UNIVERSITÉ LIBRE DE BRUXELLES

Master thesis

Identification of patients with atrial fibrillation during normal sinus rhythm using different methodologies and machine learning techniques

Alexandre Chapelle

Faculty of Sciences

Master thesis submitted under the supervision of
Bersini Hugues

the co-supervision of
Gilon Cédric

Academic year
2022 - 2023

Abstract

Atrial fibrillation, characterised by an irregular and often rapid heartbeat, is a common heart condition that, although generally benign, can lead to more serious complications. The insidious nature of this condition often leads to its onset in patients with normal sinus rhythm, making them ostensibly healthy. Early detection of atrial fibrillation during routine medical examinations is therefore essential, as early diagnosis and intervention can mitigate the progression of the disease. This thesis explores various artificial intelligence methodologies, drawing on research and techniques used on similar subjects to discern subtle distinctions between healthy individuals and those with atrial fibrillation. These micro-differences, often imperceptible even to cardiologists experienced in normal sinus rhythm, could be the key to early identification of affected patients. Using a combination of machine learning models and pre-processing methods, this thesis sought to verify the feasibility of detecting these nuanced variations. The results validate the potential of artificial intelligence in this situation by showing encouraging results in the identification of atrial fibrillation. The findings also show that accuracy needs to be improved further, paving the way for more dependable and durable diagnostic tools in the future.

Contents

Acronyms

1 Introduction **1**

2 Cardiac arrhythmia **2**
2.1 The human heart . . . 2
2.2 Heart diseases . . . 5
2.2.1 Tachycardia . . . 5
2.2.2 Bradycardia . . . 6
2.2.3 Atrial fibrillation . . . 6
2.3 Leads . . . 7
2.3.1 Usefulness of leads . . . 8

3 Artificial intelligence **9**
3.1 History . . . 9
3.2 Machine learning . . . 10
3.2.1 What is machine learning? . . . 10
3.2.2 Supervised, unsupervised and semi-supervised learning . . . 11
3.2.3 Reinforcement learning . . . 11
3.3 Neural networks . . . 12
3.3.1 Activation function . . . 14
3.4 Recurrent neural network . . . 16
3.4.1 Long Short-Term Memory . . . 17
3.5 Convolutional Neural Network . . . 18
3.5.1 Local receptive fields . . . 18
3.5.2 Shared weights and biases . . . 19
3.5.3 Activation function and pooling . . . 19
3.6 RandomForest . . . 20
3.7 Deep learning . . . 21
3.8 Residual Networks . . . 22
3.9 Statistics . . . 22
3.9.1 Confusion Matrix . . . 22
3.9.2 Accuracy . . . 23
3.9.3 Sensitivity or true positive rate . . . 23
3.9.4 False positive rate . . . 23
3.9.5 Specificity or false negative rate . . . 23
3.9.6 Precision . . . 23
3.9.7 F1 score . . . 24
3.9.8 ROC curve . . . 24

4 State of the art **26**
4.1 AI in cardiology . . . 26

4.1.1 AF detection . . . 27
4.1.2 AI usage for AF . . . 27
4.1.3 New techniques . . . 28
4.1.4 Summary Table . . . 29

5 Materials **30**
5.1 Datasets description . . . 30
5.1.1 MIT-BIH Normal Sinus Rhythm Database . . . 30
5.1.2 MIT-BIH Atrial Fibrillation Database . . . 31
5.1.3 MIT-BIH Arrhythmia Database . . . 31
5.2 Data Processing Tools . . . 33
5.2.1 Python . . . 33
5.2.2 Waveform Database Software Package . . . 34
5.2.3 NeuroKit2: A Comprehensive Tool for Neurophysiological Data Analysis . . . 35
5.2.4 PyWavelets . . . 35
5.2.5 SciPy . . . 36
5.2.6 Other tools . . . 37

6 Methods **39**
6.1 Objectives . . . 39
6.2 Preprocessing . . . 41
6.2.1 Databases . . . 41
6.2.2 Common base . . . 43
6.2.3 10-second methodology preprocessing . . . 43
6.2.4 Beats methodology preprocessing . . . 53
6.2.5 RR Intervals methodology preprocessing . . . 53
6.2.6 Peaks methodology preprocessing . . . 55
6.2.7 Overlapping strategy . . . 57
6.3 Models . . . 58
6.3.1 Convolutional Neural Network . . . 58
6.3.2 LSTM Model . . . 59
6.3.3 ResNet Model . . . 60
6.3.4 Random Forest . . . 62
6.4 Data preparation . . . 63
6.4.1 10-second methodology . . . 63
6.4.2 Beats methodology . . . 64
6.4.3 RR methodology . . . 65
6.4.4 Peaks methodology . . . 65
6.4.5 Random Under Sampler . . . 65
6.4.6 Train-Validation-Test ratio . . . 66
6.5 Summary . . . 66
6.6 Computer ressources . . . 67

7 Results **69**
7.1 Different combinations . . . 69
7.2 Table of results . . . 69
7.3 Analysis of results . . . 74
7.3.1 Bad results . . . 74
7.3.2 Other results . . . 74
7.3.3 10-second methodology . . . 75

7.3.4 Beats methodology 76
7.3.5 RR Intervals methodology 77
7.3.6 Peaks 78
7.3.7 Preprocessing comparaison 78
7.3.8 Models comparaison 80
7.3.9 Different sampling frequencies 80

8 Conclusion **82**

A Complete Model Performances Table **92**

List of Figures

2.1 Diagram of a human heart [1] . . . 2
2.2 Electrical conduction [2] . . . 3
2.3 ECG [3] . . . 4
2.4 Electrocardiograph [4] . . . 4
2.5 Different ECGs. Each square on an ECG reflects a duration of 10 millisecond, a big square is equal to 40ms (4x10ms). . . . 5
2.6 Axillary lines, the right side of the thorax with lines labeled [5] . . . 7
2.7 Einthoven triangle [6] . . . 8
2.8 12-lead placement [3] . . . 8

3.1 Learning process [7] . . . 10
3.2 Kind of learning [8] . . . 11
3.3 Reinforcement learning, agent and environment [9] . . . 12
3.4 A simple neural network [10] . . . 12
3.5 Structure of an artificial neuron [11] . . . 13
3.6 Neural network classification [12] . . . 14
3.7 Linear function [13] . . . 14
3.8 Sigmoid function [14] . . . 15
3.9 Tanh function [14] . . . 15
3.10 ReLU function [14] . . . 16
3.11 Unfolded RNN [15] . . . 16
3.12 RNN applications [16] . . . 17
3.13 Long short-term memory [15] . . . 17
3.14 CNN architecture [17] . . . 18
3.15 Random Forest [18] . . . 20
3.16 Deep Learning representation [19] . . . 21
3.17 Residual Block [20] . . . 22
3.18 ROC curve [21] . . . 24

4.1 Health App View of Irregular Rhythm Measurements [22] . . . 28

5.1 Comparison of ECG data from different databases . . . 32
5.2 Initial visualisation using the WFDB package . . . 34
5.3 PyWavelets signal denoising . . . 36
5.4 SciPy signal cleaning . . . 37

6.1 Third methodology: representation of a R-R Interval . . . 40
6.2 P-Wave, QRS Complex and other measurements [23] . . . 41
6.3 Raw signals of the first three beats, with varying X-axis scaling due to individual cardiac rhythm differences. . . . 42
6.4 Example of an annotation file for record 100 in the MIT-BIH database. . . . 44

6.5 Normal 10-second segment . . . 45
6.6 Inverted ECG signal . . . 47
6.7 Resampled ECG signal, from 128 Hz to 360 Hz . . . 48
6.8 Butterworth filter . . . 49
6.9 Wavelet filter . . . 50
6.10 Complete manual ECG cleaning . . . 51
6.11 Manual vs Neurokit2 cleaning comparaison . . . 52
6.12 Rescaled ECG . . . 53
6.13 First 10 beats of NSRDB ECG Signal . . . 54
6.14 10 first RR Intervals . . . 55
6.15 Metrics Extraction . . . 56
6.16 Simple overlapping . . . 58
6.17 CNN Model . . . 59
6.18 LSTM Model . . . 60
6.19 ResNet Model . . . 61
6.20 ResNet Block . . . 62
6.21 RandomForest model . . . 63
6.22 Padding percentages per database . . . 64
6.23 Random undersampler . . . 66
6.24 Pipeline for preprocessing, data preparation and model testing . . . 68

7.1 Metrics by methodology . . . 75
7.2 Comparative analysis of the methodologies. Error bars represent a 95% confidence interval. . . . 75
7.3 ROC and AUC of 10-second, Neurokit2, 300 minutes, ResNet combination . 76
7.4 ROC and AUC of Beats, Neurokit2, 60 minutes, CNN combination . . . 77
7.5 ROC and AUC of RRI, Manual, 600 minutes, RandomForest combination . 78
7.6 ROC and AUC of Peaks, Manual, 600 minutes, LSTM combination . . . 79
7.7 Metrics by preprocessing method . . . 79
7.8 Comparison of the preprocessing methods . . . 79
7.9 Metrics by model . . . 80
7.10 Comparison of the models . . . 80

Acronyms

1D 1 Dimensional

2D 2 Dimensional

AF Atrial Fibrillation

AFDB Atrial Fibrillation Database

ANN Artificial Neural Network

API Application Programming Interface

AUC Area Under the Curve

aVL Augmented Vector Left

BIH Boston's Beth Israel Hospital

bpm Beats per minute

CNN Convolutional neural network

CT Computed Tomography

ECG Electrocardiograms

FN False Negative

FP False Positive

GRU Gated Recurrent Unit

Hz Hertz

LSTM Long-Short-Term-Memory

MAD Median Absolute Deviation

MIT Massachusetts Institute of Technology

MRI Magnetic Resonance Imaging

ms Millisecond

mV Millivolts

NSRDB Normal Sinus Rhythm Database

PSF Python Software Foundation

ReLU Rectified Linear Unit

ResNet Residual Network

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

RRI RR Interval

SMOTE Synthetic Minority Oversampling TEchnique

SVM Support Vector Machine

TN True Negative

TP True Positive

WFDB Waveform Database

Chapter 1

Introduction

Atrial Fibrillation (AF) affects around 12.6 million women and 20.9 million men worldwide. The prevalence in persons over the age of 20 is believed to be 3%, with larger percentages in the elderly population and patients with hypertension, diabetes, obesity. AF increases mortality, by a factor of two in women and a factor of 1.5 in males, and morbidity, such as heart failure and stroke [24]. The typical yearly incidence of stroke in patients with AF is 1%, and the annual mortality rate is 3%, with stroke accounting for just a minority of deaths, whereas sudden death and heart failure are more frequently implicated [25].

The promptness of treating atrial fibrillation is vital, and the process starts with the identification of potentially afflicted patients during a routine check-up. Even though these patients may exhibit normal sinus rhythm and are considered healthy, they could be harbouring this latent disease. The objective of this research is to leverage emerging artificial intelligence techniques specifically machine learning and to discern between healthy individuals and those who may be ill, even when their sinus rhythm appears normal. This distinction is crucial, as it enables early detection and intervention, potentially averting serious heart complications.

This study holds significant importance as it could lay the groundwork for enhanced patient management at the beginning of the disease. By utilising public data from various sources (as most patient data is sensitive and private), this research navigates the challenges posed by potential variations in data quality, recognising that these differences can directly impact the final results. The insights gained from this study could pave the way for a more proactive and personalised approach to cardiac care, exploiting the power of artificial intelligence to augment traditional medical practises.

The thesis begins with an overview of the heart, its essential functions, and various types of electrocardiograms and basic arrhythmias. Following this theoretical foundation, the focus shifts to a detailed presentation of the machine learning models employed, as well as the statistical methods used to interpret the results. A current state of the art review offers an overview of current practises and advancements in the field. The thesis then advances to its core, beginning with a detailed examination of the materials and methods used, providing a thorough understanding of the research process. It is completed with the presentation of the findings, which leads to a concise conclusion that encapsulates the results and implications of the research.

Chapter 2

Cardiac arrhythmia

To comprehend what a cardiac arrhythmia is and how it develops, it is necessary to first understand how the human heart operates and how it serves the entire body.

2.1 The human heart

The heart is, along with the brain, probably one of the most important organs in our body. It is a hollow muscular organ about the size of a closed fist located in the thorax between the two lungs (figure 2.1). Its role is to ensure blood circulation through various arteries and blood vessels by means of rhythmic contractions, thus allowing the blood pumped to carry oxygen and nutrients to the rest of the body [26].

The heart is divided into 4 parts separated by two vertical septa: a thick muscular septum called the ventricular septum situated between the ventricles and a thin septum called the atrial septum located between the atria. Adjacent to each septum are an atrium and a ventricle. The right atrium is connected to the right ventricle by the tricuspid valve, while the left atrium is linked to the left ventricle through the mitral valve. In total, there are 4 valves that prevent blood from flowing in the wrong direction.

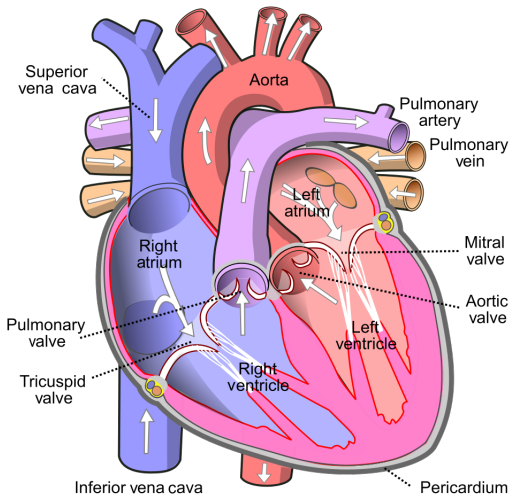


Figure 2.1: Diagram of a human heart [1]

The circulation of the blood is done in the following way:

1. The deoxygenated blood enters the right atrium which contracts, the tricuspid valve opens and the blood enters the right ventricle. The tricuspid valve closes while the pulmonary valve opens, and the ventricle then contracts in turn and evacuates the blood into the pulmonary artery.

2. The blood reaches the lungs where it can be charged with oxygen and release its carbon dioxide.

3. The oxygenated blood arrives via the pulmonary veins in the left atrium which contracts, the mitral valve opens and the atrium expels the blood into the left ventricle. Following the same principle as the right side, the mitral valve closes while the aortic valve opens, the ventricle contracts and expels the blood into the aorta from where it will join the other parts of the body.

The contractions occur in parallel in both parts of the heart, and the opening and closing of the valves are the source of the characteristic sound of a beating heart. These contractions, also called heartbeats, are caused by an electrical signal passing through the heart from top to bottom as we can see in figure 2.2.

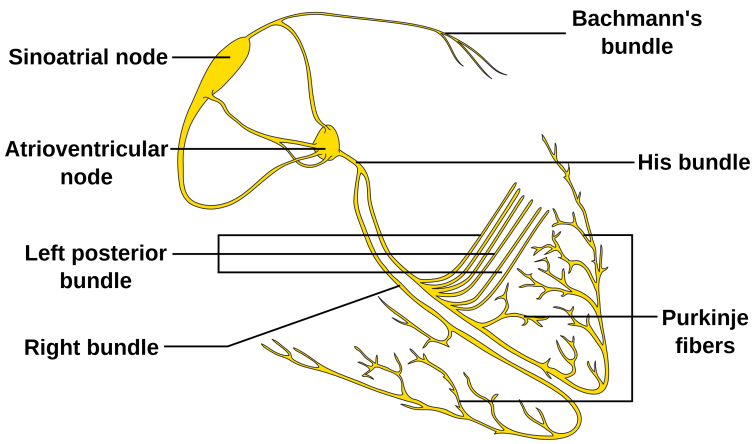


Figure 2.2: Electrical conduction [2]

The electric current crossing the heart from one side to the other is generated by the sinoartial node (located at the upper tip of the right atrium). The flow produced will then cross the two atria which will cause a first contraction. Between the atria and the ventricles there is an electrically hermetic partition, the only access point of the current to the ventricles is via the atrioventricular node located in the middle of the heart. From this node, the flow will travel through the Purkinje fibers to the inferior tip of the heart and cause a second contraction, this time of the ventricles.

For an adult human, the average is between 60 and 70 beats per minute (bpm) at rest and between 40 and 50 when sleeping [1].

Electrocardiograms (ECG), referenced in figure 2.3, are used by cardiologists to analyze this electrical current. ECGs are recorded via electrocardiographs (figure 2.4). This is a device connected to probes placed on specifics parts of the chest and the members to collect the various electrical variations produced by the heart during each contraction. We can see that we have several waves and intervals on this diagram. In order to understand the data that will be analyzed in this thesis, it is important to define the role of each.

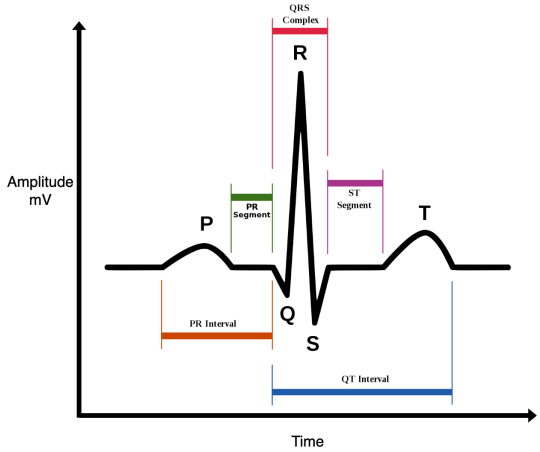


Figure 2.3: ECG [3]

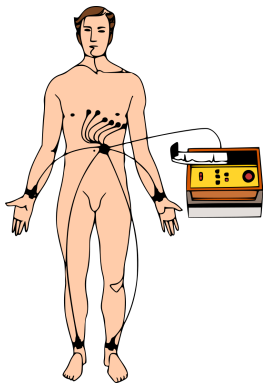


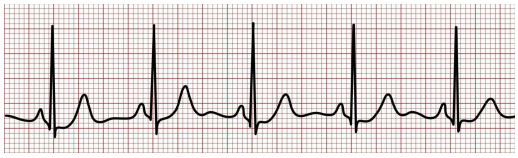
Figure 2.4: Electrocardiograph [4]

- The P-wave: this wave corresponds to the activation of the atria, so the contraction of both left and right atrium.
- The QRS complex: as for the artia, here it is the ventricles that contract.
- The T-wave: electric recovery phase of the ventricles.

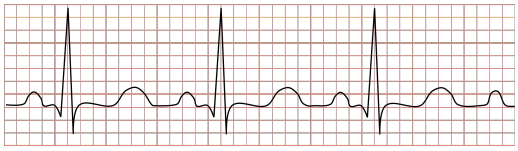
We also have several segments and intervals:

- PR interval: this interval corresponds to the time taken by the electrical impulse to travel from the sinoatrial node to the atrioventricular node.
- QT Interval: this one corresponds to the ventricles depolarisation and repolarization. The lengthening of the QT interval causes rhythm disorders of the ventricles. This interval can be extended through the use of medications [27].
- ST Segment: period during which the ventricles are depolarised. If there is still electric activity, this means that the electrical system of the heart is problematic, like expressing ischaemic diseases (these diseases will be explained in detail later).

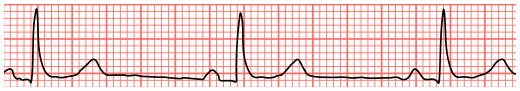
Two separate heartbeats always produce the same results. The interval between the two heartbeats is called the RR interval. The electrical system of the heart is functioning properly if the size of this interval remains almost constant. On the contrary, irregularities



(a) Normal ECG [28]



(b) Tachycardia [29]



(c) Bradycardia [30]

(d) Artial fibrillation [31]

Figure 2.5: Different ECGs. Each square on an ECG reflects a duration of 10 millisecond, a big square is equal to 40ms (4x10ms).

are the consequences of a malfunction. A short QRS interval, for example, may indicate that the patient has supraventricular tachycardia. Ventricular dysfunction is commonly the cause of a large QRS and in some cases, the P-wave may appear during or even after the QRS.

2.2 Heart diseases

Now that we have a good idea of how the heart works and what it does in the body, let us take a closer look at the diseases that affect it. An abnormal heart rhythm, also known as arrhythmia, is most commonly caused by an irregularity in the electrical flow in the heart. We can observe 3 types of arrhythmia: when the heart beats too fast, too slow or in an irregular way.

2.2.1 Tachycardia

As said before, the human heart beats at rest between 60 and 70 bpm for an adult (figure 2.5a). Each square on an ECG reflects a duration of 10 millisecond (ms), a big square is equal to 40ms (4x10ms), therefore the time between the two intervals on this figure is 160ms. When the body produces an important physical effort (while doing sports for example), this rhythm must be lower to

$$\text{Maximum heart rate} = 220\text{bpm} - \text{age of the person.} \tag{2.1}$$

So for a 40 year old human, his maximum non-hazardous rate can reach 180 bpm.

At rest, tachycardia is when the rhythm is greater than 100 bpm. We can easily observe that phenomenon through closer RR intervals with no other changes on the ECG, as we can observe in figure 2.5b. The time delay between two QRS periods is about 120ms in this case, which is clearly too short for normal heart electrical activity.

There exists several types of tachycardia [32].

1. Sinusal tachycardia: this is the most common tachycardia. It can be encountered physiologically during effort, fever or during a strong emotion.

2. Atrial tachycardia: due to ectopic node causing abnormal flow in the atria. This will cause different P-waves than normal sinusal P-waves.

3. AV nodal reentrant tachycardia: to be simple, the AV node is longitudinally dissociated in 2 path: one faster than the other. This will cause retrograde P-waves (the current flow will go back into atrium).

And there are many more; the important thing to remember is that tachycardias can originate anywhere in the heart.

2.2.2 Bradycardia

As opposed to tachycardia, we have bradycardia. This happens when the bpm is lower than 60 at rest for an adult. As for the tachycardia, only the RR intervals are modified: they move away from each other with no other modifications (figure 2.5c).

2.2.3 Atrial fibrillation

The atrial fibrillation is the most common cause of arrhythmia and the incidence increases with age. The electrical activity is completely disorganized and leading to the loss of mechanical contraction in the atria. This causes a blood stasis and thus favour thrombus formation. This thrombus can detached and represents an important risk of stroke (cardiovascular accident). On the ECG, this will cause an oscillation of the baseline without P-waves as we can see in figure 2.5d. There are several forms of atrial fibrillation:

- Paroxysmal, or, more simply, on an irregular basis. It can linger from a few seconds to many days before stopping on its own.
- Persistent, which means that it longs more than 7 days and ceases if treatment is administered to assist in restoring the heart's regular rhythm.
- Permanent, thus a persistent atrial fibrillation in which normal cardiac rhythm cannot be restored.

AF often proceeds from scattered paroxysmal episodes to more regular episodes, then to persistent AF and finally to chronic AF. This is why, in order to heal the disease as effectively as possible, it is critical to recognise and prevent it in its early stages.

In the case of a patient suffering from fibrillation, the use of medication is often mandatory for his health [33]. The purpose of these drugs is to restore the normal rhythm of the heart, trying to minimise the side effects that can be fatigue, low blood pressure, insomnia, nausea and many others. The goal is to have a resting heart rate of fewer than 90 beats per minute. The use of these drugs is frequently accompanied by anticoagulants, which, as the name implies, prevent the blood from coagulating and thus forming blood clots, thereby greatly reducing the risk of stroke.

When medicine fails to function, i.e. when there is no reaction or, worse, a negative reaction, other solutions are considered. One of this solution is the electrical cardioversion. A high-energy shock is delivered to the heart during electrical cardioversion to restore normal cardiac rhythm. Chemical cardioversion, on the other hand, involves the use of medications to try to restore a normal heart rhythm. The main objective of this technique is to stop the electrical ectopy and to allow the sinoatrial node to restart, which is responsible for the electrical impulses of the heart.

When these two solutions are not successful, a medical operation becomes mandatory. This operation is known as catheter ablation. A clinician inserts a thin, flexible tube (catheter) into the heart through a blood artery to terminate (ablate) the abnormal electrical communication route in the heart tissue. Any medical operation is risky, which is why it is essential to avoid the condition as soon as possible so that it may be treated less harshly.

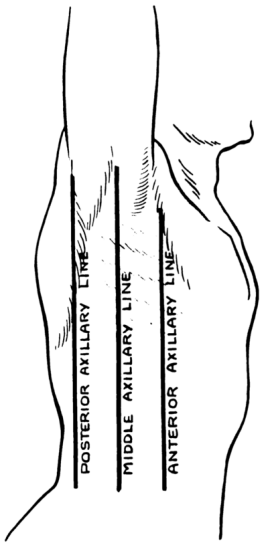


Figure 2.6: Axillary lines, the right side of the thorax with lines labeled [5]

2.3 Leads

As previously stated, probes, also known as pads, attached to the body's surface capture the current flowing through the heart, and measure the electrical potential difference between them. Two pads forms a lead, and there are twelve of them in total that can be captured by 10 pads. We then talk about 12-lead ECG [34]. Leads are divided into three categories: limb, augmented limb, and precordial (thoracic region). If we take the 12-lead ECG again, we will have 3 limb leads, 3 augmented leads, in a wheel-spoke pattern, and 6 precordial leads placed horizontally on the chest. The placement of these electrodes forms the Einthoven triangle (figure 2.7).
Following is the placement of the 10 electrodes that make up a 12-lead ECG:

1. RA: put on the right arm while avoiding a large muscle.
2. LA: put on the left arm, at the same place as RA.
3. RL: placed on the right leg, inner side of the calf.
4. LL: same as RL, but on the left leg.
5. V_1: on the right sternum between ribs 4 and 5.
6. V_2: same as V_1, but on the left.
7. V_3: placed between V_1 and V_2.
8. V_4: between ribs 5 and 6 in the middle axillary line (can be seen on figure 2.6).
9. V_5: horizontally with V_4 in the left anterior axillary line.
10. V_6: horizontally with V_4 and V_5 in the middle axillary line.

For example, lead 1 results from LA as the positive electrode and RA as the negative one (LA-RA). Lead 2 from LL as positive and RA as negative (LL-RA) and lead 3 from LL as positive and LA as negative (LL-LA) [35].
The combination of the different electrodes is then used to form the different leads that go up to 12. Lead 2 has been defined as the standard lead.

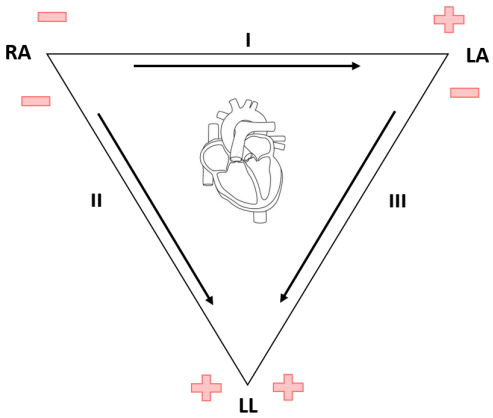


Figure 2.7: Einthoven triangle [6]

2.3.1 Usefulness of leads

Leads provide different "views" of the electrical activity of the heart, opening up a wider margin of analysis and therefore allowing for improved identification and prevention of specific diseases. They enable practitioners, and therefore cardiologists, to comprehensively analyse electrical activity from various viewpoints, detecting even the smallest aberration that may arise. Indeed, it is not because a lead does not present any anomaly that this same anomaly cannot be detected by another lead.

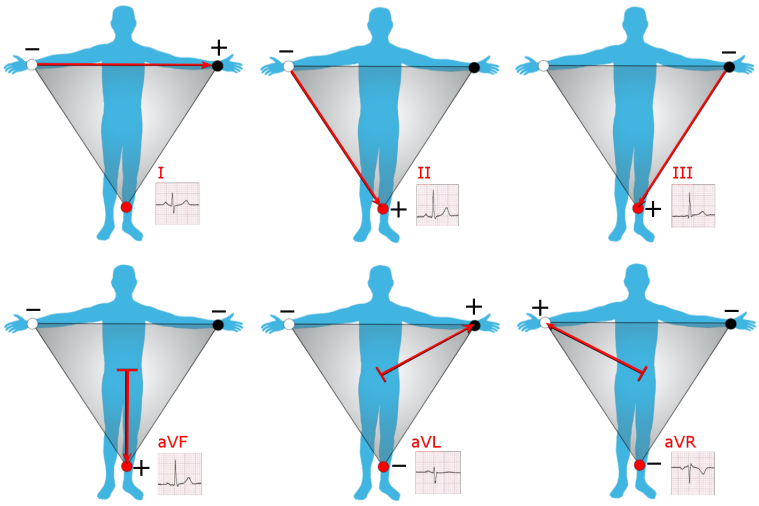


Figure 2.8: 12-lead placement [3]

Chapter 3

Artificial intelligence

Artificial intelligence (AI) can be described as a set of techniques and methods used by machines to reproduce natural intelligence, hence the one seen in humans or animals brains.

AI as an autonomous intelligence, capable of learning and solving problems as a human would and thus somehow copying cognitive functions. However, we must separate fiction from reality: for the moment, it is only algorithms and mathematical logic rather than real intelligence [36].

3.1 History

In 1949, Warren Weaver, an American scientist, published a memorandum suggesting that tasks requiring natural intelligence (and therefore a human) could be automated by a machine.

The first time the idea of artificial intelligence was mentioned was around the 1950's when Alan Turing, pioneer of the foundations of modern computing, approached the idea of a machine thinking for itself. In his article, Computing Machinery and Intelligence [37], he explores the notion and the concept of a thinking machine in order to determine at what point a machine can be said to be conscious, and thus become fully intelligent. Since then, the advancement of technology and the development of computer techniques has resulted in a phenomenal increase in the field of artificial intelligence. Here are some of the most important dates that led to the AI we know today:

- 1948: general computer architecture by Von Neumann [38],
- 1950: the Turing test, to determine if a machine is conscious,
- 1997: IMB's Deep Blue defeat the world chess champion Kasparov [39],
- 2009: first self-driving car by Google [40],
- 2016: the world champion of Go Lee Sedol is beaten by Google DeepMind's AlphaGo IA [41].

During the 1980s, a revolutionary concept was born: automatic learning, also known as machine learning. This principle is simple: on the basis of input data, rules and predefined operating modes, the machine must deduce the best output (or result). It should be able to solve problems that it has not been directly taught for, hence the name machine learning. Most of the algorithms developed during these years are still used today, such as reinforcement learning or neural networks.

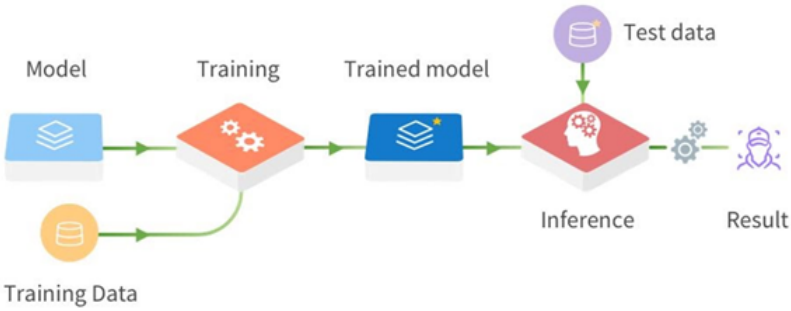


Figure 3.1: Learning process [7]

With the development of the internet and large computing infrastructures offering a volume of data never reached before, a new learning technique appeared: deep learning. This technique allows machines to surpass human skills in many fields such as translation, recognition (visual or auditory) or even document analysis. [42]

3.2 Machine learning

As previously stated, machine learning techniques developed in the 1980s are still in use today. Using old techniques may appear counterproductive in a fast-paced world where techniques and technologies are rapidly becoming obsolete. Let us take a look at how these algorithms function.

3.2.1 What is machine learning?

Machine learning is a branch of artificial intelligence that uses algorithms to make predictions or classifications and improve their performance over time, and therefore try to mimic cognitive functions.

As we can see in Figure 3.1, the process is as following:

1. the model will train itself using and training data. The model will be more accurate as we collect more data,
2. during training, the model will discover patterns and even begin to make predictions,
3. after the model has been trained, it will be tested in real-world scenarios using previously unseen data,
4. depending on the given output, modifications and/or improvements can be made to optimise the results.

This gives us a general idea of the concept of a machine that learns from the data it is given. This concept will then vary depending on the type of algorithm used [43]. We can classify machine learning algorithms into 4 types:

- Supervised learning.
- Unsupervised learning.
- Semi-supervised learning.
- Reinforcement learning.

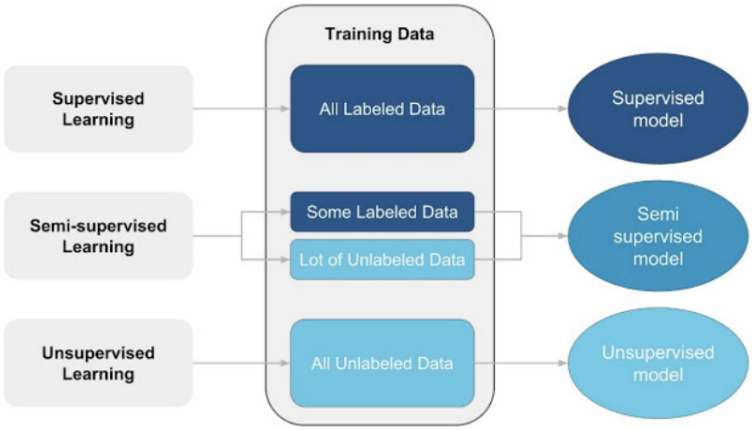


Figure 3.2: Kind of learning [8]

3.2.2 Supervised, unsupervised and semi-supervised learning

We have seen that the model is self-teaching on training data throughout the process. It exists 3 important types of learning, as shown in Figure 3.2 [44].

This data can be pre-processed for supervised learning, which means it has been sorted and labeled to allow the agent to give the correct output with precision. As a result, the input and output data are provided ahead of time, allowing the model to learn quickly. Image recognition, spam detection, and even predictive analysis are examples. The disadvantages of this learning are human errors in data labeling (resulting in incorrect outputs), or the fact that the model cannot cluster or classify data on its own, making it more likely to make mistakes with unseen data.

In the case of unsupervised learning, the situation is the polar opposite. The model will be trained on unlabeled data to see if it can find patterns to solve clustering or association problems; this means that the model will be given inputs but not the expected output. Image compression, targeted movie recommendations, and other applications benefit from this type of learning. The only real disadvantage of this learning process is the amount of time it takes during the learning phase; starting from scratch, analyzing and sorting data correctly takes a long time.

Semi-supervised learning is a hybrid of the two methods mentioned above, in which the model learns from a small amount of labeled data and a large amount of unlabeled data. This method of learning has been shown [45] to be the most effective of the three because it eliminates the disadvantages of the other two. Semi-supervised learning is commonly used in image classification and object detection.

3.2.3 Reinforcement learning

Whereas supervised learning is primarily concerned with regression and classification, unsupervised learning handles clustering and associations problems and semi-supervised deals with classification of unlabeled data based on labeled data, reinforcement learning is used to treat exploitation and exploration of an environment as shown in Figure 3.3 [46]. This autonomous system is called an agent [47]. It can be thought of as something that perceives an environment (receives data), analyses the best way to process those data, and then acts on that same environment.

Depending on its response, it might either be rewarded or punished so that it improves its response each time and thus learns continuously. Each learning agent's goal is to

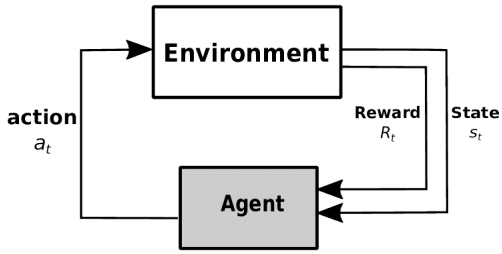


Figure 3.3: Reinforcement learning, agent and environment [9]

progress from a starting point to an end point. The path that must be taken is defined by default in supervised and unsupervised learning, but in reinforcement learning, the path must be optimised and chosen by the agent [48].

3.3 Neural networks

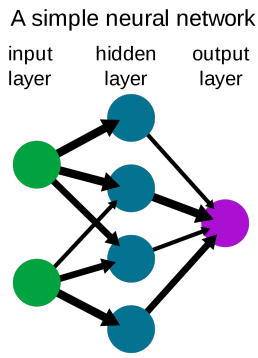


Figure 3.4: A simple neural network [10]

A neural network can be thought of as a network of artificial neurons (from which the term "Artificial Neural Network", simply called ANN, arose), also known as nodes or perceptron, that are linked together by edges as shown in the Figure 3.4. They were inspired by biological neural networks found in animal brains [49]. The network is decomposed into 3 parts:

- The input layer: this layer (or, in some cases, multiple input layers) is in charge of getting data into the network. They are referred to as "passive" because they receive no information from previous layers. The number of neurons in the input layer is equal to the number of input variables in the data being processed [50].
- The hidden layer: it is located between the input and output layers and uses an activation function to apply weights to the inputs and redirect them to one or more outputs (this point will be explained in detail later). A hidden layer is required if and only if the data must be separated non-linearly.
- The output layer: this is the final layer of neurons that generates and outputs information. The number of neurons in the output layer is the same as the number of outputs associated with each input.

The structure of a neural network will remain constant in most cases (there are a number of ways to deviate from the fundamental framework). Because of their classification

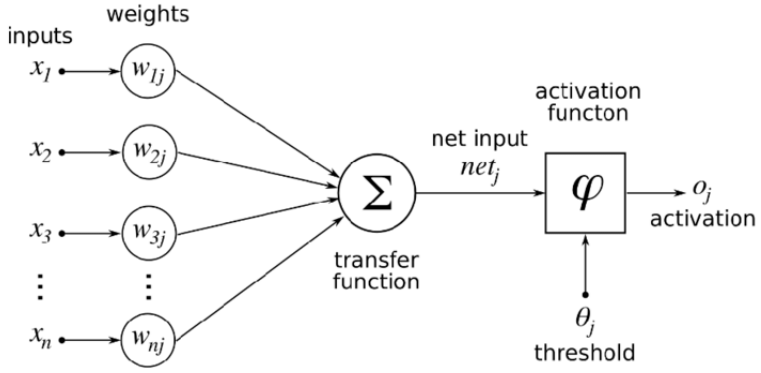


Figure 3.5: Structure of an artificial neuron [11]

and generalisation abilities, neural networks are frequently used in statistical problems [51].

A neuron can be thought of as an automaton that uses a transfer function to convert inputs into outputs. As shown in Figure 3.5, we can think of it as a neuron that adds up its entries, compares the result to a threshold, and responds by sending a signal to its outputs if the threshold is exceeded. This signal corresponds to the input value that the transfer function has changed, the formula can be written as:

$$output = \sum(weight * inputs) + bias \tag{3.1}$$

The parameters, so the weights and the bias, are numerical values which will be adjusted by training to achieve the best results.

Synaptic weight refers to the strength or amplitude of a connection between two nodes. This weight are modulated by learning rules in order for the network to converge to a value that ensures optimal classification. This value can be negative, we then speak of inhibition, or positive, called excitation [52]. The only requirement is that the output values be positive and between 0 and 1 (since we are seeking for a percentage, this makes sense).

To summarise, input nodes accept numericals that represent information (it could be an image, a vector of values, and so on). This value is regarded as an activation value received by each node; the greater the value, the greater the activation (and therefore the precision). Based on connection strengths (weights), inhibition or excitement, and transfer functions, the activation value is transported from node to node. Each node adds up the activation values it receives and adjusts the value according to its transfer function. The activation travels through the network, passing through the hidden layers and arriving at the output nodes. When the output layer is reached, the value of each node in this layer meaningfully mirrors the given input. The difference between the output value and the expected value, referred to as the error, is then propagated backward by assigning weights to each node based on how much of the error is attributable to that node. This method is known as gradient descent. This strategy can be used by the neural network to fine-tune and improve the accuracy of its future predictions.

A simple example is a dog and cat classification, shown in Figure 3.6. The input is a picture of a dog, which is subsequently altered such that the nodes can interpret it and the information flows in the network. At the end of the process, output nodes produce a result: 95% of chance that this is a dog and 5% a cat. The missed 5% will then be corrected via a backward propagation through the network and the weights will be adjusted.

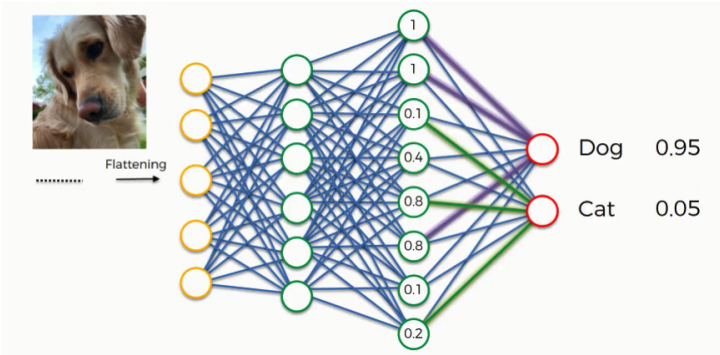


Figure 3.6: Neural network classification [12]

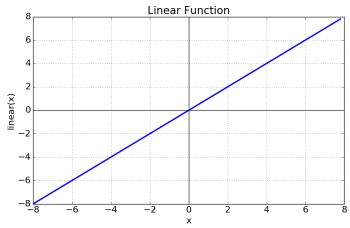


Figure 3.7: Linear function [13]

3.3.1 Activation function

As previously stated, information can be transformed in different ways [53]. Activation functions are classified into two types: linear and non-linear. Only the most well-known and widely used will be mentioned in this section.

With a linear function depicted in Figure 3.7 also known as identity function where

$$f(x) = x \tag{3.2}$$

as shown in the Equation 3.2, the activation is proportional to the input. Because each neuron uses the same formula, only the last layer (hence the output layer) can be defined, rendering the remaining layers worthless. It is also not possible to use the backward propagation, since the derivative of

$$f'(x) = 1 \tag{3.3}$$

as shown in Equation 3.3 making the optimisation process pointless.

Non-linear functions are far more interesting due to their properties: because the derivative of their equation may be related to the input, they allow backward propagation; it is thus feasible to go back in the network and determine which neuron weights in the input neurons can produce a better forecast. They also permit the usage of multiple layers since the output is now a non-linear combination of inputs passed through the network. To name a few, the sigmoid, tanh, and Rectified Linear Unit (ReLU) functions are among the most well-known.

Sigmoid function

This function accepts any real value as input and returns a value between 0 and 1 [54]. As shown in Figure 3.8, the larger the input (more positive), the closer the output value is to 1, whereas the smaller the input (more negative), the closer the output is to 0 [14].

This property is useful since we try to obtain probabilities as outputs. The mathematical expression is the following:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

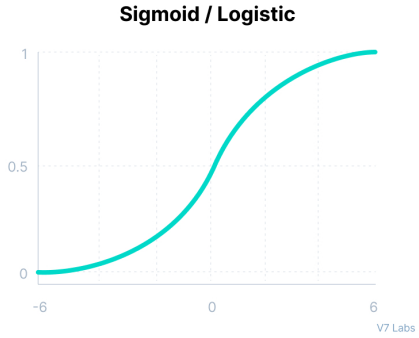


Figure 3.8: Sigmoid function [14]

Tanh function

Like the sigmoid function, the tanh function is S-shaped with the difference in output range of -1 to 1. The closer the input value is to ∞ the closer the output will be to 1, and vice versa. It is therefore easy to center the outputs around zero; as a consequence, the output values are readily characterised as highly negative, neutral, or positively strong. Its equation is:

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \tag{3.5}$$

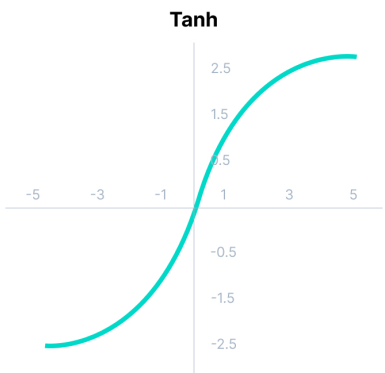


Figure 3.9: Tanh function [14]

ReLU function

ReLU is an abbreviation for Rectified Linear Unit. Despite its name (linear), the function is not linear and offers the same benefits as sigmoid (backpropagation) but with better performance. As we can see in Figure 3.10 and Equation 3.6, the formula is simply

$$f(x) = max(0, z) \tag{3.6}$$

where z is any real number strictly greater than 0. Since negative values are not taken into account, the ReLU function does not activate all the neurons at the same time which is an interesting property.

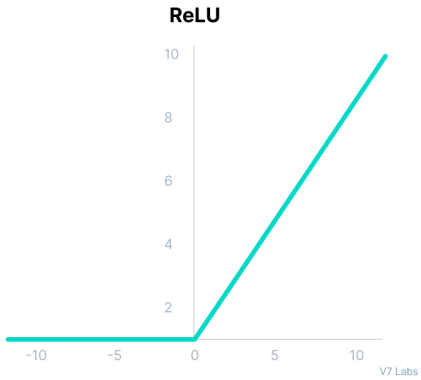


Figure 3.10: ReLU function [14]

3.4 Recurrent neural network

Recurrent neural networks (RNN) are a subset of neural networks in which neurons can have recurrent connections. This feature allows the network to behave in a temporal dynamic behavior and solves 2 problems encountered with ANNs; handling variable-length sequences of inputs and processing input data where temporal or sequential relationships between elements are crucial. It is now possible to make predictions based on past events, and then for an input x, an output y and a time t:

$$t : y_t = \sum_{t=1}^{t-1} x_t \tag{3.7}$$

which demonstrates the concept of recurrence connections (we can think of it as memory). Figure 3.11 shows this.

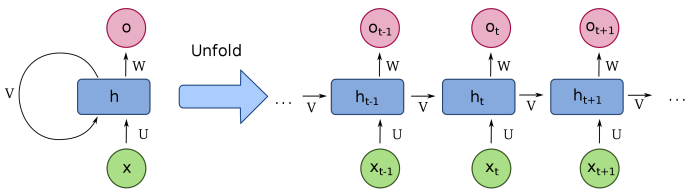


Figure 3.11: Unfolded RNN [15]

Used mainly for speech and handwriting predictions, they were designed to solve sequence prediction problems which is very useful when analysing an ECG sequence.

To have an idea of sequence predictions problems, let us take a look at Figure 3.12.

- Many to One: Find the correct output from a set of inputs. This could be the result of a group of words attempting to find the end of a sentence.

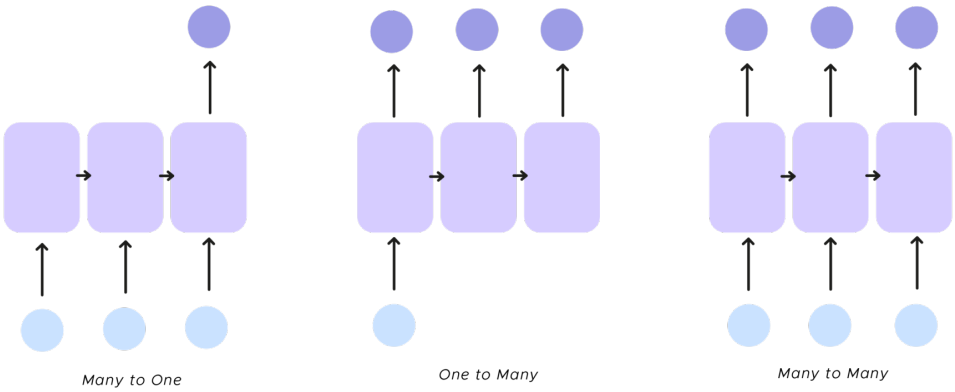


Figure 3.12: RNN applications [16]

- One to Many: From a single input, try to extrapolate the possibilities.
- Many to Many: From several inputs we produces outputs with specificity that the number of inputs and outputs can vary. It could be text translation for example.

However, RNNs face a major challenge: short-term memory. As sequences become longer, RNNs struggle to retain previous information, leading to inaccurate or incorrect results. This problem is amplified by gradient vanishing, where the gradients of the loss function become infinitesimally small, making it difficult for the network to learn and update its weights for previous layers of the sequence.

3.4.1 Long Short-Term Memory

Long Short-Term Memory, abbreviated as LSTM, was introduced as a solution to the short-term memory problem and the vanishing gradient issue faced by traditional RNNs [55]. The principle can be seen in Figure 3.13.

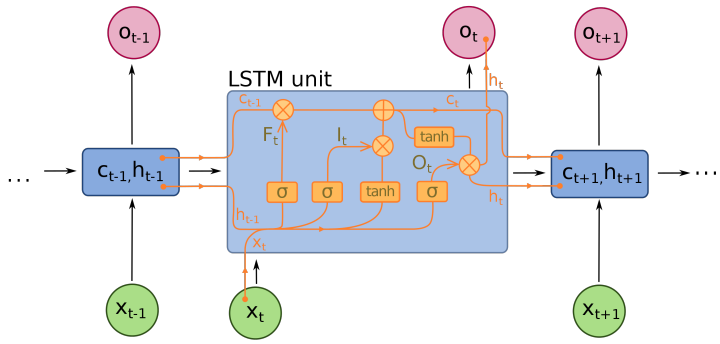


Figure 3.13: Long short-term memory [15]

The neuronal cell is more complex, allowing it to solve this problem. There are four key points to remember:

1. The first gate, known as the forget gate, is responsible for removing irrelevant information.

2. The store gate, on the other hand, saves new information.

3. Based on the forget and store gates, the third gate, known as the update gate, updates information to the RNN.

4. Finally, there is the output gate.

These 4 gates enable the RNN to control and store pertinent information during the whole process. Other than LSTM, there are versions in which just the content of the cell is altered while the concept stays same.

3.5 Convolutional Neural Network

Convolutional neural networks (CNN) are mostly used for image analysis, such as distinguishing between a car and a motorcycle. They may be thought of as a pre-processing variation of image classification methods [56].

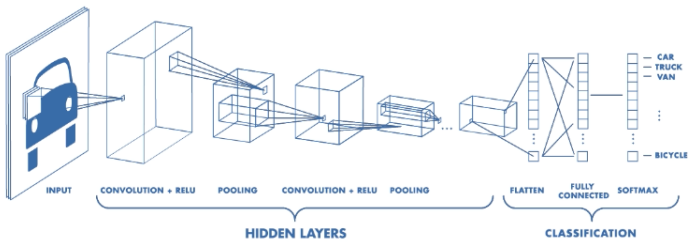


Figure 3.14: CNN architecture [17]

There are three fundamental principles to grasp in order to comprehend what a CNN is and what makes it unique [17]:

1. local receptive fields.

2. shared weights and biases.

3. activation function and pooling.

3.5.1 Local receptive fields

In a traditional neural network, every input-layer neuron is linked to a hidden-layer neuron. This means the input data's global patterns are discovered by the network. The local receptive field, also referred to as the kernel or each neuron in a Convolutional Neural Network, is connected to a relatively small area of the input layer. Since local spatial correlations are more significant for tasks like image recognition, this enables the CNN to focus on learning local patterns in the input data [57].

The local receptive field covers the whole input image, and it is then convolved or slid across to find the dot product between the weights of the kernel and the weights of the input neurons in the receptive field. A feature map is produced after this operation is performed across each and every possible receptive field position on the input. The feature map displays how the detected feature is distributed spatially throughout the input at various points.

The network can learn spatial feature hierarchies automatically and adaptively thanks to the convolutional process. In the network's early layers, the kernels may learn how to react to basic features like edges and textures. The kernels combine the more basic features to represent complex visual concepts in the deeper layers. The capability of CNNs to learn features in a hierarchical structure is one of their main advantages.

CNNs derive their name from the process termed "convolution," which involves the combination of two functions to produce a third. This convolution is achieved by applying kernel operations to the input data, enabling CNNs to generate feature maps. These kernels, often referred to as local receptive fields, allow CNNs to concentrate on local patterns within the input, distinguishing them from traditional neural networks. By applying these kernels across the entire input, CNNs exhibit a robust capability to learn hierarchical spatial features.

3.5.2 Shared weights and biases

The idea of shared weights and biases in a CNN represents an important difference from standard neural networks. In a CNN, not every neuron in the convolutional layer is linked to every neuron in the layer below it. Instead, the size of the kernel or filter determines how many neurons in the previous layer are connected to those in the convolutional layer.

The fundamental idea behind shared weights is that each neuron in the convolutional layer uses the same set of weights and biases (the values added to the results of the multiplications), which are typically represented by a matrix called a filter or a kernel. Each neuron in that layer is therefore detecting the same feature but in a different area of the input space.

CNNs can recognise a feature regardless of where it appears in the input space thanks to the shared weights' capacity to be translation invariant. If a CNN learns to recognise a cat's face, it can do so regardless of where the face appears in the image—top left corner, bottom right corner, or anywhere else. This is a significant distinction between CNNs and traditional neural networks because it allows the network to learn fewer parameters, lowering computational costs and the risk of overfitting while retaining the network's ability to recognise features anywhere in the input space [58].

In conclusion, shared weights and biases in CNNs allow the network to recognise particular features regardless of where they are in the input space, making the network more effective and robust in handling translational variations in images.

3.5.3 Activation function and pooling

Neurons in a neural network use an activation function to introduce non-linearity, which enables the network to learn and model complex patterns. Due to its effectiveness and simplicity, ReLU is a frequently used activation function. If the input is positive, it outputs the input directly; if not, it outputs zero. This function for the network reduces the vanishing gradient issue, which can impede learning in deep networks [59].

The pooling layer in a CNN significantly reduces the spatial size (width and height) of the input volume. By reducing the number of parameters and computations in the network, overfitting is prevented. Different types of pooling exist, including maximum Pooling, Average Pooling, and others. For instance, Max Pooling reduces the input size while preserving the most important information (the maximum value in a specific window).

The network learns more complex features as it advances through the layers. For instance, in the context of image recognition, the first hidden layer might be taught to

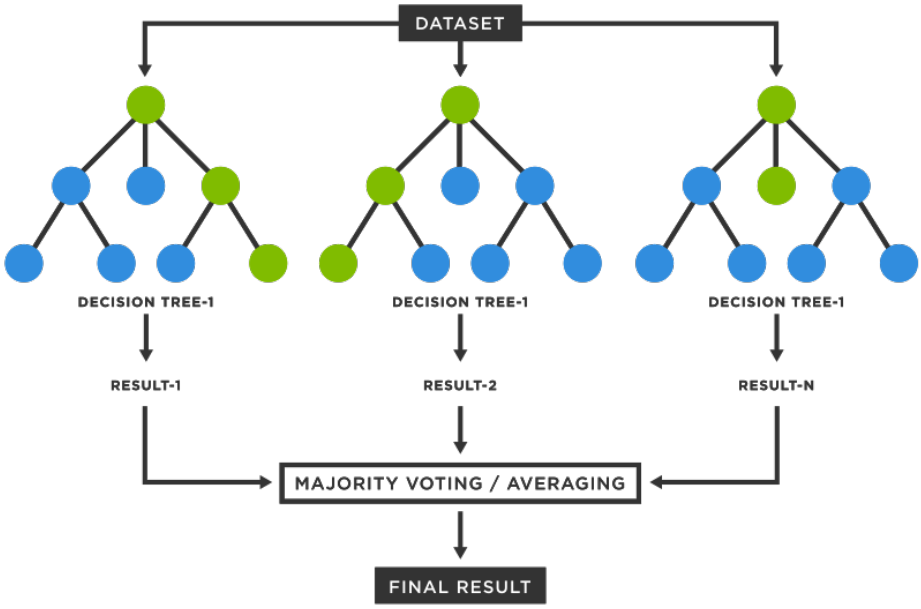


Figure 3.15: Random Forest [18]

recognise simple edges, while later layers might learn to recognise more intricate shapes or objects by building on the features discovered in the earlier layers.

Similar to traditional neural networks, the final layer in a CNN is typically a fully connected layer where each neuron receives input from every neuron in the preceding layer. In simple terms, this layer uses the high-level features that the convolutional layers have learned to categorise the input into different classes according to these features. The output of this layer is then given to a softmax activation function, which reduces the outputs to values between 0 and 1, which correspond to the probability distributions of potential output classes.

If we take our example from Figure 3.14, the first hidden layer learns to recognize edges, whereas the final learns to detect more complicated forms. The last layer, like in a traditional neural network, connects every neuron from the last hidden layer to the output neurons. This yields the final result.

3.6 RandomForest

RandomForest [18] is a machine learning algorithm that is common for many tasks because of its effectiveness, adaptability, and simplicity. It is a member of the group of algorithms known as ensemble methods, which combine the predictions of several base estimators to increase generalizability and robustness.

The RandomForest algorithm is a collection of Decision Trees, as suggested by its name. The structure of a decision tree is similar to a flowchart, with each internal node standing in for a feature (or attribute), each branch for a decision rule, and each leaf node for an outcome. Decision trees' main goal is to separate the data into subsets based on different criteria, making the data simpler to understand and predict.

A single decision tree, however, has a tendency to overfit the data it is trained on, which means it captures the data noise and performs poorly on unobserved data. The RandomForest algorithm comes into play here. From a set of randomly chosen subsets

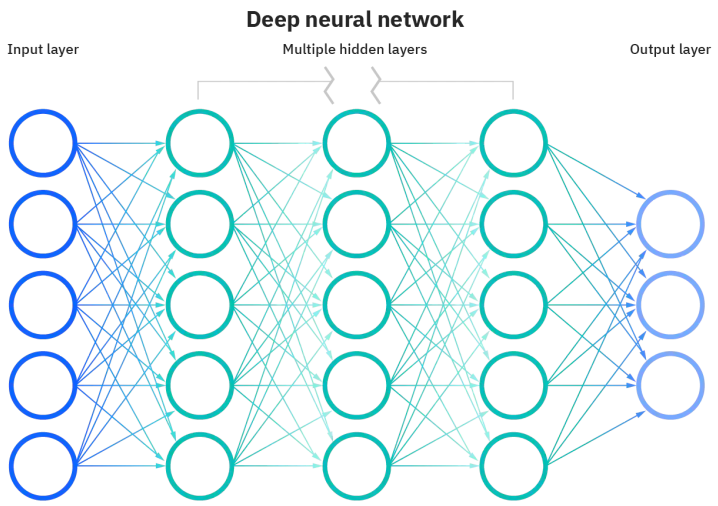


Figure 3.16: Deep Learning representation [19]

of the training set, it builds a number of Decision Trees, and then it adds up their votes to produce the final prediction. By averaging the outcomes of various Decision Trees, the overfitting issue is reduced and prediction accuracy is increased as shown in Figure 6.21.

Each Decision Tree in the RandomForest gives a vote in the case of a binary classification problem, predicting either the positive class or the negative class for the given input. The final prediction is made using the class that receives the most votes across all of the forest's trees.

The number of trees in the forest and the maximum depth of the trees are two hyperparameters that can be adjusted to improve the performance of the RandomForest algorithm. Additionally, it offers an accurate estimation of feature importance, which is very helpful in comprehending the underlying data.

The RandomForest algorithm is a strong and adaptable tool in the machine learning toolbox. It is compatible with a variety of data structures and types, and with little tuning, it frequently delivers excellent predictive performance.

3.7 Deep learning

Deep learning, a subset of artificial neural networks, involves computer programmes designed to mimic and learn like the human brain. The term "deep" stems from the multiple layers of neural networks in these models, as illustrated in Figure 3.16. By adding an input and output layer to an existing neural network, it becomes "deep." Each layer consists of several neurons, or nodes, that process the input uniquely, passing the results to the subsequent layer until reaching the output. While issues like overfitting, the disappearing or expanding gradient problem, and the challenge of training very deep networks may limit standard neural networks, deep learning offers a set of strategies and designs that help address these issues.

In conclusion, while neural networks serve as the foundation for deep learning, the latter entails a wider range of methods and architectures intended to go around the constraints imposed by conventional neural networks and allow the construction of more sophisticated and effective models.

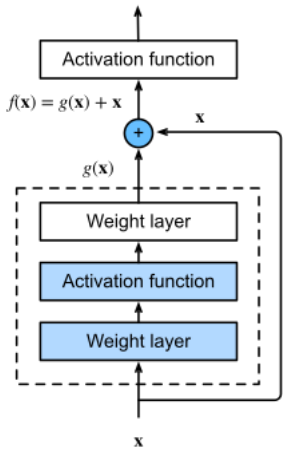


Figure 3.17: Residual Block [20]

3.8 Residual Networks

ResNet, or Residual Networks, stands as a traditional neural network foundational to many computer vision tasks. This was introduced by He et al. in their 2015 paper titled "Deep Residual Learning for Image Recognition" [60]. Their work presented ResNet as a solution to the challenges faced when training very deep neural networks.

One such challenge is the vanishing gradients issue. In deep neural networks, the gradient of the loss function can become infinitesimally small due to repeated multiplication as it's backpropagated to the earlier layers. Consequently, the weights of these early layers hardly change, preventing them from acquiring new knowledge. ResNet's innovative approach to this problem is the introduction of a shortcut connection or skip connection. This involves identity shortcut connections that bypass one or more layers, as illustrated in figure 3.17. The outputs from both the stacked layers and the identity shortcut connection are then combined. This combined output undergoes processing through a ReLU activation function. This unique structure empowers the model to learn the differences, or residuals, between the input and output of the stacked layers.

Thanks to these shortcut connections, ResNet can be trained with a multitude of layers without succumbing to the vanishing gradient problem. It's noteworthy that the original ResNet model was trained with an impressive 152 layers, showcasing a significant leap from earlier models.

3.9 Statistics

Some useful statistics allowing a better understanding of the results. These will be explained in a simple and condensed way.

3.9.1 Confusion Matrix

The confusion matrix serves as an initial insight into the performance of a predictive model, offering a comprehensive analysis of its accuracy and quality. Comprising four essential components, the confusion matrix elucidates the model's predictive capabilities in the following manner:

- True Positives (TP): Instances where the model correctly identified positive elements.

- True Negatives (TN): Instances where the model accurately predicted negative elements.
- False Positives (FP): Instances where the model erroneously classified negative elements as positive.
- False Negatives (FN): Instances where the model incorrectly predicted positive elements as negative.

Table 3.1 gives a clearer idea of its structure.

	Disease present	Disease absent
Positive test	True positive (TP)	False positive (FP)
Negative test	False negative (FN)	True negative (TN)

Table 3.1: Confusion Matrix

3.9.2 Accuracy

The accuracy represents the percentage of correctly classified data in the whole test data. The higher the accuracy, the better the overall performance of the classifier.

$$Accuracy = \frac{TP + TN}{TN + TP + FN + FP} \tag{3.8}$$

3.9.3 Sensitivity or true positive rate

The sensitivity represents the proportion of data classified as positive to actual positive data, so the proportion of true positives.

$$Sensitivity = \frac{TP}{FN + TP} \tag{3.9}$$

3.9.4 False positive rate

The false positive rate is the proportion of tests that were incorrectly labeled as positive to the total number of actual absent, so negative, disease.

$$False\ positive\ rate = \frac{FP}{FP + TN} \tag{3.10}$$

3.9.5 Specificity or false negative rate

The specificity represents the proportion of data classified as negative to actual negative data, so the proportion of true negatives.

$$Specificity = \frac{TN}{FP + TN} \tag{3.11}$$

3.9.6 Precision

The precision is the proportion of positive results that are true positive, therefore the ratio of the number of correctly identified positive samples to the total number of positive samples.

$$Precision = \frac{TP}{TP + FP} \tag{3.12}$$

3.9.7 F1 score

The F1 score is the harmonic mean of precision and sensitivity.

$$F1\ score = 2 * \frac{precision * sensitivity}{precision + sensitivity} = \frac{2TP}{2TP + FP + FN} \tag{3.13}$$

3.9.8 ROC curve

The ROC (Receiver operating characteristic) curve represents the difference between the true positive rate against the false positive rate based on different threshold settings [61]. More items can be categorised as positive when the classification threshold is lowered, which raises the proportion of both true and false positives. The ROC curve is represented in Figure 3.18.

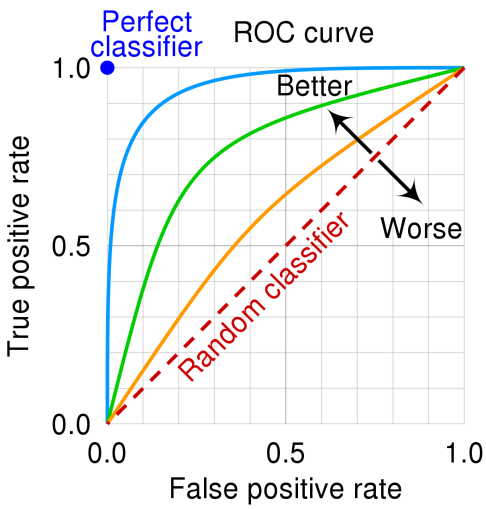


Figure 3.18: ROC curve [21]

This curve gives us several information:

- In the case of (0, 0), the classifier identifies every negative; there are no true positives or false positives.
- There are no true negatives or false negatives in (1, 1); the classifier classifies all positives.
- The line that a random classifier will draw will go from (0, 0) to (1, 1).
- In (0, 1), the classifier is completely accurate, never making a mistake, and has no false positives or false negatives.
- In the case of (1, 0), the classifier is perfectly incorrect since it lacks both true negatives and true positives. It is therefore constantly in error.

We may test a logistic regression model several times with different classification criteria to get the points of a ROC curve, however this is ineffective. The AUC is then used to resolve this inefficiency.

AUROC

Area under the ROC Curve is referred to as AUC. In other words, the AUC calculates the total area under the ROC curve in two dimensions, from (0.0) to (1.1). The better the classifier, the greater the area under the curve.
The following two factors make AUC desirable:

- AUC is unchanging: instead than calculating the absolute value of the predictions, it ranks them.
- Classification thresholds have no bearing on AUC: it evaluates the accuracy of the model's projections without regard to the categorisation criterion used.

Chapter 4

State of the art

Now that we understand how a heart works, how diseases affect it, and what neural networks are and particularly what deep learning is, let us look at artificial intelligence in the medical field, specifically in cardiology.

Artificial intelligence, particularly machine learning, can assist in the processing of medical data and providing critical information to healthcare professionals, thereby improving health outcomes and patient experience [62].

AI fulfills six functions in today's medical world:

- Predictive medicine: disease prediction and/or disease evolution.
- Precision medicine: for treatments tailored to each individual patient.
- Decision support: for more dependable diagnostics and therapies.
- Assistance robots: to physically assist people in need.
- Computer assisted surgery: to lighten the workload during an intervention.
- Prevention: to avert new pandemics such as covid.

All of these categories cover a wide range of applications, from medication prescribing to technical assistance during an intervention to preventing illnesses in their early stages. The best example to demonstrate the advance of artificial intelligence in the medical world is the Covid-19 pandemic. For almost 2 years, this highly contagious virus has turned the medical system upside down on a worldwide level. This pandemic then provided an opportunity to highlight even more artificial intelligence in the medical field, and more particularly in the detection of cluster, so when 2 or more cases are linked [63]. Detection and diagnosis of a patient's infection were also significantly faster as AI can quickly analyse and detect irregular symptoms and other red flags. AI allowed for speedier medication testing to establish efficacy against Covid, as well as faster vaccine development than ever before. There are still many examples, all proving the usefulness of artificial intelligence.

4.1 AI in cardiology

As previously stated, the goal of artificial intelligence is to improve the speed and efficiency of a patient's treatment. In the case of heart problems, specifically arrhythmias, the primary goal is disease prevention.

The irregularity of the RR intervals and the absence of a P-wave on an ECG trace lasting at least 30 seconds are used to diagnose AF [24]. Screening for AF episodes can be

opportunistic (i.e., during a clinical examination during a routine consultation, followed by electrocardiogram confirmation) or systemic (i.e., offered to a specific population) [64]. The European Society of Cardiology advocated for systematic opportunistic screening of patients over the age of 65 [65], and a systematic screening for those over the age of 75 or at high risk (stroke or transient ischaemic attack). Indeed, it has been demonstrated that opportunistic screening is more effective than routine follow-up in randomised groups of people over the age of 65 [66].

4.1.1 AF detection

The first indirect suspicion of AF may arise during a brain computed tomography (CT) or magnetic resonance imaging (MRI) scan when the specialist encounters bilateral ischaemic lesions, or in order words, a temporary interruption of the blood supply to the brain.

There are currently different methods for detecting AF in individuals with various characteristics: direct ECG (i.e., opportunistic screening), implanted devices (i.e. directly on the heart), loop recorder implantable, a portable device under the skin and easy to access, and holters, portable electrocardiographs that record the electrical activity of the heart in real time, which are worn for a period generally between 24 and 72 hours [67].

The data is then analysed by a cardiologist at the end of the recording period. The longer the ECG recording time, the greater the likelihood of detecting AF [68]. However, the amount of data recorded is sometimes too much for one person to handle, and critical information can be missed. This is where artificial intelligence enters the picture.

4.1.2 AI usage for AF

Because the amount of volume to be analysed by the clinician is frequently large, various techniques have been implemented; each capable of detecting, predicting, or identifying AF. A deep neural network model with CNN layers made up of bidirectional gated recurrent units (GRU), for example, has been shown to be effective in predicting episodes of paroxysmal atrial fibrillation [69].

Another method of first classifying ECGs to improve prediction accuracy of paroxysmal atrial fibrillation was also successful, yielding better results than most other prediction methods [70]. In general, using artificial intelligence to detect AF improves patient diagnosis and treatment while also allowing the detection of new types of arrhythmia [71]. Other deep learning methods, like the end-to-end RNN-CNN approach, have proven effective in real-time detection of AF, facilitating prompt patient care [72].

New, more widely available technology have also enabled the development of new methods for detecting cardiac arrhythmias. This is the case with connected watches from Samsung or Apple, for example. These connected watches contain a heart rate monitor that provides information on the pulse rate at any time using the photoplethysmography approach (analysis of the pulse by the LED of the device) [22]. In Apple's case, the priority algorithm does not always monitor the user, but does so opportunistically, when there is sufficient signal to be collected and analyzed as shown in figure 4.1. If five of six sequential tachygrams, including the first one, are classified as irregular within a 48-hour period, the user is alerted to the risk of arrhythmia.

In conclusion, using a deep neural network to detect arrhythmias is as accurate as a cardiologist in more than 95% of cases [73]. It is important to note, however, that AI will never replace the practitioner, who is always the final decision maker. It is essential to understand that this new technology is a supplement, not a replacement.

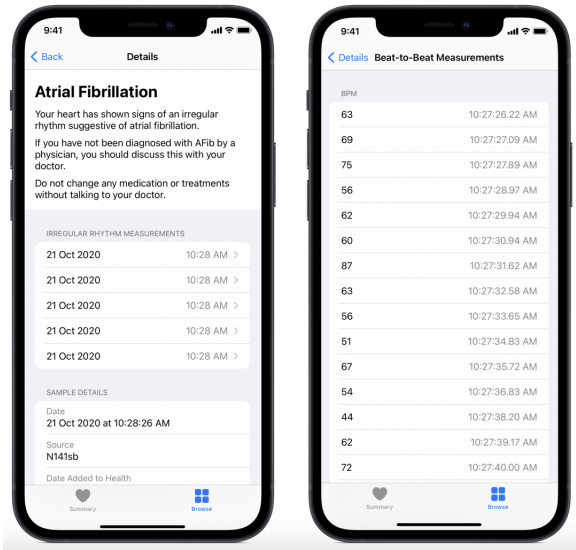


Figure 4.1: Health App View of Irregular Rhythm Measurements [22]

4.1.3 New techniques

For detection, the most compelling research to date comes from Duan et al. [74]. They suggested the use of RR intervals for arrhythmia detection and achieved an accuracy of 98.4% using a Support Vector Machine (SVM), a machine learning technique. Similarly, Castro et al. introduced a method for paroxysmal fibrillation detection [23]. Their approach relies on the P-wave, heart rate variability, and other potential measurements implemented through a feed-forward network. They attained an accuracy of 97.4%. Numerous other studies have also addressed detection, consistently delivering efficient and favourable results. While this is only the beginning, the outcomes so far are encouraging.

In 2019, Attia et al. published a paper offering a revolutionary and more convenient way to detect cardiac arrhythmias in patients of all ages in a non-intrusive manner [75]. Using a regular 10-second, 12-lead ECG, they created an artificial intelligence using a CNN capable of recognising the electrocardiographic signature of an artial fibrilation present during normal sinus rhythm.

Their research is based on mounting evidence that people who acquire atrial fibrillation have anatomical changes in their atria that predispose them to an atrial arrhythmia, even though their hearts appear to be normal. The pathophysiology of an embolic or ischemic stroke may be influenced by these alterations. As a result, they proposed that a neural network might be trained to recognise small alterations brought on by structural abnormalities associated with atrial fibrillation in a regular 12-lead ECG recorded during a normal sinus rhythm.

While some research have looked at distinct ECG characteristics, such as the characteristics of the P-wave, to predict AF (Chen et al. 2021 [76]), some have even tried to mix CNN and RNN to make real-time detection (Andersen et al. 2019 [77]) and others have tried to automate the extraction of features, thus characteristics of the ECG that can be exploited (Wang 2020 [78]), they all have the same defect: the model and utilisation of the neural network need a significant amount of computer power, which is often not available to any clinician.

A neural network with adequate depth and training on more than 500 000 ECGs might be effective at extracting and remembering subtle characteristics that are not often seen or

officially recorded by human observers. Once trained, the network may be used to interpret signals from any 12-lead digital standard ECG with little computational power needed; for instance, a smartphone could do so. This advance would allow any practitioner to verify during an opportunistic examination the possibility of AF in the near future. This would allow a relatively accurate detection (about 90% for the AUC) at lower cost.

4.1.4 Summary Table

Authors	Article Title	Description	Accuracy
Attia et al. [71]	An artificial intelligence-enabled ECG algorithm for the identification of patients with atrial fibrillation during sinus rhythm: a retrospective analysis of outcome prediction	Identification of patients with a history of AF using 10-second, 12-lead ECGs with a ResNet model.	79.0%
Chen et al. [76]	Atrial fibrillation detection based on multi-feature extraction and convolutional neural network for processing ECG signals	Atrial fibrillation detection algorithm based on multi-feature extraction of atrial activity. Use of convolutional neural network	98.9%
Andersen et al. [77]	A deep learning approach for real-time detection of atrial fibrillation	Extraction of high level features from segments of RR intervals. Utilisation of combined CNN and RNN	98.8
Jibin Wang [78]	Automated detection of atrial fibrillation and atrial flutter in ECG signals based on convolutional and improved Elman neural network	Detection of atrial fibrilation and atrial flutter from ECG signals. Utilisation of CNN.	98.8%
Duan et al. [74]	Accurate detection of atrial fibrillation events with R-R intervals from ECG signals	Use of RR intervals extracted from ECG signals for arrhythmia detection. Utilisation of SVM.	98.4%
Castro et al. [23]	Methodology for detection of paroxysmal atrial fibrillation based on P-Wave, HRV and QR electrical alternans features	Detection of Paroxysmal Atrial Fibrillation using ECG signals focusing on P-Wave, QRS complex, and heart rate variability. Use of feedforward neural network.	97.4%

Table 4.1: Overview of key articles and studies

Chapter 5

Materials

5.1 Datasets description

5.1.1 MIT-BIH Normal Sinus Rhythm Database

The Normal Sinus Rhythm Database (NSRDB) [79] is a compilation of 18 annotated, 2-channel ECG recordings sourced from PhysioNet, which is a platform dedicated to the exploration of physiologic signals. The database contains electrocardiogram data of 18 individuals exhibiting normal sinus rhythm. The recordings were taken for a period of 30 minutes.

The NSRDB includes the following elements:

- There are 18 data files in the .dat format, each of which comprises a 30-minute electrocardiogram data set with a sampling rate of 128 Hz.
- Each recording is accompanied by a header file containing metadata pertaining to the recording. This metadata includes the age and sex of the subject, as well as the ECG lead configuration. The header file is denoted by the file extension ".hea".
- The files denoted by the extension ".atr" comprise annotations for beats in the electrocardiogram data, which specify the locations of the QRS complex and the types of beats.

Figure 5.1a comprises three subplots, each showcasing distinct aspects of the processed ECG data. These visual representations enhance our understanding of the data and provide a clearer view of the material we are analysing. It is structured as follows:

- The uppermost panel illustrates the raw electrocardiogram signal represented in grey and the processed ECG signal represented in blue for the designated time interval, which is the initial 10 seconds in this instance. The filtered signal is obtained after the implementation of a bandpass filter for the elimination of noise and artefacts. The horizontal axis denotes time in seconds, while the vertical axis represents the amplitude of the ECG signal. The red vertical lines signify the R-peaks that have been detected. These peaks correspond to the points of highest amplitude in the QRS complex of the ECG waveform. R-peaks play a crucial role in determining the heart rate and evaluating the rhythmicity of the cardiac cycle.
- The panel at the bottom displays the heart rate that is instantly computed from the R-peaks that have been detected. The independent variable is time measured in seconds, while the dependent variable is heart rate measured in beats per minute.

The plot facilitates the visualisation of heart rate variability and the detection of any anomalies or variations in the heart rate within the designated time frame.

- The right panel displays a set of individual heartbeats that have been extracted from the cleaned ECG signal. Each heartbeat waveform is displayed on top of one another. The horizontal axis denotes time in seconds, while the vertical axis signifies the amplitude of the ECG signal. The ECG plot facilitates the observation of distinct ECG components such as the P-wave, QRS complex, and T-wave. It also enables the comparison of the morphology and duration of these components across multiple heartbeats. This aids in detecting any anomalies or aberrations in the cardiac rhythm.

5.1.2 MIT-BIH Atrial Fibrillation Database

The Atrial Fibrillation Database AFDB [80] is a compilation of ECG recordings that are tailored for the examination of AF, which is a frequently occurring cardiac arrhythmia. The database is being hosted on PhysioNet, which was developed by the Massachusetts Institute of Technology (MIT) and Boston's Beth Israel Hospital (BIH).

The AFDB consists of 25 ambulatory ECG recordings, each lasting around 10 hours. The recordings are two-channel and have been sampled at a rate of 250 Hz. The resolution of the recordings is 12-bit and the range is 20 mV. The dataset comprises audio recordings obtained from a sample of 23 participants, consisting of 18 males and 5 females, with ages ranging from 32 to 89 years. The participants exhibited paroxysmal or sustained atrial fibrillation, atrial flutter, or a combination of both. Certain recordings exhibit ventricular arrhythmias and conduction disturbances as well.

The AFDB comprises annotated records, each of which has been evaluated by a minimum of two expert cardiologists. These experts have utilised the conventional MIT-BIH annotation format to identify ECG beats and alterations in rhythm. The annotations comprise of beat labels, rhythm labels, and signal quality labels. The beat labels categorise every beat as either normal or different types of arrhythmias, whereas the rhythm labels indicate intervals of atrial fibrillation, atrial flutter, or other rhythms. The signal quality labels serve as an indicator for the existence of artefacts or noise in the electrocardiogram signal.

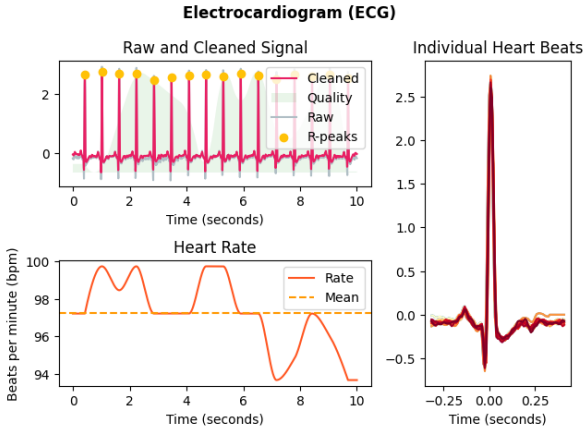
The figure 5.1b allows us to see the difference in rhythm directly and gives us a first analysis of the heart rate. The plotting explanations remain the same as those for figure 5.1a.

5.1.3 MIT-BIH Arrhythmia Database

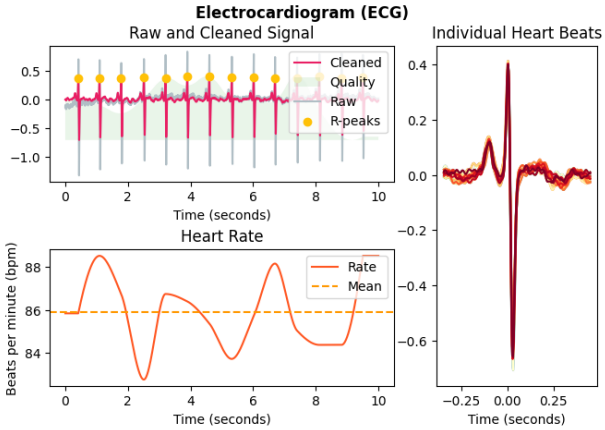
The first release of the MIT-BIH Arrhythmia Database [81], version 1.0.0, was made available on March 1, 2009, and the most recent revision was released on June 16, 2021. George B. Moody and Roger G. Mark created and provided the database.

The database includes 48 snippets of two-channel, half-hour ambulatory ECG recordings. These were gathered from 47 participants who took part in a 1975–1979 study at Boston's Beth Israel Hospital Arrhythmia Laboratory. These recordings were chosen from two distinct groups: from a larger collection of 4000 24-hour ambulatory ECG recordings drawn from more than 60,000 healthy people, 23 recordings were chosen at random. In order to include less frequent but clinically significant arrhythmias that would not be adequately represented in a small random sample, the remaining 25 recordings from the same set were purposefully chosen.

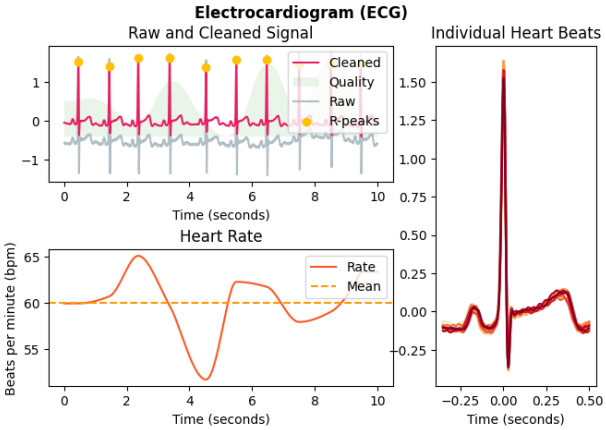
With an 11-bit resolution over a 10-mV range, the recordings were digitalized at a rate of 360 samples per second for each channel. Two or more cardiologists independently annotated each record. The final, computer-readable reference annotations for each beat were created after any differences between the annotators were settled. The database contains about 110,000 annotations in total. A wide variety of arrhythmia detection algorithms have been developed and extensively tested using the MIT-BIH Arrhythmia Database. The figure 5.1c provides a brief overview of the ECGS in this database and is presented exactly as it was in the previous databases.



(a) Normal Sinus Rhythm (NSRDB)



(b) Atrial Fibrillation (AFDB)



(c) Arrhythmia (MIT-BIH)

Figure 5.1: Comparison of ECG data from different databases

Table 5.1 consolidates the essential characteristics of each dataset, including the number of subjects, duration, sampling frequency, annotations, channels, and release date. This provides a comprehensive overview, enabling a clear comparison of the various datasets utilised in our study.

Feature	NSRDB	AFDB	MIT-BIH
Database Name	Normal Sinus Rhythm Database	Atrial Fibrillation Database	MIT-BIH Arrhythmia Database
Subjects	18	25	47
Duration (hours)	24	10	0.5 (each excerpt)
Sampling Frequency	128 Hz	250 Hz	360 Hz
Annotations	Beat	Beat and rhythm	Beat
Channels	2 (Lead II & aVL)	2 (Lead II & aVL)	2 (Lead I & Lead II)
Release Date	1998	1997	1980

Table 5.1: Comparison of Databases Used

5.2 Data Processing Tools

5.2.1 Python

Python [82] is a programming language that is interpreted, object-oriented, and high-level, characterised by its dynamic semantics. The language's advanced built-in data structures, coupled with its dynamic typing and dynamic binding capabilities, render it highly appealing for Rapid Application Development purposes. Additionally, it serves as a scripting or glue language, facilitating the integration of pre-existing components. The syntax of Python is characterised by its simplicity and ease of learning, which places a strong emphasis on readability. As a result, this feature contributes to a reduction in the expenses associated with programme maintenance. Python provides support for modules and packages, thereby promoting the principles of programme modularity and code reuse.

One of the primary benefits associated with Python is its ability to enhance productivity. The absence of a compilation step results in a significantly expedited edit, test, and debug cycle. The process of debugging Python programmes can be considered relatively straightforward, as the occurrence of a bug or erroneous input is unlikely to result in a segmentation fault. In contrast, when the interpreter encounters an error, it triggers an exception. In the event that the programme fails to capture the exception, the interpreter will generate and display a stack trace. A source-level debugger facilitates the examination of local and global variables, the evaluation of arbitrary expressions, the establishment of breakpoints, and the sequential execution of code line by line, among other functionalities. The debugger is implemented using Python, which serves as evidence of Python's ability to introspect. Frequently, the most expeditious method for troubleshooting a programme involves incorporating a limited number of print statements into the source code. The rapid iteration of editing, testing, and debugging greatly enhances the efficacy of this straightforward approach.

The Python programming language offers a comprehensive standard library that is accessible in both source and binary formats at no cost for all major platforms and can be freely distributed. Python's versatility allows it to be utilised for a diverse array of tasks, encompassing both rudimentary scripting and intricate machine learning algorithms. The Python Software Foundation (PSF) is the official entity responsible for the development and promotion of the Python programming language, with the primary objective of furthering the software's capabilities and fulfilling its overarching goals.

Throughout this research, Python was the primary tool of choice. Its user-friendly

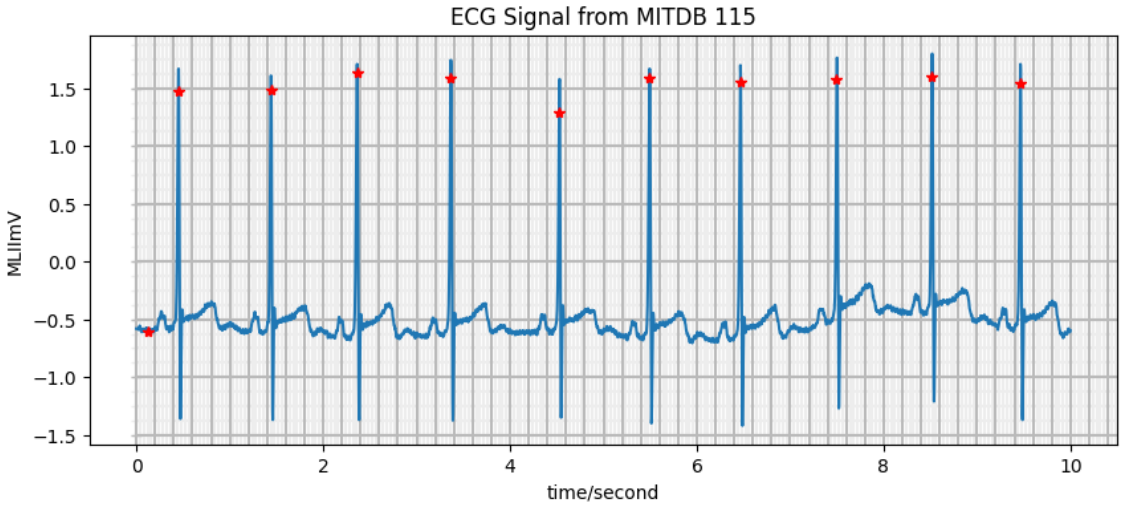


Figure 5.2: Initial visualisation using the WFDB package

nature, combined with its comprehensive capabilities, ensured that the research process was both efficient and effective. The versatility of Python meant that every aspect of the research, from data analysis to visualisation, could be handled seamlessly within a single environment. This not only streamlined the research process but also expedited the time to results, underscoring the power and utility of Python in academic and scientific endeavours.

5.2.2 Waveform Database Software Package

A comprehensive set of tools and libraries called the Waveform Database (WFDB) Software Package has been developed to make it easier to manipulate and analyse physiologic signals and time series [83]. The WFDB software package provides a flexible solution for researchers working with physiological data and was created to be compatible with the databases hosted on PhysioNet, among other sites.

The WFDB software package stands out for its portability as a result of its development on UNIX platforms and subsequent adaptation for a variety of operating systems, including DOS, MS-Windows, Macintosh, and Linux. This wide compatibility makes the software usable in a variety of computational environments, increasing its accessibility to researchers around the world.

The flexibility of the WFDB software package is one of its main advantages. The package offers more than 50 command-line tools, each of which aims to carry out a particular task. These cover everything from fundamental operations like file conversion and signal processing to more intricate ones like complex signal analysis and algorithm evaluation. This adaptability makes the software a useful tool for a variety of projects by enabling researchers to customise its use to their particular needs.

The software package for the WFDB is intended to be interoperable with a variety of data formats. This covers, among others, the formats employed by EDF, SCP-ECG, HL7 aECG, DICOM, Nihon Kohden, and GE MUSE XML ECG. The WFDB software package can be seamlessly incorporated into current research workflows thanks to its wide range of compatibility.

The WFDB software suite has a wide range of applications. It is frequently used to view, examine, and simulate physiological signals in accurate ways. It is extensively used in

academic and medical institutions for research and education across a wide range of fields, including biomedical engineering, clinical cardiology, and medical informatics. Throughout the research, the WFDB package played a pivotal role, especially during the pre-processing phase. It streamlined the process of retrieving various ECGs and facilitated the creation of initial plots, providing a preliminary glimpse into the data at hand. While the visualisation might not capture every nuance, Figure 5.2 offers a general overview of the ECG being studied.

Readers are advised to visit the PhysioNet website for more details, tutorials, and guides on the WFDB software suite.

5.2.3 NeuroKit2: A Comprehensive Tool for Neurophysiological Data Analysis

NeuroKit2 [84] is a Python package that has been developed with a strong focus on its ability to effectively analyse neurophysiological data. For researchers and practitioners in neuroscience and related fields, it offers a comprehensive suite of functionalities that make it easier to process and analyse physiological signals.

The package exhibits a user-friendly interface, rendering it accessible to individuals with diverse levels of programming expertise, including novices in Python. NeuroKit2 provides a diverse array of capabilities, facilitating the execution of advanced analyses on neurophysiological data. One of the key characteristics of NeuroKit2 is its inherent dynamism and adaptability. The developers actively promote and solicit contributions from the user community, encompassing a wide range of involvement, from minor corrections such as rectifying typographical errors to significant improvements such as introducing novel features. The utilisation of a collaborative approach guarantees the ongoing relevance and progressive development of NeuroKit2.

The package's popularity can be attributed to its utility and user friendliness, which are evidenced by the rapidly growing and welcoming community of users and contributors. The package in question is experiencing rapid growth within its respective field. Although the authors of NeuroKit2 have made diligent efforts to develop a dependable and resilient tool, it is important to note that no warranty is provided. It is the responsibility of users to ensure that the software is suitable for their specific requirements, and it is recommended that any analyses performed using NeuroKit2 are thoroughly validated. Throughout our research, the NeuroKit2 package proved invaluable. It facilitated swift data manipulation, ensuring accurate results. Moreover, its diverse functionalities, including plotting capabilities, enhanced our data visualisation and comprehension. This package is the most important of them all. Figure 5.1 displays the plotting quality and provides an initial visual analysis.

To access comprehensive information, tutorials, and guides on NeuroKit2, readers are advised to refer to the official NeuroKit2 documentation.

5.2.4 PyWavelets

PyWavelets [85] is an open-source software package designed for conducting wavelet transforms in the Python programming language. The software offers a user-friendly interface that facilitates efficient utilisation of low-level C and Cython capabilities, thereby simplifying its adoption and initiation processes. PyWavelets has the ability to execute the Forward and Inverse Discrete Wavelet transforms (DWT and IDWT) in one dimension (1D), two dimensions (2D), and multiple dimensions (nD). Additionally, it can perform Multilevel DWT and IDWT, as well as the Stationary Wavelet Transform, also known as

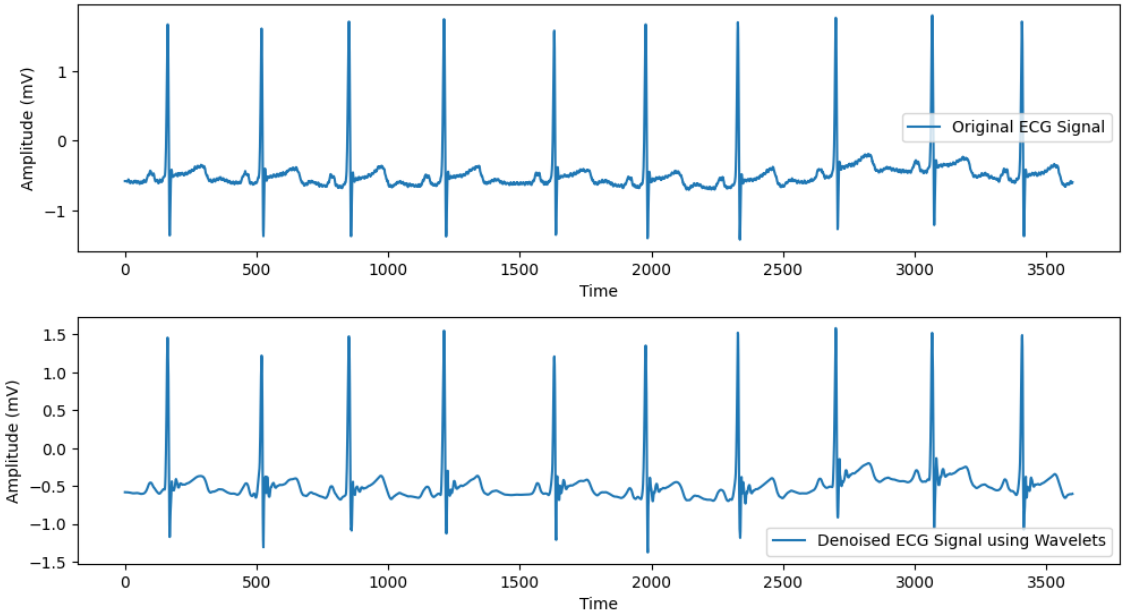


Figure 5.3: PyWavelets signal denoising

the Undecimated Wavelet Transform. Additionally, it provides support for the decomposition and reconstruction of 1D and 2D Wavelet Packets, as well as the 1D Continuous Wavelet Transform.

The software provides a wide range of wavelet filters, exceeding 100 in number, that are pre-installed and also allows for the utilisation of customised wavelets. The computational capabilities of the system encompass both single and double precision calculations, in addition to accommodating real and complex calculations. The findings align with the functionality of the Matlab Wavelet Toolbox.

One of the primary characteristics of PyWavelets is its capacity to calculate approximations of wavelet and scaling functions. The combination of this feature, along with its comprehensive functionality and user-friendly interface, renders it a potent instrument for conducting wavelet analysis in the Python programming language. PyWavelets is distributed under the MIT licence, thereby granting users the freedom to use it in open-source projects without any cost. When incorporating this project into a scientific publication, the developers kindly request that proper citation be given to the project by referencing the Journal of Open Source Software publication.

PyWavelets is primarily employed for its proficiency in signal manipulation and denoising. The package facilitates effortless adjustments to various denoising parameters, making it an invaluable tool for manual signal refinement. Figure 5.3 illustrates an example of the denoising capabilities offered by PyWavelets, showcasing its effectiveness and precision. For a comprehensive understanding of PyWavelets, readers are advised to refer to the PyWavelets documentation, which provides detailed information, tutorials, and guides.

5.2.5 SciPy

The SciPy Signal library [86], a component of the broader SciPy library, provides functionality for the manipulation of electrocardiogram signals. The library encompasses a range of fundamental functionalities that are imperative for the analysis and interpretation of cardiac signals.

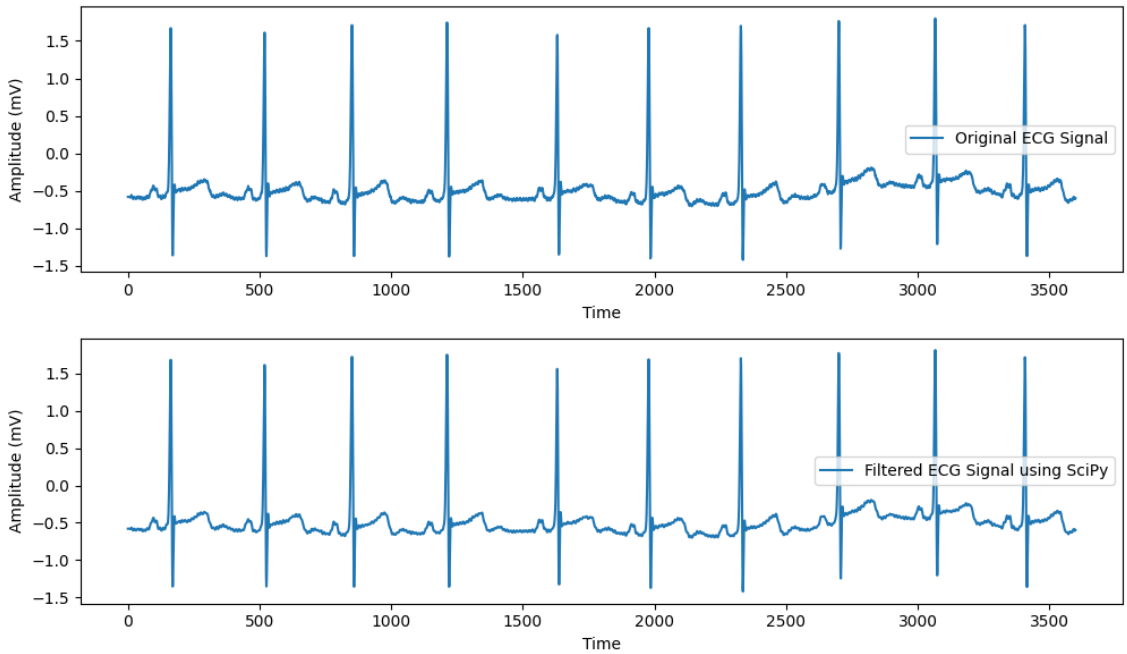


Figure 5.4: SciPy signal cleaning

One notable characteristic of SciPy Signal is its adeptness at signal filtering. The library provides resources for creating and using digital filters, which are essential for removing noise and other undesirable components from electrocardiogram signals that may significantly distort the real signal.

In addition to its filtering capabilities, the SciPy Signal library provides functions for spectral analysis. The use of spectral analysis has the potential to provide significant insights into the regularity of the cardiac rhythm, thereby aiding in the identification of potential cardiac abnormalities.

One notable feature of SciPy Signal is its ability to detect peaks. Accurate peak identification in a signal is of utmost importance in the detection of QRS complexes, P waves, and T waves within (ECG) data. The identification of these components serves as fundamental indicators in the cardiac cycle and forms the foundation for various types of cardiac analysis.

SciPy has been primarily utilised for its signal processing capabilities. Similar to PyWavelets, SciPy offers flexibility in adjusting various parameters, making it ideal for manual signal cleaning. Figure 5.4 illustrates the efficacy of this package in filtering the signal, enabling optimal noise reduction and signal enhancement. To gain a more comprehensive understanding of the functionalities and applications of SciPy Signal, readers are advised to refer to the official documentation provided by SciPy Signal.

5.2.6 Other tools

Additional tools were employed in this thesis, primarily for the purpose of data manipulation:

- NumPy [87]: The foundational Python package for scientific computing is called NumPy. It offers a strong N-dimensional array object and C/C++ and Fortran code integration tools. Many different fields of mathematics and science benefit from it. While the majority of scientific computing in Python begins with the NumPy API,

it also offers a flexible platform for more specialised computations. Once the data was preprocessed, NumPy was employed for efficient data manipulation.

- Pandas [88]: Pandas is a robust open-source Python data analysis and manipulation library. It includes functionality for manipulating numerical tables and time-series data, as well as the data structures and functions required to manipulate structured data. Finance, neuroscience, economics, statistics, advertising, and web analytics are among the academic and commercial domains where it is used. Once the data was pre-processed, Pandas was also employed for efficient data manipulation.

- Matplotlib [89]: Python's Matplotlib library is a complete tool for building static, animated, and interactive visualisations. It offers a variety of functionalities for building various types of plots. Matplotlib is primarily used in this thesis to visualise ECG signals before, during, and after preprocessing.

Chapter 6

Methods

6.1 Objectives

The primary objective of this research is to determine whether atrial fibrillation can be identified by its subtle traces even when the heart is in normal sinus rhythm and functioning as it should.

This research has important potential advantages. If the models are successful, they may be able to recognise micro-changes unique to atrial fibrillation, enabling the early detection of this condition. This might improve the delivery of cardiac preventative care. For instance, a doctor might discover a patient who appears to be in good health and is asymptomatic but actually has atrial fibrillation during a routine checkup. Early detection would enable prompt treatment, possibly delaying the development of more serious cardiac complications.

In essence, the goal of this thesis is to investigate the forefront of predictive cardiac care by using machine learning to find hidden patterns and give early warnings for conditions like atrial fibrillation.

In order to identify patients with atrial fibrillation during normal sinus rhythm, we found inspiration in a range of articles on the subject. This led us to devise four distinct methodologies, each promising in its own right and offering different perspectives on the problem at hand.

The first methodology was influenced by the work of Attia et al. in their article titled "An artificial intelligence-enabled ECG algorithm for the identification of patients with atrial fibrillation during sinus rhythm: a retrospective analysis of outcome prediction" [71]. In the study done by Attia et al., a novel methodology was suggested for identifying patients who had experienced atrial fibrillation in the past during times with a normal sinus rhythm. This was accomplished using 10-second, 12-lead electrocardiograms. The authors developed a ResNet model using a sizable dataset of 180,922 patients. The model's performance was impressive, achieving an AUC of 0.87, a sensitivity of 79.0%, and a specificity of 79.5%. This shows how successful the model is at discovering patients with a history of AF. The model's performance, however, was seen to suffer in scenarios involving larger populations for screening, indicating the need for additional calibration and validation. The authors came to the conclusion that AI-enabled ECGs might improve AF identification, thereby offering a useful tool for early intervention and treatment. Inspired by this approach, we implemented a similar method in our research.

However, we also recognised the potential limitations of using a fixed duration ECG reading. To address this, we decided to adopt a different approach for the second methodol-

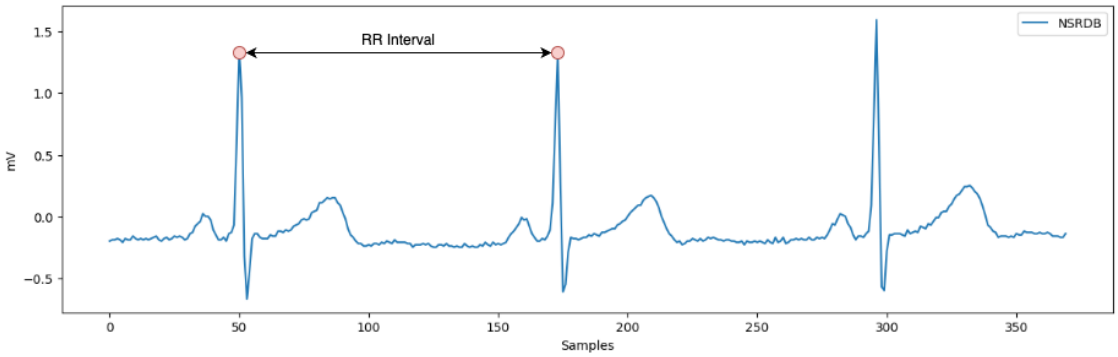


Figure 6.1: Third methodology: representation of a R-R Interval

ogy, focusing on a fixed number of heartbeats rather than a fixed duration. It is important to keep in mind that everyone has a unique rhythm, which causes their ECGs to vary in size for a fixed number of beats. Therefore, it is also a matter of analysing the ECG over time in this situation and also taking into account the ECG's variable duration.

The third methodology we implemented was inspired by the research conducted by two groups of scientists. The first inspiration came from the work of Duan et al., as presented in their article "Accurate detection of atrial fibrillation events with R-R intervals from ECG signals" [74]. In this study, the authors proposed a novel approach for arrhythmia detection, which utilizes RR intervals (see figure 6.1) extracted from ECG signals. They employed machine learning, specifically a feedforward neural network, to classify heartbeats as normal or abnormal. The effectiveness of their method was demonstrated using three databases for training and testing: MIT-BIH Arrhythmia, NSRDB, and AFDB. The overall accuracy achieved was 97.4%, indicating the potential of RR intervals for arrhythmia detection in automated ECG analysis.

The second source of inspiration was the work of Faust et al., as detailed in their article "Automated Arrhythmia Detection Based on RR Intervals" [90]. In their research, Faust et al. created a deep learning algorithm that uses RR interval signals to distinguish between various arrhythmias, specifically atrial fibrillation, atrial flutter, and normal sinus rhythm. The model used was a ResNet. The algorithm's high accuracy (99.98%), sensitivity (100.00%), and specificity (99.94%) were achieved in a 10-fold cross-validation using data from 4051 subjects. Their research utilises RR intervals for arrhythmia detection, which inspired us to incorporate this concept into our own work.

Finally, our attention was drawn to the work of Castro et al. in their article "Methodology for detection of paroxysmal atrial fibrillation based on P-Wave, HRV, and QR electrical alternans features" [23]. The authors suggest using ECG signals to detect Paroxysmal Atrial Fibrillation automatically. They focus on the P-Wave, QRS complex, and heart rate variability as they extract six features from the ECG as we can observe in figure 6.2. On the Physionet MIT-BIH arrhythmia database, a feedforward neural network with an impressive accuracy of 97.4% is used for detection. The study emphasises the benefit of simultaneously using data from P-Wave, HRV, and QR Electrical alternans, which increases detection accuracy in comparison to methods using fewer features. Their research formed the basis for the fourth methodology in our research.

In summary, our research methodologies are a culmination of insights and ideas inspired by the pioneering work of various researchers in the field. Each methodology offers a different perspective through which to view and tackle the problem of identifying atrial fibrillation during normal sinus rhythm.

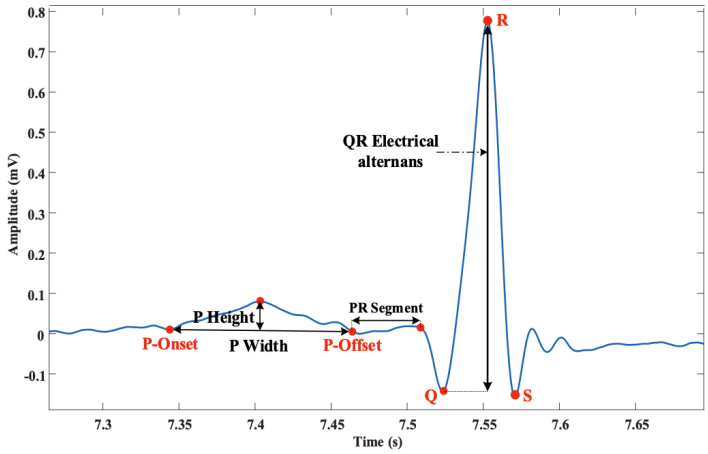


Figure 6.2: P-Wave, QRS Complex and other measurements [23]

6.2 Preprocessing

To solve the problem, we employed several approaches, each necessitating distinct preprocessing steps on our datasets. For the models that utilised 10-second ECGs and heartbeats, advanced preprocessing was required to ensure the data was clean and suitable for analysis. This allowed us to extract the relevant features from these ECGs.

On the other hand, the final two models did not require direct use of the ECGs. Instead, they relied on information derived from the ECGs, specifically the RR intervals (also called as RRI) and P-wave information. Consequently, these models did not necessitate any modifications to the ECGs themselves. This distinction in preprocessing requirements underscores the diverse nature of the methodologies we adopted in this research.

6.2.1 Databases

The three databases that were used in this study were carefully analysed in the preliminary stages. The first database to be examined was the Atrial Fibrillation Database. It was discovered that samples 00735 and 03665 did not have the.dat files, which are essential because they hold the ECG data. These samples were therefore deemed useless and were not included in the study. The evaluation of the Normal Sinus Rhythm Database came next. All of the database's files were found to be complete and formatted correctly upon inspection. Therefore, the study included all of the NSRDB's samples. The MIT-BIH Arrhythmia Database was evaluated last. The samples were all included in the study because, like the NSRDB, all of the files in this database were complete.

After carefully examining the three databases, we can see the raw data in Figure 6.3, which gives a summary of the different beat types found in each. Note the scaling of the X-axis, which varies for each plot. While only three beats are displayed, the X-axis will differ for each individual due to variations in cardiac rhythm. The databases are presented in the following order: Normal Sinus Rhythm Database, Atrial Fibrillation Database, and lastly, the Arrhythmia Database. The X-axis in the visual representations of these databases represents time, denoted by the term "Sample" and the Y-axis represents the millivolts (mV) produced by the electrical activity of the heart. This makes it possible to see the heart's rhythm clearly and precisely over time, giving useful information for the identification and evaluation of atrial fibrillation.

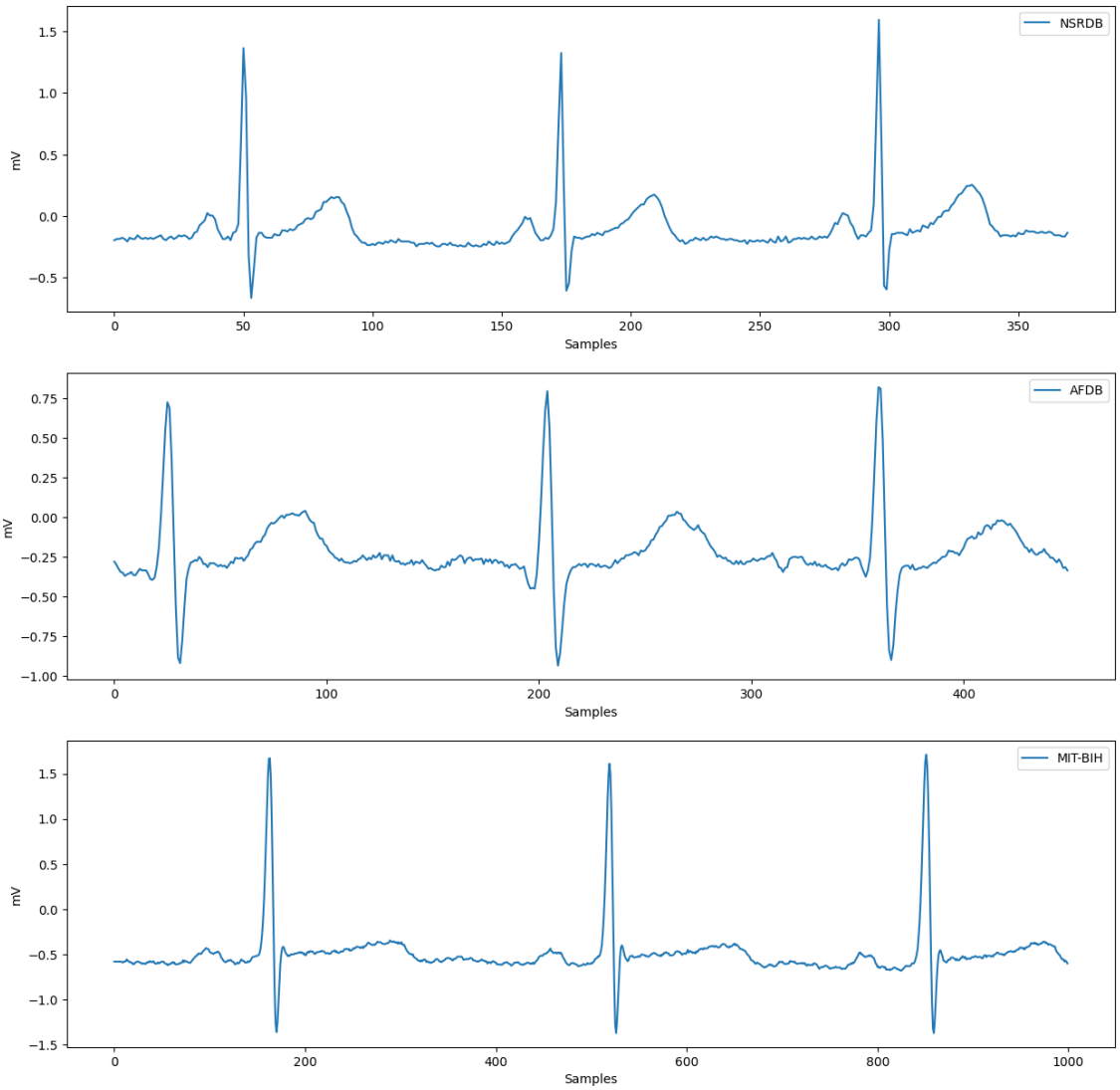


Figure 6.3: Raw signals of the first three beats, with varying X-axis scaling due to individual cardiac rhythm differences.

In the case of NSRDB, the ECG beats are well-defined, with clear P-waves (indicating atrial depolarization), QRS complexes (signifying ventricular depolarization), and T-waves (representing ventricular repolarization). This visualisation serves as a reference point, embodying the characteristics of a healthy beat devoid of any fibrillation. Moving on to AFDB, we observe an absence of the Q-wave and an irregular T-wave, while the QRS complex appears to be normal. This deviation from the norm is indicative of the abnormalities associated with atrial fibrillation. Lastly, in the MIT-BIH database, although the QRS complex is within normal parameters, the Q-wave and T-wave are notably diminished. For a comprehensive understanding of the P-wave, QRS complex, and T-wave, refer back to Figure 2.3. This Figure provides detailed explanations and visual representations of these crucial components of an ECG beat.

	Lead I	Lead II	aVL
NSRDB		×	×
AFDB		×	×
MIT-BIH	×	×	

Table 6.1: Availability of leads in the different databases.

6.2.2 Common base

The databases utilised in this study each contain two distinct signals. For both the Normal Sinus Rhythm Database and the Atrial Fibrillation Database, these signals correspond to Lead II and aVL, or augmented vector left. Similarly, the MIT-BIH Arrhythmia Database comprises signals from Lead I and Lead II (the leads placement can be seen in Figure 2.8). Table 6.1 enumerates the leads available for each database.

When comparing the databases, it is clear that Lead II is a common signal present in all three. This commonality is significant as Lead II is frequently employed as the primary lead in continuous monitoring scenarios, such as telemetry or surgical procedures. The reason for its prevalent use lies in the orientation of the heart's electrical axis towards Lead II. This orientation allows Lead II to provide an optimal view of the P-wave, QRS complex, and T-wave, which are the principal components of the ECG waveform. Therefore, the focus of this study will be on the analysis of Lead II signals to ensure effective comparison across the databases.

In this research, the primary focus is on what are referred to as normal QRS complexes or normal segments within the context of this study. The extraction of these normal segments is the first step in this procedure.

While there are many automated and manual methods for extracting QRS complexes, it is important to remember that these techniques do not provide information about the condition of a QRS complex. Only a healthcare professional can determine clinically whether a QRS complex is normal or abnormal. As detailed in section 5.1, each ECG in the databases is accompanied by either a `.qrs` or `.atr` file. The `.qrs` files contain annotations for the QRS complexes, allowing for the direct identification of their location within the ECG and their classification as 'normal' or 'abnormal'. On the other hand, `.atr` files contain reference annotations, which are typically generated through a combination of automated algorithms and manual review by one or more expert annotators. These annotations can mark a variety of features within the signal, including the locations of QRS complexes or heartbeats, changes in heart rhythm, or issues related to signal quality as we can observe in Figure 6.4. With the establishment of a unified foundation and a reliable method for extracting normal segments, we are now well-prepared to proceed with the rest of the pre-processing.

6.2.3 10-second methodology preprocessing

After finding inspiration in the work of Attia et al., we started preprocessing the data for the first method. It is worth noting that they did not preprocess the data for their study. This could be because they directly oversaw the data collection procedure, allowing them to establish some uniform standards early on. But in our case, the information came from a variety of databases. Because standardising the data necessitated the use of multiple preprocessing techniques, this increased the complexity. To enable future use with different machine learning models, the data needed to be as consistent as possible.

We used the scientific literature, concentrating on studies that used the same databases

Annotations

Annotator: atr

Time (elapsed)	Sample #	Type	Sub	Chan	Num	Aux
00:00:00.050	18	+	0	0	0	(N
00:00:00.214	77	N	0	0	0	
00:00:01.028	370	N	0	0	0	
00:00:01.839	662	N	0	0	0	
00:00:02.628	946	N	0	0	0	
00:00:03.419	1231	N	0	0	0	
00:00:04.208	1515	N	0	0	0	
00:00:05.025	1809	N	0	0	0	
00:00:05.678	2044	A	0	0	0	
00:00:06.672	2402	N	0	0	0	

Figure 6.4: Example of an annotation file for record 100 in the MIT-BIH database.

as our research, to accomplish this. Finding the most popular and, more importantly, the most dependable preprocessing techniques for ECG data was our objective. In order to prepare the data for the machine learning models that would be used later in the research process, it was essential to take this approach. Several potential articles were identified from the initial selection, but two were specifically chosen to guide the preprocessing phase of this research due to their outstanding relevance.

The first article, "Deep Learning-Based ECG Arrhythmia Classification: A Systematic Review" by Xiao et al. [91], proposed a methodology that included data augmentation, resampling at 360 Hz and denoising. These techniques were considered beneficial for improving the quality and diversity of ECG data.

The second article, "Deepaware: A hybrid deep learning and context-aware heuristics-based model for atrial fibrillation detection" by Kumar et al. [92], proposed an alternative strategy. The authors suggested resampling the data at 250 Hz and applying a band-pass filter in order to eliminate baseline wanders (below 0.5 Hz) and high-frequency noises (above 40 Hz). In addition, they used a Savitsky-Golay filter to smooth the ECG signals and improve the data's clarity.

This study adopted a hybrid preprocessing strategy by combining the insights from both of these articles. This method aimed to maximise the cleanliness and usability of ECG signals, laying the foundations for the subsequent application of machine learning models. The preprocessing steps are executed in the following sequence: identifying the 10-second window, inverting the ECG, resampling the ECG, cleaning the ECG, rescaling the ECG, and finally retaining only the high-quality segments.

Identifying the 10 second window

The objective is to extract ECG segments labelled as normal sinus rhythm, which is the standard rhythm of a healthy heart. The segments are extracted based on the QRS complex. It is essential to precisely segment the ECG data into manageable segments. Along with the raw ECG data, each ECG file also contains additional metadata, such as the sampling frequency (recorded as `record.fs`), which represents the number of samples recorded per second. The window length in samples for a given window length in second can be

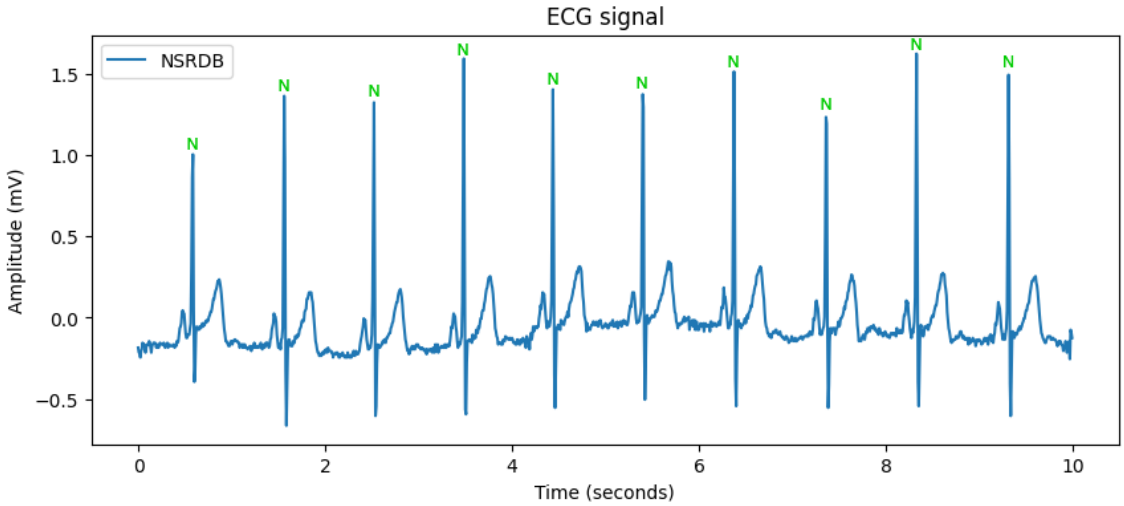


Figure 6.5: Normal 10-second segment

calculated using this information.

Consider the standard sampling rate for ECG data, which is 360 samples per second, as an example. If we want to create a window length of 10 seconds, we calculate it as $window_length = 360 * 10$ in samples. A window length of 3,600 samples is the result. Basically, each window contains 3,600 samples, or an ECG data duration of 10 seconds.

Iterating through the QRS annotations, which are labels indicating the type of heartbeat at each point in the ECG, starts the process. If a QRS complex is labelled with the letter 'N', which stands for normal sinus rhythm, we proceed to define a window around it. The window begins at the location of the current QRS complex, and it ends 10 seconds later, or, if less time has passed since the start of the QRS complex, at the conclusion of the ECG. The remainder of the window must also be marked as normal, not just the first QRS. To accomplish this, we identify all QRS annotations within the window. We extract the corresponding symbols representing the type of heartbeat and check to see if they are all labelled as 'N'. If they are, the entire window represents a normal sinus rhythm.

Finally, if the entire window is found to be normal, the corresponding ECG segment is extracted as we can observe in figure 6.5. This segment, a continuous 10-second portion of the ECG representing normal sinus rhythm, can then be used for further analysis or as input to a machine learning model. The pseudocode is this can be found in Algorithm 1.

ECG inversion

The inversion of an ECG signal may be required due to the placement of the electrodes during the ECG recording. The polarity of the ECG can be inverted at times, causing the waveforms to appear upside down. This inversion has no effect on the information content of the ECG, but it can complicate signal analysis and interpretation.

The NeuroKit2 package includes a pre-existing method called `ecg_invert()` that is intended to detect and correct any inversion in an ECG signal. This function works by subtracting the average first. It then moves a 2-second sliding window across the signal, identifying the original value corresponding to the maximum of the squared signal within each window. If the function detects that the median of these maximum values is negative, it concludes that the signal is inverted and proceeds to correct it. An example can be seen in Figure 6.6.

Algorithm 1 10-second Window Extraction

```
Load QRS annotations
Initialize i = 0
while i < len(QRS annotations) − 1 and QRS annotation sample[i] < len(ECG) do
    if QRS annotation symbol[i] is 'N' then
        Calculate the start and end of the window
        if the next 10 seconds are also labeled as 'N' then
            Locate the indices of annotations that fall within the window
            Extract the symbols within the window
            if all symbols are 'N' then
                Extract the window
            end if
        end if
    end if
    Increment i
end while
```

ECG resampling

We work with several databases, each with a different signal frequency. NSRDB operates at 128 Hz, AFDB at 250 Hz, and MITBIH at 360 Hz. This variation in reference frequencies reflects the absence of a standardised frequency in the field of medical signal processing.

Out of the 13 articles studied for the preprocessing of 10-second methodology, seven used resampling at 360 Hz, four at 500 Hz, and two at 250 Hz. Given these findings and the prevalence of 360 Hz resampling in the literature, it has been decided to resample all ECG signals at 360 Hz for consistency. Neurokit2's automatic method `signal_resample` has been used for this. Resampling at higher frequencies inherently extends the signal, necessitating interpolation between each data point, especially for signals that started with lower initial frequencies. The Fourier method was specifically chosen for this interpolation task due to its unique advantages when dealing with periodic signals like ECGs. This method leverages the principle of representing signals in the frequency domain, allowing for a more accurate and smooth reconstruction in the time domain. While various interpolation techniques are available, the Fourier method's ability to precisely handle the nuances of ECG signals and preserve their inherent characteristics made it the most suitable choice for this application.

The Fourier Transform is used in the Fourier method, also referred to as spectral interpolation, to represent a signal in its frequency domain [93]. The procedure entails converting the initial signal to the frequency domain, expanding this domain by introducing additional points (thereby introducing additional sinusoidal components), and then converting the expanded signal back to the time domain. The quasi-periodic nature of ECG signals, which is characterised by the regular cycle of the heart's activity, makes this method effective.

The precise form, duration, and amplitude of these cycles, however, may differ between beats as a result of normal heart rate variability, autonomic nervous system activity, and other dynamic physiological processes. Furthermore, cardiac arrhythmias and other heart conditions can cause significant deviations from periodicity. Since ECG signals are not strictly periodic, even though they have a distinct cyclic pattern, Fourier-based signal interpolation must take this into account.

The resulting ECG signals after resampling are shown in figure 6.7, where the process

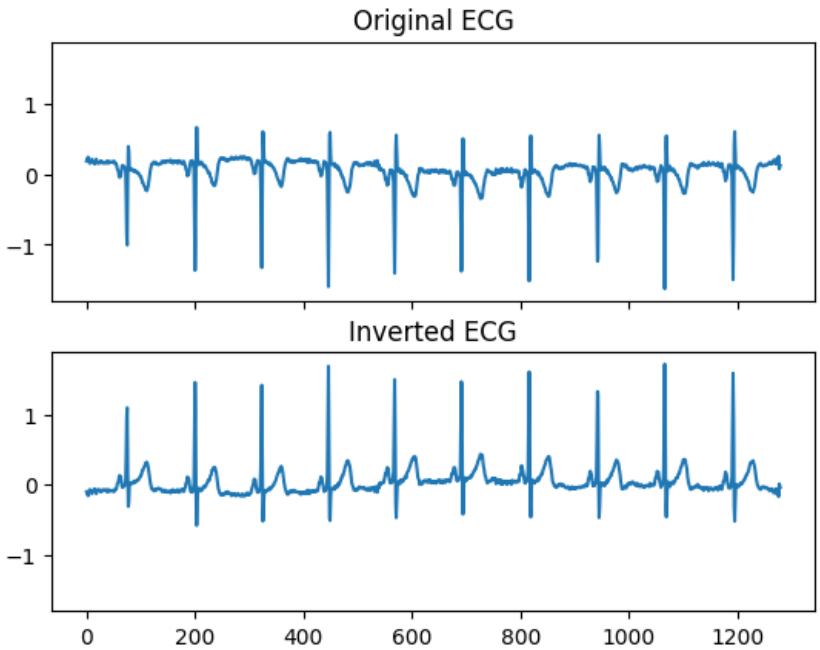


Figure 6.6: Inverted ECG signal

was done from 128 Hz to 360 Hz. Note the changes along the X-axis. It is clear that the waveform of a QRS complex retains its original shape, and the P-wave and T-wave also show strong similarities to their pre-resampled counterparts. This shows that the ECG signal's inherent characteristics were preserved during the resampling process. Furthermore, it can be seen that the overall waveform is largely unaltered after resampling when looking at a 10-second segment. This is an important outcome because it implies that the ECG signal's essential characteristics or information have not been lost during the resampling process. The resampled ECG signal maintains a clear and accurate representation of the original waveform upon closer inspection at various zoom levels. This serves as additional evidence of how well the resampling procedure preserves the integrity of the ECG signal.

Manual ECG cleaning

In this research, two ECG signal cleaning strategies were tested: automatic cleaning with the Neurokit2 toolkit and manual cleaning. The comparison of these two techniques aims to highlight the benefits and potential drawbacks of each method. This section will concentrate on the manual cleaning of the ECG signal. When processing biomedical signals, such as ECG data, it is frequently necessary to focus on a specific range of frequencies while minimising the influence of extraneous frequencies that may represent noise or other non-informative signals.

Butterworth filter A procedure known as bandpass filtering was used to meet this requirement. The concept of "band-pass" allows a specific frequency band or range to pass through the filter while significantly reducing frequencies outside of this range. This filter was built using a Butterworth filter [94], which is known for having the most flat frequency response within the passband. This property ensures that no frequencies within the area of interest are disproportionately amplified or attenuated. The filter order is a parameter that affects the filter's roll-off characteristics and determines its complexity. The

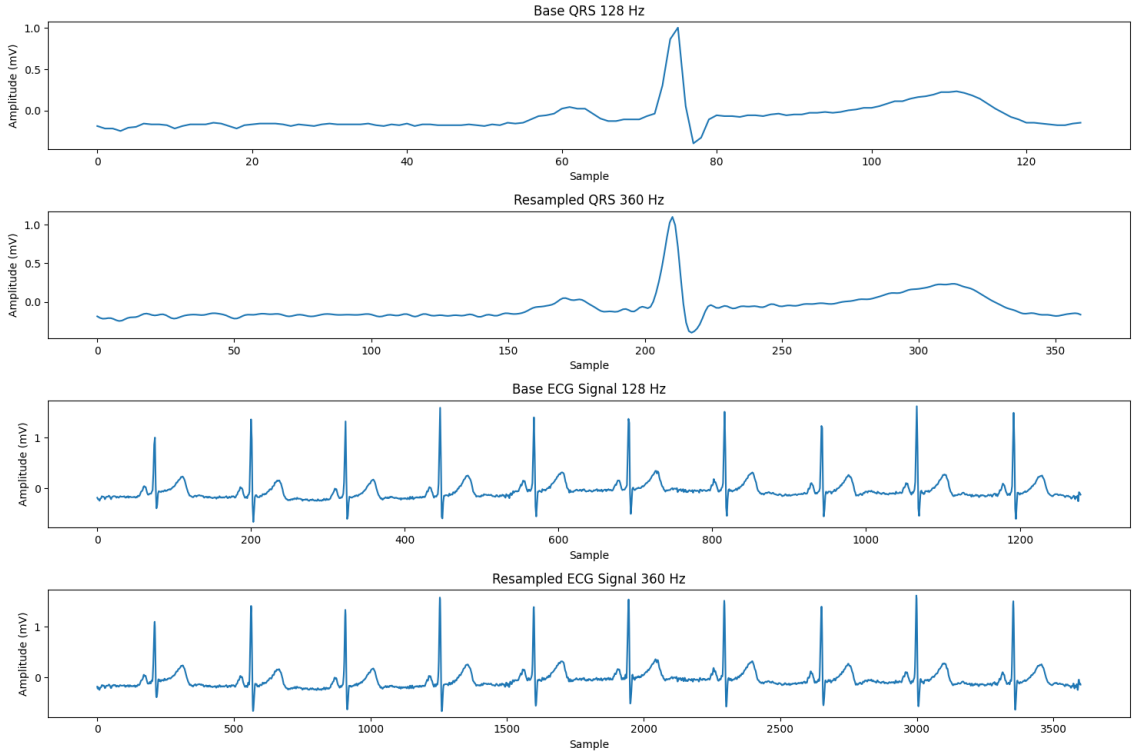


Figure 6.7: Resampled ECG signal, from 128 Hz to 360 Hz

Nyquist frequency, defined as half of the data sampling rate, holds significant importance in the design of digital filters. The maximum frequency that a specific sampling rate can accurately represent is known as the Nyquist frequency. The bandpass filter's lower and upper cutoff frequencies were determined relative to the Nyquist frequency, effectively normalising these cutoffs to a range between 0 and 1. After constructing the filter, it was applied to the data in both forward and reverse passes to ensure zero phase distortion. This method ensures that the filter does not cause any temporal shifts in the signal, preserving the signal's original timing. This process produces a modified version of the original data in which frequencies outside the designated band are suppressed, resulting in a signal that primarily reflects the frequencies of interest. The result of this bandpass filtering can be observed in Figure 6.8.

It can be seen that after applying the Butterworth filter to the ECG data, the filtered signal has significantly less high-frequency noise than the unfiltered signal did. The Butterworth filter preserves the frequencies inside the passband while attenuating frequencies outside the specified passband, in this case, from 0.5 Hz to 50 Hz.

This frequency range has been chosen specifically to capture the key components of the ECG signal, which typically fall within it. For instance, the frequency range of the QRS complex, one of the most significant elements of the ECG waveform, is around 10 Hz to 50 Hz. The filtered ECG signals appear smoother and less distorted by high-frequency noise when compared to the original signal. This is especially apparent in the case of the QRS complexes, which in the filtered signals appear sharper and more pronounced. Additionally, the filtering process reduces baseline wander, a typical type of low-frequency noise in ECG signals. The relatively flat baseline seen in the filtered signals when compared to the original signals serves as confirmation of this. It is also important to note that the filtering process maintains the overall shape of the signal, including important components

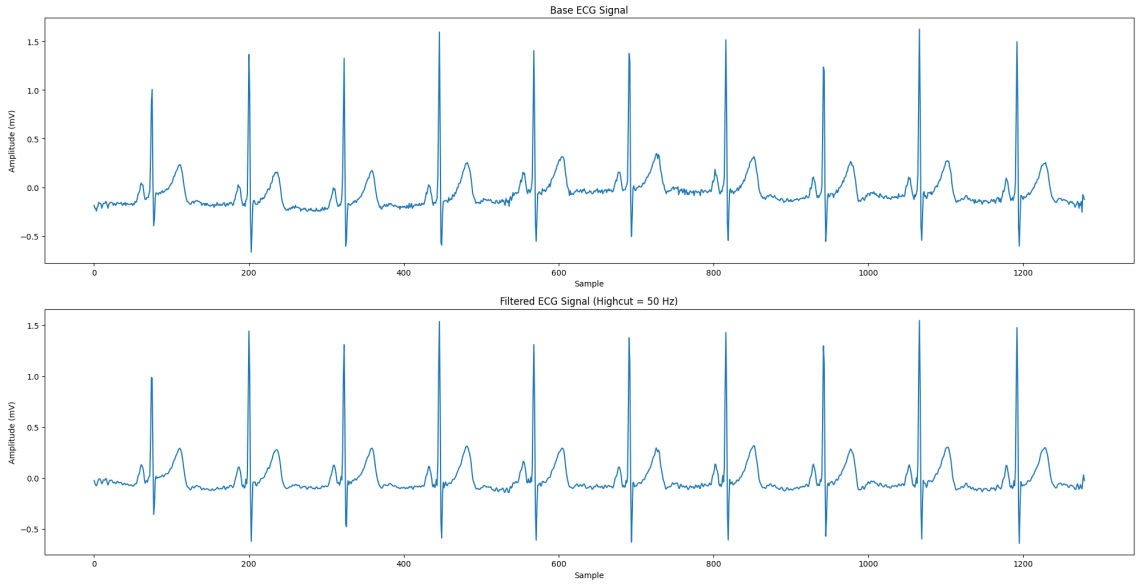


Figure 6.8: Butterworth filter

like the P-wave, QRS complex, and T-wave. The zero-phase property of the Butterworth filter used in this study, which ensures that the filter does not cause any temporal shifts in the signal, is primarily responsible for this preservation.

Wavelet denoising The decomposition of the signal into wavelet coefficients is the first step in the wavelet denoising process [95]. This is accomplished through wavelet decomposition, which divides the signal into different frequency components. Each decomposition level provides a more detailed frequency representation of the signal. The specifics of this breakdown are determined by the wavelet function ('db5' in this case) and the decomposition level (1 in this case).

The wavelet transform uses the 'db5' wavelet, also known as the Daubechies 5 wavelet. The number 5 refers to the number of vanishing moments, which means that this wavelet can represent polynomials up to degree 4. This wavelet is well-known for its ability to capture higher-frequency details. The process employs decomposition level 1. The number of times the wavelet decomposition process is applied, each time breaking down the signal into a lower-resolution approximation and a detailed component, is referred to as the decomposition level. A decomposition level of one indicates that the wavelet transform was applied to the signal only once, resulting in a coarse approximation and one level of detailed coefficients. Higher decomposition levels result in a finer, more detailed frequency representation of the signal, but they are also more computationally demanding.

Following the calculation of the wavelet coefficients, a threshold value is calculated to distinguish the signal from the noise. The Median Absolute Deviation (MAD), a reliable measure of variability, is used to calculate this threshold. At the specified decomposition level, the threshold is set as a multiple of the MAD of the wavelet coefficients. The MAD makes the threshold more resistant to outliers, which are common in noisy data. The wavelet coefficients are then subjected to the threshold. All coefficients less than the threshold are assumed to be noise and are thus set to zero, while those greater than the threshold are assumed to be signals and are left alone. This is referred to as soft thresholding. As a result, a new set of wavelet coefficients is generated, with the ones caused by noise set to zero.

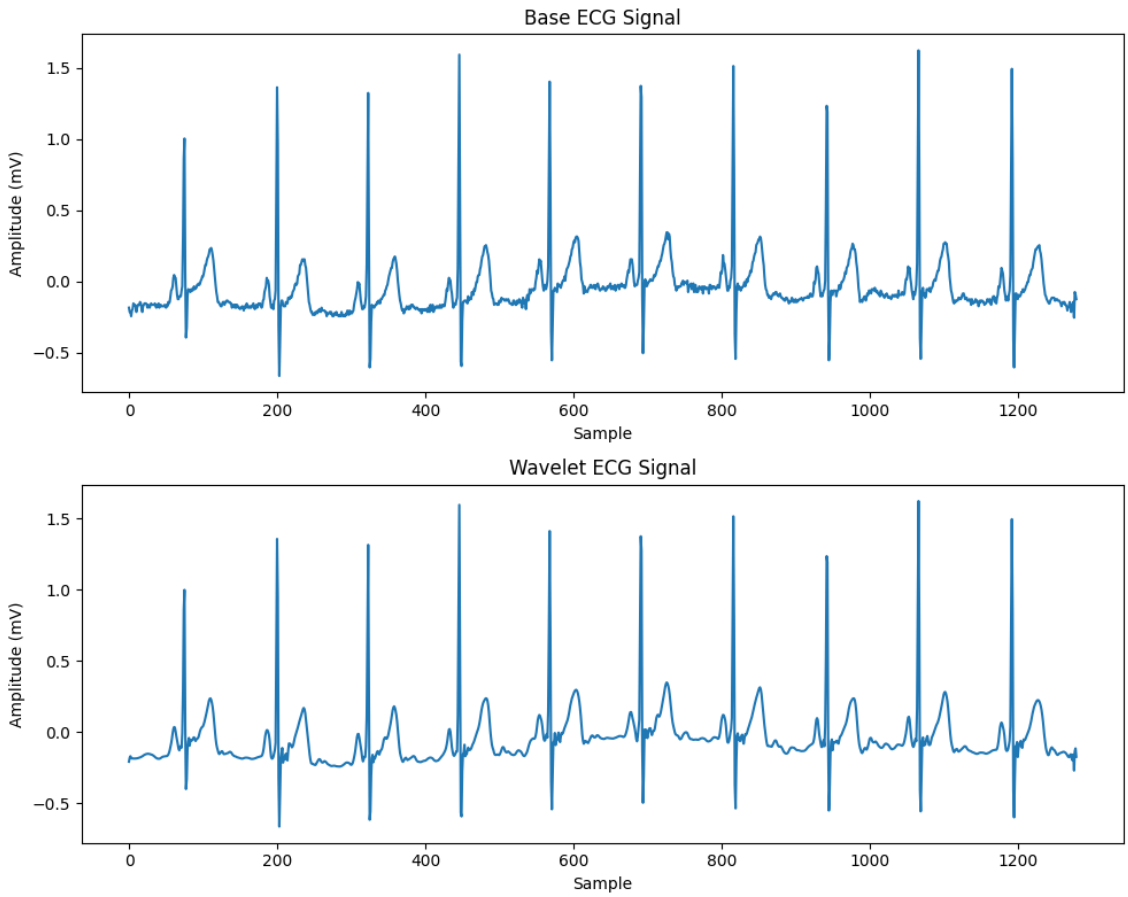


Figure 6.9: Wavelet filter

Finally, using wavelet reconstruction, the original signal is reconstructed from the thresholded coefficients. The reconstructed signal retains the original frequency characteristics represented by the wavelet coefficients but without the noise components. As a result, the original signal has been denoised. This entire procedure is known as wavelet denoising, which is a powerful method for removing noise from signals while retaining their original properties. The result of this technique can be seen in Figure 6.9.

Complete cleaning The ECG signal processing technique presented here uses two key steps to cleanse the raw ECG data. To begin, a Butterworth filter is used to remove any unwanted noise from the signal. Then, the frequency content of the filtered signal was analysed using a wavelet transform. The combined effect of these two cleaning techniques, as shown in Figure 6.10, is a significantly more interpretable ECG signal. This two-step cleaning process reduces noise and improves frequency resolution, allowing for a better understanding of the ECG signal.

Automated ECG cleaning

The automatic cleaning of the ECG signals was done using the Neurokit2 package, which includes a simple method for this task known as `ecg_clean`. To refine the ECG data, this method performs two primary operations.

To begin, a high-pass Butterworth filter with a cutoff frequency of 0.5 Hz and an order

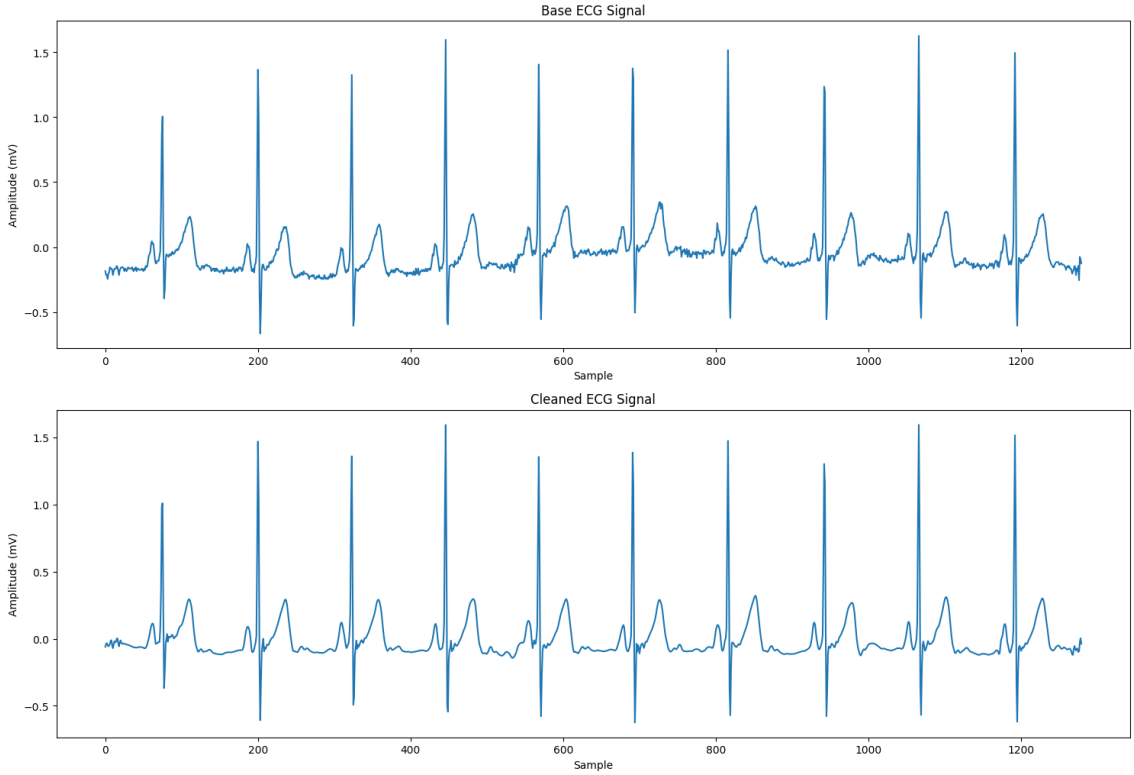


Figure 6.10: Complete manual ECG cleaning

of 5 is used, effectively eliminating any frequency components below this cutoff. This is especially useful for removing any low-frequency drift in the signal. Second, it filters powerlines to eliminate noise introduced by electrical equipment operating at powerline frequencies. In most countries, this frequency is 50 Hz, and it is also the default setting in the `ecg_clean` method. An additional comparison was made between the manually cleaned signals and those processed with Neurokit2's `ecg_clean` method. Figure 6.11 shows the comparison results. This contrast allows for a direct evaluation of the efficacy of both manual and automatic cleaning methods.

ECG rescaling

Data can be transformed using the ECG rescaling technique, which involves linearly scaling the data to a new range with a new minimum and maximum value. For algorithms that are sensitive to the absolute or relative scale of the input features, this is also referred to as min-max normalisation.

Finding the current minimum and maximum input data values initiates the procedure. The scaling calculation is built on top of these values. After that, the desired new range (from minimum to maximum) is mapped onto the current data range (from minimum to maximum). To achieve this mapping, the current minimum is subtracted from each data point, causing the data to shift until the current minimum lines up with zero. The result is then multiplied by the new range to current range ratio, which scales the data so that the current maximum matches the new maximum. Finally, the new minimum is applied to each data point, causing the data to shift so that the zero now corresponds to the new minimum. The resulting dataset retains the original data's relative distribution and relationships but is scaled to fit within the desired new range. After that, the rescaled data

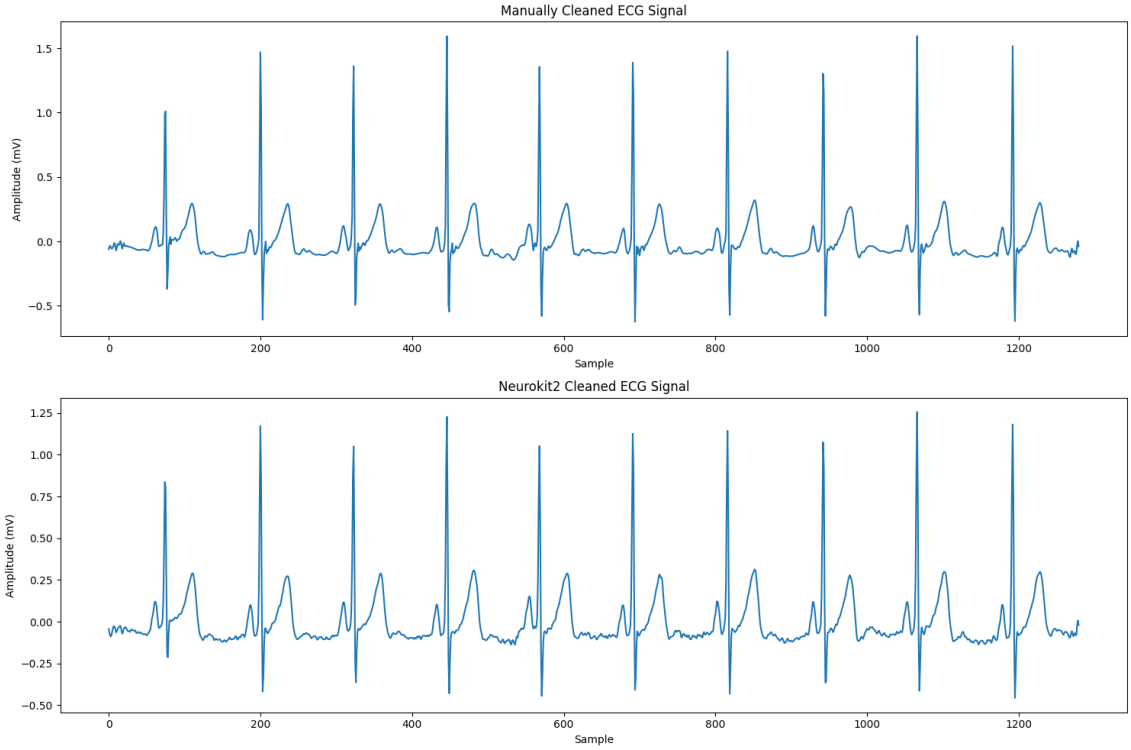


Figure 6.11: Manual vs Neurokit2 cleaning comparaison

is returned for further use or analysis.

We then achieved an ECG comprised between 0 and 1 as we can observe in Figure 6.12.

Excellent segments

The Neurokit2 Python package contains a comprehensive set of tools for performing physiological signal processing, including a feature for evaluating the quality of electrocardiogram signals. The Neurokit2 quality assessment procedure is based on the methodology proposed by Zhao et al. [96], which provides a practical and effective approach for evaluating the quality of ECG segments based on an examination of the distribution of QRS complexes in the ECG signal. Zhao's method categorises ECG quality into three levels: Excellent, Barely Acceptable, and Unacceptable. Only the ECG segments classified as Excellent were retained for this study. This rigorous selection criteria was applied to ensure that the data for subsequent analyses was of the highest possible quality. The goal was to reduce the possibility of errors or inaccuracies caused by the use of low-quality ECG data.

A total of 20.3% of segments from the AFDB were discarded, compared to 7.2% from the MIT-BIH and 3.6% from the NSRDB. The quantities of discarded segments can be attributed to patients that may not remain perfectly still over extended periods, leading to interference in the data.

Raw Data

The same set of metrics and segments were derived directly from the unprocessed data to evaluate the efficacy of the pre-processing steps applied to the raw data. This comparative analysis allowed us to assess the impact of the cleaning procedures on the data's quality and characteristics. The goal of this approach was to ensure that the data cleaning pro-

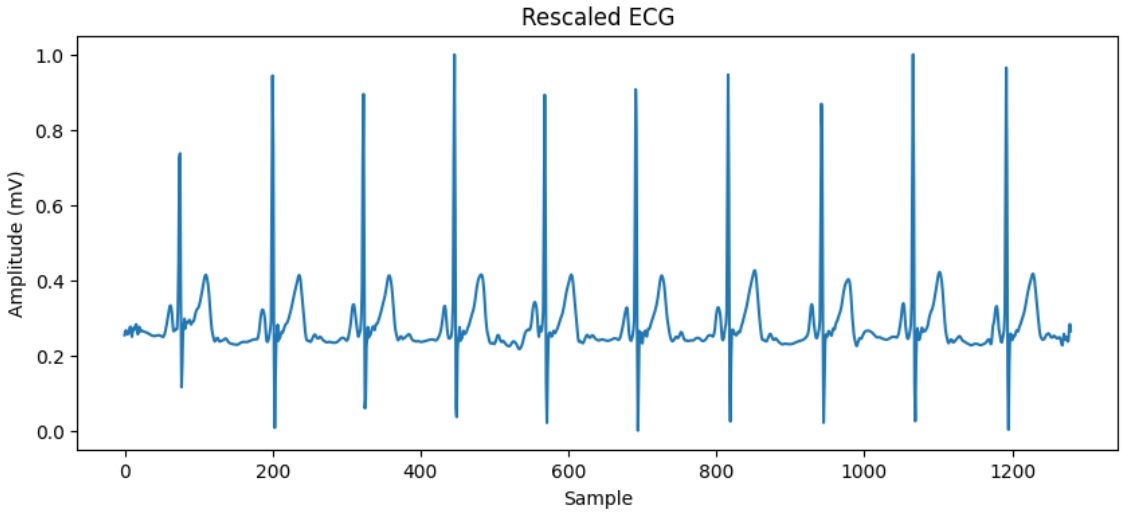


Figure 6.12: Rescaled ECG

cess improved data quality without distorting or losing important information. The raw, unaltered data was also used in all of the other methodologies employed in this study.

6.2.4 Beats methodology preprocessing

The way the beats are recovered differs slightly from the 10-second method. This time, we will look directly in the `.atr` or `.qrs` file for windows containing only QRS normals and containing the number of beats of interest. The goal is to find windows of normal sinus rhythm, specifically windows with 100 consecutive normal beats. The procedure starts at the beginning of the annotated ECG data and looks for a 'N' label, which stands for normal beat. This is repeated in a loop until there are enough beats in the data for a complete window. After identifying a normal beat, the labels of the following beats are examined. If all of the beats are normal, a window of normal sinus rhythm is identified. The window's boundaries are then determined. The index at the start is the current beat, and the index at the end is the beat that comes just after the last beat in the window. After that, the normal sinus rhythm window is extracted from the cleaned ECG signal, and the cleaning can begin. The pseudocode can be found in Algorithm 2.

This method's data cleaning procedures are the same as those outlined for the 10-second methodology. Further elaboration on the cleaning process is not required in this context due to the consistency of the approach. In this case, the primary focus remains on the identification and analysis of the specified rhythm windows, rather than the preprocessing steps described earlier. An example of the result of this preprocessing can be seen in Figure 6.13.

6.2.5 RR Intervals methodology preprocessing

We will go through the `.qrs` or `.atr` files, similarly to the Beats approach for this methogology. These annotations are important since they show where the QRS complexes are located. To accurately separate the ECG signal into individual beats or windows, this information is required.

The current analysis identifies windows of normal sinus rhythm. Iterating through the QRS annotations is the first step in this process. It is determined whether each QRS com-

Algorithm 2 Beat Window Extraction

```
Load QRS annotations
Initialize i = 0
while i < len(QRS annotations) − n_beats do
    if QRS annotation symbol[i] is 'N' then
        Extract next symbols from i + 1 to i + n_beats
        if all symbols are 'N' then
            Start is QRS annotation sample[i]
            End is QRS annotation sample[i + n_beats]
            Extract the window
        end if
    end if
    Increment i
end while
```

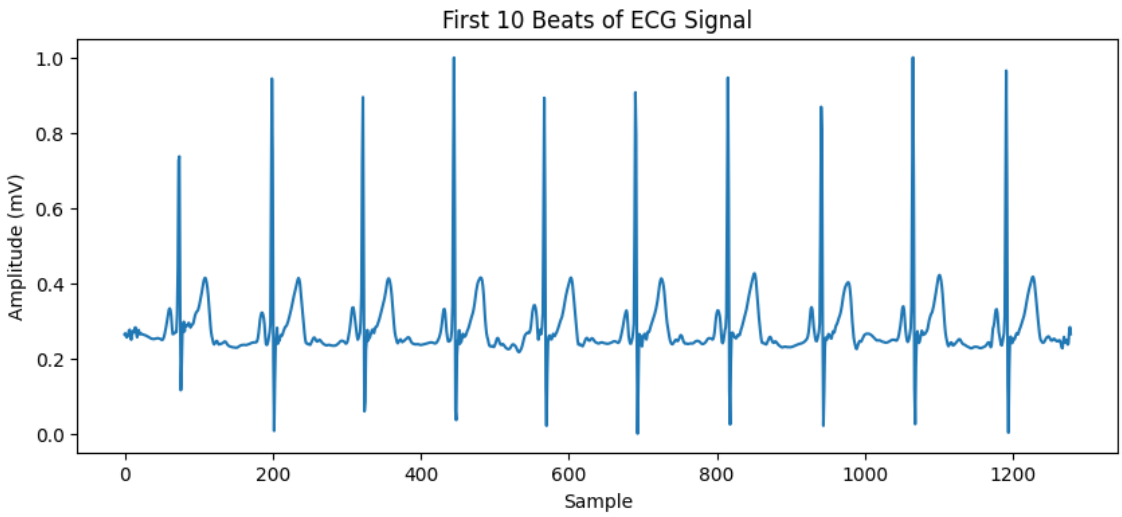


Figure 6.13: First 10 beats of NSRDB ECG Signal

plex is designated as a normal sinus rhythm ('N'). If so, the procedure continues by defining a window that begins at the current QRS complex and includes a predetermined number of subsequent normal sinus rhythms (referred to as "RRs"). Depending on the requirements of the analysis, this number can be modified. In this research, a window of 100 beats has been chosen. The start and end indices of this window in the ECG signal are found once a window containing the desired number of normal sinus rhythms has been found. These indices represent the first and last QRS complexes in the window, respectively. Then, all QRS complexes that are contained within the window are identified by their indices. The RR intervals within the window are calculated from these R-peak indices. The RR interval, which is an important indicator of heart rate variability, represents the period between consecutive R-peaks. The difference between two consecutive R-peak indices is used to calculate these intervals. The RR intervals are then resampled to the ECG recording frequency to convert them to millisecond units. This adjustment is required because the original RR intervals are expressed in terms of samples, and the precise amount of time that each sample represents depends on the ECG recording's sampling frequency. The RR intervals are converted into a common time unit through resampling, enabling a more

Algorithm 3 RR Interval Metrics Extraction

```
Load QRS annotations
Initialize the index i to 0
while i < the length of QRS annotations - RRs AND the sample at i in QRS annotations < the length of the ECG signal do
    if the symbol at i in QRS annotations is 'N' then
        Calculate the start and end of the window
        Get the symbols for the next RRs after i
        if all symbols in the window are 'N' then
            Locate the indices of R-peaks that fall within the window
            Calculate the RR intervals within the window
            Resample to the frequency to convert intervals to milliseconds
        end if
    end if
    Increment i
end while
```

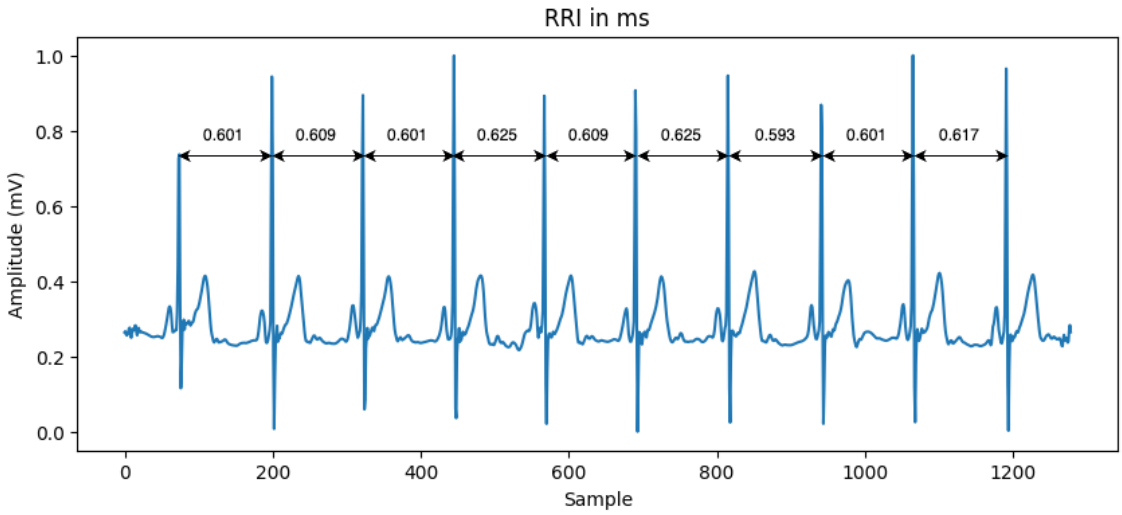


Figure 6.14: 10 first RR Intervals

insightful interpretation of their values.

Each QRS complex in the ECG recording goes through this entire process once, resulting in a set of RR interval windows. The pseudocode can be found in Algorithm 3 and the result of the preprocessing is represented in Figure 6.14.

6.2.6 Peaks methodology preprocessing

The reference work that Castro and his colleagues wrote used a number of intricate metrics, including the PR segment and the surface area of the P-wave, among others. While retrieving these measures can be complicated and does not always ensure the best results or model performance, they do offer insightful information. The current study uses metrics that are both meaningful and easier to retrieve. The metrics P-peak, Q-peak, R-peak, S-peak, and T-peak have been determined to be appropriate for this research. Despite being straightforward to extract, these characteristics have been found to be important arrhythmia indicators [97]. This research tries to produce an analysis that is more effec-

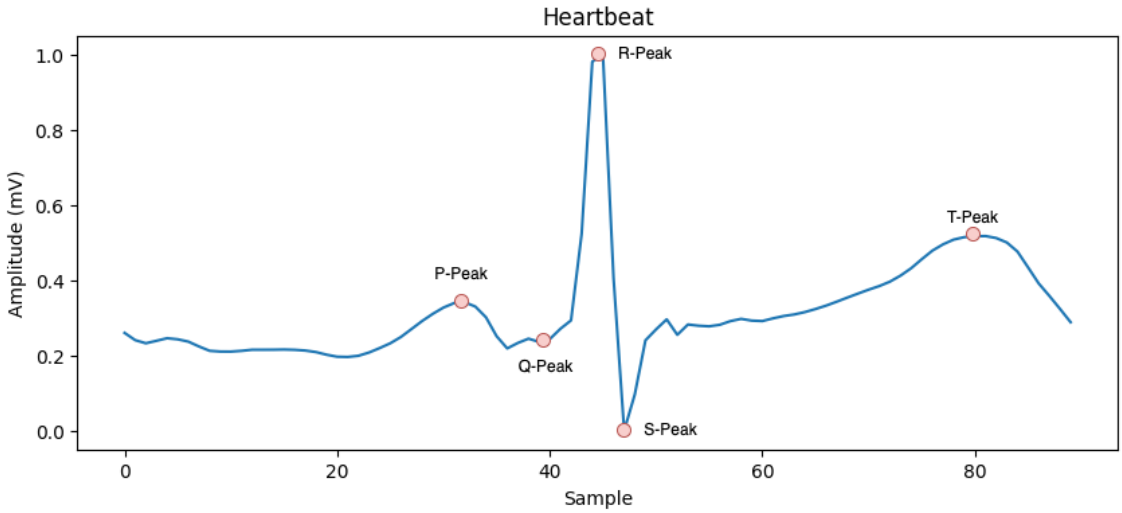


Figure 6.15: Metrics Extraction

tive and simple while maintaining the reliability of the results by focusing on these more approachable attributes. The metrics that have been extracted are clearly displayed in Figure 6.15.

The process begins by loading QRS annotations, which provide important information about the locations of the QRS complexes, indicating the beats of the heart. After the annotations are loaded, the algorithm proceeds to locate segments of normal sinus rhythm within the ECG signal. These are defined as continuous sequences of R-peaks that are labelled as normal sinus rhythm. To find these segments, the algorithm iterates through the QRS annotations, checking the labels of consecutive R-peaks. When it finds a sequence of R-peaks all labelled as normal sinus rhythm, it marks the start and end of this sequence in the ECG signal. Following the identification of a normal sinus rhythm segment, the corresponding segment of the ECG signal is extracted. This segment contains the raw ECG data for the duration of the normal sinus rhythm segment. This segment is then processed to extract the relevant features of the ECG signal. The NeuroKit2 library facilitates this process by including functions designed specifically for ECG signal processing. This process produces a cleaned ECG signal as well as various features of interest within the ECG signal, such as the locations of P-peaks, Q-peaks, R-peaks, S-peaks, and T-peaks. The algorithm extracts the values of the peaks in the ECG signal after processing the segment. These peak values represent the amplitude of the ECG signal at different points in the cardiac cycle and are important features for heart rhythm analysis. To store these peak values, an empty 2-dimensional array is initialised. This array has a number of rows equal to the number of R-peaks in the segment and five columns, one for each type of peak. Next, the algorithm iterates through each type of peak. For each type, it locates the indices where these peaks occur in the processed ECG signal. It then uses these indices to extract the corresponding peak values from the cleaned ECG signal. These values are then stored in the corresponding column of the peak values array. If no peaks of a certain type are found in the segment, the corresponding column in the array is filled with zeros.

Finally, the peak values are rescaled to a predefined range. This is done to normalise the values and ensure that they are on a comparable scale. This step is essential for subsequent analyses, as it allows for a more meaningful comparison of values across different ECG signals and patients. After rescaling, the array of peak values is ready to be used for

Algorithm 4 Extract and Process Peaks

```
Load QRS annotations
i = 0
while i < the length of QRS annotations - RRs do
    if QRS annotation symbol at i is 'N' then
        next_symbols = QRS annotation symbols from i + 1 to i+RRs
        if all next_symbols are 'N' then
            window = cleaned_ecg from sample[i] to sample[i + RRs]
            Initialize a 2D array all_peak_values
            for each peak_type in ['P', 'Q', 'R', 'S', 'T'] do
                peak_indices = indices where processed_ecg at peak_type is 1
                peak_values = values of cleaned_ecg at peak_indices
                fill corresponding column in all_peak_values with peak_values
            end for
            rescale all_peak_values between min_mV and max_mV
        end if
    end if
    Increment i
end while
```

further analysis. The pseudocode can be found in Algorithm 4.

6.2.7 Overlapping strategy

In the initial phase of data processing, ECG segments were extracted in a continuous manner. This implies that each segment was distinct, with no overlap of data between consecutive segments. While this approach is straightforward to implement and understand, it poses a significant limitation. It restricts the capacity to generate a more extensive dataset for the training and validation of the model, a goal that could be achieved via a technique known as data overlapping.

Data overlapping entails the generation of new segments by adjusting the start and end points of each segment by a few time units. Consequently, segments partially share their data with adjacent segments. This technique is beneficial as it facilitates the creation of a larger dataset from a relatively smaller initial set of data. By leveraging data overlap, the dataset can be expanded, thereby providing the model with a wider spectrum of data to learn from. This could potentially enhance the model's ability to generalise and perform well on unseen data. In the context of this study, where ECG data is limited, this technique proves to be particularly beneficial. A simple example can be observed in Figure 6.16. Unbalanced datasets are a common problem in machine learning, particularly in classification tasks. When one class outnumbers the other, the learning algorithm may become biassed towards the majority class, resulting in overfitting. Overfitting occurs when a model has learned the training data so well that it performs poorly on unseen data.

The overlapping strategy was especially effective in addressing the issue of class imbalance. We were able to generate a more balanced dataset by creating overlaps for the minority class based on the number of samples in the majority class. This not only reduced the risk of overfitting but also enhanced the model's ability to generalise and perform well on previously unseen data.

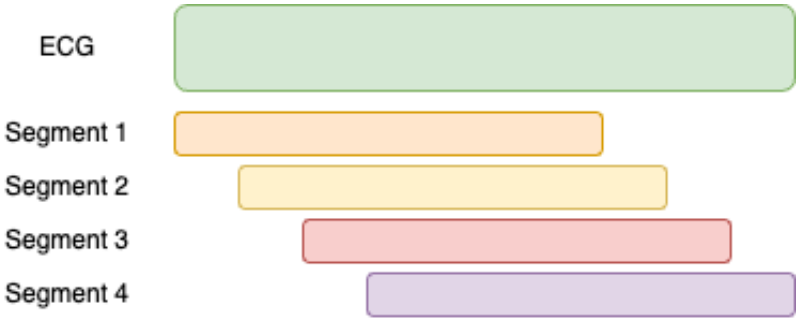


Figure 6.16: Simple overlapping

6.3 Models

Machine learning employs a wide variety of models, each with its own benefits and drawbacks. Some research has also looked into CNNs, RNNs, and their variants, or even hybrid models that mix the architectural complexities of CNNs and RNNs. Given the abundance of choices, empirical comparison is the most effective way to select the optimal model for a given job. For that purpose, this research examines four separate models, each of which represents a different class of machine learning models. A CNN, a LSTM network (a RNN variant), a Residual Network (ResNet), and a Random Forest classifier were chosen.

CNNs and LSTMs represent the class of neural networks, with CNNs excelling at handling spatial data such as images and LSTMs excelling at dealing with temporal sequences. ResNet, a deep learning model, is well-known for its novel approach to addressing the vanishing gradient problem in very deep network training. The Random Forest classifier, a popular ensemble learning method, on the other hand, takes a non-neural approach to classification tasks. This study aims to compare the performance of these various models for the specific task at hand by juxtaposing them. A comparison study like this provides insights into the complexities of powerful models versus simpler ones as well as a comprehensive understanding of their effectiveness in the given context.

6.3.1 Convolutional Neural Network

In the development of this model, we were inspired by the research paper "An Accurate CNN Architecture For Atrial Fibrillation Detection Using Neural Architecture Search" authored by Fayyazifar et al. [98]. The proposed model in their study served as the foundational architecture for the model in this research.

The model can be seen in Figure 6.17.

Its structure is as follows:

1. Conv1D Layer: The model's first layer is a 1-dimensional convolutional layer. This layer processes the input data through 64 filters of size 5. The activation function ReLU is used, which aids the model in learning complex patterns and avoids the vanishing gradient problem. The input shape is (3600, 1), corresponding to 3600 time points for each ECG signal (this is true only for the 10-second methodology).

2. MaxPooling 1D Layer: This layer reduces data dimensionality by taking the maximum value over a 2-dimensional window. By providing a form of translation invariance, this helps to reduce overfitting and computational cost.

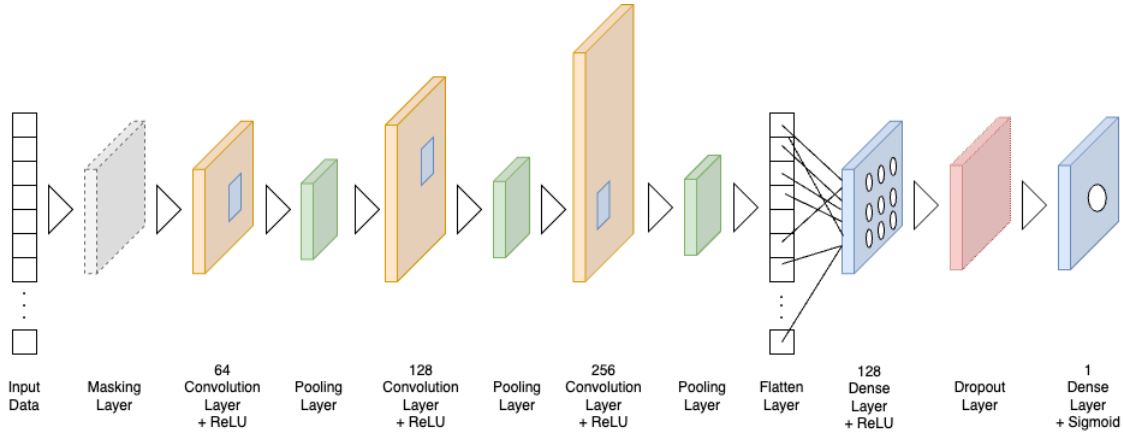


Figure 6.17: CNN Model

3. Conv1D Layer: The second convolutional layer adds 128 filters of size 5 to the previous layer's output. The ReLU activation function is used once more.

4. MaxPooling 1D Layer: A second max pooling layer with the same window size of 2 is added.

5. Conv1D Layer: The third convolutional layer employs 256 5-size filters. The ReLU activation function is used yet again.

6. MaxPooling 1D Layer: A final max pooling layer with a window size of 2 is added.

7. Flatten Layer: This layer converts the previous layer's 2D output to a 1D vector. This is required because the dense layer that follows expects input in this format.

8. Dense Layer: This layer is fully connected and contains 128 neurons. The activation function ReLU is used.

9. Dropout Layer: To prevent overfitting, this layer randomly sets a fraction (0.5, or 50%) of the input units to 0 at each update during training.

10. Dense Layer: The model's final layer is another fully connected layer with a single neuron. The sigmoid activation function is used, which limits the output between 0 and 1, making it appropriate for binary classification.

Finally, the model is built using the Adam optimizer, a binary cross-entropy loss function (suitable for binary classification), and accuracy as the evaluation metric. The Adam optimizer is an adaptive learning rate optimisation algorithm that has been demonstrated to perform well in practise and on a wide range of problems. The loss function of binary cross-entropy is used in binary classification tasks.

6.3.2 LSTM Model

The LSTM model was inspired by "A stacked LSTM for atrial fibrillation prediction based on multivariate ECGs" by Sun et al. [99]. The proposed model served as inspiration for the model in this research. The model of this research can be found in Figure 6.18.

Here's a breakdown of each layer and its function:

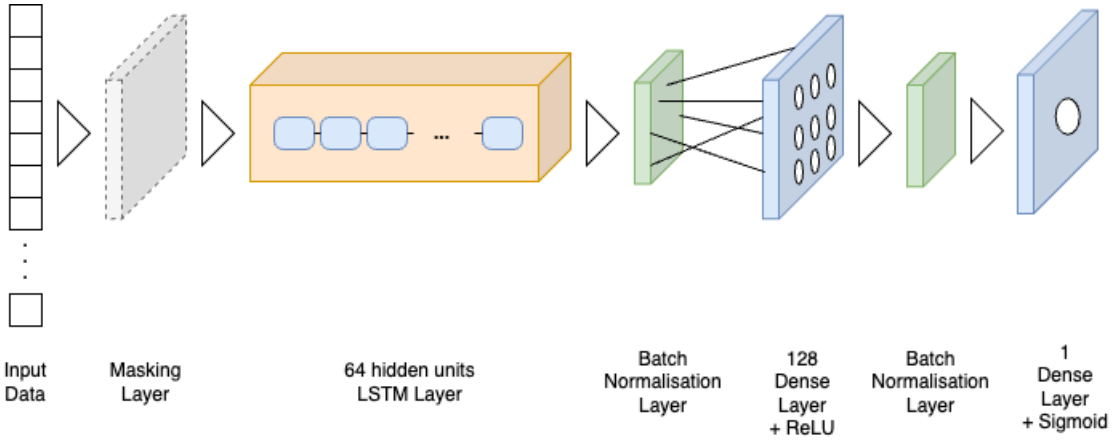


Figure 6.18: LSTM Model

1. Masking Layer: The Masking layer is used to ignore specific inputs based on the value of a mask. Any input with a value of -1 will be ignored in this case. This is useful when dealing with variable-length inputs.

2. LSTM Layer: A Long Short-Term Memory layer. This LSTM layer consists of 64 hidden units.

3. Batch Normalisation Layer: At each batch, this layer normalises the activations of the previous layer by applying a transformation that keeps the mean activation close to 0 and the activation standard deviation close to 1. It helps in the acceleration of the training process and the reduction of overfitting.

4. Dense Layer: This is a fully connected layer, with each input and output node connected. It has 128 neurons and is activated by the ReLU function.

5. Batch Normalisation Layer: Adds another batch normalisation layer to the previous dense layer's activations.

6. Dense Layer: This is the network's output layer. Because this is a binary classification problem, it only has one neuron and employs the sigmoid activation function, which squashes the output values between 0 and 1, providing a probabilistic interpretation.

In alignment with the previous model, the chosen loss function for this model is binary cross-entropy. The optimizer employed is Adam, which is a popular choice due to its efficiency and low memory requirement. For evaluating the model's performance, we utilise accuracy as the primary metric. This choice allows us to directly understand the proportion of correct predictions made by the model.

6.3.3 ResNet Model

The ResNet model used in this context is a modified version of the one featured in the research article by Attia et al. The article, titled "An artificial intelligence-enabled ECG algorithm for the identification of patients with atrial fibrillation during sinus rhythm: a retrospective analysis of outcome prediction" [71], presents the original model. The modifications were made to better suit the specific requirements of our project. The structure is shown in Figure 6.19.

The structure of the model is as follows:

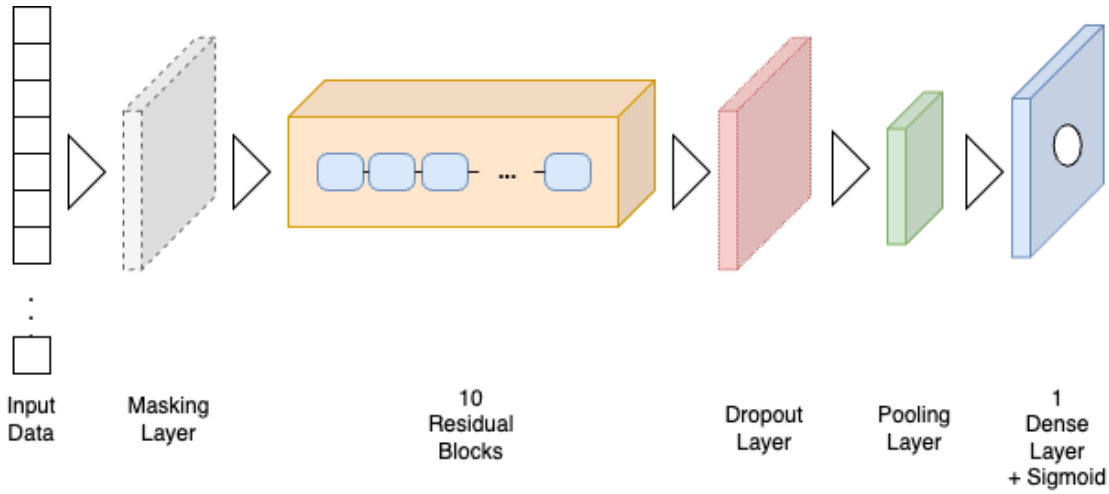


Figure 6.19: ResNet Model

1. Masking Layer: As for LSTM, this layer serves to selectively disregard certain inputs according to the value of a mask. Specifically, inputs bearing a value of -1 are omitted. This feature is essential when handling inputs of varying lengths.

2. 10 Residual Blocks: The essence of the model lies in its residual blocks, which is illustrated in Figure 6.20. The structure is outlined as follows:

   (a) The block initiates a shortcut connection that bypasses the primary layers it contains. This shortcut uses a 1D convolution layer with a kernel size of 1 and a step size of 1. While preserving the spatial dimensions, it changes the number of feature maps to 64.

   (b) The block's primary pathway starts with a batch normalisation layer. This layer standardises the inputs, ensuring they have a zero mean and a variance of one, which enhances the learning process's stability and speed.

   (c) Following this, a ReLU activation function is applied to introduce non-linearity into the model.

   (d) Subsequently, the block contains convolutional layers equipped with 64 filters. The layer employs a kernel size of 3 and incorporates padding to preserve the spatial dimensions.

   (e) Following this, the architecture incorporates another Batch Normalisation Layer, succeeded by a ReLU activation function and a subsequent Convolutional Layer. The block concludes with a final Batch Normalisation Layer.

   (f) The outputs of the shortcut and the main path are then added together in the Add layer. This is the key step in a residual block, as it allows the model to learn residual functions with reference to the layer inputs, which can help mitigate the vanishing gradient problem in deep networks.

   (g) Finally, another ReLU activation is applied, followed by a Pooling Layer.

3. Dropout Layer: A Dropout Layer of 0.5 that helps prevent overfitting by randomly setting half of the input units to 0 during training.

4. Pooling Layer: It reduces the spatial dimensions of the feature maps, summarising the information from the Residual Blocks while retaining multiple values per feature map.

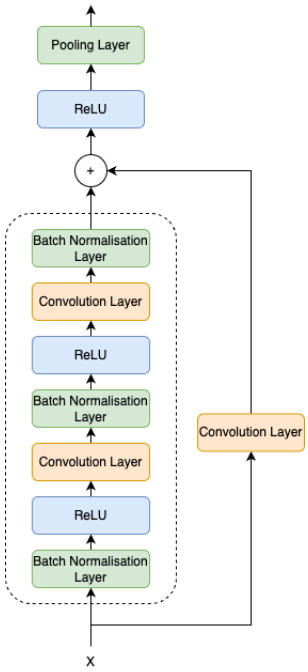


Figure 6.20: ResNet Block

5. Denser Layer: Finally, the output layer is a dense layer with a single unit and a sigmoid activation function, which is suitable for binary classification tasks. The output of this layer is a value between 0 and 1, representing the probability of the positive class.

The model is compiled with the Adam optimizer and binary cross-entropy loss function. The accuracy of the predictions is used as the performance metric.

6.3.4 Random Forest

The Random Forest classifier was the last model used in this study. This model was chosen as a benchmark for assessing the performance of the more complex models previously discussed because of its simplicity and efficacy. The Random Forest classifier has a reputation for handling a variety of datasets and providing high accuracy. Its dependability and ease of use make it an excellent option for a comparative analysis in this situation.

The Random Forest algorithm is set up in the implemented model with a particular parameter: the number of estimators. The parameter known as "the number of trees in the forest" or more technically, the number of estimators, is set to 200 in this model. This key parameter significantly influences both the model's performance and its computational demands.

Each estimator in the Random Forest structure is a distinct decision tree that contributes to the final prediction. The algorithm works by building several decision trees during training and then returning the class that represents the mean of the classes for classification or the mean prediction for regression of the individual trees.

The Random Forest algorithm will create 200 different decision trees after selecting 200 estimators. These trees will each individually assess the input data and provide a prediction. The final prediction of the Random Forest model is then calculated by averaging the predictions for regression or classification across all 200 decision trees.

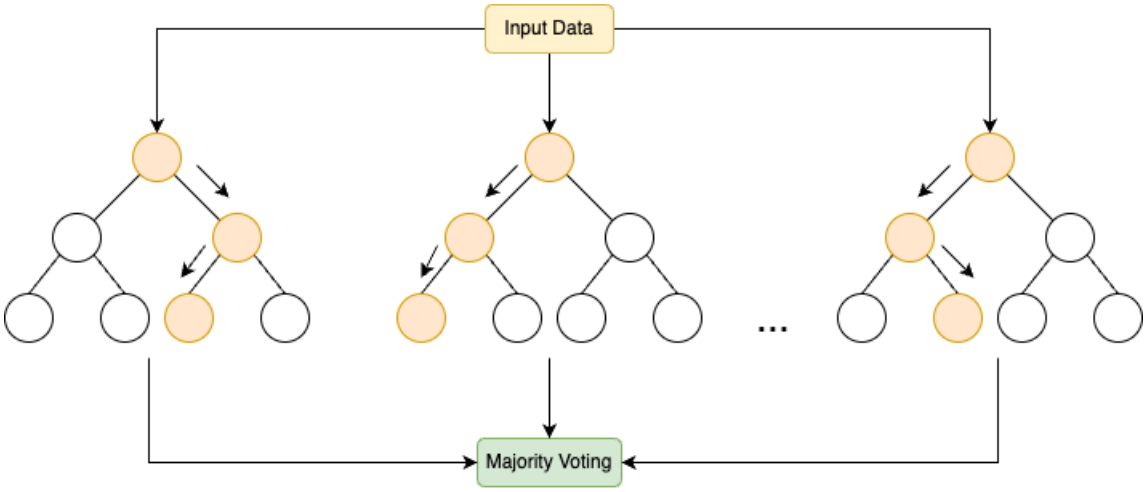


Figure 6.21: RandomForest model

For a visual depiction, see Figure 6.21.

6.4 Data preparation

The data was prepared for input into the respective models after successful extraction, cleaning, and recording of the ECG segments, as well as the subsequent definition of the models.

The first two methodologies utilised 1D data, which is fitting for ECG signals as they fundamentally represent sequences of millivolt amplitudes, making them inherently 1D signals. The data format for these methodologies is typically represented as $Batch \times Length \times Features$, where:

- $Batch$ denotes the number of data samples.
- $Length$ corresponds to the duration or number of time points in each sample.
- $Features$ signifies the number of attributes or measurements at each time point, which, in the case of ECG, is usually 1, representing the amplitude.

The third methodology also employed 1D data, but in this context, the data represented a sequence of numbers indicating the differences between RR intervals, measured in milliseconds. For the last methodology, 1D data was again used, but it was structured differently. The data was organised in the format $RRs \times 5$, where RRs represents the number of beats analysed per segment and 5 denotes the five key points in the ECG waveform: P-Peak, Q-Peak, R-Peak, S-Peak, and T-Peak. Consequently, the overall data format became $Batch \times Length \times 5Features$. Several preparatory steps were undertaken to ensure the data was appropriately formatted and structured for each model.

6.4.1 10-second methodology

Due to the uniformity of the segment lengths, the data preparation process was relatively simple in the context of this methodology. Because each ECG segment was the same length, there was no need for additional preprocessing steps like padding or truncation, which are frequently required when dealing with sequences of varying lengths. Labels were

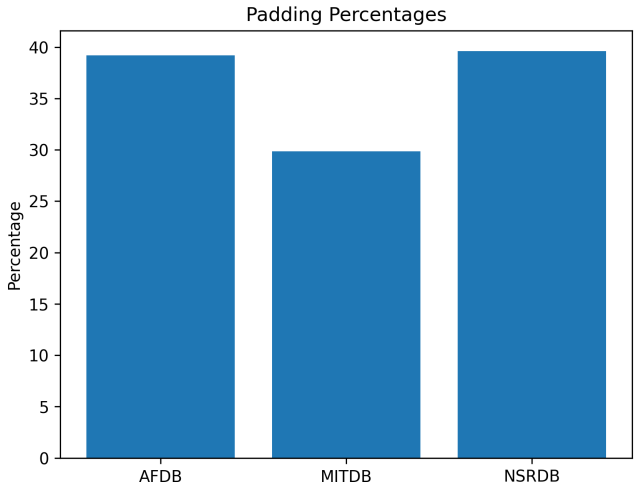


Figure 6.22: Padding percentages per database

assigned to the segments based on their classes. This step is essential in supervised learning because it provides the ground truth on which the model is trained. The labels were binary, indicating whether atrial fibrillation was present or not in the corresponding ECG segment. To ensure the model's robustness and to prevent it from learning any potential order in the data, the segments and labels were shuffled. This random data rearrangement helps to ensure that the model's performance is not influenced by the initial order of the segments and that it can generalise well to unseen data.

6.4.2 Beats methodology

In this study, we examined heartbeat segments of varying lengths, an approach grounded in the understanding that each individual's heart rate differs. However, this posed a problem as our machine learning models require input of a uniform size. Consequently, we needed to standardise the size of our segments.

Two principal strategies are available to address this issue. The first, trimming, involves cutting the segments to match the length of the shortest segment. However, this method invariably leads to data loss. The second strategy, padding, fills the shorter segments with predefined values until they match the length of the longest segment, preserving all the original data. We opted for padding, given its advantage in retaining complete information. The concept of padding is straightforward: identify the longest segment, then extend the remaining segments with a predefined value until they match the length of the longest segment. Consequently, each segment becomes uniform in size, making it suitable for input into the model. To handle these padded inputs, our models incorporate a Masking layer designed to ignore certain values, thus accommodating inputs of various sizes. Given that all our preprocessed values range between 0 and 1, we selected -1 as the padding value. Thus, whenever the model encounters a value of -1, it understands to disregard it. Figure 6.22 illustrates the average padding added to each segment for each database. This visual representation serves to highlight the effectiveness of padding in standardising the length of segments without sacrificing valuable data.

6.4.3 RR methodology

We had segment lengths in this phase of our study as well, similar to the 10-second segment methodology. This uniformity facilitated a simple labelling process for the classes. We implemented a shuffling mechanism to prevent the model from overfitting to the order of the data and to improve its generalizability. This step ensures that the learning process is not influenced by the data's original order, resulting in a more robust model.

6.4.4 Peaks methodology

Consistency in data dimensions was preserved when the Peak method was applied. The data was arranged into matrices with RRs*5 dimensions. In this case, "RRs" stands for the segment of heartbeats that was extracted, and "5" stands for the five feature peaks that were extracted from each heartbeat, namely the P-Peak, Q-Peak, R-Peak, S-Peak, and T-Peak. After extraction, each matrix was given a class label and put through a shuffle operation. An important step in ensuring data order randomness, this shuffling mechanism prevents potential biases during model training. The model inputs have been modified to allow for these matrices. Furthermore, these modifications were only limited to the model input. The model structure and hyperparameters remained consistent, adhering to the guidelines of the earlier methodologies. For comparative analysis across methodologies to be valid, model structure and hyperparameter uniformity must be maintained. It ensures that the observed variations in results are due to differences in the methodologies rather than the underlying models.

6.4.5 Random Under Sampler

Preprocessing, as mentioned in Section 6.2.7, entailed using an overlapping technique to enrich the data provided to the models. However, due to the diverse nature of the datasets and the possibility of inconsistencies during data recording (for example, an electrode displacement might compromise recording quality), class imbalances occasionally emerged. These imbalances persisted despite the quality control measures implemented in accordance with Zhao et al.'s method as well as the precautions taken during the preprocessing phase.

Oversampling, which involves augmenting the data through techniques such as duplicating, adding noise, or making minor data modifications; using Synthetic Minority Oversampling Technique (SMOTE) [100], an oversampling technique widely used; or undersampling were considered potential solutions for addressing this class imbalance. Oversampling and SMOTE may appear to be viable solutions at first glance. However, creating new segments by adding noise or unwanted features may introduce inadvertently misleading patterns, potentially impairing model performance.

Given these considerations, the use of a random undersampler was deemed the best strategy for redressing the class imbalance. A random undersampler is an effective strategy for dealing with class imbalances. This technique works by randomly removing instances from over-represented classes until a balance with the minority class is achieved as we can observe in Figure 6.23. Despite the risk of losing potentially valuable information due to the random elimination of instances, it was determined to be the best method for this study. Random undersampling provides a practical and computationally efficient solution to class imbalance by directly resolving it without the use of synthetic data or distributional assumptions.

Figure 6.23: Random undersampler

6.4.6 Train-Validation-Test ratio

The datasets were split into three separate subsets, labelled training, validation, and test sets, following an extensive data preparation process. A 7:1:2 ratio was used for this partitioning, meaning that 70% of the data was set aside for training, 10% was designated as a validation set to be used during training, and the remaining 20% was designated as the test set.

The training set is the primary dataset from which the models learn, providing examples, features, and target labels or outcomes for the models to use in making decisions. The validation set is important for the model tuning process. The model's performance on the validation set during training provides a measure of how well the model is learning to generalise beyond the training data. It guides the model's hyperparameter adjustment and helps prevent overfitting to the training data. The test set is a reserve of previously unseen data. It is used to assess the model's performance after training, providing an unbiased assessment of the model's ability to generalise to new, previously unseen data.

The models used in this study were trained using ten epochs, which means that the entire dataset was run through the learning algorithm ten times, both forward and backward. The number of epochs, a hyperparameter, determines how many times the learning algorithm will iterate through the entire training dataset. The large amount of data in this study presented significant computational challenges, particularly when implementing k-fold validation. The k-fold validation technique, which divides the data into 'k' subsets and iteratively uses each subset for testing while the remaining subsets are used for training, can be computationally demanding. Given the size of the data and the variety of methodologies and models investigated in this study, k-fold validation was deemed impractical due to the costly computational demands. This approach would have added another level of validation and potentially improved the model evaluation's robustness.

6.5 Summary

Our entire process, encompassing preprocessing, data preparation, and model testing, is summarised in Figure 6.24.

To summarise the whole process:

1. The ECGs from the three datasets are partitioned into training, validation, and test sets to prevent data leakage. Data leakage happens when information from the training set inadvertently influences the test set, leading to biased results. The

datasets are divided as follows: 70% for training, 10% for validation, and 20% for testing.

2. ECGs are subjected to preprocessing. The result varies according to the method used, the nature of the data cleaning, and the duration of the processing. Each unique combination results in distinct data sets. Within each data set, the distribution remains consistent: 70% for training, 10% for testing, and 20% for validation.

3. For each dataset, the data is prepared before being fed into the models. This involves processes such as reshaping, padding, shuffling, and other steps, depending on the methodology employed.

4. The models are then compiled and trained using training and validation data.

5. Finally, the models are evaluated using the test data, and the results are subsequently saved.

6.6 Computer ressources

The computational experiments for the models were conducted on a system equipped with an 11th Generation Intel Core i7-11700 processor operating at 2.5 GHz, 16 GB of RAM, and an RTX3070 GPU. The code used can be found on GitHub: `https://github.com/Alexinator/MasterThesis`.

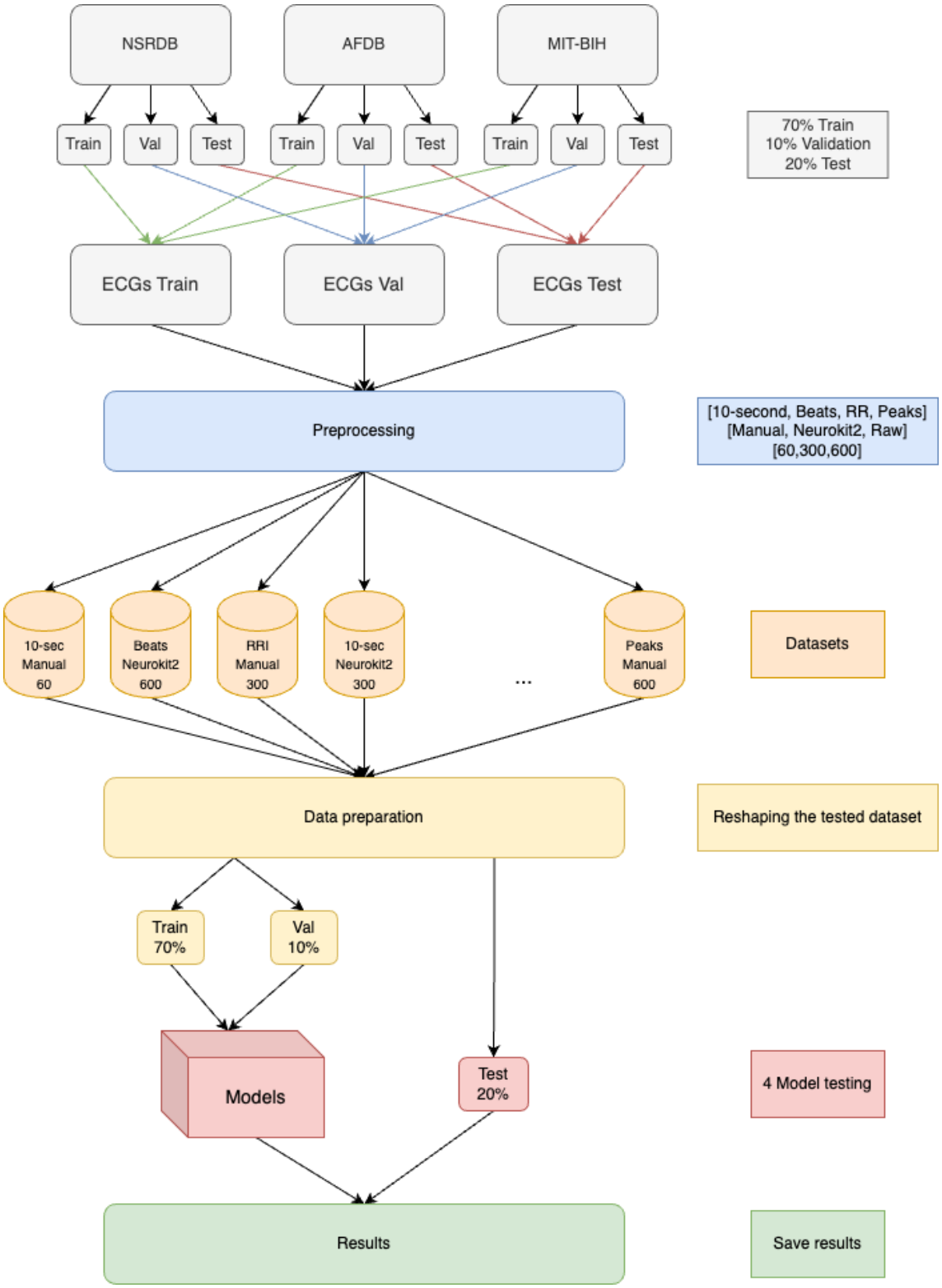


Figure 6.24: Pipeline for preprocessing, data preparation and model testing

Chapter 7

Results

7.1 Different combinations

In this study, four distinct models underwent a process of training, validation, and testing across a diverse set of 24 datasets. These datasets were a representation of the various methodologies, preprocessing techniques, and ECG recording durations that were utilised. The ECG recording durations were categorised into three groups: one hour, five hours, and the full ten-hour recording. This approach's main objective was to make comparing alternative methodologies, preprocessing methods, and recording times simpler. This was done in order to fully comprehend each strategy that might be applied to solve the problem.

7.2 Table of results

Upon completion of the model runs across the diverse datasets, several key performance metrics were gathered. These included accuracy, sensitivity, specificity, and the F1 score. As previously discussed in Chapter 3.9 *Statistics*, these metrics provide a comprehensive evaluation of the model's performance. The columns of the subsequent table are arranged in the following order:

- Row: The combination ranking, sorted by accuracy.
- Met: The methodology used (10-second, Beats, RRI, Peaks).
- Qua: The preprocessed data quality (Manual, Neurokit2, Raw).
- Dur: The recording duration used for preprocessing, specified in minutes. The durations include 60 minutes (equivalent to 1 hour), 300 minutes (equivalent to 5 hours), and 600 minutes (equivalent to 10 hours).
- Acc: The accuracy in percentage.
- Sen: The sensitivity in percentage.
- Spe: The specificity in percentage.
- F1: The F1 Score in percentage.

For a comprehensive overview of the various results, please see Table A.1 in the appendix.

Row	Met	Pre	Dur	Acc	Sen	Spe	F1
5	10-second	Neurokit2	600	94.9	94.9	95.0	94.9
8	10-second	Manual	300	94.2	92.7	94.7	87.9
10	Peaks	Manual	600	93.0	95.0	91.2	92.9
11	Peaks	Manual	300	93.0	91.9	94.0	92.8
12	10-second	Manual	600	92.7	91.9	93.0	84.9
15	10-second	Manual	60	92.5	97.4	90.6	87.7
16	10-second	Neurokit2	300	92.3	94.2	91.9	81.3
17	10-second	Neurokit2	60	91.7	96.1	90.1	86.3
24	Beats	Neurokit2	60	89.6	89.9	89.2	90.5
26	10-second	Raw	60	88.7	88.5	88.8	82.6
28	Peaks	Manual	60	87.7	87.8	87.6	87.8
33	Beats	Manual	300	86.0	83.5	88.2	85.1
35	Beats	Neurokit2	300	85.4	89.2	81.6	85.8
38	RRI	Manual	600	84.4	81.2	94.7	88.8
45	Beats	Manual	600	82.8	78.2	89.3	84.2
46	Beats	Neurokit2	600	82.3	85.9	79.3	81.6
63	Beats	Manual	60	72.9	86.7	39.7	81.8
65	10-second	Raw	300	71.6	86.0	67.2	58.7
71	RRI	Manual	60	68.0	67.4	70.0	76.6
74	RRI	Manual	300	62.8	53.2	93.1	68.5
79	10-second	Raw	600	53.6	99.1	41.1	47.9
86	Beats	Raw	60	49.0	34.8	61.8	39.2

Table 7.1: Model performance metrics for CNN

In the evaluation of our CNN model in Table 7.1, the methodology employing a 10-second window with Manual preprocessing appears to be among the top-performing configurations. Specifically, when the preprocessing is set to Manual and the duration is 300 minutes, the model achieves an accuracy of 94.2%. This notable performance underscores the importance of careful feature extraction and the potential advantages of Manually cleaned data. The Beats methodology, on the other hand, displayed significant variability, with accuracy results ranging from a commendable 89.6% (with Neurokit2 preprocessing at a 60-minute duration) to a low of 49.0% (with raw preprocessing at the same 60-minute duration). This emphasises the role of preprocessing and the choice of data window in determining model performance. Moreover, the disparities in sensitivity and specificity metrics across configurations underline the necessity for a balanced model. For instance, while the Peaks methodology with Manual preprocessing at a 600-minute duration achieved a sensitivity of 95.0%, its specificity stood at 91.2%. These variations suggest the importance of model tuning and configuration, especially when responding to the specific needs of a clinical or diagnostic application.

For the LSTM model shown in Table 7.2, it is evident that the 10-second window with Neurokit2 preprocessing is among the top-performing configurations. Specifically, when the preprocessing is set to Neurokit2 and the duration is 300 minutes, the model achieves an accuracy of 91.1%. This robust performance is further observed with a 600-minute duration, where the accuracy slightly dips to 90.9%. These results spotlight the effectiveness of LSTM when paired with the Neurokit2 preprocessing technique. In contrast, the

Row	Met	Pre	Dur	Acc	Sen	Spe	F1
19	10-second	Neurokit2	300	91.1	92.0	90.3	91.2
20	10-second	Neurokit2	600	90.9	91.8	90.1	91.0
21	10-second	Manual	60	90.9	92.7	89.1	91.0
30	RRI	Manual	600	86.9	78.7	95.3	85.9
31	10-second	Manual	600	86.7	80.8	92.6	85.9
36	10-second	Manual	300	85.3	90.4	80.1	86.2
40	Peaks	Manual	600	84.3	84.8	83.8	83.8
49	Beats	Neurokit2	600	81.9	88.4	75.8	82.6
50	Peaks	Manual	300	81.3	82.5	80.2	81.4
52	Beats	Manual	600	81.0	87.3	74.2	82.5
54	10-second	Raw	600	80.1	29.6	93.9	39.1
60	10-second	Raw	300	76.4	15.2	93.0	21.5
66	10-second	Raw	60	71.4	20.7	93.5	30.5
70	Peaks	Manual	60	69.0	42.3	96.1	57.8
72	Beats	Neurokit2	300	67.8	85.0	51.1	72.3
75	Beats	Manual	60	62.0	71.6	39.0	72.7
76	Beats	Manual	300	61.0	20.6	98.1	33.6
77	RRI	Manual	300	55.1	99.1	10.4	69.0
80	RRI	Manual	60	53.4	96.5	8.1	67.9
85	Beats	Raw	60	49.8	32.1	65.8	37.7
87	Beats	Neurokit2	60	48.8	94.5	6.3	64.0

Table 7.2: Model performance metrics for LSTM

RRI methodology showcased relatively modest results, with a notably lower accuracy of 55.1% for the 300-minute duration. This might suggest a potential misalignment between LSTM's capabilities and the characteristics of RRI data. The Beats and Peaks methodologies exhibited variable performances. While configurations like the Peaks methodology with Manual preprocessing at a 600-minute duration achieved a commendable accuracy of 84.3%, others, particularly within the Beats methodology, recorded performances as low as 48.8% (with Neurokit2 preprocessing at a 60-minute duration). These discrepancies underline the pivotal role of preprocessing in shaping outcomes. It is also worth noting that while some configurations, such as the RRI approach with Manual preprocessing for a 300-minute duration, achieved impressive sensitivities nearing 99.1%, their specificities were considerably low, standing at 10.4%. Such variations highlight the challenge of achieving balanced model performance across both sensitivity and specificity metrics.

The performance metrics shown in Table 7.3 for ResNet are varied. The methodology using a 10-second window, particularly when combined with Manual preprocessing for a 600-minute duration, achieved an outstanding accuracy of 96.0%. The sensitivity and specificity metrics for this configuration were 94.8% and 96.4%, respectively, indicating commendable model performance. The RRI methodology paired with Manual preprocessing for durations of 300 and 600 minutes also displayed impressive accuracies of 90.1% and 92.5% respectively. With the 300-minute configuration reaching a specificity of 78.8% and a sensitivity of 93.7%, while the 600-minute configuration showcased a higher sensitivity of 95.1% but a slightly lower specificity of 84.2%. The 10-second window combined with Neurokit2 preprocessing, especially for the 600-minute duration, showcased an accuracy of

Row	Met	Pre	Dur	Acc	Sen	Spe	F1
1	10-second	Manual	600	96.0	94.8	96.4	91.4
2	10-second	Neurokit2	600	96.0	95.3	96.6	95.9
6	10-second	Manual	300	94.8	99.7	89.8	95.1
7	Peaks	Manual	600	94.5	89.8	98.9	94.0
14	RRI	Manual	600	92.5	95.1	84.2	95.0
22	RRI	Manual	300	90.1	93.7	78.8	93.5
25	10-second	Manual	60	88.9	99.8	84.9	83.2
32	Beats	Manual	300	86.5	82.9	89.7	85.4
53	RRI	Manual	60	80.6	99.9	60.4	84.1
55	10-second	Neurokit2	300	80.0	99.6	75.8	63.7
56	10-second	Raw	300	80.0	27.6	87.5	25.8
59	10-second	Raw	600	77.9	38.6	93.3	49.7
61	Beats	Neurokit2	600	74.5	90.7	60.8	76.4
62	Beats	Neurokit2	300	73.0	74.4	71.7	73.1
64	10-second	Neurokit2	60	72.2	79.5	68.4	66.0
67	Beats	Manual	60	71.0	78.0	54.2	79.1
68	Beats	Manual	600	70.0	98.7	29.3	79.4
69	10-second	Raw	60	69.6	0.0	100.0	0.0
78	Peaks	Manual	300	54.3	88.8	20.4	65.8
81	Beats	Neurokit2	60	53.3	80.2	20.0	65.5
83	Beats	Raw	60	52.2	0.8	98.3	1.7
84	Peaks	Manual	60	50.3	81.0	19.1	62.1

Table 7.3: Model performance metrics for ResNet

96.0%. Interestingly, while the accuracy is the same as that achieved by Manual preprocessing for the same duration, the sensitivity was slightly higher at 95.3%, and the specificity was also higher at 96.6%. The Beats methodology presented a spectrum of outcomes. With Manual preprocessing for 300 minutes, it achieved a respectable accuracy of 86.5%. However, with Neurokit2 preprocessing, especially for shorter durations like 60 minutes, the accuracy dropped to 53.3%. The Peaks methodology, when combined with Manual preprocessing for a 600-minute duration, achieved a commendable accuracy of 94.5%. Yet, for shorter durations, such as 60 minutes, this accuracy significantly dropped to 50.3%.

The RandomForest model, as presented in Table 7.4, showcases commendable stability across various preprocessing methodologies and techniques. Notably, the 10-second window, when paired with Manual and Neurokit2 preprocessing, especially for durations of 600 minutes, has achieved accuracy rates of 95.3% and 94.9%, respectively. The sensitivity values for these configurations are 93.4% and 94.3%, highlighting the model's balanced performance. The RRI methodology, particularly for a duration of 600 minutes, achieved an accuracy of 87.3%. While its specificity was relatively lower at 64.2%, it maintained a high sensitivity of 94.6%, emphasising the model's ability to effectively identify positive cases in this context. The Peaks methodology, combined with Manual preprocessing, demonstrated consistent performance, reaching accuracies of 93.1% for 600 minutes and 91.4% for 300 minutes. This consistency is evident in the Sensitivity and Specificity metrics, which remain balanced across durations. However, the Beats methodology displayed a broader spectrum of outcomes. While it managed to achieve a respectable accuracy of 85.3% with

Row	Met	Pre	Dur	Acc	Sen	Spe	F1
3	10-second	Manual	600	95.3	93.4	95.8	89.8
4	10-second	Neurokit2	600	94.9	94.3	95.6	94.9
9	Peaks	Manual	600	93.1	92.2	94.0	92.8
13	10-second	Neurokit2	300	92.7	96.0	92.0	82.3
18	Peaks	Manual	300	91.4	89.7	93.1	91.2
23	10-second	Neurokit2	60	89.7	86.4	90.9	81.9
27	10-second	Manual	60	88.0	81.1	90.6	78.8
29	RRI	Manual	600	87.3	94.6	64.2	91.8
34	10-second	Manual	300	86.0	84.4	87.6	85.9
37	Beats	Manual	300	85.3	88.5	80.9	87.5
39	RRI	Manual	60	84.4	91.9	57.6	90.1
41	Beats	Neurokit2	60	84.0	82.7	85.7	85.1
42	Beats	Manual	600	82.9	88.1	75.6	85.8
43	RRI	Manual	300	82.9	90.7	58.2	88.9
44	Peaks	Manual	60	82.9	82.0	83.7	82.8
47	10-second	Raw	60	82.1	75.3	85.0	71.9
48	Beats	Neurokit2	300	82.0	89.9	75.4	81.9
51	10-second	Raw	600	81.1	72.6	83.4	62.3
57	10-second	Raw	300	78.6	97.0	72.9	68.0
58	Beats	Neurokit2	600	78.2	89.6	68.7	79.0
73	Beats	Manual	60	67.0	75.2	47.5	76.3
82	Beats	Raw	60	53.2	53.5	53.0	52.0

Table 7.4: Model performance metrics for RandomForest

Row	Met	Pre	Dur	Mod	Acc	Sen	Spe	F1
5	10-second	Neurokit2	600	CNN	94.9	94.9	95.0	94.9
19	10-second	Neurokit2	300	LSTM	91.1	92.0	90.3	91.2
3	10-second	Manual	600	RandomForest	95.3	93.4	95.8	89.8
1	10-second	Manual	600	ResNet	96.0	94.8	96.4	91.4

Table 7.5: Best model performance metrics by accuracy

Manual preprocessing for 300 minutes, its performance varied significantly across other configurations. For instance, with Neurokit2 preprocessing for 60 minutes, the accuracy was 84.0%, but the sensitivity dropped to 47.5% with the same preprocessing for a shorter duration. This variability accentuates the intricate interplay between preprocessing techniques, data duration, and the RandomForest model.

In Table 7.5, when assessing the best-performing models based on their accuracy across various methodologies and preprocessing techniques, a distinct ranking becomes apparent. Leading the pack is the ResNet model, which, when combined with a 10-second window, Manual preprocessing, and a 600-minute duration, attains an impressive accuracy of 96.0%. Not far behind is the RandomForest model using a 10-second window, Manual preprocessing, and a 600-minute duration, which achieves an accuracy of 95.3%. The CNN model, employing a 10-second window with Neurokit2 preprocessing for 600 minutes, posts a commendable accuracy of 94.9%. The LSTM model, in its best configuration with a 10-second window, Neurokit2 preprocessing, and a 300-minute duration, secures an accuracy of 91.1%, which, while comparatively lower, remains commendably high.

7.3 Analysis of results

Overall, the models produced good results, with a few significant exceptions. The results show that the models are capable of learning from a variety of datasets, indicating their adaptability and robustness.

Before delving into the broader findings, it is essential to first examine these exceptions.

7.3.1 Bad results

The models at the bottom of the performance tables have accuracies close to or even less than 50%, which is equivalent to the performance of a random prediction model. This suggests that identifying the relevant features for accurate prediction can be difficult. A multitude of factors may be behind this mediocre performance:

- Model selection: The choice of model itself can greatly influence the results. It is essential to choose a model that matches the nature of the data and the problem posed.
- Model architecture: The architecture, including the number and type of layers, activation functions, and connections, can have a significant impact on the model's ability to learn from the data.
- Hyperparameter selection: Incorrectly set hyperparameters can limit a model's potential, resulting in under- or over-fitting.
- Data quality: The integrity and quality of the data sets used cannot be overemphasised. Noisy or poor-quality data can seriously affect the performance of a model.

It highlights the difficulty of the work and the necessity of thoroughly analysing and optimising each step of the model creation process.

7.3.2 Other results

The remaining results, with the exception of those involving Raw data, where caution must be used to ensure that the model does not overfit on bad features such as ECG recording noise, present a convincing argument for the models' effectiveness across various combinations of methodologies, data quality, ECG recording durations, and model types. These models have accuracy rates ranging from 70% to 96%, as well as equally convincing specificity, sensitivity and F1 scores. This indicates that the models were successful in learning from the data and can correctly identify patients with atrial fibrillation based on ECGs recorded during normal sinus rhythm.

Interestingly, the results show that the same model architecture (for example, ResNet) can perform very differently depending on the methodology and preprocessing techniques used. The ResNet model, for example, achieved a precision of 88.9% on 10-second, 60-minute Manual preprocessing, whereas the same model trained on Beats 60-minute preprocessed data with Neurokit2 had an accuracy of 53.3%, a specificity of 20%, and an F1 score of 65.5%. This emphasises the significance of data preprocessing and methodology selection for model performance.

Furthermore, the findings indicate that specific models may demonstrate a preference or better alignment with certain data types. For instance, the RandomForest model consistently exhibits strong performance with the 10-second data when preprocessed using Neurokit2. On the other hand, while the CNN model showcases commendable outcomes

with several datasets, it particularly stands out with Beats data. Such observations underscore the importance of aligning model selection with data characteristics. Figure 7.1 and Figure 7.2 offer a detailed comparative analysis of performance across various methodologies.

	Accuracy	**Sensitivity**	**Specificity**	**F1 score**
10-second	85.4	79.1	87.5	72.4
Beats	71.9	75.0	66.1	71.5
RRI	77.4	86.8	64.6	83.4
Peaks	81.2	84.0	78.5	82.1

Figure 7.1: Metrics by methodology

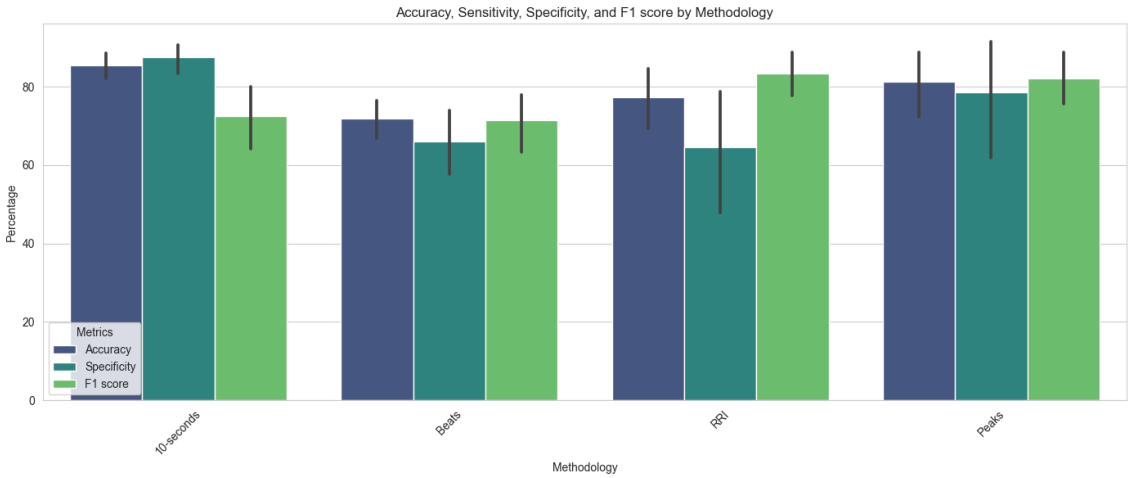


Figure 7.2: Comparative analysis of the methodologies. Error bars represent a 95% confidence interval.

The 10-second method stands out as the most efficient and robust of the four. Its efficiency is evident from the higher values of the metrics, while its robustness is illustrated by the smaller error bars (representing a 95% confidence interval) in the plot. This limited variability in the results emphasises the consistency of the 10-second method. On the other hand, the Beats method exhibits the lowest performance among the four methodologies. This inferiority can be attributed to the intrinsic nature of the method, which relies on longer and variably sized segments. Such a structural approach might contribute to the observed lower scores. The RRI and Peaks methodologies have distinct performance characteristics when assessed on the three parameters of accuracy, specificity, and F1 score. More specifically, while the Peaks methodology has higher accuracy and specificity than RRI, the latter has a slight advantage in terms of sensitivity. However, the specificity of RRI is significantly lower, indicating potential difficulties in correctly identifying true negative cases. The data highlights that, compared with Peaks, the RRI method may be less robust, particularly in terms of specificity.

7.3.3 10-second methodology

A comprehensive overview of all the results prompts a more detailed examination of the performance of the different methodologies. The 10-second method consistently ranks highly across all dataset combinations and model types, suggesting its robustness and general applicability. For instance, attention may be drawn to the combination of 10-

second, Neurokit2, 300, and ResNet, as shown in row 55 of the Table 7.3. This combination closely resembles Attia et al.'s strategy, which served as a model for this method. Their study reported an accuracy of 79.4%, a specificity of 79.5%, and an F1-score of 39.2% on a proprietary dataset, with uniformity in ECG recording maintained throughout.

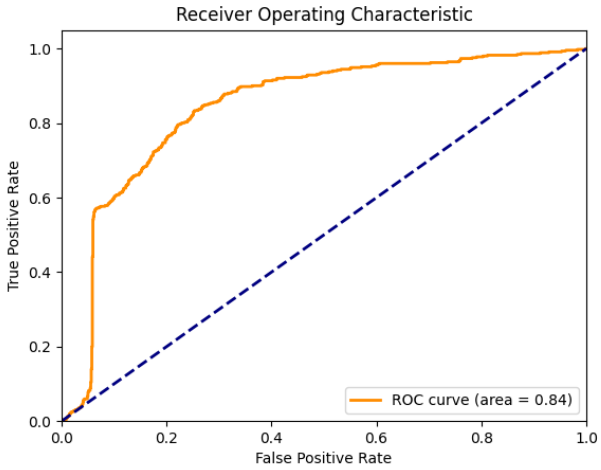


Figure 7.3: ROC and AUC of 10-second, Neurokit2, 300 minutes, ResNet combination

	Predicted Label	
True Label	NS	AF
NS	4694	1498
AF	4	1322

Table 7.6: Confusion Matrix of 10-second, Neurokit2, 300 minutes, ResNet combination

In contrast, the present study achieved an accuracy of 80%, a sensitivity of 99.6%, a specificity of 75.8%, and an F1-score of 63.7% with the same method and model. A closer look at the ROC and AUC in Figure 7.3, as well as the confusion matrix in Table 7.6, provides further insight into these impressive results. These results, while excellent, can be attributed to a key factor: the use of data from various external sources. Despite rigorous and objective preprocessing to ensure data uniformity, it is not always possible to completely eliminate or mask certain undesirable features unique to each dataset. Nevertheless, the model has successfully learned the essential features and has also leveraged certain undesirable features to achieve high performance.

It is crucial to note, however, that while this approach is viable and correct, it would benefit from further fine-tuning with cleaner data.

7.3.4 Beats methodology

The performance of the Beats methodology, as evidenced by its placement at the bottom of the ranking when trained with 60 minutes of data, was notably below. Despite efforts to improve outcomes through additional preprocessing and model reruns, significant enhancements in performance were not observed. Several factors could contribute to these not convincing results. One potential issue is the scarcity of data. Even with overlap, the model may not have sufficient data for effective learning. Given that 100 beats often span more than a minute, locating segments of 100 consecutive beats can pose a challenge, particularly when a single 'unhealthy' beat is embedded within the entire segment. The quality of the data provided to the model may also be a contributing factor. As with the 10-second methodology, the datasets are sourced externally, and despite rigorous preprocessing to standardise them, some undesirable features may persist. Because larger segments inherently contain a greater number of undesirable features than smaller

segments, the presence of undesirable features may be amplified as segment size increases. This is due to the fact that a 100-beat segment is longer by definition than a 10-second segment.

Nonetheless, it is important not to dismiss the approach's effectiveness. One illustrative instance can be observed in row 24 of Table 7.1, where a combination of Beats, Neurokit2 and 60 minutes is presented. The resulting combination produced an accuracy of 89.6%, a sensitivity of 89.9%, a specificity of 89.2%, and an F1 score of 90.5%. Further insights into the model's performance can be gained from the ROC and AUC in Figure 7.4 and the confusion matrix in Table 7.7.

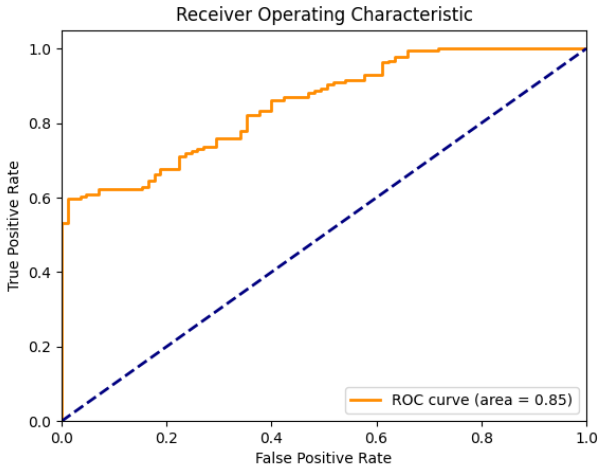


Figure 7.4: ROC and AUC of Beats, Neurokit2, 60 minutes, CNN combination

	Predicted Label	
True Label	NS	AF
NS	401	48
AF	56	499

Table 7.7: Confusion Matrix of Beats, Neurokit2, 60 minutes, CNN combination

These results indicate that the model is functional and capable of learning effectively. Thus, it can be concluded that the Beats methodology can be employed to identify atrial fibrillation in normal sinus rhythm. However, it's worth mentioning that, in a comparative analysis, this method generally exhibits less effectiveness than the others.

7.3.5 RR Intervals methodology

The RRI method's results are proof of its effectiveness. Regardless of the model used—whether it's an LSTM, a CNN, or one of the other two models—the outcomes remain satisfactory. The study by Duan et al. about fibrillation detection, which served as a reference for this model, reported an accuracy of 96% and a specificity of 99% across various simple models, including the RandomForest model that inspired this work. These are exceptional scores. It's worth noting that the same datasets used in this work were also used by the author, underscoring the value of using RR intervals to identify AF in normal sinus rhythm.

Row 29 of Table 7.4 further supports this utility. For the combination of RRI, Manual preprocessing, 600 minutes, and RandomForest, an accuracy of 87.3%, a sensitivity of 94.6%, a specificity of 64.2%, and an F1-score of 91.8% were achieved. These results are far from those reported in the study, but they nevertheless demonstrate the effectiveness of this method. The ROC and AUC depicted in Figure 7.5, along with the confusion matrix presented in Table 7.8, further corroborate this assertion.

The results obtained can be attributed to the data's uniformity. By retaining only

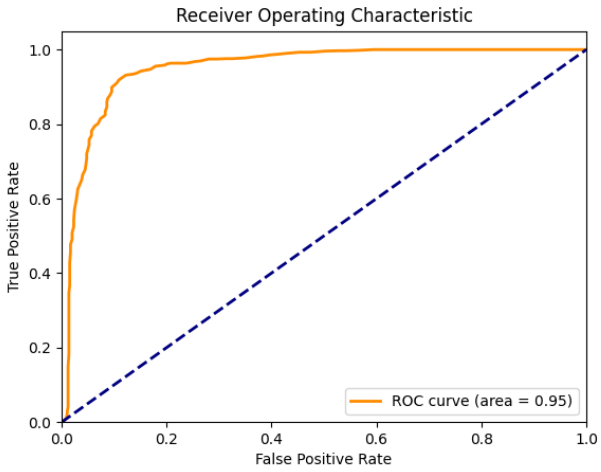


Figure 7.5: ROC and AUC of RRI, Manual, 600 minutes, RandomForest combination

	Predicted Label	
True Label	NS	AF
NS	1569	873
AF	417	7314

Table 7.8: Confusion Matrix of RRI, Manual, 600 minutes, RandomForest combination

RR intervals, a significant portion of undesirable features are removed, leaving relatively clean data behind. Furthermore, the conversion from sample to millisecond is critical because it ensures that all data is represented on a common scale. This data uniformity and standardisation contribute significantly to the model's effectiveness.

7.3.6 Peaks

The final methodology implemented, Peaks, is an adaptation of the approach proposed by Castro et al. It has been modified to retain only the most significant peaks within a heartbeat. This simplification, rather than being a disadvantage, has proven beneficial. In fact, the Peaks methodology generally exhibits commendable performance. A notable achievement is illustrated in row 40 of Table 7.2, where the combination of Peaks, Manual preprocessing, 600 minutes of data, and the LSTM model resulted in an accuracy of 84.3%, a sensitivity of 84.8%, a specificity of 83.8%, and an F1-score of 83.8%. The efficiency of this approach is further corroborated by Figure 7.6 and Table 7.9.

The success of this approach can be attributed to the nature of the data used. Similar to the RR Interval methodology, only a select few measurements, directly extracted from heartbeats, were utilised. The simplicity and uniformity of this data facilitated the extraction of important features. The data was less susceptible to unintended features, such as noise, thereby yielding more pertinent results.

7.3.7 Preprocessing comparaison

Figure 7.7 and Figure 7.8 give us an evaluation of the different preprocessing methods. In evaluating the outcomes of various preprocessing methods applied to this problem, it becomes evident that Manual preprocessing exhibits a higher efficacy compared to the automatic preprocessing facilitated by Neurokit2. While the disparity between the two approaches may not be substantial, it constitutes a noteworthy difference. However, it must be acknowledged that Manual preprocessing is intricate and demands a significant investment of time, making it a less appealing option in scenarios where expedience is a priority. In such cases, utilising Neurokit2 may be a more pragmatic choice. Additionally,

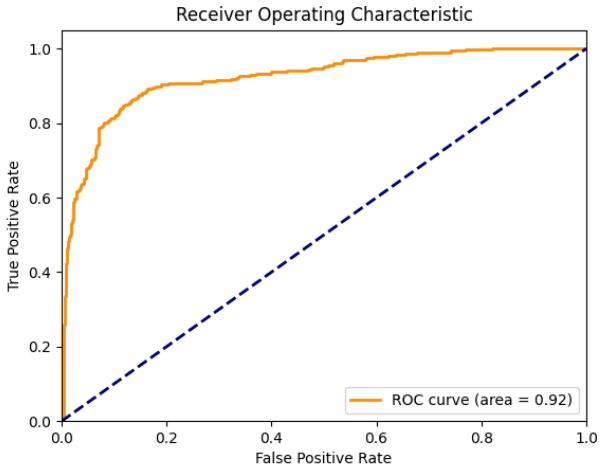


Figure 7.6: ROC and AUC of Peaks, Manual, 600 minutes, LSTM combination

	Predicted Label	
True Label	NS	AF
NS	3373	648
AF	564	3151

Table 7.9: Confusion Matrix of Peaks, Manual, 600 minutes, LSTM combination

Preprocessing Method	Accuracy	Sensitivity	Specificity	F1 Score
Manual	83.4	85.2	75.2	84.7
Neurokit2	82.1	89.6	75.8	81.2
Raw	69.8	48.3	80.0	43.1

Figure 7.7: Metrics by preprocessing method

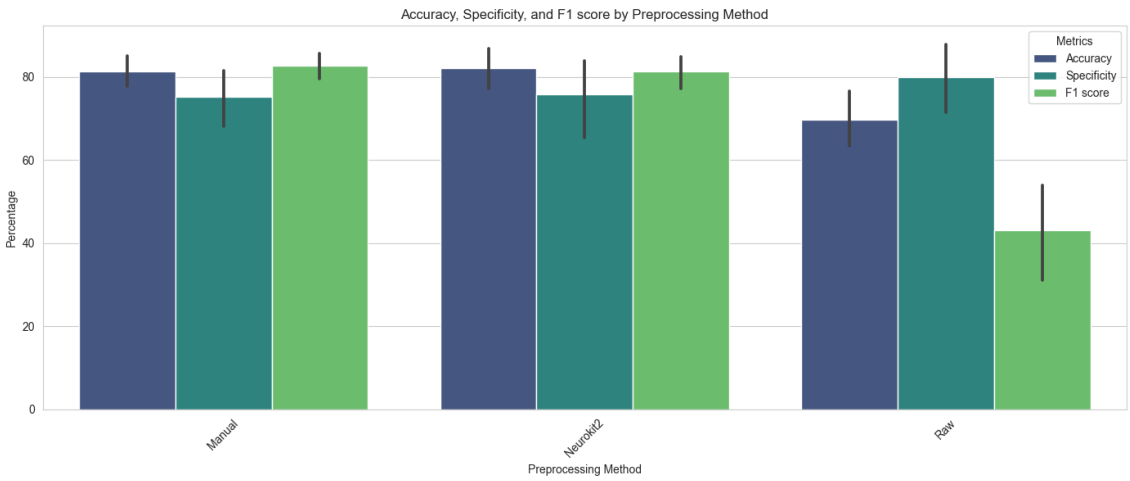


Figure 7.8: Comparison of the preprocessing methods

it is evident that the Raw method significantly lags behind the other two in terms of accuracy. Its sensitivity is particularly low, indicating that the model fails to identify more than half of the actual AF cases when using raw data.

7.3.8 Models comparaison

Model	Accuracy	Sensitivity	Specificity	F1 Score
CNN	82.3	84.8	82.4	80.4
LSTM	74.1	70.4	71.8	68.0
RandomForest	83.8	85.9	78.7	81.9
ResNet	77.2	76.8	71.8	69.4

Figure 7.9: Metrics by model

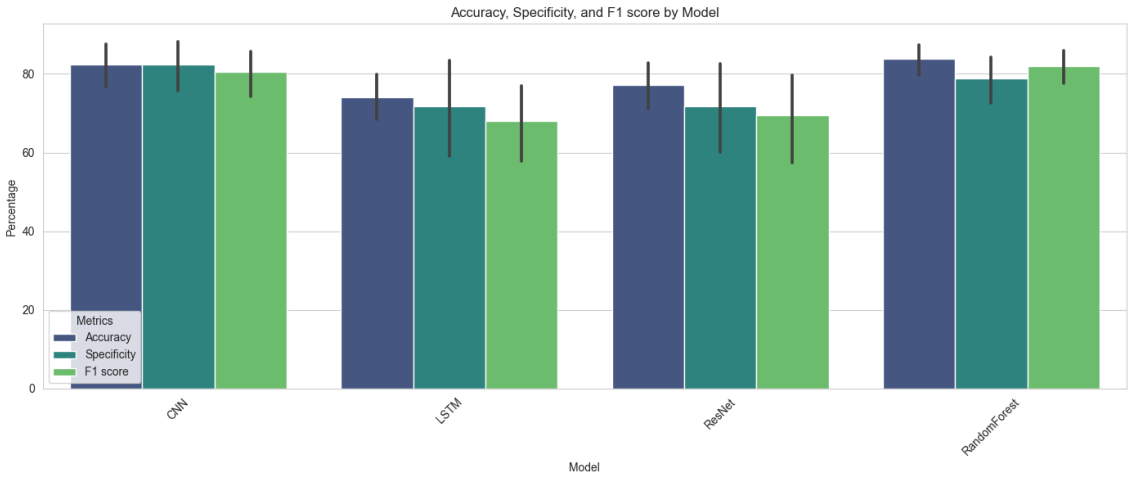


Figure 7.10: Comparison of the models

Figure 7.9 and Figure 7.10 provide an insightful comparison of the efficacy of the various models examined in this study. While the performance of these models appears to be relatively consistent for the task at hand, the LSTM model stands out as an exception, demonstrating a marked lag in efficiency. This discrepancy can be attributed to the LSTM's less effective structure, which consequently hindered its learning capability. The results obtained from both CNN and ResNet are comparable and satisfactory, affirming that they represent feasible and viable solutions to the problem at hand. However, caution must be exercised with ResNet, as it exhibits greater variability in results compared to CNN, rendering it a somewhat less consistent solution in terms of outcomes.

It is important to remember that, while the results were valid, they could be improved. To ensure an unbiased comparison, the models' structure, layers, and hyperparameters were kept constant across multiple dataset tests. This method guarantees consistency, but it might not be the best option for implementation. A more effective method would involve individually tailoring the models to suit each specific test. By doing so, the models could potentially be fine-tuned to better align with the unique characteristics of each dataset, thereby improving overall performance.

7.3.9 Different sampling frequencies

It is also interesting to examine the results obtained using different sampling frequencies. For this project, we used a frequency of 360Hz, a standard often adopted in research that

Row	Met	Pre	Dur	Mod	Acc	Sen	Spe	F1
5	10-second	Neurokit2	600	CNN	90.4	94.4	93.9	73
19	10-second	Neurokit2	300	LSTM	61.9	53	54.1	52.4
3	10-second	Manual	600	RandomForest	91.2	94.3	95.3	81.7
1	10-second	Manual	600	ResNet	82.7	85.8	84.6	83.4

Table 7.10: 128 Hz Model Performance Metrics

Row	Met	Pre	Dur	Mod	Acc	Sen	Spe	F1
5	10-second	Neurokit2	600	CNN	95	79.9	94.1	88.2
19	10-second	Neurokit2	300	LSTM	95.7	96.6	95.4	90.3
3	10-second	Manual	600	RandomForest	96.8	96.9	96.7	91.9
1	10-second	Manual	600	ResNet	73.8	76.4	67.7	59.1

Table 7.11: 250 Hz Model Performance Metrics

provides a solid baseline. By comparing our best results with those obtained at frequencies of 128 Hz and 250 Hz - corresponding to the sampling rates of the other two datasets - we can discern the differences. The comparative results are shown in Tables 7.10 and 7.11.

The variations observed in the results, whether improvements or regressions, are particularly enlightening. They highlight the fact that sampling frequency is closely linked to the preprocessing steps, which in turn have a significant influence on the final results. It is therefore important to carefully study sampling frequency in conjunction with preprocessing techniques in order to obtain optimal results.

Chapter 8

Conclusion

In the pursuit of identifying patients with atrial fibrillation during periods of normal sinus rhythm through the application of machine learning, a diverse array of methodologies has been examined.

Four distinct methodologies were employed, each rooted in scientific literature that proposes innovative approaches. These methodologies utilised a 10-seconds segment, a specific number of heartbeats, RR intervals, or peaks from a heartbeat.

Preprocessing emerged as a pivotal component in this process. Inspired by scientific research, two primary methods were developed: a manual method employing a Butterworth filter and a Denoising Wavelet, and an automatic method derived from a reputable package. These were used to cleanse the data, thereby facilitating a comparison between the two. Additionally, raw data was used to validate the efficacy of the aforementioned techniques.

Data was recovered from three different-sized portions of the dataset: one hour, five hours, and ten hours (the entire dataset). This was done to investigate and analyse the learning capacity of the different models and methodologies when provided with an increasing volume of data. The datasets for each combination were subsequently extracted from the baseline ECGs, thereby generating a broad spectrum of data for future utilisation.

Models from the same articles were implemented to establish a common foundation across all the methodologies. These models included a CNN, an LSTM, a ResNet, and a RandomForest. These models, which remained consistent across each dataset, were directly sourced from the articles. The model type, structure, and hyperparameters were closely aligned with those used in the articles, thereby facilitating a more accurate comparison of results. These models were subsequently run and tested.

The results obtained were promising. It was demonstrated that it is indeed feasible to identify patients with atrial fibrillation using the various methods and models employed. Although some models yielded poor results, the majority displayed encouraging outcomes. The 10-second, Beats, and RRI methods all yielded promising results. The Beats method also demonstrated good results, albeit slightly trailing the others.

The findings of this study highlight the possibility of identifying patients with atrial fibrillation quickly through simple screening during routine medical appointments.

From a medical standpoint, these findings highlight the possibility of disease prevention by rapidly identifying and treating patients. Even if an ECG appears normal at first, it may mask the onset of a severe cardiac condition that would otherwise go undetected until it was too late. The ability to detect atrial fibrillation during normal sinus rhythm is a significant step forward in improving healthcare delivery and accelerating patient treatment.

These findings, from the perspective of an AI developer, demonstrate the immense po-

tential for AI to augment patient management in the medical field. This study emphasises the advancement and potential of artificial intelligence and the opportunities it could bring to a variety of sectors.

Furthermore, it implies that the medical business and information technology, particularly artificial intelligence, may become more integrated in the near future than is currently envisaged. This proposed merger could usher in a new age in healthcare, one in which innovations in technology and medicine collaborate to improve patient outcomes.

This research, despite its significant findings, is not without its limitations. The primary constraint arises from the fact that the data was sourced from various external entities. This necessitates the alteration of raw data, either manually or automatically, which could potentially lead to the modification or even loss of important features.

Another limitation lies in the uniform application of the models across all runs. While this ensures a degree of objectivity in the results across different combinations, it is inevitable that some models, due to their inherent structure and nature, may not perform as optimally as others depending on the type of data used.

Lastly, the methodologies employed, particularly Peaks, have been slightly adapted from the descriptions provided in the reference articles. These approaches had to be altered to closely coincide with the reference material due to ambiguities or omissions in the articles. These little differences, while appearing insignificant, could have an enormous influence on the final results.

This research opens up several potential avenues for future exploration. One possible direction could be to source a uniform dataset and test the methodologies employed in this study on data that is entirely clean and uniform, possibly with multiple leads. Applying the entire pipeline, including the methodology, preprocessing, and models, to such a dataset and analysing the results could provide valuable insights. A comparison of these results with those presented in this study could also prove enlightening.

Building on the findings of this research, another promising avenue could involve a detailed analysis of the most successful results. With a reduced number of models, it would be feasible to perform k-fold cross-validation, an approach that was not possible in this study due to the volume of data and combinations used.

Lastly, the application of other models, including deep learning models or even simpler ones, could be explored. Numerous articles propose specific solutions and models, and investigating these ideas could further enrich this field of study.

In summary, this research underscores the utility of employing machine learning techniques in the medical domain, specifically for the early detection of atrial fibrillation. The methodologies, pre-processing techniques, and models utilised have demonstrated effectiveness, with evidence supporting their significant contribution to enhancing patient management, even during routine check-ups.

This study has the potential to contribute to or serve as a foundation for further research on the identification of patients with atrial fibrillation in the presence of normal sinus rhythm.

Bibliography

[1] Wikipedia. Heart — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Heart&oldid=1083496604, 2022. [Online; accessed 9-April-2022]. [Cited on pages , 2, and 3.]

[2] Wikipedia. Electrical conduction system of the heart — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Electrical%20conduction%20system%20of%20the%20heart&oldid=1065264721, 2022. [Online; accessed 9-April-2022]. [Cited on pages and 3.]

[3] Wikipedia. Electrocardiography — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Electrocardiography&oldid=1082518486, 2022. [Online; accessed 24-April-2022]. [Cited on pages , 4, and 8.]

[4] Wikipedia. Électrocardiographie — Wikipedia, the free encyclopedia. http://fr.wikipedia.org/w/index.php?title=%C3%89lectrocardiographie&oldid=190884204, 2022. [Online; accessed 9-April-2022]. [Cited on pages and 4.]

[5] Axillary lines, Oct 2021. [Cited on pages and 7.]

[6] Amit says:, FeirozHumayara says:, Christopher says:, Sandra Bailey says:, Ata says:, and Rachel says:. The ecg leads, polarity and einthoven's triangle, Nov 2016. [Cited on pages and 8.]

[7] Ash Bhalgat. Scaling out the deep learning cloud efficiently. https://developer.nvidia.com/blog/scaling-out-the-deep-learning-cloud-efficiently/, Jan 2021. [Online; accessed 19-April-2022]. [Cited on pages and 10.]

[8] Alfonso Ibañez. Semi-supervised learning... the great unknown. https://business.blogthinkbig.com/semi-supervised-learning-the-great-unknown/, May 2019. [Online; accessed 19-April-2022]. [Cited on pages and 11.]

[9] Roohollah Amiri, Hani Mehrpouyan, Lex Fridman, Ranjan Mallik, Arumugam Nallanathan, and David Matolak. A machine learning approach for power allocation in hetnets considering qos. 03 2018. [Cited on pages and 12.]

[10] Wikipedia. Neural network — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Neural%20network&oldid=1077721872, 2022. [Online; accessed 18-April-2022]. [Cited on pages and 12.]

[11] Ali Polat. Hidden bank charges amidst ethics, transparency, and regulation: Case of rebate programs of international banks. 11 2017. [Cited on pages and 13.]

[12] Vishal Jhaveri. Understanding how convolutional neural networks work. https://vishalj7.github.io/Convolutional-Neural-Networks/. [Online; accessed 19-April-2022]. [Cited on pages and 14.]

[13] Sagar Sharma. Activation functions in neural networks. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, Jul 2021. [Online; accessed 19-April-2022]. [Cited on pages and 14.]

[14] Pragati Baheti. 12 types of neural networks activation functions: How to choose? https://www.v7labs.com/blog/neural-networks-activation-functions. [Online; accessed 19-April-2022]. [Cited on pages , 14, 15, and 16.]

[15] Wikipedia. Recurrent neural network — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Recurrent%20neural%20network&oldid=1083099560, 2022. [Online; accessed 23-April-2022]. [Cited on pages , 16, and 17.]

[16] Sébastien S. and Team redac. Recurrent neural network (rnn) : De quoi s'agit-il ? https://datascientest.com/recurrent-neural-network, Jul 2021. [Online; accessed 19-April-2022]. [Cited on pages and 17.]

[17] Video MATLAB. Introduction to deep learning: What are convolutional neural networks? video. https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html. [Online; accessed 22-April-2022]. [Cited on pages and 18.]

[18] TIBCO. What is a random forest?, 2023. Accessed: 2023-07-29. [Cited on pages and 20.]

[19] Eda Kavlakoglu. Ai vs. machine learning vs. deep learning vs. neural networks: What's the difference? https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks, May 2020. [Online; accessed 24-April-2022]. [Cited on pages and 21.]

[20] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Resnet - dive into deep learning 0.17.0 documentation, 2023. Accessed: July 29, 2023. [Cited on pages and 22.]

[21] Receiver operating characteristic, Oct 2022. [Cited on pages and 24.]

[22] Apple. *Using Apple Watch for Arrhythmia Detection.* 12 2020. [Cited on pages , 27, and 28.]

[23] Henry Castro, Juan David Garcia-Racines, and Alvaro Bernal-Noreña. Methodology for detection of paroxysmal atrial fibrillation based on p-wave, hrv and qr electrical alternans features. *International Journal of Electrical and Computer Engineering (IJECE)*, 10(4):4023, 2020. [Cited on pages , 28, 29, 40, and 41.]

[24] Paulus Kirchhof, Stefano Benussi, Dipak Kotecha, Anders Ahlsson, Dan Atar, Barbara Casadei, Manuel Castella, Hans-Christoph Diener, Hein Heidbuchel, Jeroen Hendriks, Gerhard Hindricks, Antonis S Manolis, Jonas Oldgren, Bogdan Alexandru Popescu, Ulrich Schotten, Bart Van Putte, Panagiotis Vardas, and ESC Scientific Document Group. 2016 ESC Guidelines for the management of atrial fibrillation developed in collaboration with EACTS. *European Heart Journal*, 37(38):2893–2962, 08 2016. [Cited on pages 1 and 26.]

[25] Mary Elizabeth Lewis, Feng-Chang Lin, Parin Nanavati, Neil Mehta, Louisa Mounsey, Anthony Nwosu, Irion Pursell, Eugene H Chung, J Paul Mounsey, and Ross J Simpson. Estimated incidence and risk factors of sudden unexpected death. *Open Heart*, 3(1), 2016. [Cited on page 1.]

[26] Susan Standring and Henry Gray. *Gray's anatomy: The anatomical basis of Clinical Practice*. Churchill Livingstone/Elsevier, 2008. [Cited on page 2.]

[27] Jun 2022. [Cited on page 4.]

[28] Thomas Burnell and Bethany Turner. Electrocardiograms. `https://simplemed.co.uk/subjects/cardiovascular/electrocardiograms`, Aug 2019. [Online; accessed 15-May-2022]. [Cited on page 5.]

[29] Wikipedia. Tachycardia — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Tachycardia&oldid=1077404880`, 2022. [Online; accessed 27-April-2022]. [Cited on page 5.]

[30] Wikipedia. Bradycardia — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Bradycardia&oldid=1084686149`, 2022. [Online; accessed 27-April-2022]. [Cited on page 5.]

[31] Wikipedia. Fibrillation atriale — Wikipedia, the free encyclopedia. `http://fr.wikipedia.org/w/index.php?title=Fibrillation%20atriale&oldid=192488350`, 2022. [Online; accessed 28-April-2022]. [Cited on page 5.]

[32] Medtronic. Informations sur la tachycardie : Les causes, les symptômes, les facteurs de risque. [Cited on page 5.]

[33] NHS choices. Treatment artial fibrillation. `https://www.nhs.uk/conditions/atrial-fibrillation/treatment/`. [Online; accessed 15-May-2022]. [Cited on page 6.]

[34] João Loures Salinet and Olavo Luppi Silva. Chapter 2 - ecg signal acquisition systems. In João Paulo do Vale Madeiro, Paulo César Cortez, José Maria da Silva Monteiro Filho, and Angelo Roncalli Alencar Brayner, editors, *Developments and Applications for ECG Signal Processing*, pages 29–51. Academic Press, 2019. [Cited on page 7.]

[35] Benjamin E. Jin, Heike Wulff, Jonathan H. Widdicombe, Jie Zheng, Donald M. Bers, and Jose L. Puglisi. A simple device to illustrate the einthoven triangle. *Advances in Physiology Education*, 36(4):319–324, 2012. [Cited on page 7.]

[36] Rosemarie Velik. Ai reloaded: Objectives, potentials, and challenges of the novel field of brain-like artificial intelligence. *BRAIN Broad Research in Artificial Intelligence and Neuroscience*, 3:25–54, 10 2012. [Cited on page 9.]

[37] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. [Cited on page 9.]

[38] Wikipedia. John von Neumann — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=John%20von%20Neumann&oldid=1081421880`, 2022. [Online; accessed 12-April-2022]. [Cited on page 9.]

[39] Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1):57–83, 2002. [Cited on page 9.]

[40] Wikipedia. Waymo — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Waymo&oldid=1080499226, 2022. [Online; accessed 12-April-2022]. [Cited on page 9.]

[41] AlphaGo. The challenge match. https://www.deepmind.com/research/highlighted-research/alphago/the-challenge-match. [Online; accessed 12-April-2022]. [Cited on page 9.]

[42] *Intelligence artificielle: Etat de l'art et perspectives pour la France: Rapport final.* DGE, Direction generale des entreprises, 2019. [Cited on page 10.]

[43] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms.* Cambridge University Press, 2021. [Cited on page 10.]

[44] Y Reddy, Viswanath Pulabaigari, and Eswara B. Semi-supervised learning: a brief review. *International Journal of Engineering Technology*, 7:81, 02 2018. [Cited on page 11.]

[45] Jesper E. van Engelen and Holger H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2019. [Cited on page 11.]

[46] Joseph M. Carew. What is reinforcement learning? a comprehensive overview. https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning, Mar 2021. [Online; accessed 18-April-2022]. [Cited on page 11.]

[47] Wikipedia. Intelligent agent — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Intelligent%20agent&oldid=1077936658, 2022. [Online; accessed 13-April-2022]. [Cited on page 11.]

[48] Surbhi Arora. Supervised vs unsupervised vs reinforcement. https://www.aitude.com/supervised-vs-unsupervised-vs-reinforcement/, Jan 2020. [Online; accessed 18-April-2022]. [Cited on page 12.]

[49] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. [Cited on page 12.]

[50] Ahmed Gad. Beginners ask "how many hidden layers/neurons to use in artificial neural networks?". https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e, Jun 2018. [Online; accessed 20-April-2022]. [Cited on page 12.]

[51] Wikipedia. Réseau de neurones artificiels — Wikipedia, the free encyclopedia. http://fr.wikipedia.org/w/index.php?title=R%C3%A9seau%20de%20neurones%20artificiels&oldid=192353567, 2022. [Online; accessed 19-April-2022]. [Cited on page 13.]

[52] Saed Sayad. Data science for biomarkers discovery. https://www.saedsayad.com/artificial_neural_network.htm. [Online; accessed 19-April-2022]. [Cited on page 13.]

[53] Heudin Jean-Claude. *Comprendre Le Deep Learning: Une introduction aux Reseaux de Neurones.* Science-eBook, 2016. [Cited on page 14.]

[54] Lokesh Kumar. Activation function in neural network. https://blog.knoldus.com/activation-function-in-neural-network/, Dec 2021. [Online; accessed 24-April-2022]. [Cited on page 14.]

[55] Felix Gers, Nicol Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 01 2002. [Cited on page 17.]

[56] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, and N.I. Chervyakov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177:232–243, 2020. [Cited on page 18.]

[57] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. [Cited on page 18.]

[58] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning, 2015. [Cited on page 19.]

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. [Cited on page 19.]

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. IEEE, Jun 2016. [Cited on page 22.]

[61] Classification: Roc curve and auc nbsp;|nbsp; machine learning nbsp;|nbsp; google developers. [Cited on page 24.]

[62] Thomas Davenport and Ravi Kalakota. The potential for artificial intelligence in healthcare. *Future Healthcare Journal*, 6(2):94–98, 2019. [Cited on page 26.]

[63] Raju Vaishya, Mohd Javaid, Ibrahim Haleem Khan, and Abid Haleem. Artificial intelligence (ai) applications for covid-19 pandemic. *Diabetes Metabolic Syndrome: Clinical Research Reviews*, 14(4):337–339, 2020. [Cited on page 26.]

[64] Georges H. Mairesse Jean-Marie Grégoire. Dépistage de la FA: de la prise du pouls aux nouvelles technologies. *VCP : Vaisseaux - Cœur - Poumons*, 24(5):7–15, 07 2019. [Cited on page 27.]

[65] Jan Heeringa, Deirdre A.M. van der Kuip, Albert Hofman, Jan A. Kors, Gerard van Herpen, Bruno H.Ch. Stricker, Theo Stijnen, Gregory Y.H. Lip, and Jacqueline C.M. Witteman. Prevalence, incidence and lifetime risk of atrial fibrillation: The rotterdam study. *European Heart Journal*, 27(8):949–953, 2006. [Cited on page 27.]

[66] David A Fitzmaurice, F D Hobbs, Sue Jowett, Jonathon Mant, Ellen T Murray, Roger Holder, J P Raftery, S Bryan, Michael Davies, Gregory Y Lip, and et al. Screening versus routine practice in detection of atrial fibrillation in patients aged 65 or over: Cluster randomised controlled trial. *BMJ*, 335(7616):383, 2007. [Cited on page 27.]

[67] Jul 2023. [Cited on page 27.]

[68] Paulus Kirchhof, Jeroen Bax, Carina Blomstrom-Lundquist, Hugh Calkins, A. John Camm, Ricardo Cappato, Francisco Cosio, Harry Crijns, Hans-Christian Diener, Andreas Goette, Carsten W. Israel, Karl-Heinz Kuck, Gregory Y.H. Lip, Stanley Nattel, Richard L. Page, Ursula Ravens, Ulrich Schotten, Gerhard Steinbeck, Panos

Vardas, Albert Waldo, Karl Wegscheider, Stephan Willems, and Günter Breithardt. Early and comprehensive management of atrial fibrillation: executive summary of the proceedings from the 2nd AFNET-EHRA consensus conference 'research perspectives in AF'. *European Heart Journal*, 30(24):2969–2980, 06 2009. [Cited on page 27.]

[69] Cédric Gilon, Jean-Marie Grégoire, and Hugues BERSINI. Forecast of paroxysmal atrial fibrillation using a deep neural network. pages 1–7, 07 2020. [Cited on page 27.]

[70] Elias Ebrahimzadeh, Maede Kalantari, Mohammadamin Joulani, Reza Shahraki, Farahnaz Fayaz, and Fereshteh Ahmadi. Prediction of paroxysmal atrial fibrillation: A machine learning based approach using combined feature vector and mixture of expert classification on hrv signal. *Computer Methods and Programs in Biomedicine*, 165, 08 2018. [Cited on page 27.]

[71] Zachi Attia, Peter Noseworthy, Francisco Lopez-Jimenez, Samuel Asirvatham, Abhishek Deshmukh, Bernard Gersh, Rickey Carter, Xiaoxi Yao, Alejandro Rabinstein, Brad Erickson, Suraj Kapa, and Paul Friedman. An artificial intelligence-enabled ecg algorithm for the identification of patients with atrial fibrillation during sinus rhythm: a retrospective analysis of outcome prediction. *The Lancet*, 394, 08 2019. [Cited on pages 27, 29, 39, and 60.]

[72] Rasmus Andersen, Rahman Peimankar, and Sadasivan Puthusserypady. A deep learning approach for real-time detection of atrial fibrillation. *Expert Systems with Applications*, 115, 08 2018. [Cited on page 27.]

[73] Awni Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey Tison, Codie Bourn, Mintu Turakhia, and Andrew Ng. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine*, 25, 01 2019. [Cited on page 27.]

[74] J. Duan, Q. Wang, B. Zhang, C. Liu, C. Li, and L. Wang. Accurate detection of atrial fibrillation events with r-r intervals from ecg signals. *PLoS ONE*, 17(8):e0271596, 2022. [Cited on pages 28, 29, and 40.]

[75] Zachi I Attia, Peter A Noseworthy, Francisco Lopez-Jimenez, Samuel J Asirvatham, Abhishek J Deshmukh, Bernard J Gersh, Rickey E Carter, Xiaoxi Yao, Alejandro A Rabinstein, Brad J Erickson, and et al. An artificial intelligence-enabled ecg algorithm for the identification of patients with atrial fibrillation during sinus rhythm: A retrospective analysis of outcome prediction. *The Lancet*, 394(10201):861–867, 2019. [Cited on page 28.]

[76] Xianjie Chen, Zhaoyun Cheng, Sheng Wang, Guoqing Lu, Gaojun Xv, Qianjin Liu, and Xiliang Zhu. Atrial fibrillation detection based on multi-feature extraction and convolutional neural network for processing ecg signals. *Computer Methods and Programs in Biomedicine*, 202:106009, 2021. [Cited on pages 28 and 29.]

[77] Rasmus S. Andersen, Abdolrahman Peimankar, and Sadasivan Puthusserypady. A deep learning approach for real-time detection of atrial fibrillation. *Expert Systems with Applications*, 115:465–473, 2019. [Cited on pages 28 and 29.]

[78] Jibin Wang. Automated detection of atrial fibrillation and atrial flutter in ecg signals based on convolutional and improved elman neural network. *Knowledge-Based Systems*, 193:105446, 2020. [Cited on pages 28 and 29.]

[79] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. Normal sinus rhythm database. `https://www.physionet.org/content/nsrdb/1.0.0/`, 1997. Accessed: [2023/07/08]. [Cited on page 30.]

[80] George B. Moody and Roger G. Mark. Mit-bih atrial fibrillation database. `https://www.physionet.org/content/afdb/1.0.0/`, 1997. Accessed: [2023/07/08]. [Cited on page 31.]

[81] George B. Moody and Roger G. Mark. Mit-bih arrhythmia database. `https://www.physionet.org/content/mitdb/1.0.0/`, 2009. Accessed: [2023/07/08]. [Cited on page 31.]

[82] Python Software Foundation. Python language reference, version 3.x. `https://www.python.org`, 2023. Accessed: 2023/07/09. [Cited on page 33.]

[83] PhysioNet. Wfdb software package. `https://www.physionet.org/content/wfdb/`, Current Year. Accessed: [2023/07/08]. [Cited on page 34.]

[84] Dominique Makowski, Tam Pham, Zen J. Lau, Jan C. Brammer, François Lespinasse, Hung Pham, Christopher Schölzel, and S. H. Chen. Neurokit2: A python toolbox for neurophysiological signal processing. *Behavior Research Methods*, 53(4):1689–1696, 2021. [Cited on page 35.]

[85] The PyWavelets Development Team. Pywavelets - wavelet transforms in python, 2023. Accessed: [2023/07/08]. [Cited on page 35.]

[86] The SciPy Development Team. Scipy: Open source scientific tools for python, 2023. Accessed: [2023/07/09]. [Cited on page 36.]

[87] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. [Cited on page 37.]

[88] The pandas development team. pandas - python data analysis library, 2021. Version 1.5.3. [Cited on page 38.]

[89] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, 2007. [Cited on page 38.]

[90] Oliver Faust, Murtadha Kareem, Ali Ali, Edward J. Ciaccio, and U. Rajendra Acharya. Automated arrhythmia detection based on rr intervals. *Diagnostics*, 11(8):1446, 2021. [Cited on page 40.]

[91] Qiao Xiao, Khuan Lee, Siti Aisah Mokhtar, Iskasymar Ismail, Ahmad Luqman bin Md Pauzi, Qiuxia Zhang, and Poh Ying Lim. Deep learning-based ecg arrhythmia classification: A systematic review. *Applied Sciences*, 13(8):4964, 2023. [Cited on page 44.]

[92] Devender Kumar, Abdolrahman Peimankar, Kamal Sharma, Helena Domínguez, Sadasivan Puthusserypady, and Jakob E Bardram. Deepaware: A hybrid deep learning and context-aware heuristics-based model for atrial fibrillation detection. *Comput Methods Programs Biomed*, 221:106899, 2022. [Cited on page 44.]

[93] Meinard Müller. *The Fourier Transform in a Nutshell*, pages 39–57. 08 2015. [Cited on page 46.]

[94] Mokhtar Shouran and Elmazeg Elgamli. Design and implementation of butterworth filter. *International Journal of Innovative Research in Science Engineering and Technology*, 9:7975, 09 2020. [Cited on page 47.]

[95] Chitrangi Sawant and Harishchandra T. Patii. Wavelet based ecg signal de-noising. *2014 First International Conference on Networks amp;amp; Soft Computing (ICNSC2014)*, 2014. [Cited on page 49.]

[96] Zhidong Zhao and Yefei Zhang. SQI quality evaluation mechanism of single-lead ECG signal based on simple heuristic fusion and fuzzy comprehensive evaluation. *Frontiers in Physiology*, 9, June 2018. [Cited on page 52.]

[97] Gary Tse, Mengqi Gong, Wing Tak Wong, Stamatis Georgopoulos, Konstantinos P Letsas, Vassilios S Vassiliou, Yat Sun Chan, Bryan P Yan, Sunny Hei Wong, William K K Wu, Ana Ciobanu, Guangping Li, Jayaprakash Shenthar, Ardan M Saguner, Sadeq Ali-Hasan-Al-Saegh, Aishwarya Bhardwaj, Abhishek C Sawant, Paula Whittaker, Yunlong Xia, Gan-Xin Yan, and Tong Liu. The tpeak - tend interval as an electrocardiographic risk marker of arrhythmic and mortality outcomes: A systematic review and meta-analysis. *Heart Rhythm*, 14(8):1131–1137, 2017. [Cited on page 55.]

[98] Najmeh Fayyazifar. An accurate cnn architecture for atrial fibrillation detection using neural architecture search. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 1135–1139, 2021. [Cited on page 58.]

[99] Le Sun, Yukang Wang, Jinyuan He, Haoyuan Li, Dandan Peng, and Yilin Wang. A stacked lstm for atrial fibrillation prediction based on multivariate ecgs. *Health Information Science and Systems*, 8(1):19, 2020. PMID: 32346472, PMCID: PMC7174515. [Cited on page 59.]

[100] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. [Cited on page 65.]

Appendix A

Complete Model Performances Table

Table A.1: Model Performance Metrics

Row	Met	Qua	Dur	Mod	Acc	Sen	Spe	F1
1	10-second	Manual	600	ResNet	96.0	94.8	96.4	91.4
2	10-second	Neurokit2	600	ResNet	96.0	95.3	96.6	95.9
3	10-second	Manual	600	RandomForest	95.3	93.4	95.8	89.8
4	10-second	Neurokit2	600	RandomForest	94.9	94.3	95.6	94.9
5	10-second	Neurokit2	600	CNN	94.9	94.9	95.0	94.9
6	10-second	Manual	300	ResNet	94.8	99.7	89.8	95.1
7	Peaks	Manual	600	ResNet	94.5	89.8	98.9	94.0
8	10-second	Manual	300	CNN	94.2	92.7	94.7	87.9
9	Peaks	Manual	600	RandomForest	93.1	92.2	94.0	92.8
10	Peaks	Manual	600	CNN	93.0	95.0	91.2	92.9
11	Peaks	Manual	300	CNN	93.0	91.9	94.0	92.8
12	10-second	Manual	600	CNN	92.7	91.9	93.0	84.9
13	10-second	Neurokit2	300	RandomForest	92.7	96.0	92.0	82.3
14	RRI	Manual	600	ResNet	92.5	95.1	84.2	95.0
15	10-second	Manual	60	CNN	92.5	97.4	90.6	87.7
16	10-second	Neurokit2	300	CNN	92.3	94.2	91.9	81.3
17	10-second	Neurokit2	60	CNN	91.7	96.1	90.1	86.3
18	Peaks	Manual	300	RandomForest	91.4	89.7	93.1	91.2
19	10-second	Neurokit2	300	LSTM	91.1	92.0	90.3	91.2
20	10-second	Neurokit2	600	LSTM	90.9	91.8	90.1	91.0
21	10-second	Manual	60	LSTM	90.9	92.7	89.1	91.0
22	RRI	Manual	300	ResNet	90.1	93.7	78.8	93.5
23	10-second	Neurokit2	60	RandomForest	89.7	86.4	90.9	81.9
24	Beats	Neurokit2	60	CNN	89.6	89.9	89.2	90.5
25	10-second	Manual	60	ResNet	88.9	99.8	84.9	83.2
26	10-second	Raw	60	CNN	88.7	88.5	88.8	82.6
27	10-second	Manual	60	RandomForest	88.0	81.1	90.6	78.8
28	Peaks	Manual	60	CNN	87.7	87.8	87.6	87.8
29	RRI	Manual	600	RandomForest	87.3	94.6	64.2	91.8
30	RRI	Manual	600	LSTM	86.9	78.7	95.3	85.9

Continued on next page

Row	Met	Qua	Dur	Mod	Acc	Sen	Spe	F1
31	10-second	Manual	600	LSTM	86.7	80.8	92.6	85.9
32	Beats	Manual	300	ResNet	86.5	82.9	89.7	85.4
33	Beats	Manual	300	CNN	86.0	83.5	88.2	85.1
34	10-second	Manual	300	RandomForest	86.0	84.4	87.6	85.9
35	Beats	Neurokit2	300	CNN	85.4	89.2	81.6	85.8
36	10-second	Manual	300	LSTM	85.3	90.4	80.1	86.2
37	Beats	Manual	300	RandomForest	85.3	88.5	80.9	87.5
38	RRI	Manual	600	CNN	84.4	81.2	94.7	88.8
39	RRI	Manual	60	RandomForest	84.4	91.9	57.6	90.1
40	Peaks	Manual	600	LSTM	84.3	84.8	83.8	83.8
41	Beats	Neurokit2	60	RandomForest	84.0	82.7	85.7	85.1
42	Beats	Manual	600	RandomForest	82.9	88.1	75.6	85.8
43	RRI	Manual	300	RandomForest	82.9	90.7	58.2	88.9
44	Peaks	Manual	60	RandomForest	82.9	82.0	83.7	82.8
45	Beats	Manual	600	CNN	82.8	78.2	89.3	84.2
46	Beats	Neurokit2	600	CNN	82.3	85.9	79.3	81.6
47	10-second	Raw	60	RandomForest	82.1	75.3	85.0	71.9
48	Beats	Neurokit2	300	RandomForest	82.0	89.9	75.4	81.9
49	Beats	Neurokit2	600	LSTM	81.9	88.4	75.8	82.6
50	Peaks	Manual	300	LSTM	81.3	82.5	80.2	81.4
51	10-second	Raw	600	RandomForest	81.1	72.6	83.4	62.3
52	Beats	Manual	600	LSTM	81.0	87.3	74.2	82.5
53	RRI	Manual	60	ResNet	80.6	99.9	60.4	84.1
54	10-second	Raw	600	LSTM	80.1	29.6	93.9	39.1
55	10-second	Neurokit2	300	ResNet	80.0	99.6	75.8	63.7
56	10-second	Raw	300	ResNet	80.0	27.6	87.5	25.8
57	10-second	Raw	300	RandomForest	78.6	97.0	72.9	68.0
58	Beats	Neurokit2	600	RandomForest	78.2	89.6	68.7	79.0
59	10-second	Raw	600	ResNet	77.9	38.6	93.3	49.7
60	10-second	Raw	300	LSTM	76.4	15.2	93.0	21.5
61	Beats	Neurokit2	600	ResNet	74.5	90.7	60.8	76.4
62	Beats	Neurokit2	300	ResNet	73.0	74.4	71.7	73.1
63	Beats	Manual	60	CNN	72.9	86.7	39.7	81.8
64	10-second	Neurokit2	60	ResNet	72.2	79.5	68.4	66.0
65	10-second	Raw	300	CNN	71.6	86.0	67.2	58.7
66	10-second	Raw	60	LSTM	71.4	20.7	93.5	30.5
67	Beats	Manual	60	ResNet	71.0	78.0	54.2	79.1
68	Beats	Manual	600	ResNet	70.0	98.7	29.3	79.4
69	10-second	Raw	60	ResNet	69.6	0.0	100.0	0.0
70	Peaks	Manual	60	LSTM	69.0	42.3	96.1	57.8
71	RRI	Manual	60	CNN	68.0	67.4	70.0	76.6
72	Beats	Neurokit2	300	LSTM	67.8	85.0	51.1	72.3
73	Beats	Manual	60	RandomForest	67.0	75.2	47.5	76.3
74	RRI	Manual	300	CNN	62.8	53.2	93.1	68.5
75	Beats	Manual	60	LSTM	62.0	71.6	39.0	72.7
76	Beats	Manual	300	LSTM	61.0	20.6	98.1	33.6

Continued on next page

Row	Met	Qua	Dur	Mod	Acc	Sen	Spe	F1
77	RRI	Manual	300	LSTM	55.1	99.1	10.4	69.0
78	Peaks	Manual	300	ResNet	54.3	88.8	20.4	65.8
79	10-second	Raw	600	CNN	53.6	99.1	41.1	47.9
80	RRI	Manual	60	LSTM	53.4	96.5	8.1	67.9
81	Beats	Neurokit2	60	ResNet	53.3	80.2	20.0	65.5
82	Beats	Raw	60	RandomForest	53.2	53.5	53.0	52.0
83	Beats	Raw	60	ResNet	52.2	0.8	98.3	1.7
84	Peaks	Manual	60	ResNet	50.3	81.0	19.1	62.1
85	Beats	Raw	60	LSTM	49.8	32.1	65.8	37.7
86	Beats	Raw	60	CNN	49.0	34.8	61.8	39.2
87	Beats	Neurokit2	60	LSTM	48.8	94.5	6.3	64.0