may involve identifying the P and T waves, as well as the QRS complex over many cycles. It may also involve measuring the time between R peaks for consistency. In the frequency domain, a number of other features are worth identifying, including the very low frequency, low frequency and high frequency components of the signal [6].

From the features extracted, the signal can be classified as normal or abnormal. It may also be possible to determine the type of abnormality present, and further group the signal according to this. Various machine learning techniques have been found to be effective for this purpose. These include artificial neural networks (ANNs), the K-Nearest-Neighbour (KNN) Rule, Support Vector Machine (SVM) and decision tree classifiers [1] [3] [4] [6].

The SVM is a classification algorithm which involves using a hyperplane to provide an optimal decision boundary to separate classes [1], in this case normal and abnormal ECG signals. It can efficiently learn nonlinear functions and has been used previously in various pattern classification and regression applications [7].

One abnormality which can be detected from an ECG are heart murmurs. Heart murmurs are a "whooshing, humming or rasping" sound between heartbeats, and are caused by turbulent blood flow through the heart [8] [9]. Many are innocent, meaning they do not correspond to an underlying problem, but heart murmurs can also be linked to a range of disorders including congenital heart disorders, cardiac tissue damage and emotional stress [8]. They are often asymptomatic and are only picked up during routine health checks. Hence it is important that these are detected early, and treatment administered.



Feature	Description	Duration
RR	interval between R wave and the next R wave	0.6-1.2 s
P	first short upward movement of the ECG	80ms
PR	measured from the beginning of the P wave to the beginning of the QRS complex	120-200 ms
QRS	normally begins with a downward deflection Q, a larger upwards deflection R and ends with a downward S wave	80-120 ms
PR	connects the P wave and the QRS complex	50-120 ms
J-point	The point at which the QRS complex finishes and the ST segment begins is called J-point.	Not applicable
ST	connects the QRS complex and the T wave	80-120 ms
T	normally a modest upward waveform	160 ms
ST	measured from the J point to the end of the T wave	320 ms
QT	measured from the beginning of the QRS complex to the end of the T wave	420 ms
U	normally has low amplitude and often it is completely absent	Not mentioned

Figure 1: ECG signal, points of interest and their descriptions

References

[1] A. Ahamed, et al.; *ECG Heartbeat Classification Using Ensemble of Efficient Machine Learning Approaches on Imbalanced Datasets*; IEEE, 2020; Accessed 20 March 2021; [Online] DOI: 10.1109/ICAICT51780.2020.9333534

[2] K. Chen, A. Mudvari, F. G. G. Barrera, L. Cheng, and T. Ning; *Heart Murmurs Clustering Using Machine Learning*; IEEE, 2018; Accessed 25 March 2021; [Online] DOI: 10.1109/ICSP.2018.8652329

[3] S. H. Jambukia, V. K. Dabhi, H. B. Prajapati; *Classification of ECG signals using machine learning techniques: A survey*; IEEE, 2015; Accessed: 16 March 2021; [Online] DOI: 10.1109/ICACEA.2015.7164783

[4] A. D. Levin, A. Ragazzi, S. L. Szot, T. Ning; *A Machine Learning Approach to Heart Murmur Detection and Classification*; IEEE, 2020; Accessed 25 March 2021; [Online] DOI: 10.1109/CISP-BMEI51763.2020.9263528

[6] C. Venkatesan, et al.; *ECG Signal Preprocessing and SVM Classifier-Based Abnormality Detection in Remote Healthcare Applications*; IEEE, 2018; Accessed 20 March 2021; [Online] DOI: 10.1109/ACCESS.2018.2794346

[7] S. Celin, K. Vasanth; *ECG Signal Classification Using Various Machine Learning Techniques*; 2018; Accessed: 16 March 2021; [Online] Available: https://doi.org/10.1007/s10916-018-1083-6

[8] BetterHealth; *Heart Murmur*; Accessed 23 March 2021; [Online] Available: https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/heart-murmur

[9] L. Sherwood; *Human Physiology: From Cells to Systems*, 9th ed. Cengage Learning, 2016; ISBN: 879-1-285-86693-2

Appendix B. MATLAB Code

B1 Modified MathWorks Example

The original code can be accessed through this page: https://au.mathworks.com/help/wavelet/ug/ecg-classification-using-wavelet-features.html [23]. The appendix here shows a version of the code used to classify our data (note that since class names are hard-coded into this script, a few different versions of this code were actually used, depending on the number of classes being examined).

```
%% Signal Classification Using Wavelet-Based Features and Support Vector
Machines
% Copyright 2018 The MathWorks, Inc.
% Modified by: S.L. Kleinig (a1740773), May 2021

%% Create Training and Test Data
% Randomly split the data into two sets - training and test data sets.
percent_train = 70;
[trainData,testData,trainLabels,testLabels] = ...
    helperRandomSplit(percent_train,ourECGData);
%% Data Set Percentages
% Examine the percentage of each class in the training and test sets. The
% percentages in each are consistent with the overall class percentages in
% the data set.
Ctrain = countcats(categorical(trainLabels))./numel(trainLabels).*100
Ctest = countcats(categorical(testLabels))./numel(testLabels).*100
%% Plot Samples
% Plot the first few thousand samples of four randomly selected records
% from |ECGData|.
figure
helperPlotRandomRecords(ourECGData,1)
%% Feature Extraction
% Extract the features used in the signal classification for each signal.

timeWindow = floor(length(ourECGData.Data(1,:))/8);
ARorder = 4;
MODWPTlevel = 4;
[trainFeatures,testFeatures,featureindices] = ...

helperExtractFeatures(trainData,testData,timeWindow,ARorder,MODWPTlevel);
%% Examining Some Features
% As an example, examine the range of Holder exponents in the singularity
% spectra for the first time window. Plot the data for the entire data set.
allFeatures = [trainFeatures;testFeatures];
allLabels = [trainLabels;testLabels];
figure
boxplot(allFeatures(:,featureindices.HRfeatures(1)),allLabels,'notch','on')
ylabel('Holder Exponent Range')
title('Range of Singularity Spectrum by Group (First Time Window)')
grid on

% You can perform a one-way analysis of variance on this feature and
% confirm what appears in the boxplot.
[p,anovatab,st] = anova1(allFeatures(:,featureindices.HRfeatures(1)),...
```

```
    allLabels);
c = multcompare(st,'display','off')

% As an additional example, consider the difference in variance in the
% second-lowest frequency (second-largest scale) wavelet subband for the
% three groups.
boxplot(allFeatures(:,featureindices.WVARfeatures(end-
1)),allLabels,'notch','on')
ylabel('Wavelet Variance')
title('Wavelet Variance by Group')
grid on

%% Signal Classification
features = [trainFeatures; testFeatures];
rng(1)
template = templateSVM(...
    'KernelFunction','polynomial',...
    'PolynomialOrder',2,...
    'KernelScale','auto',...
    'BoxConstraint',1,...
    'Standardize',true);
model = fitcecoc(...
    features,...
    [trainLabels;testLabels],...
    'Learners',template,...
    'Coding','onevsone',...
    'ClassNames',{'N','O','A','~'});
kfoldmodel = crossval(model,'KFold',5);
classLabels = kfoldPredict(kfoldmodel);
loss = kfoldLoss(kfoldmodel)*100
[confmatCV,grouporder] =
confusionmat([trainLabels;testLabels],classLabels);

%% Precision, Recall, and F1 Score
CVTable = helperPrecisionRecall(confmatCV);
disp(CVTable)

%% Predict Test Data
model = fitcecoc(...
     trainFeatures,...
     trainLabels,...
     'Learners',template,...
     'Coding','onevsone',...
     'ClassNames',{'N','O','A','~'});
predLabels = predict(model,testFeatures);

% Use the following to determine the number of correct predictions and
% obtain the confusion matrix.
correctPredictions = strcmp(predLabels,testLabels);
testAccuracy = sum(correctPredictions)/length(testLabels)*100
[confmatTest,grouporder] = confusionmat(testLabels,predLabels);

% Obtain precision, recall, and the F1 scores for the test set.
testTable = helperPrecisionRecall(confmatTest);
disp(testTable)


%% Supporting Functions
% *helperPlotRandomRecords* Plots four ECG signals randomly chosen from
```

```
% |ECGData|.
function helperPlotRandomRecords(ECGData,randomSeed)

if nargin==2
    rng(randomSeed)
end

M = size(ECGData.Data,1);
idxsel = randperm(M,4);
for numplot = 1:4
    subplot(2,2,numplot)
    plot_points = min(3000,length(ECGData.Data(1,:)));
    plot(ECGData.Data(idxsel(numplot),1:plot_points))
    ylabel('Volts')
    if numplot > 2
        xlabel('Samples')
    end
    title(ECGData.Labels{idxsel(numplot)})
end

end
%%
% *helperExtractFeatures* Extracts the wavelet features and AR coefficients
% for blocks of the data of a specified size. The features are concatenated
% into feature vectors.
function [trainFeatures, testFeatures,featureindices] =
helperExtractFeatures(trainData,testData,T,AR_order,level)
% This function is only in support of XpwWaveletMLExample. It may change or
% be removed in a future release.
trainFeatures = [];
testFeatures = [];

for idx =1:size(trainData,1)
    x = trainData(idx,:);
    x = detrend(x,0);
    arcoefs = blockAR(x,AR_order,T);
    se = shannonEntropy(x,T,level);
    [cp,rh] = leaders(x,T);
    wvar = modwtvar(modwt(x,'db2'),'db2');
    trainFeatures = [trainFeatures; arcoefs se cp rh wvar']; %#ok<AGROW>

end

for idx =1:size(testData,1)
    x1 = testData(idx,:);
    x1 = detrend(x1,0);
    arcoefs = blockAR(x1,AR_order,T);
    se = shannonEntropy(x1,T,level);
    [cp,rh] = leaders(x1,T);
    wvar = modwtvar(modwt(x1,'db2'),'db2');
    testFeatures = [testFeatures;arcoefs se cp rh wvar']; %#ok<AGROW>

end

featureindices = struct();
% 4*8
featureindices.ARfeatures = 1:32;
startidx = 33;
endidx = 33+(16*8)-1;
```

```
featureindices.SEfeatures = startidx:endidx;
startidx = endidx+1;
endidx = startidx+7;
featureindices.CP2features = startidx:endidx;
startidx = endidx+1;
endidx = startidx+7;
featureindices.HRfeatures = startidx:endidx;
startidx = endidx+1;
endidx = startidx+13;
featureindices.WVARfeatures = startidx:endidx;
end


function se = shannonEntropy(x,numbuffer,level)
numwindows = numel(x)/numbuffer;
y = buffer(x,numbuffer);
se = zeros(2^level,size(y,2));
for kk = 1:size(y,2)
    wpt = modwpt(y(:,kk),level);
    % Sum across time
    E = sum(wpt.^2,2);
    Pij = wpt.^2./E;
    % The following is eps(1)
    se(:,kk) = -sum(Pij.*log(Pij+eps),2);
end
se = reshape(se,2^level*numwindows,1);
se = se';
end


function arcfs = blockAR(x,order,numbuffer)
numwindows = numel(x)/numbuffer;
y = buffer(x,numbuffer);
arcfs = zeros(order,size(y,2));
for kk = 1:size(y,2)
    artmp =  arburg(y(:,kk),order);
    arcfs(:,kk) = artmp(2:end);
end
arcfs = reshape(arcfs,order*numwindows,1);
arcfs = arcfs';
end


function [cp,rh] = leaders(x,numbuffer)
y = buffer(x,numbuffer);
cp = zeros(1,size(y,2));
rh = zeros(1,size(y,2));
for kk = 1:size(y,2)
    [~,h,cptmp] = dwtleader(y(:,kk));
    cp(kk) = cptmp(2);
    rh(kk) = range(h);
end
end
%%
% *helperPrecisionRecall*  returns the precision, recall, and F1 scores
% based on the confusion matrix. Outputs the results as a MATLAB table.
function PRTable = helperPrecisionRecall(confmat)

precisionA = confmat(1,1)/sum(confmat(:,1))*100;
```

```
precisionO = confmat(2,2)/sum(confmat(:,2))*100;
precisionN = confmat(3,3)/sum(confmat(:,3))*100;
precisionX = confmat(4,4)/sum(confmat(:,4))*100;
recallA = confmat(1,1)/sum(confmat(1,:))*100;
recallO = confmat(2,2)/sum(confmat(2,:))*100;
recallN = confmat(3,3)/sum(confmat(3,:))*100;
recallX = confmat(4,4)/sum(confmat(4,:))*100;
F1A = 2*precisionA*recallA/(precisionA+recallA);
F1O = 2*precisionO*recallO/(precisionO+recallO);
F1N = 2*precisionN*recallN/(precisionN+recallN);
F1X = 2*precisionX*recallX/(precisionX+recallX);
% Construct a MATLAB Table to display the results.
PRTable = array2table([precisionA recallA F1A;...
    precisionO recallO F1O; precisionN recallN F1N; precisionX recallX
F1X],...
    'VariableNames',{'Precision','Recall','F1_Score'},'RowNames',...
    {'A','O','N','~'});

end

%%
% Randomly split the data so there are equal proportions of each class in
% the training and test sets.
function [trainData, testData, trainLabels, testLabels] =
helperRandomSplit(percent_train_split,ECGData)

    Labels = ECGData.Labels;
    Data = ECGData.Data;
    percent_train_split = percent_train_split/100;
    idxAbegin = find(strcmpi(Labels,'A'),1,'first');
    idxAend = find(strcmpi(Labels,'A'),1,'last');
    Na = idxAend-idxAbegin+1;
    idxObegin = find(strcmpi(Labels,'O'),1,'first');
    idxOend = find(strcmpi(Labels,'O'),1,'last');
    No = idxOend-idxObegin+1;
    idxNbegin = find(strcmpi(Labels,'N'),1,'first');
    idxNend = find(strcmpi(Labels,'N'),1,'last');
    Nn = idxNend-idxNbegin+1;
    idxXbegin = find(strcmpi(Labels,'~'),1,'first');
    idxXend = find(strcmpi(Labels,'~'),1,'last');
    Nx = idxXend-idxXbegin+1;

    % Obtain number needed for percentage split
    num_train_a = round(percent_train_split*Na);
    num_train_o = round(percent_train_split*No);
    num_train_n = round(percent_train_split*Nn);
    num_train_x = round(percent_train_split*Nx);
    rng default;
    Pa = randperm(Na,num_train_a);
    Po = randperm(No,num_train_o);
    Pn = randperm(Nn,num_train_n);
    Px = randperm(Nx,num_train_x);

    notPa = setdiff(1:Na,Pa);
    notPo = setdiff(1:No,Po);
    notPn = setdiff(1:Nn,Pn);
    notPx = setdiff(1:Nx,Px);
    Adata = Data(idxAbegin:idxAend,:);
    ALabels = Labels(idxAbegin:idxAend);
    Odata = Data(idxObegin:idxOend,:);
```

```
    OLabels = Labels(idxObegin:idxOend);
    Ndata = Data(idxNbegin:idxNend,:);
    NLabels = Labels(idxNbegin:idxNend);
    Xdata = Data(idxXbegin:idxXend,:);
    XLabels = Labels(idxXbegin:idxXend);

    trainA = Adata(Pa,:);
    trainALabels = ALabels(Pa);
    testA = Adata(notPa,:);
    testALabels = ALabels(notPa);
    trainO = Odata(Po,:);
    trainOLabels = OLabels(Po);
    testO = Odata(notPo,:);
    testOLabels = OLabels(notPo);
    trainN = Ndata(Pn,:);
    trainNLabels = NLabels(Pn);
    testN = Ndata(notPn,:);
    testNLabels = NLabels(notPn);
    trainX = Xdata(Px,:);
    trainXLabels = XLabels(Px);
    testX = Xdata(notPx,:);
    testXLabels = XLabels(notPx);
    trainData = [trainA ; trainO; trainN; trainX];
    trainLabels = [trainALabels ; trainOLabels; trainNLabels;
trainXLabels];
    testData = [testA ; testO; testN; testX];
    testLabels = [testALabels; testOLabels; testNLabels; testXLabels];

end

% Copyright 2018 The MathWorks, Inc.
```

B2 Data Collating Code

This code was used to convert the signals from a collection of MATLAB files to a single variable. This enabled these signals to be passed to the Modified MathWorks Classifier [23] (see Appendix B1). Again, a few versions of this code were used to collate the data in different ways.

```
% Collating Code
% S.L. Kleinig (a1740773)
% May 2021
% Combines all ECG signals into a single variable to pass to the Classifier

clearvars;

% Needed variables
f_s = 300;
t_s = 1/f_s;
time = 10; %seconds
req_samples = f_s*time;

% Location of the data signals
```

```
data_dir =
'C:\Users\slkle\OneDrive\Documents\AdelaideUNI\Honours\Physionet_Records\tr
aining\training2017\';

% Load the list of records in the validation set.
RECLIST = readtable([data_dir 'REFERENCE.csv']);
RECORDS = table2array(RECLIST(:,1));
labels = table2array(RECLIST(:,2));

% Load each ECG recording in turn, and add it and its label to the data
% strucure
for i = 1:1:length(RECORDS)
    fname = RECORDS{i};
    tic;
    file = [data_dir fname];
    load(file);

    data = val;

    % Do not include recordings which are too short, otherwise collect the
    % middle section of the recording
    if length(val) >= req_samples
        mid = round(length(data)/2);
        sidx = mid - req_samples/2;
        eidx = mid + req_samples/2;
        generated_data(n,1:req_samples) = data(1,sidx+1:eidx);
        kept_labels(n,1) = labels(i);
        n = n + 1;
    end

    clear data;
end

fprintf('Data generated...\n');

ourECGData.Data = sortECGData(generated_data,labels);

clearvars -except ourECGData;

fprintf('Done.\n');

% Function to sort the data by label
function [ourECGData] = sortECGData(generated_data,labels)
arrLabels = cell2mat(labels);
Alocs = find(arrLabels == 'A');
Olocs = find(arrLabels == 'O');
Nlocs = find(arrLabels == 'N');
Xlocs = find(arrLabels == '~');
ourECGData.Data = generated_data(Alocs,:);
ourECGData.Labels = labels(Alocs);
ourECGData.Data = [ourECGData.Data;generated_data(Olocs,:)];
ourECGData.Labels = [ourECGData.Labels;labels(Olocs)];
ourECGData.Data = [ourECGData.Data;generated_data(Nlocs,:)];
ourECGData.Labels = [ourECGData.Labels;labels(Nlocs)];
ourECGData.Data = [ourECGData.Data;generated_data(Xlocs,:)];
ourECGData.Labels = [ourECGData.Labels;labels(Xlocs)];
```

```
fprintf("Summary:\n   A = %d records\n   O = %d records\n   N = %d
records\n   X = %d records\n",sum(arrLabels == 'A'),sum(arrLabels ==
'O'),sum(arrLabels == 'N'),sum(arrLabels == '~'));
end
```

B3 Modified Pan-Tompkins Code

This code was developed based on the method proposed in [20] but has not been copied from any source.

```
% Modified Pan-Tompkins
% Sonia Kleinig (a1740773)
% Using the following as a guide:
https://ieeexplore.ieee.org/abstract/document/9182298

f_s = 300;

% LOAD A SIGNAL
% data_dir = [pwd filesep 'validation' filesep];
data_dir =
'C:\Users\slkle\OneDrive\Documents\AdelaideUNI\Honours\Physionet_Records\sa
mple2017/validation/';
% Load the list of records in the validation set.
fid = fopen([data_dir 'RECORDS'],'r');
if(fid ~= -1)
    RECLIST = textscan(fid,'%s');
else
    error(['Could not open ' data_dir 'RECORDS for scoring. Exiting...'])
end
fclose(fid);
RECORDS = RECLIST{1};
j = input("Select record: ",'s');
j = str2num(j);
fname = RECORDS{j};
tic;
file = [data_dir fname];
load(file);
heading = ['loaded ' fname '\n'];
fprintf(heading);

val = val(2000:5000);

% 1. REMOVE DC COMPONENTS
ecg_dc_rem = val - mean(val);
ecg_normalised = ecg_dc_rem/max(abs(ecg_dc_rem));

% 2. BAND PASS FILTER
H1_N = [1,0,0,0,0,0,-2,0,0,0,0,0,1];
% H1_D = (1 - z^(-1))^2;
H1_D = [1,-2,1];
% H2_N = z^(-16) - z^(-17) - 1 + z^(-32);
H2_N = [-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,-
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1];
% H2_D = 1 - z^(-1);
H2_D = [1,-1];
ecg_filtered = filter(H1_N,H1_D,ecg_normalised);
```

```
% 3. DERIVATIVE FILTER
ecg_diff = gradient(ecg_filtered);
grad_integ = integrate(ecg_diff);

% 4. SQUARING FUNCTION
ecg_squared = ecg_diff.^2;

% 5. AVERAGING FUNCTION
windowSize = 15;
b = (1/windowSize)*ones(1,windowSize);
a = 1;
ecg_averaged = filter(b,a,ecg_squared);

% 6. INTEGRATOR
ecg_integ = integrate(ecg_averaged);

% 7. RESULT
% Plot Figures
figure(1)
plot(ecg_dc_rem/max(abs(ecg_dc_rem)));
hold on
plot(ecg_filtered/max(abs(ecg_filtered)));
plot(ecg_diff/max(abs(ecg_diff)));
plot(ecg_squared/max(abs(ecg_squared)));
plot(ecg_averaged/max(abs(ecg_averaged)));
plot(ecg_integ/max(abs(ecg_integ)));
hold off
legend("DC removed","Filtered","Derivative
Filtered","Squared","Averaged","Integrated");
figure(2)
plot([1:1:3001]/f_s,ecg_dc_rem/max(ecg_dc_rem));
hold on
plot([1:1:3001]/f_s,ecg_integ/max(ecg_integ));
hold off
legend("Original ECG","Integraged ECG");
title(heading2);xlabel("Time (seconds)");ylabel("Normalised Amplitude");
figure(3)
subplot(6,1,1)
plot([1:1:3001]/f_s,ecg_normalised);title("(a) ECG Signal after DC Removed
and Normalisation");
xlabel("Time (seconds)");ylabel("Magnitude");
subplot(6,1,2)
plot([1:1:3001]/f_s,ecg_filtered);title("(b) ECG after Filtering");
xlabel("Time (seconds)");ylabel("Magnitude");
subplot(6,1,3)
plot([1:1:3001]/f_s,ecg_diff);title("(c) ECG after Derivative Filtering");
xlabel("Time (seconds)");ylabel("Magnitude");
subplot(6,1,4)
plot([1:1:3001]/f_s,ecg_squared);title("(d) ECG after Squaring");
xlabel("Time (seconds)");ylabel("Magnitude");
subplot(6,1,5)
plot([1:1:3001]/f_s,ecg_averaged);title("(e) ECG after Averaging");
xlabel("Time (seconds)");ylabel("Magnitude");
subplot(6,1,6)
plot([1:1:3001]/f_s,ecg_integ);title("(f) ECG after Integration");
xlabel("Time (seconds)");ylabel("Magnitude");
sgtitle(heading2);
```