

# **Automatic Atrial Fibrillation Detection using Artificial Neural Network**

Ahood Mansoor Al Darmaki

A thesis submitted in partial fulfillment  
of the requirements for the degree

Master of Science

in

Electrical and Computer Engineering

Department of Electrical and Computer Engineering

College of Engineering

Sultan Qaboos University

Sultanate of Oman

2021

©

**Thesis of:** Ahood Mansoor Al Darmaki      **ID:** 85066

**Title of Thesis:** Machine Learning Techniques for Automatic Atrial Fibrillation Detection


**Thesis Committee:**

**1. Supervisor:** Dr. Lazhar Khriji

Title: Associate Professor

Department: Electrical and Computer Engineering

Institution: Sultan Qaboos University


Signature.......... Date. 1-6-2021

**2. Member:** Dr. Ahmed Chiheb Ammari

Title: Associate Professor

Department: Electrical and Computer Engineering

Institution: Sultan Qaboos University

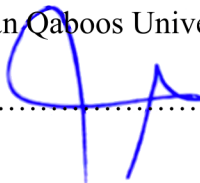
Signature.......... Date. 1-6-2021

**3. Member:** Dr. Medhat Hussein Awadalla

Title: Associate Professor

Department: Electrical and Computer Engineering

Institution: Sultan Qaboos University

Signature.......... Date. 1/6/2021

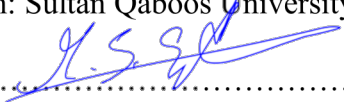
**Thesis Examining Committee:**

**1. Chair:** Dr. Mohamed Hassanien

Title: Assistant Professor

Department: Civil and Architectural Engineering

Institution: Sultan Qaboos University

Signature..........Date...6/6/2021

**2. Supervisor:** Dr. Lazhar Khriji

Title: Associate Professor

Department: Electrical and Computer Engineering

Institution: Sultan Qaboos University

Signature..........Date...1-6-2021

**3. Member (HoD representative):** Dr. Mostefa Mesbah

Title: Associate Professor

Department: Electrical and Computer Engineering

Institution: Sultan Qaboos University


Signature..........Date...03/06/2021

**4. External Examiner:** Dr. Hamza Zidoum

Title: Assistant Professor

Department: Computer Science

Institution: Sultan Qaboos University

Signature..........Date...June 6, 2021

## **Automatic Atrial Fibrillation Detection using Artificial Neural Network**

### **Abstract**

Atrial fibrillation is one of the serious heart diseases in which the heartbeats are irregular. Patients with this disease usually have shortness of breath, dizziness, and tiredness. Atrial fibrillation is considered a serious disease because the heart may develop clots of blood that might travel to the brain and cause a stroke, which may lead to death.

Since it is heart disease, atrial fibrillation can be detected by observing the electrocardiograph (ECG) of the patient. The ECG is usually characterized by its peaks, intervals, and segments, and using the ECG, many features have been extracted to detect atrial fibrillation. Atrial fibrillation can be identified by observing the heart rate variability and the atrial activity from the ECG.

In this work, I used an open-source feature extraction program that uses a modified version of the Pan-Tompkins algorithm and several other open-source algorithms to detect QRS complex and R peaks. Using the extracted features, I built an artificial neural network for automatic atrial fibrillation detection. Moreover, I measured the effect of using feature selection algorithms in enhancing the classification results. Also, I was able to overcome the challenge of the unbalanced dataset by using the weighted neural network and under-sampling the dataset to be almost balanced.

Based on the achieved results, and using feature selection tools, the performance of the classification could be improved. The obtained performance results are compared with other powerful tools. In this work and using an artificial neural network, I got an overall F1 score of 76.5% on the test dataset which is in the range of results achieved by other researchers. The complexity of the proposed system is light and fast which makes it a good choice to be used in portable devices and real-time applications.

## الكشف التلقائي عن الرجفان الأذيني باستخدام الشبكة العصبية الاصطناعية

### الخلاصة

الرجفان الأذيني هو أحد أمراض القلب الخطيرة التي تكون فيها ضربات القلب غير منتظمة. عادة ما يعاني المرضى المصابون بهذا المرض من ضيق في التنفس ودوخة وإرهاق. يعتبر الرجفان الأذيني مرضًا خطيرًا لأن القلب قد يصاب بجلطات دموية قد تنتقل إلى المخ وتسبب سكتة دماغية قد تؤدي إلى الوفاة.

نظرًا لأنه مرض يصيب القلب، يمكن اكتشاف الرجفان الأذيني من خلال مراقبة إشارة تخطيط القلب للمريض. تتميز إشارة تخطيط القلب عادةً بقممها وفترات وشرائحها، وباستخدام إشارة تخطيط القلب، تم استخراج العديد من الميزات للكشف عن الرجفان الأذيني. يمكن التعرف على الرجفان الأذيني من خلال ملاحظة تقلب معدل ضربات القلب والنشاط الأذيني من إشارة تخطيط القلب.

في هذا العمل، استخدمت برنامج مفتوح المصدر لاستخراج ميزات الرجفان الأذيني يستخدم نسخة معدلة من خوارزمية Pan-Tompkins والعديد من الخوارزميات مفتوحة المصدر الأخرى لاكتشاف مجمع QRS وقم R باستخدام الميزات المستخرجة، قمت ببناء شبكة عصبية اصطناعية للكشف التلقائي عن الرجفان الأذيني. علاوة على ذلك، قمت بقياس تأثير استخدام خوارزميات اختيار الميزات في تحسين نتائج التصنيف. أيضًا، تمكنت من التغلب على التحدي المتمثل في مجموعة البيانات غير المتوازنة باستخدام الشبكة العصبية الموزونة وأخذ عينات أقل من مجموعة البيانات لتكون متوازنة تقريبًا.

بناءً على النتائج المحققة، يمكن تحسين أداء التصنيف باستخدام أدوات اختيار الميزات. تمت مقارنة نتائج الأداء التي تم الحصول عليها مع الأدوات القوية الأخرى. في هذا العمل وباستخدام شبكة عصبية اصطناعية، حصلت على مجموع F1 بنسبة 76.5٪ في مجموعة بيانات الاختبار والتي تقع ضمن نطاق النتائج التي حققها باحثون آخرون. يعد تعقيد النظام المقترح خفيفًا وسريعًا مما يجعله خيارًا جيدًا لاستخدامه في الأجهزة المحمولة وتطبيقات الوقت الفعلي.

## Table of Contents

CHAPTER 1: INTRODUCTION .....	1
1.1 Cardiology Basics.....	1
1.1.1 Heart.....	1
1.1.2 Electrocardiogram .....	1
1.1.3 Atrial Fibrillation .....	3
1.2 Problem Description.....	4
1.3 Related Work.....	5
1.3.1 Beyond CinC 2017.....	9
1.3.2 Other works.....	11
1.4 Thesis Outline.....	13
CHAPTER 2: LITERATURE REVIEW .....	14
2.1 Signal Processing .....	14
2.1.1 Atrial Fibrillation Features.....	14
2.1.2 Other Abnormalities.....	16
2.2 Features Selection.....	16
2.2.1 Minimum Redundancy Maximum Relevance (mRMR) Algorithm ....	17
2.2.2 Chi-Square ( $\chi^2$ ) Test.....	17
2.2.3 ReliefF Algorithm .....	17
2.3 Artificial Neural Network .....	18
2.3.1 Neural Network Optimization.....	19
2.3.2 Activation Functions .....	21
CHAPTER 3: Atrial Fibrillation Automatic Detection.....	23
3.1 Tools.....	23
3.2 Data .....	23
3.3 Methodology .....	24
3.3.1 Pre-processing.....	25

3.3.2	Features Extraction.....	25
3.3.3	Features Selection .....	27
3.3.4	Training Neural Network .....	28
3.3.5	Results Evaluation.....	29
CHAPTER 4: RESULTS AND ANALYSIS.....		31
4.1	Deep Neural Network.....	31
4.1.1	Feature Selection .....	31
4.1.2	Tuning Network .....	34
4.2	Weighted Neural Network.....	36
4.2.1	Tuning Network .....	36
4.3	Under-sampling Dataset .....	39
4.3.1	Feature Selection .....	39
4.3.2	Tuning Network .....	42
4.4	Changing Activation Functions .....	45
4.5	Changing Learning Algorithm .....	46
4.6	Comparing with related work .....	49
CHAPTER 5: CONCLUSIONS & RECOMMENDATIONS.....		51
REFERENCES.....		52

## List of Figures

Figure 1.1: Heart Illustration [3] .....	1
Figure 1.2: ECG Morphology [5].....	3
Figure 1.3: Normal Sinus Rhythm vs Atrial Fibrillation [8].....	4
Figure 2.1: RR Interval [23].....	15
Figure 2.2: Example of Neural Network [31] .....	19
Figure 2.3: Hyperbolic Tangent Transfer Function [36].....	22
Figure 2.4: Positive Linear Transfer Function [37] .....	22
Figure 3.1: The ECG Signal of the Four Classes [38] .....	23
Figure 3.2: The Proposed System Block Diagram.....	24
Figure 3.3: Signal Before and After Pre-processing .....	25
Figure 3.4: Neural Network for 113 Input Features.....	28
Figure 4.1: Ranking of Features using ReliefF Algorithm .....	31
Figure 4.2: Ranking of Features using mRMR Algorithm .....	32
Figure 4.3: Ranking of Features using Chi-Square Algorithm .....	33
Figure 4.4: Comparison of Overall F1 using 3 Feature Selection Algorithms .....	36
Figure 4.5 Comparison of Overall F1 on Weighted Neural Network.....	39
Figure 4.6 Ranking of Features using the mRMR Algorithm from Under-Sampled Dataset.....	40
Figure 4.7 Ranking of Features using ReliefF Algorithm from Under-Sampled Dataset .....	41
Figure 4.8 Ranking of Features using Chi-Square Algorithm from Under-Sampled Dataset.....	42
Figure 4.9: Comparison of Overall F1 on Under-sampled Dataset .....	44
Figure 4.10: Comparing Learning Algorithms for Training Network .....	47
Figure 4.11 Changes in Performance with Each Epoch.....	48
Figure 4.12: The Best Results Neural Network Structure.....	49



## List of Tables

Table 1.1: ECG Characteristics and Their Normal Durations .....	2
Table 1.2: Data Profile of the Training Set .....	5
Table 1.3: Performance of Datta et al Proposed System.....	6
Table 1.4: Proposed System Performance of Hong et al. ....	7
Table 1.5: Proposed System Performance of Teijeiro et Al.....	8
Table 1.6: Proposed System Performance of Zabihi et al. ....	9
Table 1.7: Proposed System Performance of P. Cao et Al.....	10
Table 1.8: Performance of X. C. Cao et. al Proposed System .....	11
Table 1.9: Proposed System Performance of Wang et Al.....	12
Table 1.10 Performance of Faust et. al Proposed System.....	12
Table 3.1: Data Division .....	24
Table 3.2: Confusion Matrix .....	29
Table 4.1: Results of Deep Neural Network using mRMR Algorithm.....	34
Table 4.2: Results of Deep Neural Network using ReliefF Algorithm.....	35
Table 4.3: Results of Deep Neural Network using Chi-Square Algorithm.....	35
Table 4.4: Results of Weighted Neural Network using ReliefF Algorithm.....	37
Table 4.5 Results of Weighted Neural Network using mRMR Algorithm.....	38
Table 4.6 Results of Weighted Neural Network using Chi-Square Algorithm.....	38
Table 4.7: Results of Under-Sampling using ReliefF Algorithm .....	43
Table 4.8: Results of Under-Sampling using mRMR Algorithm .....	43
Table 4.9: Results of Under-Sampling using Chi-Square Algorithm .....	44
Table 4.10: Changing Activation Functions .....	46
Table 4.11: Results Using Different Learning Algorithms for Training Network ....	47
Table 4.12: Best Achieved Results .....	48
Table 4.13: Comparison of Result with Related Works .....	50

## **List of Abbreviations**

**ADI** Amplitude Dispersion Index

**AF** Atrial Fibrillation

**CCI** Cross-Correlation Index

**CNN** Convolutional Neural Network

**CoSEn** Coefficient of Sample Entropy

**ECG** Electrocardiograph

**KFD** Katz Fractal Dimension

**MAWSD** Mean of Absolute Weighted Successive Difference

**mRMR** Minimum Redundancy Maximum Relevance

**N** Normal

**O** Other

**RMSSD** Root Mean Square of Successive Differences

**TPC** Turning Point Count

**WI** Warping Index

# CHAPTER 1: INTRODUCTION

---

## 1.1 Cardiology Basics

### 1.1.1 Heart

The human heart is a muscular organ. It works to pump oxygenated blood to the rest of the body at each heartbeat. The heart consists of four chambers: two small upper atria and two large lower ventricles. One important part of the work of the heart is the sinoatrial node (sinus node) which is a group of cells that act as a natural pacemaker of the heart [1]. The normal heartbeat happens when the sinus node generates an electrical signal that travels through the heart causing the heart muscles to contract [2].

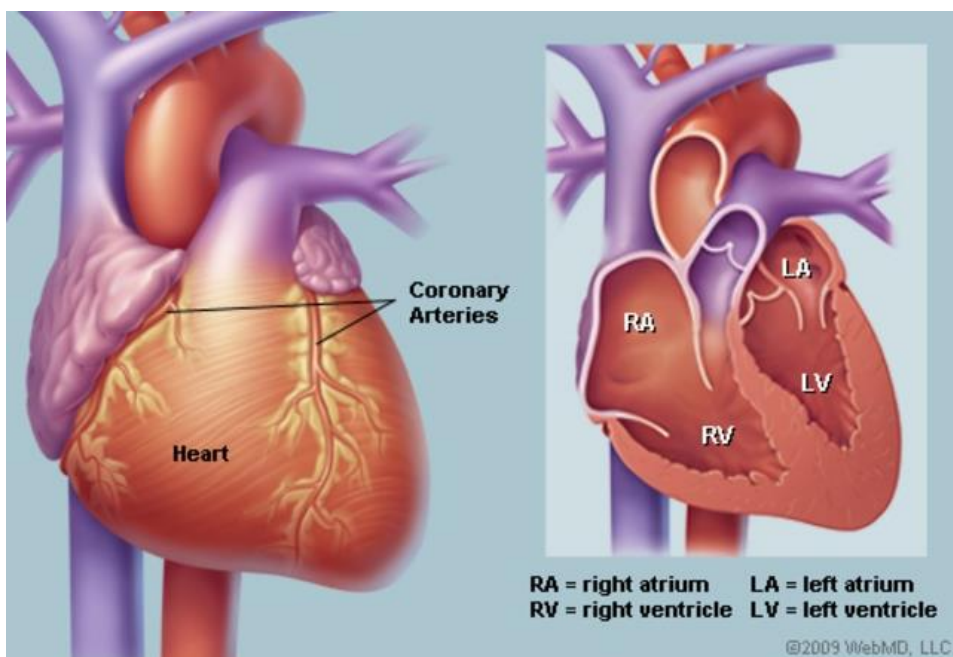


Figure 1.1: Heart Illustration [3]

### 1.1.2 Electrocardiogram

Electrocardiogram (ECG) is a graph that represents the measure of the electrical activity of the heart. Whenever the sinus node sends an electrical pulse, the ECG shows one wave that corresponds to one heartbeat. This wave represents the electrical signal traveling through the atria and ventricles, and it consists of three main parts, which are the P wave, the QRS complex, and the T wave. As shown in figure 1.2, the ECG signal's main characteristics are

the peaks (P, Q, R, S, and T), intervals (PR, RR, QRS, and QT), and segments (PR and ST). These characteristics have a normal amplitude or duration as can be seen in table 1.1[4].

Table 1.1: ECG Characteristics and Their Normal Durations

<b>Feature</b>	<b>Description</b>	<b>Duration</b>
RR interval	The interval between the R wave and the next R wave	0.6-1.2 s
P wave	First short upward movement of the ECG tracing	80ms
PR interval	Measured from the beginning of the P wave to the beginning of the QRS complex	120-200ms
QRS complex	Normally begins with a downward deflection Q, a larger upwards deflection R, and ends with a downward S wave	80-120ms
PR segment	Connects the P wave and the QRS complex	50-120ms
ST segment	Connects the QRS complex and the T wave	80-120ms
T wave	Normally a modest upward waveform	160ms
QT interval	Measured from the beginning of the QRS complex to the end of the T wave	420ms

In a normal person, the P wave is formed by the depolarization of the atria in which the electrical signal travels through the atria causing them to contract. Next, the QRS complex is formed by the depolarization of the ventricles and it is considered the reference point for signal analysis. The T wave is formed by repolarization in which the heart goes to its resting state [5].

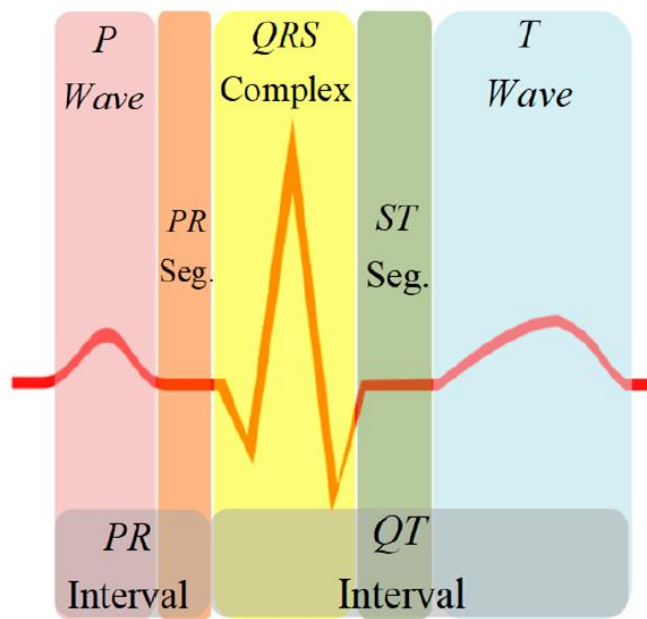


Figure 1.2: ECG Morphology [5]

### 1.1.3 Atrial Fibrillation

Atrial Fibrillation (AF or AFib) is an abnormal heart activity. The normal heart beats 60 – 100 beats per minute at rest while in atrial fibrillation patients the heart beats rapidly irregular beats that may vary between 140-180 beats per minute. In atrial fibrillation, the atria produce random electrical signals that override the sinus node signal. These signals cause the atria to quiver and contract at a high rate of 400 times per minute. Some of these impulses affect the ventricles and cause them to contract at a varying force but with a lower rate of 140 -180 times per minute [6].

The variation of contracting rate between the atria and the ventricles causes the ventricles to fill partially and pump blood to the lung with insufficient amounts. This causes the patient to feel dizziness and shortness of breath. Other symptoms include chest pain, heart palpitation, and low blood pressure [7].

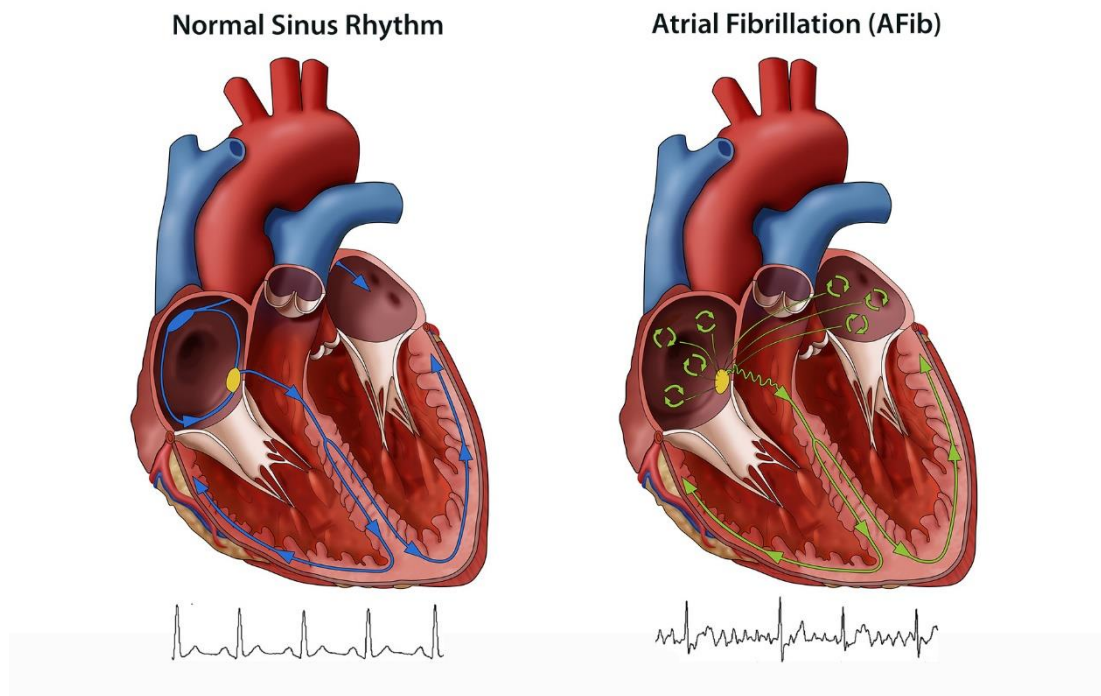


Figure 1.3: Normal Sinus Rhythm vs Atrial Fibrillation [8]

## 1.2 Problem Description

Atrial Fibrillation is considered a serious heart disease that can result in a heart attack or a stroke when it is not detected in the early stages. Because of this abnormality, the heart can develop small clots of blood that might travel to the brain, which causes a stroke, which may lead to death. The risk of developing a blood clot and having a stroke depends on various factors and the doctor can assess this. Atrial fibrillation can be diagnosed with several tests, which include the measurement of heart rate variability. Early detection of atrial fibrillation makes it possible to save patients' life.

The heart rate variability is the variation of the heartbeat intervals that can be detected from the ECG. The ECG has many features that can identify different rhythms of the heartbeat, which include normal sinus rhythm and abnormal arrhythmia. Many works have been done in this field to find the features of ECG that help in detecting atrial fibrillation.

In this work, I extracted atrial fibrillation features from the ECG records after preprocessing and removing useless information. I compared three algorithms for feature selection. I applied an artificial neural network for classifying the ECG records into four normal classes, atrial fibrillation, other, and noisy records.

### 1.3 Related Work

The PhysioNet/Computing in Cardiology (CinC) Challenge 2017 is a contest that invited researchers to solve this problem in which the competitors should differentiate atrial fibrillation from normal, noisy, and other rhythms. The challenge provides short-term ECG recordings (from 9-61 s) performed by patients using AliveCor handheld devices. The dataset is divided into 8,528 in the public training set and 3,658 in the private hidden test set. The number of recordings in each class and the time length of recordings are shown in table 1.2. The data are stored as 300 Hz, 16-bit files with bandwidth 0.5-40 Hz and a  $\pm 5$  mV dynamic range. The dataset has 4 classes which are normal sinus rhythm, atrial fibrillation, other rhythms, and noise. Another rhythm is all non-AF abnormal rhythms and noisy is the signal that is too noisy to be classified. 75 teams entered the challenge and 4 of them won with an equal score [9]. Here I will highlight the four best score works.

Table 1.2: Data Profile of the Training Set

Type	# Recording	Time length (s)				
		Mean	SD	Max	Median	Min
<b>Normal</b>	5154	31.9	10.0	61.0	30	9.0
<b>AF</b>	771	31.6	12.5	60	30	10.0
<b>Other rhythm</b>	2557	34.1	11.8	60.9	30	9.1
<b>Noisy</b>	46	27.1	9.0	60	30	10.2
<b>Total</b>	<b>8528</b>	<b>32.5</b>	<b>10.9</b>	<b>61.0</b>	<b>30</b>	<b>9.0</b>

The first work that won the challenge is proposed by Datta et al. [10]. They proposed a two-layer binary cascaded approach. They extracted more than 150 features categorized into: morphological, prior art AF, HRV, frequency, statistical, other abnormalities, and detecting noisy recording features. They used feature selection to improve the classification using

statistical feature selection tools like Maximal Information Coefficient (MIC) and minimum Redundancy Maximum Relevance (mRMR). For classification, they used adaptive boosting in two-layer cascaded classifiers. In the first layer, the signals are classified into AF + noisy or normal + other. In the second layer, there are two classifiers for each of the two classes from the previous step and they will give the final classification. For each of the three classifiers, two parameters of the ensemble classifiers which are the number of learning cycles and learning rate are optimized using the Bayesian optimization function. Feature extraction is also applied before each classifier in the two layers but there is a different set for each classifier. The performance of their work is shown in table 1.3. This table shows how their proposed system performs on the training dataset and the hidden test set. The values in the table are the F score of normal, AF, other, and noisy, respectively. The limitation to this work is the unavailability of actual disease information corresponding to each recording in the other rhythm class which makes the classification task more challenging to identify the proper features, thus reducing the classification accuracy.

Table 1.3: Performance of Datta et al Proposed System

<b>Data set</b>	<b>F<sub>normal</sub></b>	<b>F<sub>AF</sub></b>	<b>F<sub>other</sub></b>	<b>F<sub>noisy</sub></b>
Complete training data	0.99	0.94	0.98	0.96
Complete test data	0.92	0.82	0.75	0.83

Another work that won the challenge has been proposed by Hong et al [11]. They propose an ensemble classifier using three different kinds of features, which are Expert features, Deep features, and Center wave features. The total number of features extracted is over 600 features. The expert features are categorized into statistical features, signal-processing features, and medical features. The deep features are the last hidden layer extracted from the deep neural network as features. Two different deep feature extractors have been used. One deep feature extractor is based on a deep residual convolution neural network and it is trained using expanded data. The second deep feature extractor is based on a recurrent neural network and it is trained using center wave data. The center wave features are extracted from the center wave, which is the most representative wave among the signal. They extract the three types of features and combine them to be trained using individual



classifiers. The individual classifiers used are extreme gradient boosting of decision trees (XGBoost). The classifiers are assembled by averaging the predicted probabilities. The final system ensemble five XGBoost, and each XGBoost has 3000 trees with max depth = 9, min child weight = 3. The performance of their proposed system is shown in table 1.4. The table shows the F score for the four classes, normal, AF, other, and noisy. The F score calculated using Expert Features, Expert and Centerwave Features, and Expert, Centerwave, and Deep Features. In addition, the overall F score is shown. Moreover, they have experimented with the system in three different scenarios, the first using only expert features, the second using expert and center wave features, and the third using all three types of features. The researchers in this work state that this system could detect more classes of heart diseases if providing more data.

Table 1.4: Proposed System Performance of Hong et al.

<b>Features</b>	<b>F<sub>normal</sub></b>	<b>F<sub>AF</sub></b>	<b>F<sub>other</sub></b>	<b>F<sub>noisy</sub></b>	<b>F1</b>
E	0.9059	0.7908	0.7543	0.6574	0.7771
E + C	0.9086	0.7899	0.7622	0.6603	0.7803
E + C + D	0.9204	0.8692	0.8068	0.8156	0.8530

Teijeiro et al. [12] proposed the third work that won the challenge. The proposed work is based on features provided by abductive interpretation of the signal using the Construe algorithm. In their work, they extracted two types of features, which are global features and pre-beat features. The global features are categorized into rhythm features, morphological features, and signal quality features. The global features, which are 79 features in total, summarize the information provided by the Construe algorithm for adductive interpretation. Pre-beat features are extracted after the global features to classify the records that cannot be classified globally, and these features are global but disaggregated to the heartbeat scope. After the feature extraction, global classification is applied to the global features and sequence classification is applied to the pre-beat features. For the global classification, they used Extreme Tree Gradient Boosting (XGBoost) algorithm. They used exhaustive grid search and 8-fold cross-validation for tuning the hyperparameters. The final optimized classifier was with the following parameters: Maximum tree depth: 6, Learning rate: 0.2,

Gamma: 1.0, Column subsample by tree: 0.9, Min. child weight: 20, Subsample: 0.8, and Number of boosting rounds: 60. The sequence classification method used is based on Recurrent Neural Network (RNN), specifically; they used Long Short-Term Memory networks (LSTM). The classifier uses 4 LSTMs, the first one preprocesses the sequence of transformed features and returns a new sequence, which is subsequently used by the other LSTMs. All the LSTMs used has 128 units. The two classifiers are stacked using Linear Discriminant Analysis (LDA) classifier and the final classification is the output of the stack classifier. The performance of their proposed system is shown in table 1.5. The table shows the F score of the system when using only either one of the classifiers and when using them both stacked using 8-fold cross-validation.

Table 1.5: Proposed System Performance of Teijeiro et Al.

Method	Fold Number								Mean
	0	1	2	3	4	5	6	7	
<b>XGBoost</b>	0.84	0.84	0.85	0.85	0.82	0.80	0.82	0.82	0.83
<b>RNN</b>	0.82	0.81	0.84	0.83	0.86	0.83	0.83	0.83	0.83
<b>LDA-stacker</b>	0.85	0.84	0.86	0.86	0.85	0.83	0.84	0.85	0.85

Teijeiro et al. [12] proposed the third work that won the challenge. The proposed work is based on features provided by the abductive interpretation of the signal using the Construe algorithm. In their work, they extracted two types of features, which are global features and pre-beat features. The global features are categorized into rhythm features, morphological features, and signal quality features. The global features, the fourth work that won the challenge was proposed by Zabihi et. al. [13]. In their work, they manually extracted 491 features, which are a combination of features extracted from the signal, and features extracted from the prediction of the base-level classifiers. These features are ranked using a random forest classifier based on their importance, which is evaluated based on the reduction of entropy. The 150 highest ranked features are selected and listed as base-level time domain and morphological features, base-level frequency domain features, base-level time-frequency domain features, base-level nonlinear (phase space) features, and meta-level

features. For classification, they used an external random forest classifier using 500 decision trees and a random selection of features at each node creation. Then, they used bagging to train each decision tree, and 30 features are randomly selected for each node. The performance of their proposed classifier is shown in table 1.6. The table shows the F score of the four classes and the mean F score when applied to the training dataset and the test dataset.

Table 1.6: Proposed System Performance of Zabihi et al.

<b>Evaluation metrics</b>	<b>F<sub>normal</sub></b>	<b>F<sub>AF</sub></b>	<b>F<sub>other</sub></b>	<b>F<sub>noisy</sub></b>	<b>F1</b>
<b>Training set (%)</b>	90.49±0.96	79.43±4.52	75.64±3.11	61.11±7.53	81.85±2.57
<b>Testing set (%)</b>	90.87	83.51	73.41	50.42	83

### 1.3.1 Beyond CinC 2017

After the challenge, many researchers have tackled the problem using the same dataset used in the challenge to improve the results. P. Cao et al. [14] presents one of these works. In their work, they focused on balancing the dataset to get better results. They propose a data augmentation strategy for balancing the data. The strategy starts by detecting the QRS complex using the Pan-Tompkins algorithm and assigning the beginning of each complex to then take the segment between the first and last assigned starting point as a selected sequence. Then, the selected sequence is duplicated and concatenated to the duplicate to be resampled using a sliding window. For comparison, they used two different methods for data augmentation, which are window slicing and permutation. After preparing the balanced dataset, they used a recurrent neural network for classification. They used a 2-layer LSTM network with cross-entropy as a loss function. For optimizing the network, they used Adam and stochastic gradient descent optimizers. The stochastic gradient descent replaces the Adam optimizer for a better result after it fails to decrease the loss for 5 epochs. They use 10-fold cross-validation to evaluate the performance. The performance of the system while using the data augmentation and without using it is shown in table 1.7. The table shows the F score for only three classes while they ignored the noisy signal class. The limitation of this system is that it may disrupt the patterns of RR interval of the raw signal thus generate

new samples with non-physiological rhythms. In addition, the data augmentation was not tested in other classical deep neural network architectures, such as the convolutional neural network.

Table 1.7: Proposed System Performance of P. Cao et Al.

<b>Evaluation metrics</b>	<b>F<sub>normal</sub></b>	<b>F<sub>AF</sub></b>	<b>F<sub>other</sub></b>	<b>F1</b>
<b>With Data Augmentation</b>	0.860±0.028	0.754±0.029	0.677±0.035	0.764±0.028
<b>Without Data Augmentation</b>	0.380±0.075	0.308±0.120	0.380±0.066	0.356±0.046

X. C. Cao et al. [15] propose another work that also tackled the same problem using the same dataset. Like the previous work, their concern was also the unbalanced dataset. To solve this issue, they re-segmented the ECG records into short samples of 9 seconds. They intercept short samples depending on the length of the signal. For normal rhythm and other abnormalities classes, they intercept from the middle without overlapping. For atrial fibrillation and noisy classes, they intercept more segments with overlapping since they have few numbers of samples. After this process, they end up with 19188 samples. After balancing the dataset, they apply derived wavelet frames for signal decomposition to further prepare the data. This work proposes two convolutional neural network models, which are fast down-sampling residual convolutional neural network (FDResNet) and multi-scale decomposition, enhanced fast down-sampling residual convolutional neural network (MSResNet). FDResNet model is composed of three main parts, which are a fast down-sampling module, a residual convolution module, and a classification module. The fast down-sampling module is mainly two convolutional layers. The residual convolutional module is three modules consisting of convolutional layers followed by a residual short circuit. The classification module consists of 1 fatten layer, 2 full connection layers, and a softmax classifier. The other convolutional model is MSResNet and it consists of three FDResNet followed by a small neural network. The performance of this work can be seen in table 1.8 represented as the F score for three classes and the overall F score.

Table 1.8: Performance of X. C. Cao et. al Proposed System

$F_{\text{normal}}$	$F_{\text{AF}}$	$F_{\text{other}}$	Overall F1
0.881	0.966	0.851	0.899

### 1.3.2 Other works

Other than these works, some researchers have also worked to solve this problem using other sources of data. One most used source is the Massachusetts Institute of Technology (MIT)-Boston's Beth Israel Hospital (BIH) Atrial fibrillation database. The database has 23 long-term ECG recordings of human subjects with atrial fibrillation with a duration of 10 hours for each record. Each record contains two ECG signals each sampled at 250 samples per second with 12-bit resolution over a range of  $\pm 10$  millivolts. They are made using ambulatory ECG recorders with a typical recording bandwidth of approximately 0.1 Hz to 40 Hz. The recordings contain rhythm annotations of types AFIB (atrial fibrillation), AFL (atrial flutter), J (AV junctional rhythm), and N (used to indicate all other rhythms) [16].

Wang et al. [17] propose one of the recent works that have used this database. In their work, they extract features based on wavelet packet transform to be fed to the artificial neural network for classification. After filtering the ECG segments, the wavelet coefficients are obtained from the decomposed ECG and divided into an equal number of segments. From the segments, the correlation matrix is computed, and based on it the histogram is constructed. The features needed for the classifier are extracted from the histogram and assembled as a feature set. To classify the records, these features are fed to a 3-layers neural network. The network has an input layer of 4 neurons, an output layer of 2 neurons, and a hidden layer of 10 neurons. They used the sigmoid function as an activation function in the hidden layer. The value of adaptive learning rate was 0.1 by setting mean square error no more than 0.001. The work they proposed performed well as can be seen in table 1.9. The table shows the performance for different types of classifiers that have been used; support vector machine (SVM), k-nearest neighbors (KNN), and artificial neural network (ANN).

Table 1.9: Proposed System Performance of Wang et Al.

<b>Classifier</b>	<b>Accuracy (%)</b>	<b>Sensitivity (%)</b>	<b>Specificity (%)</b>
<b>SVM</b>	97.2	97.8	97.4
<b>KNN</b>	96.3	95.4	96.1
<b>ANN</b>	98.8	98.7	98.9

Faust et al. [18] propose another work that used the same dataset. In this work, they propose a deep recurrent neural network to classify the signal. They used 20 of the signals from the MIT-BIH database for training and 3 for testing. They partition the long signals with a sliding window of 100 beats into blocks that are fed directly to a recurrent neural network. The proposed model consists of two bidirectional long-short term memory (LSTM), two fully connected layers, and a global max-pooling layer. The two LSTM layers, forward and backward, have cells twice the length of the input sequence and they work to learn and extract the features from the heart rate data sequence. Then, the resulting features are passed to the global max-pooling layer to be compressed before proceeding to the fully connected layers where they will be given the final classification. The model has been trained and evaluated with 10-fold cross-validation and tested with blind-fold evaluation. The performance of their proposed model can be seen in table 1.10. The 10-fold cross-validation was applied to 20 records of the database while the other 3 records were used for testing the proposed model using blind-fold validation.

Table 1.10 Performance of Faust et. al Proposed System

<b>Evaluation Metrics</b>	<b>Accuracy</b>	<b>Sensitivity</b>	<b>Specificity</b>
<b>Cross-Validation</b>	98.51%	98.32%	98.67%
<b>Blind Fold Validation</b>	99.77%	99.87%	99.61%

## **1.4 Thesis Outline**

This thesis is organized as follows:

Chapter 2 presents the literature survey in which AF features, selection tools, and machine learning techniques are mentioned. Chapter 3 shows the details of the methodology and the steps that have been taken to accomplish the work. Each step is explained in detail. Chapter 4 analyzes the obtained results with explanatory tables and graphs. Chapter 5 concludes the report with recommendations and future work.

## CHAPTER 2: LITERATURE REVIEW

---

To know more about the problem and how to solve it, I first start with signal processing and knowing about atrial fibrillation features in literature and the other abnormalities. Moreover, I also highlight the use of feature selection algorithms in the next sections. Since I am using neural networks, I also give brief information about them and their usage in the last sections.

### 2.1 Signal Processing

As mentioned previously, the ECG is characterized by the peaks, the intervals, and the segments. From these characteristics, many features can be extracted to detect atrial fibrillation and other abnormalities.

#### 2.1.1 Atrial Fibrillation Features

One of the most used characteristics for extracting features is the RR interval. The RR interval is the time between two successive R peaks as can be seen in figure 2.1. Tateno et al. [19] proposed to use the Kolmogorov-Smirnov test and the standard coefficients of variation test based on RR and  $\Delta RR$ , which is the difference between two successive RR intervals. They use the standard density of histogram of RR and  $\Delta RR$  as a template for atrial fibrillation detection. They compare the coefficient of variation with the standard coefficient of variation (CV test) and compare the density histogram with the standard density of histogram (Kolmogorov-Smirnov test).

On the other hand, Ghodrati et. al [20] extracted features based on the absolute deviation of RR interval and  $\Delta RR$ , and they call them normalized absolute deviation and normalized absolute difference. In another work, Ghodrati et Al. [21] presented other features based on the statistical analysis of the RR interval in which they compared Gaussian and Laplace probability density functions when applied to the histogram of the normalized RR differences using the Neyman-Pearson detection approach.

Billeci et al. have mentioned other features, which are extracted from the RR interval, [22] include the mean, the minimum, and the maximum value of the RR intervals. The root means square of the successive differences (RMSSD), the mean of the absolute weighted



successive difference (MAWSD), the coefficient of sample entropy (CoSEn), the turning point count (TPC), and the Katz Fractal Dimension (KFD).

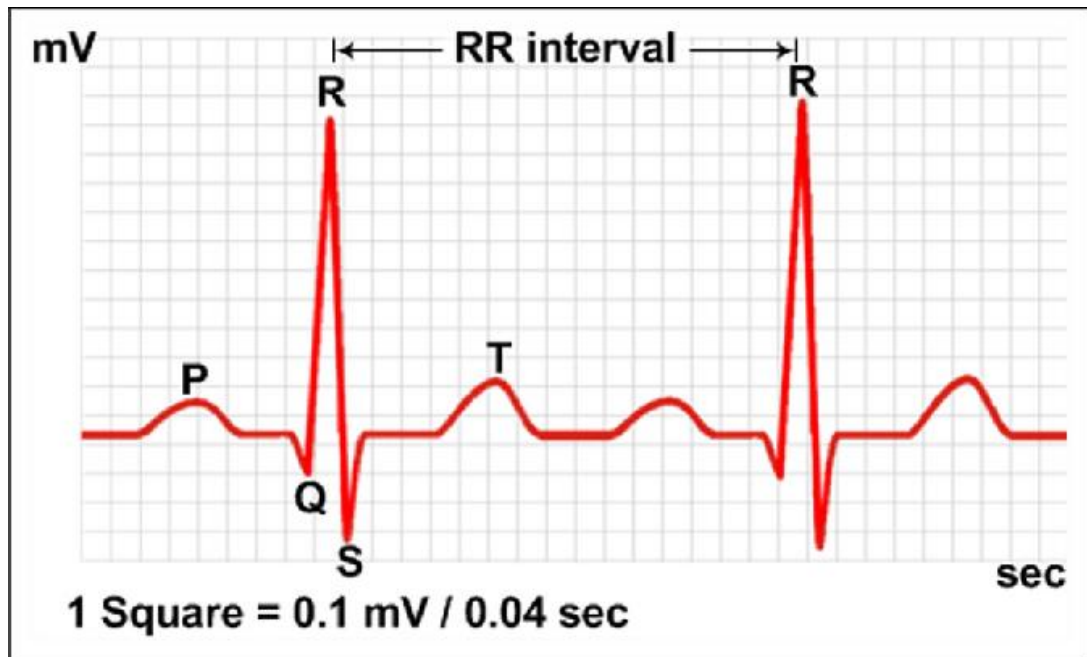


Figure 2.1: RR Interval [23]

Another characteristic that is used to detect atrial fibrillation is the P-wave, which happens because of the atrial activity. The absence of the P-wave indicates the presence of atrial fibrillation. Firoozabadi et al. [24] extracted several features based on the P-wave. These features include the mean and standard deviation of the following measures within the segment: PR interval, P-wave duration, P-wave onset-peak duration, P-wave amplitude (peak-onset). Also, the number of P-waves detected in the segment, the presence or absence of potential P-wave in average beat, mean and standard deviation of the correlation of P-waves in average beat with each beat in the segment.

In the same field, Censi et al. [25] worked to quantify the P-wave variability over time using three algorithms; based on cross-correlation function, butterfly plots, and dynamic time warping. Based on these algorithms, they extracted three novel indices: the first one is based on the cross-correlation coefficients among the P-waves (Cross-Correlation Index, CCI), the second one is associated with the variation in the amplitude of the P-waves (Amplitude Dispersion Index, ADI), and the third one is sensible to the phase shift among P-waves (Warping Index, WI). Based on these indices and from P-wave templates, they extracted features that were used to identify atrial fibrillation patients.

### **2.1.2 Other Abnormalities**

Other features can distinguish atrial fibrillation from other abnormalities. Some of the features can be extracted in the time domain, other features can be extracted in the frequency domain, and others in the time-frequency domain.

There are other heartbeat abnormalities or arrhythmia, which are categorized by the chambers of the heart in which they occur and by what effect they have on the heart's rhythm. The two main types of heart arrhythmia are tachycardia and bradycardia. Tachycardia refers to a fast heart rhythm of a rate over 100 beats per minute. Bradycardia refers to a slow heart rhythm of heart rate below 60 beats per minute, and it is caused by a failure of the heart signals to fire, as they should. There are three major types of tachycardia: Atrial tachycardia (starting in the atria), Supraventricular tachycardia (starting above the ventricles), and Ventricular tachycardia (starting in the ventricles) [23]. Atrial fibrillation is one of the most common supraventricular tachycardia. Other abnormalities that are in the same category are paroxysmal supraventricular tachycardia (PSVT), atrial flutter, and Wolff–Parkinson–White syndrome.

## **2.2 Features Selection**

Many studies have addressed the importance of feature selection in classification problems. Feature selection is the action of choosing a subset of features from an already existing set. Feature selection is used to improve classification accuracy by removing irrelevant and redundant features. Feature selection algorithms can be categorized into wrapper, filter, and embedded. Wrapping methods compute models with a certain subset of features and evaluate the importance of each feature. Then they iterate and try a different subset of features until the optimal subset is reached. Filter methods use a measure other than error rate to determine whether that feature is useful. Rather than tuning a model, a subset of the features is selected by ranking them by a useful descriptive measure. Embedded methods perform feature selection as a part of the model creation process. This method is between the two methods of feature selection previously explained, as the selection is done in conjunction with the model tuning process [26].

In this thesis, three different algorithms based on the filter type were compared, the minimum redundancy maximum relevance (mRMR) algorithm, Chi-square tests, and ReliefF algorithm. I chose to use filter methods to avoid overfitting the model.

### 2.2.1 Minimum Redundancy Maximum Relevance (mRMR) Algorithm

Minimum Redundancy Maximum Relevance (mRMR) Algorithm selects the features that have a high correlation with the target class but low correlation among themselves. This algorithm uses the mutual information difference (MID) criterion as an objective function of relevance and the mutual information quotient (MIQ) criterion as an objective function of redundancy [27]. As can be seen in equation (2.1), the purpose of this algorithm is to maximize the relevance which is  $I(x, y)$  and minimize the redundancy which is  $\frac{1}{|S|} \sum_{z \in S} I(x, z)$  by finding the optimal set  $S$  of features. The relevance and redundancy are defined with mutual information  $I(\cdot)$  [28].

$$\max_{x \in S^c} MIQ_x = \max_{x \in S^c} \frac{I(x, y)}{\frac{1}{|S|} \sum_{z \in S} I(x, z)} \quad (2.1)$$

### 2.2.2 Chi-Square ( $\chi^2$ ) Test

Chi-Square ( $\chi^2$ ) Test is a simple and general algorithm that measures how a model compares the actual observed data and the expected value. The algorithm depends on the difference between the actual and the expected values, the degrees of freedom that refer to the maximum number of logically independent values, and the size of the samples. This algorithm can be used to test whether two variables are related or independent from each other. Also, it can be used to test the goodness of fit between an observed distribution and a theoretical distribution of frequencies. Equation (2.2) represents the formula of the Chi-square algorithm, where  $E$  is the expected value or in this case the category,  $O$  is the observed value, and  $c$  is the degree of freedom [29].

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i} \quad (2.2)$$

### 2.2.3 ReliefF Algorithm

ReliefF Algorithm is one of the most widely used algorithms for feature selection and it is one of the Relief algorithms family. It is an enhanced version of the Relief algorithm to be used for multi-class classification problems and to be more robust against noise and incomplete data. This algorithm is based on finding the nearest hit and nearest miss of an instance, which means the neighboring features of the same class and the neighboring

features of the opposite class, respectively. ReliefF algorithm takes an instance and finds its neighbors and according to the distances between them, it gives them weights. The weight value is high if the instance distinguishes the opposite class and not the same class [30]. Equation (2.3) shows the weight update for the features using the ReliefF algorithm where  $x_r$  is a random observation and  $x_q$  is the nearest neighbor in the same class.

$$W_j^i = W_j^{i-1} - \frac{\Delta_j(x_r, x_q)}{m} \cdot d_{rq} \quad (2.3)$$

The weight update for the features in the case where  $x_r$  and  $x_q$  are not in the same class will be as in equation (2.4).

$$W_j^i = W_j^{i-1} + \frac{p_{y_q}}{1 - p_{y_r}} \cdot \frac{\Delta_j(x_r, x_q)}{m} \quad (2.4)$$

Where  $\Delta_j(x_r, x_q)$  is the difference in the value of the feature between observations  $x_r$  and  $x_q$ .  $d_{rq}$  is a distance function,  $m$  is the number of iterations,  $p_{y_r}$  is the prior probability of the class to which  $x_r$  belongs, and  $p_{y_q}$  is the prior probability of the class to which  $x_q$  belongs.

### 2.3 Artificial Neural Network

Machine Learning is a part of artificial intelligence that studies the algorithms and tools used to learn and improve its analyses. These algorithms use input and output datasets to recognize the patterns and learn from experience, which trains the machine to forecast future events and make recommendations without human interaction. Machine learning techniques can be categorized into two main categories: supervised and unsupervised algorithms. Supervised algorithms use datasets labeled as inputs and outputs to train a model. The trained model is used for any other set of data to predict and forecast their output. The supervised algorithms can be seen in classification problems. On the other hand, unsupervised algorithms use unlabeled data, and the system will try to recognize the relations between the data and divide them into groups of the same category that will be decided by the algorithm. Unsupervised algorithms can be seen in clustering problems [31].

Artificial Neural Network is one of the machine learning techniques that is being widely used in many applications to analyze the data and learn to recommend the output. The neural network can be thought of as an artificial model of how the human neural system work. The

neural network consists of neurons that are organized into layers. It starts with an input layer that has several neurons equal to the number of input variables. The input layer is followed by the hidden layer, which varies in size depending on the application, it can be one hidden layer (shallow neural network) or more than one layer (deep neural network). At the end of the network, there is an output layer. In classification problems, the output layer has several neurons equal to the number of classes. In each layer, there are biases and weights for every neuron and these keep changing in a process called training until the network reaches the target results by satisfying an objective function. For training, special optimization algorithms are used. Each neuron has an activation function that works by linearly combining the inputs of the neuron into one output. An example of a neural network is shown in figure 2.2 [31].

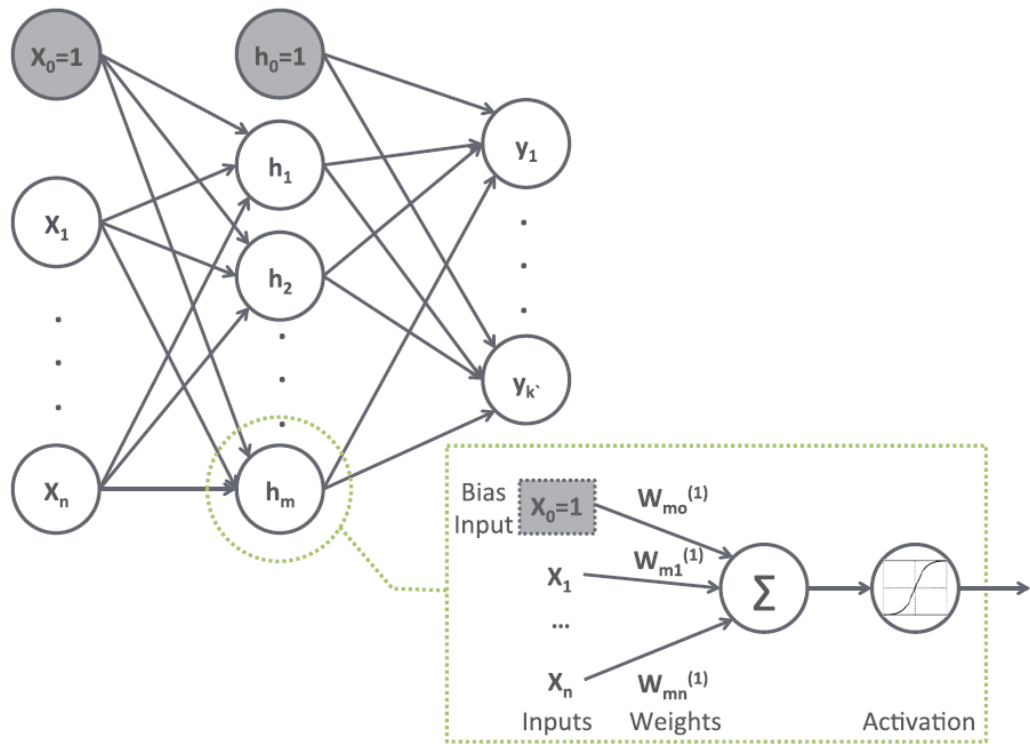


Figure 2.2: Example of Neural Network [31]

### 2.3.1 Neural Network Optimization

Optimization or training algorithms are used in a neural network to update the weights and biases of the neurons to satisfy an objective function. The weights and biases are updated using small steps called the learning rate. The learning rate is a hyperparameter that should be set to control how much the weights and biases are adjusted. Setting the learning rate to

a very small value may result in training for a long time and setting it to a very large value may result in missing the optimal result. The objective function or the cost function is a function that measures the performance of a machine-learning model for given data by quantifying the error between predicted values and expected values. It is the real-valued function whose value is to be either minimized or maximized over the set of feasible alternatives. The cost function that aims to minimize the error between the prediction and the expectation is usually called the loss function.

Loss functions can be categorized based on their application into regression loss functions and classification loss functions. In classification, there are exponential loss, square loss, hinge loss, logistic loss, savage loss, and tangent loss. Square loss is more commonly used in regression but can be utilized in classification. It is convex and smooth. It is slower than hinge and logistic loss functions but can solve for the regularization parameter using cross-validation. Logistic loss is also called cross-entropy loss and it is less sensitive to outliers because it is convex and grows linearly for negative values. The exponential loss is convex and grows exponentially for negative values, which makes it more sensitive to outliers. Hinge loss is convex and continuous but not smooth (is not differentiable) so cannot be used with gradient descent methods. Tangent loss and savage loss are quasi-convex and bounded for large negative values, which makes them less sensitive to outliers. Both have been used in gradient descent methods [32].

Optimization algorithms can be divided into two groups; one is used for differentiable loss functions and the other for non-differentiable loss functions. Using the gradient to optimize a function has been proven to be easier and so many research works have been done in algorithms that use the gradient. Some groups of these algorithms are bracketing algorithms, local descent algorithms, first-order algorithms, and second-order algorithms.

One of the most popular algorithms and most common way for optimization is the first-order algorithms, which are generally called the gradient descent algorithms. The gradient descent algorithms use the gradient to choose the direction to move in the search space. It starts by calculating the gradient of the function and then following it in the opposite direction. The gradient descent has three variations that use a different amount of data to compute the gradient and they try to make a trade-off between the accuracy and the time consumed. These variants are batch gradient descent (using all samples), stochastic gradient descent (using one sample), and mini-batch gradient descent (using a small subset of

samples). Some of the algorithms that depend on gradient descent are momentum, Adagrad, and Adam. Momentum helps accelerate stochastic gradient descent in the relevant direction and so the convergence will be faster with fewer oscillations. Adagrad adapts the learning rate to the parameters, which eliminates the need to manually tune the learning rate. Adaptive Moment Estimation (Adam) also computes adaptive learning rates for each parameter while keeping an exponentially decaying average of the past gradient [32].

Another algorithm is scaled conjugate gradient backpropagation, which is based on conjugate directions, but this algorithm does not perform a line search at each iteration. The conjugate gradient method is generalized by the nonlinear conjugate gradient method for nonlinear optimization. In addition, there is a resilient backpropagation algorithm, which eliminates the harmful effects of the magnitudes of the partial derivatives by only considering the sign of the derivative to determine the direction of the weight update [33].

### 2.3.2 Activation Functions

Activation Functions or transfer functions are used in artificial neural networks to transform an input signal into an output signal to be fed as input to the next layer. In an artificial neural network, the sum of products of inputs and their weights are calculated and then passed to an activation function to get the output of that particular layer and supply it as the input to the next layer [34].

Many different activation functions can be categorized as linear and non-linear functions. Some of the known functions are sigmoid function, hyperbolic tangent function, softmax function, softsign function, rectified linear unit function, softplus function, exponential linear function, max out function, swish function, the exponential linear squashing, and hard exponential linear squashing [35]. In this section, I focus on three functions, which are used in the proposed system.

One of the most used activation functions is the hyperbolic tangent transfer function known as tanh function, which is a smooth zero-centered function with an output bound between -1 and 1. The definition of the hyperbolic tangent transfer function is given in equation (2.5). The shape of the function is shown in figure 2.3.

$$\tanh(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}} \quad (2.5)$$

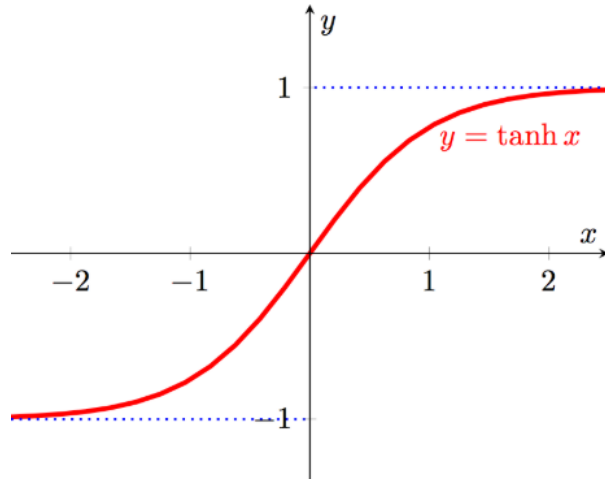


Figure 2.3: Hyperbolic Tangent Transfer Function [36]

Rectified Linear Unit (ReLU) also known as Positive linear transfer function is a function that is a straight line only for positive values and zeroes elsewhere. It is defined in equation (2.6) and the shape in figure 2.4.

$$\max(0, n) = \begin{cases} n_i, & \text{if } n_i \geq 0 \\ 0, & \text{if } n_i < 0 \end{cases} \quad (2.6)$$

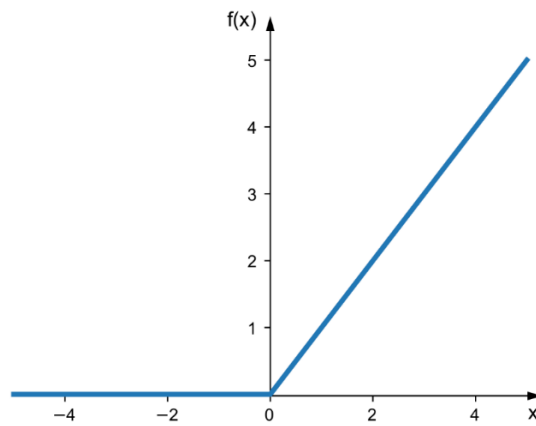


Figure 2.4: Positive Linear Transfer Function [37]

The Softmax function is mostly used for classification problems in the output layer. It is used to compute probability distribution from a vector of real numbers. The Softmax function produces an output which is a range of values between 0 and 1, with the sum of the probabilities equal to 1. This function is defined in equation (2.7).

$$\text{softmax}(n) = \frac{e^{n_i}}{\sum_j e^{n_j}} \quad (2.7)$$



## CHAPTER 3: Atrial Fibrillation Automatic Detection

---

In this chapter, I will explain the details of the system that I used to solve the problem at hand. The chapter starts with sections that explain the tools and dataset used in this project. Next, a section that explains the details and steps followed in the methodology.

### 3.1 Tools

Since I have a prior experience with MATLAB, I chose to code my system using MATLAB R2020a. All experiments were run on a 64-bit based laptop with Intel Core i7 CPU at 2.7 GHz, 2 cores, and 4 logical processors. The system runs 8 GB RAM.

### 3.2 Data

The dataset used for this work is the same dataset used in Physionet Challenge 2017. I used the training set which contains a collection of 8528 ECG recordings. The dataset has an unbalanced number of each class as follows: 5154 normal rhythms, 2557 other rhythms, 771 atrial fibrillations, and 46 noisy signal records. The signals' waveforms can be seen in figure 3.1. The recordings are labeled by an outsourced company and provided by AliveCor with the data.

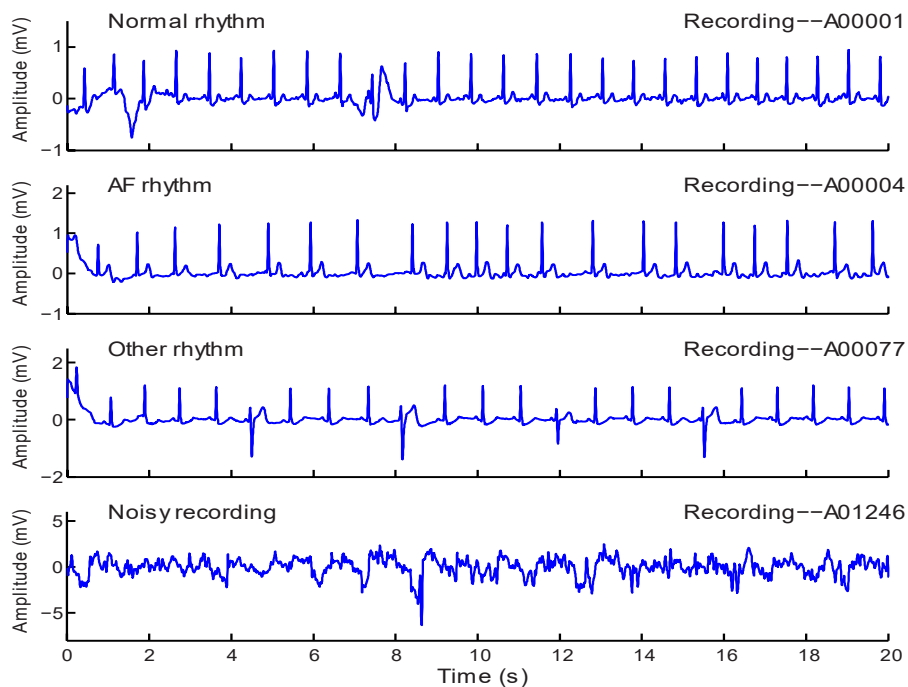


Figure 3.1: The ECG Signal of the Four Classes [38]

The data was found to have some recordings that are labeled as normal, atrial fibrillation, or other rhythm are being so noisy and hard to be identified by eye. So, the data has been re-labeled again by the challenge organizing team until they end up with version 3 of the dataset. The third version of the dataset has 5076 Normal Rhythm, 2415 Other Rhythm, 758 Atrial Fibrillation, and 279 Noisy Signals.

I used the challenge training dataset for training, validation, and testing by partitioning it into three parts: 70% for training, 15% for validation, and 15% for testing. The data division can be seen in table 3.1 is after using a partitioning function that divides the data by blocks of indices.

Table 3.1: Data Division

<b>Subset</b>	<b>Total</b>	<b>Normal Rhythm</b>	<b>Other Rhythm</b>	<b>Atrial Fibrillation</b>	<b>Noisy Signal</b>
<b>Training</b>	<b>5970</b>	3614	1652	512	192
<b>Testing</b>	<b>1279</b>	724	379	130	46
<b>Validation</b>	<b>1279</b>	738	384	116	41

### 3.3 Methodology

The proposed system has five stages: pre-processing, features extraction, features selection, classifier training, and results in the evaluation. The block diagram of the system is shown in figure 3.2.

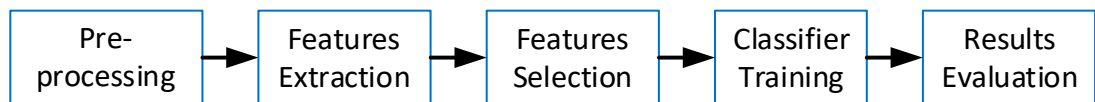


Figure 3.2: The Proposed System Block Diagram

Based on this diagram, I developed a MATLAB program which is attached in appendix A.

### 3.3.1 Pre-processing

For the first stage of the system, I used the approach developed in [10]. The program performs noise cancelation on the signal. Noise removal is needed because the ECG signal is prone to noise that is caused by several sources like breathing, improper contact of electrodes, or body movement. The approach is based on spectrogram analysis to identify the noisy parts in the signal. First, it computes the spectrogram of the signal. Next, it searches for the regions between successive RR intervals that have more than 50 Hz spectral power because all-important cardiac information is stored within 20 Hz. Then, the baseline movement is removed using a high pass filter with a cut-off frequency of 0.5 Hz.

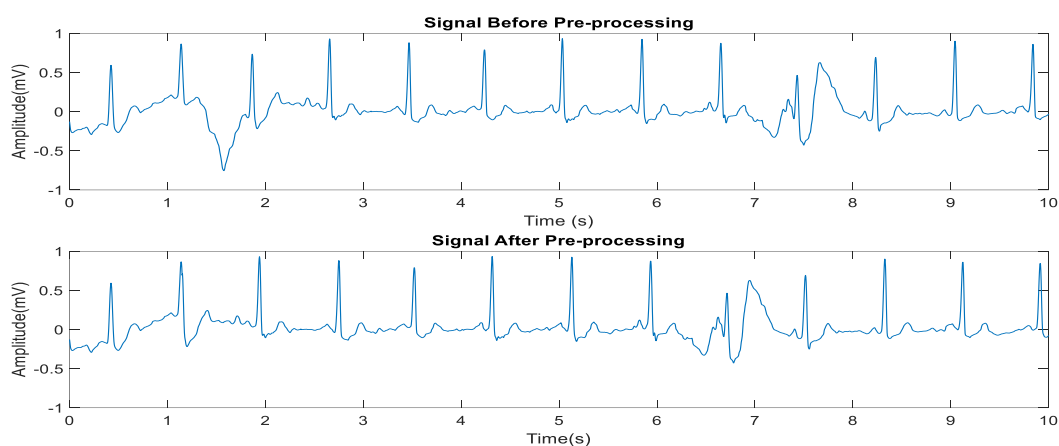


Figure 3.3: Signal Before and After Pre-processing

### 3.3.2 Features Extraction

The function of the second stage is to extract the features from the basic ECG features: the peaks, the segments, and the intervals. I used the approach used in [10] which extracts 188 features categorized as follows: morphological features, prior art AF features, HRV features, frequency features, statistical features, other abnormalities features, and detecting noisy recording features.

**Morphological Features** are the features extracted from the peaks P, Q, R, S, and T of the ECG waveform. These features are used usually by medical staff for identifying cardiac abnormalities. Morphological features include measuring the median, variance, and range of different aspects like the corrected QT interval (QTc), QR and QRS widths, slopes of QR, RS and ST intervals, depth of the Q and S points concerning R, amplitude difference of the TR wave, ratio of the number of P waves to the number of R waves and distance of the ST segment crossing from the S point.

**Prior Art AF Features** are features available in the prior art and can identify atrial fibrillation events. RR irregularity is an important feature of atrial fibrillation and there are several similar features. Some of these features are AF Evidence, Original Count, Irregularity Evidence, Pace Count, Density Evidence, Anisotropy Evidence, AF Evidence from Lorentz plot of RR intervals. Also, there are features derived from the inter-beat intervals using Poincare plots. Other features include approximate and sample entropy-based features and coefficient of variation of RR and delta RR intervals.

**HRV Features** are features related to heart rate variability. HRV features are like the number of RR intervals above  $x$ , normalized by duration of recording, where  $x$  lies between 20 and 500 ms (pRR $x$ ). Other features are the standard deviation of RR intervals (SDRR), the standard deviation of RR difference, and the normalized root means square of successive differences (RMSSD). Also, the normalized spectral power of the RR interval time series within the frequency region of 0- 0.04 Hz, 0.04-0.15 Hz, and 0.15-0.5 Hz are used as features.

**Frequency Features** are also important in this work, and they are extracted in a process where raw time signal is divided into small windows of 2 seconds duration having 50% overlapping using hamming window. Then, the Short Time Fourier Transform of each window is computed for frequency analysis. The extracted features are mean spectral centroid, spectral roll-off, spectral flux, and normalized spectral power between 0-10 Hz and 10-20 Hz across all windows in a measurement.

**Statistical Features** are features defined and calculated through statistical analysis. These features include the mean, median, variance, range, kurtosis, and skewness of RR intervals and the probability density estimate (PDE) of the RR intervals and the delta RR intervals. Other features in this category are the number of peaks on the PDE of the RR and delta RR intervals and the variation of energy in between the RR peaks. Also, the list of statistical features includes The Shannon, Tsallis, and Renyi entropy, Linear Predictive Coefficients (LPC) of the raw time-series data.

**Other Abnormalities Features** are used to distinguish atrial fibrillation from other abnormalities. These features are extracted using a sliding window with six peaks per window and its average RR interval, maximum of the first difference of some samples in the window with a magnitude exceeding 0.1mV, the normalized power spectrum density (nPSD) of the window. The heart rate was estimated using an adaptive frequency tracking

algorithm to derive features like mean of RR interval, a decrease of HR, maximum SPI index, average HR, abnormal HR, and others.

**Detecting Noisy Recording Features** are features used to detect noise and motion artifacts in the different portions of the signals. This part of the feature extract uses domain-dependent time and frequency features along with certain statistical features that exploit the rise and fall in the morphology of the ECG signal. These features distinguish well between the regularities of the clean ECG signal versus the randomness in a noisy waveform.

### 3.3.3 Features Selection

For feature selection, three different methods are used: ReliefF algorithm, Chi-square test, and minimum Redundancy Maximum Relevance (mRMR) algorithm. The proposed system starts by applying each of the algorithms to the features extracted. The algorithms work to give each feature a weight according to its importance in predicting the output and so the features are ranked starting with the most important feature ending with the lowest importance.

To rank the features in this work, I used existing functions in MATLAB which are `relieff`, `fscmrmr`, and `fscchi2`. `relieff` function is an implementation for the ReliefF algorithm. The function takes the features matrix, target output, and the number of nearest neighbors as input. The output of the `relieff` function is the indices of the ranked features and the weight which is in the range -1 and 1. For the mRMR algorithm, `fscmrmr` function was used. The function takes the features matrix and the target output as input. The output of the function is the indices of the ranked features and the score of each feature. Finally, `fscchi2` function was used as an implementation for the Chi-square algorithm. The function takes the features matrix and the target output as input. The output of the function is the indices of the ranked features and the scores for each feature. The score is  $-\log(p)$ , and  $p$  is a small value of the test that indicates the dependence of the feature with the target.

The next step is to choose the best number of features to be used for each algorithm before the training. For this step, I calculated the differences in weights between the ranked features. Then, I specified a threshold by intuition and experiment that should not be exceeded by the difference. The threshold I choose is the mean of the differences that I found to be working for the mRMR algorithm and ReliefF algorithm. For the Chi-square

algorithm, some features had an infinite value which indicates higher importance, and so I chose to take all the infinite weighted features as the selected features.

### 3.3.4 Training Neural Network

I started with a basic simple network that consists of the input layer, output layer, and 2 hidden layers. For initializing the neural network, I used the patternnet function to create a neural network for classification. The input layer has neurons equal to the number of features used. The output layer has four output neurons, one for each of the four classes (Normal rhythm, Atrial fibrillation, another rhythm, Noisy signal). I used the soft-max transfer function for the output layer and that was kept unchanged across the whole work. I started the work with two hidden layers each having 20 neurons and using hyperbolic tangent transfer function and rectified linear unit transfer function, respectively. The initial neural network architecture is shown in figure 3.4.

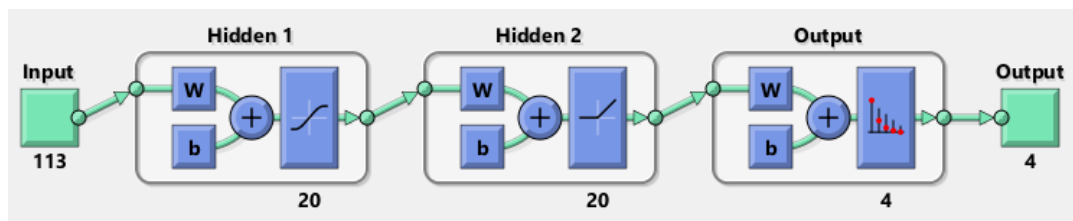


Figure 3.4: Neural Network for 113 Input Features

The network was initialized with the following parameters:

- Maximum number of epochs to train = 1000
- Performance goal = 0
- Maximum time to train in seconds = 300
- Minimum performance gradient = 0
- Maximum validation failures = 10
- Learning rate = 0.01
- Increment to weight change = 1.2
- Decrement to weight change = 0.5
- Initial weight change = 0.07
- Maximum weight change = 50.0
- Regularization parameter = 0.1
- Bias = 0.1 for ReLU layer

The number of neurons in each hidden layer was tuned several times until the best results were found. I started the training using the Resilient backpropagation algorithm as it works best for classification.

After initializing the neural network, the next step was to train it using the chosen ranked features. The objective function used is cross-entropy and the goal was to minimize it. After training, the network was tested with the test dataset and the achieved output is compared with the target data. The process of initialization, training, and finding the results were repeated 30 times for each set of experiments. Then, the average of the results was taken.

### 3.3.5 Results Evaluation

To evaluate the results, I used four metrics: recall, precision, accuracy, and F1 measure. To calculate these metrics, four other values should be calculated: true positive, true negative, false positive, and false negative. These values determine the correctness of the classification. True positive is the correctly classified samples as a positive class while true negative is the correctly classified samples as a negative class. On the other hand, false-positive is the sample that is wrongly classified as positive and false negative is the sample that is wrongly classified as negative. The four values together form the confusion matrix as shown in table 3.2 [39].

Table 3.2: Confusion Matrix

Data Class		Predicted by Classifier	
		Positive	Negative
Marked by Human Expert	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

After calculating these four values of the confusion matrix, I calculate the four metrics that evaluate the classification. The recall or sensitivity is a measure of how well the system correctly classifies the positive samples and it is calculated by equation (3.1):

$$Recall = \frac{TP}{TP + FN} \quad (3.1)$$

Precision is the ability of the system to avoid the wrong predictions and it is calculated using equation (3.2):

$$Precision = \frac{TP}{TP + FP} \quad (3.2)$$

Accuracy is the overall performance of classification. It is a measure that depends highly on the number of samples in each class and requires a balanced dataset. Since the dataset has an unbalanced number of samples in each class, the accuracy was not a good measure to evaluate the system, the reason why the F1 measure was needed. Accuracy is calculated using equation (3.3):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.3)$$

F1 measure, F score, or simply F measure is the harmonic mean of the recall and the precision which solves the problem of evaluating a system with an unbalanced dataset. This measure has been used in PhysioNet Challenge 2017 to evaluate the works proposed and it is calculated as follows:

$$F1\ measure = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (3.4)$$

In this work, I wrote a piece of program to first count the four elements of the confusion matrix for the four classes, so at the end four confusion matrices were found, one for each class. Next, the confusion matrices were used to calculate the four metrics of performance for each class. In the end, four performance measures for each of the four classes were calculated.



## CHAPTER 4: RESULTS AND ANALYSIS

---

To solve this problem, I followed three different approaches. The first approach is just using a deep neural network to detect atrial fibrillation among four classes. As explained before, the dataset used is unbalanced, which also makes it a challenge to be solved. To solve the challenge of an unbalanced dataset, I followed two more approaches: using a weighted neural network and using the method of under-sampling of the dataset.

### 4.1 Deep Neural Network

For the first approach, I used a deep neural network. I started the experiments by first selecting the features to be used for classification. Then, I used a deep neural network fed with the whole dataset without removing any sample from it to classify the samples into four classes. I compared three groups of features chosen by three different algorithms.

#### 4.1.1 Feature Selection

First, the three different ranking algorithms: mRMR, Chi-square, and ReliefF algorithm are explored. The algorithms gave different rankings for the features. The features have been given a weight using the algorithms and accordingly they were ranked starting by the feature having the highest weight to be the first down to the lowest at rank 188. The feature ranking using the ReliefF algorithm is shown in figure 4.1.

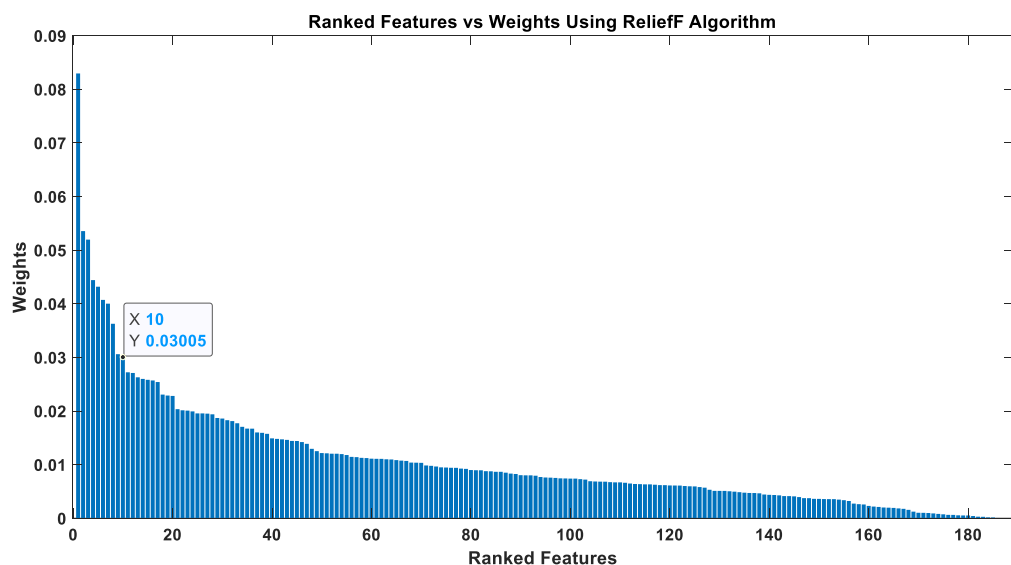


Figure 4.1: Ranking of Features using ReliefF Algorithm

After ranking the features, I calculated the difference of the weights and found the mean to be at point 10 which means the first 10 ranked features have a weight that satisfies the threshold and hence chosen for the next step.

The selected features using the ReliefF algorithm are as the following:

1. Bradycardia binary feature
2. Tachycardia binary feature
3. Number of RR intervals above 20 ms, normalized by duration of recordingBasic ratio of the difference in P location difference to R location difference
4. P to R ratio
5. Nonlinear HRV feature of short-range scaling exponent alpha from detrended fluctuation analysis
6. Maximum heart rate in a segment
7. Irregularity evidence
8. Areas relative to the total area within the frequency bands 0-2
9. Areas relative to the total area within the frequency bands 10-150

The same procedure is repeated with the mRMR algorithm. mRMR algorithm has a different behavior where two features have the highest weights and all other features have a weight that is comparably very low. The ranking of features using the mRMR algorithm is shown in figure 4.2.

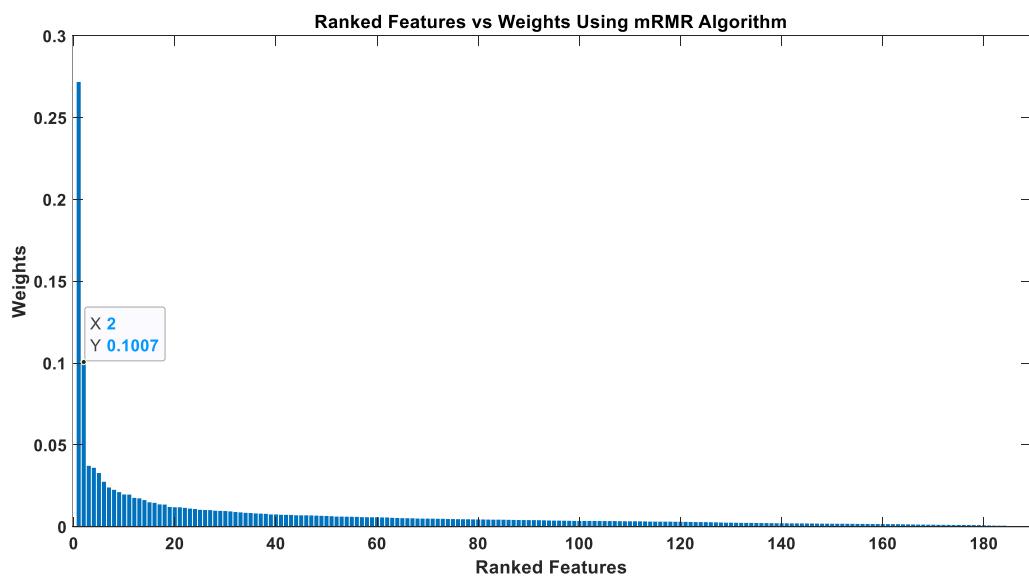


Figure 4.2: Ranking of Features using mRMR Algorithm

As shown in figure 4.2, the two features have high weights compared with the others and it is also proved by the mean of differences. The two features are:

1. Mean Stepping Increment of Inter Beat Intervals
2. The mean of RR interval

The last selection algorithm is Chi-square. The same procedure has been repeated using the Chi-square test and the ranking of the features is shown in figure 4.3. As shown in the figure, the weights are widely varying. 45 of the features had weights that reach infinity which means their effect on the output is very high. Without calculating the differences and taking the average, I chose to take the 45 features that have infinity weights.

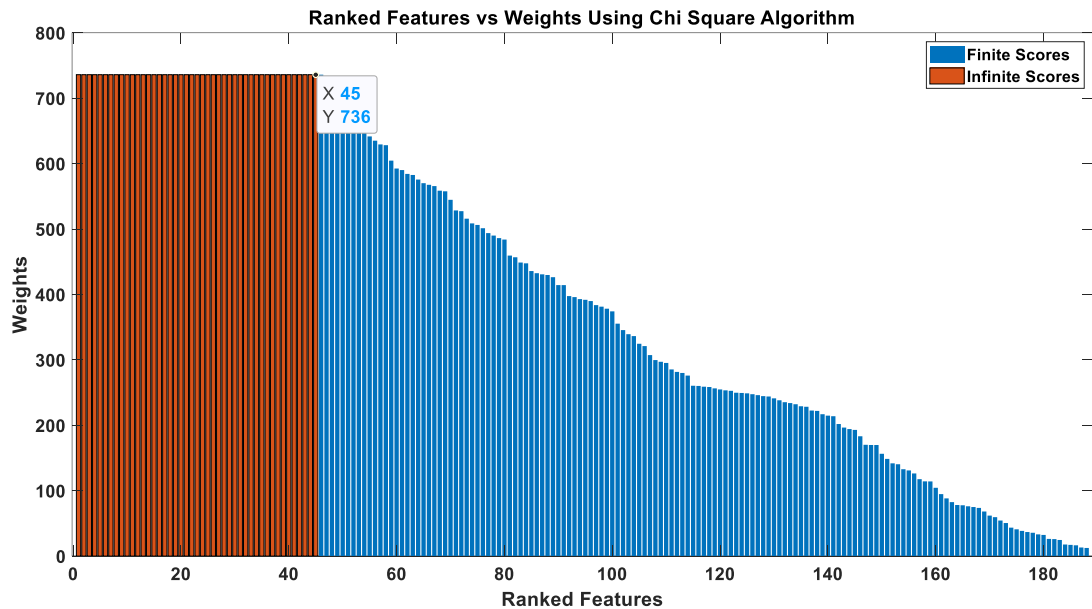


Figure 4.3: Ranking of Features using Chi-Square Algorithm

The top ten of the 45 infinity-weighted features using the Chi-Square selection tool are the following:

1. AF Evidence
2. Number of points in the bin containing the Origin
3. Irregularity Evidence
4. Density Evidence
5. Anisotropy Evidence
6. Coefficient of variation of RR
7. Coefficient of variation of  $\Delta RR$
8. Mean Stepping Increment of Inter Beat Intervals

9. Dispersion of points around the diagonal line in Poincare Plots
10. The median of RR interval

At the end of this step, I had 3 groups of features, one group for each selection algorithm. 2 features resulted from the mRMR algorithm, 10 features resulted from the ReliefF algorithm, and 45 features resulted from the Chi-Square algorithm. All these groups of features are used for the next step.

#### 4.1.2 Tuning Network

In this step, I used the 3 groups of features to tune the neural network. I started by using features selected by the mRMR algorithm, I tried to tune the number of layers and neurons to minimize the cross-entropy and find the best network that can classify the data. Since there is no rule of thumb to tune the number of layers and neurons, I started with a simple 2-layer network with 20 neurons in each layer. I trained the network 30 times and took the average. Then, I added one more layer and doubled the number of neurons in each layer, and trained the network again. I performed 5 experiments as can be seen in table 4.1. I had to make a tradeoff between the performance and the elapsed time consumed for training. For this reason, I only had 5 layers with 160 neurons. The table shows the average F1 score for three classes after 30 iterations, the overall F1 performed on the training dataset, and the testing dataset for features extracted using the mRMR algorithm.

Table 4.1: Results of Deep Neural Network using mRMR Algorithm

# Layers	# Neurons per layer	AF		Normal		Other		Overall F1
		F1	std	F1	std	F1	std	
<b>Training</b>								
2	20	0.328	0.291	0.724	0.329	0.543	0.058	0.531
3	40	0.371	0.287	0.874	0.006	0.585	0.022	0.610
4	80	0.296	0.301	0.868	0.020	0.554	0.082	0.572
5	160	0.263	0.286	0.836	0.046	0.408	0.224	0.502
<b>Testing</b>								
2	20	0.308	0.275	0.715	0.325	0.528	0.043	0.517
3	40	0.341	0.264	0.857	0.008	0.549	0.024	0.582
4	80	0.271	0.276	0.853	0.022	0.520	0.082	0.548
5	160	0.256	0.279	0.820	0.055	0.393	0.216	0.490

The same process is performed using features extracted using the ReliefF algorithm. I started with a simple 2-layer network and ended up with a 5-layer network with 160 neurons in each layer. Table 4.2 shows the resulting F1 score for three classes and the overall F1.

Table 4.2: Results of Deep Neural Network using ReliefF Algorithm

# Layers	# Neurons per layer	AF		Normal		Other		Overall F1
		F1	std	F1	std	F1	std	
<b>Training</b>								
2	20	0.697	0.190	0.890	0.002	0.688	0.013	0.758
3	40	0.763	0.209	0.905	0.004	0.742	0.024	0.804
4	80	0.629	0.287	0.886	0.030	0.674	0.126	0.730
5	160	0.407	0.339	0.868	0.042	0.589	0.201	0.621
<b>Testing</b>								
2	20	0.671	0.183	0.875	0.004	0.664	0.014	0.737
3	40	0.650	0.178	0.875	0.003	0.660	0.015	0.728
4	80	0.571	0.262	0.864	0.032	0.631	0.115	0.689
5	160	0.360	0.305	0.847	0.045	0.559	0.191	0.589

The last set of experiments in this section were conducted using the Chi-Square algorithm and the 45 features selected by this algorithm. Also, here I started with a simple 2-layers network. The results are shown in table 4.3.

Table 4.3: Results of Deep Neural Network using Chi-Square Algorithm

# Layers	# Neurons per layer	AF		Normal		Other		Overall F1
		F1	std	F1	std	F1	std	
<b>Training</b>								
2	20	0.599	0.325	0.881	0.005	0.670	0.025	0.717
3	40	0.606	0.340	0.880	0.005	0.665	0.034	0.717
4	80	0.735	0.200	0.880	0.005	0.670	0.026	0.762
5	160	0.582	0.306	0.853	0.041	0.541	0.224	0.659
<b>Testing</b>								
2	20	0.571	0.315	0.858	0.005	0.639	0.026	0.689
3	40	0.584	0.328	0.859	0.003	0.637	0.027	0.693
4	80	0.711	0.194	0.860	0.005	0.639	0.022	0.737
5	160	0.554	0.296	0.833	0.046	0.520	0.217	0.636

Based on the obtained results using these three feature selection algorithms, it is found that the ReliefF and Chi-Square algorithms outperform the mRMR approach. Comparing the three algorithms, it can be seen clearly in figure 4.4 that ReliefF and Chi-Square algorithms gave better F1 score values than the mRMR algorithm since it is using only 2 features. Yet, these numbers are not good enough for the system to identify atrial fibrillation. For this reason, I had to go for other approaches.

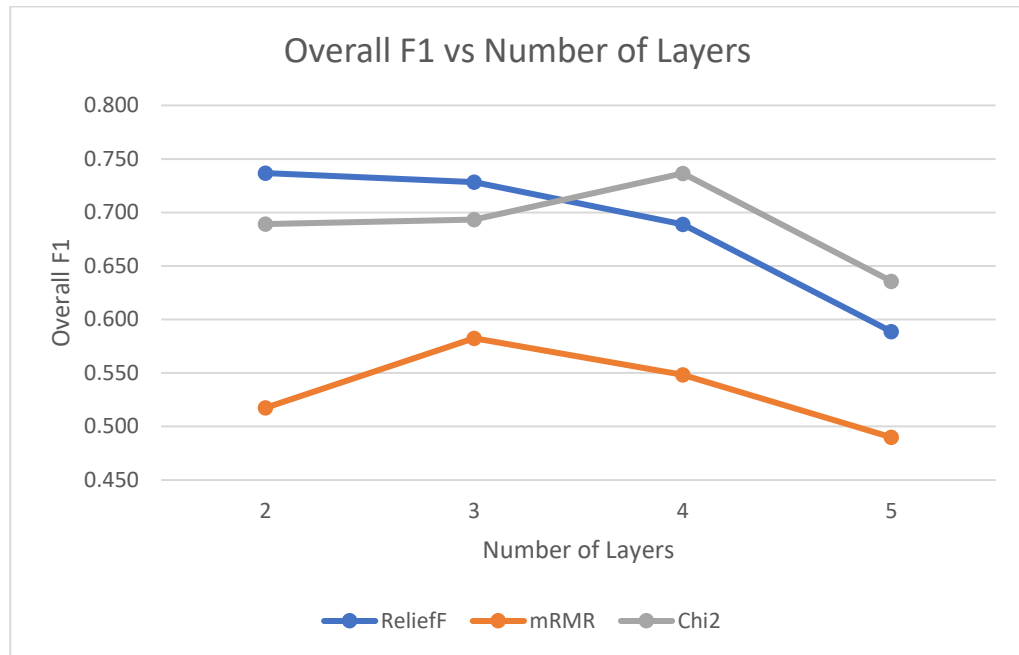


Figure 4.4: Comparison of Overall F1 using 3 Feature Selection Algorithms

## 4.2 Weighted Neural Network

The dataset I used for this project is unbalanced which made it a challenge in detecting atrial fibrillation among the other classes. After comparing multiple state-of-the-art approaches, I found that one way to solve the issue is by assigning weights to the classes. Since atrial fibrillation is the subject of my study and it is considered a minority class, I gave it a high weight compared to the other classes. The weights assigned to the classes are as follows: atrial fibrillation = 1, normal rhythm = 0.1, other rhythm = 0.3, and noisy signal = 0. For this set of experiments, I used trial and error to tune the parameters and hyperparameters in the network. I found that the initial values work well for this step, so I kept them unchanged.

### 4.2.1 Tuning Network

For tuning the number of layers and number of neurons in each layer, I used the same approach as in the previous step. I started with a 2-layers network with 20 neurons in each

layer. I repeated the experiment 30 times and calculated the average of the F1 score for all classes. Then I added one more layer and doubled the number of neurons in each layer. I started by using the features extracted using the ReliefF algorithm and the results are shown in table 4.4.

Table 4.4: Results of Weighted Neural Network using ReliefF Algorithm

# Layers	# Neurons per layer	AF		Normal		Other		Overall F1
		F1	std	F1	std	F1	std	
<b>Training</b>								
2	20	0.689	0.008	0.851	0.003	0.644	0.003	0.728
3	40	0.762	0.012	0.868	0.003	0.698	0.006	0.776
4	80	0.710	0.030	0.832	0.157	0.663	0.049	0.735
5	160	0.626	0.176	0.743	0.297	0.617	0.140	0.662
<b>Testing</b>								
2	20	0.655	0.010	0.831	0.003	0.622	0.005	0.702
3	40	0.685	0.014	0.848	0.004	0.655	0.009	0.729
4	80	0.665	0.026	0.811	0.153	0.629	0.040	0.702
5	160	0.589	0.162	0.725	0.289	0.588	0.127	0.634

Next, I used the features extracted using the mRMR algorithm. I followed the same procedure. The results obtained at this stage are shown in table 4.5.

Table 4.5 Results of Weighted Neural Network using mRMR Algorithm

# Layers	# Neurons per layer	AF		Normal		Other		Overall F1
		F1	std	F1	std	F1	std	
<b>Training</b>								
2	20	0.510	0.003	0.824	0.009	0.371	0.019	0.568
3	40	0.555	0.020	0.794	0.150	0.516	0.035	0.622
4	80	0.547	0.021	0.796	0.151	0.508	0.030	0.617
5	160	0.557	0.022	0.761	0.207	0.507	0.044	0.609
<b>Testing</b>								
2	20	0.579	0.010	0.825	0.008	0.428	0.031	0.611
3	40	0.604	0.014	0.778	0.147	0.524	0.023	0.635
4	80	0.602	0.016	0.780	0.148	0.522	0.022	0.635
5	160	0.610	0.017	0.749	0.204	0.521	0.033	0.626

Finally, I used the features extracted using the Chi-square algorithm to see how they perform following the same procedure. The results of the weighted neural network using features extracted using the Chi-square algorithm are shown in table 4.6.

Table 4.6 Results of Weighted Neural Network using Chi-Square Algorithm

# Layers	# Neurons per layer	AF		Normal		Other		Overall F1
		F1	std	F1	std	F1	std	
<b>Training</b>								
2	20	0.715	0.006	0.831	0.003	0.626	0.009	0.724
3	40	0.732	0.019	0.840	0.010	0.661	0.023	0.744
4	80	0.712	0.022	0.833	0.014	0.639	0.025	0.728
5	160	0.669	0.141	0.775	0.211	0.589	0.161	0.678
<b>Testing</b>								
2	20	0.728	0.011	0.815	0.006	0.618	0.006	0.720
3	40	0.737	0.011	0.820	0.011	0.649	0.021	0.735
4	80	0.725	0.017	0.814	0.013	0.629	0.018	0.723
5	160	0.679	0.136	0.755	0.206	0.575	0.158	0.670



At the end of these experiments, I compared the overall F1 score of the three algorithms. As can be seen in figure 4.5, the features extracted using the Chi-Square algorithm outperform the other features by having a higher F1 score. However, these results are not good enough for the network to detect atrial fibrillation. Moreover, I wanted to try the second approach and compare how it performs when compared with this approach.

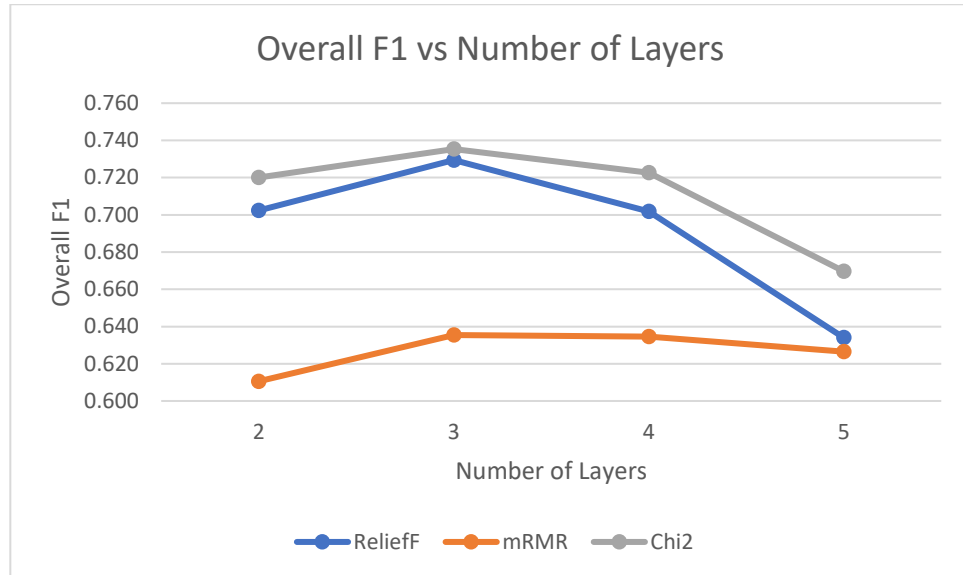


Figure 4.5 Comparison of Overall F1 on Weighted Neural Network

### 4.3 Under-sampling Dataset

The second approach for solving the unbalanced dataset challenge is under-sampling. For this approach, I under-sampled the dataset by removing samples from the majority classes which are in this case the normal rhythm class and the other rhythms class. The new dataset has 846 samples in the normal rhythm class, 805 samples in the other rhythm class, 758 samples in the atrial fibrillation class, and 279 samples in the noisy signal class. The total number of observations in the new under-sampled dataset is 2688 records. In this set of experiments, I also used the same parameters and hyperparameters in the network after trial-and-error tuning.

#### 4.3.1 Feature Selection

Before training, I started by feature selection because of the dataset change. I applied the three algorithms to select other features based on the new under-sampled dataset. As can be seen in figure 4.6, the number of features selected by the mRMR algorithm is now 6 features.

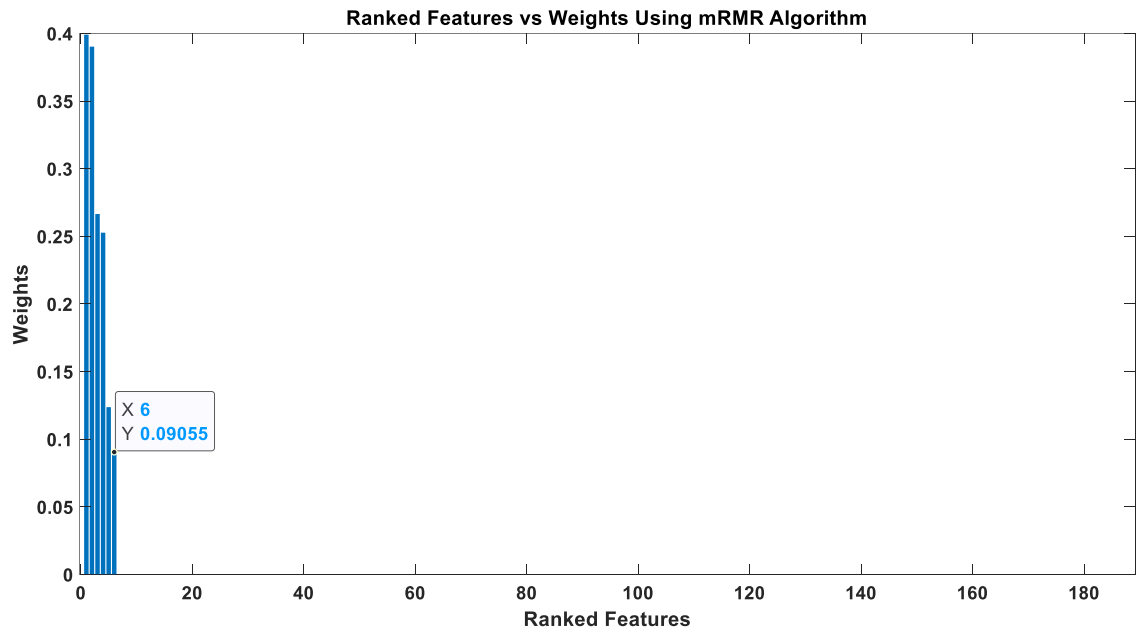


Figure 4.6 Ranking of Features using the mRMR Algorithm from Under-Sampled Dataset

The selected features using the mRMR algorithm from the under-sampled dataset are different from the features selected from the original dataset, and these features are:

1. Median of absolute difference of heartrate
2. Sample entropy estimates
3. Frequency feature
4. Tsallis Entropy
5. The basic ratio of the median of difference of R location
6. Spectral Centroid

On the other hand, the features selected by the ReliefF algorithm are almost identical with a small variation in the number wherefrom the under-sampled dataset I got 11 features. Figure 4.7 shows the ranking of the features using the ReliefF algorithm, which looks similar to the figure of ranked features on the original dataset.

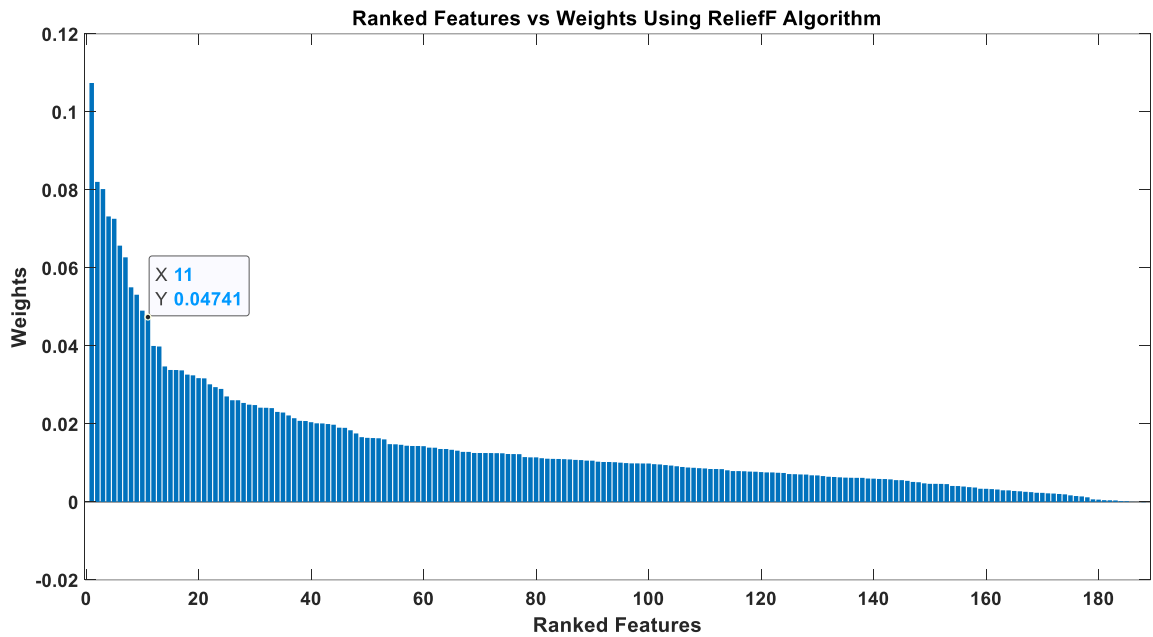


Figure 4.7 Ranking of Features using ReliefF Algorithm from Under-Sampled Dataset

The features selected using the ReliefF algorithm are the following:

1. Bradycardia binary feature
2. The basic ratio of the difference in P location difference to R location difference
3. P to R ratio
4. Number of RR intervals above 20 ms, normalized by duration of recording
5. Standard deviation in R location after removing outliers
6. Irregularity Evidence
7. Tachycardia binary feature
8. Maximum heart rate in the segment
9. Short-range scaling exponent
10. AF Evidence
11. Minimum of RR

Finally, the features selected using the Chi-Square algorithm are also different than the ones selected from the original dataset. From the modified dataset, I found only 6 features having infinity weight and those were selected for the next stage. The 6 features can be seen in figure 4.8 as the red bars at the start of the graph.

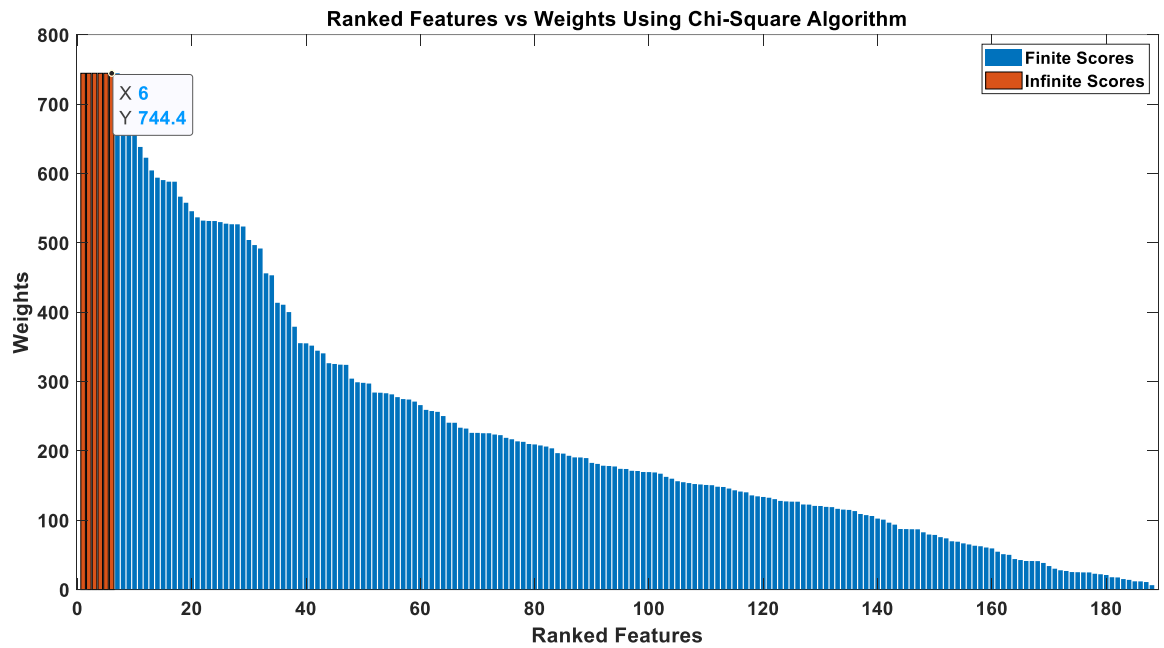


Figure 4.8 Ranking of Features using Chi-Square Algorithm from Under-Sampled Dataset

The 6 selected features are the following:

1. AF Evidence
2. Irregularity Evidence
3. Coefficient of variation of delta RR
4. Mean Stepping Increment of Inter Beat Intervals
5. The standard deviation of Empirical Mode Decomposition
6. Median of absolute difference of heartrate

### 4.3.2 Tuning Network

For tuning the network, I followed the same procedure used in the previous steps. I started with a 2-layers network with 20 neurons in each layer. Then I added one more layer and doubled the number of neurons in all layers. Starting with the features extracted using the ReliefF algorithm, I calculated the F1 score for each class for 30 iterations and took the average of 30. The resulted F1 score using the ReliefF algorithm is shown in table 4.7. The table shows improvement in the results than the results obtained in the previous stages.

Table 4.7: Results of Under-Sampling using ReliefF Algorithm

# Layers	# Neurons per layer	AF		Normal		Other		Overall F1
		F1	std	F1	std	F1	std	
<b>Training</b>								
2	20	0.873	0.008	0.826	0.004	0.712	0.010	0.803
3	40	0.889	0.024	0.839	0.015	0.740	0.030	0.823
4	80	0.794	0.162	0.667	0.307	0.636	0.083	0.699
5	160	0.779	0.102	0.696	0.267	0.591	0.180	0.689
<b>Testing</b>								
2	20	0.807	0.014	0.793	0.009	0.665	0.020	0.755
3	40	0.797	0.016	0.778	0.013	0.644	0.023	0.740
4	80	0.757	0.151	0.637	0.294	0.610	0.063	0.668
5	160	0.758	0.106	0.667	0.255	0.576	0.170	0.667

The next algorithm was mRMR. Using the same procedure, I obtained the results shown in table 4.8.

Table 4.8: Results of Under-Sampling using mRMR Algorithm

# Layers	# Neurons per layer	AF		Normal		Other		Overall F1
		F1	std	F1	std	F1	std	
<b>Training</b>								
2	20	0.707	0.041	0.656	0.032	0.386	0.066	0.583
3	40	0.751	0.069	0.710	0.055	0.520	0.059	0.660
4	80	0.720	0.088	0.706	0.078	0.519	0.071	0.649
5	160	0.603	0.220	0.564	0.269	0.411	0.202	0.526
<b>Testing</b>								
2	20	0.677	0.053	0.641	0.041	0.338	0.085	0.552
3	40	0.700	0.065	0.655	0.052	0.456	0.061	0.604
4	80	0.686	0.071	0.645	0.072	0.452	0.078	0.595
5	160	0.576	0.190	0.512	0.244	0.357	0.179	0.482

Finally, the results using the features extracted using the Chi-square algorithm are shown in table 4.9.

Table 4.9: Results of Under-Sampling using Chi-Square Algorithm

# Layers	# Neurons per layer	AF		Normal		Other		Overall F1
		F1	std	F1	std	F1	std	
<b>Training</b>								
2	20	0.834	0.009	0.751	0.010	0.556	0.020	0.714
3	40	0.846	0.013	0.766	0.013	0.601	0.028	0.738
4	80	0.815	0.080	0.730	0.139	0.531	0.182	0.692
5	160	0.777	0.147	0.739	0.048	0.547	0.105	0.687
<b>Testing</b>								
2	20	0.772	0.010	0.744	0.027	0.575	0.039	0.697
3	40	0.781	0.017	0.747	0.015	0.594	0.028	0.707
4	80	0.758	0.062	0.708	0.136	0.535	0.183	0.667
5	160	0.705	0.135	0.735	0.056	0.551	0.107	0.664

Comparing the Overall F1 obtained using the three algorithms shows that the ReliefF algorithm outperforms as seen in figure 4.9.

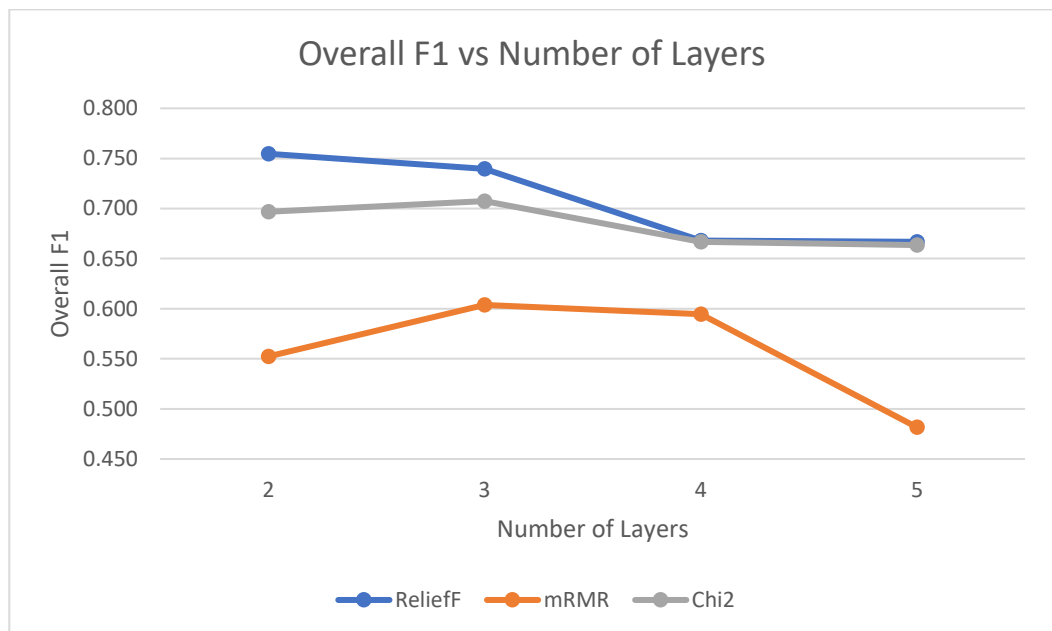


Figure 4.9: Comparison of Overall F1 on Under-sampled Dataset

At the end of this stage, I can conclude that using the ReliefF algorithm yields the best features for the network to identify atrial fibrillation. Also, using the under-sampling approach helps in improving the network performance and ability to distinguish the different classes. Moreover, the highest Overall F1 score was 82.3 % on the training dataset and was found using a network of 3 layers and 40 neurons in each layer. For the next step, I tune some parameters and hyperparameters to further improve the results.

#### **4.4 Changing Activation Functions**

For this stage, I have fixed the number of layers and the group of features to be used. For the next steps, I used the group of features extracted using the ReliefF algorithm from the balanced dataset and a network of 3 layers with 40 neurons in each layer. At this stage, I tried to measure the effect of changing the activation functions used in the 4-layers network. Along with changing the activation layer, I tuned the number of neurons and some parameters to achieve the best results possible. I used 5 different setups as follows:

1. ReLU for all layers. 80 neurons per layer, Regularization = 0.9
2. tanh for all layers. 80 neurons per layer, Regularization = 0.7
3. tanh for layer 1, ReLU for layers 2 and 3. 40 neurons per layer, Regularization = 0.7
4. tanh for layers 1 and 3, ReLU for layer 2. 40 neurons per layer, Regularization = 0.7
5. tanh for layers 1 and 2, ReLU for layer 3. 80 neurons per layer, Regularization = 0.5

The results shown in table 4.10 show that changing the activation function along with tuning the number of neurons and the regularization parameter gave good results. In addition, I was able to achieve networks that do not overfit the data. From these results, I fixed the activation functions as in setup 2 in which uses tanh in all layers and the number of neurons in each layer is set to 80 with a regularization parameter of 0.7.

Table 4.10: Changing Activation Functions

Setup	AF		Normal		Other		Overall F1
	F1	std	F1	std	F1	std	
<b>Training</b>							
1	0.824	0.026	0.806	0.009	0.649	0.028	0.760
2	0.857	0.015	0.816	0.009	0.694	0.019	0.789
3	0.823	0.007	0.801	0.004	0.645	0.005	0.756
4	0.837	0.016	0.809	0.007	0.662	0.018	0.769
5	0.870	0.014	0.823	0.011	0.712	0.022	0.802
<b>Testing</b>							
1	0.809	0.018	0.777	0.017	0.625	0.018	0.737
2	0.818	0.011	0.794	0.007	0.680	0.018	0.764
3	0.818	0.009	0.793	0.005	0.650	0.012	0.754
4	0.818	0.007	0.790	0.007	0.651	0.016	0.753
5	0.814	0.017	0.792	0.009	0.684	0.020	0.763

#### 4.5 Changing Learning Algorithm

After fixing the numbers in the previous stage, other learning algorithms were used to train the network. The algorithms that I used here, and their parameters that I tuned by trial and error to get the best results are as follows:

- **SCG:** Scaled conjugate gradient backpropagation, with parameters: Marquardt adjustment parameter=0.005, Change in weight for second derivative approximation=5.0e-5, Parameter for regulating the indefiniteness of the Hessian = 5.0e-7
- **RP:** Resilient backpropagation, with parameters: Learning rate=0.01, Increment to weight change=1.2, Decrement to weight change=0.5, Initial weight change=0.07, Maximum weight change=50
- **GD:** Gradient descent backpropagation, with parameters: Learning rate=1
- **GDA:** Gradient descent with adaptive learning rate backpropagation, with parameters: Learning rate=1, Ratio to increase learning rate=1.05, Ratio to decrease learning rate=0.7, Maximum performance increase=1.04
- **GDM:** Gradient descent with momentum, with parameters: Learning rate=5, Momentum constant=0.5



Table 4.11: Results Using Different Learning Algorithms for Training Network

Learning Algorithm	AF		Normal		Other		Overall F1
	F1	std	F1	std	F1	std	
<b>Training</b>							
SCG	0.845	0.061	0.780	0.165	0.674	0.122	0.766
RP	0.859	0.016	0.817	0.009	0.696	0.020	0.791
GD	0.808	0.220	0.796	0.151	0.688	0.051	0.764
GDA	0.765	0.209	0.762	0.145	0.639	0.047	0.722
GDM	0.802	0.218	0.788	0.149	0.674	0.039	0.755
<b>Testing</b>							
SCG	0.794	0.039	0.739	0.159	0.637	0.109	0.723
RP	0.816	0.016	0.797	0.007	0.683	0.017	0.765
GD	0.741	0.202	0.759	0.144	0.629	0.044	0.710
GDA	0.753	0.206	0.743	0.141	0.635	0.053	0.710
GDM	0.756	0.206	0.763	0.144	0.652	0.035	0.724

The results shown in table 4.11 show that the obtained results are in the range of 71-76%. They mainly differ in the time consumed for training. The highest results achieved when using the resilient backpropagation algorithm. The graph in figure 4.10 shows how these algorithms compared among themselves.

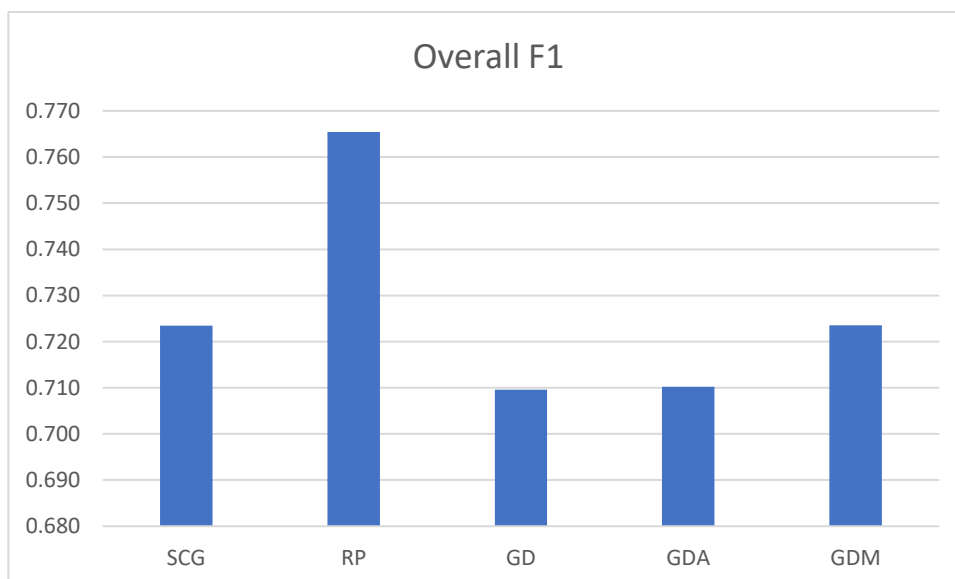


Figure 4.10: Comparing Learning Algorithms for Training Network

The best results achieved after all these experiments are shown in table 4.12. The table shows the F1 score, precision, recall, and accuracy for atrial fibrillation, normal rhythm,

and other rhythm classes. Also, the table shows the overall performance measures achieved from the test dataset.

Table 4.12: Best Achieved Results

Performance Measure	AF	Normal	Other	Overall
F1	0.816	0.797	0.683	0.765
Precision	0.774	0.801	0.722	0.766
Recall	0.864	0.794	0.650	0.769
Accuracy	0.885	0.883	0.819	0.862

Figure 4.11 shows how the performance changes to minimize the cross-entropy cost function at each epoch until it reaches the optimal value at epoch 116.

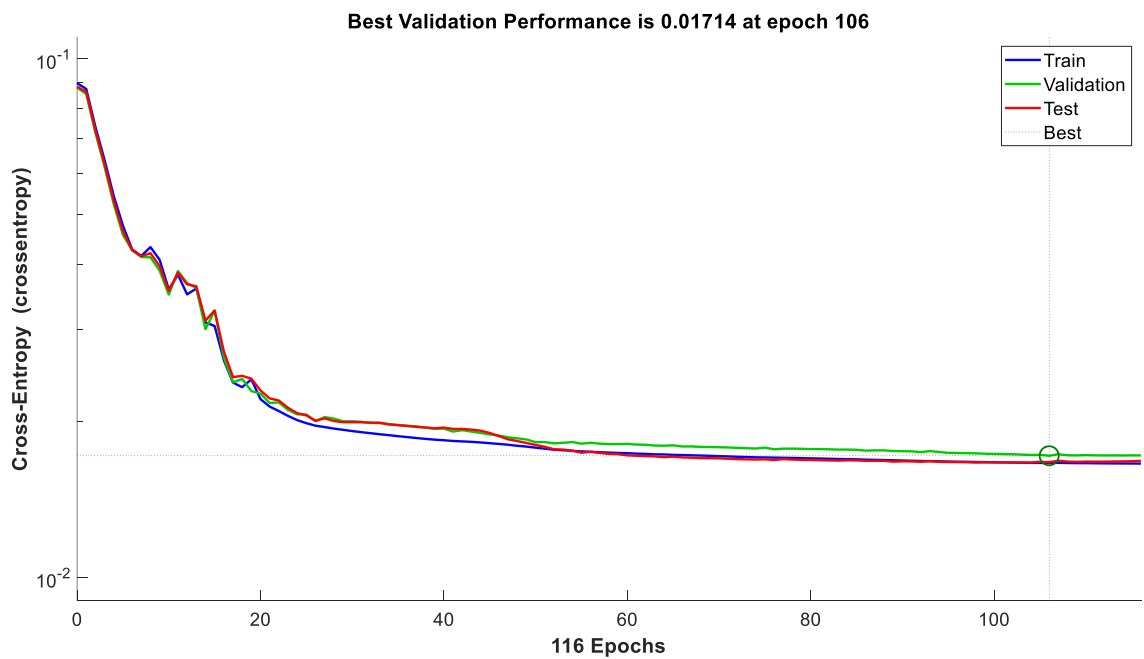


Figure 4.11 Changes in Performance with Each Epoch

These results are obtained using the following parameters and hyperparameters:

- 3 layers and 80 neurons in each layer
- Learning algorithm = Resilient backpropagation
- Maximum number of epochs to train = 1000
- Performance goal = 0

- Maximum time to train in seconds = 300
- Minimum performance gradient = 0
- Maximum validation failures = 10
- Learning rate=0.01
- Increment to weight change=1.2
- Decrement to weight change=0.5
- Initial weight change=0.07
- Maximum weight change=50
- Regularization parameter = 0.7
- Activation function = tanh for all layers

The network structure can be seen in figure 4.12. The network takes 11 features as input and gives 4 outputs representing the four classes.

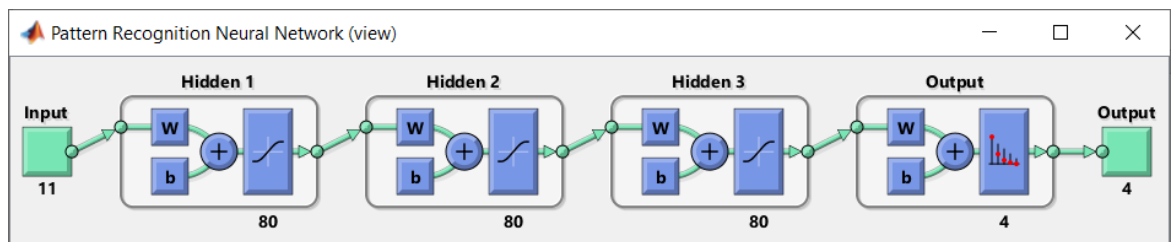


Figure 4.12: The Best Results Neural Network Structure

After some more statistical analysis of the results, I calculated the p-value for the results obtained from training and testing and found that  $p < 0.05$  which means that the results are statistically significant.

#### 4.6 Comparing with related work

To validate the developed neural network architecture, a qualitative comparison with the most related approaches was conducted. Table 4.13 shows the result from this work compared with the result from the works mentioned in the related work section. I compared the overall F1 score achieved by the proposed system with the overall F1 score achieved by related works. The results show that the work that has been done in this project compared

well with the previous works. However, I got slightly lower values because of the lower complexity of the system in comparison with the other models.

The result achieved in this project might not be as good as the results achieved in other works. However, in this work, I was merely trying to see how a simple feedforward neural network would compare with complex approaches like a convolutional neural network, recurrent neural network, and forest tree ensemble. The results of this work proves that a simple feedforward neural network can be a good classifier for such complex problems. The complexity of the system proposed in this project is light and fast than others and consumes less energy. It can be used for portable devices and real-time applications better than other techniques.

Table 4.13: Comparison of Result with Related Works

Method	F1-Score (%)
Proposed Neural Network	76.50
Datta et al.	79.00
Hong et al.	86.92
Teijeiro et al.	85.00
Zabihi et al.	79.43
Mahajan et al.	76.00

## CHAPTER 5: CONCLUSIONS & RECOMMENDATIONS

---

This thesis describes the details of the work I did to detect atrial fibrillation from ECG records. The thesis introduces the work with cardiology basics, description of the problem, and related work. A brief literature review about AF features, selection tools, and machine learning techniques is also included. The methodology is explained in detail followed by the results and analysis.

In this work, I used an artificial neural network to detect atrial fibrillation from ECG records. I extracted 188 features from the ECG records but not all of them were useful. I used different algorithms for feature selection and reduction to minimize computation time and maximize classification accuracy and these algorithms are the minimum redundancy maximum relevance (mRMR) algorithm, Chi-square tests, and ReliefF algorithm. I faced the challenge of the unbalanced dataset and tried to solve it by using the weighted neural network and by under-sampling the dataset to be almost balanced. Furthermore, I optimized the network by tuning the number of features, number of neurons in the hidden layers, number of layers, and parameters and hyper-parameters of the network.

Comparing the results achieved in this work with the most related approaches, I found that the developed neural network could achieve an overall F1 score of 76.5%, which is comparable with the results achieved by other researchers.

These experiments have been using a feedforward neural network with scaled conjugate gradient backpropagation for training the network. Usually, researchers use convolutional neural networks and recurrent neural networks to solve this kind of problem. Others also use forest tree and ensemble models, which perform very well on this kind of problem. However, I wanted to see how a simple feedforward neural network would perform against those approaches. The results show good numbers for a light system which makes it a good choice to be used in portable devices and real-time applications.

## REFERENCES

---

- [1] I. Kucybała, K. Ciuk, and W. Klimek-Piotrowska, “Clinical anatomy of human heart atria and interatrial septum - Anatomical basis for interventional cardiologists and electrocardiologists. Part 1: Right atrium and interatrial septum,” *Kardiol. Pol.*, vol. 76, no. 3, pp. 499–509, 2018.
- [2] C. Bianco, “How Your Heart Works Anatomy of the Heart,” *Heart*, 2003. [Online]. Available: <https://www.bhf.org.uk/information-support/how-a-healthy-heart-works>.
- [3] H. Anatomy, “Picture of the Heart,” 2018. .
- [4] S. H. Jambukia, V. K. Dabhi, and H. B. Prajapati, “Classification of ECG signals using machine learning techniques: A survey,” *Conf. Proceeding - 2015 Int. Conf. Adv. Comput. Eng. Appl. ICACEA 2015*, no. March, pp. 714–721, 2015.
- [5] J. Aspuru *et al.*, “Segmentation of the ECG signal by means of a linear regression algorithm,” *Sensors (Switzerland)*, vol. 19, no. 4, 2019.
- [6] National Institutes of Health, “Types of Atrial Fibrillation - NHLBI, NIH,” 2014. [Online]. Available: <https://patient.info/heart-health/atrial-fibrillation-leaflet>.
- [7] K. H. Kuck, “Atrial fibrillation,” *Herz -Munich-*, 2017. [Online]. Available: <https://www.nhlbi.nih.gov/health-topics/atrial-fibrillation>.
- [8] H. Prevent and A. Attack, “How to Help Prevent an AFib Attack,” 2021. [Online]. Available: <https://www.webmd.com/heart-disease/atrial-fibrillation/ss/slideshow-help-prevent-afib-attack>.
- [9] G. D. Clifford *et al.*, “AF classification from a short single lead ECG recording: The PhysioNet/computing in cardiology challenge 2017,” *Comput. Cardiol. (2010)*, vol. 44, pp. 1–4, 2017.
- [10] S. Datta *et al.*, “Identifying normal, AF and other abnormal ECG rhythms using a cascaded binary classifier,” *Comput. Cardiol. (2010)*, vol. 44, pp. 1–4, 2017.
- [11] S. Hong *et al.*, “ENCASE: An ENsemble CIASSifier for ECG classification using expert features and deep neural networks,” *Comput. Cardiol. (2010)*, vol. 44, pp. 1–

4, 2017.

- [12] T. Teijeiro, C. A. García, D. Castro, and P. Félix, “Arrhythmia classification from the abductive interpretation of short single-lead ECG records,” *Comput. Cardiol. (2010).*, vol. 44, pp. 1–4, 2017.
- [13] M. Zabihi, A. B. Rad, A. K. Katsaggelos, S. Kiranyaz, S. Narkilahti, and M. Gabbouj, “Detection of atrial fibrillation in ECG hand-held devices using a random forest classifier,” *Comput. Cardiol. (2010).*, vol. 44, pp. 1–4, 2017.
- [14] P. Cao *et al.*, “A novel data augmentation method to enhance deep neural networks for detection of atrial fibrillation,” *Biomed. Signal Process. Control*, vol. 56, p. 101675, 2020.
- [15] X. C. Cao, B. Yao, and B. Q. Chen, “Atrial Fibrillation Detection Using an Improved Multi-Scale Decomposition Enhanced Residual Convolutional Neural Network,” *IEEE Access*, vol. 7, pp. 89152–89161, 2019.
- [16] G. B. Moody and R. G. Mark, “MIT-BIH Atrial Fibrillation Database,” 1992. [Online]. Available: <https://physionet.org/content/afdb/1.0.0/><https://doi.org/10.13026/C2MW2D>.
- [17] J. Wang, P. Wang, and S. Wang, “Automated detection of atrial fibrillation in ECG signals based on wavelet packet transform and correlation function of random process,” *Biomed. Signal Process. Control*, vol. 55, p. 101662, 2020.
- [18] O. Faust, A. Shenfield, M. Kareem, T. R. San, H. Fujita, and U. R. Acharya, “Automated detection of atrial fibrillation using long short-term memory network with RR interval signals,” *Comput. Biol. Med.*, vol. 102, no. July, pp. 327–335, 2018.
- [19] K. Tateno and L. Glass, “Automatic detection of atrial fibrillation using the coefficient of variation and density histograms of RR and  $\Delta$ RR intervals,” *Med. Biol. Eng. Comput.*, vol. 39, no. 6, pp. 664–671, 2001.
- [20] A. Ghodrati, B. Murray, and S. Marinello, “RR interval analysis for detection of Atrial Fibrillation in ECG monitors,” *Proc. 30th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS’08 - "Personalized Healthc. through Technol.*, no. 1, pp. 601–604, 2008.

- [21] A. Ghodrati and S. Marinello, “Statistical analysis of RR interval irregularities for detection of atrial fibrillation,” *Comput. Cardiol.*, vol. 35, pp. 1057–1060, 2008.
- [22] L. Billeci, F. Chiarugi, M. Costi, D. Lombardi, and M. Varanini, “Detection of AF and other rhythms using RR Variability and ECG spectral measures,” *Comput. Cardiol. (2010)*., vol. 44, pp. 1–4, 2017.
- [23] C. C. Lin, C. Y. Yang, Z. Zhou, and S. Wu, “Intelligent health monitoring system based on smart clothing,” *International Journal of Distributed Sensor Networks*, 2018. [Online]. Available: [https://columbiasurgery.org/conditions-and-treatments/arrhythmiaatrial-fibrillation#:~:text=Atrial+Fibrillation-,Atrial+fibrillation+\(AF\)+is+a+form+of+arrhythmia%2C+or,+over+65+years+of+age.](https://columbiasurgery.org/conditions-and-treatments/arrhythmiaatrial-fibrillation#:~:text=Atrial+Fibrillation-,Atrial+fibrillation+(AF)+is+a+form+of+arrhythmia%2C+or,+over+65+years+of+age.)
- [24] R. Firoozabadi, R. E. Gregg, and S. Babaeizadeh, “P-wave Analysis in Atrial Fibrillation Detection Using a Neural Network Clustering Algorithm,” in *Computing in Cardiology*, 2018, vol. 2018-Septe.
- [25] F. Censi *et al.*, “P-wave Variability and Atrial Fibrillation,” *Sci. Rep.*, vol. 6, pp. 1–7, 2016.
- [26] I. Iguyon and A. Elisseff, “An introduction to variable and feature selection,” *J. Mach. Learn. Res.*, vol. 3, no. April, pp. 1157–1182, 2003.
- [27] M. Radovic, M. Ghalwash, N. Filipovic, and Z. Obradovic, “Minimum redundancy maximum relevance feature selection approach for temporal gene expression data,” *BMC Bioinformatics*, vol. 18, no. 1, pp. 1–14, 2017.
- [28] C. Ding and H. Peng, “Minimum redundancy feature selection from microarray gene expression data,” *Proc. 2003 IEEE Bioinforma. Conf. CSB 2003*, pp. 523–528, 2003.
- [29] M. L. Mchugh, “The Chi-square test of independence Lessons in biostatistics,” *Biochem. Medica*, vol. 23, no. 2, pp. 143–9, 2013.
- [30] M. ROBNIK SIKONJA MarkoRobnik and friuni-ljsi IGOR KONONENKO IgorKononenko, “Theoretical and Empirical Analysis of ReliefF and RReliefF,” *Mach. Learn.*, vol. 53, pp. 23–69, 2003.
- [31] F. Musumeci *et al.*, “An Overview on Application of Machine Learning Techniques in Optical Networks,” *IEEE Commun. Surv. Tutorials*, vol. 21, no. 2, pp. 1383–1408,



2019.

- [32] Y. Shen, “Loss Functions for Binary Classification and Class,” 2005.
- [33] M. F. Møller, “PREPRINT A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning Supervised Learning,” 1990.
- [34] S. Sharma and S. Sharma, “Understanding Activation Functions in Neural Networks,” *Int. J. Eng. Appl. Sci. Technol.*, vol. 4, no. 12, pp. 310–316, 2017.
- [35] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” *arXiv*, pp. 1–20, 2018.
- [36] P. Tutorials, “Understand tanh ( x ) Activation Function : Why You Use it in Neural Networks,” 2021. [Online]. Available: <https://www.tutorialexample.com/understand-tanhx-activation-function-why-you-use-it-in-neural-networks/>.
- [37] R. L. Unit, “Machine Learning FAQ Why is the ReLU function not differentiable at  $x = 0$  ?,” 2021. [Online]. Available: <https://sebastianraschka.com/faq/docs/relu-derivative.html>.
- [38] G. D. Clifford *et al.*, “AF classification from a short single lead ECG recording: The PhysioNet/computing in cardiology challenge 2017,” *Computing in Cardiology*, 2017. [Online]. Available: <https://physionet.org/content/challenge-2017/1.0.0/>.
- [39] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009.
- [40] D. Stathakis, “How many hidden layers and nodes?,” *Int. J. Remote Sens.*, vol. 30, no. 8, pp. 2133–2147, 2009.

## Appendix A: MATLAB Program

### training\_net.m

```
close all
clear all

%% Prepare input/output data

% load('3Features-v3.mat')
load('3Features_balanced-v3.mat')

inR = FeatR';
inM = FeatM';
inC = FeatC';
Y = reference_tab;

labels = {'A' 'N' 'O' '~'};
Out = Y;
Outbi = cell2mat(cellfun(@(x)
strcmp(x,labels),Out,'UniformOutput',0));
Outde = bi2de(Outbi);
Outde(Outde == 4) = 3;
Outde(Outde == 8) = 4;

t=Outbi';

%% Neural Network

accuracy=[];
recall=[];
precision=[];
f1_measure=[];
f1_max=[];
elapsed_time=[];

X=inR;
tic
k=4; %number of layers

for i=1:30
    i
    %% Prepare neural network
    hiddenLayerSize = [80 80 80 80];
```

```

net = patternnet(hiddenLayerSize);

net.performFcn = 'crossentropy';
net.performParam.regularization = 0;;
net.performParam.normalization = 'standard';

net.layers{1}.transferFcn='tansig';%tansig
net.layers{2}.transferFcn='poslin';
%     net.layers{3}.transferFcn='poslin';
%     net.layers{4}.transferFcn='poslin';
%     net.layers{5}.transferFcn='poslin';

net.input.processFcns =
{'removeconstantrows','mapminmax'};
net.output.processFcns =
{'removeconstantrows','mapminmax'};

net.divideFcn = 'divideblock'; % Divide data by blocks
net.divideMode = 'sample'; % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

net.trainFcn = 'trainscg'; % Resilient backpropagation
net.plotFcns =
{'plotperform','plottrainstate','ploterrhist','plotregressio
n','plotfit'};

net.trainParam.epochs =500; % Maximum number of epochs
to train. The default value is 1000.
net.trainParam.goal =0;%1e-16; %Performance goal. The
default value is 0.
net.trainParam.time =300; %Maximum time to train in
seconds. The default value is inf.
net.trainParam.min_grad =0;%1e-30; %Minimum performance
gradient. The default value is 1e-6.
net.trainParam.max_fail =10; %Maximum validation
failures. The default value is 6.

net.trainParam.mu =1e-3; %Marquardt adjustment
parameter. The default value is 0.005.

```

```

    net.trainParam.sigma =1e-5; %Determine change in weight
for second derivative approximation. The default value is
5.0e-5.
    net.trainParam.lambda =1e-5; %Parameter for regulating
the indefiniteness of the Hessian. The default value is
5.0e-7.

    %    net.trainParam.lr=0.05;    %0.01 Learning rate
    %    net.trainParam.delt_inc=1.5;    %1.2    Increment to
weight change
    %    net.trainParam.delt_dec=0.5;    %0.5    Decrement
to weight change
    %    net.trainParam.delta0=0.05;    %0.07    Initial
weight change
    %    net.trainParam.deltamax=100;    %50.0    Maximum
weight change

    %% Train neural network
    %
ew=(t==[1;0;0;0])*1+(t==[0;1;0;0])*0.1+(t==[0;0;1;0])*0.3+(t
==[0;0;0;1])*0; %setting weight
    [net,tr] = train(net,X,t);
    %    [net,tr] = train(net,X,t,[],[],ew); %training with
weight

    %% Test the Network
y = net(X);
e = gsubtract(t,y);
performance(i) = perform(net,t,y);
    %    performance(i) = perform(net,t,y,ew); %performance
with weight
    tind = vec2ind(t);
    yind = vec2ind(y);
    percentErrors(i) = sum(tind ~= yind)/numel(tind);

    %% Recalculate Training, Validation and Test Performance

    trainPerformance =
perform(net,t(:,tr.trainInd),y(:,tr.trainInd));
    valPerformance =
perform(net,t(:,tr.valInd),y(:,tr.valInd));
    testPerformance =
perform(net,t(:,tr.testInd),y(:,tr.testInd));

```

```

%% Performance metrics
o = double(y>=max(y));

tt=t(:,tr.testInd);      % test target
ot=o(:,tr.testInd);     % test output

for j=1:4
    tp=length(find(tt(j,:)==1 & ot(j,:)==1)); %true
positive
    fn=length(find(tt(j,:)==1 & ot(j,:)==0)); %false
negative
    tn=length(find(tt(j,:)==0 & ot(j,:)==0)); %true
negative
    fp=length(find(tt(j,:)==0 & ot(j,:)==1)); %false
positive

    acc(i,j)=(tp+tn)/(tp+tn+fp+fn); % accuracy
    rec(i,j)=tp/(tp+fn); % recall
    prc(i,j)=tp/(tp+fp); % precision
    f1(i,j)=(2*rec(i,j)*prc(i,j))/(rec(i,j)+prc(i,j)); %
F measure

    Ftrain(i,j)=(2*sum(t(j,tr.trainInd)==1 &
o(j,tr.trainInd)==1))/(sum(t(j,tr.trainInd)==1)+
sum(o(j,tr.trainInd)==1)); % F measure of training
    Ftest(i,j)=(2*sum(t(j,tr.testInd)==1 &
o(j,tr.testInd)==1))/(sum(t(j,tr.testInd)==1)+
sum(o(j,tr.testInd)==1)); % F measure of testing
end
clear net
end
prc=fillmissing(prc,'constant',0);
f1=fillmissing(f1,'constant',0);

et=toc
elapsed_time=[elapsed_time;et];

accuracy=[accuracy;k mean(acc);k std(acc)];
recall=[recall;k mean(rec);k std(rec)];
precision=[precision;k mean(prc);k std(prc)];
f1_measure=[f1_measure;k mean(f1);k std(f1)];
f1_max=[f1_max;k max(f1)];

```

```

perf=[k mean(performance),k mean(percentErrors);k
std(performance),k std(percentErrors)];
Ftot=[k mean(Ftrain),k mean(Ftest); k std(Ftrain),k
std(Ftest)];

%% Save results to xl file
filename='step2.xlsx';
range='B33'; %3,15,27,39 %3,5,7,9
writematrix(accuracy,filename,'Sheet','Accuracy','Range',range)
writematrix(recall,filename,'Sheet','Recall','Range',range)
writematrix(precision,filename,'Sheet','Precision','Range',range)
writematrix(f1_measure,filename,'Sheet','F1-
Measure','Range',range)
writematrix(perf,filename,'Sheet','Performance','Range',range)
writematrix(Ftot,filename,'Sheet','F1 total','Range',range)
writematrix(elapsed_time,filename,'Sheet','Elapsed
Time','Range',range)

```

## rearrange\_dataset.m

```
clear all

%% load data
load('myFeats-v3.mat')

%% under-sample Normal class by 6

idxN = find(ismember(reference_tab, 'N'));
idxNN = [];
for i=1:6:size(idxN)
    idxNN = [idxNN; idxN(i)];
end
features(idxN(ismember(idxN, idxNN)==0), :) = [];
reference_tab(idxN(ismember(idxN, idxNN)==0)) = [];

%% under-sample Others class by 3

idxO = find(ismember(reference_tab, 'O'));
idxOO = [];
for i=1:3:size(idxO)
    idxOO = [idxOO; idxO(i)];
end
features(idxO(ismember(idxO, idxOO)==0), :) = [];
reference_tab(idxO(ismember(idxO, idxOO)==0)) = [];

%% Feature Selection

A = features;
features = features(:, :);

in = features;
Y = reference_tab;

%% Rank features by importance %ReliefF
[idxR, scoreR] = relieff(in, Y, 10);
figure(1); bar(scoreR(idxR))
diffR = -diff(scoreR(idxR));
new_idxR = idxR(1:min(find(diffR < mean(diffR)))-1);
FeatR = in(:, new_idxR);
namesR = A(1, new_idxR).Properties.VariableNames;
```

```

%% mRMR
[idxM,scoreM] = fscmrmr(in,Y);
figure(2);bar(scoreM(idxM))
diffM=-diff(scoreM(idxM));
new_idxM=idxM(1:min(find(diffM<=mean(diffM)))-1);
FeatM=in(:,new_idxM);
namesM=A(1,new_idxM).Properties.VariableNames;

%% Chi2
[idxC,scoreC] = fscchi2(in,Y);
idxInf = find(isinf(scoreC));
figure(3);bar(scoreC(idxC))
hold on
bar(scoreC(idxC(length(idxInf)+1))*ones(length(idxInf),1))
legend('Finite Scores','Infinite Scores')
hold off
new_idxC=idxInf;
FeatC=in(:,new_idxC);
namesC=A(1,new_idxC).Properties.VariableNames;

%% Save data to xl file
save('3Features_balanced1-
v3.mat','FeatR','FeatM','FeatC','reference_tab');

```



## features\_selection.m

```
close all
clear all

%% prepare data
load('myFeats-v3.mat')
A=features;
features=features{:, :};
in = features;
Y = reference_tab;

%% Rank features by importance %ReliefF
[idxR,scoreR] = relieff(in,Y,10);
figure(1);bar(scoreR(idxR))
diffR=-diff(scoreR(idxR));
new_idxR=idxR(1:min(find(diffR<mean(diffR)))-1);
FeatR=in(:,new_idxR);
namesR=A(1,new_idxR).Properties.VariableNames;

%% Rank features by importance %mRMR
[idxM,scoreM] = fscmrnr(in,Y);
figure(2);bar(scoreM(idxM))
diffM=-diff(scoreM(idxM));
new_idxM=idxM(1:min(find(diffM<=mean(diffM)))-1);
FeatM=in(:,new_idxM);
namesM=A(1,new_idxM).Properties.VariableNames;

%% Rank features by importance %Chi2
[idxC,scoreC] = fscchi2(in,Y);
idxInf = find(isinf(scoreC));
figure(3);bar(scoreC(idxC))
hold on
bar(scoreC(idxC(length(idxInf)+1))*ones(length(idxInf),1))
legend('Finite Scores','Infinite Scores')
hold off
new_idxC=idxInf;
FeatC=in(:,new_idxC);
namesC=A(1,new_idxC).Properties.VariableNames;

%% Save data to xl file
save('3Features-
v3.mat','FeatR','FeatM','FeatC','reference_tab');
```