

Final Master Thesis

**Màster Universitari en Enginyeria Industrial
Especialitat biomèdica**

**Analysis of ECG signals using deep
neural networks**

REPORT

Author: Miguel Rueda Sotorra
Director: Jordi Fonollosa Magrinya
Alexandre Perera i Lluna
Date: September 2019



Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona



Summary

The study of the ECG signals is essential to detect several diseases. The present project aims to analyze the ECG signals in order to detect different types of heartbeats associated with arrhythmia, using data from the MIT-BIH Arrhythmia database. The proposed pipeline is the following: (a) design and program a data visualizer for the MIT-BIH Arrhythmia database; (b) reduce the dimensionality of the data using a Principal Component Analysis; and (c) compress the beats using a deep autoencoder.

The first step of the analysis consists of an approach to the MIT-BIH Arrhythmia database for better understanding on how the data is presented. For this purpose, an ECG data visualizer is designed. This application allows the user to select several graph settings and records of the database and plots the selected signals. Diverse graphic and GUI libraries are considered, such as WFDB, TkInter and Bokeh. But the package that best fits the requirements in this case is Dash, which includes both the graph and the user interface and updates the changed settings in real time.

The second step of the analysis consists of reducing the dimensionality of the data by means of a Principal Component Analysis. PCA has been performed for each one of the following types of heartbeats: normal beat (N); left bundle branch block beat (L); right bundle branch block beat (R); premature ventricular contraction (V); and atrial premature beat (A). The importance of every principal component for the shape of each beat is discussed in this section.

The final step is to compress the data using a deep autoencoder. The encoder pretends to represent an input heartbeat in a reduced number of neurons. The decoder reconstructs the initial beat using the same configuration of layers but arranged in reverse order. The first proposal is a convolutional autoencoder with 10 neurons. The obtained loss is 0.228 (22.8%). The correlation coefficient between the input vector and the output of the autoencoder is 0.99; 0.89; 0.96; 0.93; and 0.92 for beats N; L; R; V; and A, respectively. For further compression, the same autoencoder but using only 5 neurons has been trained as well. The loss results are also around 0.228, but the correlation coefficients are, respectively, 0.95; 0.86; 0.83; 0.89; and 0.71. The performance is worse in this case. Hence, the autoencoder with 10 neurons is better for the data compression.

A recurrent LSTM autoencoder is proposed for future work. It would allow both to use input data of variable length and to predict next steps in ECG sequences, which might improve the performance of the autoencoder and thus, the ECG beats classification for arrhythmia detection.

Contents

SUMMARY	3
CONTENTS	5
1. INTRODUCTION	7
1.1. Motivation	7
1.2. Objectives of the project and scope	7
1.3. State-of-the-art	8
1.4. Deep learning	11
1.4.1. Autoencoder	11
1.4.2. Convolutional neural networks (CNN)	12
1.4.3. Recurrent neural networks (RNN)	13
2. ECG SIGNALS VISUALIZER FOR MIT-BIH DATABASE	15
2.1. MIT-BIH Arrhythmia Database	15
2.2. Data display using WFDB and TkInter packages	17
2.3. Data display using Bokeh and TkInter packages	19
2.4. Data display using Dash	19
3. PRINCIPAL COMPONENT ANALYSIS (PCA)	21
3.1. Type N: Normal beat	25
3.2. Type L: Left bundle branch block beat	30
3.3. Type R: Right bundle branch block beat	35
3.4. Type V: Premature ventricular contraction	39
3.5. Type A: Atrial premature beat	44
4. DEEP AUTOENCODER	49
4.1. Convolutional autoencoder	49
4.2. Recurrent LSTM autoencoder proposal for future studies	59
CONCLUSIONS	60
BIBLIOGRAPHIC REFERENCES	61

1. Introduction

1.1. Motivation

The study of the biomedical signals is essential to understand how the human body works, and to detect and treat several diseases. Obtaining this kind of signals nowadays is simple, non-invasive and relatively cheap, and can be used to make medical decisions.

I have been working for four years so far in a company that studies biomedical signals and designs devices to monitor patients in hospitals. The main study in this company is the electroencephalogram (EEG): one device processes the EEG signal obtained from a patient and calculates parameters that indicate the level of consciousness of the person. This helps the doctors to monitor the depth of anesthesia. Other signals such as the electrocardiogram (ECG), that helps to study the heart rate and the cardiac output; or the electromyogram (EMG), that measures the muscular activity, are also monitored in other devices. The key point is that these signals are obtained with non-invasive and inexpensive methods, and give information that, when processed, can be very helpful to avoid problems such as overdosing of anesthesia (in the case of the depth of anesthesia monitor), that can cause postoperative problems.

Another area of interest in my professional career is the data science. The signals that I have been working on so far (mainly the EEG) are processed with traditional methods such as notch filtering or downsampling, and require, in general, considerable computational resources. Using more modern methods such as deep neural networks or the management of big data could be useful to find models that can be used, for example, to detect certain behaviors in the biomedical signals that could indicate that a patient has a disease.

1.2. Objectives of the project and scope

The main purpose of the project is to analyze the ECG signals using deep neural networks in order to extract information about different types of heartbeats. As some of these beats have been obtained from patients with arrhythmia, the development of a deep learning model using an autoencoder may help to detect this medical condition in patients by analyzing the raw ECG signal.

To accomplish this, an initial phase is proposed to introduce and understand the MIT-BIH

ECG arrhythmia database, which contains the data that will be used to train the model. The proposal is the development of a data visualizer with a user interface that allows the user to display the signals of the available recordings. Also, after this initial part, a principal component analysis will help to reduce the dimensionality of the data. The specific objectives are presented in the following lines:

- (1) Review of the different types of heartbeats included in the MIT-BIH Arrhythmia database.
- (2) Development of an application capable of displaying the signals of the MIT-BIH Arrhythmia database. The application shall include a user interface that allows the user to select several parameters related to the graphs that will be plotted.
- (3) Review the state-of-the-art related to analysis and classification of signals.
- (4) Reduce the dimensionality of the ECG signals included in the MIT-BIH Arrhythmia database. The technique used will be a principal component analysis.
- (5) Compress the ECG signals included in the MIT-BIH Arrhythmia database by designing and training an autoencoder.
- (6) Implement the previously mentioned solutions using Python programming language and its open-source libraries. For the development of the data visualizer, several packages will be considered, such as TkInter, WFDB, Bokeh and Dash; for the autoencoder, the Keras library, built on top of TensorFlow, will be used.

1.3. State-of-the-art

According to the general methodology to design an automatic system for arrhythmia identification, there are 4 tasks that need to be done: (a) signal pre-processing; (b) beats segmentation; (c) feature extraction; and (d) classification of beats.

The signal pre-processing includes digital and adaptive filters, and tools such as wavelet transform. The phase related to beats segmentation is often included, even though some recent papers do not apply heartbeat segmentation when the databases used have labeled segments that include the heartbeat.

The most important phase of the classification process is the feature extraction. The typical parameter to be studied in this stage is the RR interval (which is the time between two R

peaks in consecutive beats). However, other waves of the heartbeat can be used as well, such as the QRS complex, considering that several types of arrhythmia are based on variations in the QRS complex. Combinations of time and frequency domain features can be studied as well.

Even though the wavelet transform is the most used technique to extract ECG signal features due to its effectiveness, other methods such as principal component analysis (PCA) can be used to extract completely new parameters that represent the beat. This last method also is used to reduce dimensionality of the feature vector.

The fourth phase is the classification itself. This stage is mainly based on machine learning technologies, using data from the previous phases of the pipeline, such as artificial neural networks (ANN).

An automatic classification method was proposed by Chazal et al. [1]. This method uses recordings of the MIT-BIH Arrhythmia Database and is based on morphology and beat interval-related features. The recommendations of the Association for the Advancement of Medical Instrumentation (AAMI) were taken into account, dividing the heartbeats in the following 5 super classes: normal (N); ventricular (V); supraventricular (S); fusion of normal and ventricular (F); and unknown beats (Q).

In this study, the ECG signal pre-processing was based on removing the baseline by means of median filters (one of them removing QRS intervals and P waves; the second one removing T waves). Then, the signal was filtered with a low-pass filter. In this case, the study was not focused on beats segmentation, considering that MIT-BIH database includes annotations with labeled segments within the heartbeat.

Regarding the feature extraction, it was focused in studying the heartbeat interval features and segmented ECG morphology features for each ECG lead. It was also performed an interpolation to reduce the number of features (ending up with 18 samples while the original beat contained 250 samples). For the final stage, a linear discriminant classifier was used, due to its simplicity and non-iterative nature (which means lower training time).

An alternative study was presented in 2016 by Al Rahhal et al. [2]. The approach was based on deep learning for active ECG signals classification. The influence of the expert labeling heartbeats for classifier adjustment was reduced by using active learning techniques. A denoising autoencoder (symmetrical neural network) was used to get a proper feature representation (using unsupervised learning manner). A pre-training of the autoencoder was performed on the training dataset, divided following the Chazal et al. [1] scheme. These results were used to initialize the deep neural network. Actually, active learning was proposed so as it was possible to improve the model by selecting the most useful samples

in order to reduce the number of training samples. Regarding the difficult samples, an expert labeled them and then the system was retrained.

The pre-processing performed in this study was similar to the one in Chazal et al. [1]. Median filters were applied (one to remove QRS intervals and P waves and the other one to remove the T wave). Then, these resulting signals were subtracted from the original ones, so as to correct the baseline. Furthermore, a low-pass filter (order 12) was used to eliminate noise. During the feature extraction phase, the complete ECG waveform only of the first lead was used in order to obtain proper feature representation of the input data with an autoencoder. For validation, the experimental results were obtained using three databases: (a) MIT-BIH Arrhythmia Database; (b) St.-Petersburg Institute of Cardiological Technics 12-Lead Arrhythmia Database (INCART); and (c) MIT-BIH Supraventricular Arrhythmia Database.

It is worth mentioning, though, that these two presented studies have some limitations. In Chazal et al. [1] paper, records 201 and 202 belong to the same male subject, but they were referred to different subsets. Furthermore, the record 232 in the second dataset is unbalanced since it contains more than 75% of heartbeats of supraventricular (S) type. On the other hand, although the results presented in Al Rahhal et al. [2] study were more robust (since the validation was performed on three different arrhythmia databases), the limitation here was the need of expert participation for the proper classification. Conventional machine learning algorithms have some resource limitations, especially in the feature extraction stage, since the features need to be defined beforehand. Deep learning algorithms could overcome this kind of limitations, as it can extract the information without the need of predefining these parameters.

In a recent study presented by Awni Y. Hannun et al. [3] it is used a deep neural network to detect and classify arrhythmia in ambulatory electrocardiograms. It was constructed a large ECG dataset that underwent expert annotation for a broad range of ECG rhythm classes. The deep neural network was developed to detect 12 rhythm classes from raw single-lead ECG inputs using a training dataset consisting of 91,232 ECG records from 53,549 patients. The DNN was designed to classify 10 arrhythmias as well as sinus rhythm and noise for a total of 12 output rhythm classes. When validated against an independent test dataset annotated by a consensus committee of board-certified practicing cardiologists, the DNN achieved an average area under the receiver operating characteristic curve of 0.97. Besides, the harmonic mean of the positive predictive value and sensitivity for the DNN (0.837) exceeded that of average cardiologists (0.780).

These findings demonstrate that a deep learning approach can classify a broad range of distinct arrhythmias from single-lead ECGs with high diagnostic performance similar to that of cardiologists.

1.4. Deep learning

Machine learning techniques work well for feature extraction and classification of ECG signals. Simple machine learning algorithms, such as principal component analysis, manage to solve a wide variety of problems. However, they have not been successful in solving main problems in AI, like speech or object recognition, in terms of computational costs.

Deep learning allows to design more complex models and to implement these methods in a simple way, using high-level open-source libraries, while reducing the computation time using GPU.

1.4.1. Autoencoder

The autoencoder is a deep learning model that aims to learn a representation of a data set in a different space so that the error of reconstructing the data from this representation space back to the original is as small as possible.

The structure consists of two symmetrical encoding and decoding parts (Figure 1.1), each possibly having multiple hidden layers. The middle layer is called the representation layer.

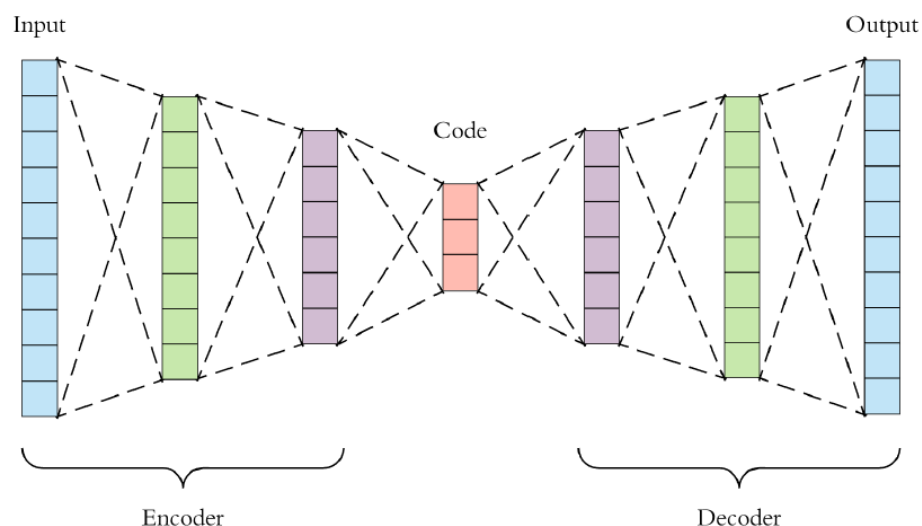


Figure 1.1 – Autoencoder scheme [Source: <https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368>]

Autoencoders are used mainly for: (a) compressing high-dimensional data into considerably smaller spaces; and (b) finding alternative representations of data and new features.

Instead of accuracy, autoencoders are measured using loss function, which is basically the amount of information lost during the reconstruction. As it is a symmetrical neural network, both output and input layers should have the same size.

1.4.2. Convolutional neural networks (CNN)

The fundamental difference between traditional (fully connected) and convolutional neural networks is the pattern of connections between consecutive layers. In the fully connected case, each unit is connected to all of the units in the previous layer. In a convolutional layer, each unit is connected to a number of nearby units in the previous layer (Figure 1.2). Furthermore, all units are connected to the previous layer in the same way, with the exact same weights and structure. This leads to an operation known as convolution. The output of a convolutional layer is often called feature map.

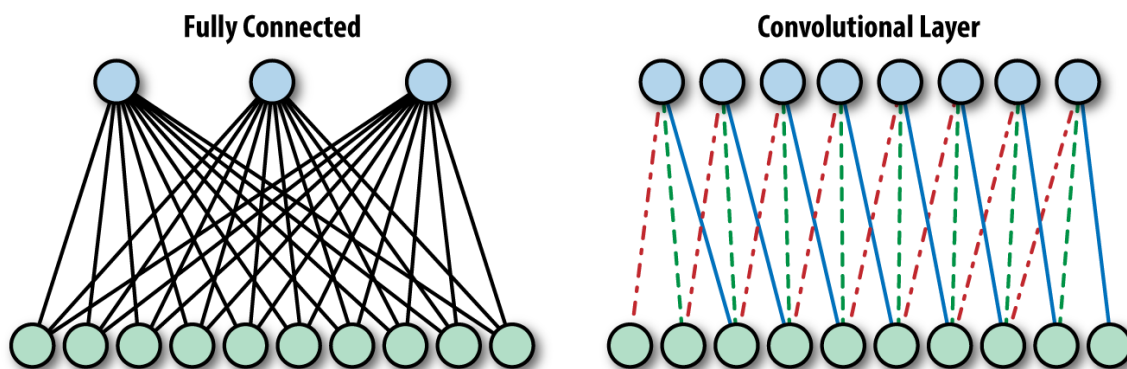


Figure 1.2 – Fully connected and convolutional layers comparison [4]

So, convolutional layers are like fully connected layers, but instead of searching for weights in the full space of matrices, the search is limited to matrices describing fixed-sizes convolutions, reducing the number of degrees of freedom to the size of the convolution, which is typically very small.

It is common to follow convolutional layers with pooling of outputs. Technically, “pooling”

means “reducing the size of the data with some local aggregation function”. The reason to perform the pooling is that it reduces the size of the data to be processed downstream. This can drastically reduce the number of overall parameters in the model, especially if there are fully connected layers after the convolutional ones. This operation can consist on replacing the output of a layer in a certain region with its maximum, average, or other statistics. The typical forms of pooling are (a) max pooling, which consists on computing the maximum of outputs in a region; and (b) average pooling, computing the average.

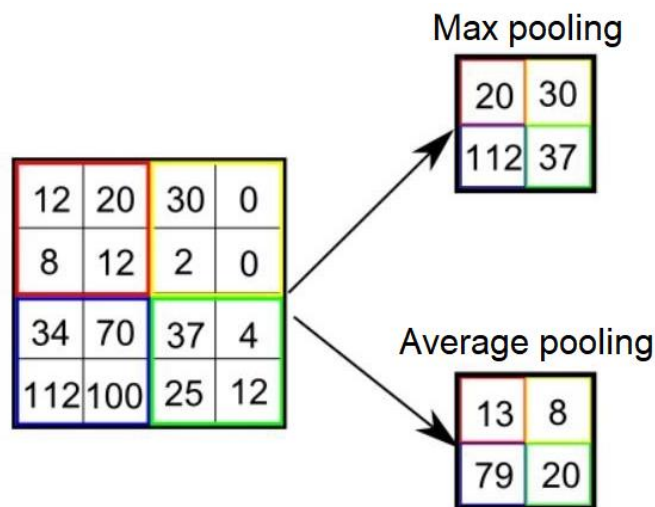


Figure 1.3 – Pooling process for a 2-D image [Source: <https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks>]

1.4.3. Recurrent neural networks (RNN)

Recurrent neural networks are a powerful and widely used class of neural network architectures for modeling sequence data. The basic idea behind RNN models is that each new element in the sequence contributes some new information, which updates the current state of the model. This kind of neural network is able to process input of not fixed size.

RNN models are based on a notion of a chain structure, and vary in how exactly they maintain and update information. When a new input is processed, it will be done in some manner that is dependent on the history of the sequence. This means that each element of the output is a function of the previous ones. RNN limits itself to learn only short-term dependencies due to vanishing gradients influence (as more layers are added to neural networks, the gradients of the loss function approach zero, making the network hard to

train).

A very popular recurrent network is the long short-term memory (LSTM) network. It differs from traditional RNN mainly by having some special memory mechanisms that enable the recurrent cells to better store information for long periods of time, thus allowing to capture long-term dependencies better than plain RNN. These memory mechanisms simply consist of some more parameters added to each recurrent cell, enabling the RNN to overcome optimization issues and propagate information. These trainable parameters act as filters that select what information is worth “remembering” and passing on, and what is worth “forgetting”. They are trained in exactly the same way as any other parameter in a network. So, the simplest LSTM structure contains only 3 gates: (a) input gate; (b) remember/forget gate; and (c) output gate.

2. ECG signals visualizer for MIT-BIH Database

The first objective is to develop a tool that allows the user to display the ECG signals of the MIT-BIH database by setting several parameters. Before starting the development of the application, it is needed to understand how the MIT-BIH Arrhythmia database is presented.

2.1. MIT-BIH Arrhythmia Database

MIT-BIH Arrhythmia Database is available in PhysioBank [5], a web platform that contains over 90,000 recordings, organized in more than 80 databases.

The ECG Arrhythmia Database contains 48 half-hour excerpts of two-channel ECG recordings, obtained from 47 subjects studied by the BIH Arrhythmia Laboratory. Twenty-three recordings were chosen randomly from a mixed population of inpatients and outpatients, while the remaining twenty-five were selected to include less common but clinically significant arrhythmias that would not be well-represented in a small random sample.

The recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10 mV range. Two or more cardiologists independently annotated each record; disagreements were resolved to obtain the computer-readable reference annotations for each beat included with the database.

The information of every record is contained in 3 types of files:

Concept	File extension
Signal	.dat
Reference annotation	.atr
Header	.hea

Table 2.1 – Types of files included in the MIT-BIH Arrhythmia database

- Signal

This file contains the value of the samples in mV with 11-bit resolution. The sampling frequency is 360 Hz.

- Reference annotation

This file contains the annotations that classify every beat identified in each record. The annotations that can be found in this database are shown in Table 2.2.

Symbol	Beat
N	Normal beat
L	Left bundle branch block beat
R	Right bundle branch block beat
A	Atrial premature beat
J	Nodal premature beat
S	Supra-ventricular premature beat
V	Premature ventricular contraction
F	Fusion of ventricular and normal beat
E	Ventricular escape beat
Q	Unclassifiable beat

Table 2.2 – Classes of heartbeats included in the MIT-BIH Arrhythmia database

- Header

This file contains information about how the corresponding “Signal” file is structured.

Once the MIT-BIH Arrhythmia Database is presented, the next step is to develop an application that allows the user to visualize the recordings in a clear interface. The application will be developed in Python language. As there are several Python libraries that can be used to display data, an application will be implemented in different ways so as to decide which one is the most comfortable interface for the user.

2.2. Data display using WFDB and TkInter packages

As a first approach, a simple display program is implemented using WFDB [6], the native Python waveform-database package. The advantage is that this package consists in a library of tools for reading, writing, and processing WFDB signals and annotations. This means that is simple way to manage and display the ECG data.

This library does not include methods to implement a graphical user interface, and the application should allow the user to configure graph parameters such as record number, range of samples, or the possibility to choose between first or second channels. This is why it is needed another package to implement a user interface.

For this first version, it has been chosen the TkInter [7], which is a Python standard GUI library.

In the following Figure 2.1, it is presented the user interface with which the user selects the parameters to display the desired data of the database.

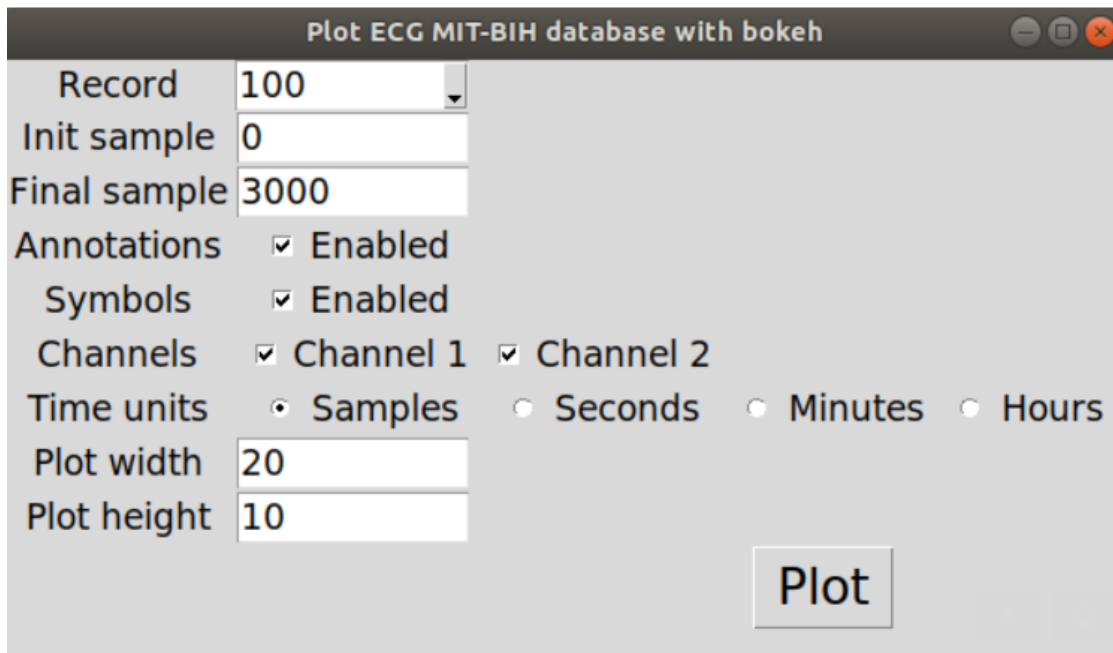


Figure 2.1 – User interface designed with TkInter package

The user can select between all the records available in the MIT-BIH database; it is also possible to enable or disable the annotations and symbols; select one or both channels; change the time units; and set the plot dimensions.

As soon as it is clicked the button “Plot”, after selecting the parameters, the graph is displayed using the WFDB package, as shown in the following Figure 2.2:

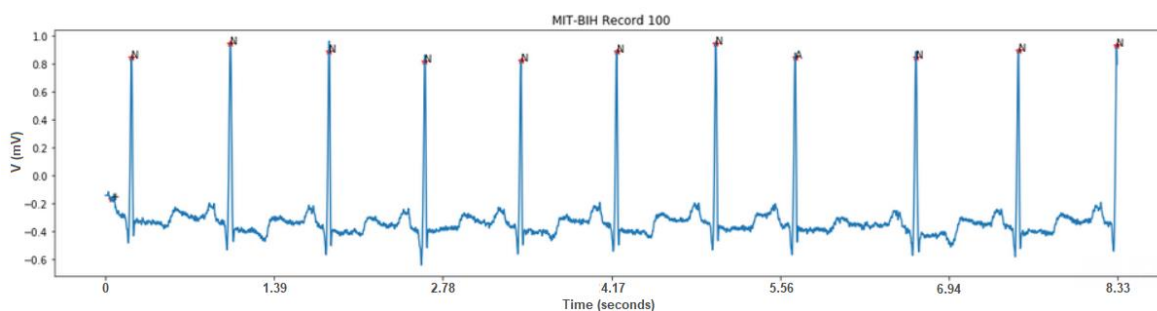


Figure 2.2 – ECG graph plotted using WFDB package

2.3. Data display using Bokeh and TkInter packages

The second implementation is based in Bokeh [8], which is a visualization library that targets web browsers for presentation. The advantage of this package is that the graph has much higher quality and precision, and several tools can be implemented to configure the plot parameters.

For this approach, the graphical user interface has been also implemented with the TkInter library. The main difference between this second approach and the one with WFDB is that the visualization is done using Bokeh, in the web browser, which allows using many tools such as detailed zoom and scroll, as shown in the following Figure 2.3:

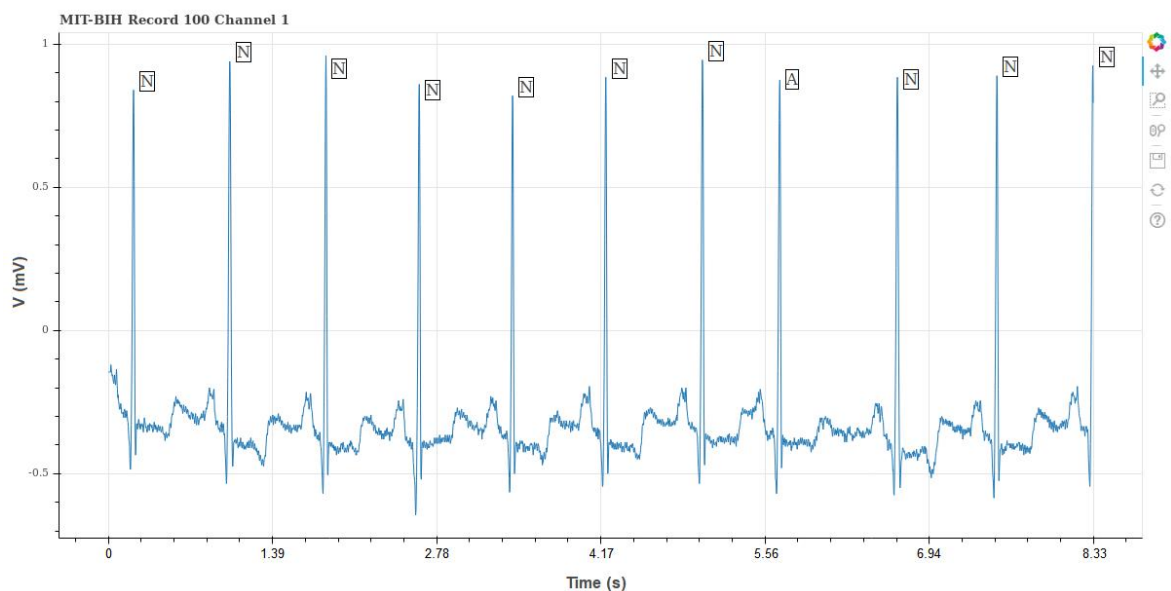


Figure 2.3 – ECG graph plotted using Bokeh library

The visualization using the Bokeh library is similar to the MATLAB plot interface.

2.4. Data display using Dash

Dash [9] is a Python open source library for building web applications. It is a framework of the company Plotly to build web-based analytics applications. The applications are rendered

in the web browser. The advantage over Bokeh is that Dash allows implementing a graphical user interface in a simple way and does not need a lot of code to do so.

As Dash allows the user to change parameters and the graph is then updated in real time, this is the best solution proposed for the ECG signals display application.

The following Figure 2.4 shows the layout of the application built with Dash. In this example, the displayed signal corresponds to the record 100, from the second 14 to the 22.

Graph ECG MIT-BIH database with Dash

Plotting a record of MIT-BIH database

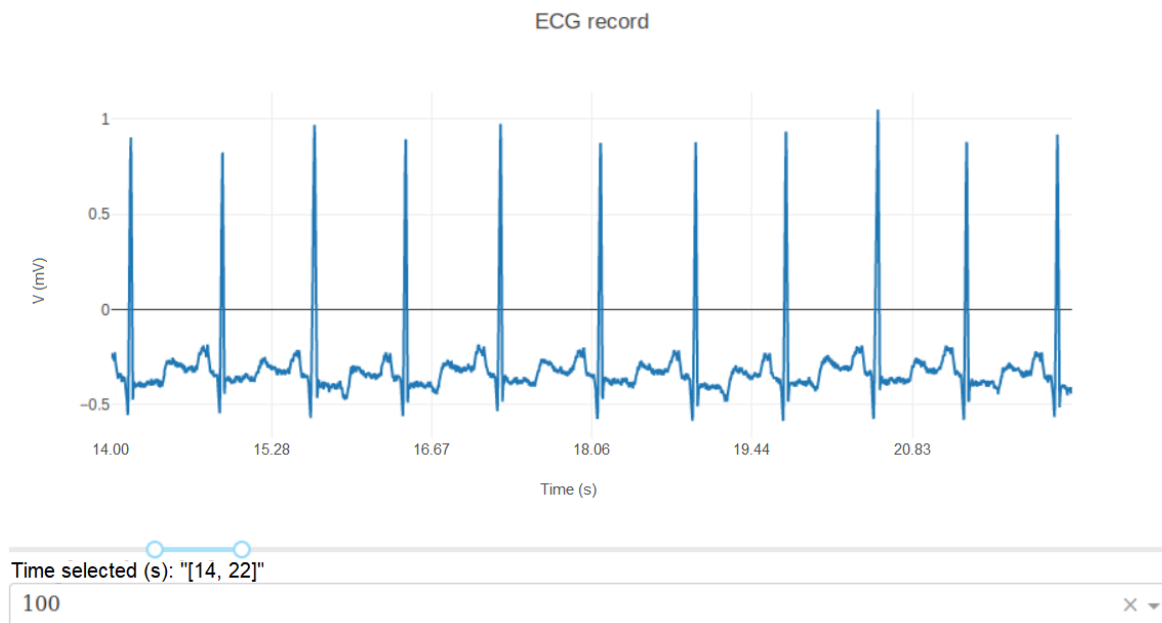


Figure 2.4 – ECG graph and user interface using Dash library

In case the user changes the record or modifies the values of the slider, the graph is updated instantly. This feature is not possible in the rest of considered libraries. Therefore, it can be concluded that this package is the best option to build the data visualizer.

3. Principal Component Analysis (PCA)

To characterize the beats obtained from the MIT-BIH Arrhythmia Database, a first approach is to perform a Principal Component Analysis.

Principal Component Analysis is a dimensionality-reduction method often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. The idea in dimensionality reduction is to trade a little accuracy for simplicity, since smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

To compute the principal component analysis it is needed, basically, the following steps: (a) standardization; (b) covariance matrix computation; (c) computation of the eigenvectors and eigenvalues of the covariance matrix to identify the principal components; and (d) calculate the number of principal components.

(a) Standardization

The aim of this step is to standardize the range of the initial variables so that each one of them contributes equally to the analysis. Specifically, the reason of performing this step prior to PCA is that the Principal Component Analysis is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges, which will lead to biased results. Mathematically, this is done by subtracting the mean and dividing by the standard deviation for each value of each variable.

$$z = \frac{\text{value} - \text{mean}}{\text{standard deviation}}$$

(b) Covariance matrix computation

The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other (to check if there is any relationship between them). The covariance matrix is a $p \times p$ symmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables.

The covariance matrix for a 3-dimensional data set with 3 variables x, y, and z, for instance, would be:

$$\begin{bmatrix} \text{Cov}(x, x) & \text{Cov}(x, y) & \text{Cov}(x, z) \\ \text{Cov}(y, x) & \text{Cov}(y, y) & \text{Cov}(y, z) \\ \text{Cov}(z, x) & \text{Cov}(z, y) & \text{Cov}(z, z) \end{bmatrix}$$

(c) Computation of the eigenvectors and eigenvalues of the covariance matrix to identify the principal components

Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (principal components) are uncorrelated and most of the information within the initial variables is compressed into the first components.

In geometrical terms, principal components represent the directions of the data that explain a maximal amount of variance (the lines that capture most information of the data). The relationship between variance and information is that, the larger the variance carried by a line, the larger the dispersion of the data points along it; and the larger dispersion along a line, the more information it has. Principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set. The second principal component is uncorrelated with (so, perpendicular to) the first principal component and it accounts for the next highest variance. This continues until a total of p principal components have been calculated, equal to the original number of variables.

The eigenvectors of the covariance matrix are actually the directions of the axes where there is the most variance (and therefore, the most information). These directions are called Principal Components. And eigenvalues are simply the coefficients attached to eigenvectors, which give the amount of variance carried in each principal component. This means that to get the principal components in order of significance, it is needed to rank the eigenvectors in order of their eigenvalues, highest to lowest.

(d) Number of Principal Components

In this step, it is needed to choose whether to keep all these components or discard those of less significance (lower eigenvalues), and form with the remaining ones a matrix of vectors that it is called Feature vector. So, the feature vector is simply a matrix that has as columns

the eigenvectors of the components to be kept. This makes it the first step towards dimensionality reduction.

In the presented case, the proposal is to perform a PCA for every type of heartbeat. For that purpose, it is built a matrix of beats for every one of the types available in the MIT-BIH database. This matrix has the segmented beats in every row. The segmentation consists of centering the R peak in a vector of samples. The size of this vector needs to include the whole peak, therefore, a number of 240 samples per beat is used. This means the matrix of beats has 240 columns and as many rows as beats are chosen for the PCA. The following sections show the result of the PCA for certain type of beats.

In every Principal Component Analysis, it has been considered the necessary number of components in order to explain, at least, 95% of the variance. So, the first step for every type of heartbeat is to calculate the necessary number of components to reach a cumulative explained variance of 95%.

Then, as the number of principal components is determined, a vector for each type of heartbeat can be defined as:

$$v = \text{mean vector} + \sum a_i \cdot PC_i$$

Being v , *mean vector* and PC_i , vectors of 240 components, as it was defined to segment the beats in groups of 240 samples.

To study each type of beat, it has been developed a simple application using Dash library. This application allows the user to change the value of every coefficient a_i . This method can help to understand the importance of every component in the physiological shape of each kind of heartbeat.

The graphical user interface consists of a graph, which plots both the mean vector and the result vector (v). The result vector is the lineal combination of the PCs in addition to the mean vector. There are also slide bars (as many as principal components have been selected in every type of beat) that can be configured by the user to give more or less importance to the principal components. The value of the slide bar varies between -1 and 1, but the result vector exaggerates this value multiplying it by 10, so as to distinguish better the difference between the mean and the result vector. An example layout of the beats of type N is shown in the following Figure 3.1.

Graph ECG with Dash: Beats N

Plotting mean vector and principal components

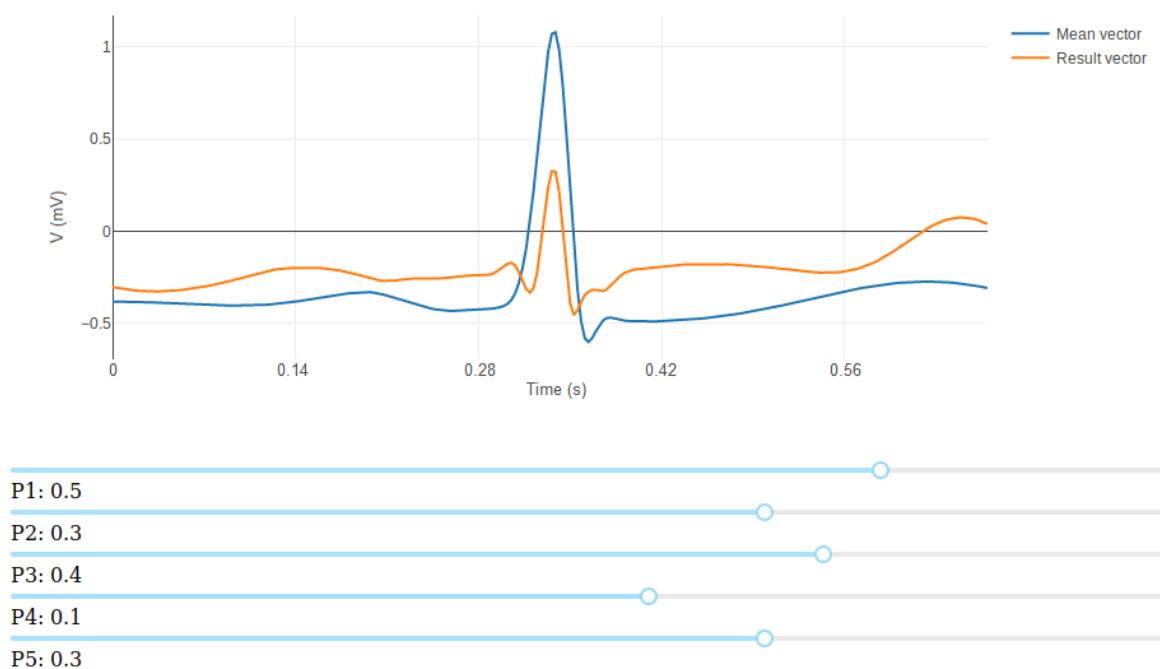


Figure 3.1 – Principal Components of the beats of type N using Dash library

As it is observed in the previous figure, the mean vector is in blue, and the result vector is in orange. The coefficients set by the user are, in this example:

a_1	a_2	a_3	a_4	a_5
0.5	0.3	0.4	0.1	0.3

In summary, the methodology in this section will be: first, to calculate the necessary number of components; and then to give values to the coefficients a_i in order to understand the importance of each component to the shape of the heartbeat. The values will be changed

one by one, setting the rest to zero, to understand the importance of each one independently.

As there are many types of beats, the study of the PCA will be performed for the 5 types with a higher number of beats in the database. So, the principal component analysis will be applied in beats N, L, R, V and A. The rest of beats have lower number of samples and thus the study would be less precise.

3.1. Type N: Normal beat

A normal beat consists of five waves (Figure 3.2) alphabetically arranged: the P wave; then the Q, R and S waves (which conform the QRS complex) and finally the T wave.

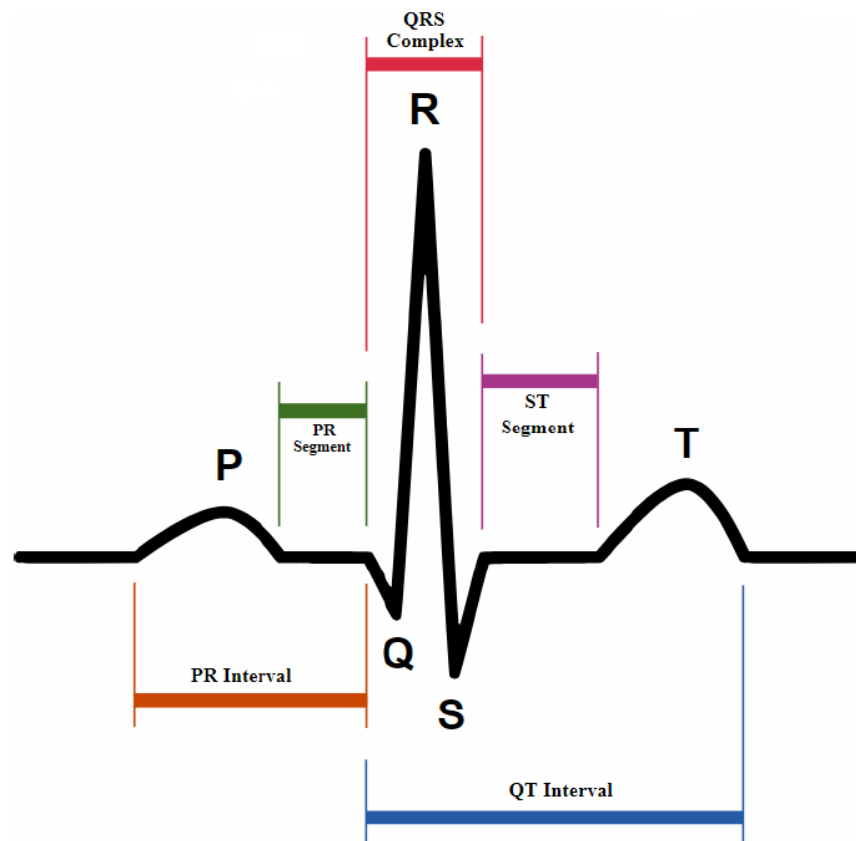


Figure 3.2 – General shape of a normal beat [Source: <https://www.firstaidforfree.com/a-basic-guide-to-ecg-kg-interpretation>]

First, to reach a cumulative explained variance of 95% there are needed 5 components, as it is shown in the following figure 3.3.

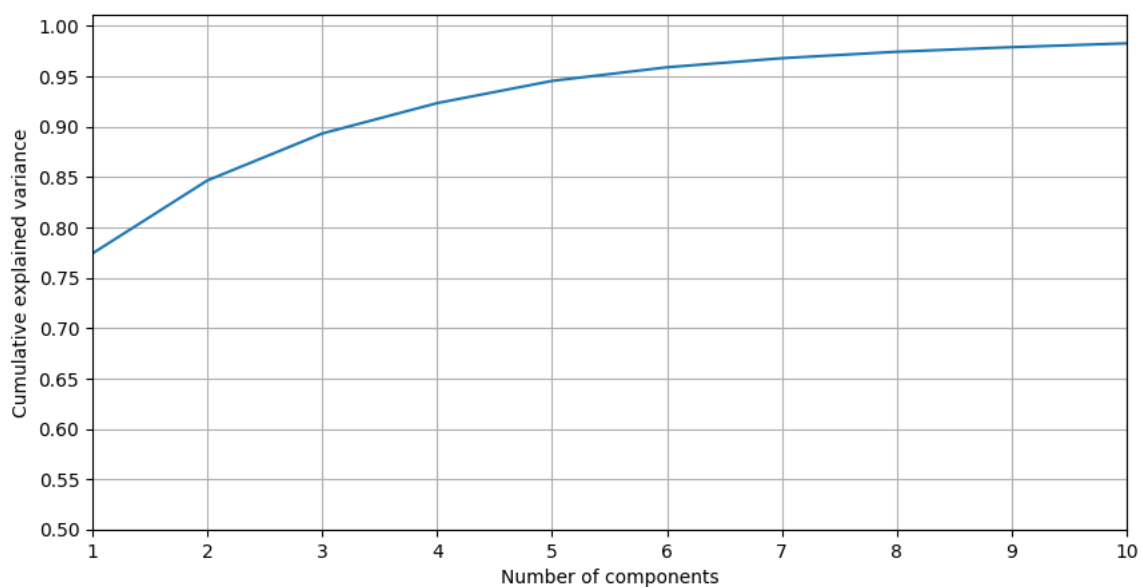


Figure 3.3 – Number of Principal Components for type N

Symbol	Name	Number of beats in the database	Number of components
N	Normal beat	74,546	5

Table 3.1 – Normal beats in the MIT-BIH Arrhythmia database

The explained variance of each Principal Component is shown in the following graph (Figure 3.4):

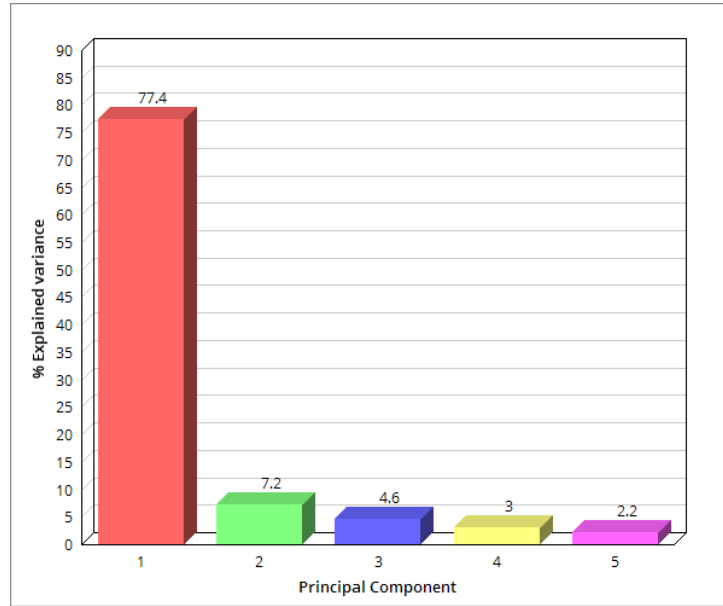


Figure 3.4 – Explained variance of each PC for beats N

Thus, the result vector for type N is:

$$\vec{v}_N = a_{N1} \cdot \vec{PC}_1 + a_{N2} \cdot \vec{PC}_2 + a_{N3} \cdot \vec{PC}_3 + a_{N4} \cdot \vec{PC}_4 + a_{N5} \cdot \vec{PC}_5$$

The vector of coefficients of type N is defined as:

$$a_N = [a_{N1}, a_{N2}, a_{N3}, a_{N4}, a_{N5}]$$

For the representation of each Principal component, each coefficient a_{Ni} will get a value of 1 and the rest of them 0, so as it can be observed the individual influence of every PC in the resulting signal.

PC1

$$\mathbf{a}_N = [1, 0, 0, 0, 0]$$

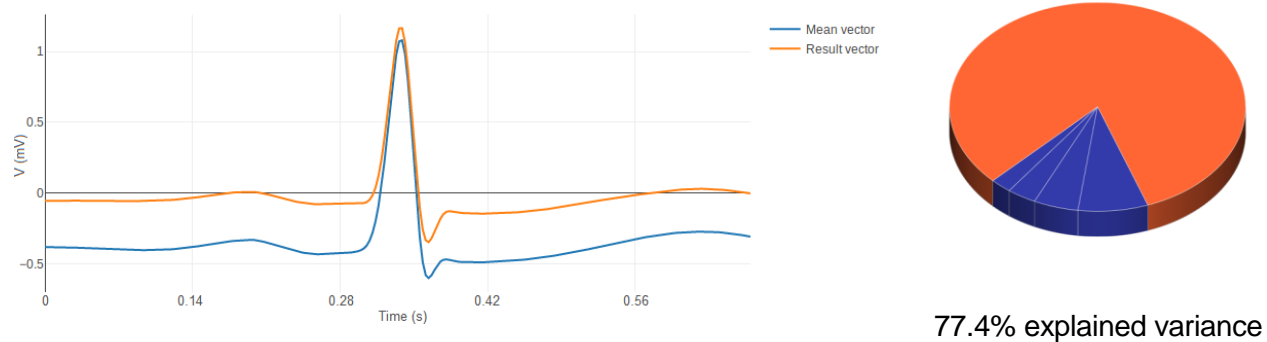


Figure 3.5 – Principal Component 1 for beats of type N

The first component increases the baseline and barely changes the QRS complex.

PC2

$$\mathbf{a}_N = [0, 1, 0, 0, 0]$$

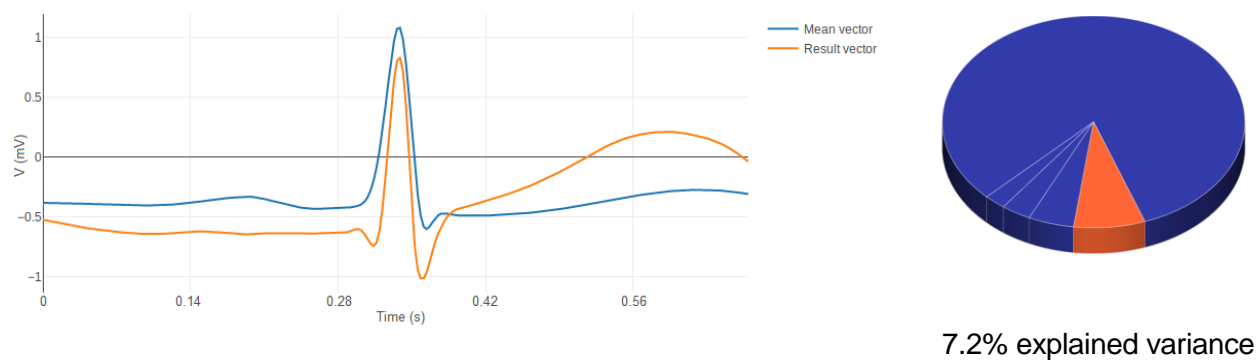
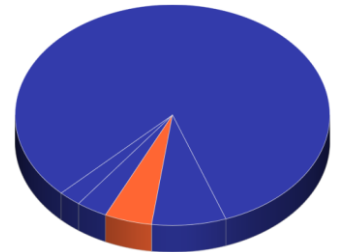
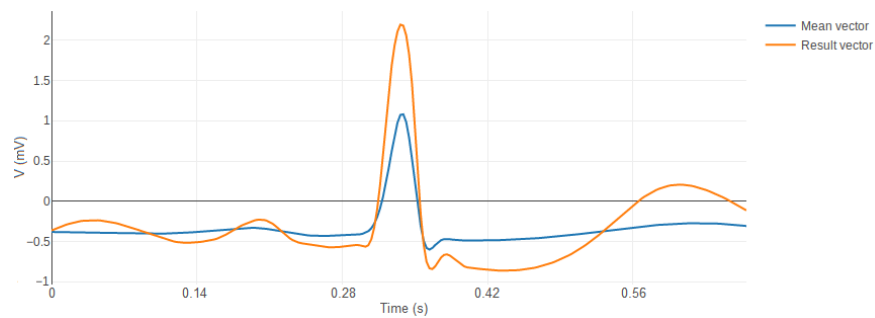


Figure 3.6 – Principal Component 2 for beats of type N

The PC2 emphasizes the Q and the S waves, as it can be observed in the previous image.

PC3

$$\mathbf{a}_N = [0, 0, 1, 0, 0]$$



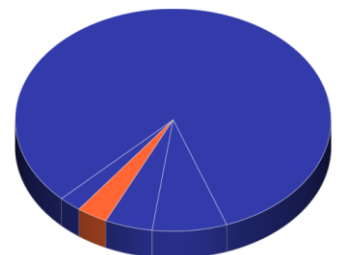
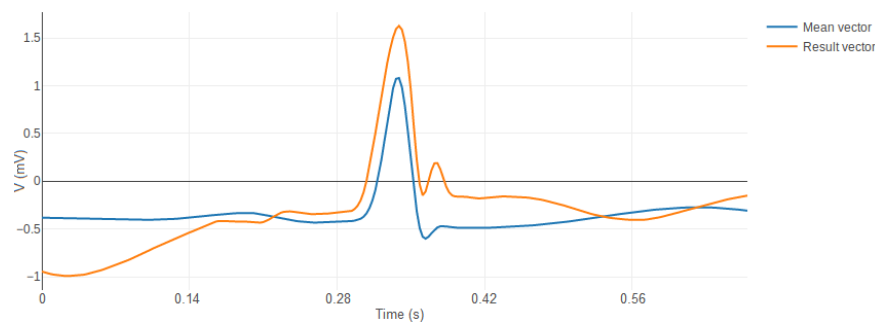
4.6% explained variance

Figure 3.7 – Principal Component 3 for beats of type N

The PC3 makes the P and T waves appear. These waves can be distinguished over the mean vector in the previous image.

PC4

$$\mathbf{a}_N = [0, 0, 0, 1, 0]$$



3.0% explained variance

Figure 3.8 – Principal Component 4 for beats of type N

The PC4 emphasizes the S wave and the peak after this wave might be the T wave showing up.

PC5

$$\mathbf{a}_N = [0, 0, 0, 0, 1]$$

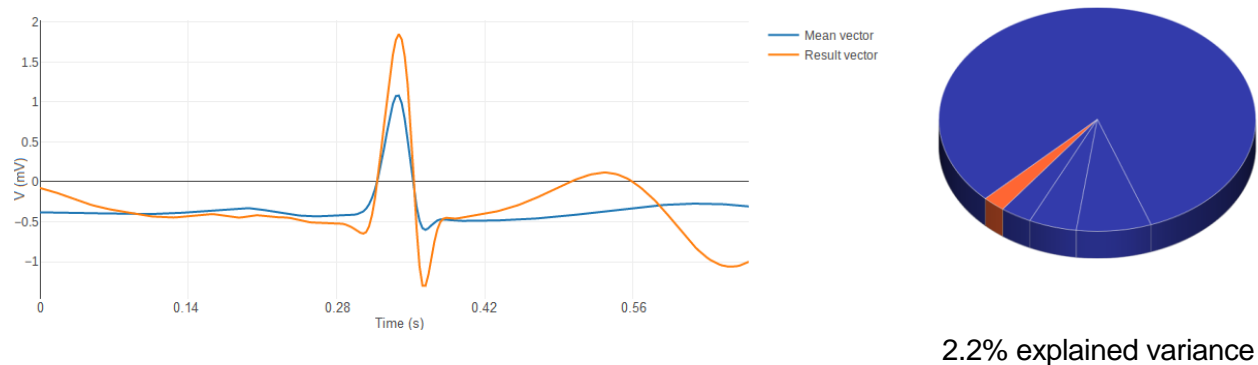


Figure 3.9 – Principal Component 5 for beats of type N

The PC5 increases the R peak and accentuates the S wave.

3.2. Type L: Left bundle branch block beat

A Left bundle branch block beat has the following general shape (Figure 3.10):



Figure 3.10 – General shape of a beat L [Source: <https://www.verywellhealth.com/bundle-branch-block-bbb-1745219>]

As it is observed in the previous figure, a delay shows up as a distinctive pattern on the ECG called a bundle branch block. The electrical impulse is delayed in reaching its respective ventricle, and thus the impulse of the heart is not being distributed normally through the cardiac ventricles. The QRS complex is no longer narrow: it becomes much wider than normal. This widening happens because it takes longer for the electrical signal to be completely distributed across both ventricles.

First, to reach a cumulative explained variance of 95% there are needed 5 components, as it is shown in the following Figure 3.11.

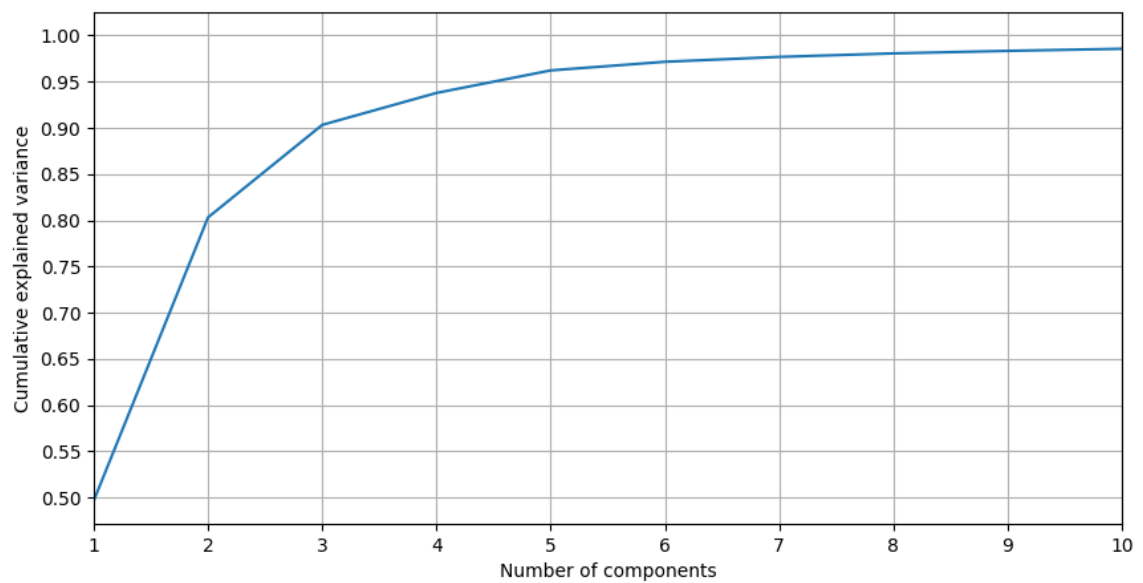


Figure 3.11 – Number of Principal Components needed for type L

Symbol	Name	Number of beats in the database	Number of components
L	Left bundle branch block beat	8,075	5

Table 3.2 – Beats of type L in the MIT-BIH Arrhythmia database

The explained variance of each Principal component is shown in the following graph (Figure 3.12):

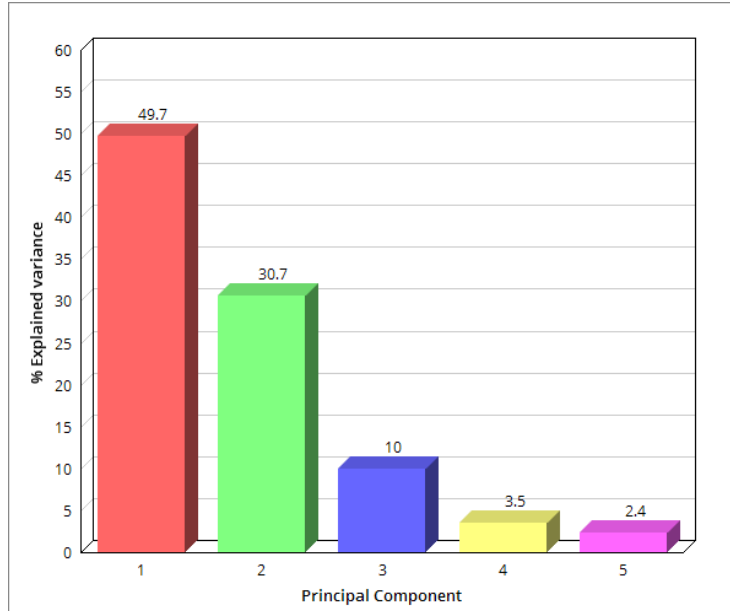


Figure 3.12 – Explained variance of each PC for beats L

Thus, the result vector for type L is:

$$\vec{v}_L = a_{L1} \cdot \vec{PC}_1 + a_{L2} \cdot \vec{PC}_2 + a_{L3} \cdot \vec{PC}_3 + a_{L4} \cdot \vec{PC}_4 + a_{L5} \cdot \vec{PC}_5$$

The vector of coefficients of type L is defined as:

$$a_L = [a_{L1}, a_{L2}, a_{L3}, a_{L4}, a_{L5}]$$

For the representation of each PC, each coefficient a_{Li} will get a value of 1 and the rest of them 0, so as it can be observed the individual influence of every PC in the resulting signal.

PC1

$$\mathbf{a}_L = [1, 0, 0, 0, 0]$$

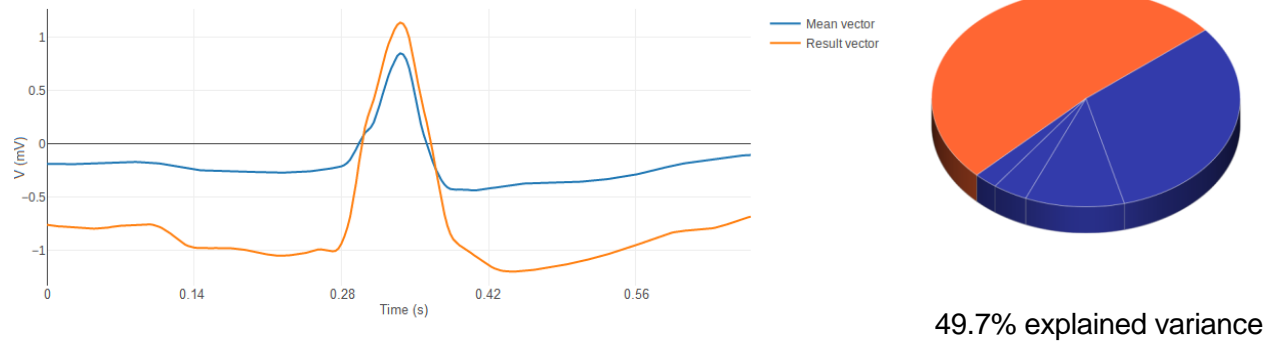


Figure 3.13 – Principal Component 1 for beats of type L

As in the normal beat, the PC1 increases the baseline and barely affects the R peak.

PC2

$$\mathbf{a}_L = [0, 1, 0, 0, 0]$$

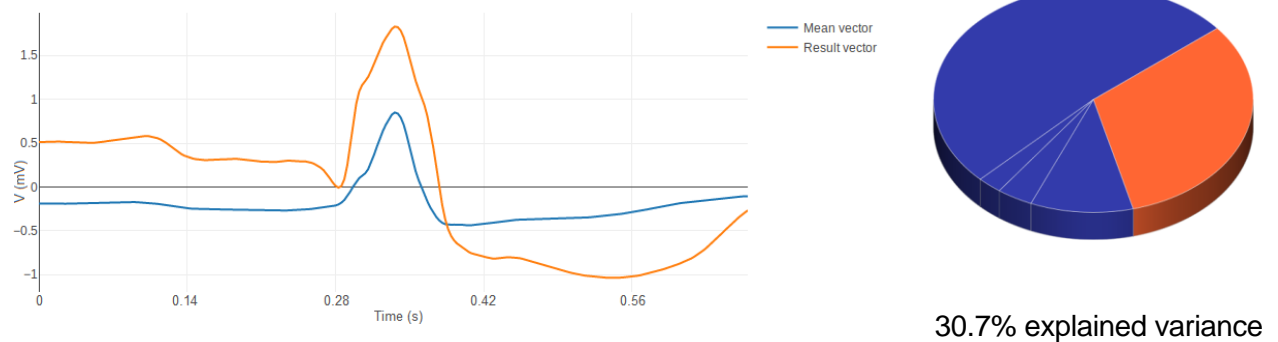
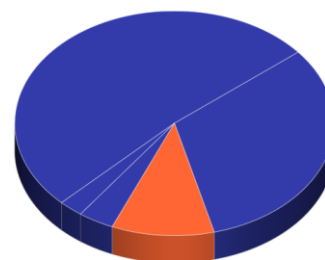
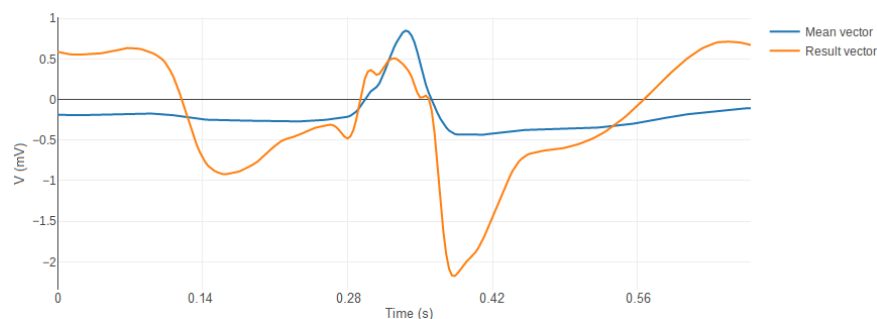


Figure 3.14 – Principal Component 3 for beats of type L

The second principal component increases the R peak, as it can be observed in the previous figure. There is also an increase of the baseline in the first half, before Q wave; and a decrease after the S wave.

PC3

$$\mathbf{a}_L = [0, 0, 1, 0, 0]$$



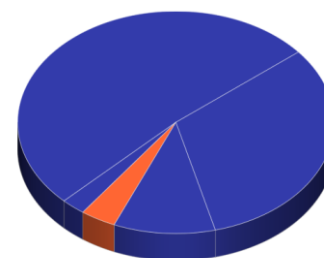
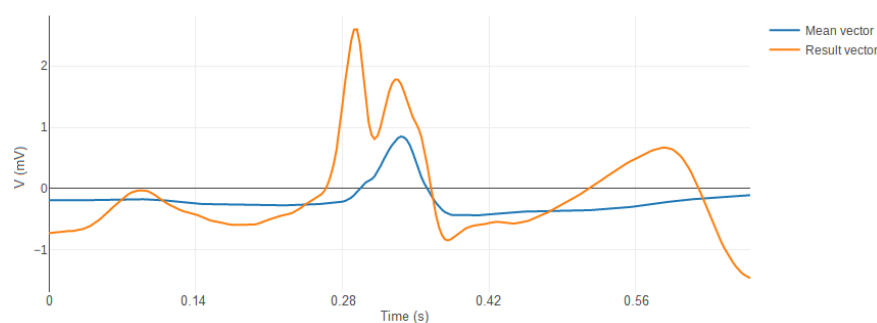
10.0% explained variance

Figure 3.15 – Principal Component 3 for beats of type L

The PC3 slightly presents the typical shape of the bundle branch block beat in the R peak.

PC4

$$\mathbf{a}_L = [0, 0, 0, 1, 0]$$



3.5% explained variance

Figure 3.16 – Principal Component 4 for beats of type L

PC4 shows the typical QRS shape of the bundle branch block beat in the R peak, where it can be observed the double peak.

PC5

$$\mathbf{a}_L = [0, 0, 0, 0, 1]$$

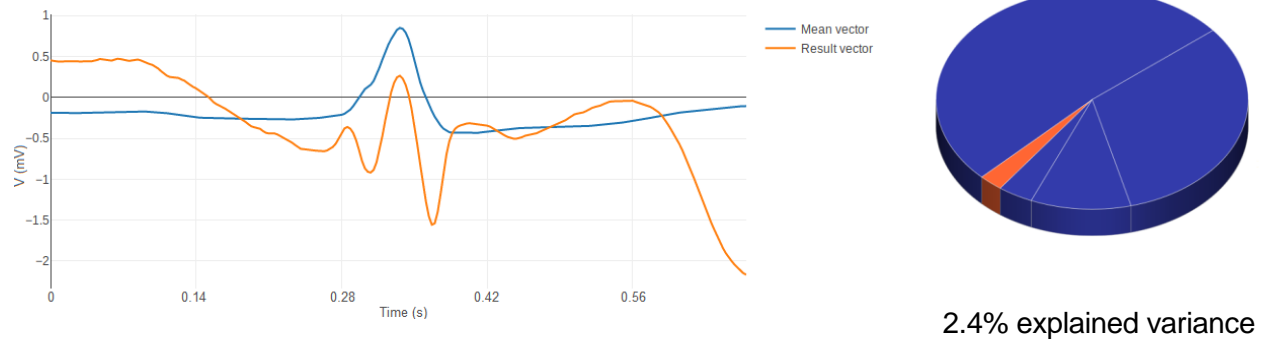


Figure 3.17 – Principal Component 5 for beats of type L

3.3. Type R: Right bundle branch block beat

A right bundle branch block beat has the following general shape (Figure 3.18):



Figure 3.18 – General shape of a beat R [Source: <https://www.verywellhealth.com/bundle-branch-block-bbb-1745219>]

As for the left bundle branch block beat, it is observed that a delay shows up as a distinctive pattern on the ECG (called bundle branch block). The electrical impulse is delayed in

reaching its respective ventricle, and thus the impulse of the heart is not being distributed normally through the cardiac ventricles. The QRS complex is no longer narrow: it becomes much wider than normal. This widening happens because it takes longer for the electrical signal to be completely distributed across both ventricles.

First, to reach a cumulative explained variance of 95% there are needed 3 components, as it is shown in the following Figure 3.19.

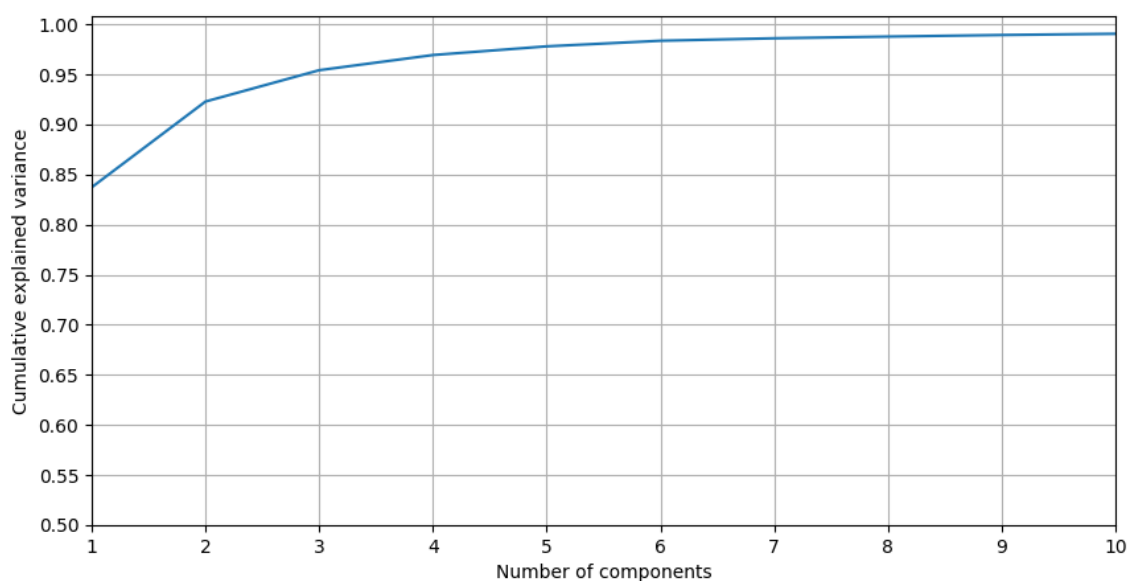


Figure 3.19 – Number of Principal Components needed for type R

Symbol	Name	Number of beats in the database	Number of components
R	Right bundle branch block beat	7,259	3

Table 3.3 – Beats of type R in the MIT-BIH Arrhythmia database

The explained variance of each Principal component is shown in the following graph (Figure 3.20):

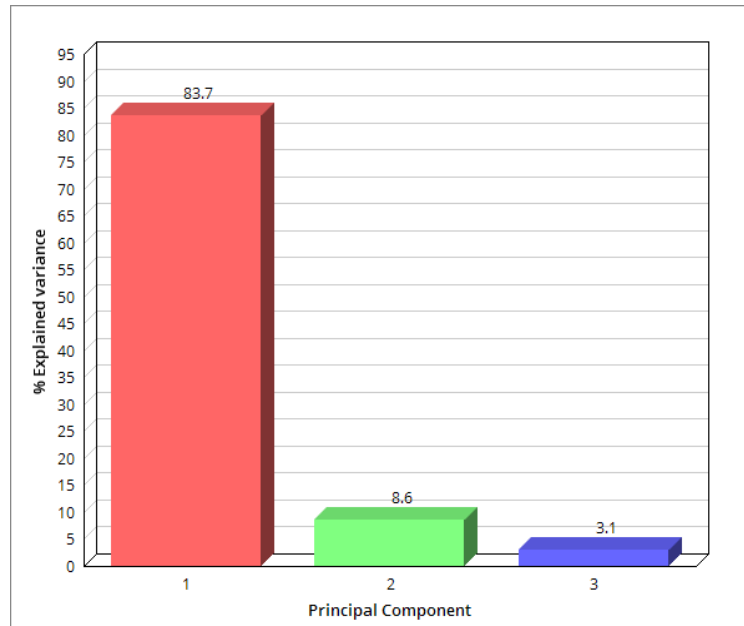


Figure 3.20 – Explained variance of each PC for beats R

Thus, the result vector for type R is:

$$\vec{v}_R = a_{R1} \cdot \vec{PC}_1 + a_{R2} \cdot \vec{PC}_2 + a_{R3} \cdot \vec{PC}_3$$

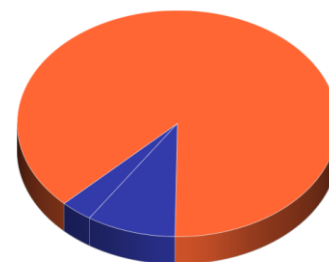
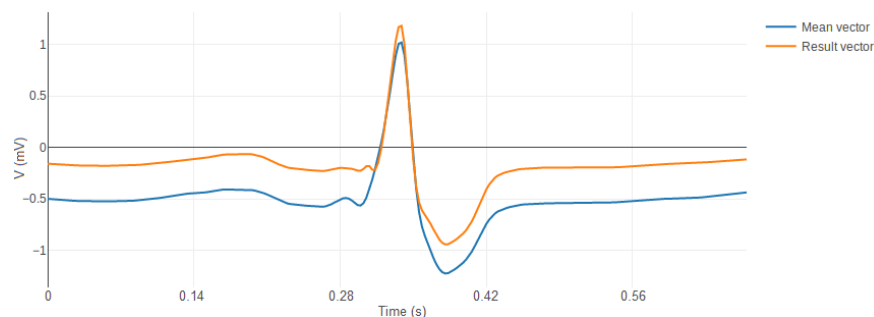
The vector of coefficients of type R is defined as:

$$a_R = [a_{R1}, a_{R2}, a_{R3}]$$

For the representation of each PC, each coefficient a_{Ri} will get a value of 1 and the rest of them 0, so as it can be observed the individual influence of every PC in the resulting signal.

PC1

$$\mathbf{a}_R = [1, 0, 0]$$



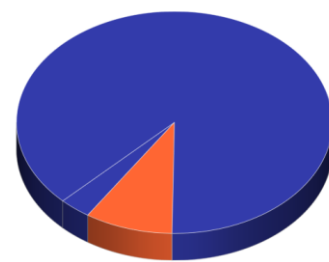
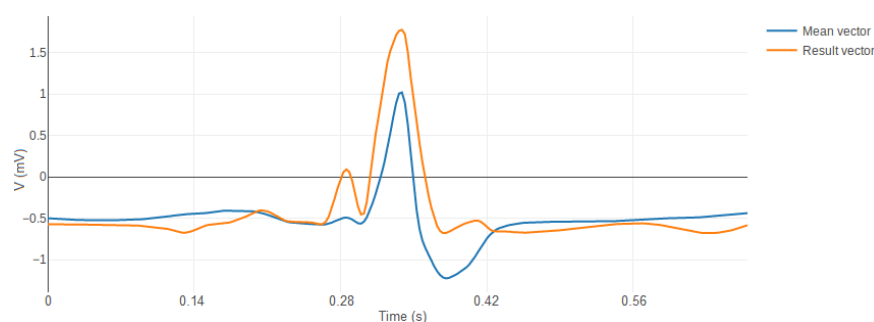
83.7% explained variance

Figure 3.21 – Principal Component 1 for beats of type R

The first principal component shows an increase in the baseline, while the R peak is kept constant.

PC2

$$\mathbf{a}_R = [0, 1, 0]$$



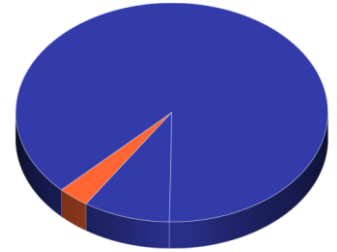
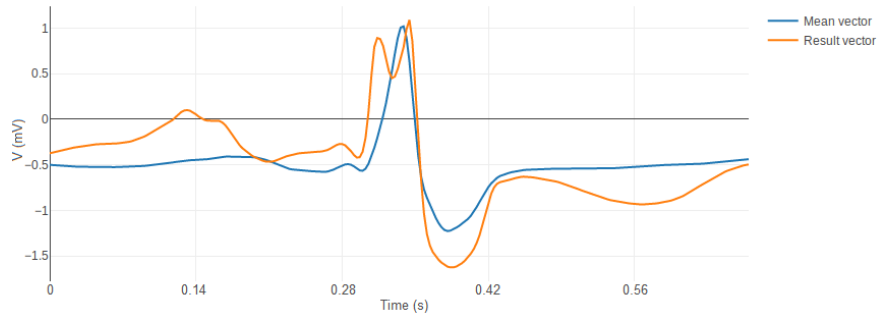
8.6% explained variance

Figure 3.22 – Principal Component 2 for beats of type R

The PC2 in this case accentuates the Q wave, and the peak before the R wave might be due to the influence of the P wave.

PC3

$$\mathbf{a}_R = [0, 0, 1]$$



3.1% explained variance

Figure 3.23 – Principal Component 3 for beats of type R

PC3 shows the typical QRS shape of the bundle branch block beat in the R peak zone, where it can be observed the double peak. The P wave is also visible (it can be distinguished over the mean vector around the second 0.14).

3.4. Type V: Premature ventricular contraction

A premature ventricular contraction has the following general shape (Figure 3.24):



Figure 3.24 – General shape of a premature ventricular contraction [Source: https://wikem.org/wiki/Premature_ventricular_contraction]

As it is observed in the previous image, the T wave is not distinguishable in this type of beat, since the electrical signal decreases significantly after the R peak.

First, to reach a cumulative explained variance of 95% there are needed 7 components, as it is shown in the following Figure 3.25.

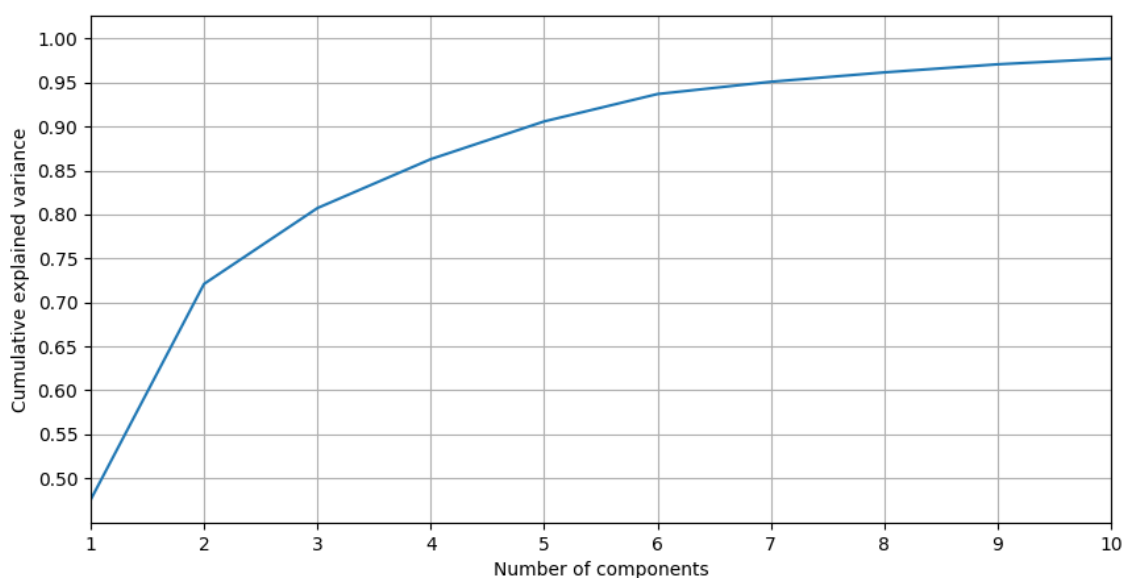


Figure 3.25 – Number of Principal Components needed for type V

Symbol	Name	Number of beats in the database	Number of components
V	Premature ventricular contraction	6,093	7

Table 3.4 – Beats of type V in the MIT-BIH Arrhythmia database

The explained variance of each Principal component is shown in the following graph (Figure 3.26):

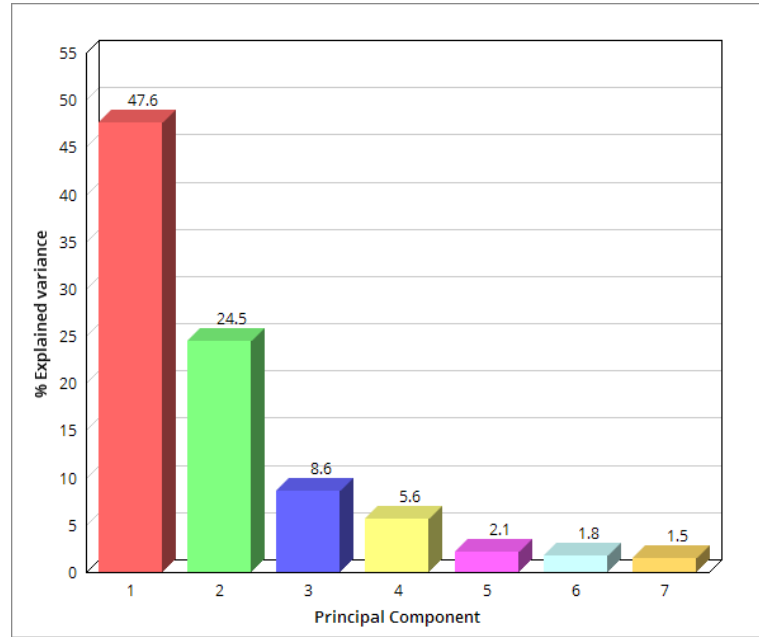


Figure 3.26 – Explained variance of each PC for beats V

Thus, the result vector for type V is:

$$\vec{v}_V = a_{V1} \cdot \vec{PC}_1 + a_{V2} \cdot \vec{PC}_2 + a_{V3} \cdot \vec{PC}_3 + a_{V4} \cdot \vec{PC}_4 + a_{V5} \cdot \vec{PC}_5 + a_{V6} \cdot \vec{PC}_6 + a_{V7} \cdot \vec{PC}_7$$

The vector of coefficients of type V is defined as:

$$a_V = [a_{V1}, a_{V2}, a_{V3}, a_{V4}, a_{V5}, a_{V6}, a_{V7}]$$

For the representation of each PC, each coefficient a_{Vi} will get a value of 1 and the rest of them 0, so as it can be observed the individual influence of every PC in the resulting signal.

PC1

$$\mathbf{a}_V = [1, 0, 0, 0, 0, 0, 0]$$

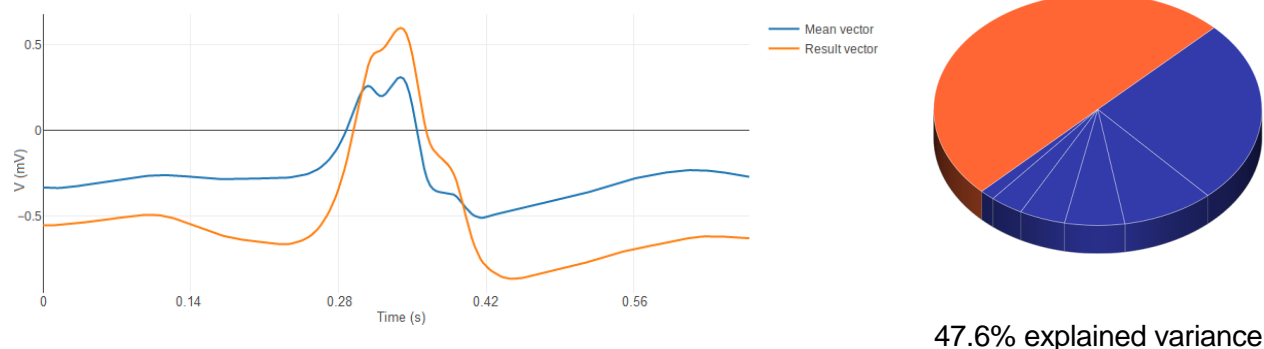


Figure 3.27 – Principal Component 1 for beats of type V

The first principal component exaggerates the R peak and shows a hard decrease that might hide the T wave. These two features are typical of the premature ventricular contraction.

PC2

$$\mathbf{a}_V = [0, 1, 0, 0, 0, 0, 0]$$

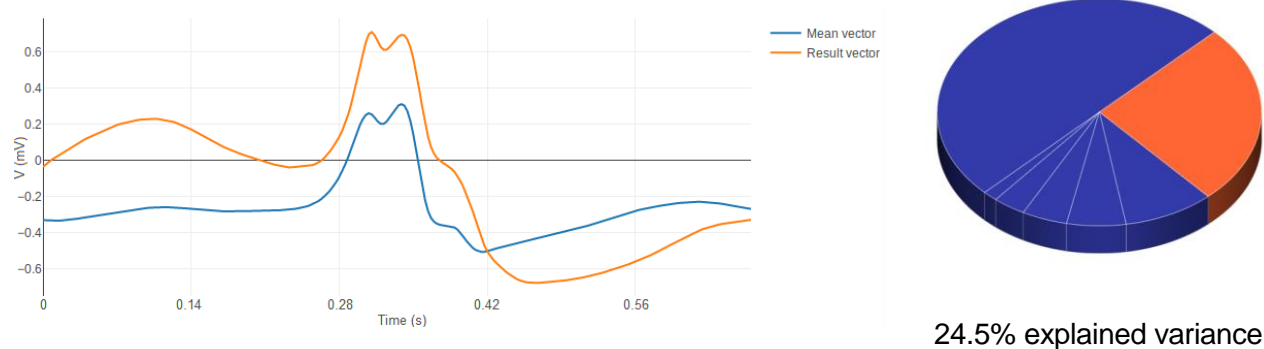
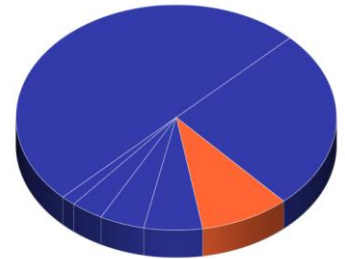
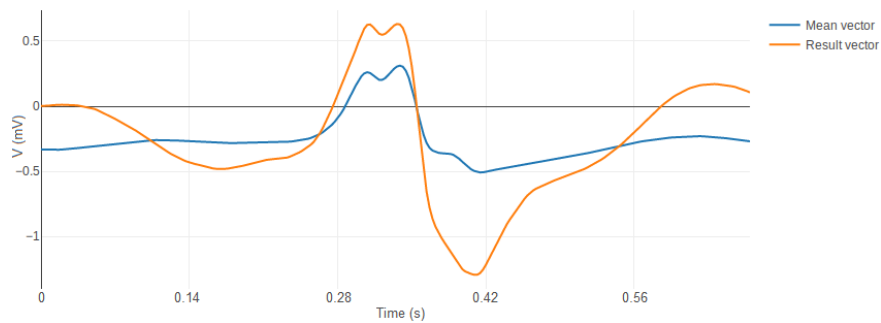


Figure 3.28 – Principal Component 2 for beats of type V

The PC2 shows the P wave around the second 0.10, and exaggerates the decrease after the R peak, which is typical in the premature ventricular contraction.

PC3

$$\mathbf{a}_V = [0, 0, 1, 0, 0, 0, 0]$$



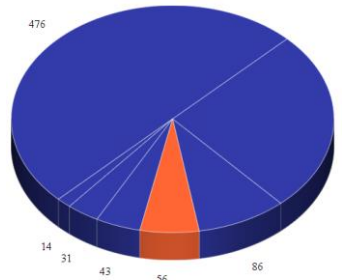
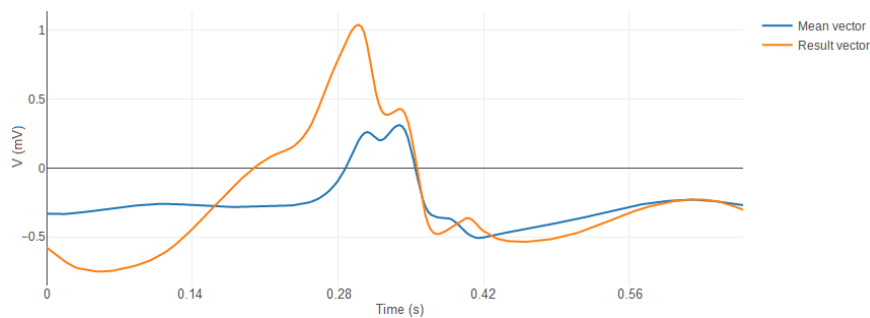
8.6% explained variance

Figure 3.29 – Principal Component 3 for beats of type V

PC3 shows the hard decrease after the R peak that ends up hiding the T wave, as indicated in the previous explanation.

PC4

$$\mathbf{a}_V = [0, 0, 0, 1, 0, 0, 0]$$



5.6% explained variance

Figure 3.30 – Principal Component 4 for beats of type V

PC5, PC6 and PC7 are not represented in this section since their explained variance is lower compared to the first ones. Thus, the importance of these three components is almost insignificant in the case of the premature ventricular contraction.

3.5. Type A: Atrial premature beat

An atrial premature beat has the following general shape (Figure 3.31):

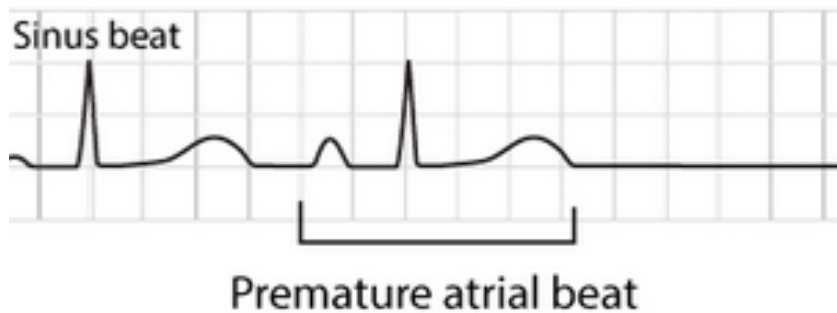


Figure 3.31 – General shape of a beat of type A [Source: <https://www.webmd.com/heart-disease/atrial-fibrillation/premature-atrial-contractions#1>]

As it is observed in the image, the individual shape of the beat is similar to the normal beat, but the main difference is that the beat occurs significantly earlier than the expected.

First, to reach a cumulative explained variance of 95% there are needed 6 components, as it is shown in the following Figure 3.32.

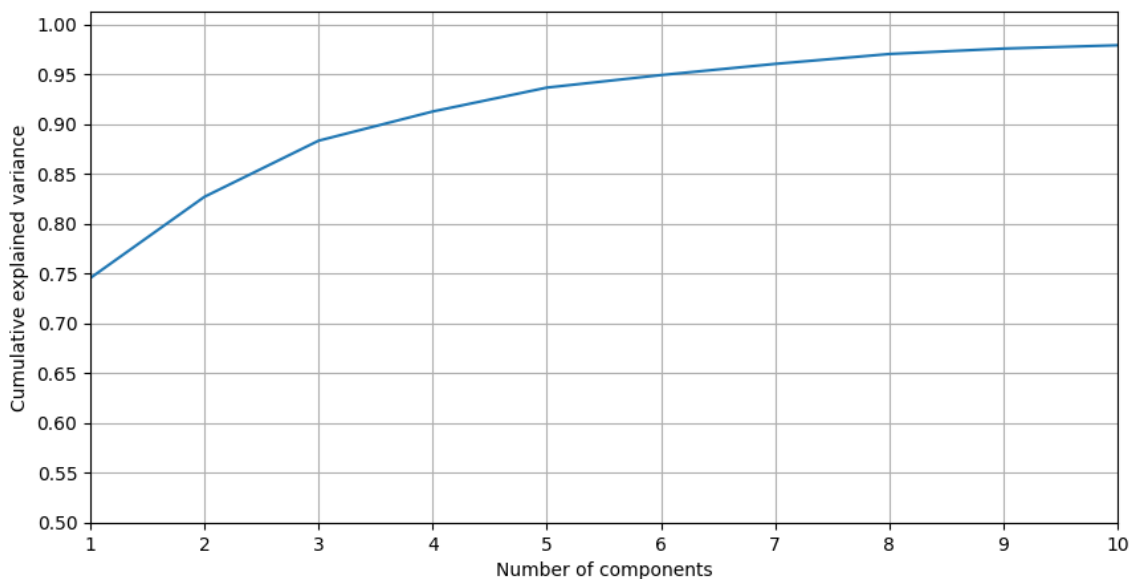


Figure 3.32 – Number of Principal Components needed for type A

Symbol	Name	Number of beats in the database	Number of components
A	Atrial premature beat	2,546	6

Table 3.5 – Beats of type A in the MIT-BIH Arrhythmia database

The explained variance of each Principal component is shown in the following graph (Figure 3.33):

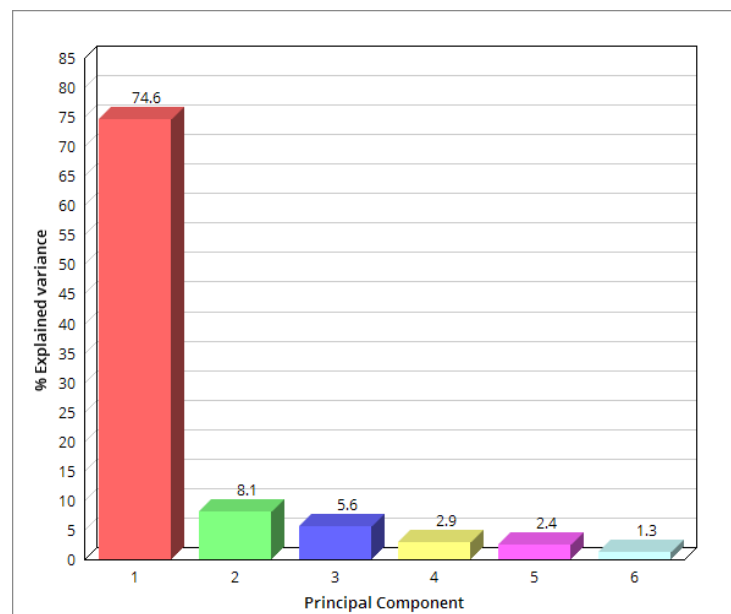


Figure 3.33 – Explained variance of each PC for beats A

Thus, the result vector for type A is:

$$\vec{v}_A = a_{A1} \cdot \overrightarrow{PC_1} + a_{A2} \cdot \overrightarrow{PC_2} + a_{A3} \cdot \overrightarrow{PC_3} + a_{A4} \cdot \overrightarrow{PC_4} + a_{A5} \cdot \overrightarrow{PC_5} + a_{A6} \cdot \overrightarrow{PC_6}$$

The vector of coefficients of type A is defined as:

$$a_A = [a_{A1}, a_{A2}, a_{A3}, a_{A4}, a_{A5}, a_{A6}]$$

For the representation of each PC, each coefficient a_{Ai} will get a value of 1 and the rest of them 0, so as it can be observed the individual influence of every PC in the resulting signal.

PC1

$$\mathbf{a}_A = [1, 0, 0, 0, 0, 0]$$

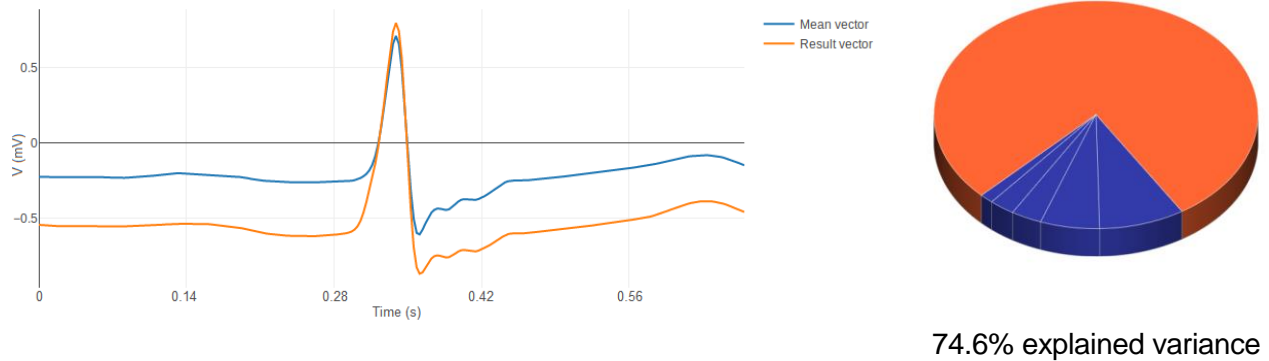


Figure 3.34 – Principal Component 1 for beats of type A

The first principal component shows a change in the baseline, while the R peak is kept constant.

PC2

$$\mathbf{a}_A = [0, 1, 0, 0, 0, 0]$$

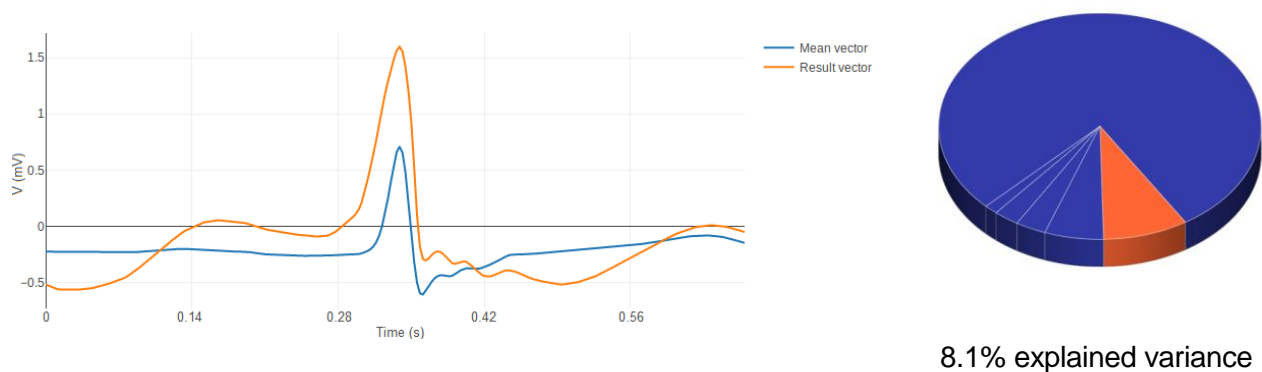
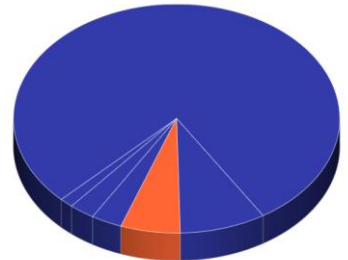
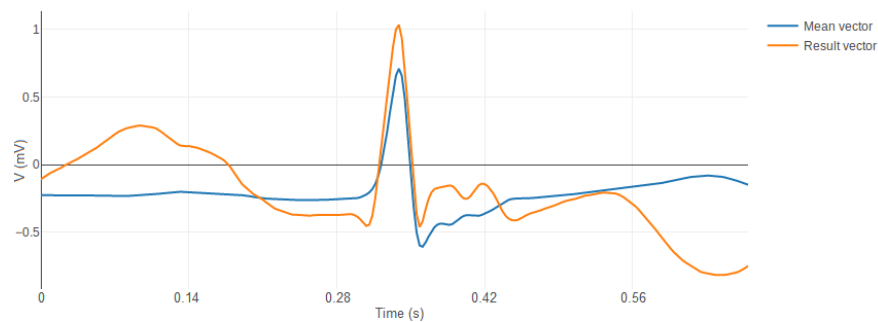


Figure 3.35 – Principal Component 2 for beats of type A

The PC2 exaggerates the R peak. Also, the P wave is slightly shown around the second 0.15, as it can be observed in the figure.

PC3

$$\mathbf{a}_A = [0, 0, 1, 0, 0, 0]$$



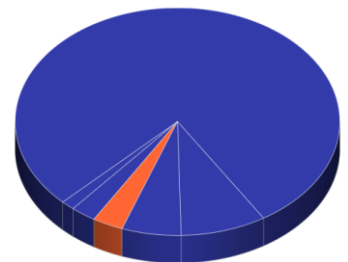
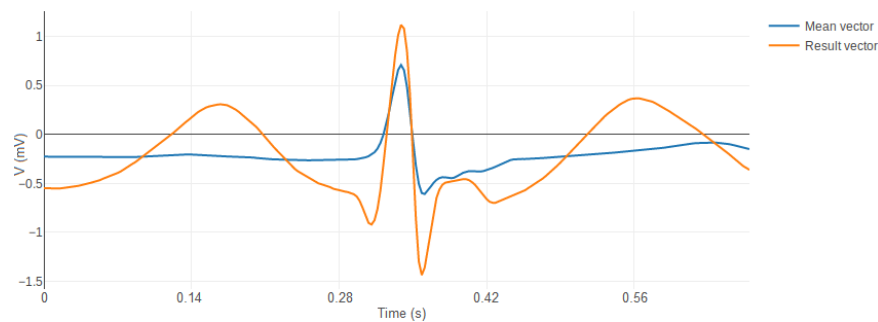
5.6% explained variance

Figure 3.36 – Principal Component 3 for beats of type A

The PC3 shows the T wave, even though there is a small decrease in the wave that could be caused by the premature atrial beat.

PC4

$$\mathbf{a}_A = [0, 0, 0, 1, 0, 0]$$



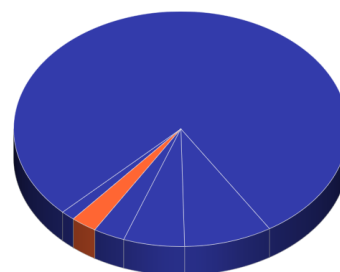
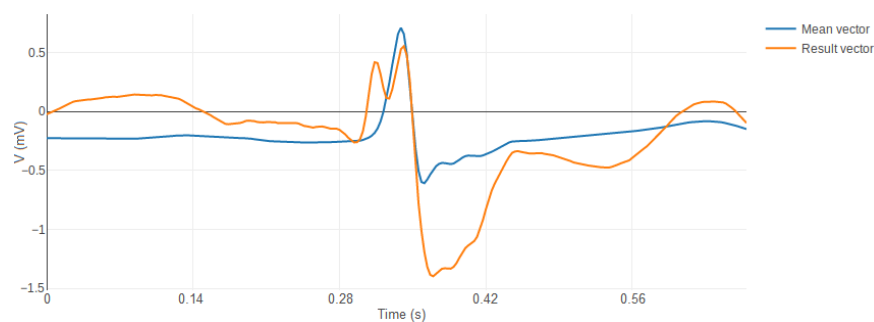
2.9% explained variance

Figure 3.37 – Principal Component 4 for beats of type A

The fourth component accentuates the shape of the QRS complex. The T wave is clearly observed as well. The first increase around second 0.16 could show the P wave.

PC5

$$\mathbf{a}_A = [0, 0, 0, 0, 1, 0]$$

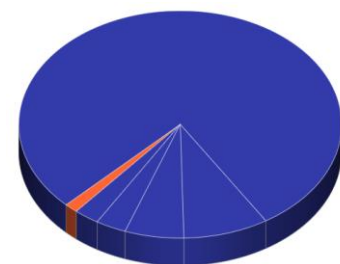
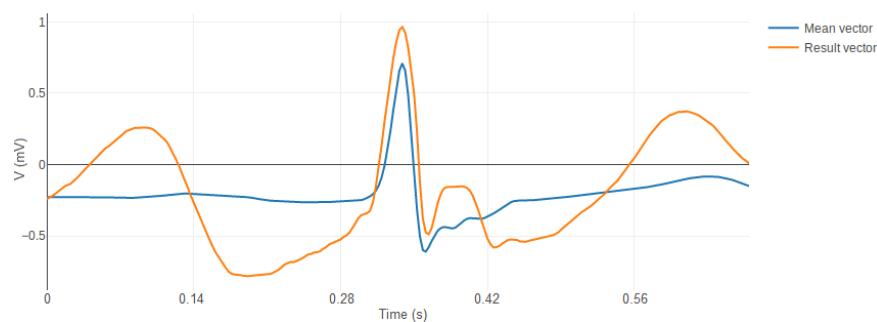


2.4% explained variance

Figure 3.38 – Principal Component 5 for beats of type A

PC6

$$\mathbf{a}_A = [0, 0, 0, 0, 0, 1]$$



1.3% explained variance

Figure 3.39 – Principal Component 6 for beats of type A

The sixth principal component slightly accentuates the shape of the QRS complex. Furthermore, the T wave is clearly shown around the second 0.40.

4. Deep autoencoder

In the previous principal component analysis, the studied types of heartbeats were the ones that had a higher number of samples in the MIT-BIH database. Following the same idea, the next proposal is to design an autoencoder to be trained with the same 5 types of heartbeats: normal (N); left bundle branch block beat (L); right bundle branch block beat (R); premature ventricular contraction (V); and atrial premature beat (A). To analyze these types of beats, a convolutional autoencoder is proposed. Before designing the neural network, the selected beats are extracted of the MIT-BIH database, as it is shown in the following capture of the Jupyter Notebook (Figure 4.1), being represented the type label and the number of beats in the database:

```
MLII_N    72945
MLII_L    8071
MLII_R    7257
MLII_V    7080
MLII_A    2536
Name: label, dtype: int64
```

Figure 4.1 – Beats extracted from the MIT-BIH Arrhythmia database

4.1. Convolutional autoencoder

The first considered solution is a convolutional autoencoder. The proposal is to use an input vector of 240 samples (as per in the PCA). The first stage consists of a convolutional layer followed by a max pooling layer, for dimensionality reduction. Then, a second stage is again a convolutional layer followed by a max pooling. The output of the max pooling is flattened to end up with a one-component vector. The final stage of the encoding part is based in fully connected layers (also called “dense”), for dense representation of the compressed input. The input data is finally represented by 10 neurons. The decoder has the same structure but arranged in reverse order.

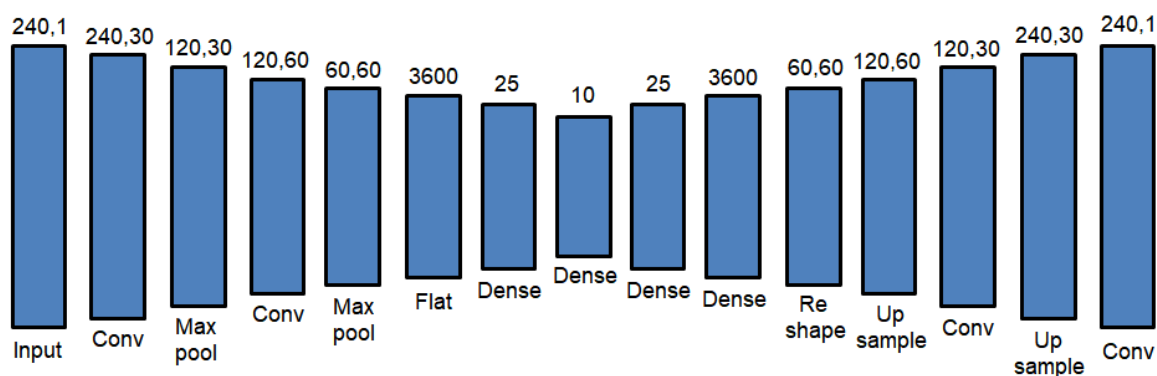


Figure 4.2 – Convolutional autoencoder proposal

The following Figure 4.3 shows the summary of the autoencoder as an output in the Jupyter Notebook, using the Keras library.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 240, 1)	0
conv1d_1 (Conv1D)	(None, 240, 30)	120
max_pooling1d_1 (MaxPooling1D)	(None, 120, 30)	0
conv1d_2 (Conv1D)	(None, 120, 60)	5460
max_pooling1d_2 (MaxPooling1D)	(None, 60, 60)	0
flatten_1 (Flatten)	(None, 3600)	0
dense_1 (Dense)	(None, 25)	90025
dense_2 (Dense)	(None, 10)	260
dense_3 (Dense)	(None, 25)	275
dense_4 (Dense)	(None, 3600)	93600
reshape_1 (Reshape)	(None, 60, 60)	0
up_sampling1d_1 (UpSampling1D)	(None, 120, 60)	0
conv1d_3 (Conv1D)	(None, 120, 30)	5430
up_sampling1d_2 (UpSampling1D)	(None, 240, 30)	0
conv1d_4 (Conv1D)	(None, 240, 1)	91

Figure 4.3 – Convolutional autoencoder with 10 neurons

The training consists of 20 epochs and the number of beats used to train the autoencoder is 2,000 for each class, so that every type of heartbeat has the same number of beats. As there are 5 classes considered, there are 10,000 beats in total, which correspond to 2,400,000 samples from the MIT-BIH database. After the training, the loss is represented in the following Figure 4.4:

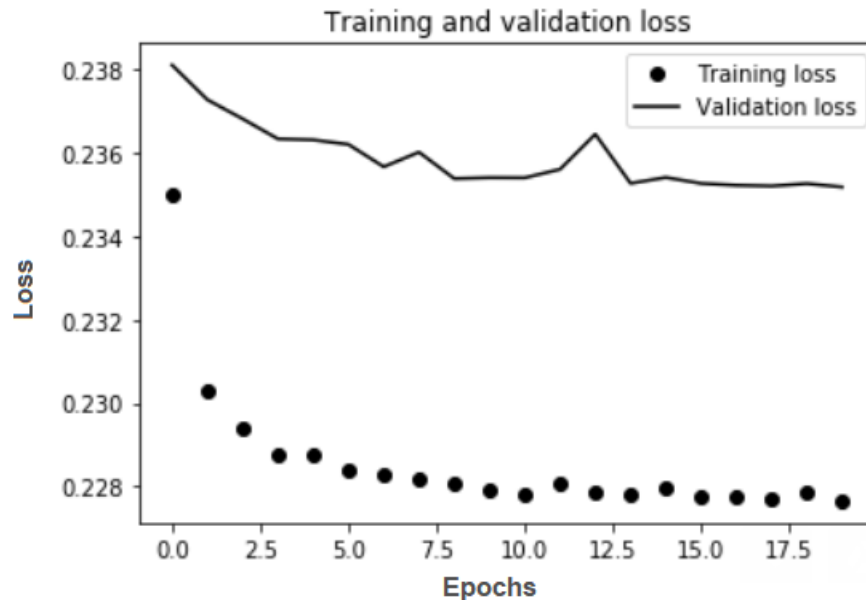
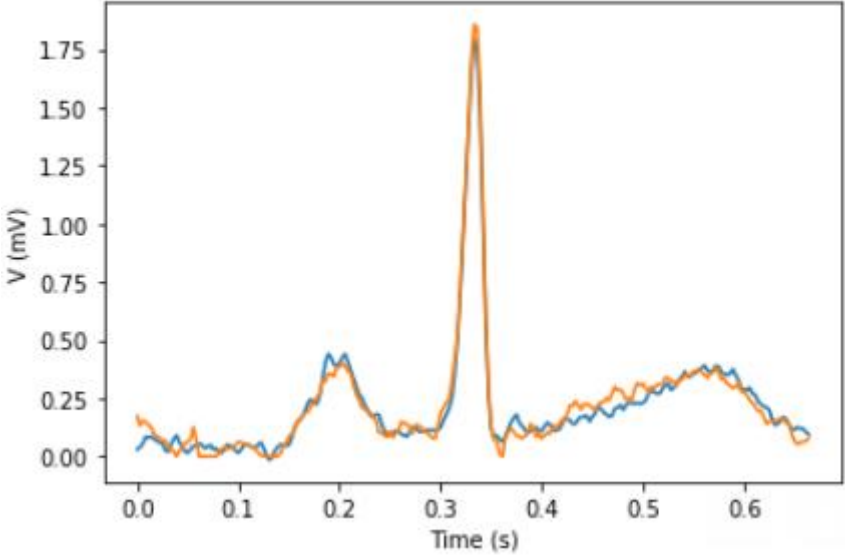
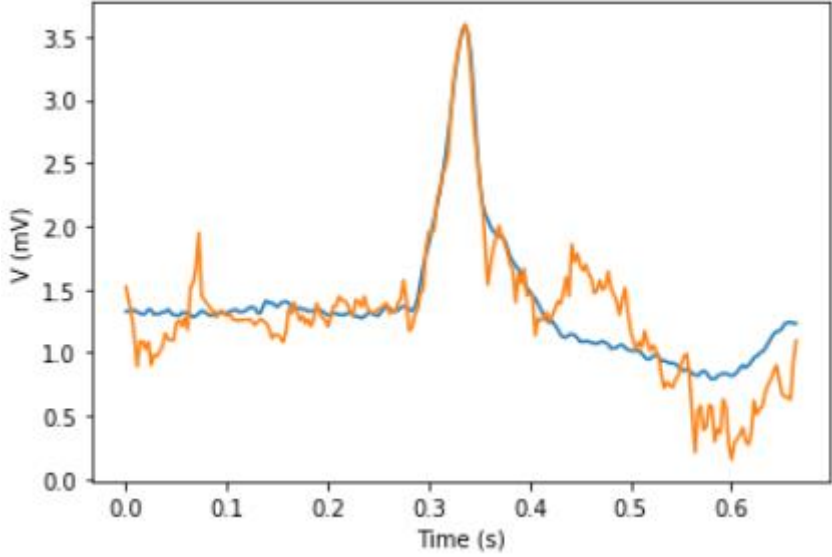
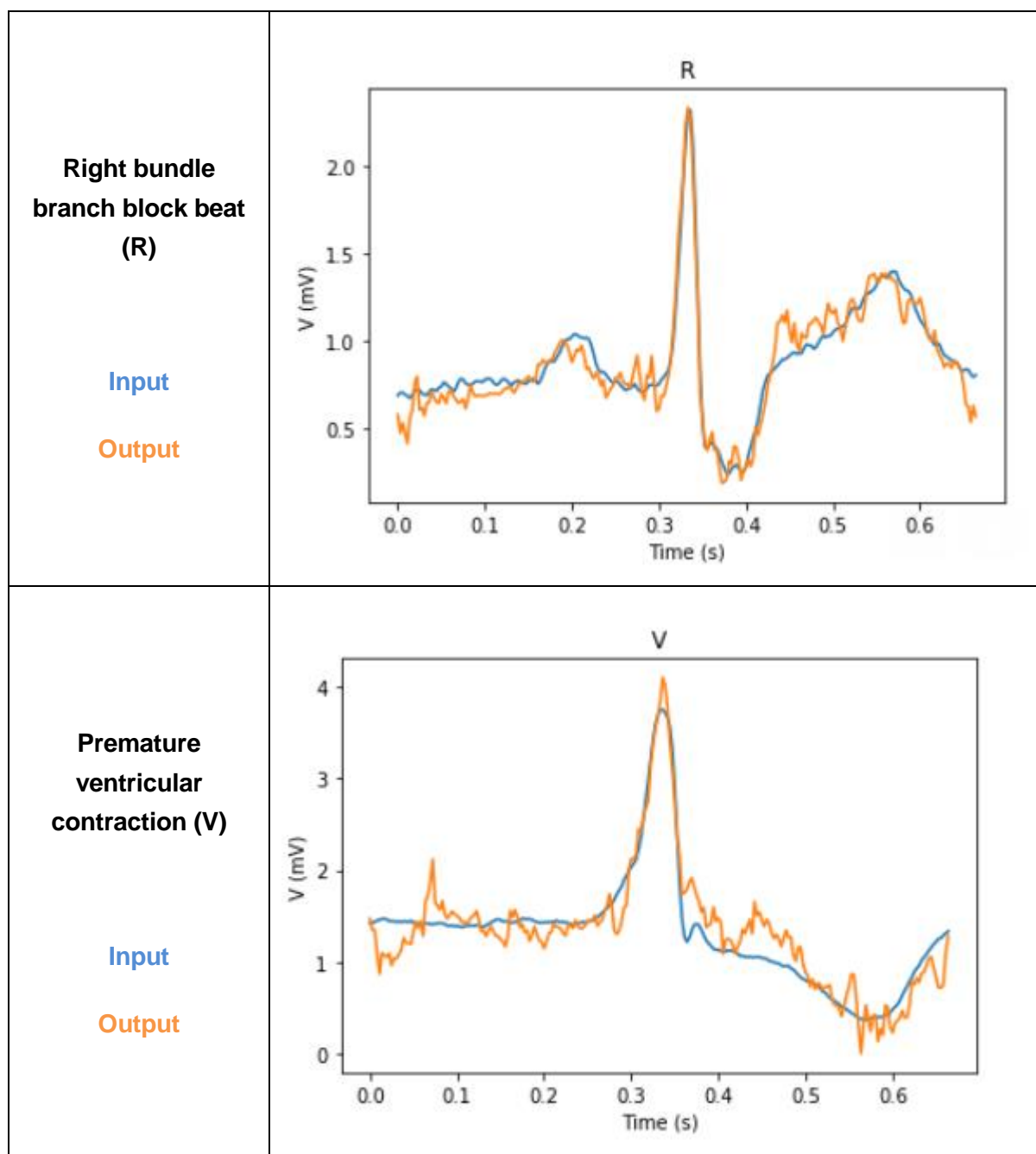


Figure 4.4 – Training and validation loss of the convolutional autoencoder

As it is observed, the loss decreases below 0.23 from the epoch 3, and becomes almost constant from epochs 5 to 20. The value of the loss shows a relatively good performance of the autoencoder, considering around a 23% of loss during decodification.

Furthermore, a beat of each type has been used as an input of the trained autoencoder, so as to see how properly it can reconstruct each type of beat. In the following Table 4.1, there is a graph for every class of beat. The blue line corresponds to the input applied to the autoencoder, whereas the orange line represents the output of the autoencoder:

<div>Normal beat (N)</div> <div><div>Input</div><div>Output</div></div>	<div>N</div> 
<div>Left bundle branch block beat (L)</div> <div><div>Input</div><div>Output</div></div>	<div>L</div> 



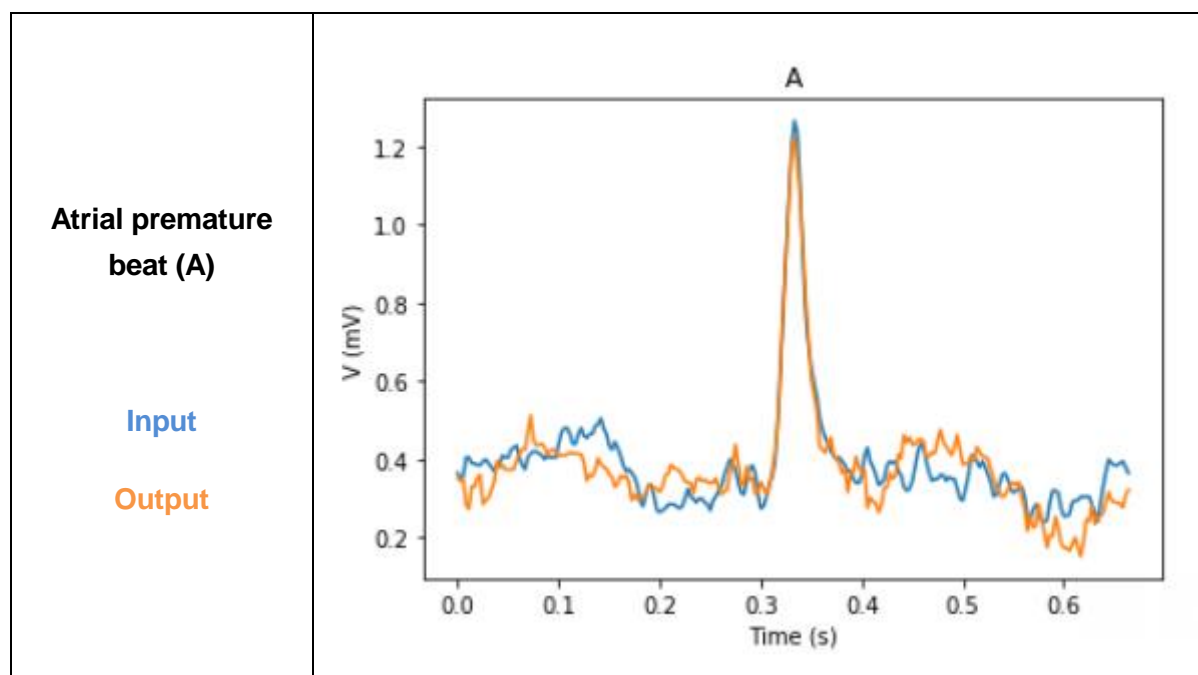


Table 4.1 – Comparison between input and output of the convolutional autoencoder

The similarities between the input and the output of the autoencoder have been tested by calculating the correlation between both signals. The obtained values are the following:

Type of beat	N	L	R	V	A
Correlation coefficient	0.99	0.89	0.96	0.93	0.92

Table 4.2 – Correlation coefficient between input and output signals

Finally, the neurons representation for each one of the types under study is shown in the following Figure 4.5:

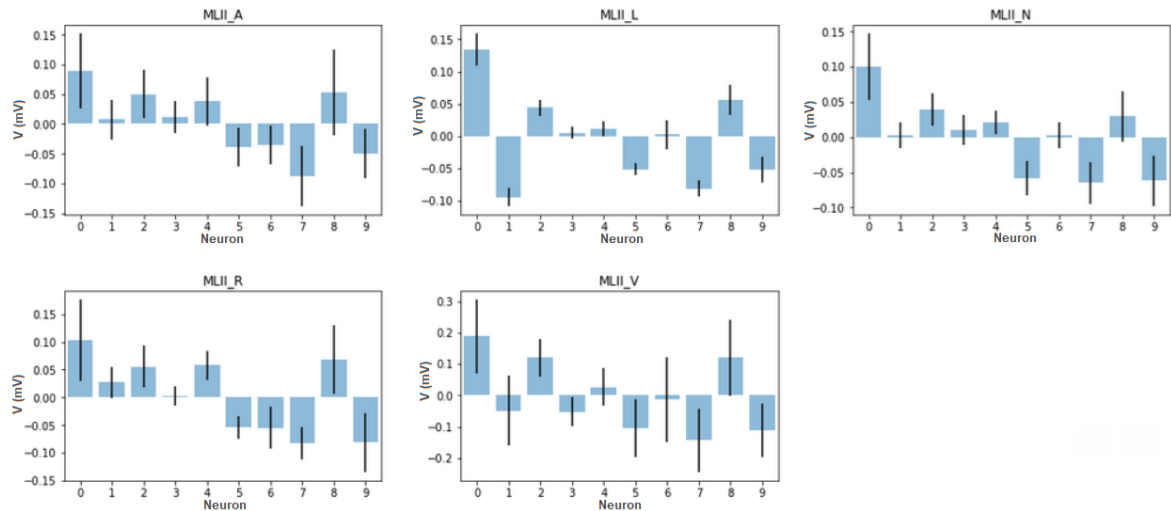


Figure 4.5 – Neuron representation for every type of heartbeat (A, L, N, R and V)

As it is observed in the previous figure, neuron representation for types N and A is slightly similar, even though the values are not the same. This could be due to the fact that the atrial premature beat (A) has a similar shape compared to the normal beat, but the difference is in the frequency between two consecutive beats. Since the present study is comparing individual beats, the shape of both beats is slightly similar, and thus the neuron representation should show resemblance as well.

To further reduce the dimension of the dataset, an autoencoder with the same types of layers but with a representation with only 5 neurons has been trained as well. The layer configuration of the autoencoder is shown in the following Figure 4.6:

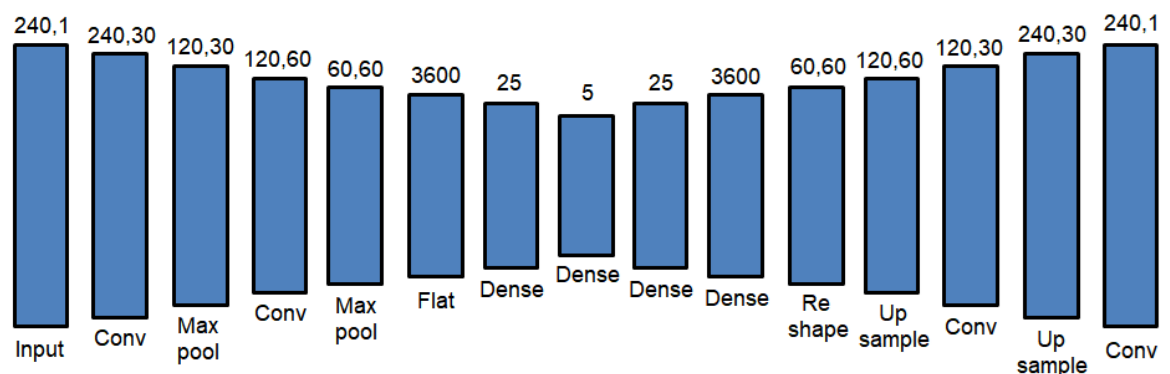


Figure 4.6 – Convolutional autoencoder with only 5 neurons

The calculated loss values are similar as per in the first autoencoder, as it is observed in the following Figure 4.7.

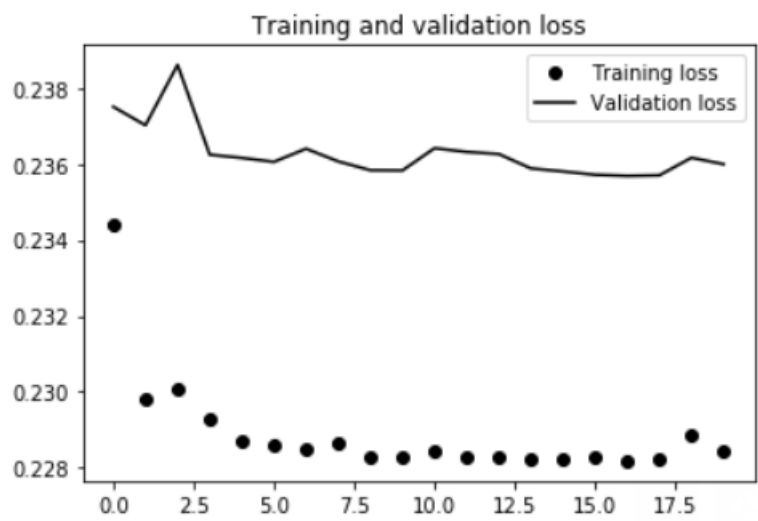
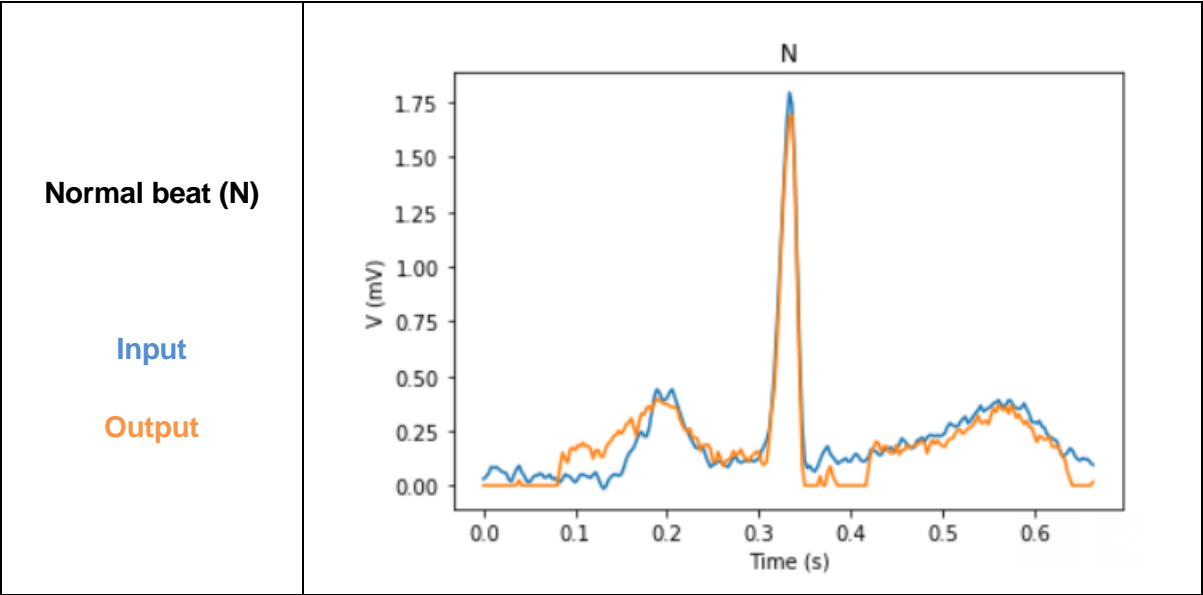
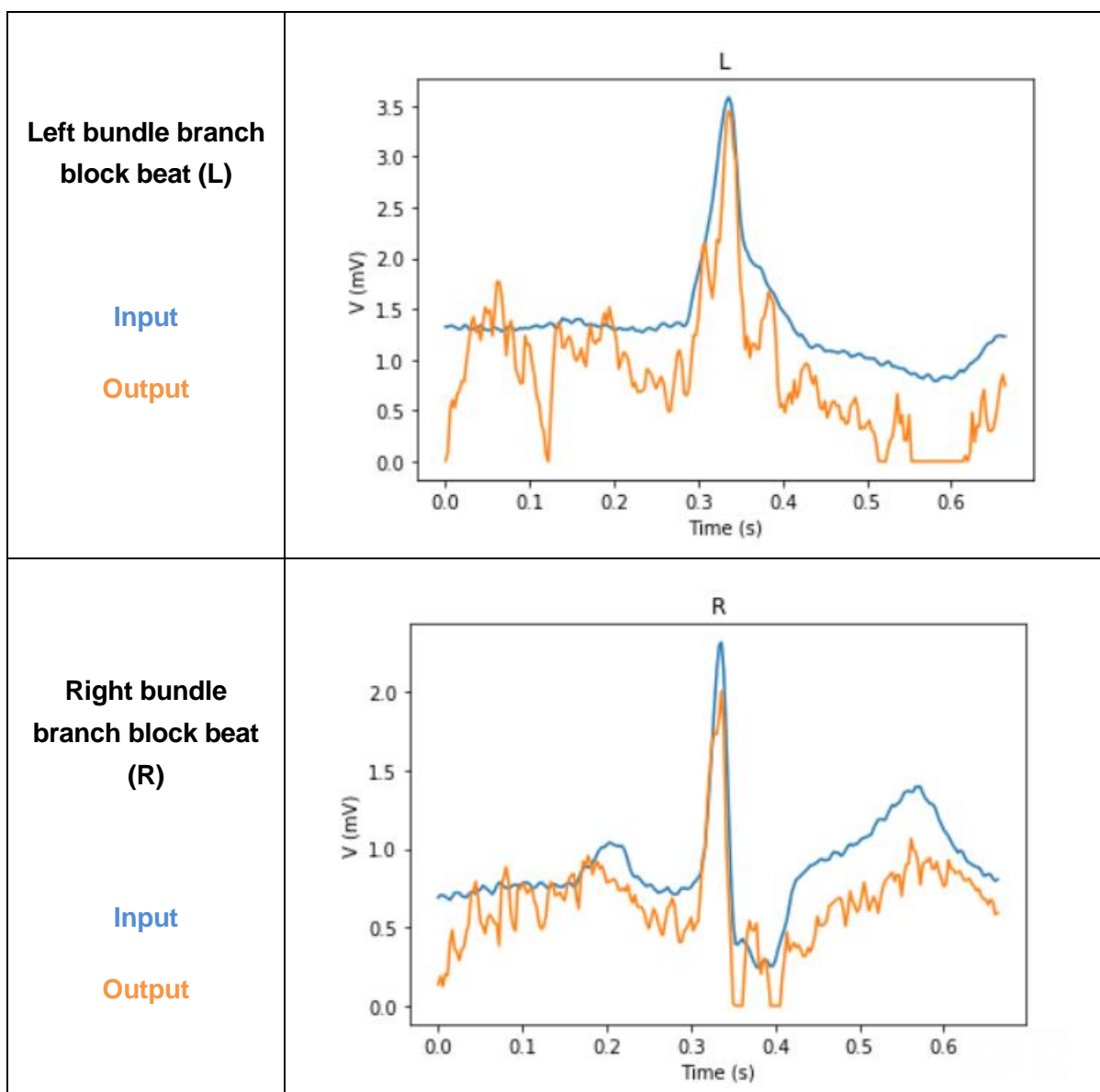


Figure 4.7 – Training and validation loss of the convolutional autoencoder with 5 neurons

As per in the previous autoencoder, the input and output signal are compared to test the performance obtained with 5 neurons. The results are presented in the following Table 4.3.





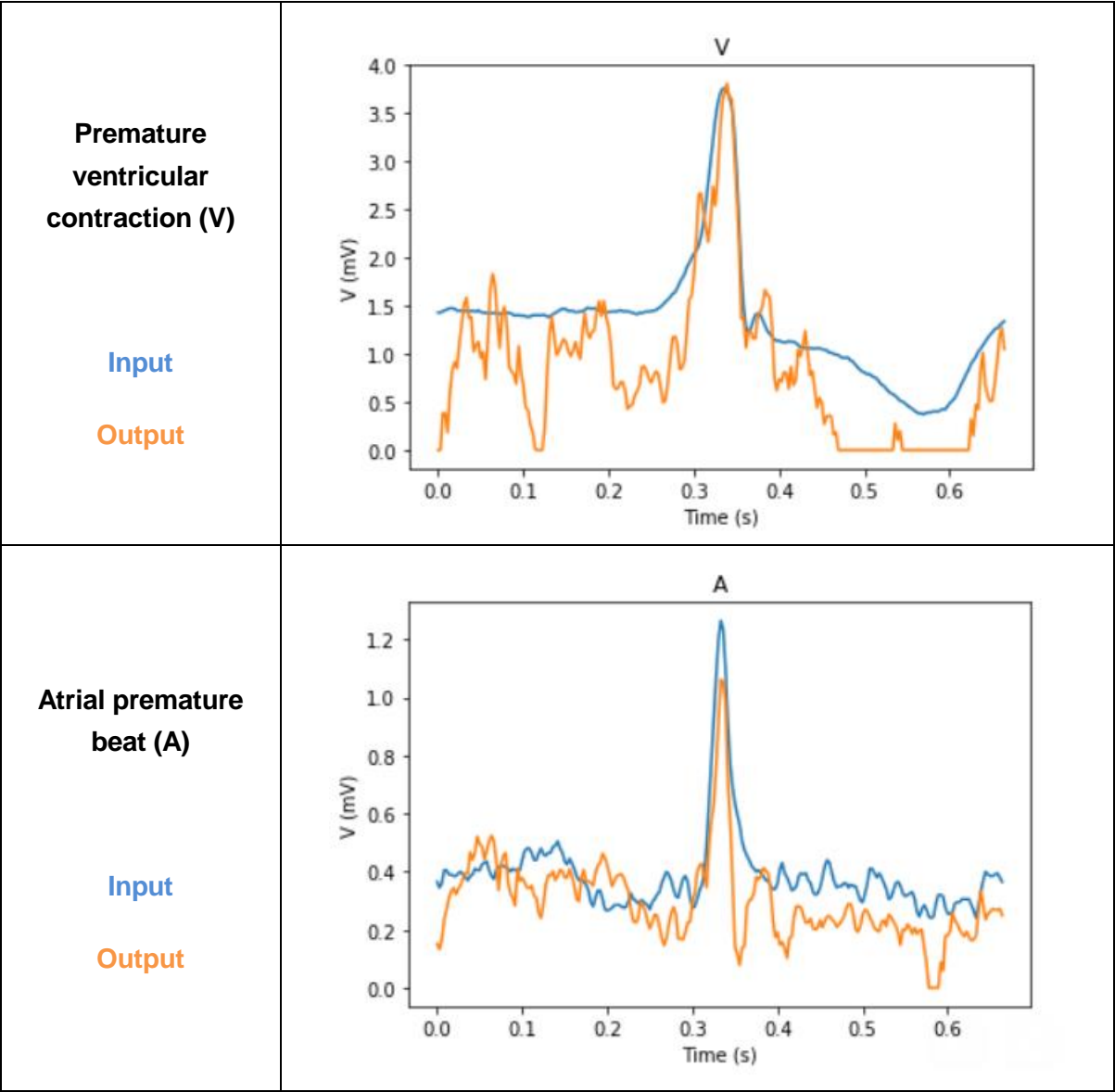


Table 4.3 – Comparison between input and output of the autoencoder with 5 neurons

The similarities between the input and the output of the autoencoder with 5 neurons have been tested by calculating the correlation between both signals. The obtained values are the following:

Type of beat	N	L	R	V	A
Correlation coefficient	0.95	0.86	0.83	0.89	0.72

Table 4.4 – Correlation coefficient between input and output signals

As it is observed, the reconstruction in this case is not as accurate as per in the convolutional autoencoder with 10 neurons. The correlation coefficients are lower for the case with 5 neurons, and thus it can be concluded that the first convolutional autoencoder has a better performance. Trying to reduce too much the dimensionality of the data can cause an incorrect reconstruction of the beat, and hence an incorrect distinction between types of beats.

4.2. Recurrent LSTM autoencoder proposal for future studies

Recurrent Long Short-Term Memory (LSTM) autoencoders are capable of learning the complex dynamics within the temporal ordering of input sequences as well as use an internal memory to remember or use information across long input sequences. The LSTM network can be organized into an architecture called the Encoder-Decoder LSTM that allows the model to be used to both support variable length input sequences and to predict or output variable length output sequences. In other words, the LSTM autoencoder can be used to predict next steps in a sequence. Actually, it can also be created a composite LSTM autoencoder that has a single encoder and two decoders, one for reconstruction and one for prediction.

As the LSTM autoencoder supports variable length input sequences, and besides has memory cells that allow to predict next steps in a sequence, an interesting option for future work would be to use this kind of autoencoders to detect arrhythmia with a better performance than of other types of deep neural networks.

In a recent study presented by Singh [10], it were used recurrent LSTM neural networks to classify ECG signals in two groups: normal beats (N) and arrhythmia beats (A). The accuracy of the classification in this paper is 88.1%. A proposal for future work would be to improve this classification by detecting not only the state of arrhythmia but also the type of beat, as per in the presented case of with the convolutional autoencoder.

Conclusions

According to the presented analysis of the ECG signals, it can be concluded:

First, regarding the ECG visualizer, the option that best fits the requirements for a user to select and display the desired signals is the one implemented with the Dash library, as it includes the user interface and the graph is updated in real time when the user changes the settings.

Second, the dimensionality of the data has been successfully reduced by applying a Principal Component Analysis for each one of the following heartbeats available in the MIT-BIH Arrhythmia database: normal beat (N); left bundle branch block beat (L); right bundle branch block beat (R); premature ventricular contraction (V); and atrial premature beat (A). The number of Principal Components needed to explain, at least, the 95% of the variance is 5; 5; 3; 7; and 6, respectively.

Third, the compression of the beats has been achieved by means of a convolutional autoencoder. The number of neurons in the central layer is 10. The loss values are below 23% and the correlation between the input signal and the output of the autoencoder is between 0.89 and 0.99 for all the considered types of heartbeat. For further compression, an autoencoder with the same configuration but using only 5 neurons has been trained too. Both the loss values and the correlation between the input signal and the output of the autoencoder have shown worse results in the autoencoder with 5 neurons. Hence, it can be concluded that the autoencoder with 10 neurons has a better performance and thus is able to detect more effectively the different types of heartbeats.

Finally, it is proposed, for future work, to use recurrent LSTM autoencoders that would allow both to be trained with input signal of variable length (thus, it would not be needed a beat segmentation before the training) and also to predict next steps in ECG sequences. These features should improve the performance of the autoencoder, and hence the accuracy of the classification of ECG signals.

Bibliographic references

- [1] De Chazal, P., O'Dwyer, M., & Reilly, R. B. (2004). Automatic classification of heartbeats using ECG morphology and heartbeat interval features. *IEEE transactions on biomedical engineering*, 51(7), 1196-1206.
- [2] Al Rahhal, M. M., Bazi, Y., AlHichri, H., Alajlan, N., Melgani, F., & Yager, R. R. (2016). Deep learning approach for active classification of electrocardiogram signals. *Information Sciences*, 345, 340-354.
- [3] Hannun, A. Y., Rajpurkar, P., Haghpanahi, M., Tison, G. H., Bourn, C., Turakhia, M. P., & Ng, A. Y. (2019). Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature medicine*, 25(1), 65.
- [4] Hope T., Resheff Y. S., Lieder I., (2017), *Learning TensorFlow: A guide to building deep learning systems*, United States of America, O'Reilly
- [5] PHYSIONET, [<https://physionet.org/physiobank/database/mitdb/>, 7th of March 2019]
- [6] WFDB, [<https://pypi.org/project/wfdb/>, 14th of March 2019]
- [7] TKINTER, [<https://wiki.python.org/moin/TkInter>, 21st of March 2019]
- [8] BOKEH, [<https://bokeh.pydata.org/en/latest/>, 11th of April 2019]
- [9] DASH, [<https://dash.plot.ly/>, 25th of April 2019]
- [10] Singh, S., Pandey, S. K., Pawar, U., & Janghel, R. R. (2018). Classification of ECG arrhythmia using recurrent neural networks. *Procedia computer science*, 132, 1290-1297.

The developed codes are public and available in the following repository:

<https://github.com/MaichelWheel/TFM>