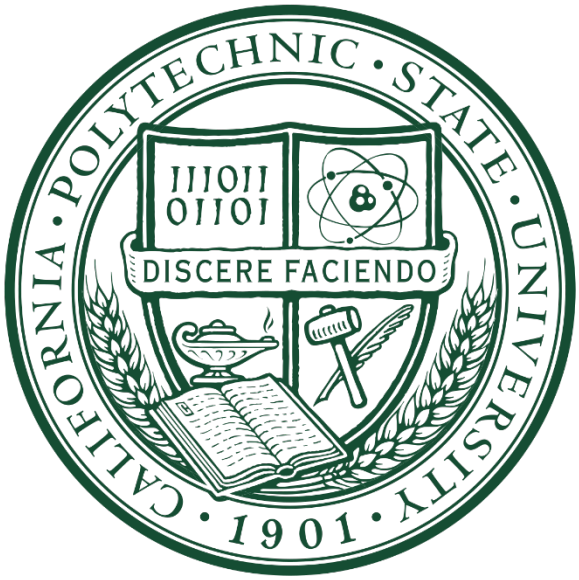
ECG Arrhythmia Classification Using Discrete Wavelet Transformation

by
Hannah Chookaszian
Nathan Diekema



Senior Project – End of Quarter Report

ELECTRICAL ENGINEERING DEPARTMENT
California Polytechnic State University
San Luis Obispo

2021

Table of Contents

List of Figures

List of Tables

Abstract

Cardiovascular diseases (CVDs) are the highest leading cause of death worldwide with an approximate 17.9 million related deaths every year according to the World Health Organization (WHO). Electrocardiographic (EKG or ECG) signals are electrical signals measured in the heart and are the main indicator for pre-existing cardiac conditions. The application of deep learning methods such as artificial neural networks (ANN) will assist in the automated detection and classification of ECG signals. The current methods for ECG analysis are lacking in accuracy and reliability considering the level of risk involved with CVDs and the importance of a correct diagnosis. Clinicians use ECG data to find irregular patterns that may indicate the existence of potentially life-threatening cardiac conditions. The proposed method uses the discrete wavelet transform for feature extraction and a feed-forward neural network for the classification of six types of heart beats. This approach achieves impressive results in terms of accuracy, sensitivity, and specificity across all classes. The best results obtained were 97.87%, 99.57%, and 97.87%, respectively. Although simpler than other state-of-the-art methods, this approach achieves competitive results and can help reduce the chance of misdiagnosis and missed diagnosis.

Chapter 1: Introduction

1.1 - Overview

Over the past century, electrocardiogram (ECG) has become the principal tool in the diagnosis of cardiovascular conditions as it is a quick and noninvasive method. This technology has been revolutionary for the diagnosis of cardiovascular conditions which remain the leading cause of deaths worldwide. According to the World Health Organization, cardiovascular diseases are responsible for approximately 31% of annual deaths worldwide [1]. For reference, it has been reported that “about 80% of sudden cardiac deaths are the result of ventricular arrhythmias or irregular heartbeats [2]. While ECG machines are great for capturing a patient's ECG signal, it requires an experienced cardiologist to accurately analyze the morphological patterns and distinguish between healthy signals and varying arrhythmias or conditions [3]. It is for this reason that a fast and reliable computer-oriented approach would substantially reduce the pressure on clinicians and the inevitable margin of error from human-oriented analysis. Current research in the area suggests that current state-of-the art methods employing neural networks can achieve accuracies of over 99% for certain arrhythmias [2]. A computer with the correct software has the ability to identify abnormalities that would normally be overlooked by a human, and at a faster rate. Additionally, using a computer for medical related analysis mitigates the potential impact of internal human biases based on race, ethnicity, or age. In addition, a lightweight software-based ECG analysis system will make this technology more accessible and affordable for hospitals and will open the door for implementation into at-home health monitoring devices.

The basic structure of the software will begin with a signal processing block that is responsible for denoising, amplitude normalization, and heartbeat segmentation. Next, features are extracted from the processed signal using the discrete wavelet transform (DWT) and notable morphological characteristics. Finally, the feature vector is passed through a feed-forward neural network (FNN) which will detect irregularities in the feature set and ultimately determining the best diagnosis based on the information provided. Understanding how to analyze ECG signals will be crucial for correct diagnoses and will be done by recognizing and breaking down the relevant features [5]. This process is referred to as feature extractions and will take important measurements of the ECG signal related to the QRS complex, P-wave, and QT duration, to name a few [6]. Feature extraction is an essential stage of the process as it allows for more accurate recognition of abnormalities that could be indicative of a disease [4]. Once trained, the FNN can detect minute abnormalities from the extracted features that may be overlooked by a human.

However, before passing the signal through the feature extraction stage the ECG signal must be processed. Our senior project advisor, Helen Yu, is well versed on ECG analysis with neural networks and signal processing. According to Yu, an adaptive filter approach is important to reduce noise in the ECG signals [7]. Denoising the signals will be crucial to accurate diagnosis on this project. It is important to address all the possible sources of noise and mitigate them without distorting the original signal or eliminating important features [8]. Filtering will occur during the signal processing stage of the system. This stage will essentially remove noise from the signal that may be induced by muscle contractions, power line interference, baseline wander, or instrumentation noise [9].

The neural network will be important to help remove the human aspect of ECG analysis. An FNN is used for this approach because it is simple, yet robust allowing for respectable performance without requiring an excessive amount of resources or time to train [4]. When utilizing neural networks, it is important to consider factors related to complexity and efficiency. These must be considered as they are directly correlated with response time of the system but are arguably inversely related to accuracy. This design must find a balance between the two that allows clinicians and medical professionals to get results in a timely manner while simultaneously ensuring said results are reliable and accurate. Moreover, the training of complex neural networks takes a substantial amount of computing power which could inflate the costs related to this project.

In terms of performance, the completed ECG classification system is aiming for a diagnosis accuracy of 97%. In 2017, an evolutionary-neural system achieved 90.20% recognition sensitivity for 17 different cardiac conditions [10]. This project will use a different type of neural network and will not be diagnosing as many conditions as the evolutionary neural network project which will make this accuracy goal feasible. There are also some patented technologies that must be addressed in the planning of this project. A project by Chen, Yang, and Jiang uses the frequency domain to analyze signals and displays markers for analysis [11]. This approach is just one of many and the end goal is to compare the performance of this model to that of existing methods.

It is worth noting that this technology should not be interpreted or seen as an outright replacement for human analysis. Instead, it is intended to serve as a complementary tool for trained physicians. It is important to understand that computers are still prone to making mistakes, especially for more discrete cardiac conditions. For instance, it has been proven that accurate computerized interpretations of myocardial infarction are incredibly difficult to achieve [12]. For this reason, medical professionals are still expected to be the basis of decision making by checking the ECG signal and ensuring the computerized diagnosis matches. Overall, the utilization of artificial neural networks for the analysis of

electrocardiogram signals could revolutionize this process. The automated assistance from our deep learning software will increase overall detection accuracy and will ultimately decrease fatality rates related to cardiac conditions. When compared to current methods, the implementation and utilization of a trained ANN will help reduce the chance of a misdiagnosis and missed diagnosis and will ultimately save lives.

1.2 - Our Approach

There are three major stages in any heartbeat classification system: preprocessing, feature extraction, and classification [3]. The preprocessing stage is arguably the most important of the three as each subsequent stage relies on the quality and accuracy of the processed signal. As a result, before applying any analytical techniques on the signal it is important to normalize and filter out any unwanted noise from the signal. A typical ECG signal may contain many types of noise, the most significant of which are induced by baseline wander, powerline interference, muscular contractions (electromyography), and instrumentation noise [9]. These interferences are known as signal artifacts and can seriously hinder the accuracy of a diagnosis. For this project, the preprocessing stage will first remove any unwanted powerline interference and electromyographic noise by passing the signal through a digital low pass filter at 30 Hz. A high pass filter will then be applied at 0.5 Hz to reduce baseline wander.

After the initial filters have been applied, the signal is segmented into individual beats. Next, the feature extraction of each beat takes place. Features refer to any notable morphological characteristics of a heart beat. These may include, but are not limited to the slopes, peaks, amplitudes, frequency, or anything that helps describe the shape of the ECG waveform. Although every heartbeat is slightly different, they all share the same morphological pattern. Every heartbeat is composed of a series of deflections away from the baseline of the ECG. A single beat of a sinus cycle may be divided into several different sections, the most significant of which, are namely the P-wave, QRS-complex, and T-wave [13]. A few popular features are illustrated in *Figure 1*.

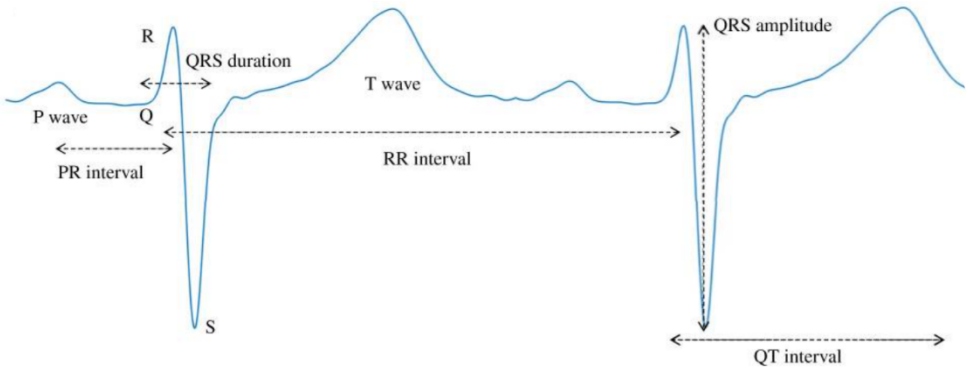


Figure 1: Sinus Cycle ECG Waveform with labeled features [13]

For this approach, features will be extracted using a wavelet transform. Feature quality and robustness has proven to be an obstacle in previous research. Poor feature extraction and filtering has proven to lead to a generally low performance system despite having powerful classification algorithms [14]. The wavelet transform, however, has been demonstrated as an effective tool for isolating relevant properties of an ECG waveform from lingering noise and other potentially impairing artifacts [2,3]. Previous studies have shown that using a down sampled wavelet transform as their feature set have demonstrated higher classification accuracies when compared to using the original waveform. A wavelet transformation is similar to a Fast-Fourier Transform (FFT) but provides a feature set describing a waveform in terms of both time and frequency. This allows for the representation of additional special features of a wavelet at multiple resolutions [15].

After a feature set is extracted from the signal it will be passed through a neural network classifier. The neural network will be trained and tested using feature sets as described above. Data will be used from the Massachusetts Institute of Technology-Beth Israel Hospital (MIT-BIH) ECG database, a publicly available database created in the late 1900s as a reference standard for ECG classification systems [16]. The classification stage is responsible for using a FNN with a common classification function to analyze and determine the irregularities found in the signal, or lack thereof. This step will solve multiple classification problems via logistic regression which will help determine the final diagnosis. In this approach, the classification stage will use a method called softmax. A softmax activation function, otherwise known as a normalized exponential function, is a non-linear logistic function generalized to multiple dimensions [17]. It is popularly used as the last activation function of a neural network due to its ability to reliably normalize the output of a network to a probability distribution over the given output classes. This makes the output of the classification stage extremely readable and easy to work with. It also allows the computer to easily determine the percentage confidence of the final diagnosis which is a very important factor when dealing with potentially life-threatening diseases.

1.3 – Report Overview

This paper aims to inform the reader about a unique approach to the automatic classification of cardiac arrhythmias through the analysis of ECG signals. First, Chapter 2 provides a detailed discussion on existing methods for the analysis and classification of ECG signals. Each method is examined in terms of advantages and limitations in comparison with the approach discussed in this paper. Chapter 3 proceeds to describe important background information pertaining to Electrocardiography, the anatomy of the human heart, cardiac arrhythmias, noise artifacts, wavelet theory, and feed-forward neural networks. Chapter 4 then goes on to explain our approach in detail. Next, Chapter 5 provides a comprehensive summary and examination of the results obtained from our model. Finally, Chapter 6 offers a concluding discussion on the chosen approach and final results and considers future improvements to the method used.

Chapter 2: Literature Review

2.1 - Support Vector Machines (SVMs) Approach

One of the more popular approaches to ECG classification employs the use of Support Vector Machines (SVMs). SVMs are a commonly used category of machine learning algorithms due to their propensity for consistent and reliable classification. They are known to produce high accuracy while requiring less computational power than other methods. The overall objective of an SVM algorithm is to find the hyperplane in an N-dimensional space where N represents the number of features [18]. Hyperplanes are essentially decision boundaries that assist in the classification of the N-dimensional data points given in a set. Data points that fall on either side of the hyperplane will be assigned to different classes. The goal of an SVM is to increase the confidence of the classification by maximizing the margin distance. In other words, the best classification occurs when the hyperplane provides the maximum distance between the data points of both classes [18].

The application of SVMs to ECG classification has been widely studied. For instance, in a study done by Ubeyli, an SVM method was applied with error output correction to classify four different arrhythmias from ECG signals: normal, congestive heart failure, ventricular tachycardia, and atrial fibrillation [19]. The pre-processing and feature-extraction of the signals was performed using a discrete wavelet transform. This model reached an accuracy of 98.61% after being tested on a dataset consisting of 360 beats [19]. Unfortunately, SVMs are not suitable for larger datasets and do not perform well when working with large feature sets. Moreover, it is difficult to implement multiclass classification with an SVM which is not ideal when designing medical diagnostic systems.

2.2 - Convolutional Neural Network with One-Hot Encoding and Information Fusion Techniques

Another method discussed in paper [3] uses information fusion and one-hot encoding techniques before using a Convolutional Neural Network (CNN) for ECG classification. In this study, information fusion is used to attain a two-dimensional information vector from the morphology and rhythm of a heartbeat. The two-dimensional vector is then passed through a CNN that includes adaptive learning rate and biased dropout methods for subsequent processing and classification. The results demonstrate that the proposed CNN structure is effective in detecting arrhythmias and irregular heartbeats using automatic feature extraction. The model was tested using the MIT-BIH arrhythmia database and achieved a consistent detection accuracy of 97% for eight different classes of abnormalities [3]. The performance achieved is higher than many state-of-the-art methods used today in terms of both sensitivity and predictive rates.

The application of a CNN reduces the need for external feature extraction as the extraction of deep features is handled within the network itself. This reduces the need for complex, and computationally expensive algorithms needed for external feature extraction. However, this approach has a tradeoff. CNNs are incredibly computationally expensive and this complexity scales exponentially with the resolution of the data being analyzed. Moreover, it should be noted that the learning rate of CNNs are typically much slower and requires more resources than that of a computationally simpler ANN.

2.3 - Multimodal Decision Learning Method

This 2016 paper [20] illustrates a new classification method for ECG signals called Multimodal Decision Learning (MDL) algorithm. The objective of the paper is to compare the performance of this method to the existing and advanced neuro-fuzzy method. For the preprocessing stage, the ECG signals are selected from the MIT-BIH database and filtered by a Gaussian Mean Variant (GMV) technique to eliminate common noise artifacts. Features are then extracted using an Integrated Peak Analyzer. After feature selection, the MDL classification method is applied which labels a signal as either "normal" or "abnormal" using a comparative analysis that uses parameters such as true positive, true negative, false positive and false negative. The MDL model used in this study employed the help of a novel kernel model for classification. The final model achieved an accuracy of 87.5%

This study was limited to binary classification between PVC and normal beats and still achieved a lower accuracy than most other popular methods that were able to achieve accurate classifications of numerous arrhythmias. However, the goal of this study was to compare the multimodal approach versus the existing neuro-fuzzy method, not to achieve the highest accuracy possible. As a result, the same comparative analysis is applied to both methods and a higher accuracy was ultimately achieved for the multimodal system. This indicates that a system that uses a neural network classifier with high quality pre-processing will be able to achieve a classification accuracy that is competitive with current state-of-the-art approaches.

2.4 - Dynamic Bayesian Network Approach

The approach discussed in paper [21] employs a dynamic Bayesian network to assist in the classification of ECG signals. In this study, three well known Bayesian classifiers were tested and compared to distinguish between PVC and sinus rhythm. The feature extraction process is simplified in the Bayesian approach by using more straightforward generative model algorithms as opposed to other more complex

methods such as wavelet transformations. After the typical denoising of the ECG signal and beat segmentation, each beat was divided into four vectors. Each of the quartered vector was then staged for the extraction of 20 features. The feature extraction stage consisted of two methods: Feature Statistic Calculation (FSC) and Samples Feature Extraction (SFE). In FSC, general mathematical characteristics such as the min, max, and standard deviation were collected from each beat segment. The SFE consisted of extracting equally spaced characteristic samples from the beat and then applying the Discrete Fourier Transform (DFT) to each. This process produced a total of 80 features to pass to the classifier. For classification, the quadratic discriminant analysis (QDA) classifier model is used to train the model weights and to make inferences from the provided features. QDA is an algorithm derived from Bayes theorem but has the advantage of determining non-linear patterns in the data. The QDA version of the dynamic Bayesian model achieved the highest accuracy of 99.7% when classifying PVC signals.

The application of a Bayesian network certainly decreases the complexity of the classification system which, in turn, increases portability by reducing the need for heavy processing power. The Bayesian approach has the advantage of not needing to tune hyper parameters as in logistic regression, SVMs, and neural networks. As a result, time and resources are saved by eliminating the need for long arduous training times. As a result, the process is much less convoluted than a CNN for instance, making it easier to understand and explain what is happening. However, this method lacks the ability to effectively scale the system for the classification of many arrhythmias. QDA is only effective when the classes are easily differentiable. The addition of classes, especially those that are similar, would complicate things which would consequently impact the accuracy and effectiveness of the model.

2.5 - Adaptive Neuro-Fuzzy Inference System Method

Previous research done in 2017 examined the performance of an Adaptive Neuro Fuzzy Inference System (ANFIS) for ECG classification. As discussed in paper [22], ANFIS incorporates a fuzzy logic system in tandem with artificial neural networks to analyze ECG signals and is proven to be an efficient way to classify heartbeats. Fuzzy logic differs from Boolean logic by allowing truth values to be any real number between 0 and 1 as opposed to truth values being limited to just the integer values of 0 and 1. This method is well applied to models with imprecise information and for classification decisions that may not always be entirely clear. Since ANFIS is a binary classifier, the outputs of this system are simply the cardiovascular condition of the ECG signal determined by the approach. Decisions made within the ANFIS system are made solely based on the rule-based structure that is developed during supervised learning. The average accuracy achieved by this approach was 98.39% when tested on six different heart conditions.

The ANFIS system has proven to be faster than a typical CNN due to smaller fan-out backpropagation and the fact that the first three layers of the network are unconnected. Unfortunately, the computational complexity of the ANFIS system scales exponentially with the number of input features. This makes it difficult to divide the feature sets into membership functions since too many rules are created due to the increase in the number of inputs. This system limits the number of features that can be passed into the classifier which may consequently impact the performance, especially if the classification of additional classes is desired. The increase in rules, as mentioned earlier, will increase the computational complexity of the system which will increase resource consumption and reduce portability. Moreover, the training time required for this method is significantly longer than a simple feed-forward ANN classifier.

Chapter 3: Background

3.1 – Electrocardiography

Electrocardiography is the measurement of the electrical activity in the human heart. The subsequent recording of such activity is, namely, an electrocardiogram (ECG or EKG). An ECG is a graphical representation of the hearts electrical activity and is typically obtained using electrodes that are placed on the surface of the skin. These electrodes detect the small electrical changes that are caused by the depolarization and repolarization of the cardiac muscle which occur during each cardiac cycle (heartbeat). A normal ECG signal can be broken down into four main events: atrial depolarization, ventricular depolarization, ventricular repolarization, and papillary muscle repolarization. Each of these events correspond to visible features recorded by the ECG, namely, the P wave, QRS complex, T wave, and U wave, respectively. The U wave is typically not seen and is generally ignored in the analysis of an ECG. The remaining three features are important for analyzing ECG signals as they can be indicative of the health of a heart. Each of these features are annotated on the ECG signal shown in Figure 3.

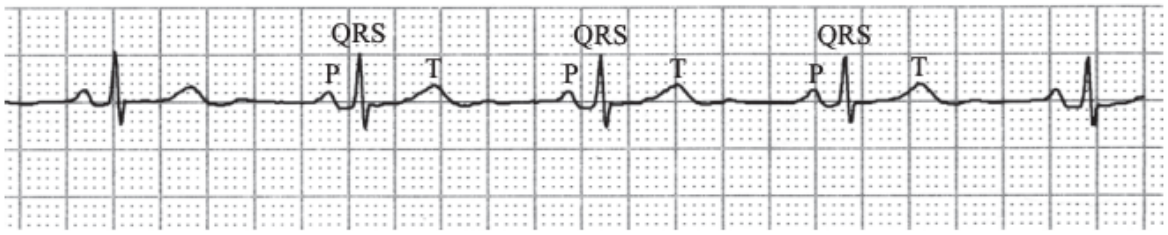


Figure 2: Typical ECG signal with annotated features [26]

When analyzing an ECG signal for abnormalities, there are several important features that should be considered. The most descriptive of these features include rate and rhythm, electrical axis, amplitude, and general waveform morphology [23]. All known cardiac arrhythmias have different effects on the electrical activity of the heart which is why it is important to consider a wide array of features when examining the health of a heart. A conventional ECG uses 10 electrodes placed on the patients' limbs and across the surface of the chest as pictured in Figure 4. This allows the electrical activity of the heart to be observed and recorded from 12 different angles ("leads") for a certain period of time. During this time, a healthy heartbeat will consist of a methodical progression of depolarization quickly followed by repolarization. Depolarization is used for the heart to contract while repolarization is used for the heart to relax [6]. Without these two electrical signals the heart will not beat properly. In this study, the ECG

signals analyzed were collected via the MLII lead, which is located in the fourth intercostal space, to the right of the sternum.

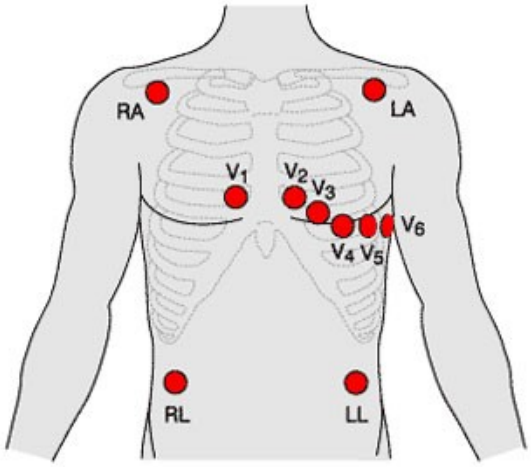


Figure 3: Typical Electrode Placement for 12-lead ECG [23]

3.2 - Anatomy of the Human Heart

The human heart is a muscular organ responsible for pumping blood throughout the circulatory system. This system consists of a network of blood vessels that supplies oxygen and nutrient rich blood to and from all areas of the body. The heart contracts to pump oxygenated blood to organs and tissues, providing the nutrients they need to function properly. At the same time, carbon dioxide is carried back towards the lungs to be breathed out. A healthy heart supplies the body with a sufficient amount of blood at a rate needed to meet the needs of all bodily organs. If the heart is weakened by disease or injury, it can hinder the capability of the cardiovascular system [6].

The contractions of the heart are triggered by electrical signals in three regions of the heart: the sinoatrial (SA) node, atrioventricular (AV) node, and the branches and fibers that deliver the signals to the heart [24]. The sinoatrial node is the internal pacemaker of the heart which produces the electrical signals for the heart to contract. The cardiac cycle begins when conduction cells in the sinoatrial node discharge an action potential that sends an electrical impulse through the atrioventricular node and into the atria and the ventricles. As the impulse spreads through the myocardium, it activates the contractile myocardial cells which respond by contracting [6]. This process generates the change in electrical potential that an ECG measures. The activation of the atrial node (atrial depolarization) correlates with the P-wave whereas the activation of the ventricles (ventricular depolarization) results in the QRS complex. The T-wave reflects

the repolarization of the ventricles (ventricular repolarization) [6]. This process is well illustrated in Figure 5.

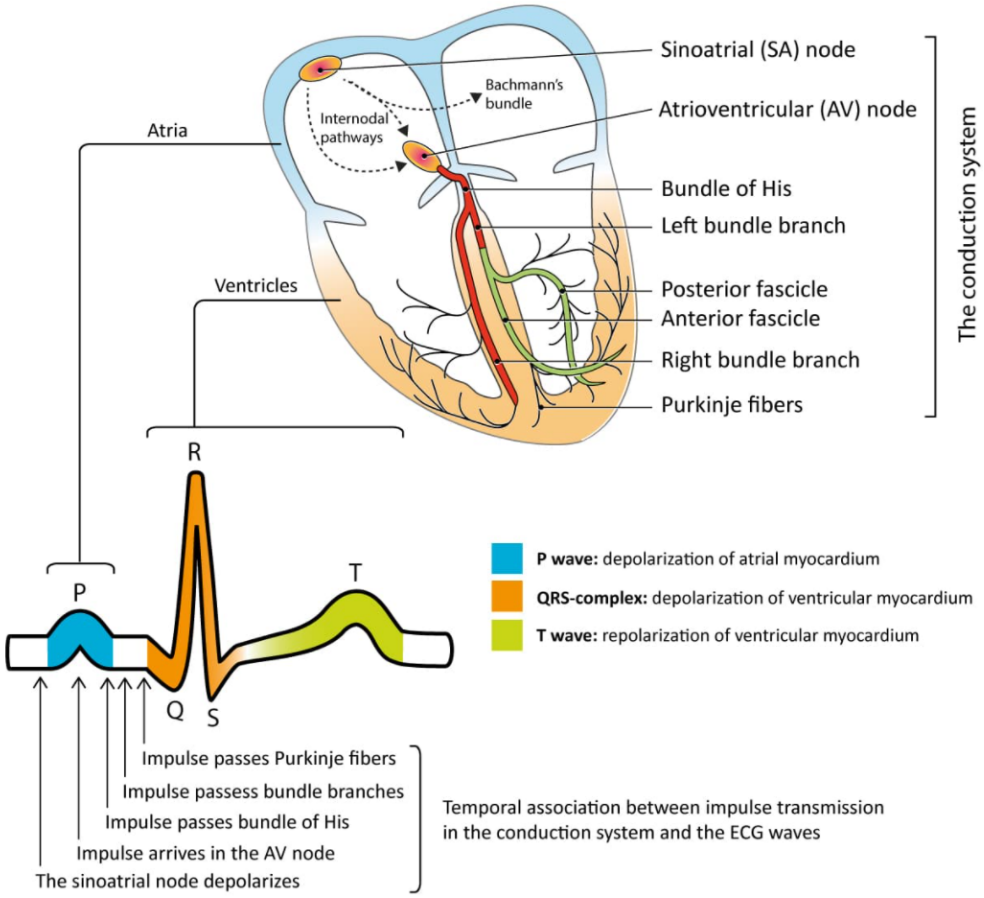


Figure 4: Electrical Signals and the Heart [6]

Each contraction of the heart is caused by electrical activation, so the electrical properties of the heart are very important. Sometimes the electrical events of the heart occur when they are not supposed to, and this can be indicative of problems in the heart [23]. A sinus heartbeat is considered normal and indicates that the heart is healthy.

3.3 - Heart Arrhythmias

This project will focus on diagnosing five different cardiac arrhythmias. These arrhythmias were chosen based on how common they are, and the abundance of available data on them, provided by the MIT-BIH database. These abnormalities, although rare, are some of the most common cardiac arrhythmias and will be important to analyze to maximize the impact and relevance of this classifier.

1. **Premature Ventricular Contraction (PVC)**

Premature ventricular contraction occurs when beats arise before they are expected in the rhythm of the heart. This premature beat is usually followed by a compensatory pulse where a supraventricular pulse occurs near the normal time of a pulse. If the PVC falls between two sinus beats and does not cause a compensatory pause it is called interpolated PVC [26]. Figure 5 shows a PVC occurring in an ECG signal and is annotated by the arrow. The normal pace of the beat is marked by the line above the ECG signal which illustrates that the signal continues with its normal R-R interval after the PVC occurs.

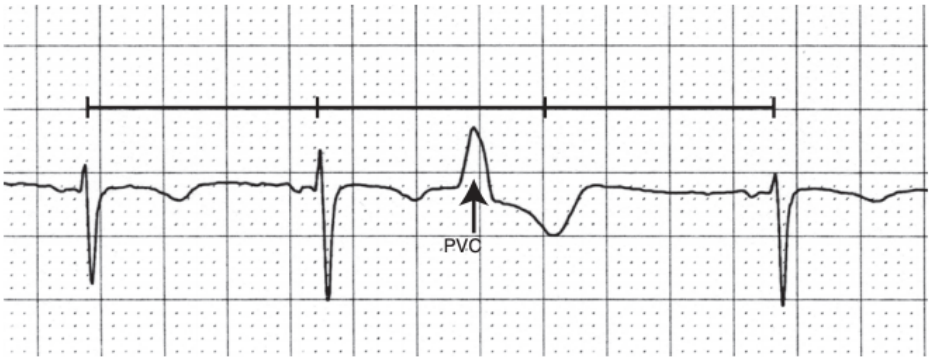


Figure 5: A PVC occurring in an ECG signal which is annotated by the arrow [26].

2. **Atrial Premature Contraction (APC or PAC)**

Atrial premature contraction is classified by a premature onset of a beat. The typical PAC beat usually has a normal QRS duration but does not fall in the normal R-R intervals. There is then a pause following the PAC and the next beat follows on a new interval, so there is no compensatory beat [26]. Figure 6 shows an example of an APC which is annotated by the arrow. The regular pace is illustrated by the segmented line above the signal, which continues at its normal pace after the APC occurs.

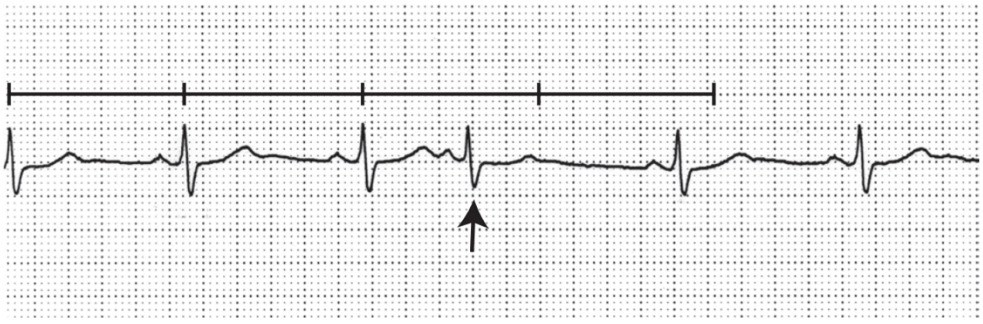


Figure 6: An APC occurring in an ECG signal [26].

3. Left Bundle Branch Block (LBBB)

A left bundle branch block occurs when there is an interference with conduction in the left bundle branch. A bundle branch block is indicated by a QRS that is abnormally prolonged and a supraventricular origin of electrical activity. A left bundle branch block has a tall, broad R wave in leads I and V6 [26]. Figure 7 demonstrates an instance of an LBBB in an ECG signal in the leads where the effects are most prevalent.

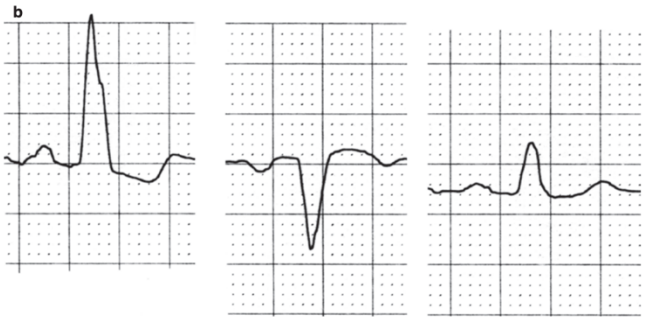


Figure 7 : An LBBB demonstrated by the leads it most significantly influences [26].

4. Right Bundle Branch Block (RBBB)

A right bundle branch block occurs when there is an interference in conduction in the right bundle branch. The right bundle branch block QRS configuration shows a broad S terminal wave in V6 and a tall, broad R wave in V1 [26]. Figure 8 demonstrates an instance of an RBBB in an ECG signal in the leads where the effects are most prevalent.

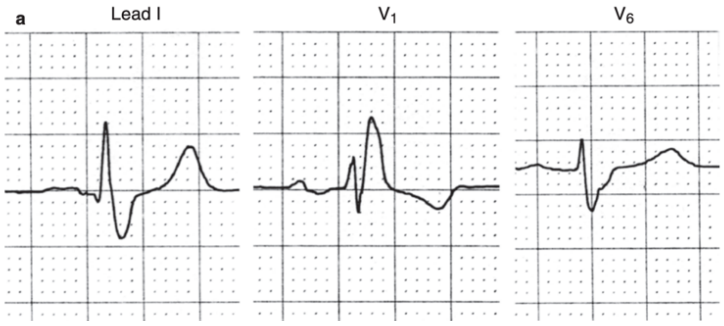


Figure 8: An RBBB demonstrated by the leads it most significantly influences [26].

5. Paced Beat

The paced beat is an electrical signal created by a pacemaker. It is important when studying electrocardiography to be able to recognize the presence of a pacemaker. The spikes produced by the pacemaker are often narrower in width and vary in height. The pacemakers beat is set by the programmed delay and the electrodes sense if depolarization has occurred in order to make sure the pacemaker does not set off another spike when one has already occurred [26]. Figure 9 demonstrates an instance of an atrial ventricular pacemaker beat; the atrial beats are labeled with an 'A' and the ventricular beats are labeled with a 'V'.

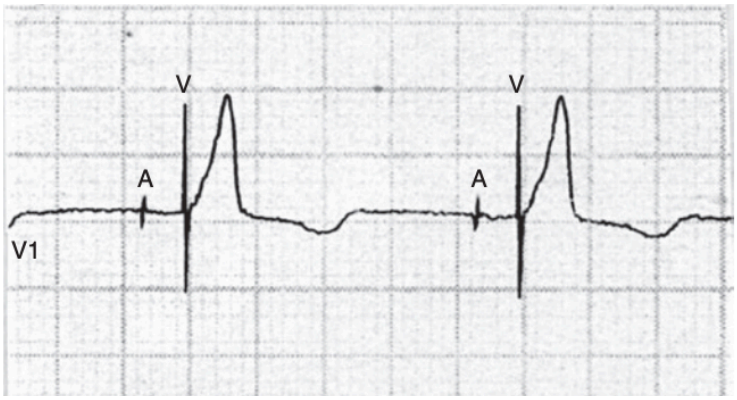


Figure 9: Atrial ventricular pacemaker beat [26].

3.4 - MIT-BIH Arrhythmia Database

The use of a reliable ECG signal database is critical for the training and testing of any arrhythmia classifier. The Massachusetts Institute of Technology-Beth Israel Hospital (MIT-BIH) is a publicly available arrhythmia database that provides enough data with a wide range of abnormalities and arrhythmias for testing. The database consists of 48, half-hour excerpts of two-channel ambulatory ECG

recordings collected from 47 patients which were studied by the BIH Arrhythmia Laboratory between the years of 1975 and 1979 [1]. Of these 48 recordings, 23 were chosen at random from a collection of over 4000 Holter tapes. The remaining 25 were specially selected to include examples of clinically important arrhythmias that would otherwise not be well-represented [27]. The subject group consisted of 25 males between the ages of 32 and 89 years and 22 women ranging from 23 to 89 years old. Approximately 60% of the subject group were inpatients. As expected in typical clinical practice, the placement of the ECG leads varied among subjects' due to anatomical variation. The digitized ECG recordings are captured at 360 samples per second with an 11-bit resolution over a 10mV range [1]. During the digitization process, a simple digital notch filter was applied to remove 60Hz interference caused by powerline interference. Each signal has been independently annotated beat-by-beat and checked by a team of professional cardiologists.

3.5 - Noise Artifacts

Electrocardiography is an easy and non-invasive method for recording heart activity through surface measurements. However, measuring electrical potential from the body's surface is prone to various types of noise. The electrodes are designed to pick up low-voltage signals which make them incredibly sensitive to minimal electrode changes of the skin [28]. Additionally, biomedical signals within the body are known to interfere with one another which often contribute to the contamination of an ECG recording. Both internal and external sources are known to induce noise which, if not handled correctly, could lead to false or inaccurate classifications of signals. Typically, the most significant sources of noise are baseline wander, muscle contractions, power line interference, and instrumentation noise. Each source of noise has a different effect on the original signal so each must be handled differently in the denoising process.

Baseline Wander

Baseline wandering is a low-frequency artifact that results in an unwanted varying of the baseline. This type of noise is the result of changes in the impedance between the electrode and the subject's skin which can often be attributed to electrically charged electrodes, perspiration, movement from the patient, or even breathing [28]. The ST segment of the ECG is especially impacted by this type of noise which is particularly relevant for an accurate classification of some arrhythmias. The frequency content of baseline wander is generally below 0.5 Hz but may be higher when stress testing [28]. The MIT-BIH database that will be employed for the testing and training of this system consists of ambulatory recordings from inactive patients so stress testing will not need to be accounted for. Since the induced noise has such a low

frequency it is best removed using a Butterworth high-pass filter with a total order of four a cut-off frequency of 0.5Hz [29]. The Butterworth filter was chosen due to its computational efficiency, robust model, and ease of implementation.

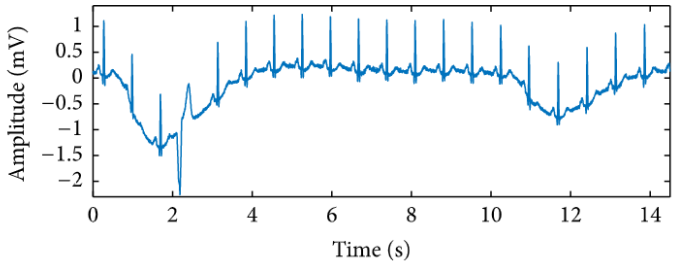


Figure 10: Example of ECG Contaminated with Baseline Wander Noise [29].

Powerline Interference

Electromagnetic fields formed by a powerline or other electrical equipment present another common source of noise in ECG recordings. Some forms of noise, like powerline interference, have known characteristics when it comes to their frequency which make them easier to pinpoint and eliminate from the original signal. Powerline interference is typically characterized by 50 or 60 Hz sinusoidal noise and is nearly unavoidable in an environment like a hospital or clinic where separate medical equipment is almost certainly in proximity. The sinusoidal noise degrades the original signal quality and tends to overwhelm smaller features such as the P-wave and T-wave which are critical for an accurate analysis of the ECG. Fortunately, the frequency spectrum of powerline interference is narrowly centered about the frequency of the powerline. The most effective method of removal for this type of noise is through the application of a narrowband filter. The MIT-BIH database has already somewhat handled the removal of this noise by using a notch filter centered around the 60Hz during the digitization process. However, the classification system should still account for the possibility of powerline interference contamination and should assume it's existence in every signal. The best way to effectively remove powerline interference without altering the original signal is through the implementation of a second order notch filter with a narrow bandwidth centered about 50/60Hz.

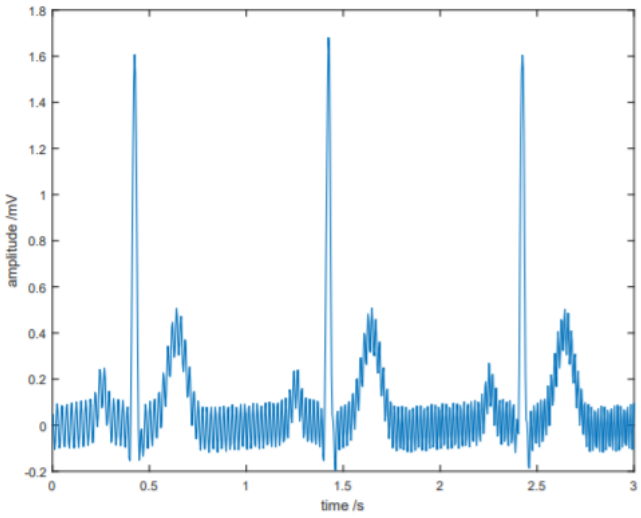


Figure 11: ECG Contaminated with Powerline Interference [28]

Muscle Contractions

Electromyography (EMG) interference is another common noise source present in ECG recordings. EMG noise is the product of skeletal muscle contractions caused by either patient movement or muscle activity [30]. In contrast to baseline wander and powerline interference, EMG noise is much more difficult to detect and effectively remove. The main challenge stems from the fact that muscle contractions are essentially random events from a computer's perspective and thus cannot be removed with narrowband filtering. Even more, the spectral content of muscle noise and the ECG signal overlap and, in some cases, may completely obscure the desired signal. EMG noise becomes more of a problem when ECG signals are collected during exercise or when the room temperature isn't being properly regulated. The application of a time-varying low-pass filter using a variable frequency is a popular approach for dealing with EMG contamination. This particular technique is effective with a Gaussian impulse response filter because the bandwidth is easily changed from one sample to another [30].

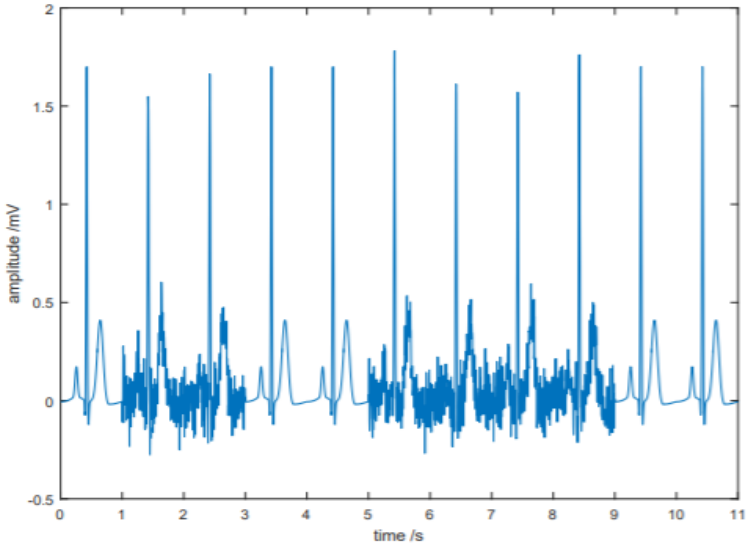


Figure 12: ECG signal with EMG disturbances [28]

Instrumentation Noise

Noise artifacts induced by the ECG machine itself or other electronic devices in the instrumentation system are incredibly difficult to correct. Instrumentation noise is a term used to describe the random electrical activity generated by other electrical devices, usually taking the form of white noise meaning it has similar power at all frequencies. This type of noise is very difficult to remove from a digitized ECG signal and can render an ECG signal unusable if serious enough. The best way to avoid this type of noise is to tackle it at the source through manual prevention and coercive action [31].

3.6 - Wavelet Theory

The fundamental theory behind wavelets is to analyze and represent data according to scale [32]. The wavelet transform provides an entirely new perspective on processing data using both spectral and temporal information. The wavelet transform is similar to the Fourier transform but is advantageous when analyzing physical situations such as an ECG where the signal may include important discontinuities or sharp spikes. The Fourier transform is also limited in regard to only having frequency resolution with no time resolution [33]. For the Fourier transform there is a single window used for all frequencies whereas these windows vary for the wavelet transform which would give us information on the frequencies present globally within an ECG, but not the times that they are present at. To overcome this problem, the wavelet transform is used to cut the signal of interest into several parts and then analyze each section separately. However, it is very important to cut the signal in such a way that retains the signals

information as accurately as possible. This becomes especially apparent when considering the underlying signal processing principle: the Heisenberg uncertainty principle, which states that it is impossible to know the exact frequency and the exact time where this frequency is present in any given signal.

In theory, continuous wavelet transforms (CWT) have an infinite set of possible basis functions. Different wavelet families provide trade-offs when it comes to how compactly the basis functions are localized in space and time and how smooth they are [32]. Wavelets have two basic properties: scale and location. The scale defines how "stretched" a signal is and location defines where the wavelet is in time or space. The equation used to generate other wavelets from the basis wavelet is given below. Scale and location are represented by the variables s and τ respectively. Unlike the Fourier transform, the wavelets basis function is not specified in the equation. This is because the theory of wavelet transforms is only interested in the general properties of the wavelets and wavelet transforms. The wavelet itself is dependent on the application and is thus up to the designer to decide.

$$\Psi_{s,\tau} = \frac{1}{\sqrt{s}}\Psi\left(\frac{1-\tau}{s}\right) \tag{3.1}$$

Regarding the continuous wavelet transform (CWT), a series of wavelets are generated from a single basis wavelet. The subsequent wavelets are made by scaling and shifting the basis wavelet, as seen in *Equation 3.1*. A signal *x(t)* can be transformed using the following equation and can be thought of as a cross-correlation of the signal with a wavelet set of varying widths [31]. The discrete wavelet transform (DWT) is the practical form of the CWT and is what will be implemented into the preprocessing stage of this classification system. However, there are a few issues we must address first. The continuous shifting of a scalable function over a signal and calculating the cross correlation between the two is bound to produce redundancies that, ideally would be avoided. Moreover, an implementation of the DWT will need to scale down the infinite number of wavelets required for a CWT to a more reasonable number without hindering the accuracy of the system.

$$\Psi_{j,k}(t) = \frac{1}{\sqrt{s_0^j}}\Psi\left(\frac{t - k\tau_0 s_0^j}{s_0^j}\right) \tag{3.2}$$

It is worth noting that the discrete wavelets are not actually time-discrete, the scale and translation steps are the only discrete components of the function. The translation and sampling factors are represented by

τ_0 and s_0 respectively. The natural choice for s_0 is 2 so the sampling frequency axis corresponds to the dyadic sampling [33]. The translation factor τ_0 is chosen to be 1 so the time axis coincides with dyadic sampling as well.

The removal of redundancies is achieved by ensuring the discrete wavelets are orthonormal. This can only be achieved with discrete wavelets. Discrete wavelets are made orthogonal to their own dilations and translations through the careful selection of a basis wavelet [33]. This implies that the information stored in the wavelet coefficient is not repeated and thus allows for the complete reconstruction of the original signal.

$$\int \Psi_{j,k}(t)\Psi^*_{m,n}(t)\,dt = \begin{cases} 1 & if\ j = m\ and\ k = n \\ 0 & otherwise \end{cases} \tag{3.3}$$

After the removal of redundancies from the transform, there is still the issue of representing an infinite number of scaling and translations in discrete time. From Fourier theory, we know that compression in time is equivalent to stretching the spectrum and shifting it upwards [33]. And since wavelets each spectrum is similar to band-pass filters, a series of dilated wavelets can be seen as a band-pass filter bank which will provide good coverage of the signal spectrum. However, there is still one more important issue to address. To provide coverage of the spectrum all the way down to zero, it would not be feasible to use a seemingly infinite amount of wavelet spectra. Instead, a low-pass spectrum belonging to the so-called *scaling function* is applied similar to what is shown below [33]. The scaling function takes care of the spectrums that would otherwise be covered by wavelets up to a certain scale, and the rest is done by wavelets. This method effectively circumvents the need to represent an infinite number of wavelets while maintaining a high level of information retention. The original signal will still be possible to reconstruct even after applying this method.

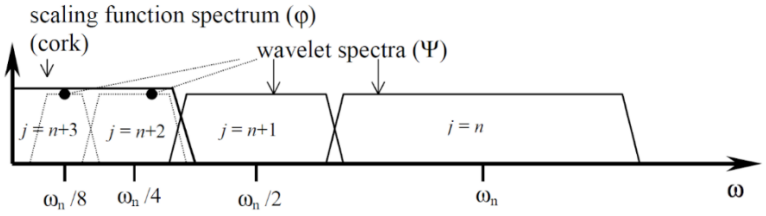


Figure 13: Example of how a scaling function can replace an infinite set of wavelets [33].

3.7 - Feed Forward Neural Networks

Feed forward neural networks (FNN) use one-way and one-dimensional connections between their neurons. The neurons on the same level form various layers of the FNN [34]. The main parts of a FNN are the input and output layers with hidden layers in between. The one-way connections in an FNN are assigned weights and biases which are responsible for determining the output of the FNN. FNNs are applied in many different areas that neural networks are used for and are very useful for the amount of signals processing needed for ECG analysis [35].

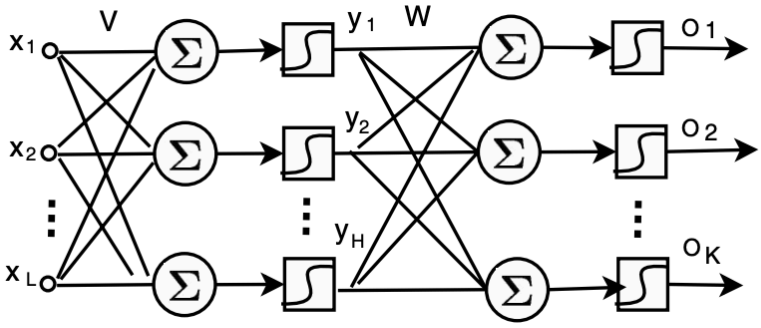


Figure 14: Example of an FNN with a single hidden layer [35].

Chapter 4: Our Approach

The classification model discussed in this paper takes a unique approach to arrhythmia detection and classification. The design can be divided into four different stages: preprocessing, feature extraction, feedforward neural network, and classification. Figure 15 provides a simple block diagram of the proposed classification model. The following sections will provide greater detail on the methodology and reasoning behind each stage.

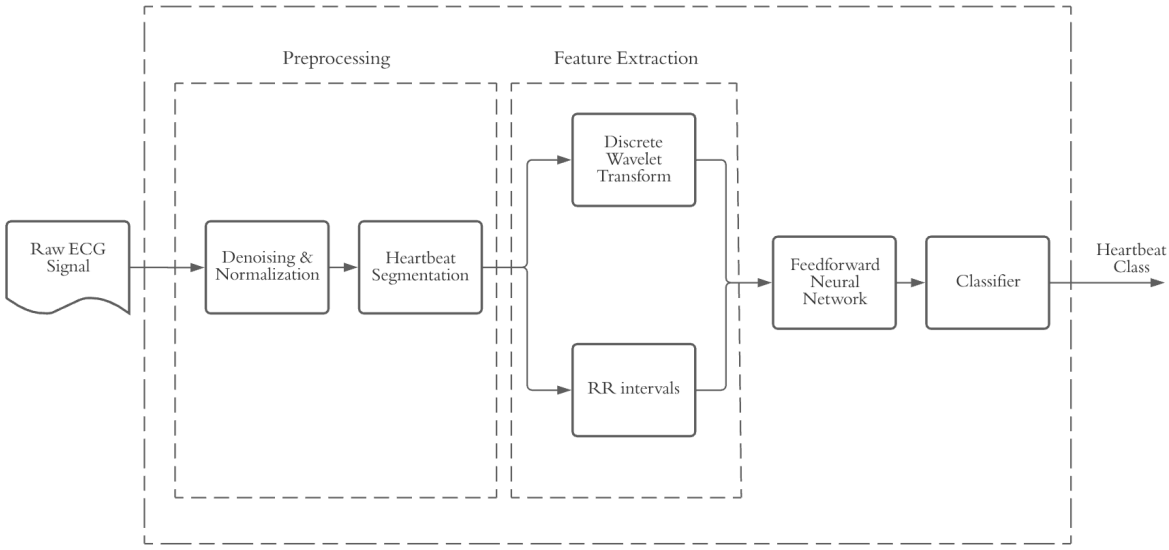


Figure 15: System block diagram

4.1 - Preprocessing

The preprocessing stage of the classification system is responsible for denoising, normalizing and segmenting the raw ECG signals. As discussed earlier, the ECG signals used are sourced from the MIT-BIH arrhythmia database. These clinically collected ECG signals are corrupted by various noises such as baseline wander, electromyography disturbance, and powerline interference. As a result, a combination of filters must be applied to the signals to remove most of the intrusive noise while ensuring that important signal information is not filtered out. As such, each individual ECG signal is first passed through a 2nd order high-pass Butterworth filter with a cutoff frequency of 0.5Hz to eliminate any baseline wandering present in the signal. The difference equation for the Butterworth filter is shown in equation 4.1. Next, a second order notch filter with a narrow bandwidth centered about 60Hz is applied to reduce any

powerline interference contaminating the signal. The difference equation for the notch filter is shown in equation 4.2.

$$y[n] - 1.9876y[n-1] + 0.9877y[n-1] = 0.9938x[n] - 1.9876x[n-1] + 0.9938x[n-2] \quad (4.1)$$

$$y[n] - 0.9828y[n-1] + 0.9656y[n-1] = 0.9828x[n] - 0.9828x[n-1] + 0.9828[n-2] \quad (4.2)$$

After denoising the ECG signals, it is important to normalize the amplitudes of the R-peaks. This is done on a patient-by-patient basis. First, the signals R-peaks must be aligned, as the annotations provided by the MIT-BIH database have a small margin of error that may impact the results. Then the entire signal is divided by the average R-peak amplitude, which is computed across all R-peaks. The signal is now ready to be segmented into individual heartbeats.

The next step is beat segmentation. The denoised ECG signals must be separated into different segments, each containing a single heartbeat. Heartbeat segmentation is a very important part of ECG analysis because it breaks up an ECG signal into individual heartbeats for analysis. ECG signals can be recorded for hours and a single patient may exhibit multiple classes of beats, so it is important to break signals up into each PQRST complex [36]. The heartbeat segments must contain the entire PQRST complex which is centered about the R peak. Each heartbeat is measured from just before the P-wave and just after the T-wave. Taking into consideration that not all heartbeats are of the same length or shape, the beats were segmented using a generous range of 250ms (90 samples) before the R peak, and 300ms (110 samples) after the R peak to account for the variations in beat length. Figure 15 shows an entire PQRST complex with important features labeled.

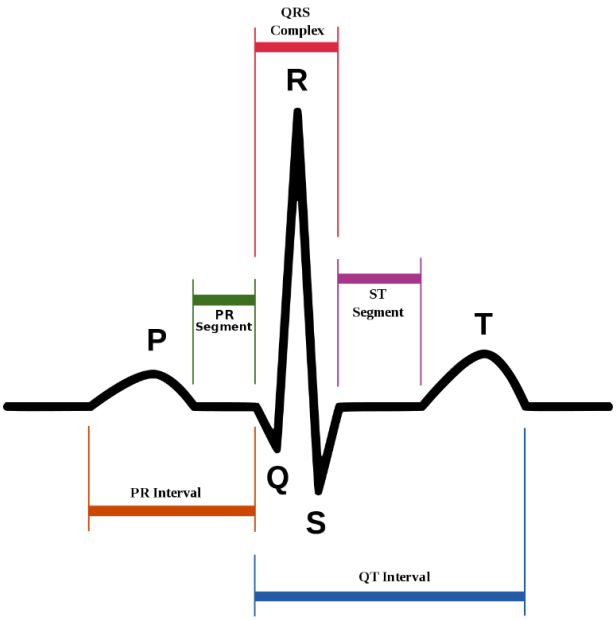


Figure 16: Annotated Example of a typical PQRST Complex [36].

After beat segmentation, the beats are organized and separated by arrhythmia. We have selected five of the most common arrhythmias for the training and testing of our classifier: Normal, PVC, APC, LBBB, RBBB, and paced beat (P). Each arrhythmia has a unique set of abnormalities that are most noticeable within the PQRST complex. The signals provided by the MIT-BIH database are carefully annotated and labeled which are used to filter out any unwanted heartbeats and construct evenly distributed datasets. After segmentation, all beat-classes that are not considered by the model are removed from the dataset. Next, the heartbeats are sorted into training, validation, and testing datasets, each with an evenly distributed number of afflicted beats. A total of 46 signals were used from the MIT-BIH database, resulting in over 100,000 heartbeats which was eventually reduced to approximately 40,000 beats after normalizing the imbalanced distribution of data. Each of these ECG signals belong to different patients. When training a classification model, it is important to use a balanced dataset with a similar distribution of each class in the training and testing datasets. After this stage is completed, the heartbeats are ready to have features extracted.

Table 1: Dataset Beat Distribution

		Beat Type					
		N	**L**	**R**	**V**	**A**	**P**
Dataset	**Training**	6800	3538	3399	3311	728	1824
	Validation	756	393	378	368	81	202
	Testing	500	500	500	500	500	500

4.2 - Feature Extraction

The feature extraction stage is responsible for extracting important information unique to each heartbeat. These feature sets represent the beat when they are passed through the artificial neural network for analysis and classification. The resulting feature sets must be of manageable size while also maintaining enough information to allow the neural network to sufficiently describe and classify each heartbeat. We used two methods to extract a total of 29 features from each heartbeat, 25 of which are products of the discrete wavelet transform, and the remaining 4 are important R-intervals computed from the time between consequent beats. The visible effects of most heart arrhythmias are concentrated within the PQRST complex. The discrete wavelet transform was chosen because of its advantage over the Fourier transform and the ability to extract both spectral and temporal information simultaneously. In this model, a 1-dimensional discrete wavelet transform is performed on each beat using the Daubechies-2 mother wavelet with a decomposition level of three due to its success in previous studies [19]. The Daubechies-2 wavelet is thought to be so effective on ECG signals because its similarity in shape. The discrete wavelet transform returns a unique multilevel array of approximation coefficients. The discrete wavelet transform is an efficient computation, and the 25 resulting coefficients are descriptive, yet do not consume too much memory.

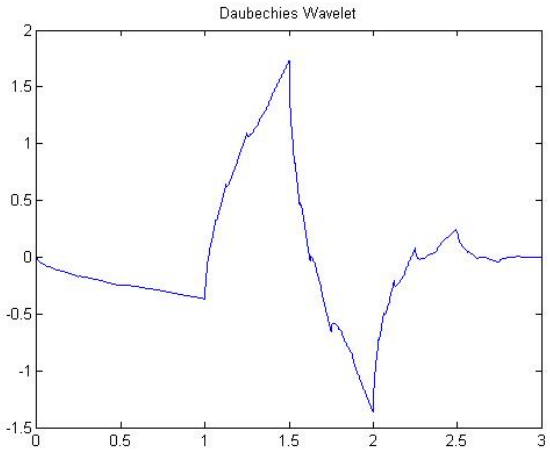


Figure 17: Daubechies-2 Mother Wavelet [43]

In addition to impacting the morphology of the heartbeat, heart arrhythmias also affect the surrounding R peak intervals (RR intervals). For this reason, four widely used features pertaining to RR intervals were computed for every heartbeat. These includes the previous-RR, post-RR, local-RR, and global-RR. The previous-RR is the time between the current RR and the previous RR. The post-RR is the time between the current RR and the next RR. The local-RR is the average of the ten most recent RR intervals, and the

global-RR is the average RR interval across the entire signal. All four of the RR interval features were normalized with respect to the average to reduce inter-patient variation and any heartbeats with unrealistic intervals were removed due to segmentation error. After extracting all 29 features, they are assigned their respective label and are ready to pass through the neural network.

4.3 - Neural Network

For the neural network, we chose to implement a feed-forward design. The feed forward neural network was designed with simplicity and training factors in mind. The final structure consists of an input layer, four hidden layers, and an output layer with a softmax activation function. The structure has over 18,000 parameters that are adjusted during the training process through supervised learning. The datasets created from the MIT-BIH database were previously split up into testing, validation, and training datasets which are using in training and validating the neural network and eventually testing the final neural network for accuracy. The training and testing datasets each use data from different patients and are kept separate throughout the entire process to ensure the network does not see the testing data before the evaluation and to ensure the results are authentic.

4.3.1 - Neural Network Structure

The feed forward neural network was chosen as the structure for its simplicity and dependability. There are two inputs in the input layer which correspond to the two feature sets that were chosen. Both layers were concatenated into a single set making up a combined 29 features for each input beat. Initial weights were assigned for every connection between the layers of the neural network. The initial weights are automatically set to random values between zero and one before they are adjusted during the training process. The structure of the neural network consists of four hidden layers followed by an output layer. The hidden layers contain a large number of neurons that perform transformations to the inputs based on the weights and activation function. The first hidden layer in our design has 128 neurons, the second and third both have 64, and the fourth has 32. These numbers were chosen based on the designs of similar studies and through our own experimentation. This structure has a total of 18,534 trainable parameters which is manageable for the scope of this project's goals. Throughout this process it was important to maintain a small number of trainable parameters to reduce the training time, but still enough to have satisfactory results.

Each hidden layer is composed of two parts: a dense layer followed by a dropout layer. The dense layer is a regular, fully connected set of neurons that apply an activation function. Each dense layer is followed by

a dropout layer which is helpful for preventing overfitting. At each dense layer of the neural network, the Relu activation function is used to adjust the weights. The Relu function, shown mathematically in *Equation 4.3*, is a piecewise function that returns zero for all inputs below zero and for inputs greater than zero it directly returns the value provided as an input. This activation function is popular in FNN classification models because it is easier to train and often achieves better performance. Relu is so effective because it tends to be less susceptible to losing gradients in the hidden layers during training [38]. The output layer of the network employs a softmax (or sigmoid) activation function to convert the vector returned by the final hidden layer into a vector of six probabilities. The formula for the softmax activation function is shown in *Equation 4.4*. These probabilities correspond to the five arrhythmias discussed in section 3.3 in addition to a normal, sinus beat. The probabilities represent the relative confidence of each class after passing a beat through the neural network. In short, the neural network outputs six weights adding up to a total of one and the weight corresponding to the best-predicted feature will hold the largest value. This output vector is then passed to the classifier to determine the best prediction out of all six classes.

$$R(z) = \max(0, z) \tag{4.3}$$

$$\boldsymbol{\sigma(z)_i} = \frac{\boldsymbol{e^{z_i}}}{\sum_{\boldsymbol{j=1}}^{\boldsymbol{k}} \boldsymbol{e^{z_j}}} \tag{4.4}$$

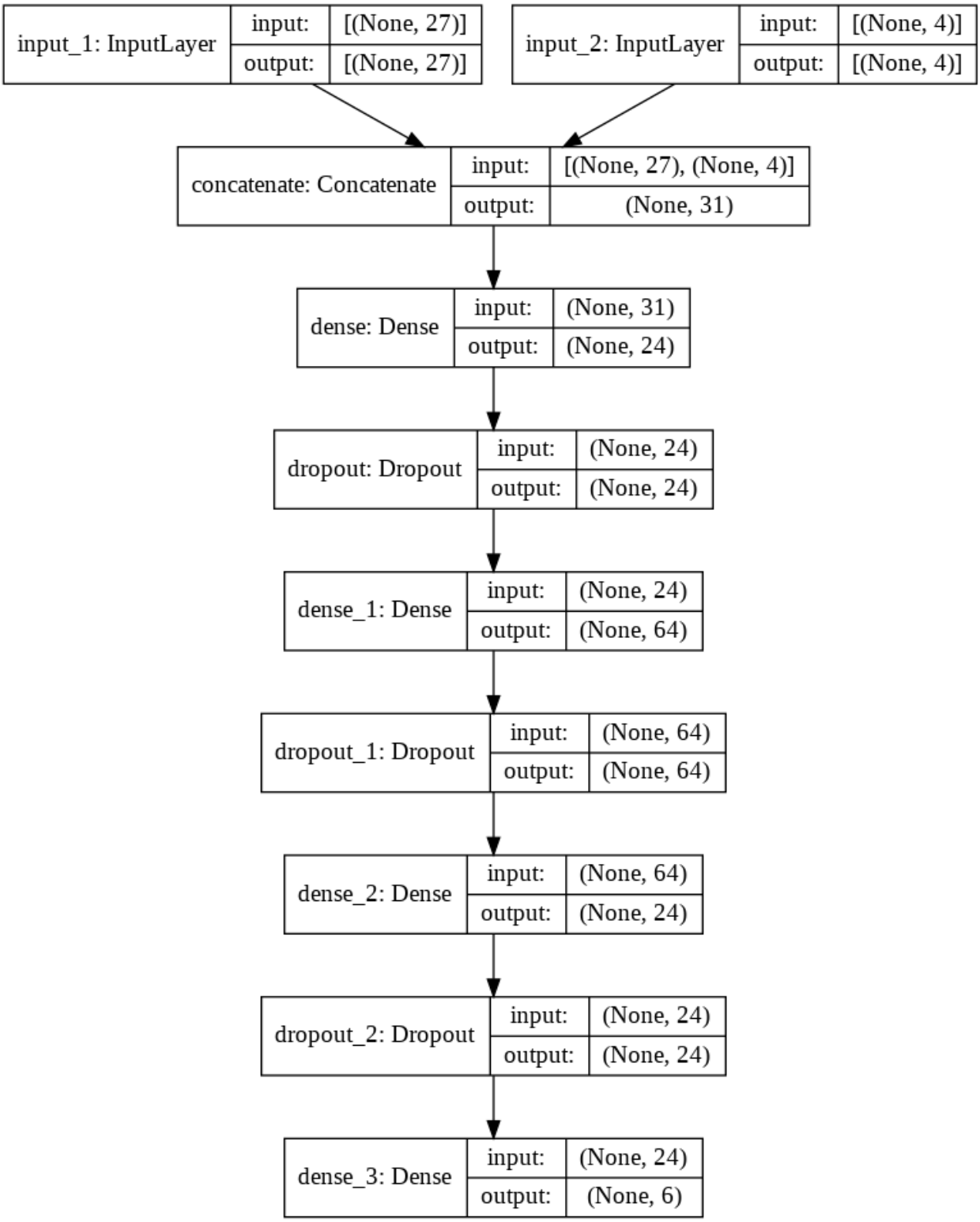


Figure 18: FNN model flow chart.

4.3.2- Neural Network Training

After the initial structure is set up, the weights of the hidden layers are trained using the training dataset which uses signals from the MIT-BIH database. As discussed in section 4.1, the training set is carefully composed from the MIT-BIH database and is properly denoised, segmented, and adjusted for training. After segmentation, the training included nearly 22,000 heartbeats with a relatively even distribution across all six classes of beats. Every beat in the MIT-BIH database is labeled with its corresponding beat type, making it far easier to perform supervised learning, which allows the network to receive instant feedback during the training process. The features that the network focused on for training were the R peak intervals and the output of the wavelet transform. These two parameters were chosen because of their relevance to the arrhythmias our project was targeting. When the R-R interval changes, it can give some indication about what abnormality may be present and the output of the wavelet helps to further reduce noise in the signal. As each epoch of training was completed the neural network weights get adjusted using the optimization function. The optimization function chosen for this project was Adam with a learning rate of 0.0005 which uses stochastic gradient descent. The Adam optimization function uses *Equation 4.3* to update the weights at each epoch. In *Equation 4.5*, η stands for the step size and m and v are moving averages where m represents the moment used in stochastic gradient descent [39].

$$w_t = w_{t-1} - \eta \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon} \tag{4.5}$$

$$CCE(p, t) = -\sum_{c=1}^{C} t_{o,c} \log(p_{o,c}) \tag{4.6}$$

Categorical cross entropy was chosen as the loss function because of its popularity and success in models that handle multiple classes [40]. The equation for categorical cross entropy is shown in *Equation 4.6.* As discussed earlier, the output layer uses a softmax activation function which requires that the labels be one-hot encoded for the supervised learning. An overview of the FNN model parameters is outlined in *Table 1*. The batch size was set at 128 to reduce the number of epochs required and to speed up the training process. A Google Colab notebook was used for the training and testing of the model which allows for a virtually hosted runtime with the optional use of a hardware accelerator such as a GPU. This environment allowed us to select a larger batch size without overloading our local machines. The number of epochs was set to 500 to allow enough iterations to make it through the entire training dataset and to ultimately give enough time to properly adjust the trainable parameters of the model. After experimenting with

learning rates, the best results were achieved using a value of 0.0005 with the Adam optimization function.

Table 2: Final Training/Testing Neural Network Parameters

Parameter	Value
Batch size	256
Learning Rate	0.0005
Epochs	750
Layers	5
Hidden Layers Activation Function	ReLu
Output Layer Activation Function	softmax
Dropout rate	0.1
Loss Function	Categorical Cross Entropy
Optimization Function	Adam

4.4 - Classifier

The classifier stage is the final stage of the network and is responsible for determining the most probable classification using the information provided by the neural network. As discussed earlier, the FNN model uses the SoftMax activation function to return a vector of readable data. The softmax activation function converts output of the neural networks fully connected layer into vector probabilities [17]. In other words, every probability in the array represents a different class, each with a percentage confidence that a particular heart belongs to that class. This concept is illustrated well in *Figure 19*. The classification stage takes this array and outputs the class with the highest confidence. This is an important part of the structure in terms of both training and testing because it determines if the model was able to accurately classify the heartbeats. Since the FNN model uses supervised training, the output of the classifier can be verified for correctness every epoch and used to adjust the weights between neurons.

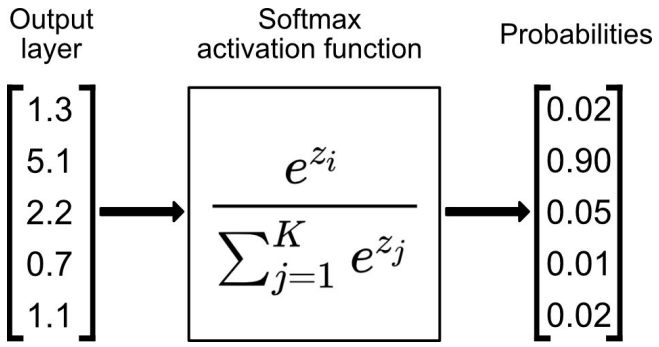


Figure 19: Visual representation of a SoftMax activation [17]

4.4.1- Evaluation Method

After the neural network is trained and the weights have been properly adjusted, the performance of the classification model is evaluated using the testing dataset. It is important to note that the test data was hidden from the network during the training stage to be representative of real use of the network on signals it has never seen before. The testing set was evaluated with the trained model and the resulting outputs were compared to the target outputs. When evaluating our classifier, we focused on a few important metrics: accuracy, sensitivity, specificity, precision, and F1 score. We used the mean squared error loss to calculate the percent accuracy because there were many different outputs, and this compares outputs to their target values. An emphasis was placed on sensitivity and specificity as they are widely used metrics in medical and biology related fields [41]. Sensitivity is the percentage of beats with an arrhythmia that are correctly identified, while specificity is the percentage of beats without the arrhythmia that are correctly excluded by the test. Clinically, these concepts are important for confirming or excluding disease during screening [44]. Ideally, the model will provide a high sensitivity and specificity.

Chapter 5: Data and Results

The following section provides a comprehensive examination of the results obtained from the classification model discussed in this report. To evaluate the performance of the proposed model, a few widely used metrics were applied: accuracy, sensitivity, specificity, precision, and F1-score. These metrics are defined below as Equations 5.1, 5.2, 5.3, 5.4 and 5.5, respectively. These equations employ the variables: TP, TN, FP, and FN, where TP represents the number of true positives, or the number of beats correctly predicted, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. This information was collected by the model for the training, validation, and testing sets, and the metrics were calculated with respect to each arrhythmia using the formulas discussed above. Each metric is important for indicating different aspects of the model's performance. Accuracy is the most straightforward as it signifies the ratio between correctly classified heartbeats compared to the number of total heartbeats. However, for medical applications it is important to consider more than just accuracy as the datasets are typically imbalanced heavily in favor of "normal" behavior which can skew the weighting of the model. In the case of ECG classification, heart arrhythmias are rare, but can pose a serious threat to a patient's health, so it is crucial that a classification model can consistently classify outliers. For this reason, we have employed a few metrics to signify more niche information about specific classes. The sensitivity and specificity metrics are popular for medical applications as they determine the percent of samples in a class correctly predicted by the model, and the percent of samples correctly identified as negatives. The precision metric is best used when a dataset is imbalanced as it signifies the ratio between correct and incorrect classifications for every class. Finally, F1-score provides a combination of sensitivity and precision, making it a convenient tool for determining the overall effectiveness of the model on each class. The metrics for the training set, validation set, and training set, are compiled into Table 3, Table 5, and Table 7, respectively.

The following graphs represent important data collected for every epoch of the training process. The plot on the left illustrates a visual of the training and validation loss measured over the training duration. As discussed earlier, when optimizing a classification network, the loss represents the difference between the predicted and actual values. There are two loss functions depicted on this graph: The training loss, in blue, corresponds to the training data set whereas the validation set, in orange, corresponds to the validation data set. Both loss functions approach zero as the model is trained which typically correlates with an increase in classification accuracy. This is the desired behavior considering that the loss function is effectively used to update the model's parameters during the training process to maximize prediction accuracy. With that in mind, the plot on the right illustrates the accuracy of both the training and

validation accuracy throughout the training process. It can be observed that both the training and validation accuracy quickly increase over the first 50 epochs and then slowly approach 1.0 for the remainder of the training. The model avoids overfitting through the utilization of a validation dataset and a limited number of epochs during training.

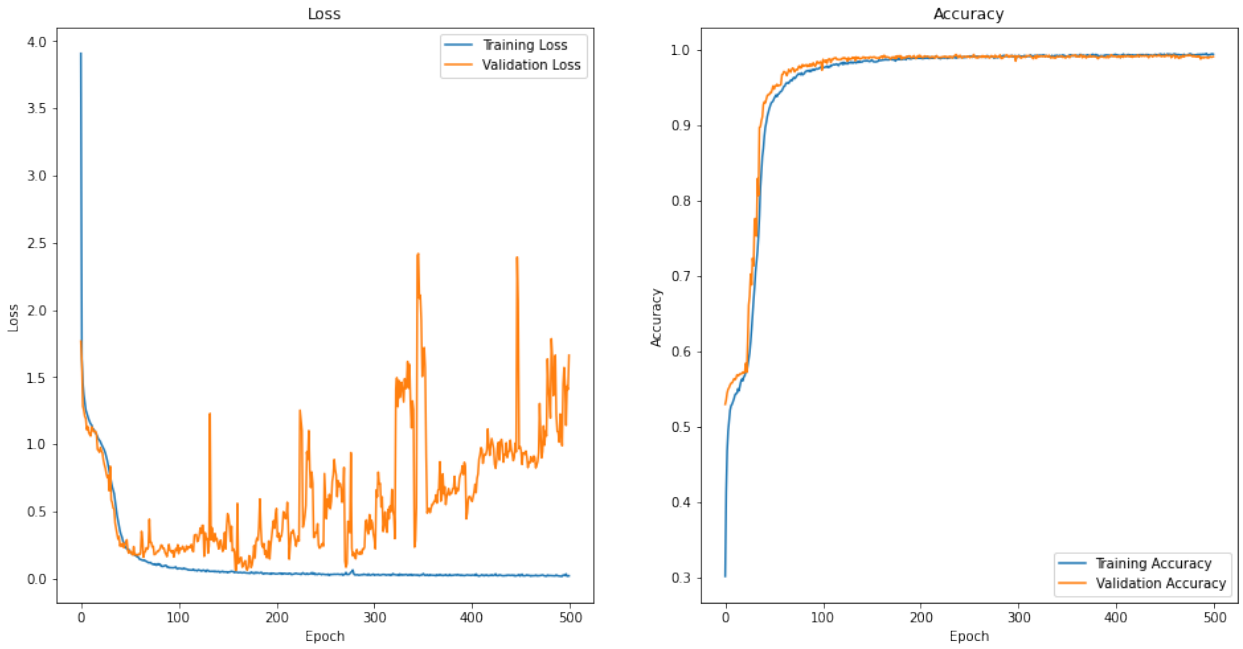


Figure 20: Graphs illustrating the loss and accuracy of the training and validation data sets throughout the training process.

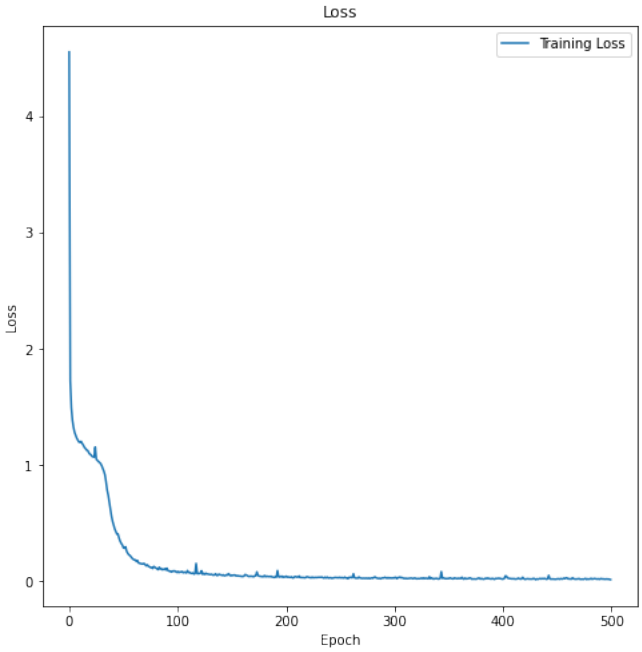


Figure 21: Loss of the model after each epoch of the training process

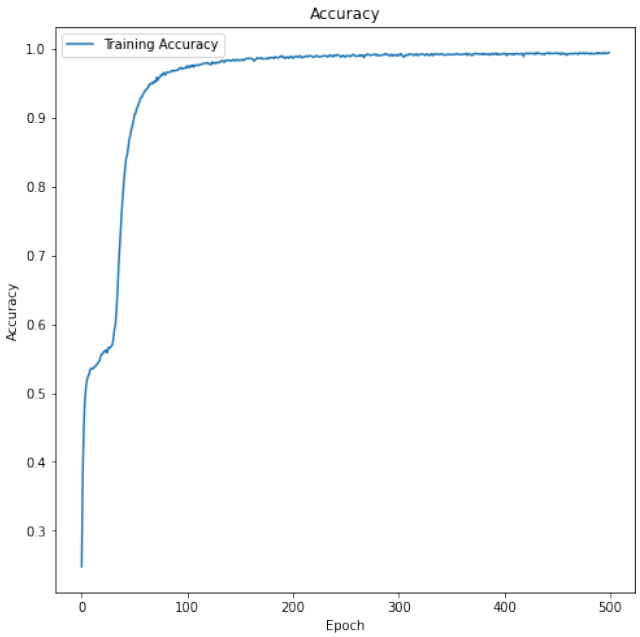


Figure 22: Accuracy of the model after each epoch of the training process

A confusion matrix is arguably the most effective visual to demonstrate the performance of a multi-class classification model. Accordingly, a confusion matrix was constructed for the training, validation, and testing data sets, shown in Table 4, Table 6, and Table 8, respectively. Each entry in the confusion matrix

denotes the number of predictions made by the model, with the columns representing the expected result and the rows representing the predicted result. This visualization technique is effective for determining the accuracy of the model relative to each class and provides insight as to what classes the model is having trouble differentiating between.

$$Accuracy(\%) = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{5.1}$$

$$Sensitivity(\%) = \frac{TP}{TP + FN} \times 100\% \tag{5.2}$$

$$\text{Specificity}(\%) = \frac{\text{TN}}{\text{TN} + \text{FP}} \times 100\% \tag{5.3}$$

$$\text{Precision}(\%) = \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100\% \tag{5.4}$$

$$\text{F1 Score}(\%) = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitivity} \tag{5.5}$$

Table 3: Training Data Accuracy, Sensitivity, and Specificity

	N	L	R	V	A	P
Accuracy	0.9943	0.9994	0.9632	0.9864	0.9299	1.0000
Sensitivity	0.9943	0.9994	0.9632	0.9864	0.9299	1.0000
Specificity	0.9969	0.9999	0.9924	0.9972	0.9973	1.0000
Precision	0.9721	0.9994	1.0000	0.9897	0.9603	0.9989
F1-Score	0.9831	0.9994	0.9813	0.9881	0.9449	0.9995

Table 4: Confusion Matrix for Training Data

		Predicted Class					
		N	**L**	**R**	**V**	**A**	**P**
True Class	**N**	6820	0	0	20	19	0
	L	1	3536	0	1	0	0
	R	121	0	3274	1	3	0
	V	36	1	0	3266	6	2
	A	38	1	0	12	677	0
	P	0	0	0	0	0	1823

Table 5: Validation Data Accuracy, Sensitivity, and Specificity

	N	**L**	**R**	**V**	**A**	**P**
Accuracy	0.9908	0.9975	0.683	0.9856	0.9783	1.0000
Sensitivity	0.9908	0.9975	0.9683	0.9783	0.9506	1.0000
Specificity	0.9950	0.9999	0.9934	0.9956	0.9981	1.0000
Precision	0.9780	0.9975	1.0000	0.9809	0.9277	0.9951
F1-Score	0.9844	0.9975	0.9839	0.9796	0.9390	0.9975

Table 6: Confusion Matrix for Validation Data

		Predicted Class					
		N	**L**	**R**	**V**	**A**	**P**
True Class	**N**	755	0	0	4	3	0
	L	0	392	0	1	0	0
	R	11	1	366	1	0	0
	V	4	0	0	360	3	1
	A	2	0	0	2	77	0
	P	0	0	0	1	0	203

Table 7: Testing Data Accuracy, Sensitivity, and Specificity

	N	**L**	**R**	**V**	**A**	**P**
Accuracy	0.9920	1.0000	0.9680	0.9860	0.9260	1.0000
Sensitivity	0.9920	1.0000	0.9680	0.9860	0.9260	1.0000
Specificity	0.9984	1.0000	0.9936	0.9972	0.9854	1.0000
Precision	0.9101	0.9980	1.0000	0.9821	0.9893	1.0000
F1-Score	0.9493	0.9990	0.9837	0.9840	0.9566	1.0000

Table 8: Confusion Matrix for Testing Data

		Predicted Class					
		N	**L**	**R**	**V**	**A**	**P**
True Class	**N**	496	0	0	2	2	0
	L	0	500	0	0	0	0
	R	15	0	484	0	1	0
	V	5	0	0	493	2	0
	A	29	1	0	7	463	0
	P	0	0	0	0	0	500

The proposed method of ECG classification achieves results comparable with other methods discussed. Tables 7 and 8 provide the best representations regarding the performance of our system as it relates to our testing dataset. The proposed method achieved an overall accuracy of 98.67%, a specificity of 99.57%, and a sensitivity of 97.87%. However, it is important to evaluate each of these metrics on a class-by-class basis since the overall result has the potential to be inflated by the results of a single class. Table 7 provides insight on the results of each class. As expected, the paced beats achieved the highest scores since they often have the same morphological features across patients and are easy to differentiate from other classes. Atrial premature beat has the worst performance by a large margin with an accuracy, sensitivity, and specificity of 92.60%, 92.60%, and 98.54% respectively. This is likely due to the lack of data available for this arrhythmia compared to the other classes. During the preprocessing stage, the class distribution was somewhat normalized but there was still notably less representation for atrial premature beats in the datasets. That said, the classifier still provides a high classification rate for the class, and it provides a solid base for future improvements.

The table below summarizes the results obtained from the testing set for the proposed method in comparison with the methods mentioned in Chapter 2 (See Table 9). More specifically, it compares the three most important, and popular factors for describing ECG classifiers: accuracy, sensitivity, and specificity. It is also important to note the difference in the number of classes that each classifier handles. This has a significant impact on the performance of the system as it is more difficult for a classifier to differentiate between classes as the number of possibilities increases.

Table 9: Comparison to Previous Studies

Study	No. of Classes	Feature Extraction Method	Classifier	Accuracy, Sensitivity, Specificity
[3]	6	-	CNN	99.5%, 99.4%, 99.9%
[19]	4	DWT	SVM	98.61%, 98.61%, 99.54%
[20]	2	Integrated Peak Analysis	MDL	87.5%, 78.6%, 91.2%
[21]	3	FSC, SFE	QDA	98.3%, 100%, 98%
[22]	6	Morphology	ANFIS	98.39%, 92.42%, 99.68%
Proposed Method	6	Discrete Wavelet Transform & R-peaks	FNN	97.67%, 98.54%, 97.87%

Chapter 6: Conclusion and Future Work

Cardiovascular diseases account for over 30% of annual deaths worldwide according to the World Health Organization [1]. Therefore, the accurate detection and diagnosis of arrhythmias in electrocardiograms is crucial for the early detection of at-risk patients. It is also important for reducing the chance of misdiagnosis and missed diagnosis. Current methods employed in medical settings are unreliable and require the scrutiny of a trained cardiologist for best results. A portable solution, such as the one discussed in this paper, will benefit both the patient, and the physician. Neural networks are able to recognize patterns too subtle for the human eye, remove inter-reader variability, and ultimately mitigate the impact of human error.

The utilization of the discrete wavelet transform for feature extraction in tandem with an FNN was ultimately a success. We were able to achieve high performance in the diagnosis of all six chosen arrhythmias. The proposed method achieved an overall accuracy of 98.67%, a specificity of 99.57%, and a sensitivity of 97.87%. Throughout the process of designing our classification system, there were quite a few parameters that needed to be adjusted and fine-tuned to maximize performance. Picking the best wavelet for our project was difficult because there was no obvious choice. Eventually, after testing a few and researching previous studies, we chose to use the Daubechies-2 wavelet. The Daubechies-2 wavelet is thought to be so effective in extracting ECG features because of its similarity in shape [43]. The variation between patients is also difficult to account for when training a model with a somewhat limited database so feature set must be chose to account for this. Every ECG differs based on the patient and the environment it was taken in, so the importance of effective filtering and feature selection was crucial. Moreover, the MIT-BIH has a large imbalance in data which was important to address before training the model. We noticed a strong bias towards the classification of normal heartbeats at first because this was overlooked. However, after adjusting the distribution of heartbeats the model performed much better in terms of both accuracy, specificity, and sensitivity of each minority class. To address the problem of overfitting we simply adjusted the number of epochs and included dropout layers after each dense layer.

In comparison to previous methods, our approach was able to achieve similar performance, and in some cases better performance. More robust methods such as the complex CNN model proposed in paper [3] achieved significantly better performance but required more time and computing power to train and run. The wavelet transforms proved to be very useful and helped simplify the process as it was able to combine spectral and temporal information into a shortened feature vector. Compared to the Rivera and Rodriguez project, we were able to attain similar results with a simpler network and training process [22].

Thanks to the utilization of the DWT and an FNN, our approach was easier to train and more portable. Overall, this approach to ECG classification was successful, but still has plenty of room for improvement.

In terms of future work, the main goal is to enhance the overall performance and robustness of the model and eventually to increase the number of arrhythmias this system can identify. If more arrhythmias are added the performance of the current model will almost certainly decrease. We are unsure how scalable this model is but it will likely never surpass robust methods that employ CNNs and SVMs, especially when considering overall performance and scalability. After all, the proposed method was designed with an emphasis on portability and ease of training while maintaining competitive results in terms of accuracy, specificity, and sensitivity. If this model is ever adjusted to classify additional arrhythmias, the error can be reduced through adding more training parameters and testing out other optimization functions such as Levenerg-Marquardt. If these changes are made, this system will be more applicable in a real medical setting and will prove to be a powerful tool for cardiologists.

References

[1] "Cardiovascular diseases (CVDs)," 17-May-2017. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[2] A. A. S. Raj, N. Dheetsith, S. S. Nair and D. Ghosh, "Auto analysis of ECG signals using artificial neural network," *2014 International Conference on Science Engineering and Management Research (ICSEMR)*, Chennai, 2014, pp. 1-4, doi: 10.1109/ICSEMR.2014.7043597.

[3] J. Li, Y. Si, T. Xu and S. Jiang, "Deep Convolutional Neural Network Based ECG Classification System Using Information Fusion and One-Hot Encoding Techniques", *Mathematical Problems in Engineering*, vol. 2018, December 2018. https://doi.org/10.1155/2018/7354081

[4] R. Silipo and C. Marchesi, "Artificial neural networks for automatic ECG analysis," in *IEEE Transactions on Signal Processing*, vol. 46, no. 5, pp. 1417-1425, May 1998, doi: 10.1109/78.668803.

[5] M. Gertsch, *The ECG: a two-step approach to diagnosis*. Berlin: Springer, 2011.

[6] A. Rawshani, "ECG interpretation: Characteristics of the normal ECG (P-wave, QRS complex, ST segment, T-wave)," *ECG & ECHO LEARNING*, September 23, 2020. Available: https://ecgwaves.com/topic/ecg-normal-p-wave-qrs-complex-st-segment-t-wave-j-point/.

[7] S. Poungponsri and X.-H. Yu, "An adaptive filtering approach for electrocardiogram (ECG) signal noise reduction using neural networks," *Neurocomputing*, vol. 117, pp. 206–213, 2013.

[8] N. V. Thakor and Y. -. Zhu, "Applications of adaptive filtering to ECG analysis: noise cancellation and arrhythmia detection," in IEEE Transactions on Biomedical Engineering, vol. 38, no. 8, pp. 785-794, Aug. 1991, doi: 10.1109/10.83591.

[9] U. Satija, B. Ramkumar and M. S. Manikandan, "A Review of Signal Processing Techniques for Electrocardiogram Signal Quality Assessment," in *IEEE Reviews in Biomedical Engineering*, vol. 11, pp. 36-52, 2018, doi: 10.1109/RBME.2018.2810957.

[10] P. Pławiak, "Novel methodology of cardiac health recognition based on ECG signals and evolutionary-neural system," *Expert Systems with Applications*, vol. 92, pp. 334–349, 2018.

[11] G. Chen, R. Yang, and Z. Jiang, "Automated ECG Analysis and Diagnosis System," October, 2018.

[12] H. Smulyan, "The Computerized ECG: Friend and Foe," *The American Journal of Medicine*, vol. 132, Is. 2, pp. 153-160, February, 2019. doi:10.1016/j.amjmed.2018.08.025.

[13] *ECG morphology*, 10-Aug-2003. [Online]. Available: https://archive.physionet.org/physiotools/ecgsyn/paper/node2.html.

[14] A. Lyon, A. Mincholé, J. Martínez, P. Laguna, et. al., Computational techniques for ECG analysis and interpretation in light of their contribution to medical advances. *Journal of the Royal Society, Interface*, *15*(138), Aug. 2018.

[15] C. Valens, *A Really Friendly Guide to Wavelets*. 1999.

[16] MIT Beth Israel Deaconess Medical Center (BIH). (2005). MIT-BIH Arrhythmia Database. Boston, Massachusetts, United States of America.

[17] D. Radecic, "Softmax Activation Function Explained," *Medium*, 18-Jun-2020. [Online]. Available: https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60.

[18] R. Gandhi, "Support Vector Machine - Introduction to Machine Learning Algorithms," *Medium*, 05-Jul-2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[19] E. Ubeyli, ECG beats classification using multiclass support vector machines with error correcting output codes, *Digital Signal Processing*, Vol. 17, 2007. https://doi.org/10.1016/j.dsp.2006.11.009.

[20] G. R. Naik and K. A. Reddy, "Comparative Analysis of ECG Classification Using Neuro-Fuzzy Algorithm and Multimodal Decision Learning Algorithm: ECG Classification Algorithm," *2016 3rd International Conference on Soft Computing & Machine Intelligence (ISCMI)*, Dubai, 2016, pp. 138-142, doi: 10.1109/ISCMI.2016.35.

[21] M. Casas, R. Avitia, F. Gonzalez-Navarro, J. Cardenas-Haro, et. al. "Bayesian Classification Models for Premature Ventricular Contraction Detection on ECG Traces." *Journal of healthcare engineering*, *2018*. https://doi.org/10.1155/2018/2694768

[22] J. Rivera, K. Rodriguez and X. Yu, "Cardiovascular Conditions Classification Using Adaptive Neuro- Fuzzy Inference System," *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, New Orleans, LA, USA, 2019, pp. 1-6, doi: 10.1109/FUZZ-IEEE.2019.8858896.

[23] "Electrocardiogram (ECG or EKG)," *Mayo Clinic*, 09-Apr-2020. [Online]. Available: https://www.mayoclinic.org/tests-procedures/ekg/about/pac-20384983.

[24] "How the heart works," *Heart Foundation*. [Online]. Available: https://www.heartfoundation.org.nz/your-heart/how-the-heart-works.

[25] 7 Countries Study, "ECG findings and coronary heart disease - Seven countries study", *Seven Countries Study*, 2012. [Online]. Available: http://www.sevencountriesstudy.com/ecg-predictors-and-coronary-heart-disease/.

[26] B. G. Petty, *Basic electrocardiography*. Cham, Switzerland: Springer, 2020.

[27] G. B. Moody and R. G. Mark, IEEE Engineering in Medicine and Biology, Cambridge, Massachusetts, publication, 2001.

[28] M. Hoglinger, "EKG Preprocessing," thesis, Institute of Signal Processing, 2016.

[29] N. Pilia, A. Loewe, W. Schulze, and O. Dossel, "Comparison of Baseline Wander Removal Techniques considering the Preservation of ST Changes in the Ischemic ECG: A Simulation Study," *Hindawi*, 08-Mar-2017.

[30] R. Kher, “Signal Processing Techniques for Removing Noise from ECG Signals,” Journal of Biomedical Engineering, 15-Feb-2019, pp. 1-9.

[31] S. Poungponsri, “An Approach Based on Wavelet Decomposition and Neural Network for ECG Noise Reduction,” California Polytechnic State University, June-2009.

[32] A. Graps, "An introduction to wavelets," in *IEEE Computational Science and Engineering*, vol. 2, no. 2, pp. 50-61, Summer 1995, doi: 10.1109/99.388960.

[33] C. Valens, “A Really Friendly Guide to Wavelets,” 1999.

[34] S. Mirjalili and S. Mirjalili, *Evolutionary Algorithms and Neural Networks Theory and Applications*. Cham: Springer International Publishing, 2019.

[35] H. Tang, K. C. Tan, and Z. Yi, *Neural networks: computational models and applications*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2010.

[36] T. F. Romdhane, H. Alhichri, R. Ouni, and M. Atri, “Electrocardiogram heartbeat classification based on a deep convolutional neural network and focal loss,” *Computers in Biology and Medicine*, vol. 123, p. 1038, 2020.

[37] V. Mondéjar-Guerra, J. Novo, J. Rouco, M.G. Penedo, M. Ortega, “Heartbeat classification fusing temporal and morphological information of ECGs via ensemble of classifiers,” *Biomedical Signal Processing and Control*, vol. 47, p. 41, 2020.

[38] J. Brownlee, “How to Choose an Activation Function for Deep Learning,” *Machine Learning Mastery*, 21-Jan-2021.

[39] V. Bushaev, “Adam - latest trends in deep learning optimization.,” *Towards Data Science*, 24-Oct-2018.

[40] J. Brownlee, “How to Choose Loss Functions When Training Deep Learning Neural Networks,” *Machine Learning Mastery*, 25-Aug-2020.

[41] S. Minaee, “20 Popular Machine Learning Metrics. Part 1: Classification & Regression Evaluation Metrics,” *Medium*, 28-Oct-2019.

[42] J. Mohajon, “Confusion Matrix for Your Multi-Class Machine Learning Model,” *Medium*, 09-Sep-2020.

[43] “Haar and Daubechies Wavelets via Gabriel Peyr,” *mason.gmu.edu*.

[44] D. A. Boyce, “Sensitivity and Specificity,” in *Orthopaedic Physical Therapy Secrets*, 3rd ed., J. D. Placzek, Ed. Elsevier, 2017, pp. 125–134.

Appendix A: Source Code

```
"""
  Automatically generated by Colaboratory.

  Original file is located at
    https://colab.research.google.com/drive/1PiEF_RmLIeca7Cl_enOzDd9Au6NArEdV


  ECG DIAGNOSIS USING DWT & FNN
  ---
  Cal Poly Senior Project - 2020-2021

  Authors:
    Nathan Diekema
    Hannah Chookaszian

  Last Edited:
    06/08/2021
"""

import os
import matplotlib.pyplot as plt
import tensorflow as tf
import scipy
import scipy.signal
import numpy as np
import pywt
from sklearn.metrics import classification_report, confusion_matrix
import wfdb
import random
from sklearn.model_selection import train_test_split
from imblearn.under_sampling import RandomUnderSampler

plt.rcParams["figure.figsize"] = (20,4)

"""
---
LOAD DATA
---

The data is provided by
https://physionet.org/physiobank/database/html/mitdbdir/mitdbdir.htm
The recordings were digitized at 360 samples per second per channel with 11-
bit resolution over a 10 mV range. Two or more cardiologists independently
annotated each record; disagreements were resolved to obtain the computer-
readable reference annotations for each beat (approximately 110,000
annotations in all) included with the database.
    Code              Description
    N          Normal beat (displayed as . by the PhysioBank ATM)
    L          Left bundle branch block beat
    R          Right bundle branch block beat
    B          Bundle branch block beat (unspecified)
    A          Atrial premature beat
```

```
    a          Aberrated atrial premature beat
    J          Nodal (junctional) premature beat
    S          Supraventricular premature or ectopic beat (atrial or nodal)
    V          Premature ventricular contraction
    r          R-on-T premature ventricular contraction
    F          Fusion of ventricular and normal beat
    e          Atrial escape beat
    j          Nodal (junctional) escape beat
    n          Supraventricular escape beat (atrial or nodal)
    E          Ventricular escape beat
    /          Paced beat
    f          Fusion of paced and normal beat
    Q          Unclassifiable beat
    ?          Beat not classified during learning
"""

wfdb.dl_database('mitdb', 'data') # Load MIT-BIH data

"""
  ==================
    PRE-PROCESSING
  ==================
"""

sampling_rate = 360 # MIT-BIH ECG records are captured at 360Hz

# Declare struct to hold important signal data
class Signal:
  def __init__(self, record, signal, r_peaks, labels):
    self.record = record # Record number
    self.signal = signal # Actual signal data from MIT-BIH
    self.r_peaks = r_peaks # Index of each R-peak in the signal
    self.labels = labels # Annotations for signal class

# These are the classes that we wish to remove in the preprocessing stage
# since they do not refer to a particular beat type.
invalid_labels = ['|', # Isolated QRS-like artifact
                  '~', # Change in signal quality
                  '!', # Ventricular flutter wave
                  '+', # Rhythm change
                  '[', # Start of ventricular flutter/fibrillation
                  ']', # End of ventricular flutter/fibrillation
                  '"', # Comment annotation
                  'x'  # Non-conducted P-wave (blocked APC)
                  ]

# These are the beat types our program will be classifying
valid_classes = ['N', # Normal beat
                 'L', #       Left bundle branch block beat
                 'R', # Right bundle branch block beat
                 'V', # Premature ventricular contraction
                 'A', # Atrial premature beat
                 '/', # Paced Beat
                 ]

def process_record(record):
  """
```

```
    Input: An integer representing one of the ECG records from the MIT-BIH
      database
    Output: A Signal object with the record number, signal data, r_peak data,
      and relevant annotations.
    Description: This function is responsible for the initial preprocessing
        of the ECG signals. It handles the denoising of the input signal by
        applying a butterworth filter (to eliminate baseline wander) and a
        notch filter(to mitigate powerline interference). It also filters out
        any unwanted annotations and adjusts the R-peak locations to ensure
        maximum accuracy.
  """
  # MLII signal only
  raw_signal = wfdb.rdrecord(f'data/{record}', channels = [0]).p_signal[:,0]
  annotation = wfdb.rdann(f'data/{record}', extension='atr')

  r_peaks = annotation.sample
  labels = np.array(annotation.symbol)

  # Eliminate baseline wander
  # Butterworth high-pass filter with an fc=0.5Hz
  b, a = scipy.signal.butter(N=2, Wn=0.5, btype='highpass',
                             analog=False, output='ba', fs=sampling_rate)
  signal = scipy.signal.filtfilt(b, a, raw_signal)

  # Reduce Powerline Interference
  # Notch filter centered at 60 Hz
  b, a = scipy.signal.iirnotch(w0=60.0, Q=30.0, fs=360.0)
  signal = scipy.signal.filtfilt(b, a, signal)

  # Remove unwanted beats
  indices = [i for (i, label) in enumerate(labels) if label not in \
    invalid_labels]
  r_peaks, labels = r_peaks[indices], labels[indices]

  # Align the signals r_peaks to maximize accuracy
  aligned_peaks = []
  for r_peak in r_peaks:
    r_left = np.maximum(r_peak - int(.05 * sampling_rate), 0)
    r_right = np.minimum(r_peak + int(.05 * sampling_rate), len(signal))
    aligned_peaks.append(r_left + np.argmax(signal[r_left:r_right]))

  r_peaks = np.array(aligned_peaks, dtype="int")

  # Normalize signal amplitudes across all patients
  signal = signal / np.mean(signal[r_peaks])

  # Return an object with all relevant signal data & annotations
  return Signal(record, signal, r_peaks, labels)


train_ds = []
test_ds = []

# Optional - Store data in gdrive
# save_path = "/content/drive/ECG_Classifier/preprocessed_mitdb.p"
# pickle.dump((train_ds, test_ds), open(save_path, "wb"))
```

```
train_records = [101, 102, 106, 108, 109, 112, 114, 115, 116, 118, 119, 122,
            124, 201, 203, 205, 207, 208, 209, 215, 220, 223, 230]

test_records = [100, 103, 105, 107, 111, 113, 117, 121, 123, 200, 202, 210,
          212, 213, 214, 219, 221, 222, 228, 231, 232, 233, 234]

for record_num in train_records:
  signal_data = process_record(record_num)
  train_ds.append(signal_data)

for record_num in test_records:
  signal_data = process_record(record_num)
  test_ds.append(signal_data)

# Visualize a waveform
# First 10 seconds of record '105'

view_signal = train_ds[12]
view_len = sampling_rate*10

plt.plot(range(view_len), view_signal.signal[0:view_len])
plt.title('Record {}'.format(view_signal.record))
print('Beat Types:', view_signal.labels[0:10], '\n')

# Entire dataset beat distribution
print('Dataset Beat Distribution:')
total_sum = 0
for vc in valid_classes:
  sum = 0
  for sig_data in train_ds:
    for beat_label in sig_data.labels:
      if beat_label == vc:
        sum += 1
        total_sum += 1
  print(f'  {vc} - {sum}')
print(f'Total: {total_sum}')

"""
========================
  FEATURE EXTRACTION
========================

*Note: RR ~ R-peak interval*

1.   Discrete Wavelet Transform (DWT)
2.   RR Intervals

Note: beats with RR not within a certain window most likely have
  segmentation errors and should be discarded.

"""

# Helper functions for extracting features from DWT
def select_coeffs(coeffs):
  features = []
  for c in coeffs:
    features.append(np.mean(c))
```

```
    features.append(np.std(c))
    features.append(max(c))
    features.append(min(c))

  return np.array(features)


def extract_features(signal_data):
  """
    Input: Dict containing all information from a
    Output: A tuple consisting of a
    Description: This function
  """
  r_peaks = signal_data.r_peaks
  labels = signal_data.labels
  signal = signal_data.signal
  leading_buf, lagging_buf = 90, 110
  valid_classes = ['N', 'L', 'R', 'V', 'A', '/']

  avg_RR = np.mean(np.diff(r_peaks)) # For removal of inter-patient
variability
  wavelets, intervals_RR, classes = [], [], []
  global_RR = 0

  # Cycle through all of the beats in the signal
  # (Skip the first and last beat)
  for i in range(1, len(r_peaks) - 1):

    # Skip all invalid classes of beats
    if labels[i] not in valid_classes:
      continue

    # Normalize the surplus number of normal beats
    if labels[i] == 'N':
      if random.random() > 0.2:
        continue

    # ===== Discrete Wavelet Transform =====
    coeffs = pywt.wavedec(signal[r_peaks[i] - leading_buf : r_peaks[i] + \
      lagging_buf], wavelet='db2', level=3)[0]

    # ======= RR intervals =======
    pre_RR = r_peaks[i] - r_peaks[i - 1]
    post_RR = r_peaks[i + 1] - r_peaks[i]
    local_RR = np.mean(np.diff((r_peaks[max(0,i-10):i + 1])))
    global_RR += pre_RR

    # Check for segmentation errors:
    if local_RR > (2 * sampling_rate) or local_RR < (0.15 < sampling_rate):
      continue

    wavelets.append(coeffs)
    classes.append(labels[i])
    intervals_RR.append([pre_RR - avg_RR, post_RR - avg_RR, local_RR -
avg_RR])

  global_RR = global_RR/len(intervals_RR)
```

```
  for i in range(len(intervals_RR)):
    intervals_RR[i].append(global_RR)

  return (wavelets, intervals_RR, classes)


x1_train, x1_test = [], []
x2_train, x2_test = [], []
y_train, y_test = [], []

# For training set
for signal_data in train_ds:
  wavelets, intervals_RR, classes = extract_features(signal_data)
  x1_train.append(wavelets)
  x2_train.append(intervals_RR)
  y_train.append(classes)

# For testing set
for signal_data in test_ds:
  wavelets, intervals_RR, classes = extract_features(signal_data)
  x1_test.append(wavelets)
  x2_test.append(intervals_RR)
  y_test.append(classes)

x1_test = np.concatenate(x1_test, axis=0).astype('float32')
x2_test = np.concatenate(x2_test, axis=0).astype('float32')
y_test = np.concatenate(y_test, axis=0)
y_test = np.array([valid_classes.index(k) for k in y_test]).astype('int64')

x1_train = np.concatenate(x1_train, axis=0).astype('float32')
x2_train = np.concatenate(x2_train, axis=0).astype('float32')
y_train = np.concatenate(y_train, axis=0)
y_train = np.array([valid_classes.index(k) for k in y_train]).astype('int64')

# Validation set
x1_train, x1_val, x2_train, x2_val, y_train, y_val = train_test_split(
    x1_train,
    x2_train,
    y_train,
    test_size=0.1,
    shuffle=True,
    random_state=42,
    stratify=y_train
  )

"""
  Undersampling
  ---
  Description: Randomly undersample the testing dataset so each class has
exactly
    500 heart beats
"""

# Undersample the testing dataset
def sampling_strategy(n_samples):
    #target_classes_all = y.value_counts().index
    targets = range(len(valid_classes))
```

```
    sampling_strategy = {}
    for target in targets:
        sampling_strategy[target] = n_samples
    return sampling_strategy

undersample = RandomUnderSampler(sampling_strategy=sampling_strategy(500),
                                 random_state = 42)

x1_test, _ = undersample.fit_resample(x1_test, y_test)
x2_test, y_test = undersample.fit_resample(x2_test, y_test)

train_size = len(y_train)
val_size = len(y_val)
test_size = len(y_test)

"""
  Print the size of each dataset
"""

print(f'Size of Training Dataset: {train_size}')
print('Distribution:')
for vc in valid_classes:
  sum = 0
  for beat_label in y_train:
    if valid_classes[beat_label] == vc:
      sum += 1
  print(f'  {vc} - {sum}')

print(f'\nSize of Validation Dataset: {val_size}')
print('Distribution:')
for vc in valid_classes:
  sum = 0
  for beat_label in y_val:
    if valid_classes[beat_label] == vc:
      sum += 1
  print(f'  {vc} - {sum}')

print(f'\nSize of Testing Dataset: {test_size}')
# Training dataset distribution
print('Distribution:')
for vc in valid_classes:
  sum = 0
  for beat_label in y_test:
    if valid_classes[beat_label] == vc:
      sum += 1
  print(f'  {vc} - {sum}')


"""
  =================
      FNN MODEL
  =================
  ---
  Libraries used:
    -Tensorflow
    -Keras
  Description: the keras library makes it much easier to build a readable
```

```
    neural network. Here we have constructed a model that takes in two
    feature sets, merges them together, then passes through 4 dense layers
    of decreasing sizes. Each dense layer uses the relu activation function
and
    is followed by a dropout layer. The output layer uses a signmoid
activation
    function.
"""
import tensorflow as tf
from keras.models import Model
from keras.layers import *

# Default learning rate of the Adam optimizer is 10e-4
learning_rate = 0.001
num_classes = len(valid_classes)

# Input Layer
input1 = Input(x1_train.shape[1])
input2 = Input(x2_train.shape[1])

# Merge Inputs
merged = Concatenate()([input1, input2])

# Dense Layer 1
dense1 = Dense(24, activation='relu')(merged)
dropout1 = Dropout(rate = 0.1)(dense1)

# Dense Layer 2
dense2 = Dense(64, activation='relu')(dropout1)
dropout2 = Dropout(rate = 0.1)(dense2)

# Dense Layer 4
dense3 = Dense(24, activation='relu')(dropout2)
dropout3 = Dropout(rate = 0.1)(dense3)

# Output Layer
output = Dense(num_classes, activation='softmax')(dropout3)

model = Model(inputs=[input1, input2], outputs=output)
model.summary()

model.compile(
    loss = 'categorical_crossentropy',
    optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate),
    metrics = ['accuracy']
  )

# Print the models flow diagram
tf.keras.utils.plot_model(model, to_file='model.png', show_shapes=True)

# Convert y-datasets to one-hot encoded datasets. Required for the
# categorical_crossentropy loss function
y_train = tf.keras.utils.to_categorical(y_train, num_classes)
y_val = tf.keras.utils.to_categorical(y_val, num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes)

"""
```

```
  Training
  ---
  Description: Train the FNN using the training and validation datasets
    defined earlier. Stores relevant training history in memory for later
    analysis.

  Parameters:
    epochs = 750
    batch size = 256
"""

epochs = 750
batch_size = 256

history = model.fit([x1_train, x2_train],
                    np.array(y_train),
                    shuffle=True,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_data = ([x1_val, x2_val], y_val))

"""
  Plotting graphs illustrating the progression of loss and accuracy of the
  model through each epoch
"""

# Plot training loss, validation loss
fig, axs = plt.subplots(1,2,figsize=(16,8))
axs[0].plot(history.history['loss'])
axs[0].plot(history.history['val_loss'])
axs[0].legend(['Training Loss','Validation Loss'])
axs[0].set_title('Loss')
axs[0].set(xlabel='Epoch', ylabel='Loss')

# Plot training accuracy, validation accuracy
axs[1].plot(history.history['accuracy'])
axs[1].plot(history.history['val_accuracy'])
axs[1].legend(['Training Accuracy','Validation Accuracy'])
axs[1].set_title('Accuracy')
axs[1].set(xlabel='Epoch', ylabel='Accuracy')

# Plot training loss, validation loss
fig, axs = plt.subplots(1,2,figsize=(16,8))
axs[0].plot(history.history['loss'])
axs[0].legend(['Training Loss'])
axs[0].set_title('Loss')
axs[0].set(xlabel='Epoch', ylabel='Loss')

# Plot training accuracy, validation accuracy
axs[1].plot(history.history['accuracy'])
axs[1].legend(['Training Accuracy'])
axs[1].set_title('Accuracy')
axs[1].set(xlabel='Epoch', ylabel='Accuracy')

plt.figure(0, figsize=(8, 8))
plt.plot(history.history['loss'])
plt.legend(['Training Loss'])
```

```
plt.title('Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')

plt.figure(1, figsize=(8, 8))
plt.plot(history.history['accuracy'])
plt.legend(['Training Accuracy'])
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')

"""
  =========================
    EVALUATION & RESULTS
  =========================

  Description: Determine and display important performance metrics of the
model
"""

# Print the accuracy of the Training, Validation, and Testing datasets
results = model.evaluate([x1_train, x2_train], y_train, verbose=0)
print('Training Accuracy: %.2f %%'%(results[1]*100))
results = model.evaluate([x1_val, x2_val], y_val, verbose=0)
print('Validation Accuracy: %.2f %%'%(results[1]*100))
results = model.evaluate([x1_test, x2_test], y_test, verbose=0)
print('Testing Accuracy: %.2f %%'%(results[1]*100))
print(results)

"""
  Confusion Matrices
  ---
  Description: Confusion matrices constructed for the Training, Validation,
and
    Testing datasets.
"""
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Print the Confusion Matrix of the Training, Validation, and Testing
datasets

# === Training Dataset ===
y_train_actual = np.argmax(y_train, axis=1)  # Convert back from categorical
y_train_pred = model.predict([x1_train, x2_train])
y_train_pred = np.argmax(y_train_pred, axis=1)
y_train_conf_matrix = confusion_matrix(y_train_actual, y_train_pred)
print('Training Confusion Matrix:\n')
print(y_train_conf_matrix)

# === Validation Dataset ===
y_val_actual = np.argmax(y_val, axis=1)  # Convert back from categorical
array
y_val_pred = model.predict([x1_val, x2_val])
y_val_pred = np.argmax(y_val_pred, axis=1)
y_val_conf_matrix = confusion_matrix(y_val_actual, y_val_pred)
print('\n\nValidation Confusion Matrix:\n')
```

```
print(y_val_conf_matrix)

# === Testing Dataset ===
y_test_actual = np.argmax(y_test, axis=1)
y_test_pred = model.predict([x1_test, x2_test])
y_test_pred = np.argmax(y_test_pred, axis=1)
y_test_conf_matrix = confusion_matrix(y_test_actual, y_test_pred)
print('\n\nTesting Confusion Matrix:\n')
print(y_test_conf_matrix)

"""
  Class Accuracy & Classification Report
  ---
  Description: Class-by-class method for determining accuracy. The following
    will compute the training, validation, and testing accuracy of each beat.
"""

# === Training Dataset ===
print("Training Dataset Accuracies:")
for beat_type in range(len(valid_classes)):
  sum_true, sum_all = 0, 0
  for i in range(len(y_train_pred)):
    if y_train_actual[i] == beat_type:
      sum_all += 1
      if y_train_pred[i] == y_train_actual[i]:
        sum_true += 1
  avg = sum_true / sum_all
  print(f' {valid_classes[beat_type]} - {avg:.4f}')

# === Validation Dataset ===
print("\nValidation Dataset Accuracies:")
for beat_type in range(len(valid_classes)):
  sum_true, sum_all = 0, 0
  for i in range(len(y_val_pred)):
    if y_val_actual[i] == beat_type:
      sum_all += 1
      if y_val_pred[i] == y_val_actual[i]:
        sum_true += 1
  avg = sum_true / sum_all
  print(f' {valid_classes[beat_type]} - {avg:.4f}')

# === Testing Dataset ===
print("\nTesting Dataset Accuracies:")
for beat_type in range(len(valid_classes)):
  sum_true, sum_all = 0, 0
  for i in range(len(y_test_pred)):
    if y_test_actual[i] == beat_type:
      sum_all += 1
      if y_test_pred[i] == y_test_actual[i]:
        sum_true += 1
  avg = sum_true / sum_all
  print(f' {valid_classes[beat_type]} - {avg:.4f}')


"""
  Description: Class-by-class method for determining specificity. The
following
```

```
    will compute the training, validation, and testing specificity of each
beat
"""

# === Training Dataset ===
print("Training Dataset Specificities:")
for beat_type in range(len(valid_classes)):
  row = y_train_conf_matrix[beat_type]
  col = y_train_conf_matrix[:,beat_type]
  FP = np.sum(row[:beat_type]) + np.sum(row[beat_type+1:])
  TN = len(y_train) - np.sum(col) - FP
  specificity = TN / (TN+FP)
  print(f' {valid_classes[beat_type]} - {specificity:.4f}')

# === Validation Dataset ===
print("\nValidation Dataset Specificities:")
for beat_type in range(len(valid_classes)):
  row = y_val_conf_matrix[beat_type]
  col = y_val_conf_matrix[:,beat_type]
  FP = np.sum(row[:beat_type]) + np.sum(row[beat_type+1:])
  TN = len(y_val) - np.sum(col) - FP
  specificity = TN / (TN+FP)
  print(f' {valid_classes[beat_type]} - {specificity:.4f}')

# === Testing Dataset ===
print("\nTesting Dataset Specificities:")
for beat_type in range(len(valid_classes)):
  row = y_test_conf_matrix[beat_type]
  col = y_test_conf_matrix[:,beat_type]
  FP = np.sum(row[:beat_type]) + np.sum(row[beat_type+1:])
  TN = len(y_test) - np.sum(col) - FP
  specificity = TN / (TN+FP)
  print(f' {valid_classes[beat_type]} - {specificity:.4f}')

# Print a classification report for each dataset

print("\n\n\t\t\t===| Training |===\n")
print(classification_report(y_train_actual, y_train_pred,
                            target_names=valid_classes, digits=4))

print("\n\n\t\t\t===| Validation |===\n")
print(classification_report(y_val_actual, y_val_pred,
                            target_names=valid_classes, digits=4))

print("\n\n\t\t\t===| TESTING |===\n")
print(classification_report(y_test_actual, y_test_pred,
                            target_names=valid_classes, digits=4))
```

Appendix B: ABET Senior Project Analysis

1. Summary of Functional Requirements:

The project identifies cardiac conditions in patients from standard 12-lead ECG signals. The ECG records electrical signals from the heart and artificial neural network identify different types of cardiac conditions. This project addresses the need for computer-based detection of cardiac conditions without the need for a trained cardiologist. Our software will accurately diagnose five common cardiovascular conditions: Premature ventricular contraction (PVC), premature atrial contraction (PAC), left bundle branch block (RBBB), right bundle branch block (LBBB), and ventricular tachycardia (VT) [15]. The system will achieve a minimum detection accuracy of 97% for these conditions while maintaining a system response time under 1 minute.

2. Primary Constraints:

This project has a few challenges that are associated with the building and testing of the product.
First, each of the ECG signals from the MIT-BIH database have 12 high-resolution leads associated with them and are each around 30 minutes in length. As a result, training the neural network will require significant processing power and memory from the computer. To resolve this, each signal must be sliced to around 1 minute in length which should theoretically provide enough data for detection of arrhythmias. However, even after reducing the memory usage the training will still take a significant amount of time and processing power, so this needs to be accounted for and worked around. Another potential issue is determining a reliable method for determining the real-world accuracy of the system. To avoid potential biases caused from using a single database for both training and testing, the software should be tested on a number of different datasets. This should not be difficult considering the plethora of reliable, open-source ECG datasets found on the internet that have been designed specifically for testing. We will use a database to test the device, but will not be able to attain new signals because there is no access to an ECG machine.

3. Economic:

Economic Impacts:

I. Human Capital - This project is designed to reduce the amount of human capital required for ECG analysis. The finished product will assist medical professionals in classification of signals and reduce the amount of analysis they need to do. The project team will need an in-depth understanding of neural networks and ECG signals in order to complete the system. This will require time and effort for the project team to learn.

II. Financial Capital - This project will require development done by humans, so labor will be a significant cost. Additionally, some MATLAB toolboxes will need to be purchased for development of the software.

III. Manufactured or Real Capital - The software for this project will need to be developed by humans and this will be the most significant cost for the project. Running the software will also require some processing power from the computer so the project will require computers that are manufactured with significant processing power.

IV. Natural Capital - This software was designed to run on an existing computer and requires a substantial amount of processing power. Based on an average power consumption of a computer being 200 watt-hours, the analysis of a single patient would require approximately 3.4 watts. The natural capital required is from the building of the computer and power used to keep the computer running.

This project is software-based so no money will need to be spent on physical resources. The only additional cost, apart from the software packages listed above, is related to labor. Labor will be a constant cost throughout the 9-month period in which we will be developing our product. A safe estimate for labor costs can be determined based on the average salary of an entry-level software developer which would be approximately $100,000 annual salary or rather $75,000 for a 9-month period. Everything else we are planning on using is either open source or provided free for students. For instance, all the ECG signal databases are free for public-use and every relevant python library is open source. The complete cost of this project from start to finish will be subsidized by the head designers.

Expected Profits:

In terms of expected profits, this project is not necessarily being done with the hopes of selling it on a commercial scale. However, if the software were to be sold commercially, the price of a license would

probably be around $100.00. A single license allows the user to download and operate the software on a single computer. This forces larger institutions such as hospitals to purchase multiple licenses to fulfill demands while keeping the cost low for small clinics and individuals. For a start, we can estimate the potential profits based purely on hospital usage. There are approximately 6,000 hospitals in the United States, so if we assume that 10% adopt our technology with an average of 15 licenses sold per hospital the total profit would be around $900,000. These profits are promising, especially considering how comparatively cheap it was to develop. That being said, we are not focused on profits; Instead, the primary focus is to research a relatively new approach to ECG analysis. The end goal is to determine whether the utilization of neural networks serves as a feasible competitor with the current state of the art methods and whether it has potential of surpassing these current methods in accuracy and efficiency.

Timing:

The development of this product will take approximately 9 months from start to finish. The final iteration of our product is expected to emerge around early June 2021. This release will not be the finished product, it is considered a starting point for research that will be continued later. There will always be ways to improve the algorithm efficiency and accuracy, especially as more research is done in this field. As a result, updates for the software are expected to be released regularly after the initial release as the research is continued by another team of engineers.

4. If manufactured on a commercial basis:

In short, if we were to distribute our product on a commercial basis, we would need to achieve exceptional results. The accuracy of our medical diagnosis would need to be competitive with the current state of the art technology to be marketable. This would require additional research, time and manpower. Seeing as how this product is directly related to the medical field, it is also important to consider how saturated the medical device market is. It would be exceptionally hard to break into, especially with the amount of research and money that circulates around this industry. Hospitals have access to the best cutting-edge technology that boasts higher accuracy rates than we will likely achieve during our research. It would simply not be feasible, or smart, to pursue this product commercially until we have achieved a level of detection accuracy that is competitive with what is currently on the market. It is also important to consider the strict regulations associated with the medical field that are enforced by the FDA. Our product is a software intended for medical use and is consequently considered to be a “Software as a Medical

Device" (SaMD) by the FDA. As a result, the distribution of this product requires the same FDA approval and compliance as a medical device.

5. Environmental

The only environmental resource directly consumed by this product is the power needed to operate the computer it is running on. As stated above, our software is going to require a substantial amount of processing power which corresponds with an increase in energy consumption. If we look at the big picture, this energy will be provided by a variety of different natural resources, both renewable and not. The energy will most likely be produced through a combination of oil, natural gas, coal, solar, hydro, and wind power plants. In the United States 62.6% of the energy produced uses fossil fuels and only 17.6% of energy is renewable [24]. Therefore, the power used to run the software will most likely not be renewable. Moreover, seeing as how our product is a software, customers from around the world will be able to easily access it and use it on their computers. Thus, it may have a greater environmental impact in some regions rather than others based on the primary source of power generation and the number of users in said region. The average power consumption of a high-end desktop computer is around 200 watt-hours whereas the energy consumption of an average U.S. hospital consumes around 31 kilowatt-hours *per square foot*. Even in the unlikely event that there were dozens of computers running this software throughout the entire day in the same hospital, it would still prove to be a trivial fraction of total energy consumption. The resources used to construct each computer must also be considered and factored into the overall environmental impact. Each computer uses large amounts of silicon and other elements that are difficult to recycle. However, most of these computers are not limited to running our software and likely would have been purchased either way so it is difficult to assign blame. Without denying the fact that there *will* be a slight impact, it will almost certainly be imperceptible, even if implemented at a large scale. Similarly, there is not expected to be any noticeable impact on the local ecosystem, or on the habitats of other species.

6. Manufacturability

This product will take the form of a software package at the end of the project. The manufacturing will be completely done by the end of the project and will not need to be repeated because the artificial neural network will already be trained. Instead, the "manufacturing" will consist of the distribution of software licenses. Under United States copyright law, all software is copyright protected and must distribute a form of software license to legally give the user permission to use the associated software. The creation and

distribution of software licenses would not cost the company very much. We would first need to design and host a publicly accessible website. This is definitely the easiest and most user-friendly method for the distribution of the software. The website will be structured in such a way so users can make a profile and can easily purchase a license and download the software. The website will also need professional information about the product, a tutorial on how to operate it, and a way to distribute and alert users of new software updates. For legal purposes, we would need to write a license agreement which may include information related to limitation of liability and warranties.

7. Sustainability

Our product does not directly impact the sustainable use of resources. Seeing as how it is a software accessible from the internet, it does not require the use of any physical resources. Additionally, there is no waste involved with maintenance since software updates can easily be distributed and downloaded through a website. Perhaps the most significant indirect impact is the need for a computer with access to the internet and a power supply. This is considered an indirect impact because the computer will more than likely be used for a vast array of functions throughout its lifetime and will not be purchased with the sole intent of running this one software. It is well-known that computers have a limited lifespan and are likely to be replaced rather frequently. This is especially true in a hospital where having state of the art technology is more important than ever. It should also be considered that computers are very difficult and expensive to recycle. If not recycled properly, computers will remain hazardous in landfills for years and will ultimately have a negative impact on the environment.

As there is more information discovered on ECG signals, the project can be improved. There is an abundance of resources that supply information on ECG analysis and researchers are still continuing to find more information. The software architecture can continue to be improved as there is more and more research and information discovered. The type of neural network can also be improved to make it better at learning. This project will use an artificial neural network, but there are many other types of neural networks which can be trained more deeply. The project can also be improved by decreasing processing power so it is available to more users and this will also decrease the environmental impact.

The biggest challenge with updating the design of this project is with distribution of the update. The software needs to have some way to notify users when an update is available and necessary. With each update, the licensing agreement and website will also need to be updated. This will not take too much time, so updating the software will not be very difficult overall.

8. Ethical

Since this project was designed to work with an electronic device, the user is morally obligated to follow the IEEE ethical code. The IEEE ethical code states that the one must accept responsibility in making decisions consistent with the safety, health and welfare of the public. To ensure this to the best of our ability, the user must be made aware of the potential for a missed diagnosis or misdiagnosis. If this software is used at a hospital the clinician is expected to double check the results of the ECG diagnosis as a fallback in case of an error. Moreover, the patient must be made aware of the very small chance that their diagnosis is wrong. All of this can be ensured by requiring all users to electronically sign a health and safety agreement before use. Furthermore, once this software is released to the public, every modification or citation must be approved by the designers of the product. Additionally, the IEEE code of ethics states that one should uphold integrity, behave responsibly, and conduct ethically. This will be done by giving proper credit to all external sources used and acknowledging everyone that contributed. The IEEE ethics code also states that one must treat all persons fairly and with respect. This is an especially important ideal to consider, especially in terms of healthcare. Unfortunately, it is a challenging feat to provide access to *everybody* in today's world, but this product is designed to follow the utilitarian ethical framework in hopes of helping the vast majority of the population. The ultimate goal of our research is to provide the greatest good for the greatest number of people. It is for this reason, that it will not just be limited to use by healthcare professionals. To make it as accessible as possible to everyone, the software will remain affordable and can easily be downloaded from the internet. Moreover, the interface design will be intuitive and approachable for those who are not medical professionals. This product is fundamentally designed to maximize the greatest good and is ethically sound under the utilitarian framework.

9. Health and Safety

From a physical perspective, the ECG procedure itself is safe and has no known risks. However, some people may be sensitive to the electrodes which can cause a local irritation of the skin. The main concern related to health and safety is perhaps the risks related to the possibility of a missed diagnosis and misdiagnosis. Our software will achieve a diagnosis accuracy of above 97% which is better than most software on the market but evidently there is still room for error. If there *is* an error made it could mean a loss of precious time that could have been used to provide treatment for the patient. In the worst cases, a missed diagnosis may even result in the death of that patient later down the road. It is also important to consider the mental impact that someone may endure if they are diagnosed with a potentially terminal

cardiac condition. After all, heart diseases are responsible for the leading cause of deaths in the United States, accounting for nearly 46% of deaths annually. Finally, it is important to consider issues with private users. This software is meant to be available to any user that is interested, but any issues found in ECG signals should ideally be checked by a qualified physician. That's not to say independent users should not trust the results, they should just be made aware of the potential for errors.

10. Social and Political

The ECG medical diagnosis software will have no direct internal biases. The analyses of the ECG signal and the subsequent medical diagnosis will perform the same duty to the best of its ability despite differences such as race, ethnicity, socioeconomic status, or religion. ECG signals will vary person to person but do not change based on demographics. For this reason, there are no inequalities in performance for anybody that can get treated. However, seeing as how this product is a software, it requires an ECG test to be done prior to the analysis of the signal. This does bring up some indirect inequities related to socioeconomic status. For instance, many people in the United States are not in an economic position to afford health insurance and consequently are not able to receive the treatment they need. This group of people will unfortunately not benefit from our product which requires an ECG test to be done in a clinic or hospital. This is a serious problem that will require a much more complex solution.

The direct stakeholders of this product are doctors that rely on ECG signals to make diagnoses or people who need to receive care from within their own home but cannot afford to purchase an ECG classification device. This project can assist those medical professionals in finding abnormal signals that they may have not noticed before and will expand the accessibility of ECG technology. Patients are also heavily impacted by this project and should be considered indirect stakeholders. Patients are relying on a reliable diagnosis from the machine so that they can get early and correct treatment for different conditions.

In terms of benefits, this technology will undoubtedly have a positive impact on healthcare providers and patients alike. The utilization of neural networks will make the ECG analysis and diagnosis process more efficient and accurate. It will cut down on the risk of missed diagnosis and misdiagnosis, both of which can be dangerous to the patient. It also serves as a more affordable option compared to those currently on the market. The only requirements are a computer and the software itself. This will increase the target market significantly and, more importantly, it will provide an economical option for those who need to be constantly monitored within their own home. Moreover, unlike a physical machine, the software on the computer is easily updated when improvements are made. This small advantage will prove to be a significant economic benefit to customers, especially those who are economically strained. One again,

unfortunately these benefits only apply to the portion of the population that can afford mid-tier computer equipment, or health insurance.

11. Development

This project requires knowledge of complex theories and algorithms related to signal processing and the construction of artificial neural networks. Additionally, we must understand how ECG signals are obtained, processed, and analyzed. Thus far, we have learned about the morphology of a standard ECG signal and how to identify important intervals and points of measurement. Building on that, we have attained an understanding of how to analyze these signals and how clinicians are able to consistently distinguish between normal and abnormal ECG signals. We have also learned about the fundamentals of neural networks and popular training methods. This is the fundamental basis of our project, so it is important we understand the concepts before developing the software itself. In order to learn more, there are python and MATLAB libraries that can assist in signal processing and development and training of the neural network.