Fall 2019

# DETECTING MYOCARDIAL INFARCTIONS USING MACHINE LEARNING METHODS

Aniruddh Mathur
*San Jose State University*

# DETECTING MYOCARDIAL INFARCTIONS USING MACHINE LEARNING METHODS

A Project Report

Presented to

Prof Robert Chun

Department of Computer Science

San José State University

In Partial Fulfilment

Of the Requirements for the Class

CS 298

By

Aniruddh Mathur

December 2019

The Designated Thesis Committee Approves the Thesis Titled

Detecting Myocardial Infarctions using ML Methods

by

Aniruddh Mathur

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

December 2019

| | |
|---|---|
| Dr. Robert Chun | Department of Computer Science |
| Dr. Katerina Potika | Department of Computer Science |
| Mr. Nikhil Lahoti | Software Engineer, Adobe |

## Abstract

Myocardial Infarction (MI), commonly known as a heart attack, occurs when one of the three major blood vessels carrying blood to the heart get blocked, causing the death of myocardial (heart) cells. If not treated immediately, MI may cause cardiac arrest, which can ultimately cause death. Risk factors for MI include diabetes, family history, unhealthy diet and lifestyle. Medical treatments include various types of drugs and surgeries which can prove very expensive for patients due to high healthcare costs. Therefore, it is imperative that MI is diagnosed at the right time. Electrocardiography (ECG) is commonly used to detect MI. ECG is a process in which the electrical signals of the heart are measured by electrodes placed on a patient's limbs and chest to measure heart signals. In recent years, the availability of medical datasets and the invention of wearable devices have opened new possibilities in early detection of this disease. Wearable devices that measure ECG correctly and have built-in machine learning models can potentially save millions of lives the world over. This research explores traditional machine learning techniques such as Support Vector Machines and Decision Trees as well as a new technique, Capsule Neural Network, for MI detection. Even though the new technique achieves remarkable results, its accuracy is less compared to the traditional machine learning techniques used for MI detection.


*Index Terms* – **classification, electrocardiogram, heart disease, machine learning, myocardial infarction, neural networks.**

## ACKNOWLEDGEMENTS

I would like to thank Dr. Robert Chun for his continued support and for the guidance necessary to work on this project. I would also like to thank my other committee members, Dr. Katerina Potika, Mr. Nikhil Lahoti and Dr. Shishir Mathur, for teaching me core skills needed to succeed and for reviewing my work. Finally, I would like to thank my family for their patience and the advice they have given me throughout my life.

TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# I. INTRODUCTION

Myocardial Infarction (MI) is a condition in which death of the heart (myocardial) cells occurs due to inadequate supply of blood oxygen [1]. The symptoms of MI typically include pain in the chest, jaw, arm or chest during exertion or at rest [1]. The pain can also originate in the centre or left of the chest and radiate towards the arm, jaw, back or shoulder. Some patients can experience symptoms such as vomiting, unexplained nausea and persistent shortness of breath. Sometimes patients do not show any of the symptoms listed above. In such a case, MI can be only detected by cardiac imaging or other such studies. Non-detection of MI is a huge problem because once the heart cells die they are not replaced by new cells. A heart impacted by MI works at a reduced capacity which, in turn, exerts extra pressure on the remaining cells. Machine Learning (ML) can help predict MI before it occurs, thereby preventing irreversible damage to the heart.

In the introductory section of this report, we will see how the various medical techniques are used to detect MI. Next, we will see what is ECG and how its readings provide a picture of the heart's health. In the later stages of this report, the research focuses on exploring traditional ML algorithms to predict MI and how CapsNet performs in comparison to them. This research does not explore modifying and improving the core architecture of CapsNet to achieve high accuracy in predicting MI.

## *A. Medical techniques for detecting MI*

The medical techniques to detect MI can be classified into two categories: pathological techniques and imaging techniques. Pathological techniques to detect MI involve the death of myocardial cells due to prolonged restriction in blood supply to heart

tissue. Cell death starts to occur within 15 minutes. Although, it takes up to 6 hours for complete death of the cells to occur. Infarcts or the death of cell tissue are classified by size: microscopic, small, medium and large. Biochemical markers used to detect MI involve the release of various proteins such as myoglobin, cardiac troponins T and I, and creatine kinase. These proteins are released when damage to the heart cells occur and thus their presence can indicate occurrence of MI.

Imaging techniques used for MI detection are echocardiography, radionuclide angiography and Electrocardiography (ECG). Echocardiography is used more commonly because it allows assessment of causes of chest pain due to other non-MI-related causes such as non-restricted blood flow and valvular heart disease, thereby preventing misdiagnosis of MI[1]. ECG is a process in which the heart's electrical pulses are calculated using electrodes attached to a person's chest and limbs[2].

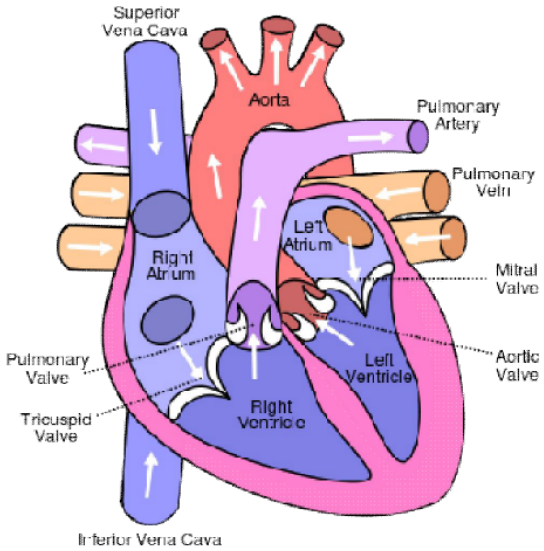


Figure 1. Diagram specifying the structure of the human heart[2]

### *B. ECG: What it is and how it detects heart's rhythm*

The heart muscles have zero charge at rest. During each heartbeat, a heart will progress from a wave of depolarization triggered by sinoatrial muscle tissue present in the right atrium (see

Figure 1). The wave spreads through the atrium, passes through the atrioventricular node (immediately above the right ventricles), and finally spreads across ventricles. This entire process is detected by electrodes placed at both ends of a heart and is represented as a wavy line on a screen or paper (see Figure 2). It indicates the rhythm of the heart and the weaknesses of different parts of a heart. A typical ECG cycle consists of 3 components: P wave, QRS complex and T wave. The P wave represents the atria contraction, the QRS complex measures the depolarization of the ventricles and the T wave measures the re-polarization of the ventricles. The ST segment represents the period between the polarization and de-polarization phase of ventricles. The T wave and the U wave in combination specify the total duration of ventrial recovery (see Figure 2). R-R interval is the time between QRS complex. The instantaneous heart rate can be calculated using the time between two QRS complex (see Figure 2).

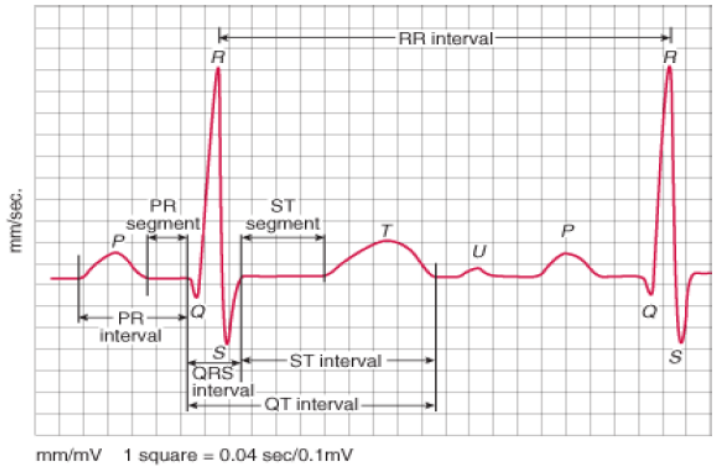


Figure 2. A normal ECG Waveform of a human heart[2]

In the past few years, researchers have started using various ML techniques to detect MI using ECG signals due to a) the advancements in ML algorithms, b) introduction of highly accurate ECG machines, c) proliferation of ECG measuring smart wearable devices and d) the availability of medical datasets. While current ML methods are accurate, they still use

supervised learning techniques to train ML models. Furthermore, since the available datasets are very small (limited to a few hundred patients), it becomes very difficult to identify hidden patterns between measurements which can be overlooked by medical professionals as they just look at the general shape of the various components of the ECG waveform. Just like mathematicians can look at a graph and deduce whether it is a sine wave or a cosine wave, similarly medical professionals look at the shape of the ECG waveform and detect whether the heart is beating normally or does it require medical attention. However, subtle changes in different parts of the ECG waveform, which often get overlooked by medical professions, can be better captured by ML techniques, allowing these techniques to accurately predict MI.

In the next section, we will explore various ML techniques used by researchers to detect MI from ECG signals. Through this exploration, we seek to provide insight into the following research question:

*"Do Capsule Neural Networks increase or decrease the accuracy of MI detection compared to traditional ML algorithms*?" The chart in Figure 3 represents the organizational structure of this report.

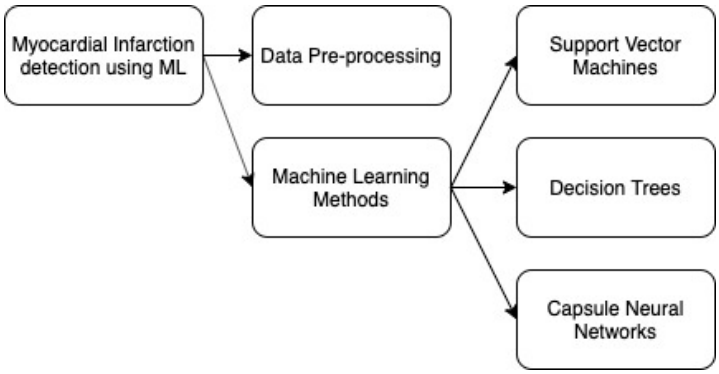


Figure 3. Organizational Structure of the report

## II. TERMINOLOGY

In this section, we will review the various ML approaches that have been used in existing research to detect MI. This review will help us to better understand the techniques and why they were used in the existing research. The existing research will be discussed in the "Related Work" section.

The various ML techniques are as follows:

### *A. Support Vector Machine*

Support Vector Machine (SVM) is a supervised learning model which is used to detect patterns. In summary, in this approach we map the training data points in such a way that a hyper-plane between them clearly divides the data points into 2 categories (see Figure 4). We then run the model on the test data points to check which side of the hyper-plane they lie, thereby giving us a prediction.

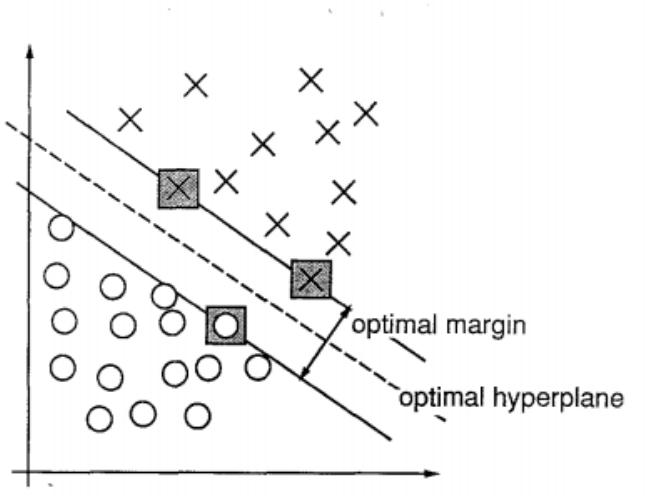


Figure 4. An example of data points represented in a two-dimensional space

The above diagram, as published by Cortex and Vapnik [3], shows an example where data are defined by two separate classes and the support vectors are marked by grey squares

which define the largest margin between the two classes. The hyper-plane is defined by the dashed line between the two classes. This line also separates the two classes with the maximum margin.

SVM is effective in high dimensional spaces. It is also very effective in cases where the number of dimensions is greater than the number of samples in the dataset. The disadvantages of SVM are that in cases where the number of features is greater than the number of samples, extreme care has to be taken to avoid over-fitting by choosing the appropriate kernel functions and regularization parameters. Kernel functions are a way to map one data sequence into another in a 1-to-1 manner. Regularization parameters tell the SVM model how much we want to avoid misclassifying each data point in the training sample.

### *B. Decision Trees*

Decision Trees (DTs) are a supervised learning if-then conditional model for decision making. They support a tree-like structure. They are used in decision analysis to identify a strategy that will most likely help us reach a particular goal. The tree-like structure consists of nodes, branches and leaves. Nodes specify the attribute of the data, branches signify test of each attribute value and leaves are the final decision of class assignment of an instance. We can understand this with the help of an example of whether to play tennis on any given day given a set of attributes [4].

| Outlook | Tempe-rature | Humidity | Wind | Play tennis |
|---|---|---|---|---|
| Sunny | Hot | High | Weak | No |
| Sunny | Hot | High | Strong | No |
| Overcast | Hot | High | Weak | Yes |
| Rain | Mild | High | Weak | Yes |
| Rain | Cool | Normal | Weak | Yes |
| Rain | Cool | Normal | Strong | No |
| Overcast | Cool | Normal | Strong | Yes |
| Sunny | Mild | High | Weak | No |
| Sunny | Cool | Normal | Weak | Yes |
| Rain | Mild | Normal | Weak | Yes |
| Sunny | Mild | Normal | Strong | Yes |
| Overcast | Mild | High | Strong | Yes |
| Overcast | Hot | Normal | Weak | Yes |
| Rain | Mild | High | Strong | No |

Table 1. A table containing different attributes

As Table 1 shows, the final result can depend on multiple attributes. The decision tree generated using Table 1 is provided in Figure 5 [4].

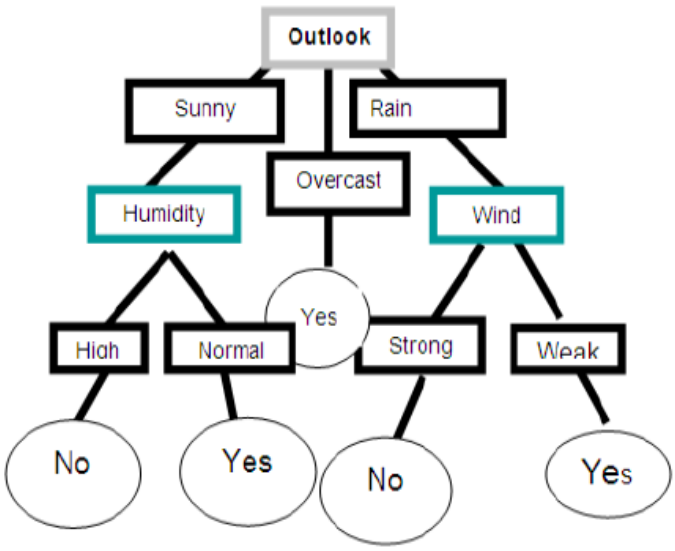


Figure 5. A Decision tree generated for data given in Table 1

As we can see in Table 1, the outcome depends on four attributes. The number of attributes can be changed, with the accuracy of the data classification increasing with the increase in attributes.

There are multiple DT algorithms. The oldest one is the Iterative Dichotomizer (ID3) algorithm which was first proposed by Quinlan [5]. It generates a tree by finding a feature for each node that gives the maximum information gain for categorical targets. After the tree has been generated to its maximum size, it is pruned so that it can generalize efficiently on previously unseen data.

The second algorithm, C4.5, is the next iteration of the ID3 algorithm. It was also proposed by Quinlan [6]. The new improved features of this algorithm are that a) it can accept both continuous and discrete features, b) it has the ability to handle incomplete data points which the ID3 algorithm cannot, c) it uses a bottom-up technique for pruning the decision tree generated and d) it can apply different weights to features that can have a negative effect on the training data.

The third algorithm, CART (Classification and Regression Trees), was first proposed by Brieman in 1984 [7]. Its implementation is very similar to C4.5 with the difference being in how it generates the tree. CART generates a tree by recursively applying a numerical-based splitting criterion on the dataset. The C4.5 constructs the rule sets by an intermediate step instead. The CART algorithm is synonymous with DTs nowadays and is used internally by ML libraries such as scikit-learn [8].

### *C. K-Nearest Neighbours*

The K- Nearest Neighbour Algorithm aims to find the label of an unlabelled example in a dataset by looking at its k- nearest neighbors in the training dataset [9]. The Euclidean distance is calculated between these neighbors. This algorithm is also called a non-generalizing method as it remembers all the data. KNN is an extremely flexible classification scheme as it does not need any pre-processing of data.

### *D. Artificial Neural Network*

Artificial Neural Network (ANN) is inspired from the actual nerve cells found in the human brain. As a result, it can perform extremely complex mathematical computations. Multi-Layer Perception (MLP) is the most popular ANN. We can see MLP's architecture in Figure 6 [10].

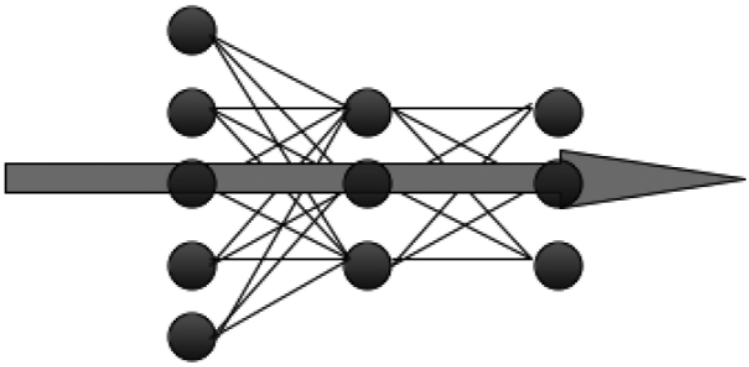

Figure 6. An example of Multi Layer Perceptron with 3 layers

The MLP consists of the following 3 layers: input layer, hidden layer and the output layer. The arrow in Figure 6 shows the direction in which the information flows in the MLP. For this reason, MLPs are also called Feed Forward Networks [10]. MLPs can generate both

classification and prediction models and have the ability to understand complex relationships between input and output variables.

If there are cycles in the MLP then it is called a Recurrent Neural Network (RNN) (see Figure 7) [11].

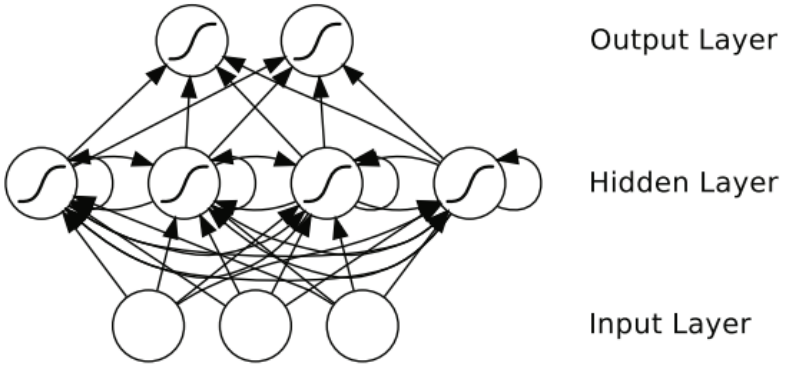


Figure 7. A Recurrent Neural Network(RNN) with cycles in the hidden layer[4]

### *E. Long short-term Memory Neural Network*

A Long short-term Memory (LSTM) Neural Network is a type of RNNs proposed by J. Schmidhuber et. Al [12]. A LSTM cell consists of a cell, an input gate, an output gate and a forget gate. The gradient-based back-propagation method in RNNs have a major disadvantage. The amount of error back-propagated through time depends heavily on the weights. This means that the back-propagation either completely vanishes or blows up. Thus RNNs are not able to learn in cases where there is a time lag greater than ten discrete time steps. LSTM does not have this shortcoming. They can bridge time gaps greater than 1000 discrete time steps by enforcing constant error carousels (CEC) in LSTM cells.

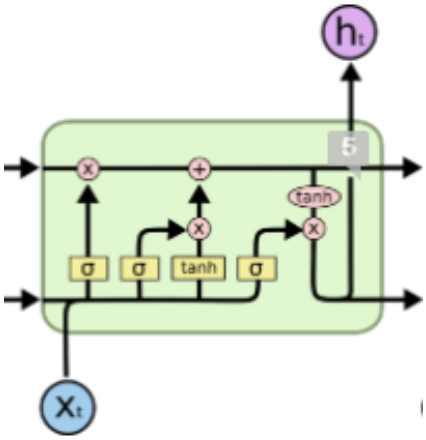


Figure 8. A LSTM cell with a recurrent self-connection [4]

The input gate, the output gate and the forget gate (marked as "X" in Figure 8) interact with the cell and regulate the flow of information to the cell. The cell keeps track of the various dependencies between the different elements of the input sequence. The input gate, which is in the top left corner, controls which information enters the cell; the output gate, to the right, controls what information leaves the cell; and the forget gate controls how long a value remains in the cell. Few of the connections to and from the gates are recurrent and the weights assigned to these connections can have a great influence on how the gates operate while training data.

### *F. Convolutional Neural Network*

A Convolutional Neural Network (CNN) is very similar to a MLP. There are 3 different layers in a CNN: convolutional layer, pooling layer and the fully connected layer. These 3 layers can be stacked repeatedly one after the other as we can see in Figure 9.

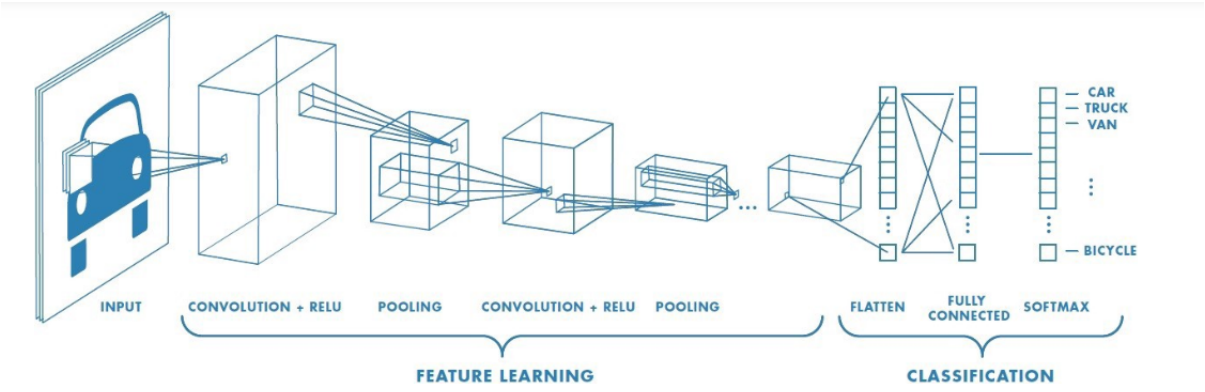


Figure 9. Architecture of CNN with convolution & pooling layer stacked together[13]

Unlike a MLP, a CNN performs very well on real-world image data for two reasons [14]. First, while MLP converts the image into an input matrix and then into a numeric vector, and in that process, does not store spatial information about these numbers, a CNN understands that the pixels that exist close to one another are strongly related compared to the pixels further away. Second, we can include different types of hidden layers in a CNN, which we cannot do in a MLP. Each CNN layer consists of multiple neurons. A neuron is arranged in three dimensions: width, height and depth. Each layer of the CNN transforms its 3D input to a 3D output.

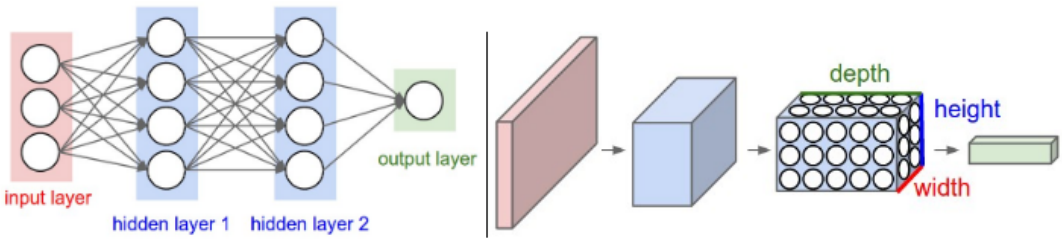


Figure 10. Architecture of regular 3 layer CNN that converts 3D input to 3D output

As we can see in the right half of Figure 10 [15], since the image is held in the red input layer, its width and height are the image dimensions. Moreover, because it is a colored image, it has red, blue and green channels. Therefore, its depth equals three.

### *G. Capsule Neural Network*

Capsule Neural Networks (CapsNet) were proposed by Hilton [16] as a variant to CNNs. The idea is to add “capsules” to existing CNNs. A capsule is a group of neurons whose activity vector represents the instantiation parameters of an object or a part of an object. Active capsules make predictions for instantiation parameters for high level capsules. The capsules in the higher levels are activated only when multiple capsules agree.

Capsules are known to solve the "Picasso problem" in image recognition. If we interchange the position of a person's mouth and nose in an image, a CNN cannot detect this change and will classify the image as that of a person. A CapsNet can identify this change. It can also detect the orientation and the pose of an object in an image. The architecture of a CapsNet is provided in Figure 11 [16].

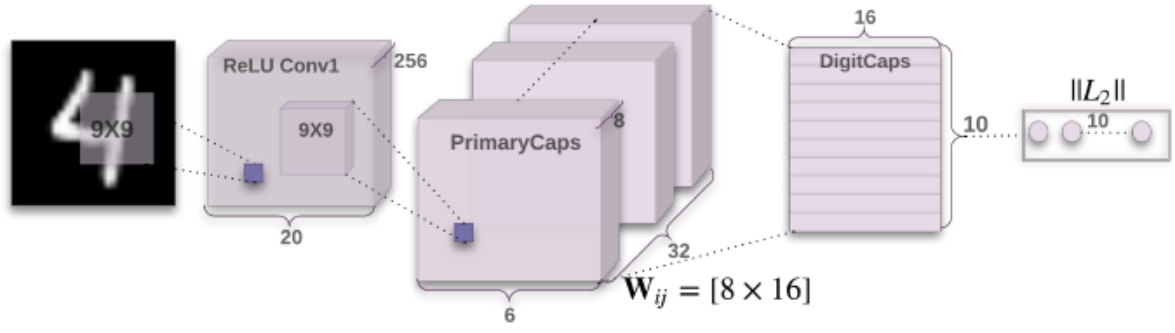


Figure 11. Architecture of CapsNet as proposed by Hilton

The architecture shows two convolutional layers with one connected layer with respect to MNIST Dataset. In Figure 11, the convolutional layer converts all the pixel

intensities to the activities of local feature detectors which are then given as input to primary capsules. The second layer is a convolutional capsule layer. It has 32 channels of convolutional capsules.

CNN connections are setup so that a cluster of pixels in an image feed forward together maintaining special locality data through network topology. Capsnet then adds an extra layer, called “capsules”, which gives it the ability to see if the expected features (eyes, noses, mouth or faces) have the right relationship with each other.

### *H. Principal Component Analysis*

Principal Component Analysis is a statistical technique in which a set of correlated variables are divided into linearly uncorrelated variables called principal components. This can be best explained using Figure 12.

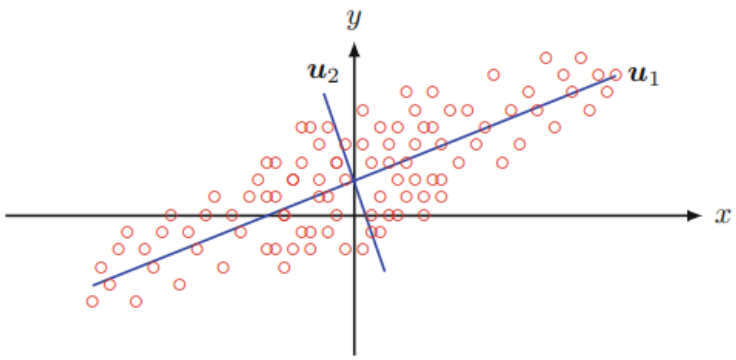


Figure 12. A 2 dimensional dataset with its two principal components

The Figure 12 shows an example [17] where there are n data points in a 2-dimensional space and where u1 and u2 are uncorrelated linear components. The first principal component u1 has the highest variance in the dataset. Principal component u2 has the highest variance for

the constraint that it should be orthogonal to the existing principal component. PCA is mostly used as a tool for data exploratory analysis and is often used in conjunction with ML models to speed them up because it reduces the dimensions of the data.

## II. DATA PREPROCESSING

Before we can run ML models on our ECG signals dataset, it is important that certain changes are made to the data as per each model's requirements. Existing research studies have employed various methods for processing data. This is also known as data pre-processing. Below, several such studies are reviwed.

Bhaskar [18] used the PTB ECG database containing data for 200 subjects. Data were down-sampled, and then Tompkins Algorithm [19] was used for R peak detection. Tomkins algorithm detects QRS in real time and was first implemented in Z80 assembly language. This algorithm detects QRS by using slope, width and amplitude information in an ECG signal. Interference, which is noise in a signal, is reduced by using a band pass filter. This allows the use of low amplitude thresholds so that high detection sensitivity can be achieved. The algorithm periodically adapts each threshold and RR intervals automatically. This way, ECG signals can be accurately used to detect various characteristics, QRS morphologies and heart rate exchanges. Bhaskar [18] then proceeded to use PCA [20] to achieve dimensionality reduction. Thus, he was able to decrease the size of the input features by 17%.

Acharya et al. [21] also used the PTB ECG database. In the database, 12 lead signals were recorded but the researchers used only lead II (lead II is one of the electrodes placed on the patient's limbs). 12 lead signals means placing six leads/electrodes (I,II,III,IV,V,VI) on arms and/or legs and placing the remaining six leads (V1,V2,V3,V4,V5,V6) on the patient's torso. To validate their findings, the researchers split the data into 2 sets. In one dataset, they removed the noise of ECG signals, and on the other they did not remove any noise. Next, they carried out R peak detection on both datasets using the Tompkins Algorithm [19]. All signals

were segmented using R -peak without including the first and the last beat. Finally, each segment was normalized so that it can be fed to the 1-dimensional CNN.

Nasiri [22] used the MIT-BIH database that had 48 records with each record of 30-minute duration. The leads II and IV (placed on one of the limbs) were used for their research. Noise reduction was achieved by wavelet transformation. Feature extraction and reduction of the important features was performed, followed by PCA.

Lui & Chow [23] also used the PTB database but used lead I only. They also used the AF-challenge database. Frequency of the former database was brought down to 300Hz from 1000Hz to match the frequency of the latter database. De-noising was performed by using Savitzky & Golay filter [24]. Savitzky & Golay filter is a digital filter that can be applied to digital points to increase the data's precision without affecting its signal tendency. The filtering is achieved by fitting successive sub-sets of data points that are adjacent and which have a low degree polynomial by using a method of least squares. This whole process is known as convolution. Since their CNN require fixed length inputs, and the dataset had variable length inputs, zero padding was performed so that all heartbeat samples had fixed length. The data were then normalized to improve classification performance.

Tsien et al. [25] used 2 datasets, one collected in Edinburgh Royal Hospital, Scotland and the other collected in Northern General Hospital in Sheffield, England. Both these datasets contained data from 1252 and 500 patients, respectively. There were 45 attributes for each data case. A part of the Edinburgh data was used as a test set, whereas the entire dataset from England was used as a test set.

## III. RELATED WORK

Choosing the appropriate ML approach can have a profound impact on correctly predicting MI. The approach used depends on the type of data available and the type of insights we want to gather. The following subsections describe the three ML approaches used by existing studies for MI detection: SVM, DTs and ANN.

### *A. SVM*

Bhaskar [18] uses 2 classes of input (MI and Healthy Control) to predict the output. He uses LIBSVM library written in C++ for his experiments. The training vectors are represented in a super-dimensional space where the SVM algorithm calculates the hyper parameter most suited for this space. He employs a radial bias kernel function which uses a Gaussian parameter. Since the effectiveness of this approach depends upon the penalty and the Gaussian parameter, a grid search is used to calculate them with exponentially growing sequences. After obtaining the best possible combination of these 2 parameters, 2-fold cross validation is performed to train and test the input data to remove any discrepancies.

Another approach, used by Nasiri et al. [22], is to implement SVM with linear and polynomial kernels along a genetic algorithm. The genetic algorithm improves accuracy through feature reduction and feature weighting. The SVM classification with genetic setup works by first generating a random population size of 50 and then running multiple SVM classifiers on it. Then multi-class SVM is used to detect the fitness of the subset of features. Next, individuals are selected from the dataset based on the fitness value of the feature subset and old individuals are generated from the new ones. If generation of old values from new ones is achieved in two iterations, the researchers would go back to running multiple

SVM classifiers on random population size. Then, they select a subset of features that have the best fitness value. Finally, they use the trained SVM to classify the ECG signals.

The result of the research done by Bhaskar [18] shows that the accuracy of the SVM approach is 91.07% during validation and 90.17% for detail coefficient. It has to be noted that this accuracy is achieved on a dataset that consists of 224 records in the training set, 113 records in the testing set and 112 records in the validation set. For comparison sake, Nasiri et al. [22] accuracy is 93.46% for linear kernel and 82.31 for polynomial kernel. Based on these results, we see that linear kernel with genetic SVM performs much better because it is able to perform feature selection in a way that removes those features that act as noise and may be detrimental to the outcome. It is important to note that the approach used by Nasiri et al. [22] does not calculate the weight of each selected feature.

***B. DTs***

C. Tsien et al. [25] use multiple DTs, namely: a) Final tree (FT), b) Goldman tree using Goldman Algorithm which states the protocol to follow for patients suffering from chest pains in the Emergency Room (ER) and c) Long Tree. Each tree uses different attributes to decide the flow. The correctly classified cases out of the total number of cases are listed for each leaf in the FT. The different variables used for tree building are: a) minimum number of cases in at least two branches of a node, b) confidence level and c) the number of trees built using partitioning of a given training set. These approaches result in a variety of accuracies. For FT, Goldman Tree and Long Tree the accuracies are 89.9%, 73.1% and 80.1%, respectively. FT seems quite favourable for this type of problem because a) it is smaller to implement than the other two trees and b) is clinically reasonable.

### *C. ANN*

The ANN used by Bhaskar [18] has randomly assigned weights initially between -1 and 1 to the inputs of a neuron. Back-propagation is used to decrease the error until ANN can realize the data. The output is a sigmoid function along with an error function. Gradient descent is also used to adjust the weights after back-propagation computes how the error varies on the output, inputs and weights. The number of hidden layers in this ANN is between 2 and 3 and the number of nodes vary from 30 to 60.

Acharya et al. [21] use a CNN that has 4 stages: One Convoluted layer does most of the computational heavy lifting. This layer extracts the features from the input signals. Convolution layers are arranged with 11 layers of feature maps. The second stage is a rectified linear activation function which is used to map non-linearity of the data. Leaky rectifier linear unit (Leaky ReLU) is used as an activation function for multiple layers. In addition to this, a softmax function is also used in layer 11. The third stage is pooling, which is essentially down sampling. The main aim is to reduce the features and computational complexity of the network. Max pooling is used in this stage to output the maximum number in each kernel. This helps reduce the feature map size. The final stage is the Full connected layer. This is a softmax layer with output as an X dimension vector where X is the number of classes desired. X is set to 2 in Acharya et al's [21] research because there are 2 classes (normal and MI ECG signals). Back-propagation is used in the training phase. The regularization, momentum and learning rate parameters are set to obtain optimum performance. Regularization prevents over-fitting of data, momentum controls how fast or slow the CNN learns during training and learning rate helps in the convergence of data. 60 epochs of training and testing are used for testing. 10-fold cross validation is employed to validate the performance of the system. ECG data are split into 10 segments. Nine segments

are used for training and one for testing. This approach is used in 10 iterations by moving the test data. The performance of the 10 iterations is recorded and averaged out.

Lui & Chow [23] use both CNN and a type of Recurrent Neural Network (RNN) called long short-term memory (LSTM). RNNs are a type of ANN where the network has recurrent connections. The recurrent node is activated by taking in a feedback from one-time step to another. Each node of LSTM comprises input, output and forest gates which control the manner in which the internal states are retained or discarded thus giving the LSTM the ability to retain and process information over multiple time-steps. The CNN used in this research had 35 layers in total. It consists of 4 convolutional blocks and three fully connected layers. All the convolutional layers use a rectified linear activation function. The final record classification is decided using majority voting in accordance to the classifications of an ECG record's constituent heart segments. For the next architecture, layer 29 of the previous CNN is replaced with an LSTM to create a CNN-LSTM classifier. All the layers up to layer 29 are wrapped in a time distributed function, aggregating the output vector for each heartbeat time-step before their output is passed on to the LSTM layer. Apart from this, back-propagation is also used to ensure the accuracy of these two classifiers.

Bhaskar [18] computes the accuracy of the results ensuring sensitivity, specificity and overall accuracy. When the number of hidden layers equals 2, then the accuracies for approximate and detailed coefficients are 81.2% and 75.8 %, respectively. When the hidden layers equal 3, the accuracies for approximate and detailed coefficients are 82.14% and 75.44%, respectively. Clearly, adding the number of hidden layers does not increase accuracy as expected. This lack of increase in accuracy is indicative of the presence of noise in the data.

The results obtained by Acharya et al. [21] are more promising. They also calculate sensitivity, specificity and overall accuracy. Their accuracy is 93.53% with noise in the data

and 95.22% without. Clearly, removing noise helps to increase the accuracy but the gain is not much after data pre-processing.

The proposed CNN-LSTM by Lui & Chow [23] achieves sensitivity and specificity of 92.4% and 97.7%, respectively. This can be attributed to the stacker decoder technique. The highlight of this research is that it can be used to classify other cardio vascular diseases as well as MI due to the fact that the dataset is split into multiple classes instead of the usual 2 classes: MI and healthy.

## CONCLUSION OF EXISTING APPROACHES

From the different ML models described in the previous sections, we see that data pre-processing is an important step to achieve a high degree of accuracy in a training model. Heterogeneous data also play a very important role in the accuracy of classifiers. As we see in the techniques above, ML classifiers used with pre-processed data performed much better than ML classifiers without pre-processed data. This situation can be changed by using powerful neural networks. The downside of this approach is that these ANN require a lot of computation hardware as seen in Lui & Chow [6]. Hopefully, this won't be a problem going forward because the cost of hardware is expected to come down with the advancement of technology, thereby enabling the usage of everyday devices. Furthermore, the existing research uses very small datasets, so the success of ML approaches in large datasets needs to be tested.

# IV. RESEARCH METHODOLOGY

## *A. Research Objective*

A review of the literature shows that existing approaches have used CNN networks to achieve high accuracy. However, we have not have yet examined how CapsNet performs on medical dataset. In this research, I first use existing ML models on the dataset to establish a baseline. Next, I train Capsnet on the dataset to check for accuracy. Since CapsNet groups neurons together, fewer parameters can exist between their various layers, thereby needing less parameters in the dataset to train upon. The existing implementations of CapsNet use image datasets to obtain high levels of accuracy, which in some cases are greater than those obtained from existing neural networks. The aim is to examine whether Capsnet performs equally good on numeric data. Since CapsNet has never been trained on ECG datasets, I hope to achieve accuracy greater than existing implementations.

## B. Experimental Setup

I used 2 datasets. The first dataset is the PTB Diagnostic ECG Database [27] with labelled records as a data source. The second dataset is the MIT-BIH Arrhythmia dataset [28]. The PTB Diagnostic Dataset consists of records from 290 subjects: 148 diagnosed with MI, 52 healthy control and the remaining 7 diagnosed with different diseases. Each record consists of ECG signals from 12 leads sampled at 1000Hz frequency. In this project, I used readings from only ECG lead II and only 2 categories—MI and healthy control—were used.

The second—MIT-BIH Arrhythmia—dataset contains the ECG recording of 47 different subjects that have sampling rate of 360Hz. Each beat was annotated by 2 cardiologists. These annotations were used to create five different beat categories in

accordance with Association for the Advancement of Medical Instrumentation (AAMI) EC57 standard [30]. Table 2 provides beat annotations.

| ***Category*** | ***Annotations*** |
|---|---|
| | |
| ***N*** | • Normal<br>• Left/Right bundle branch block<br>• Arterial Escape<br>• Nodal Escape |
| ***S*** | • Atrial Premature<br>• Aberrant Atrial Premature<br>• Nodal Premature<br>• Supra Ventricular premature |
| ***V*** | • Premature ventricular contraction<br>• Ventricular escape |
| ***F*** | • Fusion of Ventricular and normal |
| ***Q*** | • Paced<br>• Fusion of paced and normal<br>• Unclassifiable |

Table 2. Summary of mapping between beat annotations and AAMI EC57 categories[30]

In all my experiments I have used ECG lead II resampled to the sampling frequency of 125Hz as the input. The MIT-BIH Dataset is only used to test the performance of the CapsNet Network.

### *C. Implementation Details*

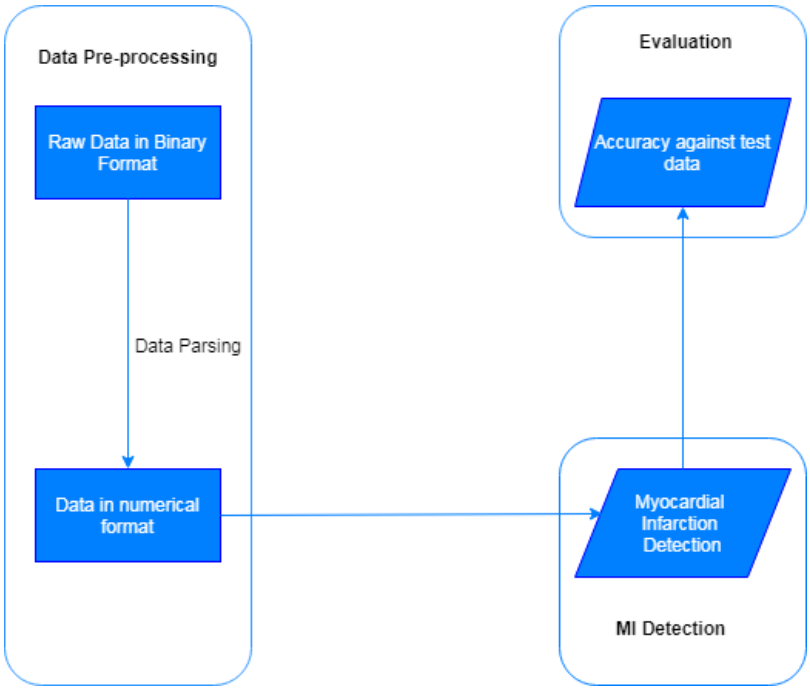


Figure 13. Experiment Implementation Flow

In the experiments, Keras neural network library was used for model training and evaluation. The code was run on Google Cloud Platform (GCP) that had the following hardware specifications:

- 8 CPUs
- 30 GB RAM
- Keras
- Python3
- IPython Notebook

### *D. Data Pre-processing*

The dataset I used was pre-processed as per the methodology implemented by Kachuee et. al [28] to extract beats from an ECG signal. The various steps are as follows:

1) First, the continuous ECG signal was split into 10-second windows. Then, a 10-second window was selected from an ECG signal.

2) Next, the amplitude values of each 10-second window were normalized to the range of values between zero and one.

3) Then, the set of all local maximums were found based on zero crossings of the first derivative.

4) The set comprising ECG R-peak candidates was found by applying a threshold of 0.9 on the normalized value of all the local maximums.

5) The median of R-R time intervals was computed as the nominal heartbeat period of that 10-second window (T).

6) For each R-peak, a signal part of length equal to 1.2T was selected.

7) Finally, each selected part was padded with zeros so that its length equalled a predefined fixed length of 187 columns.

The PTB and the MIT-BIH datasets had 188 features each. The PTB dataset had 14,000 rows and the MIT-BIH dataset had 100,000 rows. The last column in each row contains the labels for each record.

### *E. Evaluation Metrics*

Different types of ML algorithms use different metrics to evaluate the performance of a ML model. Accuracy, precision and recall are usually used as metrics for classification problems. Their definitions are provided below [29].

- **Accuracy:**

  Any ML model's accuracy is defined as the number of times it correctly predicts the class for those inputs for which it did not see the labels out of the total number of inputs provided.

$$\text{Accuracy} = \frac{(Number\ of\ Correct\ Predictions)}{(Total\ number\ of\ Input\ Samples)}$$

- **Precision:**

  Precision is the ratio of actual positive values over all the predicted positive values, that is, it is the ratio of True Positives over the number of True Positives plus the False Positives

$$\text{Precision} = \frac{TP}{TP+FP}$$

- **Recall:**

  Recall is the ratio of all the True Positives over the True Positives plus the False Negatives predicted by the ML model.

$$\text{Recall} = \frac{TP}{TP+FN}$$

## V. EXPERIMENTS AND ANALYSIS

In this section, I first used traditional ML models such as SVM, K-NN and ANN on the PTB dataset to establish a baseline accuracy, precision and recall. Next, I used CapsNet on the same dataset in order to compare the results of CapsNet with the traditional ML models. Finally, I used CapsNet on the second dataset, MIT-BIH, to validate the findings obtained from the PTB dataset.

### *A. Support Vector Machines (SVM)*

Different parameters for the ML model were tuned to obtain the highest level of accuracy. After training a ML model in SVM on the PTB dataset, I obtained highest accuracy when I used the polynomial kernel with C and gamma values of 2 and 4, respectively. Using these parameters the following values were obtained.

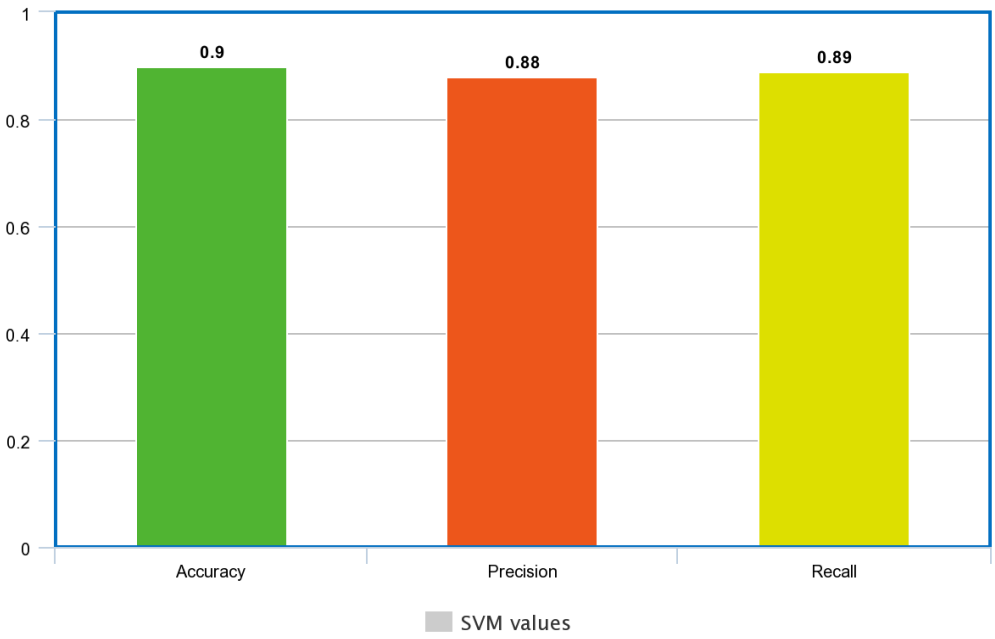


Figure 14. Accuracy, precision and recall values for SVM on PTB ECG dataset

Since SVM is the baseline algorithm, I also trained the model on MIT-BIH dataset to see what accuracy, precision and recall scores I would get. For training a ML model for multiclass classification in SVM, I kept the C and gamma values of the kernel the same as they were in the previous experiment. I generated the following values using these parameters (see Figure 15).

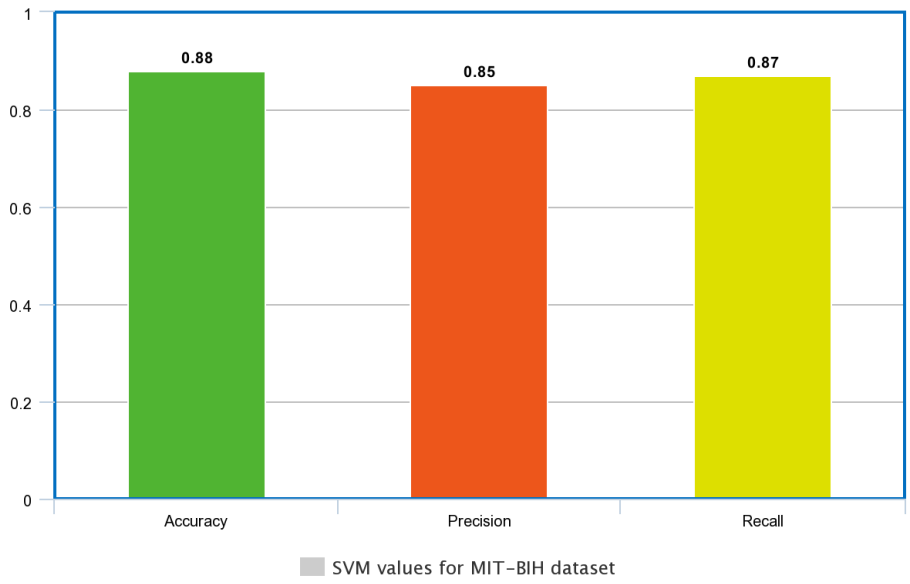


Figure 15. Accuracy, precision and recall values for SVM on MIT-BIH ECG dataset

The review of Figures 14 and 15 shows that the accuracy for MIT-BIH dataset is similar to that for the PTB ECG dataset, thereby confirming that a) the baseline algorithm is working correctly and b) we should expect similar results when we run CapsNet on these two datasets.

### *B. Artificial Neural Networks (ANN)*

For training a ML model in ANN, I used a sequential model to build the network. I also used Dense library to build the input, hidden and output layers. The network had 2 hidden layers and each hidden layer had 4 nodes. As this is a classification problem, I used ReLU as the activation function for the hidden layers and used sigmoid as an activation function for

the last layer. I used Adaptive moment estimation (Adam) to optimize the neural network. Adam is a combination of RMSProp + Momentum.

I used 10 samples per gradient update, and iterated over 100 epochs to train the model. Using these parameters, I obtained the following values for accuracy, precision and recall.

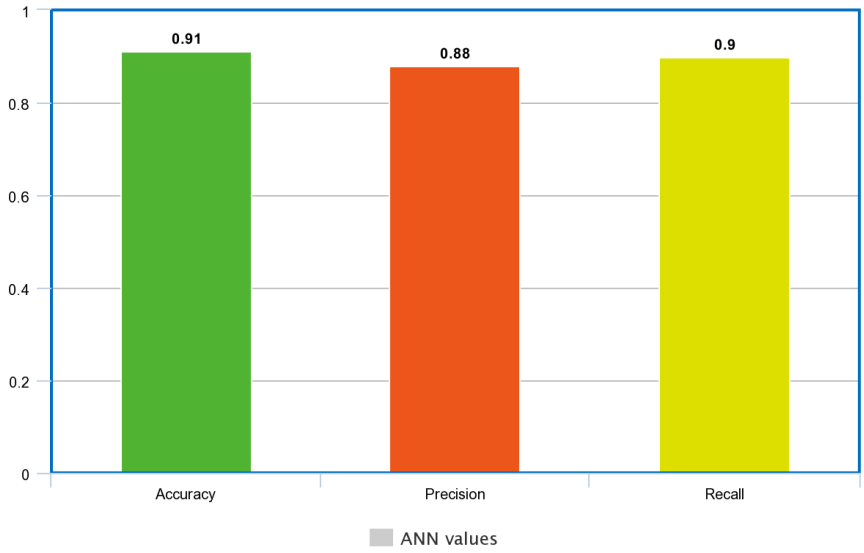


Figure 16. Accuracy, precision and recall values for ANN.

Since ANN is also one of the baseline algorithm, I also trained the model on MIT-BIH dataset to see what accuracy, precision and recall scores I would get. For training a ML model for multiclass classification in ANN, the number of layers was kept constant as the last experiment. The activation function was changed from sigmoid to Softmax as this is a multiclass classification problem. Adaptive moment estimation (Adam) was also used to optimize this neural network. Sparse Categorical Cross-Entropy was used as a loss function and Sparse Categorical Cross-Accuracy metric used in this network as it performs multiclass classification.

I used 10 samples per gradient update, and iterated over 100 epochs to train the model. Using same parameters when running ANN on PTB dataset, I obtained the following values for accuracy, precision and recall (see Figure 17).

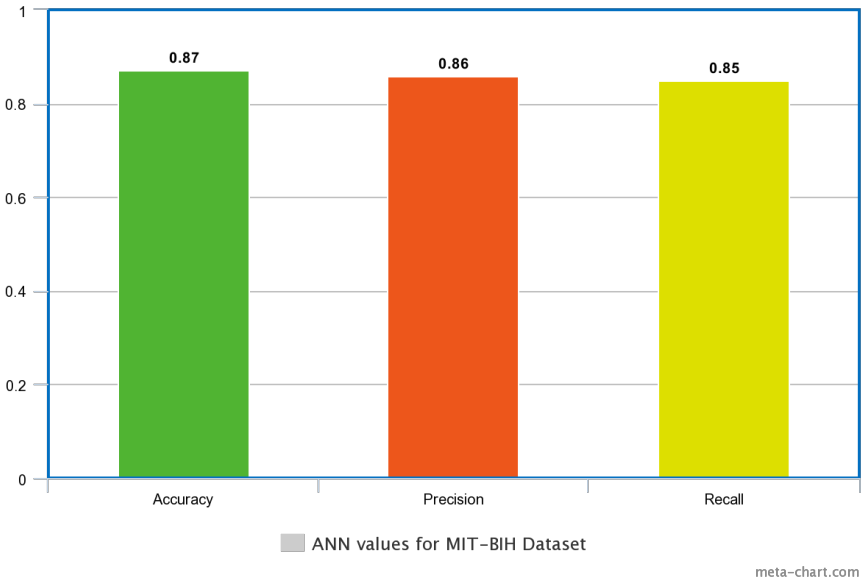


Figure 17. Accuracy, precision and recall values for ANN on MIT-BIH Dataset.

The review of Figures 16 and 17 shows that the accuracy for MIT-BIH dataset is slightly lower to that for the PTB ECG dataset. This can be attributed to the fact that in the MIT-BIH dataset, a neural network having the same layers and parameters has to predict multiple classes as compared to PTB Dataset. And it is a well-known fact that multiclass classification is more difficult than binary classification. Even though the accuracy of ANN on MIT-BIH dataset is slightly lower, it also confirms the following 2 points that a) the baseline algorithm is working correctly and b) we should expect similar results when we run CapsNet on these two datasets.

### *C. Convolutional Neural Networks (CNN)*

For training a ML model in CNN, I used a sequential model to build the network to compare the results with ANN. The network had 1 convolutional layer, 1 max pooling layer, 1 flattening layer and 2 dense layers. As this is a classification problem, I used ReLU as the activation function for the convolutional layers and used sigmoid as an activation function for the last dense layer. Adaptive moment estimation (Adam) was also used in this neural network to optimize it.

The CNN was trained on the PTB ECG dataset for multiple epoch values ranging from 10 to 100 epochs. Using these, I obtained the following values for accuracy, precision and recall.

| Epochs | Accuracy | Precision | Recall |
|---|---|---|---|
| 10 | 0.51 | 0.50 | 0.50 |
| 25 | 0.57 | 0.57 | 0.53 |
| 50 | 0.68 | 0.67 | 0.68 |
| 75 | 0.74 | 0.73 | 0.74 |
| **100** | **0.88** | **0.87** | **0.86** |

Table 3. CNN results for ECG PTB dataset

We can see that as the number of epochs increased, the accuracy also increased. But the accuracy of CNN model was slightly less as compared to ANN model.

### *D. K-Nearest Neighbours*

Since I had a labelled dataset, in the K nearest neighbours algorithm, I limited the values of from 10 to 50. The results are provided in Table 4.

| K | Accuracy | Precision | Recall |
|---|---|---|---|
| **10** | **0.88** | **0.86** | **0.87** |
| 20 | 0.84 | 0.83 | 0.84 |
| 30 | 0.83 | 0.81 | 0.82 |
| 40 | 0.81 | 0.78 | 0.79 |
| 50 | 0.81 | 0.78 | 0.79 |

Table 4. KNN results for ECG PTB dataset

As the value of K increased, the accuracy of the model decreased. The accuracy remained constant if the value of K was more than 50. For K value less than 10, the accuracy was less than 80%; hence, these values are not included in the results.

### *E. K-Nearest Neighbours with PCA*

As we can see from the results above, the accuracy that we obtained from KNN is less than what we obtained from SVM and ANN. To boost the accuracy, PCA was used to reduce the dimensionality of the data and then KNN was executed on it. Since the highest accuracy was obtained with K=10 in the previous experiment, the value of K was kept constant and the number of dimensions in PCA was changed to observe which value led to the highest accuracy. The results are provided in Table 5.

| PCA Components | Accuracy | Precision | Recall |
|---|---|---|---|
| 10 | 0.75 | 0.72 | 0.74 |
| 20 | 0.81 | 0.79 | 0.79 |
| 30 | 0.84 | 0.81 | 0.83 |
| 40 | 0.88 | 0.85 | 0.86 |
| **50** | **0.90** | **0.89** | **0.90** |

Table 5. Results for KNN with PCA when K=10

The accuracy of the model increased with an increase in the number of components. The accuracy of the model remained constant after 50; hence, these values have not been included in the results.

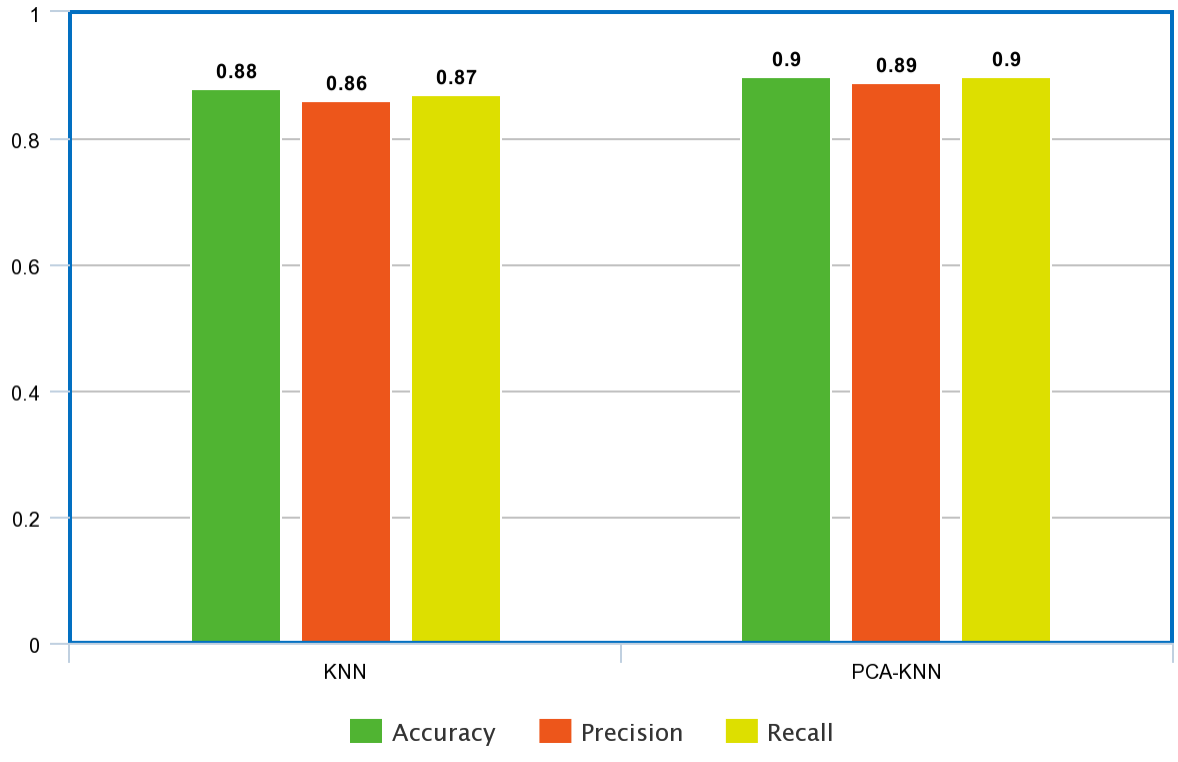


Figure 18. Results for KNN and KNN with PCA

Since I was able to successfully use PCA to increase the accuracy, precision and recall of KNN algorithm, we can hope that it will have a similar effect on the accuracy when used on other ML models.

### *F. CapsNet*

CapsNet can better model hierarchical relationships in a dataset as compared to the other neural networks like CNN. This can prove useful even though CapsNet takes more time to train as compared to CNN. For our experiment, I used a CapsNet that had 6 layers. The last three layers form the decoder network. The first layer is a 1-dimensional convolution layer that had a ReLU activation function. The 2nd and 3rd layers were 2-dimensional convolutional layer and capsule layer, respectively The 3 layers inside the decoder network are fully connected layers. The summary of the model is provided in Figure 19.

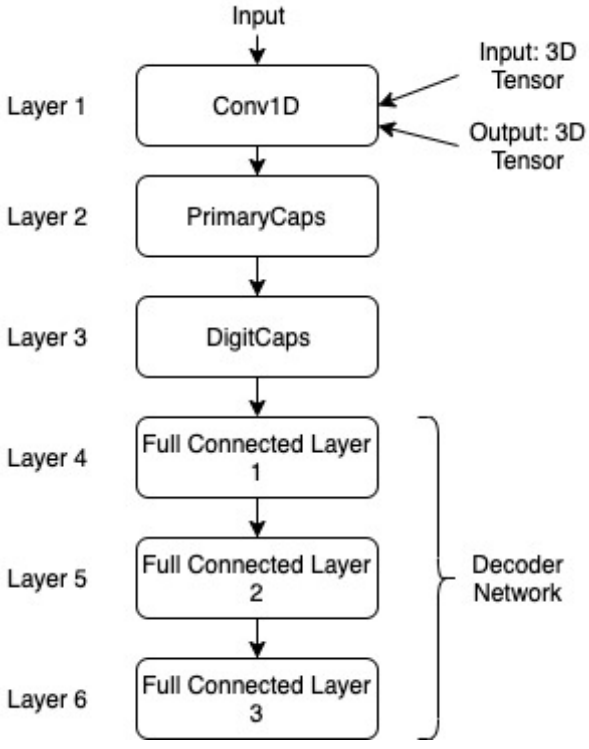


Figure 19. Architecture of CapsNet Model Used

I trained the model from 10 to 100 epochs for effective training. The accuracy remains constant after 100 epoch. Therefore, I have not included those epochs in the results.

| Epochs | Accuracy | Precision | Recall |
|---|---|---|---|
| 10 | 0.63 | 0.61 | 0.62 |
| 25 | 0.65 | 0.63 | 0.63 |
| 50 | 0.69 | 0.67 | 0.68 |
| 75 | 0.72 | 0.70 | 0.71 |
| **100** | **0.73** | **0.72** | **0.72** |

Table 6. Results for CapsNet with different epochs on PTB ECG dataset

The accuracy of the CapsNet model is less as compared to the previous approaches I tried. To ensure that this low accuracy is due to CapsNet and not due to the PTB dataset, I ran CapsNet on MIT-BIH Arrhythmia dataset to cross-check the values for accuracy, precision and recall. I trained the model on the same set of epochs to effectively compare the results. The results of running CapsNet on MIT-BIH Arrhythmia dataset are provided in Table 7.

| Epochs | Accuracy | Precision | Recall |
|---|---|---|---|
| 10 | 0.61 | 0.60 | 0.60 |
| 25 | 0.64 | 0.62 | 0.63 |
| 50 | 0.68 | 0.67 | 0.68 |
| 75 | 0.70 | 0.69 | 0.70 |
| **100** | **0.72** | **0.71** | **0.72** |

Table 7. Results for CapsNet with different epochs on MIT-BIH ECG dataset

Since I observed that the accuracy did not improve after 100 epochs, I kept it constant and tried altering the other parameters. First, I altered the kernel size which is the size of the convolution filter in Layer 1. It is always a square matrix; for example kernel size 4 means a matrix of size 2x2, kernel size 9 means matrix of size 3x3, and so on. I obtained the following results for different kernel sizes (see Table 8).

| Epochs | Kernel Size | Accuracy | Precision | Recall |
|---|---|---|---|---|
| 100 | 4 | 0.72 | 0.72 | 0.71 |
| 100 | 9 | 0.73 | 0.72 | 0.72 |
| 100 | 16 | 0.73 | 0.71 | 0.72 |
| **100** | **32** | **0.73** | **0.72** | **0.72** |

Table 8. Results for CapsNet with different kernel size on PTB ECG dataset

Table 7 shows that different kernel sizes do not have any effect on the accuracy of the model. Usually, altering the kernel size on image datasets increases the accuracy but since this dataset has numerical values, the accuracy does not change because there is not much information gain when the kernel size changes.

Next, I tried changing the values of strides in the convolutional layer of the CapsNet. Stride is the number of pixels a kernel jumps when it looks at the next set of data. I obtained the following results when I changed the number of stride (see Table 9).

| Epochs | Strides | Accuracy | Precision | Recall |
|---|---|---|---|---|
| 100 | 1 | 0.73 | 0.72 | 0.71 |
| 100 | 2 | 0.71 | 0.70 | 0.72 |
| 100 | 3 | 0.69 | 0.69 | 0.68 |
| 100 | 4 | 0.65 | 0.64 | 0.65 |
| **100** | **5** | **0.60** | **0.60** | **0.60** |

Table 9. Results for CapsNet with different strides on PTB ECG dataset

Table 9 shows that as the number of strides increases, the accuracy decreases instead of increasing. This relationship between strides and accuracy can be attributed to the fact that information loss occurs when the stride-size increases. Between the pixels which are read by the kernel, there are a lot of pixels which are not being read; hence the information loss.

Next, I tried modifying the input convolutional layer from 1-dimensional to 2-dimensional. This modification is done so that we can run numerical data on the CapsNet which is designed for image datasets. Since the PTB ECG dataset is in numerical format, to make it compatible with the new Capsnet, we have to reshape the data so that we can provide it as the input to the new Capsnet. The data need to be reshaped because modifying the underlying architecture of CapsNet is beyond the scope of this project. The architecture of the new CapsNet can be seen in Figure 20.

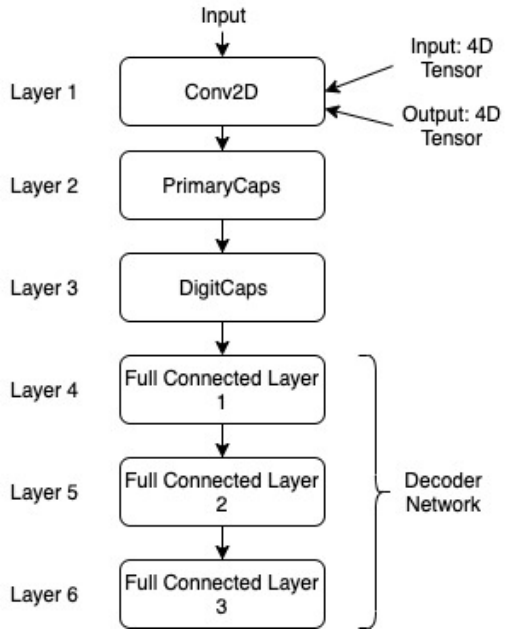


Figure 20. Architecture of CapsNet Model Using 2 dimensional convolutional layer

The kernel size for the 2-dimensional convolutional layer is 1 and the number of strides is 1. Even after running the model for 100 epochs the accuracy was 0%. Hence, the results have not been included in this report.

### *G. CapsNet with PCA*

Because the accuracy of CapsNet model was less compared to the other models, I tried different techniques to improve the accuracy of the model. Since the accuracy of KNN model increased by using PCA, I hoped to get similar results when using PCA to reduce the dimensions of the PTB ECG dataset before I trained CapsNet on it. I kept the structure of the Neural Network constant for effective comparison of results. Table 10 provides the results obtained after varying components' values.

| PCA Components | Accuracy | Precision | Recall |
|---|---|---|---|
| 10 | 0.72 | 0.70 | 0.71 |
| 20 | 0.72 | 0.71 | 0.72 |
| 30 | 0.72 | 0.71 | 0.72 |
| 40 | 0.73 | 0.72 | 0.73 |
| **50** | **0.73** | **0.73** | **0.72** |

Table 10. Results for CapsNet with different PCA components on PTB ECG Dataset.

I kept the number of epochs constant at 100 for different PCA component values so that the model had enough iterations over the data to collect all the meaningful relationships. The accuracy was constant for PCA value greater than 50. Therefore, those results were not included. Each experiment with 100 epochs took two hours to train and evaluate the model.

Since PCA-CapsNet provided the same values as I obtained when I trained CapsNet directly on the PTB dataset, I next trained the PCA-CapsNet model on the MIT-BIH Arrhythmia Dataset to examine the values for accuracy, precision and recall. I kept the number of PCA components constant so that I could effectively compare the two results. The results are provided in Table 11.

| PCA Components | Accuracy | Precision | Recall |
|---|---|---|---|
| 10 | 0.70 | 0.69 | 0.71 |
| 20 | 0.70 | 0.68 | 0.70 |
| 30 | 0.71 | 0.69 | 0.70 |
| 40 | 0.71 | 0.70 | 0.71 |
| **50** | **0.72** | **0.72** | **0.71** |

Table 11. Results for CapsNet with different PCA components on MIT_BIH Dataset

Figures 21 and 22 provide a summary of the results obtained from CapsNet and PCA-CapsNet on PTB ECG and MIT-BIH datasets. The figures show that the results are very similar.

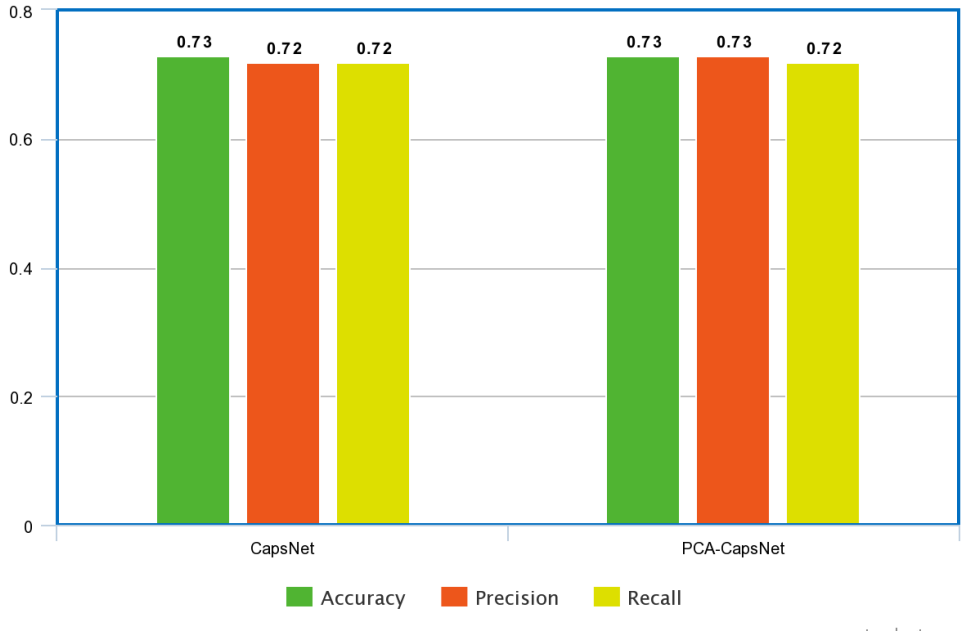


Figure 21. Results for CapsNet and CapsNet with PCA on PTB Dataset

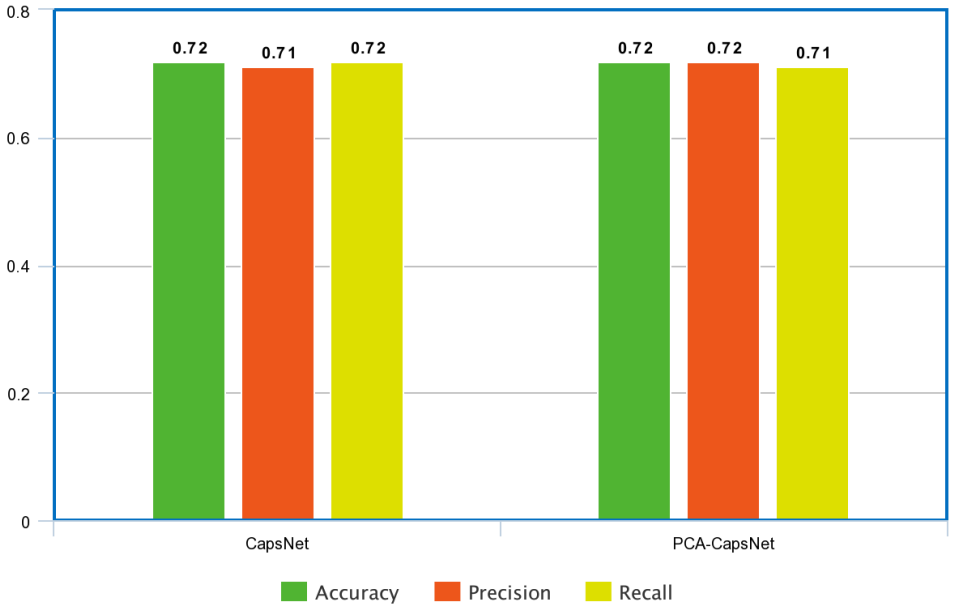


Figure 22. Results for CapsNet and CapsNet with PCA on MIT-BIH Dataset

Comparing the results of this experiment with the previous one shows that there was no improvement in the accuracy, precision and recall after using PCA to reduce the data's

dimensionality. At first glance, the decrease in accuracy could be attributed to the fact that since the dataset is in numerical format, more information loss occurred compared to a dataset that is in image format. But if that were the case, the accuracy of CapsNet model should be same as other ML models. Since the accuracy of other ML models is high, it can be surmised that no information loss occurred when the data were kept in numerical format instead of image format.

Furthermore, the accuracy, precision and recall values do not change when I ran CapsNet and PCA-CapsNet on MIT-BIH dataset and the PTB dataset. Running different variations of CapsNet on two different medical datasets reveals that its accuracy is much lower compared to the other ML models.

After running different types of experiments we see that the accuracy of CapsNet is lower as compared to existing ML models. Table 12 summarizes the accuracy of CapsNet on PTB database with the accuracies of ML models used by other researchers on the PTB dataset.

| **Work** | **Accuracy** | **ML Model Used** |
|---|---|---|
| Bhaskar[18] | 90.17% | SVM |
| Bhaskar[18] | 82.14% | ANN |
| Acharya et al. [21] | 95.22% | CNN |
| Lui & Chow[23] | 92.4% | CNN-LSTM |
| **This Report** | **73%** | **CapsNet** |

Table 12. Summary of Accuracies of Different Approached on PTB ECG dataset.

## VII. CONCLUSION

In the course of this research, I trained multiple ML models on the PTB ECG dataset and obtained high accuracy, precision and recall. The research shows that the accuracy, precision and recall values for CapsNet are much lower compared to traditional ML models on numerical data. After modifying the CapsNet to accept data in 2-dimensional format, the accuracy went down to zero percent. Since the MNIST dataset [33] had over 40,000 training images and 10,000 test images and the PTB ECG dataset had information for only 290 subjects, the low accuracy can be attributed to the small data quantity. I also ran CapsNet on MIT-BIH Arrhythmia dataset and obtained similar results to that when I ran it on the PTB dataset. Currently all the published research about CapsNet is on image datasets, not on numerical datasets. Based on previous research and the results of this writing project we can hypothesize that unlike other neural network models, CapsNet's current architecture is suitable for image data, not numerical data. With the advent of wearable devices like Apple Watch, Fitbit, etc. that record ECG data, a lot of exciting research in this domain will happen. This is because in the future, the availability of ECG datasets will increase due to the ubiquitous nature of wearable devices thereby providing ML models with tremendous data to learn from. The hardware of these devices will also be powerful enough for executing computationally expensive ML models like ANN, CNN. This will lead to earlier MI detection thereby saving a lot of lives.

## VI. FUTURE WORK

Future studies could increase the accuracy of CapsNet on the ECG dataset by conducting more experiments on CapsNet with different types of CapsNet architectures. One aspect of future work involves fine-tuning the CapsNet architecture so that it can train models much faster. Because CapsNet is a relatively new architecture as compared to other Neural Networks, it is more computationally extensive that other Neural Networks. Successive iterations of its architecture will help resolve this issue. Another aspect of future work revolves around running CapsNet on more powerful hardware. Due to hardware limitations of students' account on GCP, the CapsNet could not be trained on more powerful hardware with more memory and processor power. This led to training times in hours instead of a few minutes. Training the model on more powerful hardware could help in correctly evaluating and determining the optimal architecture for the CapsNet.

## REFERENCES

[1] E. Antman *et al*., "Myocardial infarction redefined- consensus document of The Joint European Society of Cardiology/American College of Cardiology committee for the redefinition of myocardial infarction: The Joint European Society of Cardiology/ American College of Cardiology Committee," Journal *of the American College of Cardiology*, vol. 36, no. 3, pp. 959 - 969, Sep. 2000. [Online]. Available: https://ac.els-cdn.com/S0735109700008044/1-s2.0-S0735109700008044-main.pdf?_tid=43559659-f2b9-452e-8ff4-457d2e73cf2b&acdnat=1543908141_3b448c8de4a97da7df9821ef670194d0

[2] A. K. Joshi, A. Tomar and M. Tomar, " A Review Paper on Analysis of Electrocardiograph (ECG) Signal for the Detection of Arrhythmia Abnormalities," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 3, no. 10, pp. 959 - 969, Oct. 2014. [Online]. Available: https://www.ijareeie.com/upload/2014/october/22D_A%20Review.pdf

[3] Cortes, Corinna, and Vladimir Vapnik. "Support-vector Networks." *Machine Learning* 20.3 (1995): 273-97. Web.

[4] B. Patel. "Efficient Classification of Data Using Decision Tree*". Bonfring International Journal of Data Mining*(2012). 2. 06-12. 10.9756/BIJDM.1098.

[5] Quinlan, J. "Induction of Decision Trees." *Machine Learning* 1.1 (1986): 81-106. Web.

[6] Quinlan, Jr. "Improved Use of Continuous Attributes in C4.5." *Journal Of Artificial Intelligence Research* 4 (1996): 77-90. Web.

[7] Breiman, Leo. *Classification and Regression Trees*. Belmont, Calif.: Wadsworth International Group, 1984. Print. Wadsworth Statistics/probability Ser.

[8] Decision Tree sklearn. https://scikit-learn.org/stable/modules/tree.html. Accessed on 4/9/2019

[9] Shiliang Sun, and Rongqing Huang. "An Adaptive K-nearest Neighbor Algorithm." *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery* 1 (2010): 91-94. Web.

[10] Marsland, Stephen. *Machine Learning: An Algorithmic Perspective*. Second ed. 2015. Chapman & Hall/CRC Machine Learning & Pattern Recognition Ser. Web.

[11] Graves, Alex. *Supervised Sequence Labelling with Recurrent Neural Networks*. 2012. Studies in Computational Intelligence, v. 385. Web.

[12] Gers, F A, J. Schmidhuber, and F. Cummins. "Learning to Forget: Continual Prediction with LSTM." *Neural Computation* 12.10 (2000): 2451-71. Web.

[13] S. Saha, Towards Data Science, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[14] Sewak, Mohit, Md. Rezaul Karim, and Pradeep Pujari. *Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python*. 2018. Web

[15] http://cs231n.github.io/convolutional-networks/

[16] G. Hinton et al. "Dynamic Routing Between Capsules." (2017). Web

[17] Vidal, René., Yi. Ma, and S.S. Sastry. *Generalized Principal Component Analysis*. 2016. Interdisciplinary Applied Mathematics, 40. Web.

[18] N. Bhaskar, Performance Analysis of Support Vector Machine and Neural Networks in Detection of Myocardial Infarction. Procedia Computer Science, vol. 46, pp 20-30. (2015). doi: 10.1016/j.procs.2015.01.043

[19] Pan, Jiapu, and Willis J Tompkins. "A Real-Time QRS Detection Algorithm." *IEEE Transactions on Biomedical Engineering* BME-32.3 (1985): 230-36. Web

[20] Jolliffe, Ian T. "Principal Component Analysis: A Beginner's Guide — I. Introduction and Application." *Weather*45.10 (1990): 375-82. Web.

[21] R. Acharya et al., Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals. *Information Sciences,* vol 415-416, pp 190-198. Nov 2017.

[22] Nasiri, Jalal A., Naghibzadeh, Mahmoud, Yazdi, H. Sadoghi, & Naghibzadeh, Bahram. (2009). ECG Arrhythmia Classification with Support Vector Machines and Genetic Algorithm. Computer Modeling and Simulation, 2009. EMS '09. Third UKSim European Symposium on, 187-192

[23] Lui, & Chow. (2018). Multiclass classification of myocardial infarction with convolutional and recurrent neural networks for portable ECG devices. Informatics in Medicine Unlocked, 13, 26-33.

[24] Hargittai, S. (2005). Savitzky-Golay least-squares polynomial filters in ECG signal processing. Computers in Cardiology, 2005, 32, 763-766.

[25] Tsien, C., Fraser, H., Long, W., & Kennedy, R. (1998). Using classification tree and logistic regression methods to diagnose myocardial infarction. Studies in Health Technology and Informatics, 52 Pt 1, 493-7.

[26] G. Weinberg et al., A computer-derived protocol to aid in the diagnosis of emergency room patients with acute chest pain. *The New England Journal of Medicine*, vol. 307, no10, pp 588-96, 1982.

[27] Bousseljot R, Kreiseler D, Schnabel, A. Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet. Biomedizinische Technik, Band 40, Ergänzungsband 1 (1995) S 317

[28] Moody GB, Mark RG. The impact of the MIT-BIH Arrhythmia Database. IEEE Eng in Med and Biol 20(3):45-50 (May-June 2001). (PMID: 11446209)

[29] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* **101**(23):e215-e220 [Circulation Electronic Pages; http://circ.ahajournals.org/content/101/23/e215.full]; 2000 (June 13)

[30] A. for the Advancement of Medical Instrumentation et al., “Testing and reporting performance results of cardiac rhythm and st segment measurement algorithms,” ANSI/AAMI EC38, vol. 1998, 1998.

[31] M. Kachuee et al. "ECG Heartbeat Classification: A Deep Transferable Representation." *ArXiv.org* (2018): ArXiv.org, Jul 12, 2018. Web.

[32] Ji, Xiaonan, and Po-Yin Yen. "Using MEDLINE elemental similarity to assist in the article screening process for systematic reviews." JMIR medical informatics 3, no. 3 (2015).

[33] LeCun, Yann; Corinna Cortes; Christopher J.C. Burges. "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges". Retrieved 17 August 2013