

People's Democratic Republic of Algeria  
Ministry of Higher Education and Scientific Research  
University of Mohamed Khider – BISKRA  
Faculty of Exact Sciences, Science of Nature and Life  
Computer Science Department

Order number :.....



## THESIS

Submitted in fulfilment of the requirements for the Masters degree  
in Computer Science

**Option** : information, optimization and decision system

**Title**

---

# An improved atrial fibrillation disease detection by the guided 1-D CNN

---

Presented by  
**Zineb Djihane Agli**

Defended in : dd/mm/2020

Defended before a jury composed of :

Full Name  
Salim Bitam  
Full Name

Grade  
Prof  
Grade

President  
Supervisor  
Member

Session 2020



## *Acknowledgements*

*In the name of ALLAH, the most gracious, the most merciful*

All deepest thanks are due to ALLAH, the merciful, the compassionate for the uncountable gifts given to me

I am much indebted to my supervisor **Prof. Salim Bitam** for giving me the chance to be one of his students. And whose contribution and constructive criticism has pushed me to expend the kind of efforts I have exerted to make this work as original as it can be. Thanks to him I have experienced true research and my knowledge on the subject matter has been broadened

I thank all members of the jury who agreed to review and to judge my graduation work and enrich it with their proposals

I would like to thank my advisor **Khadidja Benchaira** for her generous advices and encouragement durring caring out this work

Many thanks to all my teachers at the University of Mohamed Khider Biskra

Finally, my thanks go to who have contributed in accomplishing this work, either from near or far, may Allah grant them all the best

## *Dedication*

*This study is wholeheartedly dedicated to my beloved family.*

*To my mother who has been my source of inspiration and gave me strength when i thought of giving up, who continually provide her moral, spiritual and emotional support*

*This project is dedicated to my father May Allah be Glory to Him, saves his soul, and accept hers in his heavens*

*To my soulmate and my sister Wafa and to my brothers Youcef and Mekki Adlan*

# Abstract

The most widespread heart rhythm abnormality observed in clinical practice is atrial fibrillation (AF). AF is associated with the absence of P waves and the irregularity of RR intervals, known as among the most important heart data found by ECG signal. This disease stamps a signature in the single-lead electrocardiogram (ECG) so it represents the key for AF diagnosis, after annotation by experts. However, manual interpretation of these signals may be subjective and differs from an expert to another because many arrhythmias showcase irregular RR-intervals and lack P-waves similar to AF. On top of that, the acquired ECG signal is always tainted by noise. Hence, resilient detection of AF using the low-cost short-term, single-lead ECG is desirable but challenging.

This work proposes two solutions to improve an optimized one-dimensional convolutional neural networks (1D-CNN) model to classify four classes Normal Sinus Rhythm, AF, other rhythms, and noisy signals. The first proposal is to segment the ECG into 9 seconds records length then couple discrete wavelet transform with 1D-CNN to test the impact of preprocessing on obtained results, whereas the second proposal is the injection of precious hand-crafted features, where we give additional information to the obtained features from several convolutions layers. The idea is to merge hand-crafted features with CNN features in the fully connected layer level.

After a set of experiments, the results indicated that 1D-CNNs outperformed other deep learning approaches namely 2-D CNNs, RNNs, DNNs etc for classifying time series data and achieved delighting performances. The accuracy in the training set was 98.75% and 99.23% in the first and second propositions, respectively, where in the validation set, it was found 98.64% and 97% respectively. To experience and deploy our model, we have created a dashboard application for the benefits of doctors and workers on ECG, in which different statistics, ECG visualization, characteristics and features are provided.

# Contents

|   |          |
|---|----------|
| Acknowledgements . . . . .                                    | i        |
| Dedication . . . . .  | ii       |
| Abstract . . . . .  | iii      |
| Contents . . . . .  | iv       |
| List of Figures . . . . .                                     | viii     |
| List of Tables . . . . .                                      | xi       |
| List of Code . . . . .  | xii      |
| Acronyms . . . . .  | xiii     |
| <b>Introduction</b>   | <b>1</b> |
| <b>1 Insights into atrial fibrillation</b>                    | <b>4</b> |
| 1.1 Introduction . . . . .                                    | 4        |
| 1.2 Heart rhythm abnormalities . . . . .                      | 5        |
| 1.2.1 Definition . . . . .                                    | 5        |
| 1.2.2 Classification of arrhythmias . . . . .                 | 5        |
| 1.3 Atrial Fibrillation . . . . .                             | 7        |
| 1.3.1 What is atrial fibrillation? . . . . .                  | 7        |
| 1.3.2 What are the symptoms of atrial fibrillation? . . . . . | 8        |
| 1.3.3 Types of atrial fibrillation . . . . .                  | 9        |
| 1.3.4 What are the causes of atrial fibrillation? . . . . .   | 10       |
| 1.3.5 Risks of having atrial fibrillation . . . . .           | 11       |
| 1.4 The Electrocardiogram . . . . .                           | 12       |
| 1.4.1 Definition . . . . .                                    | 12       |

|          |  |           |
|----------|--|-----------|
| 1.4.2    | Cardiac leads . . . . .  | 12        |
| 1.4.3    | Heart rate and heart rhythm: . . . . .   | 13        |
| 1.4.4    | Characteristics of the normal heart rhythm detected by ECG . . . . .   | 14        |
| 1.5      | Atrial fibrillation signature in the ECG signal . . . . .  | 16        |
| 1.6      | e-Health to support AF management: Smart devices for a smart de-<br>tection of atrial fibrillation . . . . . | 17        |
| 1.6.1    | iBeat . . . . .  | 18        |
| 1.6.2    | KardiaMobile . . . . .   | 18        |
| 1.6.3    | Apple Watch . . . . .  | 18        |
| 1.7      | Conclusion . . . . .   | 19        |
| <b>2</b> | <b>State of the art</b>  | <b>21</b> |
| 2.1      | Introduction . . . . .   | 21        |
| 2.2      | ECG deep learning in the literature . . . . .  | 22        |
| 2.2.1    | Deep learning . . . . .  | 22        |
| 2.2.2    | Physiological Signal analysis . . . . .  | 22        |
| 2.2.3    | Physiological Signal: ECG analysis . . . . .   | 24        |
| 2.3      | Deep learning for the detection of atrial fibrillation . . . . .   | 25        |
| 2.3.1    | Deep Learning as Feature Extractor and Traditional Machine<br>Learning as Classifier . . . . .               | 25        |
| 2.3.2    | Atrial fibrillation’s Feature as input to a Neural Network . . . . .   | 25        |
| 2.3.3    | 1-D CNN to detect atrial fibrillation . . . . .  | 25        |
| 2.3.4    | 1D CNN Vs 2D CNN for atrial fibrillation detection . . . . .   | 26        |
| 2.4      | Proposed methodology . . . . .   | 28        |
| 2.5      | Conclusion . . . . .   | 28        |
| <b>3</b> | <b>An improved atrial fibrillation disease detection by the guided 1-D<br/>CNN</b>                           | <b>30</b> |
| 3.1      | Introduction . . . . .   | 30        |
| 3.2      | Global conception . . . . .  | 31        |
| 3.2.1    | ECG data collection . . . . .  | 31        |

|          |  |           |
|----------|--|-----------|
| 3.2.2    | Data length normalization . . . . .                            | 32        |
| 3.2.3    | Preprocessing . . . . .  | 32        |
| 3.2.4    | Store ECG data . . . . .                                       | 32        |
| 3.2.5    | Features selection . . . . .                                   | 32        |
| 3.2.6    | Features extraction . . . . .                                  | 33        |
| 3.2.7    | Training using the guided 1-D CNN . . . . .                    | 33        |
| 3.2.8    | Model deployment . . . . .                                     | 34        |
| 3.3      | Detailed conception . . . . .                                  | 34        |
| 3.3.1    | ECG data collection . . . . .                                  | 36        |
| 3.3.2    | Data length normalization . . . . .                            | 37        |
| 3.3.3    | Preprocessing . . . . .  | 37        |
| 3.3.4    | ECG data storing . . . . .                                     | 38        |
| 3.3.5    | Feature selection . . . . .                                    | 39        |
| 3.3.6    | Feature extraction . . . . .                                   | 39        |
| 3.3.7    | Training using the supported 1-D CNN . . . . .                 | 40        |
| 3.3.8    | Performance evaluation [13] . . . . .                          | 43        |
| 3.3.9    | Model deployment: Visualization, Interpretation, Use . . . . . | 44        |
| 3.4      | Conclusion . . . . .   | 45        |
| <b>4</b> | <b>Experimental study and results</b>                          | <b>47</b> |
| 4.1      | Introduction . . . . .   | 47        |
| 4.2      | Development tools and programming languages . . . . .          | 47        |
| 4.2.1    | C++ . . . . .  | 48        |
| 4.2.2    | Python . . . . .   | 48        |
| 4.2.3    | PyCharm . . . . .  | 48        |
| 4.2.4    | Anaconda . . . . .   | 49        |
| 4.2.5    | Flask . . . . .  | 49        |
| 4.2.6    | SQLLite . . . . .  | 49        |
| 4.3      | Implementation . . . . .                                       | 50        |
| 4.3.1    | Evaluation and used Datasets . . . . .                         | 50        |



|       |  |           |
|-------|--|-----------|
| 4.3.2 | Data length normalization . . . . .                              | 60        |
| 4.3.3 | Preprocessing and features selection . . . . .                   | 61        |
| 4.3.4 | Processing : Features extraction . . . . .                       | 64        |
| 4.3.5 | Training by the improved 1D Convolution Neural Network . . . . . | 69        |
| 4.3.6 | Training and evaluating models . . . . .                         | 75        |
| 4.3.7 | Model deployment: Visualization, Interpretation, Use . . . . .   | 77        |
| 4.4   | Results and discussion . . . . .                                 | 83        |
| 4.4.1 | Training using 1-D CNN with 30s ECG length . . . . .             | 83        |
| 4.4.2 | 1st Proposal . . . . .   | 85        |
| 4.4.3 | 2nd proposal . . . . .   | 87        |
| 4.4.4 | Discussion . . . . .   | 89        |
| 4.5   | Conclusion . . . . .   | 90        |
|       | <b>Conclusion</b>  | <b>91</b> |

# List of Figures

|      |   |    |
|------|---|----|
| 1-1  | Classification of heart rhythm abnormalities. . . . .                 | 5  |
| 1-2  | The heart is atrial fibrillation Vs in the normal case[12] . . . . .  | 7  |
| 1-3  | Prevalence of atrial fibrillation symptoms[30]. . . . .               | 8  |
| 1-4  | Vertical plane(Frontal Leads) [6]. . . . .                            | 12 |
| 1-5  | Schematic view of the precordial electrode[6]. . . . .                | 13 |
| 1-6  | Normal ECG heart beat [23]. . . . .                                   | 14 |
| 1-7  | Heart rate variability domains [5] . . . . .                          | 15 |
| 1-8  | Normal and atrial fibrillation ECG record[24]. . . . .                | 16 |
| 1-9  | Distribution of RR interval in AF (A) and normal ECG (B)[31]. . . . . | 17 |
| 1-10 | From left to right : Apple watch, Kardia mobile, iBeat . . . . .      | 18 |
|      |   |    |
| 2-1  | Deep learning Vs other classification techniques . . . . .            | 22 |
| 2-2  | Bio-signals in deep learning [44]. . . . .                            | 23 |
| 2-3  | Number of Deep Learning models used in ECG signals [44] . . . . .     | 24 |
| 2-4  | Typical 1-D CNN structure . . . . .                                   | 26 |
| 2-5  | Flatten output . . . . .  | 27 |
|      |   |    |
| 3-1  | The general architecture of the system . . . . .                      | 31 |
| 3-2  | Features extraction . . . . .   | 33 |
| 3-3  | Detailed system's architecture . . . . .                              | 35 |
| 3-4  | Noisy ECG record . . . . .  | 38 |
| 3-5  | Filtered and normalized ECG record . . . . .                          | 38 |
| 3-6  | Entity-Relationship diagram for the dashboard . . . . .               | 39 |
| 3-7  | Proposed learning architecture . . . . .                              | 42 |

|      |  |    |
|------|--|----|
| 3-8  | Model deployment: Visualisation, Interpretation, Use of our model . .                                | 45 |
| 4-1  | From left to right: Raspberry Pi3 B, The Connection bridge, The E-HP<br>[20] . . . . .               | 50 |
| 4-2  | ECG sensors plugged into the e-HP (with appropriate placements) [20]                                 | 51 |
| 4-3  | The assembled system [20] . . . . .  | 51 |
| 4-4  | ECG electrodes position [20] [22] . . . . .  | 52 |
| 4-5  | Obtained ECG values . . . . .  | 53 |
| 4-6  | The illustrative code for sending data from the Raspberry to the server.<br>(Patient side) . . . . . | 54 |
| 4-7  | The illustrative code for receiving data from the Raspberry. (Server<br>side) . . . . .              | 55 |
| 4-8  | The illustrative code for database connection . . . . .  | 56 |
| 4-9  | The illustrative code for some tables creation . . . . .   | 56 |
| 4-10 | Cooking-hacks records after saving in database . . . . .   | 57 |
| 4-11 | Example of a reading (.mat) file . . . . .   | 58 |
| 4-12 | Example of a record's (.hea) file . . . . .  | 59 |
| 4-13 | Records classes . . . . .  | 59 |
| 4-14 | Segmentation output . . . . .  | 61 |
| 4-15 | R Peaks Positions . . . . .  | 62 |
| 4-16 | Location of ECG Peaks (Peak method) where 0- T Peak 1- P Peaks 2-<br>Q Peak 3- S Peak . . . . .      | 63 |
| 4-17 | DWT method to locate beat peaks [42] . . . . .   | 63 |
| 4-18 | Normal heart rate and Atrial fibrillation's patient heart rate . . . . .                             | 64 |
| 4-19 | Heart Rate Variability features (left: Normal signal, Right: AF signal)                              | 68 |
| 4-20 | Heart Rate Variability features . . . . .  | 68 |
| 4-21 | Network Summary . . . . .  | 71 |
| 4-22 | Training CNN with denoised data . . . . .  | 72 |
| 4-23 | Proposed 02 : Network summary . . . . .  | 74 |
| 4-24 | Profile's doctor interface . . . . .   | 79 |

|      |  |    |
|------|--|----|
| 4-25 | Profile's doctor interface . . . . .   | 79 |
| 4-26 | Profile's doctor interface . . . . .   | 80 |
| 4-27 | Add patients, list and table of doctor's patients interfaces . . . . .       | 80 |
| 4-28 | Doctor dashboard . . . . .   | 81 |
| 4-29 | Appointments pages . . . . .   | 81 |
| 4-30 | Patients daily rappers . . . . .   | 82 |
| 4-31 | Dashboard for ECG reports and classification . . . . .                       | 82 |
| 4-32 | Model ECG 30s report . . . . .   | 83 |
| 4-33 | 30s ECG: model accuracy . . . . .  | 83 |
| 4-34 | 30s ECG: model loss . . . . .  | 84 |
| 4-35 | 30s ECG: model report where 0-Normal, 1-AF, 2-Others, 3-Noisy . . . . .      | 84 |
| 4-36 | First fold results . . . . .   | 85 |
| 4-37 | 1st Proposal: model accuracy . . . . .                                       | 85 |
| 4-38 | 1st Proposal: model loss . . . . .   | 86 |
| 4-39 | 1st Proposal: model report where 0-Normal, 1-AF, 2-Others, 3-Noisy . . . . . | 86 |
| 4-40 | First fold results . . . . .   | 87 |
| 4-41 | 2nd proposal: model accuracy . . . . .                                       | 87 |
| 4-42 | 2nd proposal: model loss . . . . .   | 88 |
| 4-43 | 2nd proposal: model report where 0-Normal, 1-AF, 2-Others, 3-Noisy . . . . . | 88 |

# List of Tables

|     |   |    |
|-----|---|----|
| 1.1 | Patterns of atrial fibrillation[28]. . . . .                        | 9  |
| 1.2 | Normal heart beat waves and intervals values [23]. . . . .          | 14 |
| 1.3 | Characteristics of ECG signal in atrial fibrillation [11] . . . . . | 16 |
| 3.1 | Detailed 1-D CNN architecture. . . . .                              | 40 |
| 4.1 | Data profile for the Dataset. . . . .                               | 58 |
| 4.2 | ECG records classes . . . . .                                       | 59 |
| 4.3 | Data profile for the database after segmentation . . . . .          | 61 |
| 4.4 | Description of time domain features . . . . .                       | 65 |
| 4.5 | Description of frequency domain features . . . . .                  | 66 |
| 4.6 | Resume of the results obtained in the validation set . . . . .      | 89 |

# List of Code

|   |    |
|---|----|
| ardu.m . . . . .  | 51 |
| ECGcooking.m . . . . .  | 52 |
| normalization.m . . . . .   | 54 |
| 4.1 Applying the segmentation algorithm on records . . . . .  | 60 |
| 4.2 Illustrative code of extracting time domain features . . . . .                                      | 65 |
| 4.3 Source code of extracting frequency domain features . . . . .                                       | 66 |
| 4.4 Illustrative code of extracting all heart rate variability features from<br>an ECG signal . . . . . | 67 |
| 4.5 Implementation of the 1-D CNN . . . . .   | 69 |
| 4.6 Implementation of the 1-D CNN . . . . .   | 69 |
| 4.7 Denoised and normalized signal as input . . . . .   | 72 |
| 4.8 Proposed 02 : Merged data . . . . .   | 73 |
| 4.9 Train and evaluate models . . . . .   | 75 |
| 4.10 Train and evaluate 2nd proposal model . . . . .  | 76 |
| 4.11 Load .h5 model in flask . . . . .  | 77 |
| 4.12 Deploy model in flask . . . . .  | 77 |
| results.m . . . . .   | 78 |

# Acronyms

**AF** Atrial fibrillation

**AFib** Atrial fibrillation

**ANN** Artificial Neural Network

**CVD** Cardiovascular disease

**DWT** Discrete wavelet transform

**ECG** Electrocardiogram

**e-HP** e-Health Platform

**HRV** Heart Rate Variability

**RPi3** Raspberry Pi3

**SVM** Support Vector Machine

**1-D CNN** 1 Dimensional Convolutional Neural Network

# Introduction

According to the World Health Organization (WHO), cardiovascular disease(CVD) claims the lives of 17.9 million people each year, or 31% of all deaths around the world.

One of these heart conditions is known as atrial fibrillation (AF) which is an exceedingly common rhythm disturbance in elderly patients, that is associated with significant morbidity and mortality. Hence, early and accurate detection of this disease can improve the prevention of severe complications such as stroke.

Nearly 25% of all individuals with AF are asymptomatic. In these individuals, screening with an electrocardiogram (ECG) may be the essence of diagnosis. The ECG records represent the electrical activity of the heart and in atrial fibrillation, these signals have a specific form that differs in the other heart situations.

Some wearable ECG cardiac monitoring devices are designed to ensure delicate AF detection, we mention Android Wear, Apple Watch, Kardia... Unfortunately, these devices are expensive, time-consuming, complicated, and require long-term exposure for AF measurement.

Currently, short-term ECG signal detection is prevalent in daily application. Thus, a simple algorithm for improving short-term AF detection with satisfactory results is desirable. At the same time, a simple acquisition of the ECG is insufficient because atrial fibrillation requires a permanent recording signal, which allows full-time monitoring of AF patients.

This Master project proposes the design and the development of a smart system to aid in the early detection of atrial fibrillation disease using short-term ECG signal and real-time monitoring of patients based on a continuous recording of ECG signal. In order to validate these proposals, we are going to develop a connected health



platform (Connected Health) which uses the Cooking hacks ECG acquisition device, and which is equipped with a Raspberry PI 3 type nano-computer.

This dissertation is composed of four chapters, this introduction, and a general conclusion, it is organized as follows: The first chapter is devoted to the basic conceptions of atrial fibrillation where we try to explain this disease using a simple scientific language, its signs and symptoms, risk factors, and its diagnosis where we explain how the ECG signal acts in AF.

The second chapter reviews the latest yet the most effective works about the detection of AF disease where they use the ECG signal.

In the third chapter, we expose our proposal ideas in general and in a detailed way. Where we improve an optimized 1-D CNN using segmentation and denoising then using a combination of relevant features and output CNN features.

In the last chapter, we illustrate the experimental study by explaining the different stages allowing the implementation of our project, the tests carried out, and the obtained results.

The thesis ends with a general conclusion containing our contemplated prospects.

# Chapter 01

## Insights into atrial fibrillation

# Chapter 1

## Insights into atrial fibrillation

### 1.1 Introduction

Cardiovascular diseases (CVDs) are disorders of the heart and blood vessels[48], which represent the leading cause of death worldwide. Atrial fibrillation (AF or AFib) is recognized as a major cardiovascular disease, involving a large number of the population. AF is an arrhythmia that affects mostly elderly people, in particular those who suffer from heart failure (one of the main causes of hospitalization). It is also associated with increased risks of cardiovascular events, reducing the life quality of AF patients, or even causing mortality. AFib can be managed with a doctor's care and medication, and early diagnosis and treatment can prevent such complications.

This chapter presents a detailed overview of atrial fibrillation, its impact on patients, and how an early diagnosis can prevent serious consequences such as sudden death. Also, we spotlight in the electrocardiogram (ECG) which represents the electrical activity of the heart and its uses for atrial fibrillation detection.

## 1.2 Heart rhythm abnormalities

In this section, we outline the different types of arrhythmias which are one of the most complexes, insufficiently studied, and therefore one of the most urgent problems of modern cardiology.

### 1.2.1 Definition

Heart rhythm abnormalities (arrhythmias) are sequences of irregular heartbeats that can cause the heart rate (i.e. number of times per minute that heartbeats ) to be too slow or too fast or irregular[15].

### 1.2.2 Classification of arrhythmias

The following diagram summarizes the heart rhythm abnormalities that can be detected by medical diagnosis.

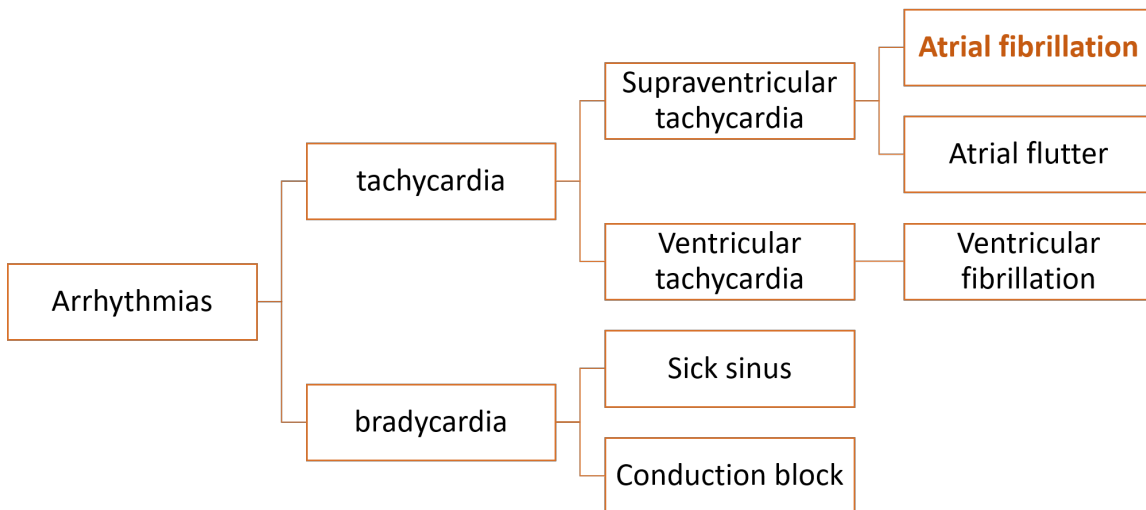


Figure 1-1: Classification of heart rhythm abnormalities.

It is appropriate to subdivide cardiac arrhythmias into the following groups:

## 1. **Bradyarrhythmias (bradycardia): Slow Heart Rate**

Bradycardia is the case when the heart rate is too slow. What is considered too slow, can depend on the age and physical condition. Elderly people, for example, are more prone to bradycardia. [11].

This type of arrhythmias is characterized by a resting heart rate inferior to 60 beats/minute for adults, in other words, a resting heart rate of fewer than 60 beats per minute (BPM) qualifies as bradycardia.

## 2. **Tachyarrhythmias (Tachycardia): Fast Heart Rate** It refers to a fast heart rate. Generally speaking, for adults, a heart rate of more than 100 beats per minute (BPM) is considered too fast.

- (a) **Ventricular Tachycardia:** is a fast heart rate that occurs in the lower chambers (ventricles) of the heart. This type of arrhythmia may be either well-tolerated, requiring immediate diagnosis and treatment[11].

The seriousness depends largely on whether other cardiac dysfunction is present and on the degree of the ventricular tachycardia.

- (b) **Supraventricular Tachycardia (SVT):** Atrial or supraventricular tachycardia (SVT) is the most common type of abnormal tachycardia in adults. with a fast heart rate that starts in the upper chambers of the heart. In atrial or supraventricular tachycardia, electrical signals in the heart's upper chambers fire abnormally.

SVTs are usually identified by an explosion of rapid heartbeats that can be chronic or begin and end suddenly. It can last a few seconds or several hours and may cause the heart to beat over than 160 times per minute. Symptoms include palpitations, chest pains, upset stomach, decreased appetite, lightheadedness or weakness.

SVTs often include atrial fibrillation and atrial flutter.

## 1.3 Atrial Fibrillation

### 1.3.1 What is atrial fibrillation?

By the American College of Cardiology (ACC), the American Heart Association (AHA) and the European Society of Cardiology (ESC) atrial fibrillation is defined as “*Tachyarrhythmia characterized by mostly uncoordinated atrial activation with consequent deterioration of atrial mechanical function*”. It is the most common cardiac arrhythmia, occurring in 1-2% of the general population” [26].

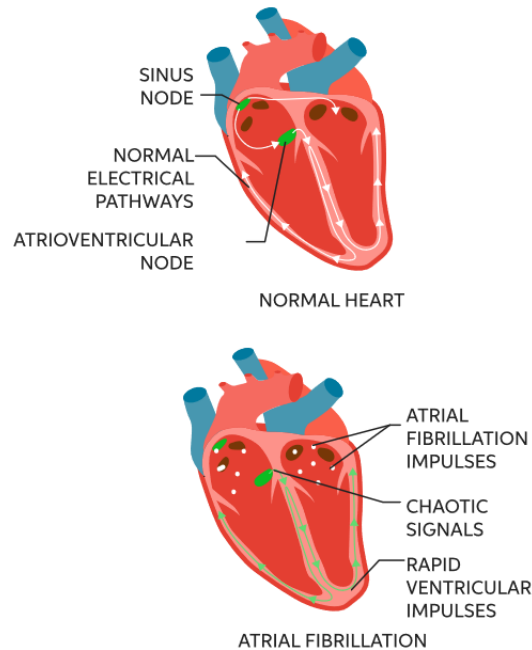


Figure 1-2: The heart is atrial fibrillation Vs in the normal case[12]

Atrial fibrillation is a rapid heart rate caused by irregular electrical impulses in the upper chambers of the heart. These signals result in rapid, uncoordinated, weak contractions of the atria. While the heart should pump blood properly so the body gets the oxygen it needs; **What happens in** atrial fibrillation (AF), the heart does not beat appropriately rather than pulsing in a normal pattern, the atria quiver in irregular and fast way[2] .

### 1.3.2 What are the symptoms of atrial fibrillation?

Approximately 25% of all individuals with AF are asymptomatic (they have no symptoms)[11] and insensible of their condition until it is often discovered first during a hospital admission or a physical examination. Those who do have atrial fibrillation often experience debilitating symptoms despite treatment such as:

- Heart palpitations, which are sensations of a racing, uncomfortable, irregular heartbeat or a flip-flopping in the chest.
- Feeling faint at times.
- Dyspnoea (i.e. Shortness of breath).
- Chest pain.
- Fatigue.
- Sleeping difficulties.

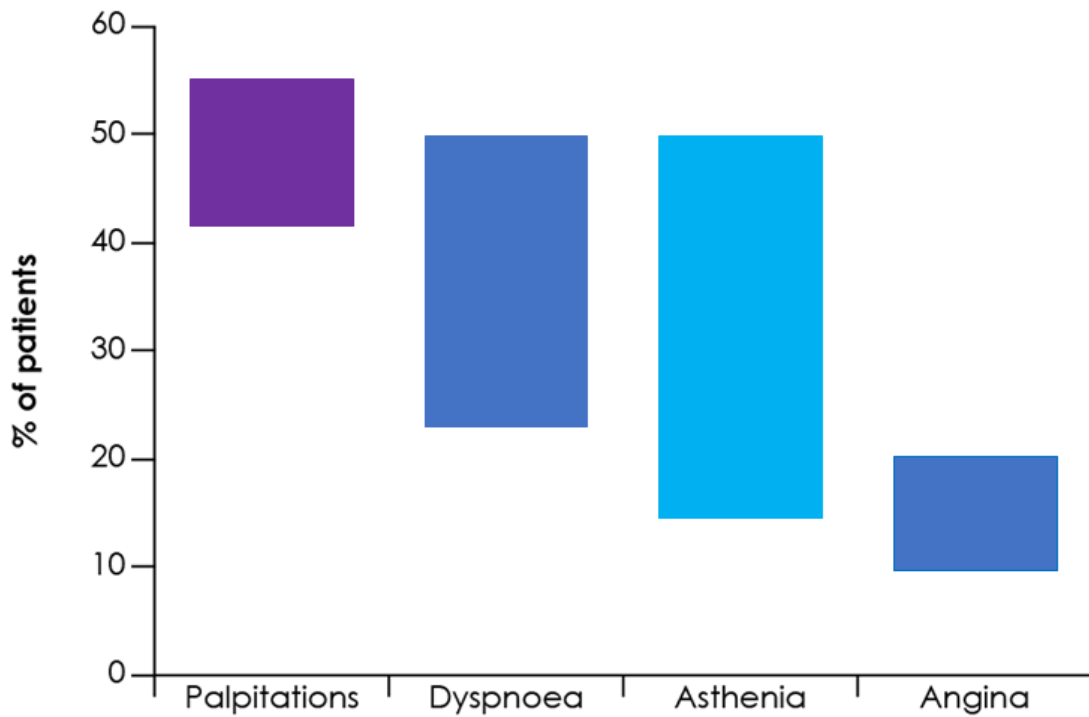


Figure 1-3: Prevalence of atrial fibrillation symptoms[30].

### 1.3.3 Types of atrial fibrillation

Atrial fibrillation (AF) can present in several ways and correct classification can guide the choice of treatment. It is grouped according to the duration of the arrhythmia.

The table below outlines the different classes of AF

| AF Pattern                      | Definition   |
|---------------------------------|--|
| <b>First diagnosed</b>          | Not previously diagnosed, irrespective of arrhythmia duration or the presence and severity of symptoms |
| <b>Paroxysmal</b>               | Self-terminating, usually within 48h, may continue for up to 7 days.                                   |
| <b>Persistent</b>               | Lasts more than 7 days   |
| <b>Long-standing persistent</b> | Continuous AF for more than 1 year with rhythm control   |
| <b>Permenent</b>                | The patient and physician accepts the AF, and the rhythm is not controlled                             |

Table 1.1: Patterns of atrial fibrillation[28].

Atrial fibrillation is a cumulative disease that usually evolves towards permanent atrial fibrillation. This is generally a stepwise process in which persons with paroxysmal atrial fibrillation head for an increasing number of episodes until the arrhythmia is persistent. Once persistent, the number of episodes with persistent atrial fibrillation tends to increase until the arrhythmia is long-standing persistent. It should be noted, however, that some patients have paroxysmal or persistent atrial fibrillation during their disease course, while others never return to sinus rhythm after a first diagnosis[7].

#### Other types of atrial fibrillation

- **Lone atrial fibrillation** is used to describe a patient younger than 60 years of age, who do not have any other concomitant heart diseases or risk factors, and whose echocardiographic examination is normal. This type of atrial fibrillation has a good prognosis and generally do not require anticoagulation therapy[49].



- **valvular atrial fibrillation** affects people who have valve disease or an artificial valve.
- **Non valvular atrial fibrillation** is caused by other things, such as high blood pressure.

### 1.3.4 What are the causes of atrial fibrillation?

A risk factor is something that increases the risk of developing a disease or condition. The main risk factors for getting atrial fibrillation are abnormalities or damage to the heart's structure are the most common cause of atrial fibrillation[2].

Possible causes of atrial fibrillation include

1. getting older, particularly being 60 or older.
2. High blood pressure.
3. diabetes.
4. lung cancer.
5. Heart attack.
6. Congenital heart defects.
7. Exposure to stimulants, such as medications, caffeine, tobacco or alcohol.
8. Sick sinus syndrome: improper functioning of the heart's natural pacemaker.
9. History of previous heart surgeries...

Atrial fibrillation is common in people with other heart conditions, also with other medical conditions. However, in lone atrial fibrillation, the cause is often unclear, and serious complications are rare where patients don't have any heart defects or damage.

## **1.3.5 Risks of having atrial fibrillation**

AFib can cause potentially life-threatening health issues, We review the following

### **1.3.5.1 Blood clots**

Having atrial fibrillation increases the risk of developing a blood clot inside the chambers of the heart. This is because the atrial fibrillation disturbs the normal flow of blood through the heart, causing turbulence. The turbulence causes the blood to form small clots. If a clot forms in the heart, it can travel through your bloodstream and cause a stroke.

### **1.3.5.2 Stroke**

The relationship between AF and stroke runs both ways. On the one hand, AF markedly increases the risk of stroke[45]. For instance, AF increases stroke risk five-fold compared with people without the arrhythmia[50],[45]. On the other hand, strokes increase the likelihood of developing AF[29]. For instance, older people are at increased risk: 25% of all stroke in people older than 80 years occur in AF patients[50]. Women are also at higher risk of experiencing a stroke from AF compared with men[45].

About 50% of people die within a year of a atrial fibrillation (AF)-related stroke. This compares with a mortality rate of 27% among people with strokes unrelated to AF[40]. A study from Ireland reported five-year survival rates after an AF-related stroke of 39.2%.

### **1.3.5.3 Heart failure**

AF can lead to heart failure, especially when the heart rate is high.

Atrial fibrillation can be diagnosed medically using the electrocardiogram(ECG) test. In the next sections, we explain what an ECG is and how it helps to detect this arrhythmia.

## 1.4 The Electrocardiogram

### 1.4.1 Definition

An electrocardiogram (EKG, ECG) is a painless process that records the heart's electrical activity [15], where small electrodes are placed on the skin of the chest wrists, ankles and connected in a specific order to a machine that, when turned on, this machine measures electrical activity all over the heart and transforms the signals into patterns or waves [11].

The ECG can help to diagnose the heart rhythm abnormalities such as atrial fibrillation.

### 1.4.2 Cardiac leads

The ECG is made up of 12 standard leads paints a complete picture of the heart's electrical activity [11].

a **Vertical plane (Frontal Leads):** They can be divided into two parts, the bipolar leads DI, DII and DIII and the monopolar leads aVR, aVL and aVF. To obtain this derivations, the electrodes are placed on the right and left arms and on the leg left to form a triangle (Einthoven triangle), and for monopolar derivations the aVR derivative uses the right arm as positive and all the others electrodes of the limbs like an earth (negative common). The other two derivations of the limbs, aVL and aVF, are obtained analogically[18].

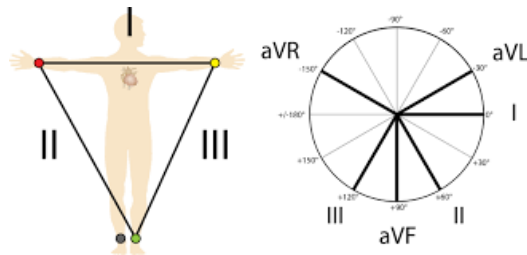


Figure 1-4: Vertical plane(Frontal Leads) [6].

b **Horizontal Plane (Transverse Leads)[19]** Chest (Precordial) Electrodes and Placement:

- V1: Fourth intercostal space on the right sternum.
- V2: Fourth intercostal space at the left sternum.
- V3: Midway between placement of V2 and V4.
- V4: Fifth intercostal space at the midclavicular line.
- V5: Anterior axillary line on the same horizontal level as V4.
- V6: Mid-axillary line on the same horizontal level as V4 and V5.

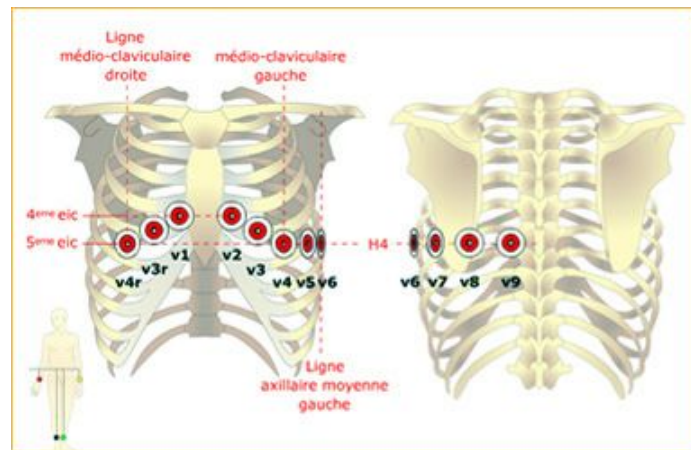


Figure 1-5: Schematic view of the precordial electrode[6].

### 1.4.3 Heart rate and heart rhythm:

- **Heart rate**

Heart rate is the number of times the heart beats in a minute. This is the number of times it pumps to push blood round the body.

- **Heart rhythm**

Heart rhythm is the pattern in which the heart beats. It may be described as regular or irregular, or fast or slow.

### 1.4.4 Characteristics of the normal heart rhythm detected by ECG

We can divide ECG characteristics into two big categories:

1. **Morphological features:** ECG interpretation includes an assessment of the morphology (appearance) of the waves and intervals on the ECG curve. Thus, Before discussing each component in detail, a brief overview of the waves and intervals is given (View Figure 1-6)

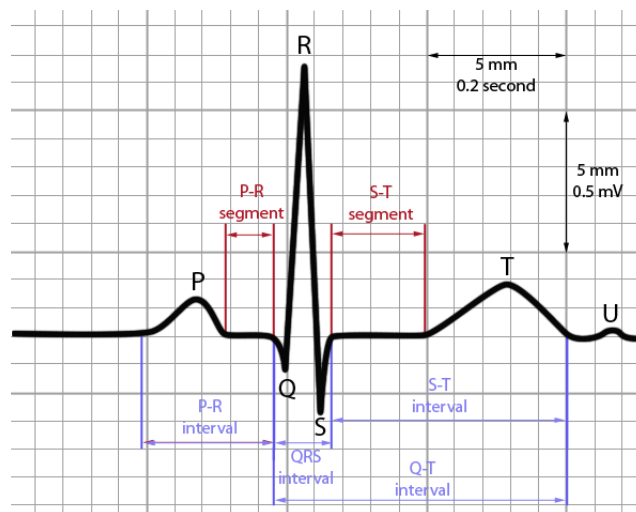


Figure 1-6: Normal ECG heart beat [23].

The values of the parameters in Figure1-6 generally observed in adults in good health are shown in the table bellow:

|                       | P wave      | PQ interval | QRS Complex           | ST interval | QT interval | T wave  |
|-----------------------|-------------|-------------|-----------------------|-------------|-------------|---------|
| <b>Duration (ms)</b>  | 0.08 - 0.10 | 0.12 - 0.20 | 0.08                  | 0.20        | 0.36        | 0.20    |
| <b>Amplitude (mv)</b> | 0.25        | 0           | $Q < 0, R > 0, S < 0$ | 0           | -           | $T > 0$ |

Table 1.2: Normal heart beat waves and intervals values [23].

2. **Heart rate variability features** Heart rate variability or HRV is the physiological phenomenon of the variation in the time interval between consecutive heartbeats in milliseconds. A normal healthy heart, when looking at the milliseconds between heartbeats, there is a constant variation.

One of the main divisions for the way in which HRV is calculated is by either using time or frequency.

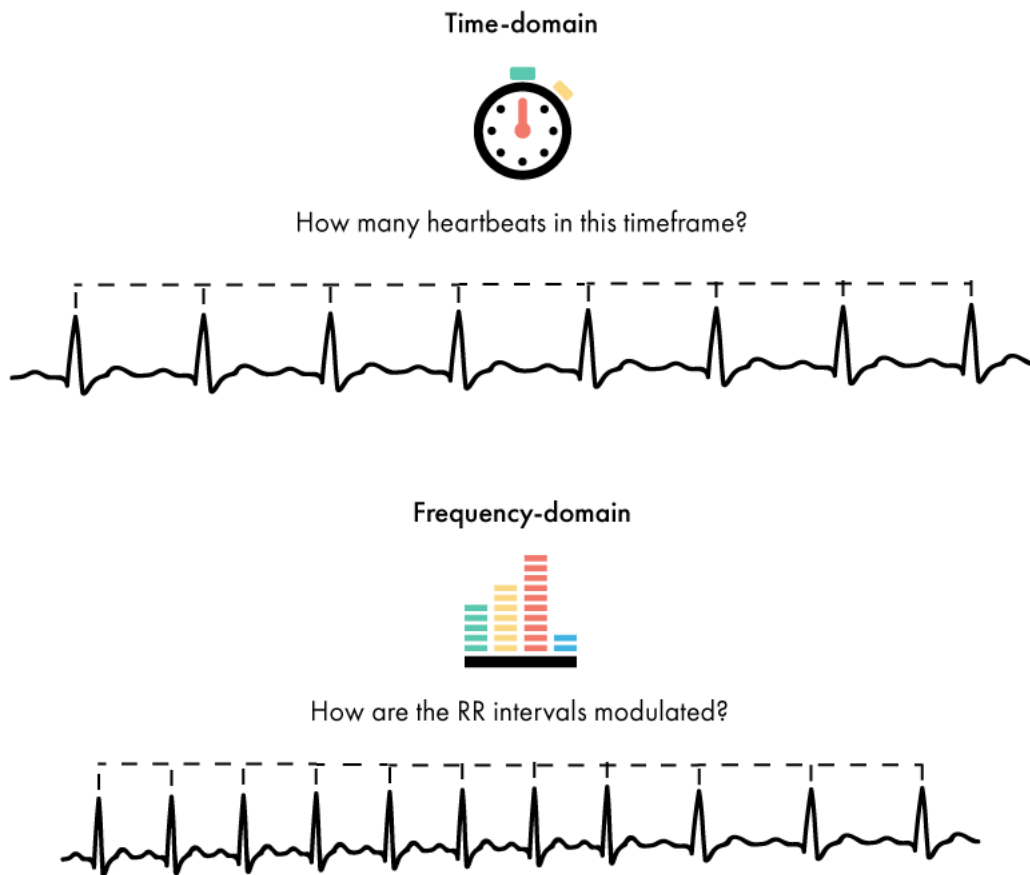


Figure 1-7: Heart rate variability domains [5]

## 1.5 Atrial fibrillation signature in the ECG signal

Detection and diagnosis of AF is done non-invasively in clinical environments through the evaluation of ECG.

Key characteristics of AF in the ECG trace include :

|                     |   |
|---------------------|---|
| <b>Heart rhythm</b> | Irregular heart rhythm (irregular RR interval)  |
| <b>Heart rate</b>   | Ventricular rate : 60-100 beat<br>Atria rate : 400 - 600 beat<br>Average heart rate : $\succ 100$ if AF is uncontrolled |
| <b>P wave</b>       | No P waves  |
| <b>PR intervals</b> | No PRi intervals since there are no P waves   |
| <b>QRS complex</b>  | 0.06 - 0.10seconds  |

Table 1.3: Characteristics of ECG signal in atrial fibrillation [11]

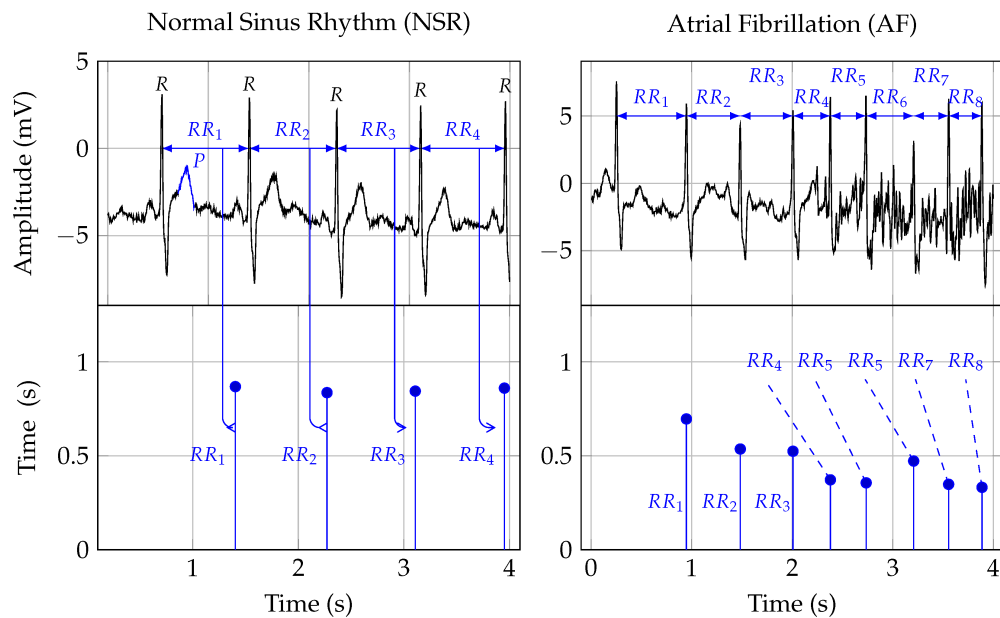


Figure 1-8: Normal and atrial fibrillation ECG record[24].

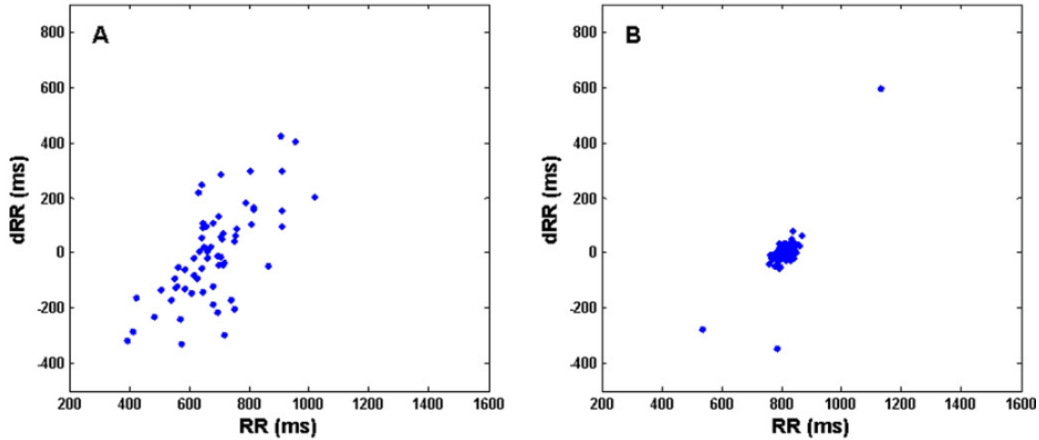


Figure 1-9: Distribution of RR interval in AF (A) and normal ECG (B)[31].

## 1.6 e-Health to support AF management: Smart devices for a smart detection of atrial fibrillation

Technological advances in all fields are exciting, particularly in the medical field. New electronic devices like smart phones and smart watches can perform other tasks for us such as assessing our daily steps, walked distance, sleeping hours, calories burned or even tell our heart rate. However, the true expectation for these technological advances is that they may help us to diagnose diseases or detect abnormalities and so we can treat them quickly and prevent complications. This is particularly necessary in atrial fibrillation, a highly prevalent entity with potentially devastating complications[17].

The most important limitation is that AFib usually requires a prolonged monitoring for its diagnosis. Therefore, smart devices with accurate monitoring capabilities might resolve this limitation.

In this section, we present some smart devices for AF detection:



### 1.6.1 iBeat

iBeat is a smartwatch that prevents heart-related conditions by monitoring heart rate. It was designed to save lives and specifically reduce the hundred thousands who died of heart conditions annually. iBeat is great for cardiovascular disease prevention because it detects symptoms of heart attack and other cardiac problems [33].

### 1.6.2 KardiaMobile

KardiaMobile captures a medical-grade ECG in 30 seconds anywhere, anytime, it is used by the world’s leading cardiac care medical professionals and patients to detect Atrial Fibrillation, Bradycardia, Tachycardia or Normal heart rhythm [14].

### 1.6.3 Apple Watch

Apple Watch identifies unusually low or high heart rates and irregular heart rhythms, and the smartwatch notifies even if patient don’t feel the symptoms[16]. In addition, this device tracks activity, blood pressure, and sleep, and it gives a report about patients activity level.



Figure 1-10: From left to right : Apple watch, Kardia mobile, iBeat

## 1.7 Conclusion

Despite good progress in the management of patients with atrial fibrillation (AF), this arrhythmia remains one of the major causes of stroke, heart failure, sudden death, and cardiovascular morbidity in the world. Also, it is associated with poor quality of life and adverse symptoms. Moreover, the number of patients with AF is expected to rise steeply in the coming years. An early diagnosis of atrial fibrillation requires rhythm documentation using an electrocardiogram (ECG) therefore it could reduce the risk of such complications.

In the next chapter, we present the latest techniques and methods used to face this disease using ECG.

# Chapter 02

## State of the art

# Chapter 2

## State of the art

### 2.1 Introduction

Twentieth-century was characterized as the century of information. Loads of data were collected and information was exchanged in several aspects. Information technology had an explosive evolution developing a number of methods to handle and process this data. Knowledge Discovery Databases (KDD), Machine Learning (ML), Pattern Recognition (PR), Data Mining (DM) etc. are only some of the uprising methods developed for data processing and knowledge extraction that can be used for automated decision making. Like in every other field, implementation was tested in medical data in cases where diagnosis and prognosis can have a major effect on human's health.

In order to improve detection and classification of ECG signals that leads to atrial fibrillation recognition, several analysis methods based on conventional machine learning have been developed. In this chapter, this field of research is presented by exposing a number of applications from different algorithms. The evolution of applications is recorded concluding on the latest trend which is the real-time diagnosis through portable devices.

## 2.2 ECG deep learning in the literature

### 2.2.1 Deep learning

Deep Learning has succeeded over traditional machine learning in the field of medical imaging analysis, due to its unique ability to learn features from raw data. Objects of interest in medical imaging such as lesions, organs, heart anomalies and tumors are very complex, and much time, experts and effort is required to extract features using traditional machine learning, which is accomplished manually[44]. Thus, deep learning in medical imaging replaces hand-crafted feature extraction by learning from raw input data, feeding into several hidden layers, and finally outputting the result from a huge number of parameters in an end-to-end learning manner. Therefore, many research works have benefited from this novel approach to apply physiological data to fulfil medical tasks.

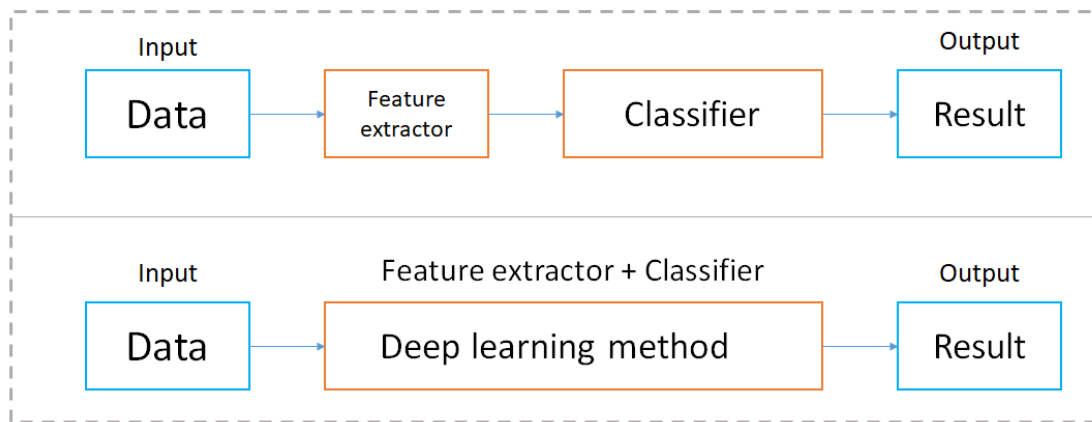


Figure 2-1: Deep learning Vs other classification techniques

### 2.2.2 Physiological Signal analysis

Physiological signal data in the form of 1D signals are time-domain data, in which sample data points are recorded over a period of time. These signals change continuously and indicate the health of a human body.

Physiological signal data categories fall into characteristics such as :

- **Electromyogram (EMG)**: which is data regarding changes to skeleton muscles
- **Electrocardiogram (ECG)**: which is data regarding changes to heart beat or rhythm
- **Electroencephalogram (EEG)**: which is data regarding changes to the brain measured from the scalp
- **electrooculogram (EOG)**: which is data regarding changes to corneo-retinal potential between the front and the back of the human eye.

The authors of [44] survey 147 contributions of physiological signal analysis.

They collected papers via search engine PubMed with keywords combining “deep learning” and a type of physiological signal such as “deep learning electrocardiogram ECG” and they found 147 papers published between January 2018 and October 2019 inclusive from various journals and publishers.

The Figure2-2 illustrates that the works on EMG, ECG, and EEG using a deep-learning approach have rapidly increased in 2018 and 2019.

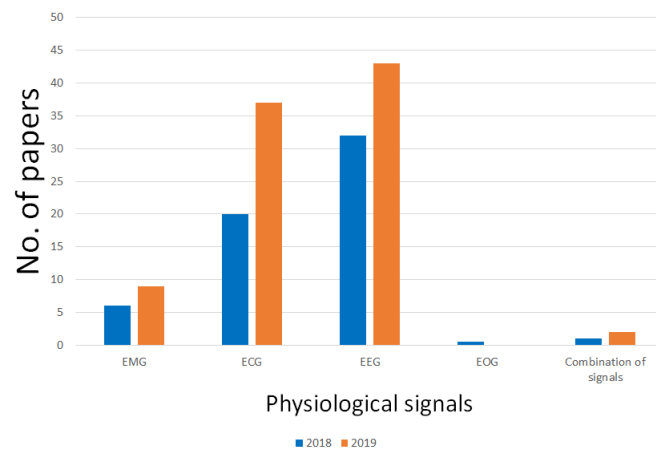


Figure 2-2: Bio-signals in deep learning [44].

### 2.2.3 Physiological Signal: ECG analysis

As atrial fibrillation is a heart disease, we are interested by using the ECG which is data regarding changes of heartbeat or rhythm.

The authors of [44] collected 47 different robust research works using deep-learning methods to analyze the ECG signals. Statistics showed that the most used method to analyse ECG signals is Convolutional Neural Networks (CNN). (View Figure 2-3)

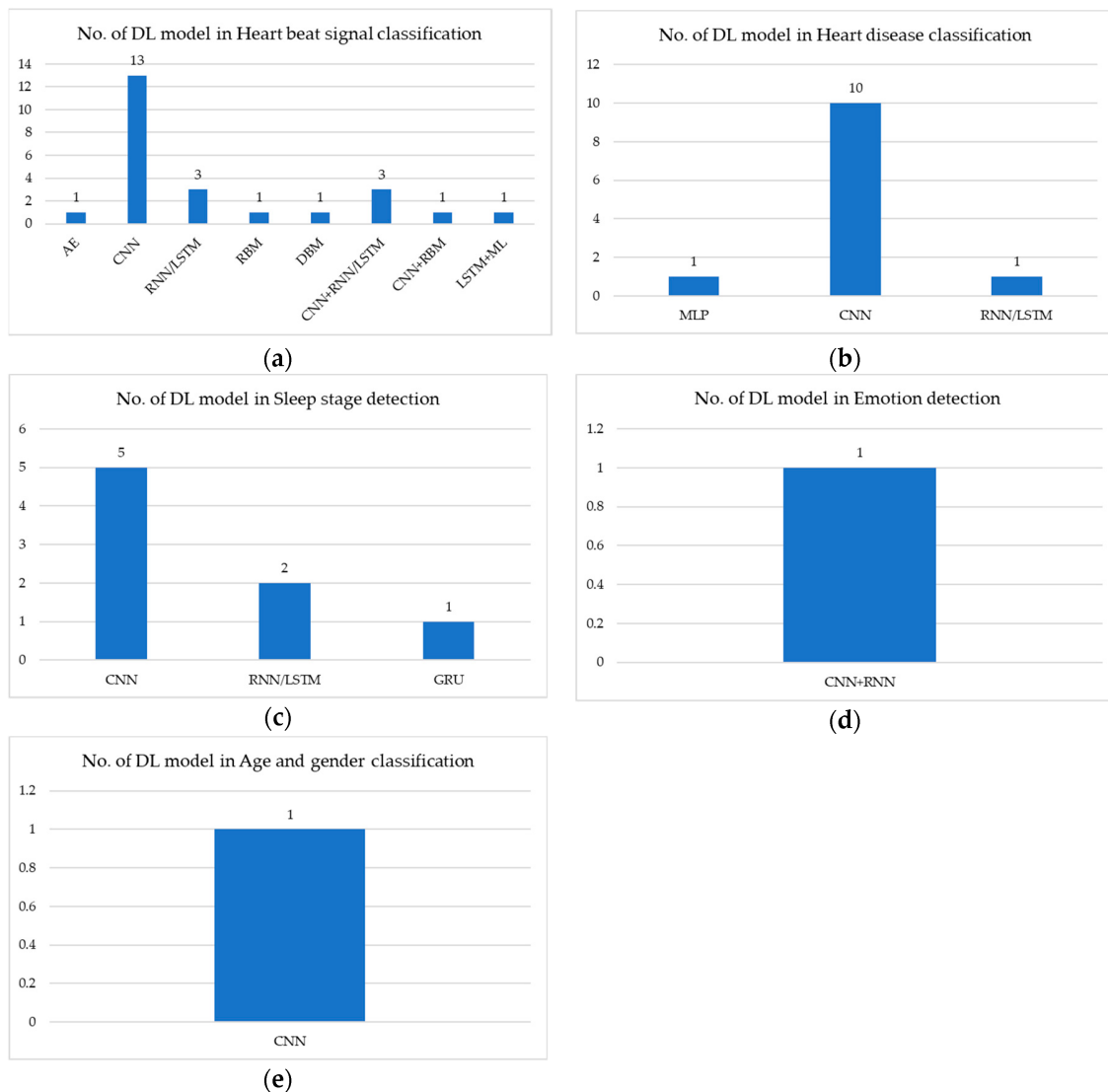


Figure 2-3: Number of Deep Learning models used in ECG signals [44]

## **2.3 Deep learning for the detection of atrial fibrillation**

In this section, we discuss in detail the approaches that closely related to our work, which are widely used to detect AF.

### **2.3.1 Deep Learning as Feature Extractor and Traditional Machine Learning as Classifier**

The authors of [46] proposed a method to combine time and frequency characteristics of the ECG signal by applying the short-time Fourier transform and visually representing the data as a spectrogram. Then they pass the spectrogram to a DenseNet model to extract features that were then classified using SVM with an overall accuracy of 93.16%.

### **2.3.2 Atrial fibrillation’s Feature as input to a Neural Network**

ECG signal in atrial fibrillation disease has several characteristics that we already mentioned in the first chapter. Thus many papers use ECG features for detection. As proposed in [32] where the authors retrieved RR intervals features and they fed them as input to an ANN model.

### **2.3.3 1-D CNN to detect atrial fibrillation**

In another study [27], the authors developed an end-to-end 1D CNN for the AF detection from time-series ECG data. By developing a pre-processing algorithm to make each recording fixed in length. The dataset they used is provided by PhysioNet Challenge 2017, containing 8528 single lead variable-length ECG recordings. The proposed architecture contains 10 convolutional blocks, 2 fully-connected layers, and a Softmax layer as the output prediction.



### 2.3.4 1D CNN Vs 2D CNN for atrial fibrillation detection

Convolutional neural network (CNN) is the most successful type of deep-learning model for 2D image analysis such as recognition, classification, and prediction. CNN receives 2D data as an input and extracts high-level features through many hidden convolution layers. Thus, to feed physiological signals into a CNN model, some research works have converted 1D signals into 2D data.

In the 1D scheme, the time-series 1D data is directly fed into the Neural Network without any transformation. On the contrary, for the 2D scheme, the time-series signal is converted into a 2D form (i.e., a spectrogram) before inputting the subsequent neural network.

#### 2.3.4.1 1-D CNN architecture:

Typically, a 1-D CNN model involves two kinds of layers (as shown in Figure 2-4) Convolution layers and Multilayer Feed Forward (MLFF) layers.

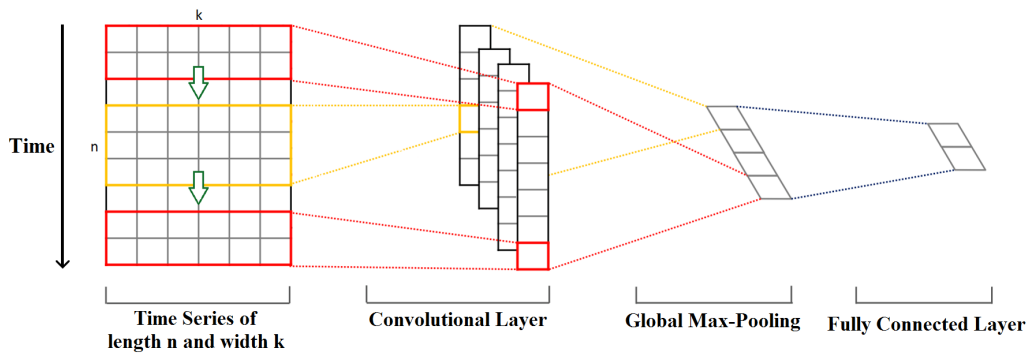


Figure 2-4: Typical 1-D CNN structure

1. **Convolution layers:** Convolution involves sliding the kernel over the input signal which is also known as shiftcompute procedure.
2. **Pooling layer** Pooling is used to reduce the dimensionality of a given mapping while highlighting the prominent features. Generally, pooling is employed after the convolution layer to reduce the dimension of the convolution output.

3. **Flatten** The output of convolution layers may have a depth greater than one. Flatten concatenates the output from the convolution layers to form a flat structure which can then be fed as input to Multilayer Feed Forward Network (MLFF) (Figure 2-5 (a))

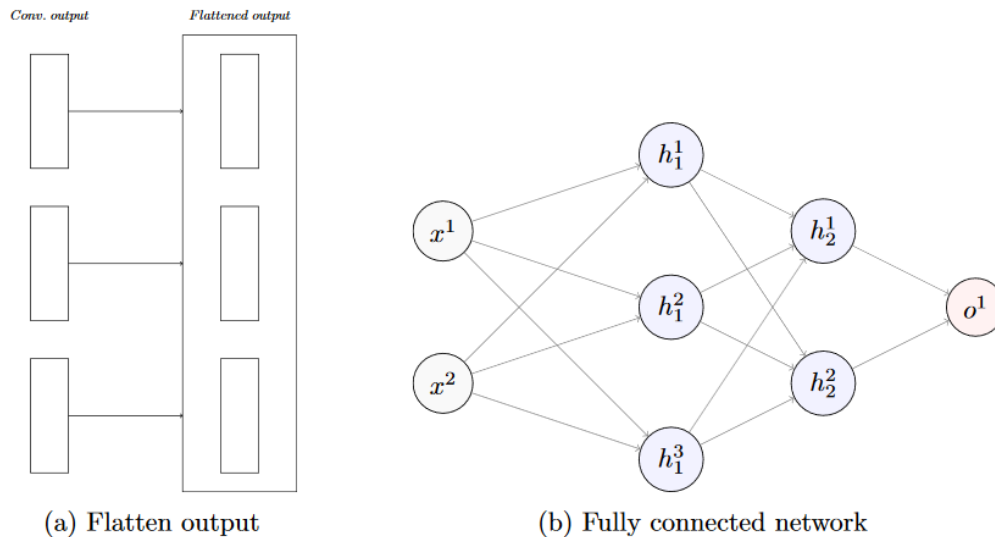


Figure 2-5: Flatten output

4. **Multilayer Feed Forward Network** MLFF or fully connected layer is a structure where neurons from a layer is connected to all the neurons in the next layer. MLFF accepts flattened output from convolution layers and maps it to the output (figure 2-5 (b))
5. **Output Layer** Most used format to represent a classification output is Onehot encoding format. For a Nclass problem, the output is a vector of dimension N. Each element of this output vector can only attain a value of either 0 (ON) or 1 (OFF). And only one element of the vector can be "ON" at a time. For example, consider a four classes problem. The four classes are represented by

Class 1: [1 0 0 0]

Class 2: [0 1 0 0]

Class 3: [0 0 1 0]

Class 4: [0 0 0 1]

## 2.4 Proposed methodology

In the previous sections, we tried to cover the most modern methods to detect atrial fibrillation. We have seen that using 1-D CNN as feature extractor from the raw data is a great detection tool, at the same time we cannot underestimate experts knowledge; (we are talking here about the hand-crafted features, in a fancier way, we cannot ignore the signs of atrial fibrillation on ECG signal. Moreover, and in the best of our knowledge, there are no research or technology allowing permanent recording of ECG, which is a requirement for specialists in Cardiology to discover AFib especially if it occurred when patient sleeping or in similar situations.

Thus, in this master project we propose :

- An improvement of a 1-D CNN architecture from [27] by using ECG length normalization, noise reduction technique and by injecting precious hand-crafted features.
- A full-time ECG signal recording, to make patients monitoring much more easier for cardiologists.
- We conceive and present a dashboard application for ECG visualization and for the online prediction of atrial fibrillation disease from only 9s ECG record.

## 2.5 Conclusion

In this chapter, we have reviewed the most interesting works in the classification of ECG signals to detect atrial fibrillation disease, we have seen that most modern works are based on deep learning techniques. Also, we mention that it is necessary to think about monitoring the heart rate by low-cost productions. Therefore, we present in the next chapter the design of our proposed system in the hope of contributing to the resolution of these issues.

## Chapter 03

An improved atrial fibrillation  
disease detection by the guided  
1-D CNN

# Chapter 3

## An improved atrial fibrillation disease detection by the guided 1-D CNN

### 3.1 Introduction

Previously, we have presented several powerful CNN models applied for atrial fibrillation detection. In this chapter, we present two propositions to improve a 1-D CNN architecture published by [27]. Our first proposition studies the impact of data length, denoising, and tuning on results while the second proposition is an experience of merging heart rate variability features with the obtained features from CNN, the idea behind it is to benefit from experts' knowledge with CNN power. The general and detailed architecture of the proposed system is explained step by step, showing how we enhance a 1D CNN to detect efficiently AF disease at an optimal time of 9s. Furthermore, we present a real-time acquisition system for a real prototype based on the cooking hacks platform's data to perform the AFib detection task.

## 3.2 Global conception

Detecting AF arrhythmia in a very short time of 9s can prevent severe complications such as stroke. In addition, it saves doctor's effort and time, and it averts medical errors resulting from faulty analysis. So, in this section, we present our automatic classification system for AF detection based on short ECG signals.

Overall, we illustrate the architecture of our early detection and monitoring system as follows:

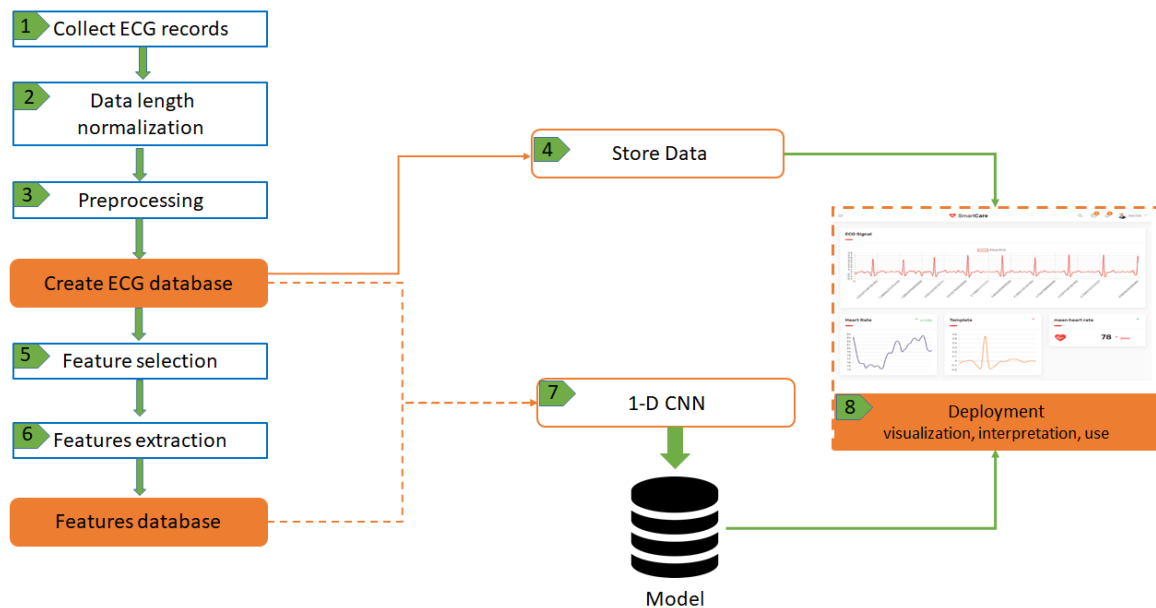


Figure 3-1: The general architecture of the system

As it is shown in Figure 3-1, we divide our work into eight big subtitles:

### 3.2.1 ECG data collection

We collect data from two sources, the first one is the physionet website which is considered a big world of ECG databases. The second source is the real patient data taken using the cooking hacks platform.

### **3.2.2 Data length normalization**

The collected ECG signals have variable length lasting from 9 seconds to just over 60 seconds, however, using 1-D Convolutional Neural Network necessitates fixed length. Thus, a segmentation must be done not only to fix this problem but also to generate more data.

### **3.2.3 Preprocessing**

Electrocardiograph (ECG) signals may be corrupted and affected by various kinds of noise and that may lead wrong interpretations [47], therefore, in the preprocessing step, different noises are analyzed and several methods and filtering process are applied to the raw ECG files in the aim to eliminate or at least minimize the noise and extract the essential meaningful features. This step is very sensible because we are dealing with a biomedical signal and the care should be taken not to alter the clinical information contained within the signal itself in any way.

In our project, we use the most efficient method for noise reduction in ECG signals we present it later.

### **3.2.4 Store ECG data**

Atrial fibrillation patients have to be monitored 24/24h a day. Consequently, ECG values should be recorded, hence we create a database that allows the permanent recording of ECG sensor data. The database not only to record raw ECG but also we store PDF report every 1h which contains raw ECG signal, cleaned ECG, ECG Peaks, information about the heart rate and other more about the last 1h recording.

### **3.2.5 Features selection**

In the first chapter, we have talked in detail about the ECG and how the heart acts in AF disease, we said that in AF there are no clear P waves, abnormal pattern of R-R interval, irregularity of heart beating etc.

The abnormality of the ECG signal is generally identified by using different methods that we mention later in the detailed conception. But for now, we mention :

- **Beat-related characteristics:** P, Q, R, S and T waves are the most crucial characteristics of ECG heart beats. Their locations are used to obtain and understand the pace statement of the ECG.
- **Heart rate variability features:** Simple measures of the small changes in each beat of the heart can provide a wealth of information on the health of the heart. AF can be detected with high accuracy using HRV features [34].

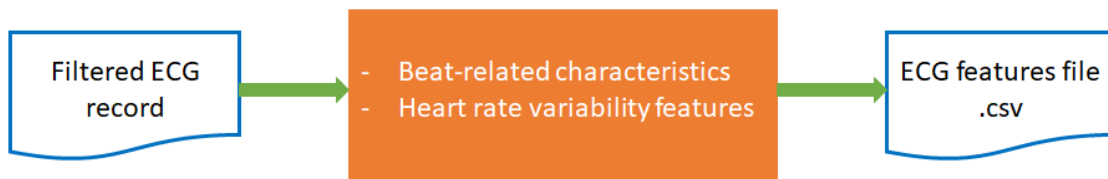


Figure 3-2: Features extraction

### 3.2.6 Features extraction

In this step, after the analysis of the ECG signal by discovering the onset and the offset of each wave, we calculate all the characteristics related to the RR intervals, for example, the calculation of the time and frequency domain features, etc.

### 3.2.7 Training using the guided 1-D CNN

The design of deep neural network architecture is rather challenging, and involves several aspects such as performance metric, loss function, optimization algorithm, and hyperparameter setting. Thus, we choose to implement an existing 1-D CNN architecture designed by the authors of the paper [27] using same database, same conditions.

However in our study, we focus on two things:

1. **If pre-processing (segmenting and adding a denoising algorithm) will affect the training process.**



2. **If injecting precious hand-crafted features as a support to the CNN improve results.**

### **3.2.8 Model deployment**

After training our data using deep learning architecture, we get a model that we use in the backend of our dashboard to test on several ECG signals.

## **3.3 Detailed conception**

In the detailed conception, we explain and highlight on the previous section element by element.

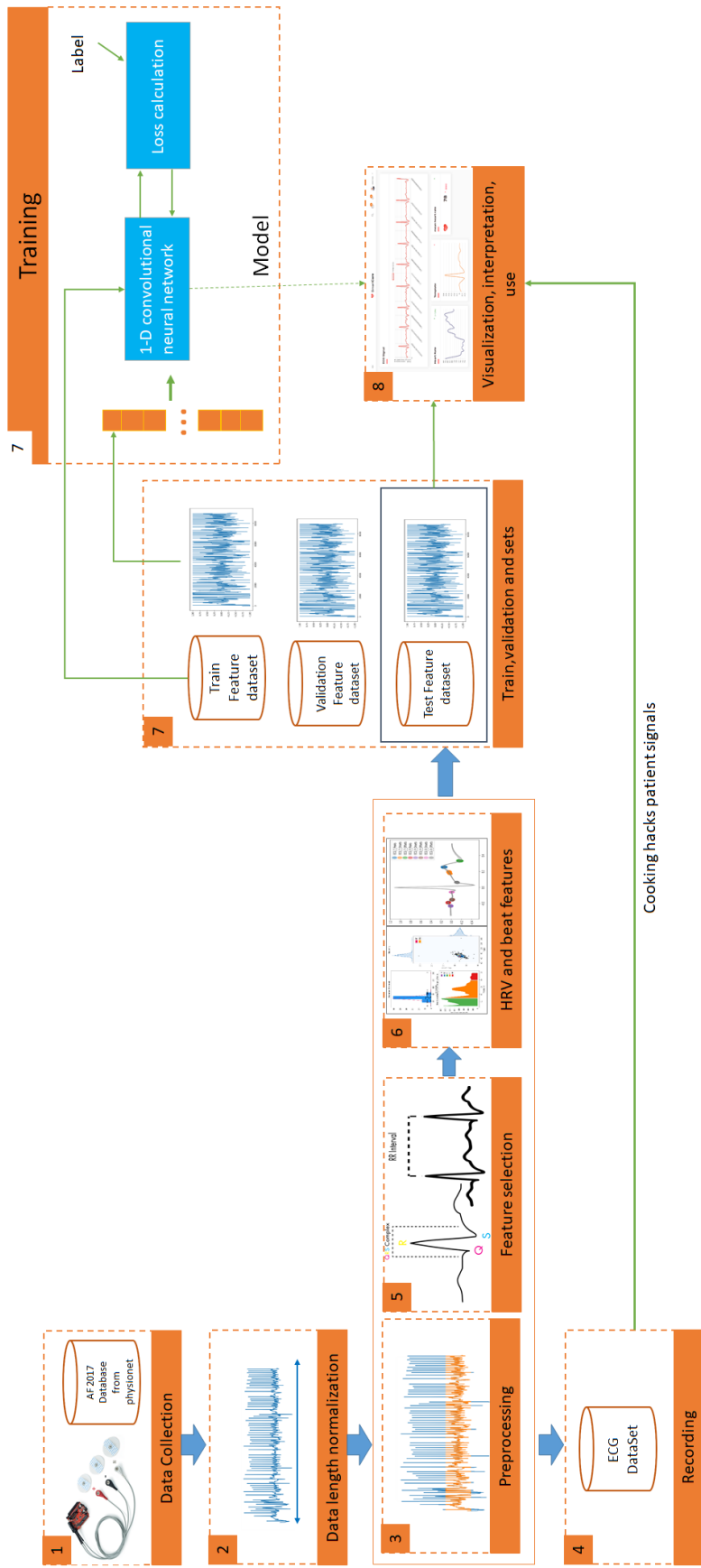


Figure 3-3: Detailed system's architecture

### 3.3.1 ECG data collection

Early and accurate detection of AF can significantly improve the prevention of complications, such as stroke.

Nearly 25% of all individuals with AF are asymptomatic. In these individuals, screening with an electrocardiogram (ECG) may be the gold standard of AF detection. Thus, we select ECG as the studied data source.

#### 3.3.1.1 PhysioNet website:

PhysioNet[39] offers free web access to large collections of recorded physiologic signals(PhysioBank)[37] and related open-source software (PhysioToolkit)[38].

#### 3.3.1.2 e-Health platform [20]:

The e-Health Sensor Shield allows Arduino and Raspberry Pi users to perform medical applications where body monitoring is needed by using several sensors. This information can be used to monitor in real time the state of a patient or to get sensitive data in order to be subsequently analyzed for medical diagnosis. Biometric information gathered can be wirelessly sent using any of the 6 available connectivity options such as: Wi-Fi, 3G, GPRS, Bluetooth, 802.15.4 and ZigBee depending on the application.

It is interesting to refer to the types of files that are acquired, namely:

- e-Health platform: real values, representing the electrical activity of the heart.
- Physionet Website: each ECG record have two files:
  - **.mat file** : binary file contains an array of real values, representing the electrical activity of the heart.
  - **.hea file (header file)** : containing additional information about the .mat file

### 3.3.2 Data length normalization

As we mention in the global conception, ECG records have variable size where the minimum length is 2700 values, which means of 9s. The implemented segmentation algorithm works as shown in the pseudocode (Algorithm 1). If the recording contains just 2700 samples (9 s), this recording will be used directly without any modification. If the recording contains more than 2700 samples, multiple segments will be generated. Each segment from the same recording has 2700 samples and will be assigned the same label.

---

**Algorithm 1.** Pseudocode of data length normalization

---

1: If the recording length is more than 2700:

    Chop record into segments of 2700 with overlapping between segments

2: If records length = 2700:

    Preserve the record

---

### 3.3.3 Preprocessing

The CNN architecture we are using takes as input raw ECG records. So, we decide to implement a denoising method to see the effect of preprocessing on results. The literature proposed many methods to reduce noise from ECG. However the most robust method is **discrete wavelet transform (DWT)** [47].

Wavelet transform (WT) is a powerful method for analyzing non stationary signals. ECG signals are time varying and non-stationary signals so WT is suitable for analyzing ECG signal. Wavelets allow both time and frequency analysis of signals.

This function not only a noise reduction method but also is considered as a compression which enhances the ECG quality, hence the learning will be easier.

- **Input** The input for this step is the raw ECG file.



Figure 3-4: Noisy ECG record

- **Output** Output of this step is the ECG file without noise and normalized.

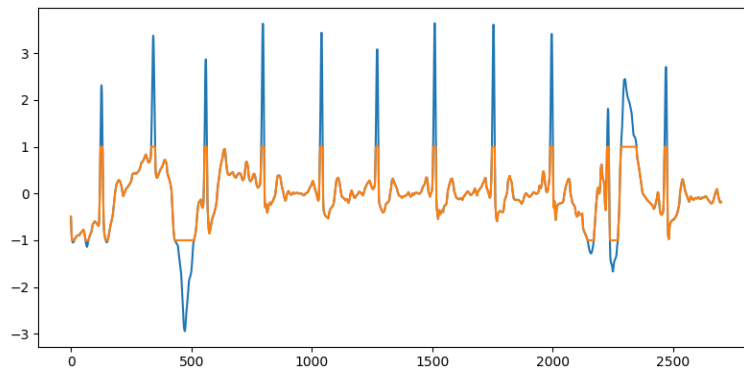


Figure 3-5: Filtered and normalized ECG record

### 3.3.4 ECG data storing

In order to create a dashboard for the benefit of doctors for an effective patients monitoring, we design an entity-relationship model integrated into the server so the access to these information will be rapid.

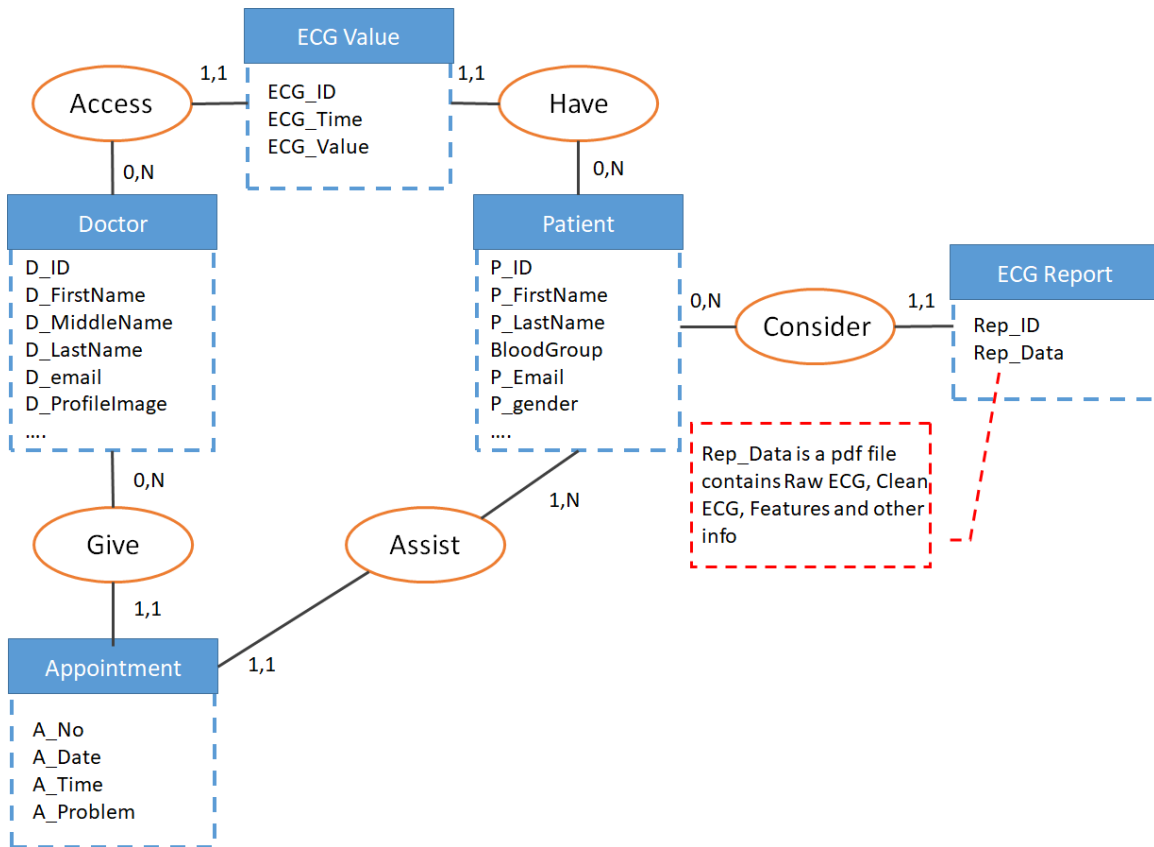


Figure 3-6: Entity-Relationship diagram for the dashboard

### 3.3.5 Feature selection

- **Input** The input to this step is represented by the noise-free ECG files resulting from the denoising step Figure 3-5
- **Output** The result of this step is a dictionary, which contains the time position of each peak (P, Q, R, S, T). These peaks will be used for calculation RR intervals features.

### 3.3.6 Feature extraction

- **Input** We use the dictionary resulting from the feature selection step as input in this step to calculate the necessary characteristics.
- **Output** We get 52 different features about RR intervals.

### 3.3.7 Training using the supported 1-D CNN

#### 3.3.7.1 1D-CNNs classifier

Studies have shown promising results by considering ECG sequence signals as 1D data using convolutions outperformed RNNs and DNNs [35]. 1-D CNNs are composed of two common layers, i.e., convolutional layer and fully connected layer. The convolutional layer aims to represent the features from the input. In the fully connected layer, the classification function is performed, the final class vector is generated.

- **Implemented CNN architecture :** The used CNN architecture from [27] contains 10 convolutional blocks, 2 fully-connected layers, and a Softmax layer as the output prediction (as shown in Table 3.1)

Table 3.1: Detailed 1-D CNN architecture.

| Layers     | Parameters          | Activation |
|------------|---------------------|------------|
| Conv1D     | Filter 32/kernel 5  | ReLu       |
| BN         |                     |            |
| Maxpooling | 2                   |            |
| Conv1D     | Filter 32/kernel 5  | ReLu       |
| Maxpooling | 2                   |            |
| Conv1D     | Filter 64/kernel 5  | ReLu       |
| Maxpooling | 2                   |            |
| Conv1D     | Filter 64/kernel 5  | ReLu       |
| Maxpooling | 2                   |            |
| Conv1D     | Filter 128/kernel 5 | ReLu       |
| Maxpooling | 2                   |            |
| Conv1D     | Filter 128/kernel 5 | ReLu       |
| Maxpooling | 2                   |            |
| Dropout    | 0.5                 |            |

**Table 3.1 – Detailed 1-D CNN architecture.**

| Layers     | Parameters          | Activation |
|------------|---------------------|------------|
| Conv1D     | Filter 256/kernel 5 | ReLu       |
| Maxpooling | 2                   |            |
| Conv1D     | Filter 256/kernel 5 | ReLu       |
| Maxpooling | 2                   |            |
| Dropout    | 0.5                 |            |
| Conv1D     | Filter 512/kernel 5 | ReLu       |
| Maxpooling | 2                   |            |
| Dropout    | 0.5                 |            |
| Conv1D     | Filter 512/kernel 5 | ReLu       |
| Flatten    |                     |            |
| Dense      | 128                 | ReLu       |
| Dropout    | 0.5                 |            |
| Dense      | 32                  | ReLu       |
| Dense      | 4                   | Softmax    |

### 3.3.7.2 1st proposal: an improvement on the 1-D CNN

1D-CNNs technique is useful to extract features of time sequence data. However, as aforementioned, the acquisition of the ECG signal contains high-gain instrumentation amplifiers, which are easily contaminated by various sources of noise.

This proposition will study four main ideas:

- **Length** feeding 9s ECG rather than 30s
- **Adding a denoising function**
- **Tuning** Changing data size necessitates changing on the architecture
- **Cross validation**



Our purpose from this proposal is to see whether size and denoising will improve results.

### 3.3.7.3 2nd proposal: inject hand-crafted features to the CNN

In this stage, we want to experience if we add extra features to CNN will help to improve model. As we know, we can divide CNN into two parts feature learning part and classification part (a fully connected layer).

Now the question is : what if we merge hand-crafted features with features that CNN extracts as input to the fully connected layer (As it is represented in Figure3-7) so we kind of use two types of features?

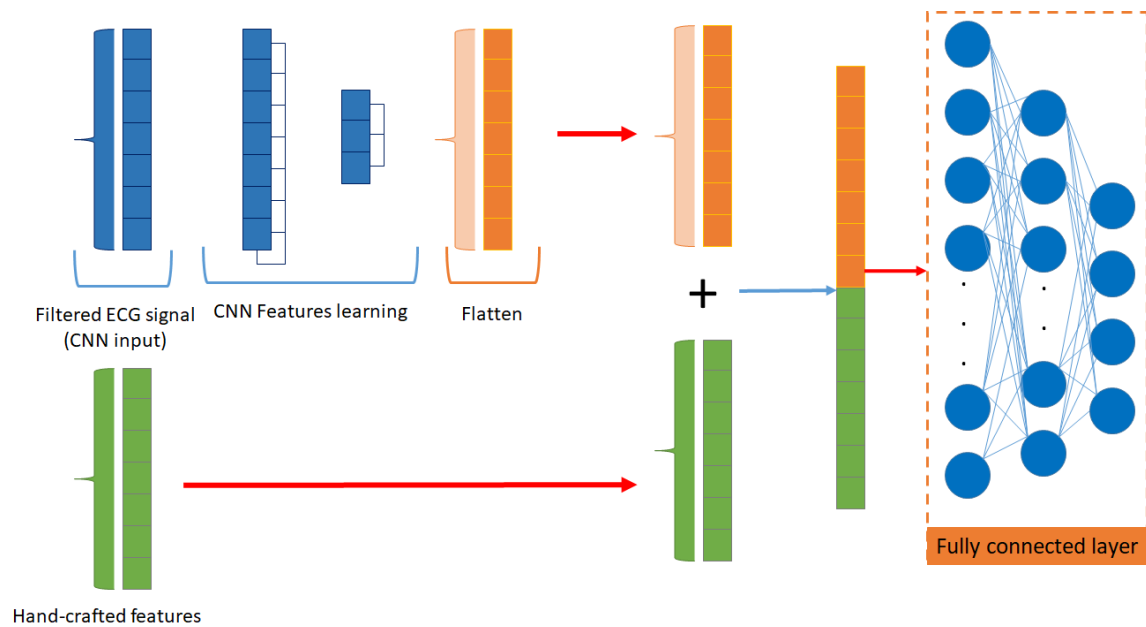


Figure 3-7: Proposed learning architecture

### 3.3.8 Performance evaluation [13]

#### 1. Model Evaluation Techniques :

(a) **Holdout:** The purpose of holdout evaluation is to test a model on different data than it was trained on. This provides an unbiased estimate of learning performance. In this method, the dataset is randomly divided into three subsets: Training set, Validation set, Test set.

(b) **Cross validation:** Cross-validation is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis.

The most common cross-validation technique is k-fold cross-validation in our work we try 10 fold cross validation.

2. **Classification Metrics:** It is very important to test the trained model in non-learned data. Many techniques and metrics are used to qualify and evaluate the obtained results by the trained model. These metrics are calculated either on the training examples themselves or on the reserved samples.

#### 3.3.8.1 Precision

The intuitive metric used is the precision of the model also called the recognition rate. It represents the ratio between the number of correctly classified data and the Number of data tested. The following equation is used to calculate the precision of the model.

#### 3.3.8.2 Confusion matrix

The confusion matrix is a matrix that gathers in rows the observations, and in columns the predictions. The elements of the matrix represent the number of examples corresponding to each case. Here, we assume the class containing AF

signals as the positive class, and therefore the sensitivity, specificity, F1 score are computed, respectively.

- **Sensitivity:** it represents the ratio of correctly predicted positive observations and the number of positive observations.

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

- **Specificity:** it represents the ratio of correctly predicted negative observations to the total number of negative observations.

$$Specitivity = \frac{TN}{TN + FP}$$

- **F1 score (Fmeasure):** the F score is the harmonic mean of the precision and recall.

$$F_1score = \frac{2 * S_v * S_p}{S_v + S_p}$$

### 3.3.9 Model deployment: Visualization, Interpretation, Use

To deploy our model, we made a dashboard (Figure 3-8) web application that contains the following features:

- Real-time monitoring for the cooking hacks patient, the ECG values sent directly to the doctor while they are recorded in the database. Also, we provide a statistical report each 1h about the last 1h recording as a pdf file and we store it with the patient documents. In addition, we control the heart rate if the heart rate exceeded the normal situation an alert will be sent to the doctor.
- Model deployment web page where the doctor or anyone can choose any file from the testset to know its class, we provide a record visualisation, ECG characteristics and the record class.

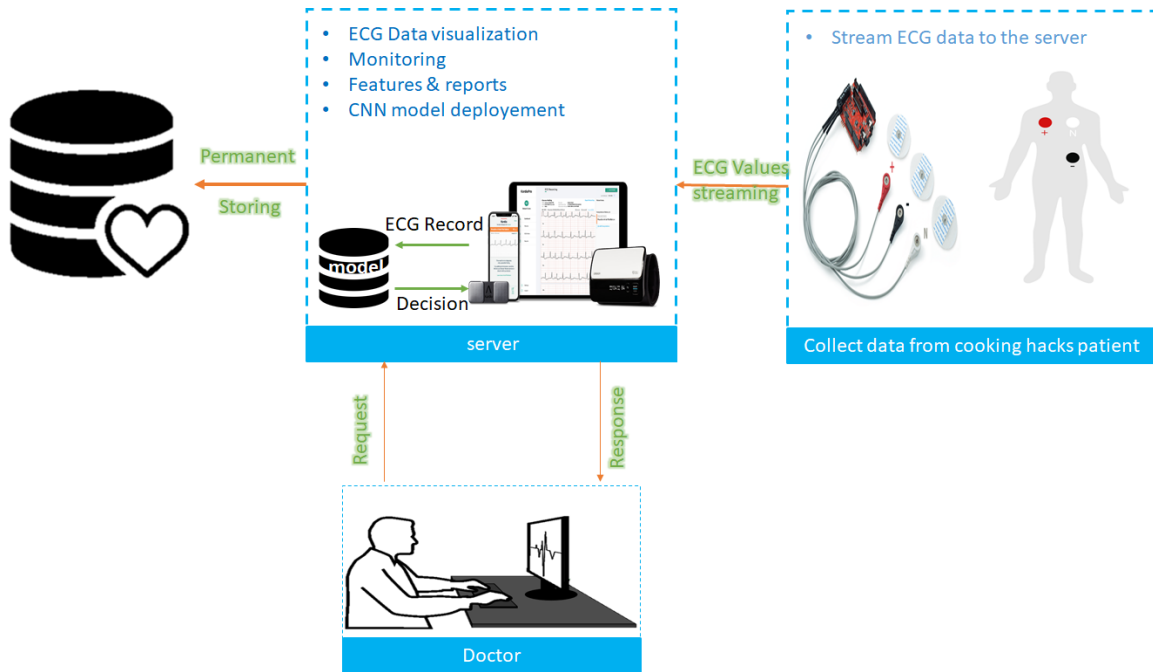


Figure 3-8: Model deployment: Visualisation, Interpretation, Use of our model

### 3.4 Conclusion

In this chapter, we have gave our ideas to detect atrial fibrillation disease. Firstly we have presented the design of a 1-D CNN architecture made by [27] then we have presented an improvement of this architecture by changing ECG records size, using discrete wavelet transform and tuning on the model. Not only this, we have suggested an experience consists of training on two levels of features (CNN extracted and hand-crafted features). We have explained step by step the followed process from collecting data, preprocessing, features extraction, learning until model deployment where we have propounded a dashboard application for using our CNN model and for the real-time monitoring of heart patients. The next chapter is assigned to the implementation of the explained steps in the conceptual chapter, also the evaluation of the proposed model and the discussion of the result obtained are explained.

## **Chapter 04**

# **Experimental study and results**

# Chapter 4

## Experimental study and results

### 4.1 Introduction

In the previous chapter, we have presented our conceptual approach: building a supported 1D CNN classifier by ECG segmentation, denoising and injecting precious hand-crafted features, and we have discussed the detailed steps, from data selection to system deployment.

In this chapter, we will try to implement and validate our idea by presenting the working environment, programming languages and tools we used to build our system. Subsequently, we will explain all the experiments carried out on the proposed methods as well as the obtained results.

### 4.2 Development tools and programming languages

During the implementation phase of this project, it was necessary to use certain areas of programming and we had the opportunity to familiarize ourselves with various development software techniques and tools which are presented below:

### 4.2.1 C++



C++ is a general-purpose programming language created by Bjarne Stroustrup as an extension of the C programming language, or "C with Classes". The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation.[3].

### 4.2.2 Python



Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc). This language can be used on a server to create web applications, connect to database systems. It can also read and modify files, to handle big data and perform complex mathematics and so many other functionalities [9].

### 4.2.3 PyCharm



PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains.

It provides code analysis, a graphical debugger, and supports web development with Flask as well as Data Science with Anaconda. [8].

#### 4.2.4 Anaconda



Anaconda is a free and open-source distribution of the Python programming language for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. [1].

#### 4.2.5 Flask



**Flask**

Flask is an open-source Python web development framework. Its main goal is to be light, in order to keep the flexibility of Python programming, associated with a system of templates.

Applications that use the Flask framework include Pinterest and LinkedIn. The distribution includes data-science packages suitable for Windows, Linux, and macOS [4].

#### 4.2.6 SQLite



SQLite is a relational database management system (RDBMS). It is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others [10].



## 4.3 Implementation

### 4.3.1 Evaluation and used Datasets

We evaluate the effectiveness of the two proposed methods with a set of experiments performed in 1h using google colab and using thinkpad i5-7300U CPU 2.7 GHZ. Moreover, as we want to stream data in real-time and to build a classification model, data comes from two different sources:

#### 4.3.1.1 e-Health Sensor Platform

The e-Health Sensor Shield platform (abbreviated e-HP) is a capture platform of health data that can be used with an Arduino or a Raspberry Pi. In our work, we use version V2.00 of the branded e-Health platform Cooking hacks equipped with a Raspberry Pi3 B. To plug the e-HP platform into the Raspberry, we need to use the Raspberry Pi connection bridge to "Arduino Shields", which allows the use of e-HP with the Raspberry Pi.



Figure 4-1: From left to right: Raspberry Pi3 B, The Connection bridge, The E-HP [20]

In order to extract ECG values from the platform, an acquisition protocol must be done:

#### a Connecting the Raspberry Pi to Arduino shields

the arduPi library allows to use Raspberry with the same code used in Arduino. By installing it we configure a communication between them. For that we can simply execute the following command in windows powershell [21]:

```
1 wget http://www.cooking-hacks.com/media/cooking/images/
  documentation/raspberry_arduino_shield/raspberrypi.zip &&
  unzip raspberrypi.zip && cd cooking/arduPi && chmod +x
  install_arduPi && ./install_arduPi && rm install_arduPi &&
  cd ../../
```

### b Installing the e-HP library

The e-Health library includes high level functions written in C++ for an easy acquisition of signals. The library can be downloaded from [20]. The e-Health library for Raspberry Pi requires the ArduPi library and both libraries should be in the same path.

### c Plugging the ECG sensors:

In this stage, the platform is ready for acquisition, so we plug ECG sensors into the platform, then we assemble our system.

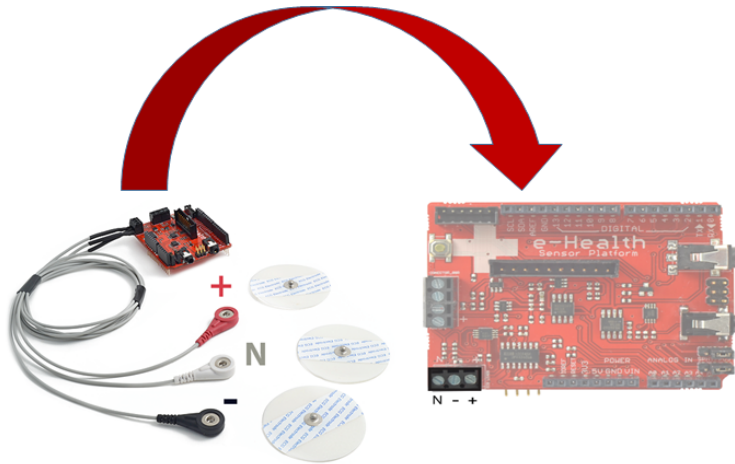


Figure 4-2: ECG sensors plugged into the e-HP (with appropriate placements) [20]

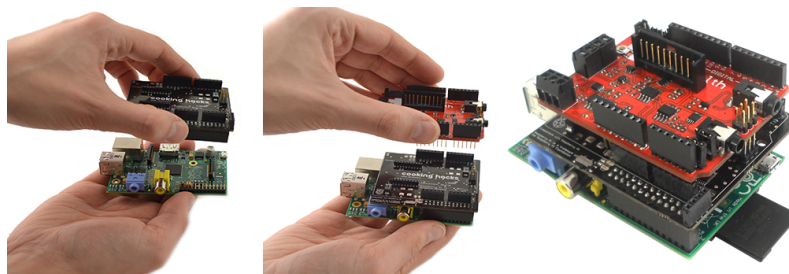


Figure 4-3: The assembled system [20]

#### d Data acquisition

Cooking hacks website offers a C++ source code to get ECG sensor samples. This ECG returns an analogic value in volts (0 – 5v) to represent the ECG waveform. The compilation of the program provided by the cooking-hacks website can be carried out as follows:

```
1 g++ -lpthread -lrt ECG.cpp arduPi.cpp eHealth.cpp -o ECG
```

Where ECG.cpp is the name of the file that contains the acquisition program and ECG the resulting file.

Here, everything is considered ready, all that remains is placing the ECG electrodes on patient and operating the system, as shown below



Figure 4-4: ECG electrodes position [20] [22]

```
pi@raspberrypi:~/Desktop/arduPi-master/arduPi $ sudo ./ecg
ECG value : 3.191593 V
ECG value : 1.813294 V
ECG value : 1.094819 V
ECG value : 2.390029 V
ECG value : 3.719453 V
ECG value : 3.318671 V
ECG value : 1.925709 V
ECG value : 1.094819 V
ECG value : 2.184751 V
ECG value : 3.558162 V
ECG value : 3.553275 V
ECG value : 2.135875 V
ECG value : 1.197459 V
ECG value : 1.901271 V
ECG value : 3.396872 V
ECG value : 3.812317 V
ECG value : 2.346041 V
ECG value : 1.339198 V
ECG value : 1.593353 V
^C
pi@raspberrypi:~/Desktop/arduPi-master/arduPi $ █
```

Figure 4-5: Obtained ECG values

Executing the program in terminal will give the corresponding output (Figure 4-5)

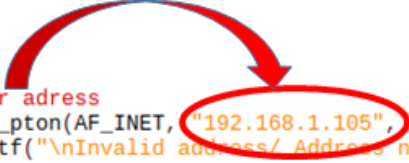
#### e Real-time data collection

As we explained in the previous chapter, and given the importance of real-time monitoring and the importance of saving data for the doctor, and due to the small size of Raspberry Pi 3 for backup, it is necessary to have a distant server for receiving, preprocessing, processing and saving data.

To achieve this, we suggest a client-server architecture to accomplish this step using sockets.

- **Client side part (Patient)** Every acquired ECG value and the exact acquisition time (by parts of milliseconds) are transmitted immediately to the server without any storage in the raspberry pi.

The source code is illustrated in the image down bellow (Figure 4-6)



```

// Server address
if (inet_pton(AF_INET, "192.168.1.105", & serv_addr.sin_addr) <= 0) {
    printf("\nInvalid address/ Address not supported \n");
    return -1;
}

if (connect(sock, (struct sockaddr * ) & serv_addr, sizeof(serv_addr)) < 0) {
    printf("\nConnection Failed \n");
    return -1;
}

while (true) {
    char fmt[64], buf[64];
    char result[100];
    struct timeval tv;
    struct tm * tm;

    //Exact time by hour, minute, seconds and milliseconds
    gettimeofday( & tv, NULL);
    if ((tm = localtime( & tv.tv_sec)) != NULL) {
        strftime(fmt, sizeof fmt, "%Y-%m-%d %H:%M:%S.%06u%z ", tm);
        sprintf(buf, sizeof buf, fmt, tv.tv_usec);
    }
    //Get the ECG Value
    float ECG = eHealth.getECG();

    char buffer[80];
    sprintf(buffer, sizeof buffer, "%f", ECG);
    strcpy(result, buf);
    strcat(result, buffer); // Concatenate Time with ECG in the format (Time,ECGValue)
    printf("%s", result);
    send(sock, result, strlen(result), 0); //Send TO the server
    printf("ECG Value sent\n");
    valread = read(sock, buffer, 1024);
    printf("%s\n", buffer);
}

```

Figure 4-6: The illustrative code for sending data from the Raspberry to the server. (Patient side)

- **Server side**

- **Getting data** The server is constantly receiving data from the client 24 hours a day, it normalize data using the operation (Normalization function from [41])

```
1 self.ECG = (self.ECG * 5) / 1023
```

Then, it transmits the received data to the doctor immediately and saves them in the application database in case the doctor want to access this data later (View Figure 4-7).

```

serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = "0.0.0.0"
port = 8000
print(host)
print(port)
serversocket.bind((host, port))
class client(Thread):
    def __init__(self, socket, address):
        Thread.__init__(self)
        self.sock = socket
        self.addr = address
        self.start()

    def run(self):
        while 1:
            print("Connected! Starting reading ECG measurements.")
            #Get ECG From Patient
            raw_measurement = self.sock.recv(1024).decode()
            #Split time and ECG values
            raw_measurement_split = raw_measurement.rstrip().split(',')
            time = raw_measurement_split[0]
            self.ECG = float(raw_measurement_split[1])
            #Normalization
            self.ECG = (self.ECG * 5) / 1023
            print(time, self.ECG)

            #send record to the database
            sendData(time, self.ECG)
            self.sock.send(b'Record saved')

```

Figure 4-7: The illustrative code for receiving data from the Raspberry. (Server side)

- **Connecting to the database** To allow remote access to the database, we use SQLAlchemy [25] which is an opensource SQL toolkit and objectrelational mapper (ORM) for the Python programming language.

It facilitates different operations that can be done to a database from creation, insertion to querying tables, so we treat our database in python.

```

from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_login import LoginManager
import os
app = Flask(__name__)
#Connect to the database from server
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///AtrialFibrillation.db'

```

Figure 4-8: The illustrative code for database connection

```

37 #To Create patient table
38 class Patient(db.Model):
39     id = db.Column(db.Integer, primary_key=True)
40     firstname = db.Column(db.String(20), nullable=True)
41     middlename = db.Column(db.String(20), nullable=True)
42     lastname = db.Column(db.String(20), nullable=True)
43     birthdate = db.Column(db.DateTime, nullable=True, default=datetime.now)
44     email = db.Column(db.String(120), unique=True, nullable=False)
45     contactD = db.Column(db.String(20), nullable=True)
46     dblood= db.Column(db.String(20), nullable=True)
47     gender= db.Column(db.String(20), nullable=True)
48     bio =db.Column(db.String(200), nullable=True)
49     adressline=db.Column(db.String(100), nullable=True)
50     city=db.Column(db.String(20), nullable=True)
51     pin=db.Column(db.String(20), nullable=True)
52     state=db.Column(db.String(20), nullable=True)
53     country=db.Column(db.String(100), nullable=True)
54     picture = db.Column(db.String(20), nullable=False, default='dash.png')
55     doctor_id = db.Column(db.Integer, db.ForeignKey('doctor.id'), nullable=False)
56     appointment = db.relationship('Appointement', backref='paapp', lazy=True)
57     repport = db.relationship('Repports', backref='rep', lazy=True)
58     ECGdt = db.relationship('ECGData', backref='ecg', lazy=True)
59 #ECG table
60 class ECGData(db.Model):
61     id = db.Column(db.Integer, primary_key=True)
62     timeECG = db.Column(db.DateTime, nullable=False)
63     ECGValues = db.Column(db.Float)
64     doctor_id = db.Column(db.Integer, db.ForeignKey('doctor.id'), nullable=False)
65     patient_id = db.Column(db.Integer, db.ForeignKey('patient.id'), nullable=False)
66 #Repports (PDF) for patients
67 class Repports(db.Model):
68     id = db.Column(db.Integer, primary_key=True)
69     repport_name = db.Column(db.String(200), nullable=False)
70     data = db.Column(db.LargeBinary)
71     patient_id = db.Column(db.Integer, db.ForeignKey('patient.id'), nullable=False)

```

We can access to all patient ECG records easily

Figure 4-9: The illustrative code for some tables creation

As you can see in Figure4-9 tables are treated as objects, We can access them by calling them :

```

from fla.models import Patient, ECGData

#get cooking hacks patient
Cooking_hacks_patient = Patient.query.get(2)

#get patient ECG records
Patient_records = Cooking_hacks_patient.ECGdt

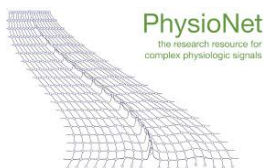
#print sample per sample
for i in Patient_records:
    print(i, i.timeECG, i.ECGValues)

```

| ID              | timeECG                         | ECGValues |
|-----------------|---------------------------------|-----------|
| <ECGData 18039> | 2020-08-16 09:12:26.722458+0100 | 1.969697  |
| <ECGData 18040> | 2020-08-16 09:12:27.784408+0100 | 1.348974  |
| <ECGData 18041> | 2020-08-16 09:12:28.816733+0100 | 3.587488  |
| <ECGData 18042> | 2020-08-16 09:12:29.865326+0100 | 1.138807  |
| <ECGData 18043> | 2020-08-16 09:12:30.902475+0100 | 1.363636  |
| <ECGData 18044> | 2020-08-16 09:12:31.934905+0100 | 3.60215   |
| <ECGData 18045> | 2020-08-16 09:12:32.972481+0100 | 3.993157  |
| <ECGData 18046> | 2020-08-16 09:12:34.005981+0100 | 2.121212  |
| <ECGData 18047> | 2020-08-16 09:12:35.040511+0100 | 1.265885  |
| <ECGData 18048> | 2020-08-16 09:12:36.072868+0100 | 3.411535  |
| <ECGData 18049> | 2020-08-16 09:12:37.138755+0100 | 1.383187  |
| <ECGData 18050> | 2020-08-16 09:12:38.187062+0100 | 3.387097  |
| <ECGData 18051> | 2020-08-16 09:12:39.222467+0100 | 1.88172   |
| <ECGData 18052> | 2020-08-16 09:12:40.245720+0100 | 3.387097  |

Figure 4-10: Cooking-hacks records after saving in database

### 4.3.1.2 PhysioNet web site [39]



As mentioned in chapter 3, the PhysioNet website offers a free web access to many recorded physiological signals in databases, most of them being dedicated to the study and the analysis of the ECG signal.



In our project, we used the **2017 PhysioNet/CinC Challenge atrial fibrillation database** [36]. This last contains 8528 single lead variable-length ECG recordings lasting from 9 s to 61 s.

The table below 4.1 describe the data profile for the database.

| Type       | Normal | Atrial fibrillation | Other rhythms | Noisy | Total |
|------------|--------|---------------------|---------------|-------|-------|
| Recordings | 5154   | 771                 | 2557          | 46    | 8528  |

Table 4.1: Data profile for the Dataset.

Each ECG record includes :

- **A .mat file** : it is a binary file containing digitized signal samples. Luckily Python offers multiple tools to treat such files, we mention:

1. **WaveForm DataBase (WFDB)**: a software for viewing, analyzing, and creating recordings of physiologic signals [38].
2. **SciPy**: it is an open-source software for mathematics, science, and engineering [43]. By reading the .mat file, we get a 1D array representing the ECG values and a dictionary contains additional information.

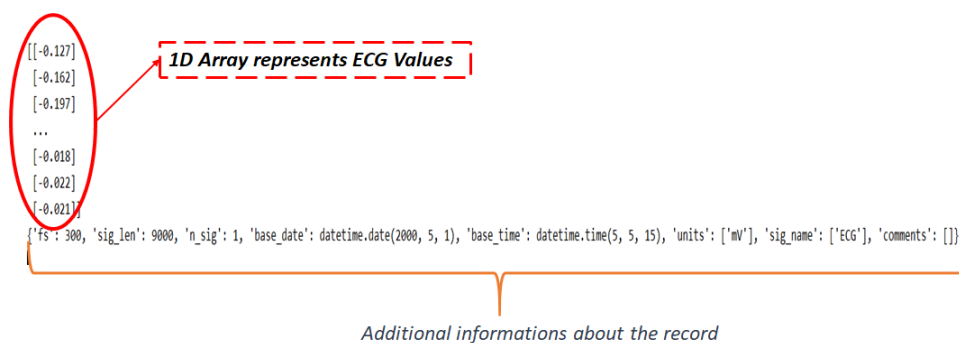


Figure 4-11: Example of a reading (.mat) file

- **A .hea file** : it is a Short text file describing the content of associated signal file (Name of record, number of leads, sampling rate, length, time of acquisition, day of acquisition, start value etc.)

```
A00001 1 300 9000 05:05:15 1/05/2000
A00001.mat 16+24 1000/mV 16 0 -127 0 0 ECG
```

Figure 4-12: Example of a record's (.hea) file

- **A .csv file** : it contains Class of each ECG record.

|      |        |   |
|------|--------|---|
| 992  | A00992 | N |
| 993  | A00993 | O |
| 994  | A00994 | N |
| 995  | A00995 | N |
| 996  | A00996 | N |
| 997  | A00997 | O |
| 998  | A00998 | N |
| 999  | A00999 | N |
| 1000 | A01000 | N |
| 1001 | A01001 | N |
| 1002 | A01002 | N |
| 1003 | A01003 | N |
| 1004 | A01004 | N |
| 1005 | A01005 | A |
| 1006 | A01006 | ~ |
| 1007 | A01007 | N |
| 1008 | A01008 | N |
| 1009 | A01009 | N |
| 1010 | A01010 | N |
| 1011 | A01011 | N |
| 1012 | A01012 | N |
| 1013 | A01013 | A |

Figure 4-13: Records classes

Where :

| Class         | <b>N</b> | <b>A</b>            | <b>O</b>             | ~     |
|---------------|----------|---------------------|----------------------|-------|
| Record's type | Normal   | atrial fibrillation | Other heart problems | Noisy |

Table 4.2: ECG records classes

## 4.3.2 Data length normalization

We create a new dataset by implementing a segmentation function for solving variable length, also for generating more data.

```
1 for i in range(size):
2     if mats[i][0] == 'A':
3         #load ECG file
4         ECG = sio.loadmat(mypath + mats[i])
5         if ('val' in ECG.keys()):
6             #GET ECG values
7             ECG = ECG['val'][0, :]
8
9             #if ECG length is greater than 2700
10            if len(ECG) > 2700:
11                #shop ECG values
12                bufOut = Segmentation(ECG, duration, dataOverlap)
13                print(bufOut.shape)
14                for segment in bufOut:
15                    #save new records with the same structer of the old
16                    one
17                    sio.savemat('SampledData/' + str(count) + '.mat',
18                                {'val':segment})
19                    #create the annotation for each segment
20                    annList.append(saved_column[ann])
21                    count+=1
22            else:
23                #if record == 2700 save it as it is
24                sio.savemat('SampledData/' + str(count) + '.mat', {'
25                val': ECG})
26                annList.append(saved_column[ann])
27                count += 1
28                print(annList)
29            ann+=1
```

Listing 4.1: Applying the segmentation algorithm on records

We have removed 33 random files from the dataset for testing on them later. After that, we have applied the segmentation algorithm

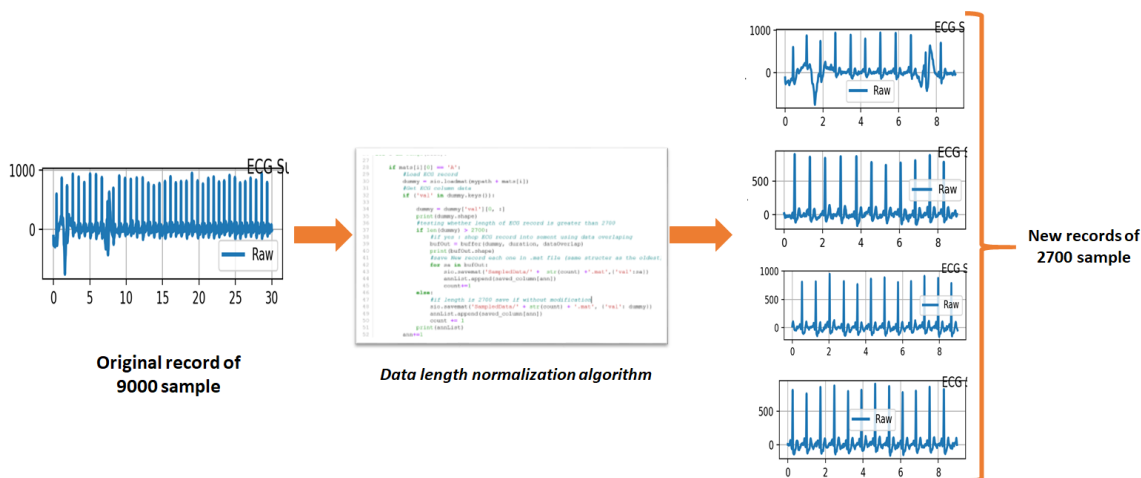


Figure 4-14: Segmentation output

After segmentation, the reached data profile is depicted in the table 4.3

| Type       | Normal | Atrial fibrillation | Other rhythms | Noisy | Total  |
|------------|--------|---------------------|---------------|-------|--------|
| Recordings | 123211 | 18231               | 66563         | 889   | 208894 |

Table 4.3: Data profile for the database after segmentation

### 4.3.3 Preprocessing and features selection

#### 4.3.3.1 ECG denoising

The challenge of ECG preprocessing is to filter and minimize these noise components, therefore making it easier to diagnose various heart disorders. In this stage many methods are proposed in the literature and many physiological libraries helps to discard noise. In our project, we are interested in using **denoise\_wavelet** function from **skimage.restoration** which is a python library for denoising and compression of ECG signals, which enhances the quality of ECG. The input and output this step are already shown in chapter 3 (Figure 3-4 and 3-5).

### 4.3.3.2 Waves positions detection

Having the position of beats waves is the gold standard to detecting any heart rhythm problem. Aforetime, we mentioned how the heart acts in atrial fibrillation; we said that RR intervals are variable and not constant so we have to locate those peaks to get heart rate variability features.

In this section, we present how we can provide such precious characteristics.

1. **R Peaks** The `ecg_peaks()` function from Neurokit2 [42] will return a dictionary containing the samples at which R-peaks are located. We can visualize them as well.

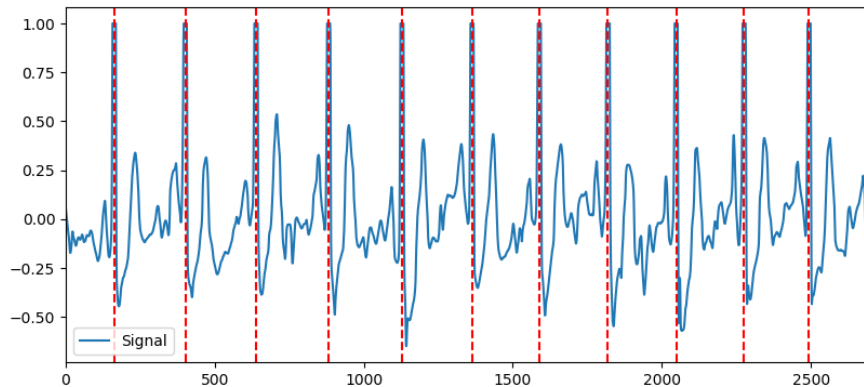


Figure 4-15: R Peaks Positions

2. **Locate other waves (P, Q, S, T) and their onset and offset**

There are many methods to locate ECG waves. We present two of them:

- (a) **Peak method** First, let's take a look at the peak method and its output. Visually, the peak method seems to have correctly identified the P-peaks, Q-peaks, S-peaks and T-peaks for this signal, at least, for the first few complexes. However, it can be quite tiring to be zooming in to each complex and inspect them one by one.

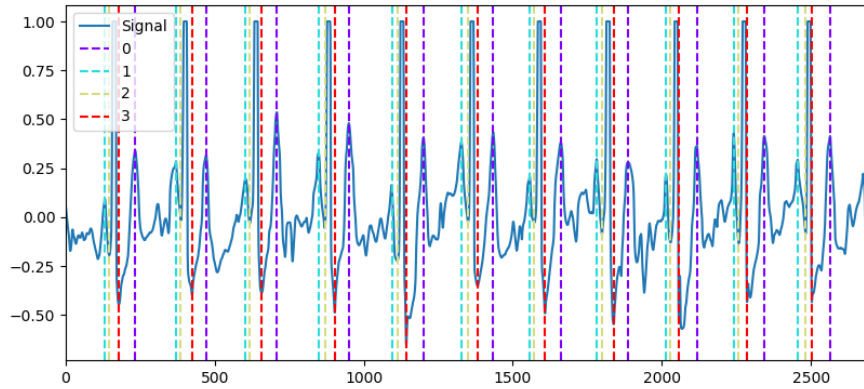


Figure 4-16: Location of ECG Peaks (Peak method) where 0- T Peak 1- P Peaks 2- Q Peak 3- S Peak

(b) **Discrete Wavelet Method (DWT)** The wavelet transform function is not only used for denoising but also used to detect the peaks and QRS complex in the ECG signal to identify the abnormality in the recorded signal [42].

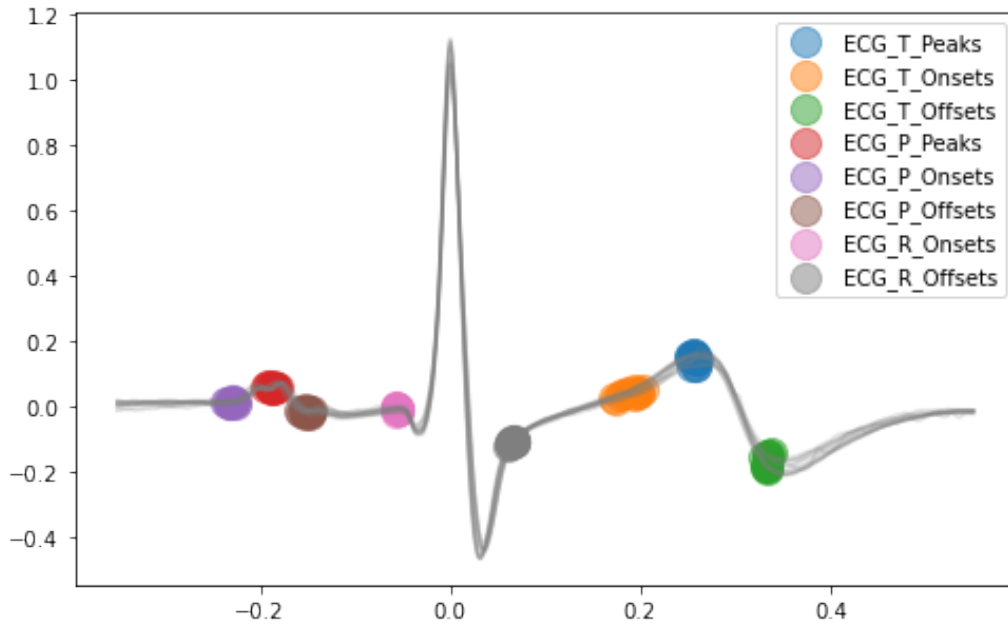


Figure 4-17: DWT method to locate beat peaks [42]

In our project, we use the DWT method provided by Neurokit library to get ECG peaks that's because it is the most accurate way.

## 4.3.4 Processing : Features extraction

### 4.3.4.1 Heart rate

AF patient's heart rate is irregular and it may exceed 100 (bpm). We obtain heart rate values using `ecg_process` function from Neurokit2.

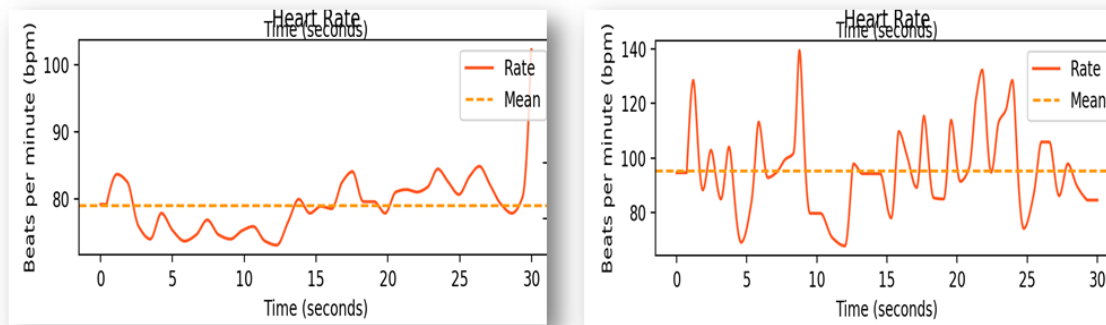


Figure 4-18: Normal heart rate and Atrial fibrillation's patient heart rate

### 4.3.4.2 Interval-related Analysis (Heart Rate Variability (HRV))

After extracting beats locations, we use them to extract HRV.

One of the first things to know when understanding heart rate is that the most informative metric relies not just on the heart rate, but how much the heart rate varies. Simple measures of the small changes in each beat of our heart can provide a wealth of information on the health of the heart.

NeuroKit2 library is the most comprehensive software for computing HRV indices that we can use to extract more than 52 feature about every ECG signal. In this section, we will show how did we use this library with our data :

#### 1. Time Domain Features

The table below (Table 4.4) describes some of the time domain features:

| Feature Number | Description of the features extracted                    |
|----------------|--|
| 1              | <b>MRRI</b> Mean of all RR interval (ms)                 |
| 2              | <b>SDNN</b> Standard deviation of NN interval (ms)       |
| 3              | <b>NN50</b> Number of NN > 50ms(ms)                      |
| 4              | <b>pNN50</b> percentage of NN > 50ms(%)                  |
| 5              | <b>MaxMin heart rate</b> Max(hr)-min(hr)(btm)            |
| 6              | <b>Mean hr</b> mean of total heart rate (btm)            |
| 7              | <b>Mean NN</b> mean of total normal-to-normal beats (ms) |

Table 4.4: Description of time domain features

Where : NN is the successive difference between normal to normal R to R peak (exclude abnormal beats).

All these features and more can simply extracted using the following code:

```

1 #AF record
2 ECGAF = sio.loadmat("training2017/A00004.mat")
3 #Read the ECG values
4 ECGAF = ECGAF['val'][0,: ]
5 #wavelet transform
6 x_denoise = denoise_wavelet(ECGAF,method="BayesShrink",mode="
    soft",wavelet_levels=3,wavelet="sym5")
7 #to get all time features R peaks should be extracted
8 peaks, info = nk.ecg_peaks(x_denoise, sampling_rate=300)
9 #Get time domain features
10 hrv_indices = nk.hrv_time(peaks, sampling_rate=300, show=True)

```

Listing 4.2: Illustrative code of extracting time domain features

We can also visualize the distribution of R-R intervals by specifying show=True in `hrv_time()`.



2. **Frequency Domain Features** The table below (Table 4.5) describes some of the frequency domain features:

| Feature Number | Description of the features extracted     |
|----------------|---|
| 1              | <b>ULF</b> ultra low frequency ( $ms^2$ ) |
| 2              | <b>VLF</b> Very low frequency ( $ms^2$ )  |
| 3              | <b>LF</b> Low frequency ( $ms^2$ )        |
| 4              | <b>HF</b> High frequency ( $ms^2$ )       |

Table 4.5: Description of frequency domain features

These features and more can simply extracted using the following code:

```
1 #AF record
2 ECGAF = sio.loadmat("training2017/A00004.mat")
3 #Read the ECG values
4 ECGAF = ECGAF['val'][0,:]
5 #wavelet transform
6 x_denoise = denoise_wavelet(ECGAF,method="BayesShrink",mode="
    soft",wavelet_levels=3,wavelet="sym5")
7 #to get all frequency features R peaks should be extracted
8 peaks, info = nk.ecg_peaks(x_denoise, sampling_rate=300)
9 #Get frequency domain features
10 hrv_indices = nk.hrv_frequency(peaks, sampling_rate=300, show=
    True)
11 print(hrv_indices)
12 #Plot frequency domain features
13 plt.show()
```

Listing 4.3: Source code of extracting frequency domain features

3. **All Domains Features** Finally, to extract HRV indices from all domains (frequency, time, Non-Linear), we can input peaks into `hrv()`, where we can specify `show=True` to visualize the combination of plots depicting the RR intervals distribution, power spectral density for frequency domains, and the poincare scattergram.

```

1 #DWT library
2 from skimage.restoration import denoise_wavelet
3 #for plotting
4 import matplotlib.pyplot as plt
5 import pandas as pd
6 #Nurokit2 library
7 import neurokit2 as nk
8 import scipy.io as sio
9 import sys
10 import numpy
11 numpy.set_printoptions(threshold=sys.maxsize)
12 #AF record
13 ECGAF = sio.loadmat("training2017/A00004.mat")
14 #Read the ECG values
15 ECGAF = ECGAF['val'][0,:]
16
17 #wavelet transform
18 x_denoise = denoise_wavelet(ECGAF,method="BayesShrink",mode="
    soft",wavelet_levels=3,wavelet="sym5")
19 #to get all domain features R peaks should be extracted
20 peaks, info = nk.ecg_peaks(x_denoise, sampling_rate=300)
21 #Get all domain features
22 hrv_indices = nk.hrv(peaks, sampling_rate=300, show=True)
23 with pd.option_context('display.max_rows', None, 'display.
    max_columns', None):
24     print(hrv_indices)
25 #Plot all domain features
26 plt.show()

```

Listing 4.4: Illustrative code of extracting all heart rate variability features from an ECG signal

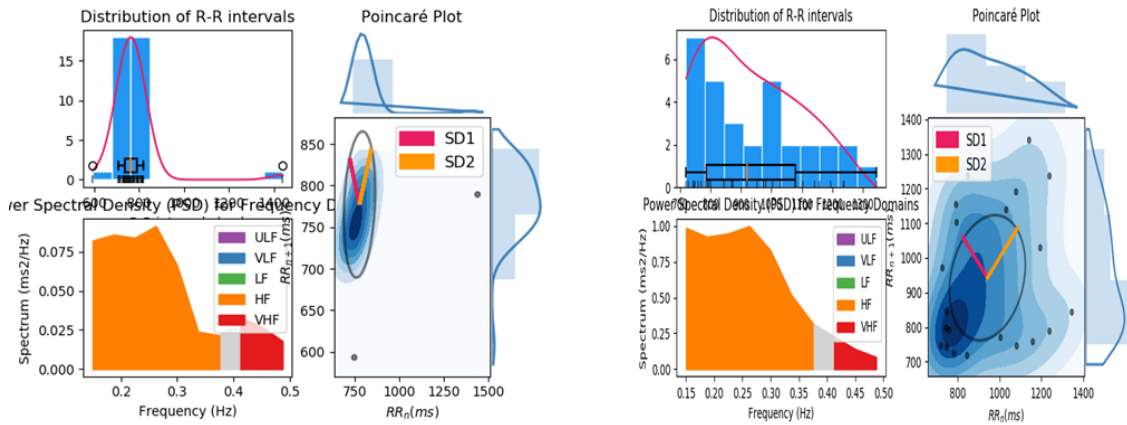


Figure 4-19: Heart Rate Variability features (left: Normal signal, Right: AF signal)

As you can see in the figure 4-19, atrial fibrillation signal shows high-dispersion in poincare plot also it shows an irregularity in the distribution of RR intervals.

After extracting all these features, we obtain the following dataframe:

|   | ECG_Rate_Mean      | HRV_RMSSD          | HRV_MeanNN         | HRV_SDNN           | HRV_SDS            | f      |
|---|--------------------|--------------------|--------------------|--------------------|--------------------|--------|
| 0 | 79.00443689527174  | 10.375832243094628 | 227.15384615384616 | 13.011206630274861 | 10.46354506125592  | 0.0572 |
| 1 | 69.67393714427897  | 83.8091209236799   | 263.5151515151515  | 71.8736396445705   | 85.1480151019767   | 0.2727 |
| 2 | 81.69454575887136  | 19.88543134424887  | 220.4875           | 19.576525434848385 | 19.999318391501117 | 0.0887 |
| 3 | 63.97292335132984  | 69.65233862450039  | 284.1333333333333  | 54.57404545002697  | 70.84962915032736  | 0.1920 |
| 4 | 113.66653144640917 | 80.74674629381073  | 161.4862385321101  | 66.30004953337337  | 81.12297669935144  | 0.4105 |
| 5 | 113.66653144640917 | 80.74674629381073  | 161.4862385321101  | 66.30004953337337  | 81.12297669935144  | 0.4105 |
| 6 | 58.04370851832204  | 9.630199413342423  | 310.67857142857144 | 19.918451470802367 | 9.429903335828895  | 0.0641 |
| 7 | 62.56387831647509  | 8.90826261720785   | 287.7586206896552  | 15.84896385169749  | 9.005876975942993  | 0.0550 |
| 8 | 69.87951101416037  | 34.767293623207905 | 258.1014492753623  | 28.123752222013042 | 35.01795601110644  | 0.1089 |
| 9 | 95.1150084989955   | 42.50668396638559  | 190.2888888888889  | 32.7855443632086   | 42.96443782417697  | 0.1722 |

Figure 4-20: Heart Rate Variability features

### 4.3.5 Training by the improved 1D Convolution Neural Network

#### 1. Input data

```
1 X = np.zeros((size, big))#initialiation of matrix of training
2 for i in range(size):#for every ECG record
3     ECG = sio.loadmat(mypath + mats3[i])#read record
4     ECG = ECG['val'][0,:] #get record in list
5     X[i, :] = ECG[0:] #add the record to the training matrix
6 X1 = np.expand_dims(X, axis=2)#change dimension of X to be 1D
   array
```

Listing 4.5: Implementation of the 1-D CNN

#### 2. The original CNN architecture We tried to implement as much as possible the same architecture of [27]. The code is presented in the figure below :

```
1 def create_model():
2     model = Sequential()
3     #input layes where it takes a 1D array of 9000 ECG value
4     #Convolution layers for features extraction
5     model.add(Conv1D(32, 5, activation='relu', input_shape
6     =(9000, 1)))
7     model.add(MaxPooling1D(2))
8     model.add(Conv1D(32, 5, activation='relu'))
9     model.add(MaxPooling1D(2))
10    model.add(Conv1D(64, 5, activation='relu'))
11    model.add(MaxPooling1D(2))
12    model.add(Conv1D(64, 5, activation='relu'))
13    model.add(MaxPooling1D(2))
14    model.add(Conv1D(128, 5, activation='relu'))
15    model.add(MaxPooling1D(2))
16    model.add(Conv1D(128, 5, activation='relu'))
17    model.add(MaxPooling1D(2))
18    model.add(Dropout(0.5))
```

```

18     model.add(Conv1D(256, 5, activation='relu'))
19     model.add(MaxPooling1D(2))
20     model.add(Conv1D(256, 5, activation='relu'))
21     model.add(MaxPooling1D(2))
22     model.add(Dropout(0.5))
23     model.add(Conv1D(512, 5, activation='relu'))
24     model.add(MaxPooling1D(2))
25     model.add(Dropout(0.5))
26     model.add(Conv1D(512, 5, activation='relu'))
27     #Flatten to make the extracted features in 1D array
28     model.add(Flatten())
29     #Training
30     model.add(Dense(128, kernel_initializer='normal', activation
31     ='relu'))
31     model.add(Dropout(0.5))
32     model.add(Dense(32, kernel_initializer='normal', activation=
33     'relu'))
33     model.add(Dense(number_of_classes, kernel_initializer='
34     normal', activation='softmax'))
34     opt = optimizers.Adam(learning_rate=0.0001)
35     model.compile(loss='categorical_crossentropy', optimizer=opt
36     , metrics=['accuracy'])
36     return model

```

Listing 4.6: Implementation of the 1-D CNN

To evaluate the network complexity, we estimated the total number of network training parameters using the **summary** function from the deep learning platform **Keras** python. The parameters of each layer, excluding non-trainable layers such as the pooling layer, dropout, and flatten layer, are shown in . The total number of network training parameters of the original 1D CNN is approximately 3 million, which is a perfect complexity comparing to other architectures [27].

| Layer (type)                   | Output Shape     | Param # |
|--------------------------------|------------------|---------|
| conv1d_1 (Conv1D)              | (None, 8996, 32) | 192     |
| max_pooling1d_1 (MaxPooling1D) | (None, 4498, 32) | 0       |
| conv1d_2 (Conv1D)              | (None, 4494, 32) | 5152    |
| max_pooling1d_2 (MaxPooling1D) | (None, 2247, 32) | 0       |
| conv1d_3 (Conv1D)              | (None, 2243, 64) | 10304   |
| max_pooling1d_3 (MaxPooling1D) | (None, 1121, 64) | 0       |
| conv1d_4 (Conv1D)              | (None, 1117, 64) | 20544   |
| max_pooling1d_4 (MaxPooling1D) | (None, 558, 64)  | 0       |
| conv1d_5 (Conv1D)              | (None, 554, 128) | 41088   |
| max_pooling1d_5 (MaxPooling1D) | (None, 277, 128) | 0       |
| conv1d_6 (Conv1D)              | (None, 273, 128) | 82048   |
| max_pooling1d_6 (MaxPooling1D) | (None, 136, 128) | 0       |
| dropout_1 (Dropout)            | (None, 136, 128) | 0       |
| conv1d_7 (Conv1D)              | (None, 132, 256) | 164096  |
| max_pooling1d_7 (MaxPooling1D) | (None, 66, 256)  | 0       |
| conv1d_8 (Conv1D)              | (None, 62, 256)  | 327936  |
| max_pooling1d_8 (MaxPooling1D) | (None, 31, 256)  | 0       |
| dropout_2 (Dropout)            | (None, 31, 256)  | 0       |
| conv1d_9 (Conv1D)              | (None, 27, 512)  | 655872  |
| max_pooling1d_9 (MaxPooling1D) | (None, 13, 512)  | 0       |
| dropout_3 (Dropout)            | (None, 13, 512)  | 0       |
| conv1d_10 (Conv1D)             | (None, 9, 512)   | 1311232 |
| flatten_1 (Flatten)            | (None, 4608)     | 0       |
| dense_1 (Dense)                | (None, 128)      | 589952  |
| dropout_4 (Dropout)            | (None, 128)      | 0       |
| dense_2 (Dense)                | (None, 32)       | 4128    |
| dense_3 (Dense)                | (None, 4)        | 132     |
| =====                          |                  |         |
| Total params: 3,212,676        |                  |         |
| Trainable params: 3,212,676    |                  |         |
| Non-trainable params: 0        |                  |         |

Figure 4-21: Network Summary

### 3. 1st proposal: Adding normalized data as input to the original CNN architecture

In our first proposal, as we mention previously, we denoised ECG signals where we use the same CNN architecture (Figure 4-21), however with filter size equal to three so the model can be compatible with our new data(9s ECG):

- Preparing data

```
1 import pywt
2 import scipy.io as sio
3 from skimage.restoration import denoise_wavelet
4 data = []
5 #for every file in the training directory
6 for i in range(size):
7     #load file
8     ECG = sio.loadmat(mypath + mats3[i])
9     #get ECG column
10    dummy = ECG['val'][0,:]
11    #denoise using wavelet
12    ECG_Denoised = denoise_wavelet(ECG,method="BayesShrink",
13    mode="soft",wavelet_levels=3,wavelet="sym5")
14    #append denoised ECG to the new database
15    data.append(ECG_Denoised)
```

Listing 4.7: Denoised and normalized signal as input

- Training

```
1 import pywt
2 import scipy.io as sio
3 from skimage.restoration import denoise_wavelet
4 data = []
5 #for every file in the training directory
6 for i in range(size):
7     #load file
8     ECG = sio.loadmat(mypath + mats3[i])
9     #get ECG column
10    dummy = ECG['val'][0,:]
11    #denoise using wavelet
12    ECG_Denoised = denoise_wavelet(ECG,method="BayesShrink",
13    mode="soft",wavelet_levels=3,wavelet="sym5")
14    #append denoised ECG to the new database
15    data.append(ECG_Denoised)
```

```
110 from keras.layers import Conv1D, MaxPool1D, Dropout, Flatten, Dense,
111     Input, concatenate, MaxPooling1D
112 from keras.models import Model, Sequential
113 def create_model():
114     model = Sequential()
115     model.add(Conv1D(32, 3, activation='relu', input_shape=(9000, 1)))
116     model.add(MaxPooling1D())
117     model.add(Conv1D(32, 3, activation='relu'))
118     model.add(MaxPooling1D())
119     model.add(Conv1D(64, 3, activation='relu'))
120     model.add(MaxPooling1D())
121     model.add(Conv1D(64, 3, activation='relu'))
122     model.add(MaxPooling1D())
123     model.add(Conv1D(128, 3, activation='relu'))
124     model.add(MaxPooling1D())
125     model.add(Dropout(0.5))
126     model.add(Conv1D(128, 3, activation='relu'))
127     model.add(Conv1D(128, 3, activation='relu'))
128     model.add(MaxPooling1D())
129     model.add(Conv1D(256, 3, activation='relu'))
130     model.add(MaxPooling1D())
131     model.add(Conv1D(256, 3, activation='relu'))
132     model.add(MaxPooling1D())
133     model.add(Conv1D(512, 3, activation='relu'))
134     model.add(MaxPooling1D())
135     model.add(Dropout(0.5))
136     model.add(Conv1D(512, 3, activation='relu'))
137     model.add(Flatten())
138     model.add(Dense(10, kernel_initializer='normal', activation='relu'))
139     model.add(Dropout(0.5))
140     model.add(Dense(10, kernel_initializer='normal', activation='relu'))
141     model.add(Dense(number_of_classes, kernel_initializer='normal', activation='softmax'))
142     opt = optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08)
143     model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
144     return model
```

Figure 4-22: Training CNN with denoised data

#### 4. 2nd proposal: Adding filtered data and injection of features as input to the CNN architecture

After long researches on the best way, we can merge data. Finally, we have reached that the best way is the concatenation in the fully connected layer.

Where we concatenate representative HRV features with features obtained by the CNN feature extractor, we make a data merging and we feed it to the fully connected layer.

The illustrative code of our proposal:

```
1 #inputs (ECG and Features)
2 sequence = Input(shape=(2700, 1), name='Sequence')
3 features = Input(shape=(52,), name='Features')
4 #Extract features from ECG
5 conv = Sequential()
6 conv.add(Conv1D(32, 3, activation='relu', input_shape=(2700, 1))
7 )
8 conv.add(MaxPool1D(2))
9 conv.add(Conv1D(32, 3, activation='relu'))
10 conv.add(MaxPool1D(2))
11 conv.add(Conv1D(64, 3, activation='relu'))
12 conv.add(MaxPool1D(2))
13 conv.add(Conv1D(64, 3, activation='relu'))
14 conv.add(MaxPool1D(2))
15 conv.add(Conv1D(128, 3, activation='relu'))
16 conv.add(MaxPool1D(2))
17 conv.add(Conv1D(128, 3, activation='relu'))
18 conv.add(MaxPool1D(2))
19 conv.add(Dropout(0.5))
20 conv.add(Conv1D(256, 3, activation='relu'))
21 conv.add(MaxPool1D(2))
22 conv.add(Conv1D(256, 3, activation='relu'))
23 conv.add(MaxPool1D(2))
24 conv.add(Dropout(0.5))
```



```

24 conv.add(Conv1D(512, 3, activation='relu'))
25 conv.add(MaxPool1D(2))
26 conv.add(Dropout(0.5))
27 conv.add(Conv1D(512, 3, activation='relu'))
28 #Flatten for 1D Array
29 conv.add(Flatten())
30 part1 = conv(sequence)
31 #Concatenate features with Obtained cnn features
32 merged = concatenate([part1, features])
33 #Start training
34 final = Dense(128, activation='relu')(merged)
35 final = Dropout(0.5)(final)
36 final = Dense(32, activation='relu')(final)
37 final = Dense(4, activation='softmax')(final)
38 model = Model(inputs=[sequence, features], outputs=[final])
39 opt = optimizers.Adam(learning_rate=0.0001)
40 model.compile(loss='categorical_crossentropy', optimizer=opt,
               metrics=['accuracy'])

```

Listing 4.8: Proposed 02 : Merged data

| Layer (type)                | Output Shape      | Param # | Connected to                       |
|-----------------------------|-------------------|---------|------------------------------------|
| Sequence (InputLayer)       | [(None, 2700, 1)] | 0       |                                    |
| sequential (Sequential)     | (None, 512)       | 1571872 | Sequence[0][0]                     |
| Features (InputLayer)       | [(None, 52)]      | 0       |                                    |
| concatenate (Concatenate)   | (None, 564)       | 0       | sequential[0][0]<br>Features[0][0] |
| dense (Dense)               | (None, 128)       | 72320   | concatenate[0][0]                  |
| dropout_3 (Dropout)         | (None, 128)       | 0       | dense[0][0]                        |
| dense_1 (Dense)             | (None, 32)        | 4128    | dropout_3[0][0]                    |
| dense_2 (Dense)             | (None, 4)         | 132     | dense_1[0][0]                      |
| Total params: 1,648,452     |                   |         |                                    |
| Trainable params: 1,648,452 |                   |         |                                    |
| Non-trainable params: 0     |                   |         |                                    |

Figure 4-23: Proposed 02 : Network summary

### 4.3.6 Training and evaluating models

To train our model, we choose 10 fold cross validation over 200 epochs, for each epoch we save the accuracy, loss, validation accuracy and validation loss at the end, we print a global report about the training using `classification_report` function from keras python.

```
1 def train_and_evaluate__model(model, X_train, Y_train, X_val, Y_val,
2     i):
3     #to use the model with our data
4     hist = model.fit(X_train, Y_train, epochs=100, batch_size=30,
5     validation_data=(X_val, Y_val), shuffle=True)
6     #save training history in csv file (loss and acc)
7     pd.DataFrame(hist.history).to_csv(path_or_buf='History ' + str(i)
8     ) + '.csv')
9     #function to save model
10    model.save('my_model ' + str(i) + '.h5')
11    #test with validation data
12    predictions = model.predict(X_val)
13    #accuracy of model
14    score = accuracy_score(change(Y_val), change(predictions))
15    print(score)
16    #classification repport to return f1 score precision, recall
17    target_names = ['Normal', 'AF', 'Other', 'Noisy']
18    print(classification_report(change(Y_val), change(predictions)),
19    target_names)
20    #from binary output to dicimal
21    df = pd.DataFrame(change(predictions))
22    df.to_csv(path_or_buf= str(format(score, '.4f')) + str(i) + '.
23    csv')
```

```
19    #save confusion matrix
20    pd.DataFrame(confusion_matrix(change(Y_val), change(predictions)
21    )).to_csv(
22        path_or_buf=str(format(score, '.4f'))+ str(i) + '.csv',
23        index=None,
24        header=None)
```

```

23 #split data into 10 folds with shuffling
24 skf = StratifiedKFold(n_splits=10,shuffle=True)
25 target_train = target_train.reshape(size,)
26
27 for i, (train_index, test_index) in enumerate(skf.split(X,
    target_train)):
28     print("TRAIN:", train_index, "TEST:", test_index)
29     X_train = X1[train_index, :]
30     Y_train = Label_set[train_index, :]
31     X_val = X1[test_index, :]
32     Y_val = Label_set[test_index, :]
33     #call cnn model
34     model = create_model()
35     #train and evaluate
36     train_and_evaluate__model(model, X_train, Y_train, X_val, Y_val, i
    )

```

Listing 4.9: Train and evaluate models

However, in the second proposition (View summary in figure4-23), we have two inputs, thus the fit model differs from other models :

```

1 #input the train data and train features
2 hist = model.fit([X_train,features_train], Y_train,
3 #to test: same thing input test data and test features
4 validation_data=([X_val,features_test], Y_val), batch_size=30,
    epochs=100, shuffle=True, callbacks=[checkpointer])
5 predictions = model.predict([X_val,features_test])

```

Listing 4.10: Train and evaluate 2nd proposal model

### 4.3.7 Model deployment: Visualization, Interpretation, Use

As patient monitoring is a necessity demanded by cardiologists, so that they can make the decision on the appropriate therapy, we have developed a dashboard application for the benefit of the doctor so that he can register, control and follow patients any where and anytime.

Also, as it is important for us to use our CNN model and create ECG reports, we make a page on the application for ECG records visualization and interpretation and CNN model use.

#### 4.3.7.1 CNN model deployment

After training, we obtain **.h5** file, we have used this file in our application to test on ECG test set:

```
1 from tensorflow.keras.models import load_model
2 #Open model in our flask app and allow it to predict
3 MODEL_PATH = 'model.h5'
4 model = load_model(MODEL_PATH)
5 model._make_predict_function()
```

Listing 4.11: Load .h5 model in flask

After opening, now we can use it:

```
1 @app.route('/Upload_file', methods=['GET', 'POST'])
2 def classification():
3     if request.method == 'POST':
4         #get the .mat file that we want to test in from user
5         file = request.files['inputFile']
6         mypath = r'C:\Users\CasaPhone\Desktop/test/'
7         #open the file
8         data = sio.loadmat(mypath + file.filename)
9         #read ECG column
10        data = data['val'][0, :]
11        #Create 9s segments
12        segments = Segmentation(data, duration, dataOverlap)
```

```

13     #result list for saving final classes
14     result=[]
15     #for each 9s in the signal
16     for i in segments:
17         #preprocess the segment
18         denoisedECG = Denoise(i)
19         #make data in wanted format for 1D CNN
20         test=[]
21         test.append(denoisedECG)
22         test = np.expand_dims(test, axis=2)
23         #predict
24         preds = model.predict(test)
25         #change binary result to number
26         res = change(preds)
27         test_res=""
28         if res == 0:
29             test_res = "Normal"
30         elif res == 1:
31             test_res = "Atrial fibrillation"
32         elif res == 2:
33             test_res = "Other rhythms"
34         else:
35             training_res ="Noisy signal"
36         result.append(training_res)
37     return result

```

Listing 4.12: Deploy model in flask

The output of this function is a list containing class of each 9s of the ECG records.

Example: a record of 9000 samples (30s) will be chopped into 4 segments with 4 classes :

```

1 ['Atrial fibrillation', 'Atrial fibrillation', 'Atrial fibrillation'
  , 'Atrial fibrillation']

```

### 4.3.7.2 Visualization, Interpretation, Use

Each doctor can be registered to create his new account.

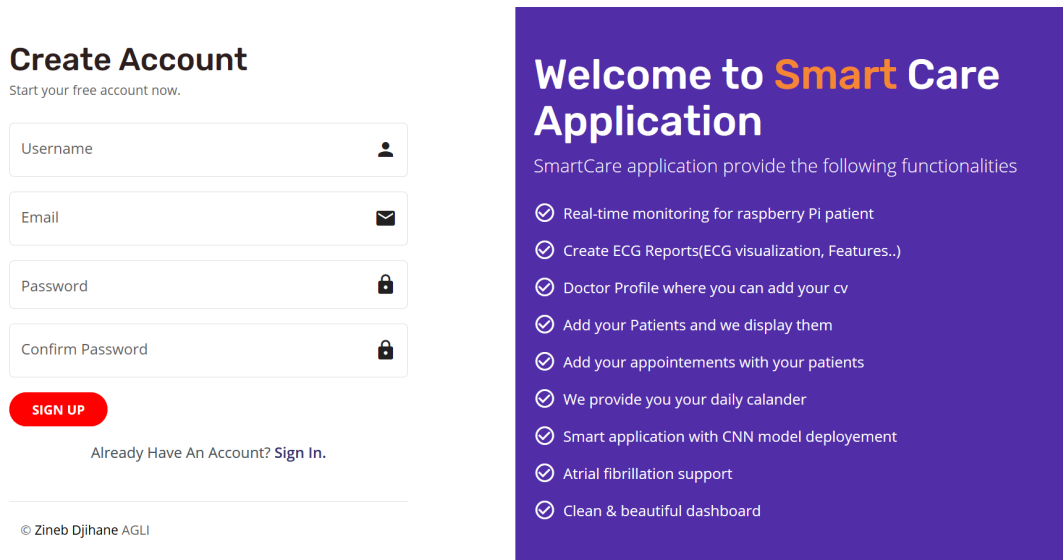


Figure 4-24: Profile's doctor interface

Each doctor can login to enter to his dashboard

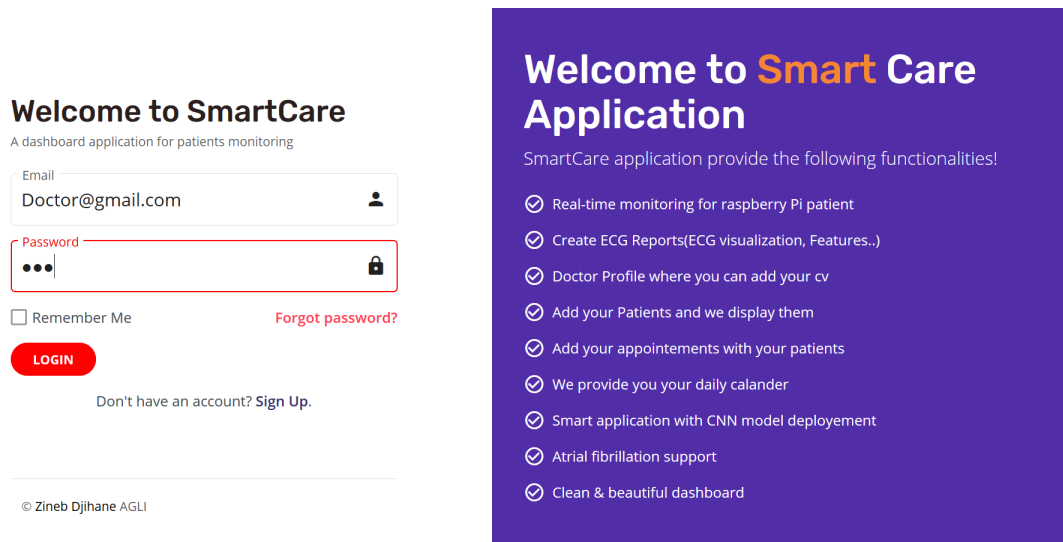


Figure 4-25: Profile's doctor interface

He can Edit his personal information or add his CV:

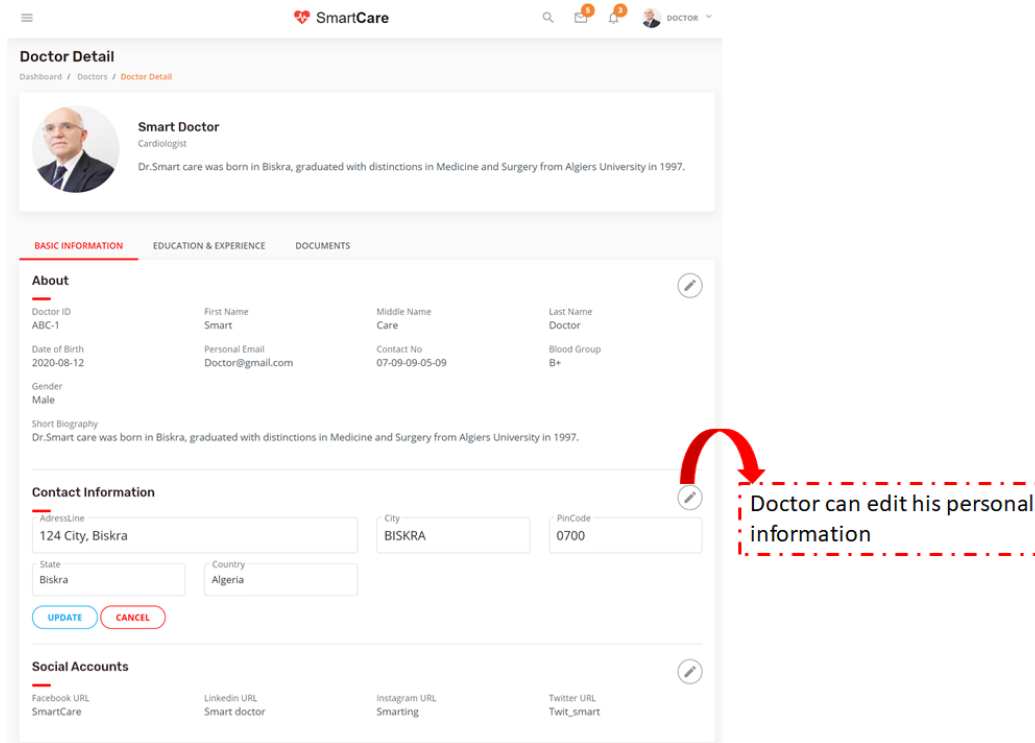


Figure 4-26: Profile's doctor interface

He can also add patients and view their data history in a table or a list:

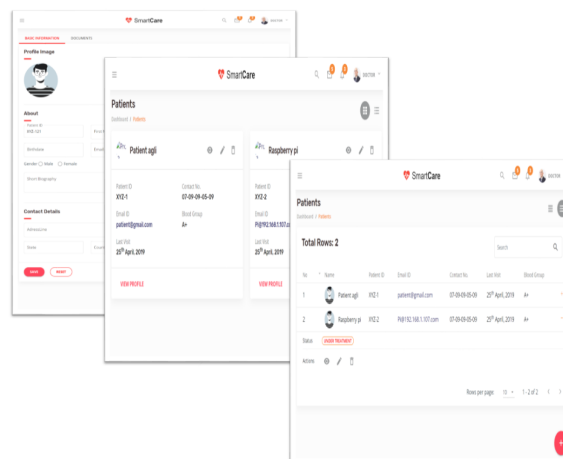


Figure 4-27: Add patients, list and table of doctor's patients interfaces

We also provide a dashboard for doctor so he follows daily progress for his patients

:

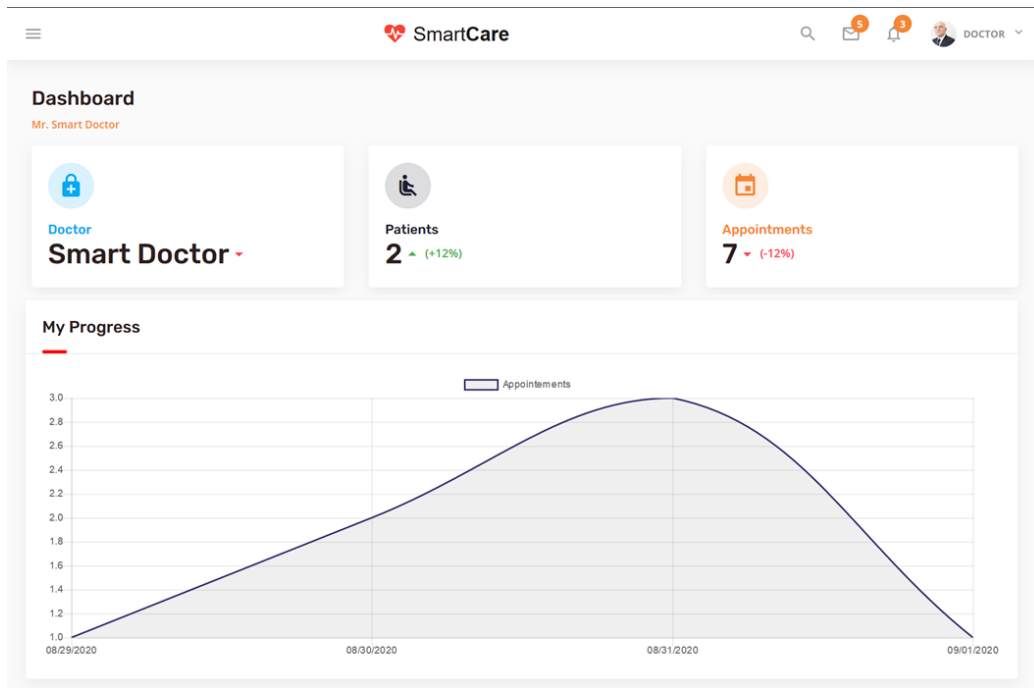


Figure 4-28: Doctor dashboard

Of course, every doctor has appointments with his patients, so we make page for adding appointments, delete and modification and viewing his appointments list:

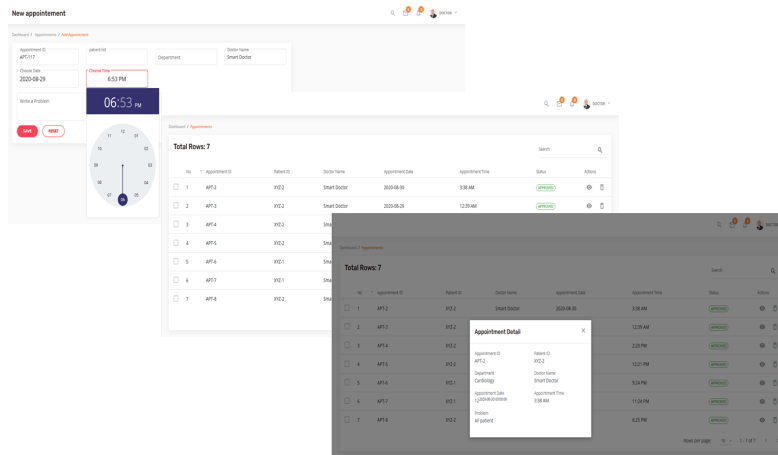


Figure 4-29: Appointments pages



He can continuously follow up and in a real-time his patients (them who wear cooking hacks platform) from wealthy dashboard, and an ECG report creates every 1 hour

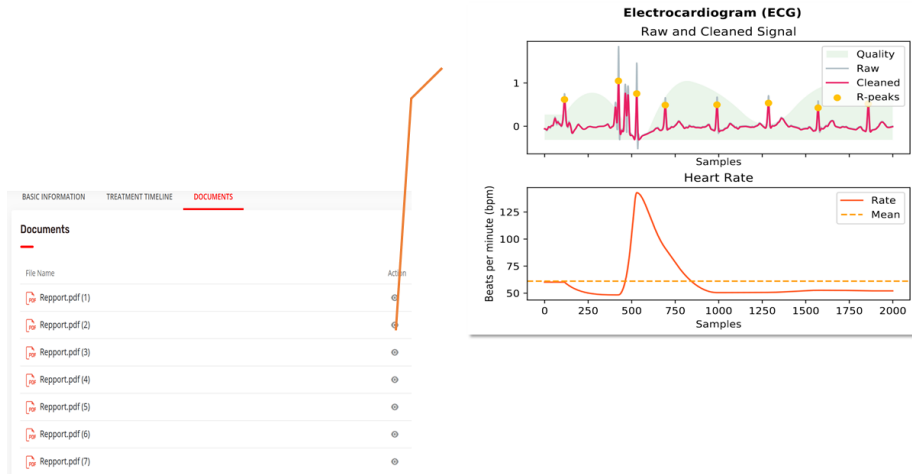


Figure 4-30: Patients daily rapports

Due to the importance of remote access to patient information, we made a web page for anyone permitted that wants to create an ECG report (visualisation and features) from (.mat file) also we have deployed our CNN model in this page.

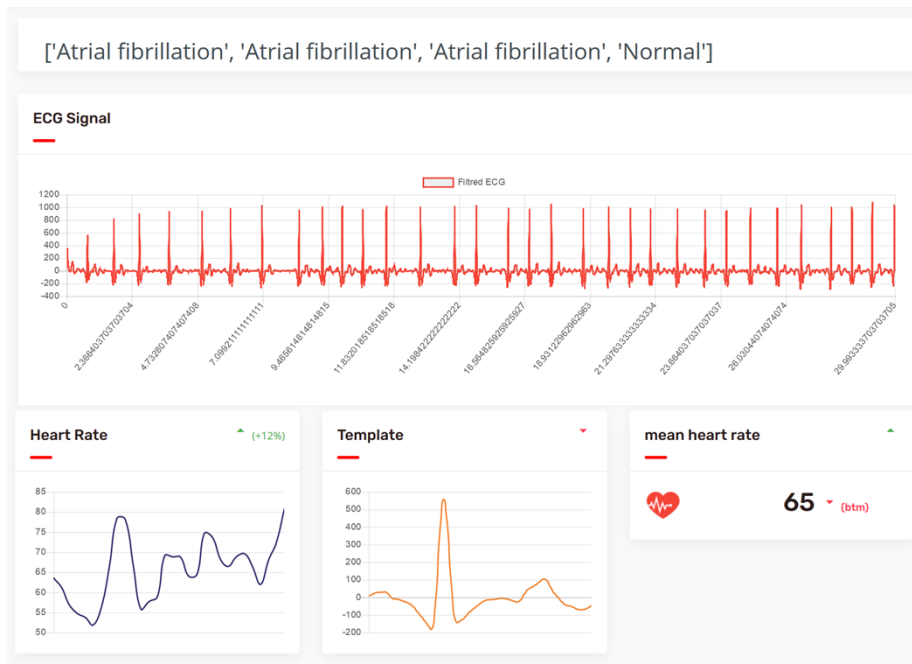


Figure 4-31: Dashboard for ECG reports and classification

## 4.4 Results and discussion

### 4.4.1 Training using 1-D CNN with 30s ECG length

We have trained raw ECG signals of 30s (9000 samples) with the implemented 1-D CNN of [27] and 10 fold cross validation.

```
Epoch 1/100
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
9119/9119 - 10s - loss: 0.9795 - accuracy: 0.5812 - val_loss: 0.8701 - val_accuracy: 0.5947
Epoch 2/100
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
9119/9119 - 9s - loss: 0.8472 - accuracy: 0.6161 - val_loss: 0.8038 - val_accuracy: 0.6420
.
.
9119/9119 - 9s - loss: 0.0914 - accuracy: 0.9648 - val_loss: 0.8514 - val_accuracy: 0.8314
Epoch 98/100
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
9119/9119 - 9s - loss: 0.1027 - accuracy: 0.9639 - val_loss: 0.7941 - val_accuracy: 0.8343
Epoch 99/100
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
9119/9119 - 9s - loss: 0.0936 - accuracy: 0.9641 - val_loss: 0.8275 - val_accuracy: 0.8284
Epoch 100/100
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
9119/9119 - 9s - loss: 0.0946 - accuracy: 0.9644 - val_loss: 0.7579 - val_accuracy: 0.8422
```

Figure 4-32: Model ECG 30s report

#### 1. Training and validation accuracy

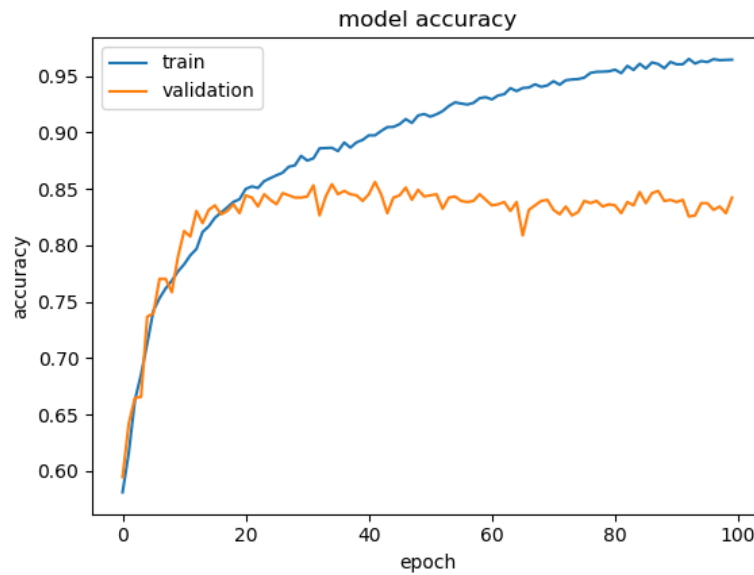


Figure 4-33: 30s ECG: model accuracy

2. **Training and validation loss** The mean training and the validation loss of the model are shown in figure 4-34.

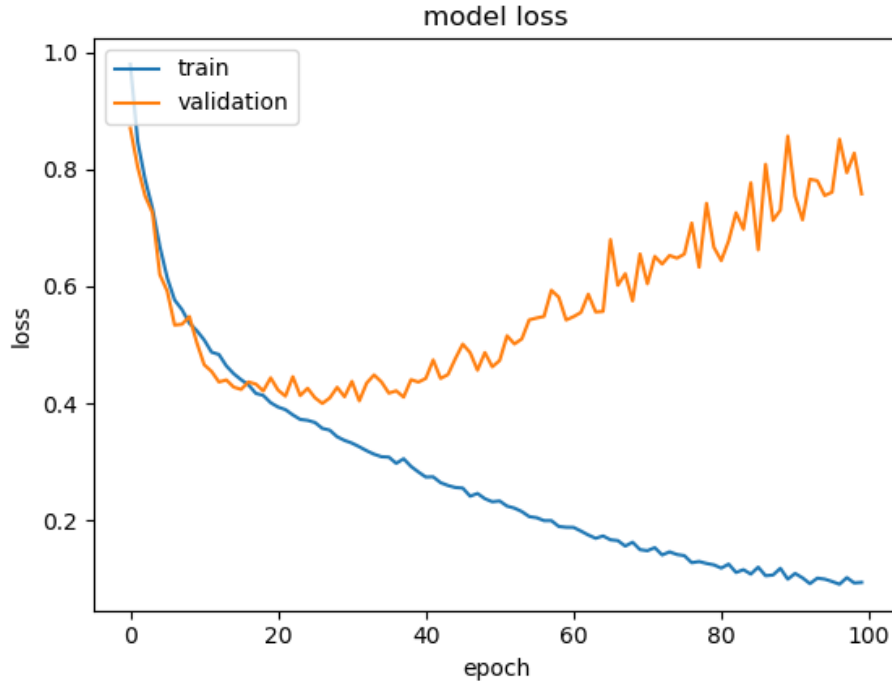


Figure 4-34: 30s ECG: model loss

3. **Classification report (validation set)**

|              | precision | recall | f1-score |
|--------------|-----------|--------|----------|
| 0            | 0.89      | 0.93   | 0.91     |
| 1            | 0.74      | 0.76   | 0.75     |
| 2            | 0.80      | 0.71   | 0.75     |
| 3            | 0.21      | 0.60   | 0.32     |
| accuracy     |           |        | 0.84     |
| macro avg    | 0.66      | 0.75   | 0.68     |
| weighted avg | 0.85      | 0.84   | 0.84     |

Figure 4-35: 30s ECG: model report where 0-Normal, 1-AF, 2-Others, 3-Noisy

## 4.4.2 1st Proposal

We have trained preprocessed ECG signals of 9s (2700 samples) with our 1-D CNN and 10 fold cross-validation over 300 epochs.

```
Epoch 1/300
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 28s - loss: 0.7956 - accuracy: 0.6460 - val_loss: 0.6807 - val_accuracy: 0.7183
Epoch 2/300
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 28s - loss: 0.6205 - accuracy: 0.7445 - val_loss: 0.5781 - val_accuracy: 0.7712
Epoch 3/300
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 28s - loss: 0.5518 - accuracy: 0.7811 - val_loss: 0.5773 - val_accuracy: 0.7691
Epoch 4/300
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 28s - loss: 0.5167 - accuracy: 0.7953 - val_loss: 0.4909 - val_accuracy: 0.8032
.
Epoch 298/300
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 29s - loss: 0.0343 - accuracy: 0.9881 - val_loss: 0.0454 - val_accuracy: 0.9853
Epoch 299/300
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 29s - loss: 0.0342 - accuracy: 0.9882 - val_loss: 0.0470 - val_accuracy: 0.9840
Epoch 300/300
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 29s - loss: 0.0356 - accuracy: 0.9875 - val_loss: 0.0414 - val_accuracy: 0.9864
```

Figure 4-36: First fold results

1. **Training and validation accuracy** The mean training and the validation accuracy of the model are shown in figure 4-37.

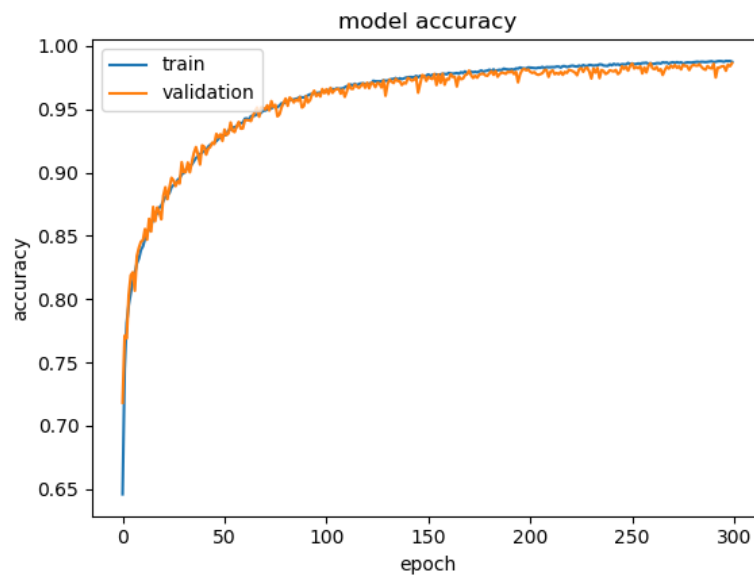


Figure 4-37: 1st Proposal: model accuracy

2. **Training and validation loss** The mean training and the validation loss of the model are shown in figure 4-38.

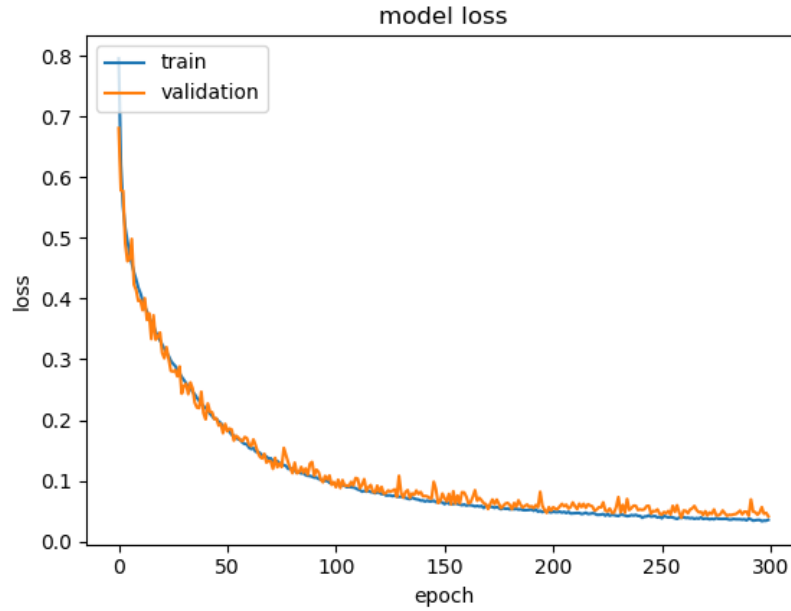


Figure 4-38: 1st Proposal: model loss

3. **Classification report (validation set)**

|              | precision | recall | f1-score |
|--------------|-----------|--------|----------|
| 0            | 0.99      | 0.99   | 0.99     |
| 1            | 0.97      | 0.99   | 0.98     |
| 2            | 0.99      | 0.97   | 0.98     |
| 3            | 0.78      | 1.00   | 0.88     |
| accuracy     |           |        | 0.99     |
| macro avg    | 0.93      | 0.99   | 0.96     |
| weighted avg | 0.99      | 0.99   | 0.99     |

Figure 4-39: 1st Proposal: model report where 0-Normal, 1-AF, 2-Others, 3-Noisy

### 4.4.3 2nd proposal

In the second proposition, we create a new CNN architecture by combining hand-crafted features and the features resulting from the CNN.

We trained a 9s ECG length with 100 epochs and 10 fold cross validation.

```
TRAIN: [ 0 2 3 ... 99997 99998 99999] TEST: [ 1 12 19 ... 99983 99990 99994]
Epoch 1/200
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 31s - loss: 0.7935 - accuracy: 0.6490 - val_loss: 0.6703 - val_accuracy: 0.7098
Epoch 2/200
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 30s - loss: 0.6336 - accuracy: 0.7334 - val_loss: 0.5893 - val_accuracy: 0.7709
Epoch 3/200
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 29s - loss: 0.5558 - accuracy: 0.7757 - val_loss: 0.5230 - val_accuracy: 0.7947
.
.
Epoch 198/200
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 29s - loss: 0.0257 - accuracy: 0.9910 - val_loss: 0.1192 - val_accuracy: 0.9687
Epoch 199/200
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 29s - loss: 0.0230 - accuracy: 0.9923 - val_loss: 0.1439 - val_accuracy: 0.9633
Epoch 200/200
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
3000/3000 - 29s - loss: 0.0231 - accuracy: 0.9923 - val_loss: 0.1278 - val_accuracy: 0.9680
```

Figure 4-40: First fold results

1. **Training and validation accuracy:** The mean training and the validation accuracy of the model are shown in figure 4-41.

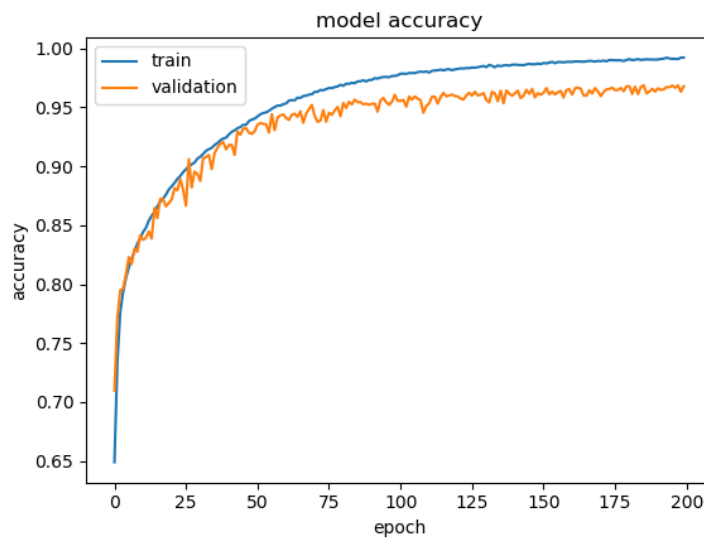


Figure 4-41: 2nd proposal: model accuracy

2. **Training and validation loss:** The mean training and the validation loss of the model are shown in figure 4-42.

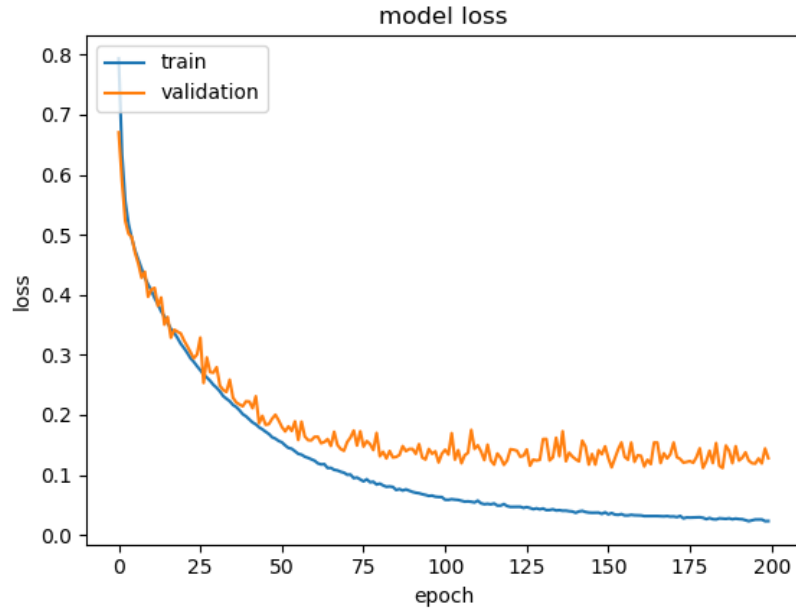


Figure 4-42: 2nd proposal: model loss

3. **Classification report (validation set):**

|              | precision | recall | f1-score |
|--------------|-----------|--------|----------|
| 0            | 0.96      | 0.99   | 0.98     |
| 1            | 0.97      | 0.97   | 0.97     |
| 2            | 0.98      | 0.92   | 0.95     |
| 3            | 0.94      | 0.91   | 0.92     |
| accuracy     |           |        | 0.97     |
| macro avg    | 0.96      | 0.95   | 0.95     |
| weighted avg | 0.97      | 0.97   | 0.97     |

Figure 4-43: 2nd proposal: model report where 0-Normal, 1-AF, 2-Others, 3-Noisy

#### 4.4.4 Discussion

We made several tests on our models. We can resume the results in the following table 4.6 :

| Method  | Length | F1 Score (N) | F1 Score (AF) | F1 Score(O) | F1 Score(~) | Average F1   |
|---|--------|--------------|---------------|-------------|-------------|--------------|
| 1-D CNN   | 30s    | 91%          | 75%           | 75%         | 32%         | 68.25%       |
| 1-D CNN   | 9s     | 97%          | 96%           | 93%         | 63%         | 87.25%       |
| <b>1st proposal:<br/>1-D CNN<br/>+<br/>DWT</b>                    | 9s     | <b>99%</b>   | <b>98%</b>    | <b>98%</b>  | <b>88%</b>  | <b>96%</b>   |
| <b>2nd proposal:<br/>1-D CNN<br/>+<br/>features<br/>+<br/>DWT</b> | 9s     | <b>98%</b>   | <b>97%</b>    | <b>95%</b>  | <b>92%</b>  | <b>95.5%</b> |

Table 4.6: Resume of the results obtained in the validation set

In our study, we have shown the superiority of the 1D-CNN model for AF detection produces high performance with short-term signal ECG, great ability of the automatic feature extraction, with low complexity.

Our model not only robust in the training and validation sets but also in the small test set that we have made it knows almost all of records.

Our proposed 1D-CNN methods for detecting atrial fibrillation can distinguish AF signal with other signals over 99% classification accuracy using 1st proposal and over 97% with the second proposal comparing to the original 1-D CNN that achieved 75%.

Our second proposal was an experience that we are still working to improve it to see whether hand-crafted features will make a difference and affect results positively.



## 4.5 Conclusion

This last chapter has described the biggest steps we have done to implement our proposals. We have started by data; its acquisition, streaming and storing. The next step has shown data segmentation and denoising using the DWT after that we spotlight on how did we use the Neurokit2 library to extract affluent features. Then we have recited the implementation of our two proposals where we achieved to improve a 1-D CNN architecture by 99% and 97% accuracy in the first and second propositions accordingly. In the end, we have implemented a dashboard application where we deployed our first CNN model, not only for the benefits of doctors but also for anyone who wants to make ECG reports from the Physionet dataset.

# Conclusion

Early detection of atrial fibrillation can save the lives of millions of people, and help to provide prompt treatment to avoid erroneous diagnosis.

The ECG is the most used tool for recognizing arrhythmias because any abnormality is reflected in the ECG signals. However, the visual assessment of ECG signals is a difficult and time-consuming task. Therefore, conceiving an efficient and accurate system that ensures permanent patient monitoring will make the diagnosis of cardiac arrhythmia such as AF easier and effective.

To confront this disease, we have proposed in this study, a new system based on an improved and optimized 1-D CNN architecture using two methods, the first improvement was about segmenting and denoising ECG data and it has achieved an accuracy of 99% on the validation set, the second one consists of injecting hand-crafted features as input to the fully connected layer with trained-on ECG features where it attained 97% on the validation set. Both of our proposals have shown high accuracy and robustness to detect atrial fibrillation only in 9s of ECG record.

On the other hand, long-lasting and uninterrupted streaming of patient ECG data is an important monitoring task. For this reason, we came up with the idea of creating a connected system that allows permanent data recording and real-time dashboard monitoring. Not only this, we also have created a web page for ECG visualization, features extraction and for normal, atrial fibrillation, other rhythms and noisy signals classification.

In this project, we have discovered a lot of things about ECG in data science world, now we can say that having a powerful CNN architecture is not enough without powerful possessing specially if you have small dataset. We have seen that many

CNN architectures give staining results in atrial fibrillation, it is not limited in one architecture. Thus, in the future, we project other combination types of models or data such as feeding the CNN output into another classifier alongside the hand-crafted features or combining two classifiers where the first takes ECG signal and the second takes features. Also studying how choosing the right percentage of ECG overlapping affects on results could be an interesting future research direction.

On the other hand, and because of the lack of preprocessing and processing tools for the cooking hacks platform, making preprocessing functions that facilitate its use might considered as an added value in this domain.

# Bibliography

- [1] Anaconda (python distribution) — Wikipedia, the free encyclopedia. [https://fr.wikipedia.org/wiki/Anaconda\\_\(Python\\_distribution\)](https://fr.wikipedia.org/wiki/Anaconda_(Python_distribution)), 2020. [Online; accessed 15-08-2020].
- [2] Atrial Fibrillation, [cardiosmart.org. https://www.cardiosmart.org/heart-conditions/atrial-fibrillation](https://www.cardiosmart.org/heart-conditions/atrial-fibrillation), 2020. [Online; accessed 14-Feb-2020].
- [3] C++ — Wikipedia, The Free Encyclopedia. [https://en.wikipedia.org/wiki/C++](https://en.wikipedia.org/wiki/C%2B%2B), 2020. [Online; accessed 15-08-2020].
- [4] Flask (web framework) — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/Flask\\_\(web\\_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework)), 2020. [Online; accessed 15-08-2020].
- [5] Heart rate variability. <https://imotions.com/blog/heart-rate-variability/>, 2020. [Online; accessed 02-Feb-2020].
- [6] L'électrocardiogramme. <http://entraide-esi-ide.com/lelectrocardiogramme/>, 2020. [Online; accessed 02-Feb-2020].
- [7] "Persistent Atrial Fibrillation: Symptoms, Treatment, and Prognosis, healthline. <https://www.healthline.com/health/atrial-fibrillation/persistent#symptoms>, 2020. [Online; accessed 20-Feb-2020].
- [8] Pycharm — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/wiki/PyCharm>, 2020. [Online; accessed 15-08-2020].
- [9] Python (programming language) — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)), 2020. [Online; accessed 15-08-2020].
- [10] Sqlite — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/wiki/SQLite>, 2020. [Online; accessed 15-08-2020].
- [11] The ultimate ECG book, ecg & echo. <https://ecgwaves.com/>, 2020. [Online; accessed 02-Feb-2020].

- [12] Aakash Healthcare. <https://aakashhealthcare.files.wordpress.com/2019/03/heart-arrhythmia-treatment-in-delhi.jpg>, 2020. [Online; accessed 02-Feb-2020].
- [13] abdelhamid djeffal. Model evaluation techniques. <http://www.abdelhamid-djeffal.net/>, 2020. [Online; 21-august-2020].
- [14] AliveCor official website. Kardia. <https://www.alivecor.com/>, 2020. [Online; accessed 02-Feb-2020].
- [15] American heart association. Arrhythmia. [www.heart.org](http://www.heart.org), 2020. [Online; accessed 02-Feb-2020].
- [16] Apple official website. Heart health notifications on your apple watch. <https://support.apple.com/en-us/HT208931>, 2020. [Online; accessed 02-Feb-2020].
- [17] Juan Benezet-Mazuecos, Camila S García-Talavera, and José Manuel Rubio. Smart devices for a smart detection of atrial fibrillation. *Journal of Thoracic Disease*, 10(Suppl 33):S3824, 2018.
- [18] Sensors C. 12-Lead ECG Placement Guide with Illustrations — Cables and Sensors, cables and sensors. <https://www.cablesandsensors.com/pages/12-lead-ecg-placement-guide-with-illustrations>, 2020. [Online; accessed 03-March-2020].
- [19] Sensors C. Dérivations précordiales unipolaires de Wilson. [http://www.ednes.com/ecg\\_ex/phyder4.htm](http://www.ednes.com/ecg_ex/phyder4.htm), 2020. [Online; accessed 03-March-2020].
- [20] Cooking hacks. e-health sensor platform v2.0 for arduino and raspberry pi [biometric / medical applications]. <https://archive.physionet.org/physiotools/>, 2020. [Online; accessed 21-august-2020].
- [21] Cooking hacks. Raspberry pi to arduino shields connection bridge. <https://www.cooking-hacks.com/documentation/tutorials/raspberry-pi-to-arduino-shields-connection-bridge.html>, 2020. [Online; 21-august-2020].
- [22] DRID ABOU BAKR SEDDIK. Pattern recognition for healthcare analytics, 2017. [Online; 21-august-2020].
- [23] Rémi Dubois. Application des nouvelles méthodes d'apprentissage à la détection précoce d'anomalies en électrocardiographie. *These de Doctorat de l'Université Pierre et Marie Curie*, 2004.
- [24] Oliver Faust, Edward J Ciaccio, and U Rajendra Acharya. A review of atrial fibrillation detection methods as a service. *International Journal of Environmental Research and Public Health*, 17(9):3093, 2020.

- [25] Flask documentation. Flask-sqlalchemy. <https://flask-sqlalchemy.palletsprojects.com/en/2.x/>, 2020. [Online; 21-august-2020].
- [26] Valentin Fuster, Lars E Rydén, Richard W Asinger, David S Cannom, Harry J Crijns, Robert L Frye, Jonathan L Halperin, G Neal Kay, Werner W Klein, Samuel Lévy, et al. Acc/aha/esc guidelines for the management of patients with atrial fibrillation: executive summary: a report of the american college of cardiology/american heart association task force on practice guidelines and the european society of cardiology committee for practice guidelines and policy conferences (committee to develop guidelines for the management of patients with atrial fibrillation) developed in collaboration with the north american society of pacing and electrophysiology. *Journal of the American College of Cardiology*, 38(4):1231–1265, 2001.
- [27] Chaur-Heh Hsieh, Yan-Shuo Li, Bor-Jiunn Hwang, and Ching-Hua Hsiao. Detection of atrial fibrillation using 1d convolutional neural network. *Sensors*, 20(7):2136, 2020.
- [28] Craig T January, L Samuel Wann, Joseph S Alpert, Hugh Calkins, Joaquin E Cigarroa, Joseph C Cleveland, Jamie B Conti, Patrick T Ellinor, Michael D Ezekowitz, Michael E Field, et al. 2014 aha/acc/hrs guideline for the management of patients with atrial fibrillation: a report of the american college of cardiology/american heart association task force on practice guidelines and the heart rhythm society. *Journal of the American College of Cardiology*, 64(21):e1–e76, 2014.
- [29] Hooman Kamel, Peter M Okin, Mitchell SV Elkind, and Costantino Iadecola. Atrial fibrillation and mechanisms of stroke: time for a new model. *Stroke*, 47(3):895–900, 2016.
- [30] Paulus Kirchhof, Stefano Benussi, Dipak Kotecha, Anders Ahlsson, Dan Atar, Barbara Casadei, Manuel Castella, Hans-Christoph Diener, Hein Heidbuchel, Jeroen Hendriks, et al. 2016 esc guidelines for the management of atrial fibrillation developed in collaboration with eacts. *European journal of cardio-thoracic surgery*, 50(5):e1–e88, 2016.
- [31] Jie Lian, Lian Wang, and Dirk Muessig. A simple method to detect atrial fibrillation using rr intervals. *The American journal of cardiology*, 107(10):1494–1497, 2011.
- [32] Fengying Ma, Jingyao Zhang, Wei Liang, and Jingyu Xue. Automated classification of atrial fibrillation using artificial neural network for wearable devices. *Mathematical Problems in Engineering*, 2020, 2020.
- [33] Michael V McConnell, Mintu P Turakhia, Robert A Harrington, Abby C King, and Euan A Ashley. Mobile health advances in physical activity, fitness, and atrial fibrillation: moving hearts. *Journal of the American College of Cardiology*, 71(23):2691–2701, 2018.

- [34] An Nguyen, Sardar Ansari, Mohsen Hooshmand, Kaiwen Lin, Hamid Ghanbari, Jonathan Gryak, and Kayvan Najarian. Comparative study on heart rate variability analysis for atrial fibrillation detection in short single-lead ecg recordings. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 526–529. IEEE, 2018.
- [35] Siti Nurmaini, Alexander Edo Tondas, Annisa Darmawahyuni, Muhammad Nafal Rachmatullah, Radiyati Umi Partan, Firdaus Firdaus, Bambang Tutuko, Ferlita Pratiwi, Andre Herviant Juliano, and Rahmi Khoirani. Robust detection of atrial fibrillation from short-term electrocardiogram using convolutional neural networks. *Future Generation Computer Systems*, 113:304–317, 2020.
- [36] PhysioNet. Af classification from a short single lead ecg recording - the physionet computing in cardiology challenge 2017. <https://physionet.org/content/challenge-2017/1.0.0/>, 2020. [Online; 21-august-2020].
- [37] PhysioNet. physiobank. <https://archive.physionet.org/physiobank/>, 2020. [Online; accessed 21-august-2020].
- [38] PhysioNet. physiotools. <https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical.html>, 2020. [Online; accessed 02-Feb-2020].
- [39] PhysioNet. The research resource for complex physiologic signals. <https://physionet.org/>, 2020. [Online; 21-august-2020].
- [40] Francesca Pistoia, Simona Sacco, Cindy Tiseo, Diana Degan, Raffaele Ornello, and Antonio Carolei. The epidemiology of atrial fibrillation and stroke. *Cardiology clinics*, 34(2):255–268, 2016.
- [41] Alif Akbar Pranata and Dong-Seong Kim. Design and implementation of electrocardiogram device based on body area network (ban). , pages 170–171, 2016.
- [42] Python. Neurokit2 library. <https://neurokit2.readthedocs.io/en/latest/index.html>, 2020. [Online; 21-august-2020].
- [43] Python. Scipy. <https://www.scipy.org/>, 2020. [Online; 21-august-2020].
- [44] Beanbonyka Rim, Nak-Jun Sung, Sedong Min, and Min Hong. Deep learning in physiological signal data: A survey. *Sensors*, 20(4):969, 2020.
- [45] Paul A Rogers, Michael L Bernard, Christopher Madias, Sudarone Thihalolipavan, NA Mark Estes III, and Daniel P Morin. Current evidence-based understanding of the epidemiology, prevention, and treatment of atrial fibrillation. *Current problems in cardiology*, 43(6):241–283, 2018.
- [46] Sara Ross-Howe and Hamid R Tizhoosh. Atrial fibrillation detection using deep features and convolutional networks. In *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 1–4. IEEE, 2019.

- [47] Aswathy Velayudhan and Soniya Peter. Noise analysis and different denoising techniques of ecg signal-a survey. *IOSR journal of electronics and communication engineering*, 3:641–644, 2016.
- [48] World Health Organization. Cardiovascular diseases (cvds). <https://www.who.int/>, 2020. [Online; accessed 02-Feb-2020].
- [49] D George Wyse, Isabelle C Van Gelder, Patrick T Ellinor, Alan S Go, Jonathan M Kalman, Sanjiv M Narayan, Stanley Nattel, Ulrich Schotten, and Michiel Rienstra. Lone atrial fibrillation: does it exist? *Journal of the American College of Cardiology*, 63(17):1715–1723, 2014.
- [50] Massimo Zoni-Berisso, Fabrizio Lercari, Tiziana Carazza, and Stefano Domenicucci. Epidemiology of atrial fibrillation: European perspective. *Clinical epidemiology*, 6:213, 2014.