UNIVERSITÉ LIBRE DE BRUXELLES
Faculty of Sciences
Department of Computer Science

Paroxysmal atrial fibrillation forecast: deep learning with RR intervals and reproducibility of existing research

Jérôme Hellinckx

Supervisor :
Professor Hugues Bersini

Master Thesis in Computer Sciences

Academic Year 2020 - 2021

Remerciements

Tout d'abord merci à mon promoteur, Hugues Bersini, pour m'avoir donné l'opportunité d'effectuer ce mémoire et de m'avoir permis de rencontrer les acteurs du groupe de travail NNAFCA.

Ensuite, merci aux cardiologues Jean-Marie Grégoire et Stéphane Carlier, qui ont partagé leurs connaissances médicales sans réserve tout au long de notre collaboration. Je ressors de cette expérience grandi d'un petit savoir médical dont je suis fier.

Merci à ma famille; ma maman, mon papa, mon frère et ma soeur pour m'avoir encouragé, épaulé et ravitaillé en énergie quand j'en avais le besoin. Merci à eux d'avoir supporté mes humeurs souvent très polarisées en cette période de pandémie. Merci aussi à Donovan et Tine pour leur soutien.

Merci à Thomas et Tom, dont les précieuses amitiés m'auront permis de garder la tête hors de l'eau durant ces nombreux mois.

Merci à mes amis; Raphaël, Aurélien, Éléonore, Alexis, Julien, Pierre-Victor et Benjamin qui ont toujours été là pour me soutenir.

Merci à Mélanie pour toute son aide et son soutien.

Enfin, *last but not least*, un merci tout particulier à Cédric, doctorant du projet NNAFCA, qui aura été d'une abnégation sans concessions durant toute la durée de mon travail. Merci pour tes conseils avisés, ton immense bienveillance et ta générosité.

Contents

List of Acronyms

AF Atrial Fibrillation

AFPDB Atrial Fibrillation Prediction Database

AI Artificial Intelligence

ANS Autonomic Nervous System

AUC Area Under the ROC Curve

CNN Convolutional Neural Network

DL Deep Learning

DNN Deep Neural Network

ECG Electrocardiogram

FC Fully Connected

FFT Fast Fourier Transform

FN False Negative

FP False Positive

GA Genetic Algorithm

GRU Gated Recurrent Unit

HF High Frequency

HOSA Higher Order Spectral Analysis

k-NN k-Nearest Neighbours

LF Low Frequency

LSTM Long Short-Term Memory

ML Machine Learning

NN Neural Network or Normal to Normal (interval)

NSR Normal Sinus Rhythm

PAC Premature Atrial Contraction

PAF Paroxysmal Atrial Fibrillation

PSD Power Spectral Density

PVC Premature Ventricular Contraction

QRS ECG waves composing one heartbeat

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

ROI Region of Interest

RR R-wave to R-wave (interval)

SVM Support-Vector Machine

TN True Negative

TP True Positive

VLF Very Low Frequency

Chapter 1

Introduction

1.1 Context and goals

Atrial fibrillation (AF) is a type of arrhythmia (abnormal heart rhythm disease) that, unless treated timely, can lead to a high rate of morbidity and mortality [20]. The diagnosis of AF is generally assessed by healthcare professionals via electrocardiograms (ECG) signals, which are recordings of the electrical activity of the heart. However, manual visual inspection of ECG signals is time-consuming, requires extensive training and experience [21] and can, moreover, be difficult due to the existing noise in the signal [26].

In this context, software methods for the automated analysis of ECG signals have been extensively studied in the literature. In particular, with the recent promising successes of deep learning techniques for image recognition, speech recognition and natural language understanding [33] (among others), applications of deep learning for automated detection of arrhythmias have emerged. For instance, Hannun et al. [27] designed a model that only used the raw ECG signal as input for multi-class arrhythmias detection. Hence, no features were manually extracted from the signal and the deep learning model learned the relevant features by itself from data. In this study, it is shown that the presented system performs, on average, better than cardiologists.

This work is a follow-up and complementary work of the masters thesis realized by Gilon [23] in 2019. We will make the same distinction between atrial fibrillation **diagnosis** and **forecast** as in [23]. That is, AF diagnosis is the detection of an AF episode while it is happening (*a posteriori*) as opposed to AF forecast, which consists of detecting the onset of AF in electrocardiograms preceding the AF episodes (*a priori*).

The inititial main objective of this work is to **improve** the **results** of **Gilon** [23] on the **AF forecast** task. This improvement is to be guided by the medical expertise of Dr. Jean-Marie Grégoire, a cardiologist who thus acts as a medical consultant for this work.

1.2 Structure

In this work, we will first present the medical aspects of atrial fibrillation and introduce some fundamentals of deep learning which lay the theoretical foundations of our implemented algorithms.

We will then assess the current trends in the literature for AF forecast systems before trying to understand how we can link medical knowledge to discriminating features.

To compete with Gilon's [23] network, we propose an alternative architecture and compare the performances. We then augment the two models by injecting them with strategic features and evaluate the eventual improvements.

Finally, our results will convince us to attempt to reproduce state of the art methods of AF forecast systems that we presented earlier.

Chapter 2

The human heart

2.1 Anatomy and physiology

The primary function of the human cardiovascular system is to deliver materials (nutrients, oxygen, etc.) carried by the blood to and from all parts of the human body. The blood is pumped by the heart through a system of blood vessels which enables the blood to circulate continuously, hence making collection and distribution of these crucial materials possible [9].

Anatomically, the heart is divided into two halves functioning as separate pumps: a right heart that pumps blood through the lungs and a left heart that pumps blood through the peripheral organs. Each heart is composed of two chambers: an **atrium** (the upper part) and a **ventricle** (the lower part). Blood flows through each heart in one direction — the atria serve as reservoirs for blood returning to the heart and the ventricles propel the collected blood through the circulation. Direction of blood flows is orchestrated by the valve system. [25] (Figure 2.1).

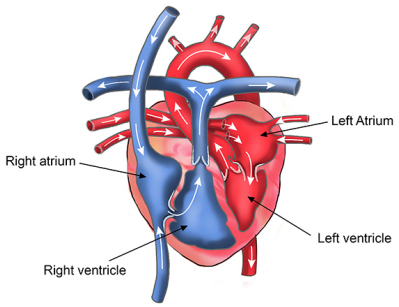


Figure 2.1: Visualisation of the two separated inner hearts and their chambers in the human heart; the left heart is colored in red, the right heart is colored in blue.

Rhythmical **electrical impulses** are generated by the sinus node, a small cellular cluster located in the right atrium which functions as the **natural pacemaker** of the heart. These impulses are conducted through the heart and cause consecutive contractions of the different heart muscles (notably the atria and the ventricles),

enabling the pumping of the blood and hence its circulation. One iteration of subsequent contractions of the atria and the ventricles is called a **heartbeat** [25]. Usually in a healthy heart a resting heart rate of 60 to 100 beats per minute is observed.

2.2 Electrocardiogram

While the cardiac impulse travel through the heart and provoke the contractions of the cardiac muscles, a portion of the electrical current is also spread to adjacent tissues surrounding the heart all the way to the surface of the body. By placing electrodes on the skin near the heart, the electrical potentials generated by this current can be recorded. The recording of this signal is known as an **electrocardiogram** (ECG) [25]. An example of a normal electrocardiogram with 4 recorded heartbeats is shown in Figure 2.2.

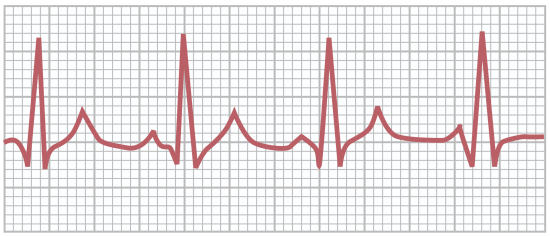
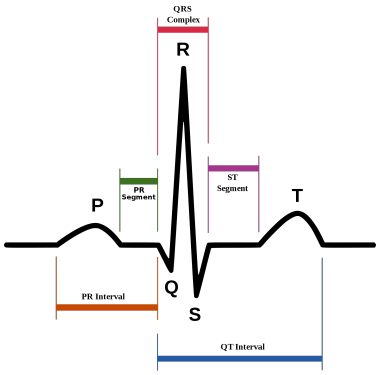
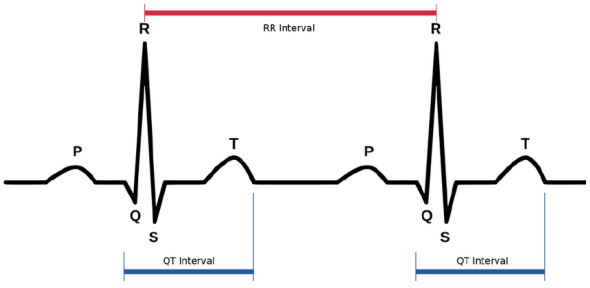


Figure 2.2: Example of a normal electrocardiogram capturing 4 heartbeats [25].

The normal electrocardiogram exhibits clear characteristics for each recorded heartbeat. Indeed, when the heart behaves normally, each heartbeat on the ECG is composed of different distinguishable waves : the P wave, the Q wave, the R wave, the S wave and the T wave (Figure 2.3a). The standard composition of these waves is called **Normal Sinus Rhythm** (NSR).



(a) The different waves that compose a single normal heartbeat on an electrocardiogram.



(b) The time between two consecutive R waves is called the RR interval.

Figure 2.3: Electrocardiogram characteristics.

These different waves are caused by the electrical impulse when it passes through the different muscles of the heart it stimulates : the P wave indicates the signal

passing trough the atria and the QRS complex is a result of the current traveling through the ventricles [25]. Because of its significance, the QRS complex can be easily used to detect one heartbeat. In particular, the time between two R waves is denoted the **RR interval** and can hence be used to measure the time between two heartbeats (Figure 2.3b).

2.3 Atrial fibrillation

2.3.1 Definition

The previous section presented electrocardiograms where the heart was beating normally (normal sinus rhythm). When the heart beats too early, late, slowly, quickly or irregularly it is said that an **arrhythmia** occurs. These heart rhythm problems surface when the electrical impulse coordinating heartbeats is dysfunctional. These malfunctions can take many different forms and among them the most common one is **atrial fibrillation** (AF).

Atrial fibrillation results from electrical impulses behaving in a completely chaotic fashion within the atria, stimulating first one portion of the atrial muscle, then another portion, then another, and eventually the signal is fed back into the same atrial muscle and the process repeats, stimulating the atrial muscle over and over again, causing the muscle to quiver, or **fibrillate**. Because of these localized stimulations within the same atrial muscle, some portions of the muscle are contracted while some other portions are relaxed at the same time. The absence of synchronized or coordinated contractions within the atrial muscle prevent the atria to pump blood.

Even though atria essentially become useless pumps in AF, blood still flows passively trough the atria into the ventricles, only the efficiency of the ventricular pumping is affected (20-30 percent decrease). Therefore, a person can live for months or even years with atrial fibrillation [25]. AF is hence not life-threatening short-term, although it is still a critical condition that can progress and lead to stroke, heart failure and even death if left untreated [26]. Early diagnosis of AF is thus crucial to enable early and efficient treatment in order to prevent progression of the condition to more serious cardiac issues [26].

2.3.2 AF diagnosis

Diagnosis of AF is achieved by healthcare professionals by analyzing ECG signals of the patient. This is because specific patterns of the ECG waves can be interpreted to assert certain conditions of the heart, especially arrhythmias. In the case of AF, the weak and chaotic nature of the electrical impulse in the atria are characterized by either the complete **absence of the P waves** in the ECG or the replacement of the P waves by inconsistent and fibrillatory waves [25, 26]. The unsynchronized impulses also translate into **irregular QRS timings**. Figure 2.4 shows an example of an ECG in AF.

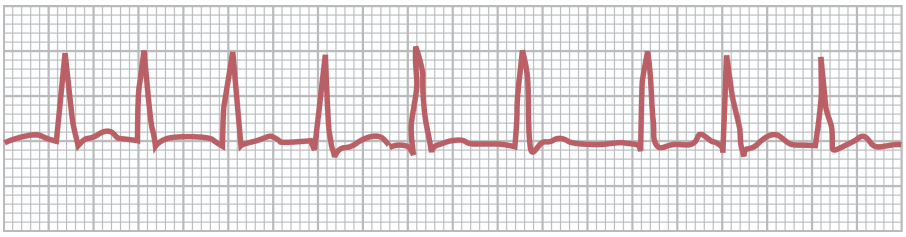


Figure 2.4: Characterization of atrial fibrillation in the ECG signal. P waves are absent and the timings of QRS complexes are irregular [25].

2.3.3 AF initiation factors: Coumel's Triangle

The onset of AF is linked to the interactions between different arrhythmogenic factors, which were first formalised by Philippe Coumel [18, 19] (Figure 2.5).

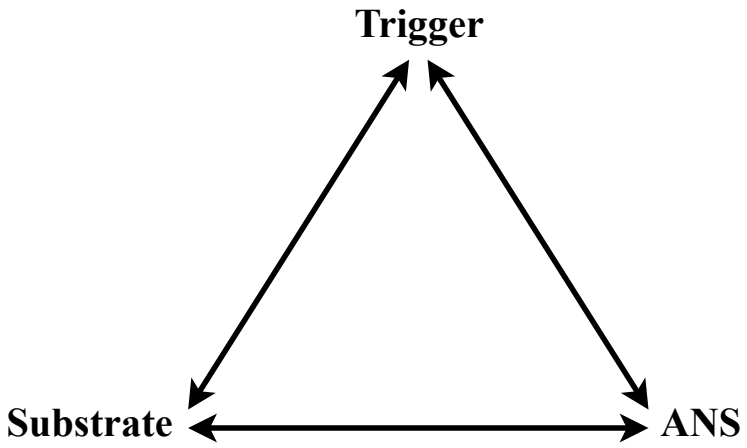


Figure 2.5: Coumel's Triangle.

The vertices of Coumel's Triangle represent the necessary components for the onset of AF:

- The substrate is the anatomical condition which favors the retention of the arrhythmia; e.g. a scar originating from a myocardial infarction or a congenital or hereditary abnormality of the myocardial tissue.
- The **trigger** is the electrical element starting the arrhythmia, which is often a **premature atrial contraction** (PAC).
- The **Autonomic Nervous System** (ANS) acts on the first two factors by augmenting the sensitivity of the arrhythmogenic substrate and the conductivity of tissues which facilitates AF initiation [19].

Chapter 3

Deep learning

In this section, we introduce the classical deep learning models that are applied on ECG patterns recognition in the literature and present an overview of different techniques and tools used in a typical machine learning setting. We, however, start by giving some general foreword on artificial intelligence, machine learning and deep learning to clarify the relations between these fields.

Artificial intelligence is a multidisciplinary field that is interested in the study of designing and building *intelligent* machines. The term intelligence refers to the fact that these machines are required to solve some tasks that would normally require human intelligence, e.g. read and comprehend human language text, recognize and translate spoken human language into text, perceive and recognize objects in the real world, etc.

Machine learning is a subfield of artificial intelligence in which the machine is not explicitly programmed to follow a given set of instructions in order to solve some task but rather *learns* how to act by acquiring its own set of rules from experiences. The term learning is, however, somewhat misleading; machines don't really learn as humans do, and one could say that machine learning is in many cases merely an application of statistics.

Deep learning is a subfield of **representation learning** (which in itself is a subfield of machine learning) where the emphasis is put on learning a **hierarchical representation** of the input data through the different layers of **neural networks** [33].

3.1 Supervised learning

In machine learning, the process of learning from experiences can be categorized into different *learning types* : supervised learning, unsupervised learning and reinforcement learning (among others). Since the aim of this work is to classify ECG signals by making use of labeled examples, we will focus this section on **supervised learning** only.

In supervised learning, the experiences to which the machine is exposed take the form of a set of data called the **dataset**. This dataset is a collection of **labeled**

examples (x, y) (a **data point**) where each element x is called a **feature vector**. Each dimension in the feature vector contains a value called a **feature** that describes the example, gives some measured information for this example. Each element y associated with the feature vector x is called the **label** for this example and can be either a real number when trying to make real-valued predictions (regression) or an element belonging to a finite set of **classes** when the task at hand is a **classification problem**.

The goal of a **supervised learning algorithm** is to find a function f called **model** that takes as input an unseen feature vector x and predicts a correct output $\hat{y}$ (a **prediction**). To achieve that, the model is composed of **learnable parameters** (i.e. coefficients of the function) that are **tuned** during a **training phase**. During training, we feed the feature vector of an example from the dataset to the model, compare the output $\hat{y}$ with the known associated label y, and update the learnable parameters in order to **minimize** some predefined error measure known as the **cost function**.

An example of a classification learning algorithm is the logistic regression. In the case of binary classification, the logistic regression model is a **linear function** of the feature vector x on which we apply the **sigmoid function** $g(.)$ (Figure 3.1):

$$f_{w,b}(x) = g(wx + b) \tag{3.1}$$

where the sigmoid function is defined as

$$g(x) = \frac{1}{1 + e^{-x}}. \tag{3.2}$$

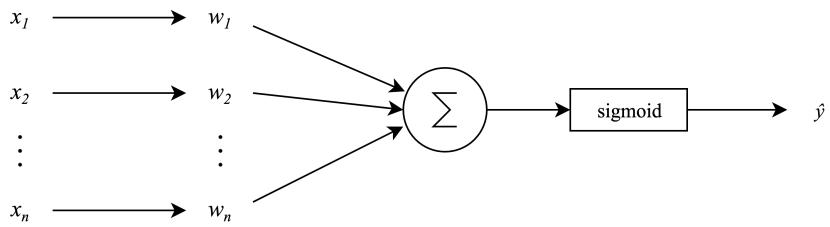


Figure 3.1: The logistic regression model.

The sigmoid function has the nice property of having a codomain $(0, 1)$ as can be observed in Figure 3.2.

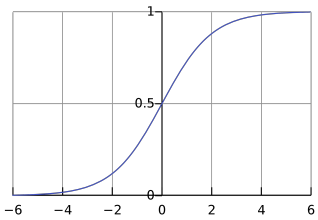


Figure 3.2: The sigmoid function.

Hence, when the value returned by our model $f_{w,b}$ for the input x is close to 0, we assign x to the first class, otherwise we assign it to the second class. This is achieved

by choosing a threshold on the value returned by the model; if the value is greater than this threshold we say the prediction is 1 else it is 0. By fixing the threshold at 0.5 we thus have :

$$\hat{y} = \begin{cases} 1 & \text{if } f_{w,b}(x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

In this model, w and b are the learnable parameters that need to be tuned in order to minimize the cost function. The cost function can actually be seen as the average of the **losses** over the whole dataset :

$$\mathcal{J}_{w,b} = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_{w,b}(x_i). \tag{3.4}$$

A **loss function** measures, for one example, how far the prediction $\hat{y}$ is from the known true label y. In logistic regression, the loss function is the **cross-entropy loss** defined as :

$$\mathcal{L}_{w,b}(x) = -(y \log(f_{w,b}(x)) + (1-y) \log(1 - f_{w,b}(x))). \tag{3.5}$$

The problem that we are trying to solve is thus an **optimization problem** in which the objective function we want to minimize is the cost function and the parameters are the weights given by the weight vector w and the bias b. The standard optimization algorithm used in machine learning is called **gradient descent**. The first step of gradient descent is to compute a **gradient** which is a vector of partial derivatives of a function. In our case, the gradient will contain partial derivatives of the cost function with respect to each learnable parameter :

$$\nabla \mathcal{J} = \Big[\frac{\delta \mathcal{J}}{\delta w_1}, \frac{\delta \mathcal{J}}{\delta w_2}, \ldots, \frac{\delta \mathcal{J}}{\delta w_k}, \frac{\delta \mathcal{J}}{\delta b} \Big]. \tag{3.6}$$

These partial derivatives are then used to update w and b using a learning rate α which controls the size of the update :

$$\begin{aligned} w_i &= w_i - \alpha \frac{\delta \mathcal{J}}{\delta w_i}; \\ b &= b - \alpha \frac{\delta \mathcal{J}}{\delta b}. \end{aligned} \tag{3.7}$$

The values of the parameters are subtracted (and not added) by the partial derivatives because the derivatives indicate the slope (i.e. the growth) of the cost. In particular, if a derivative is positive at a certain point (values of the parameters) then the cost grows at this point and since the objective is to minimize the cost, we want to move the parameter in the opposite direction.

Gradient descent proceeds by iterations. At each iteration, called **epoch**, the partial derivatives are computed with the current values of the parameters which are in turn updated. This continues until a certain number of epochs is reached or until convergence (when the parameters barely change after an update).

Hyperparameter tuning

We have introduced the learning rate α without really explaining what it is. The learning rate in the optimization algorithm is in fact a **hyperparameter**, which is a property of the algorithm. That is, a hyperparameter is not a value that is learned from data. Hyperparameters have to be set before running the training phase, and will in some way influence how the model performs. For example, setting the learning rate α to a very large value will probably make the optimization algorithm diverge. Indeed, with a high learning rate, the algorithm performs big steps at each iteration, hence reducing its precision and the global minimum of the cost function could be difficult to reach. Setting the learning rate to a very small value will make the system more precise, but very slow.

How to then find good values for hyperparameters ? This question is answered by **hyperparameter tuning**. The idea is to split the original dataset into two subsets : the **training set** and the **validation set** (sometimes also called development set). The learnable parameters (weights) are learned and tuned on the training set only. The validation set is used to tune the hyperparameters. This is achieved by creating multiple models with different values for their hyperparameters, train all the models on the training set, then assert which model and hence which values of hyperparameters perform best on the validation set.

3.2 Neural networks

3.2.1 Multilayer Perceptron

A neural network, just like logistic regression, is a function that takes an input x and produces some output $\hat{y} = f_{NN}(x)$. The difference is that instead of computing only one weighted sum of the inputs $wx + b$, as logistic regression does, neural networks are composed of many computational units performing this operation. These computational units are called **neurons** and are divided into **layers**. The first layer is called the **input layer**, the last layer is the **output layer** and the layers in-between are the **hidden layers**.

Figure 3.3 is an illustration of one particular type of neural networks called the multilayer perceptron which are a kind of **feedforward neural networks**. We call these models feedforward because the information flows through the network from the input vector x, through the intermediate computations in the hidden layers and finally to the output $\hat{y}$. That means that there are no **feedback** connections in which the output of some unit is fed back to a prior unit in the network (as opposed to recurrent networks which we introduce later).

This arrangement into layers means that instead of simply taking the feature vector x to which some weights are applied, each neuron in layer l rather uses the output of the neurons in the previous layer $l - 1$. For instance, to obtain the value of neuron h_1 in Figure 3.3 we compute :

$$h_1 = g(u_{1,1}x_1 + u_{1,2}x_2 + u_{1,3}x_3 + b). \tag{3.8}$$

This enables us to compute the output y of the network :

$$\hat{y} = g(v_{1,1}h_1 + v_{1,2}h_2 + b). \tag{3.9}$$

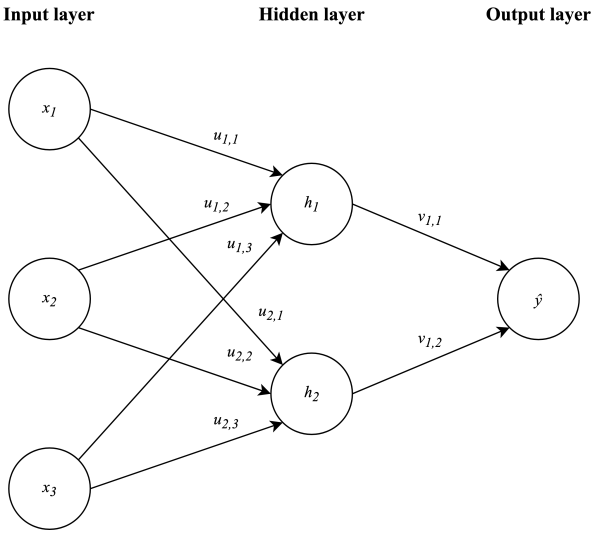


Figure 3.3: An example of a multilayer percepton with three layers : the input layer which values consists of the feature vector composed of three neurons, one hidden layer with two neurons and the output layer with one neuron.

More generally, we denote the output of neuron u in layer l by a_u^l, its associated bias by b_u^l and its weight vector by w_u^l. The output of all neurons in a layer l can then be assembled into a vector z^l :

$$z^l = [a_1^l, \ldots, a_k^l].$$

The neuron u in layer l thus computes :

$$a_u^l = g^l(w_u^l z^{l-1} + b_u^l). \tag{3.10}$$

The function $g^l(.)$ is called the **activation function** for layer l. Without activation functions, the function f_{NN} of a neural network would essentially consist of compositions of linear functions $wz + b$. The problem is that a composition of linear functions is also a linear function. Non-linear activation functions in neural networks are hence crucial to introduce non-linearity such that f_{NN} can approximate (learn) any function, not just linear ones. Some typical options for activation functions are the rectified linear unit (ReLU), the hyperbolic tangent (tanh) and the sigmoid function.

In a typical classification task, the cost function that we try to minimize when training a neural network is the same as the one we introduced for logistic regression, i.e. the average (3.4) of the cross-entropy losses (3.5) on the whole training set. Minimization of the cost function $\mathcal{J}$ is also achieved via gradient descent. Hence, at each iteration, the partial derivatives for all the learnable parameters of the network with respect to the cost function need to be computed :

$$\nabla\mathcal{J} = \Big[\frac{\delta\mathcal{J}}{\delta w_{1,1}^1}, \frac{\delta\mathcal{J}}{\delta w_{1,2}^1}, \ldots, \frac{\delta\mathcal{J}}{\delta w_{k,n}^l}, \frac{\delta\mathcal{J}}{\delta b_1^1}, \ldots, \frac{\delta\mathcal{J}}{\delta b_k^l}\Big]. \tag{3.11}$$

The **back-propagation** algorithm is used to compute these partial derivatives in neural networks. This algorithm starts by first computing the partial derivatives at

the output layer and then proceeds backwards to find, in turn, the partial derivatives of the weights at the prior layers. Intuitively, back-propagation allows the information of the cost to flow backward through the network, hence the name.

3.2.2 Convolutional neural networks

A **convolutional neural network** (CNN) is a feedforward neural network that contains one or more **convolutional layer(s)**. A typical convolutional layer consists of three stages. In the first one, the layer performs **convolutions**. This is followed by a non-linear activation function. The third stage uses a **pooling function** which further modifies the output of the layer (Figure 3.4).

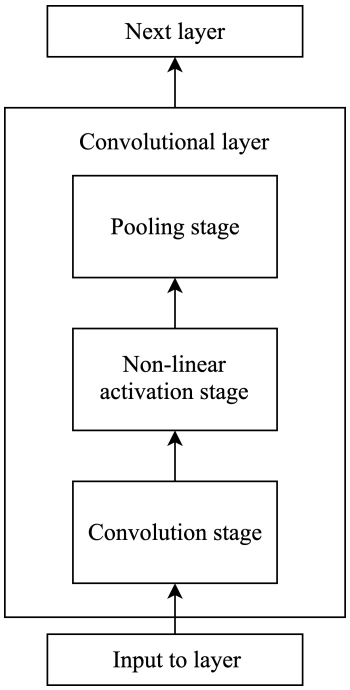


Figure 3.4: A typical convolutional layer is composed of a convolution stage, a non-linear activation function and a pooling stage [24].

During the convolution stage, a window is *overlayed* on the input and moved left-to-right on that input (it is also moved top-to-bottom when the input is two-dimensional). This moving window is called the **filter** or **kernel**. The **size** of the filter is arbitrary, and is a hyperparameter of the model. The values of the filter are however learnable parameters tuned during training. For each overlay, we perform an element-wise product of the overlayed input and the filter values, we then sum the products and hereby obtain the result of the current overlay which we store in an output matrix (or vector if the input is one-dimensional). We then move the window to the right, according to a certain step size given by the **stride** of the convolution (which is also a hyperparameter), and repeat the whole process until the entire input has been overlayed by the filter. Figure 3.5 illustrates the convolution operation and how the outputs are computed.

One convolutional layer can have multiple filters, just like one layer in a multilayer perceptron can have multiple units (neurons). When that is the case, one output matrix is produced for each one of the filters. This collection of matrices is called a **volume**. Suppose now we stack two convolutional layers and the first one has multiple filters. Hence, the output of the first convolutional layer is a volume. How does the convolution operation then operates in the second convolutional layer ? When one filter is overlayed on a region of the first matrix of the input volume and the weighted sum is computed, the same region for all the other matrices in the volume is overlayed by the filter. Then, all the obtained values are simply summed together. Thus, even if the input is a volume, the application of one overlay of a given filter still produces a scalar.

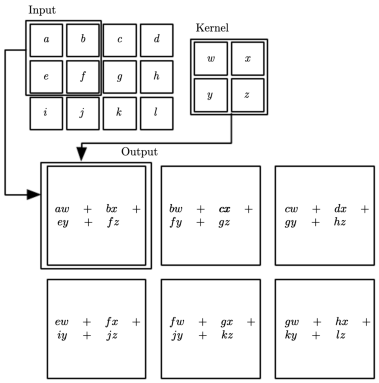


Figure 3.5: Convolution with a kernel of size 2×2 and a stride of one [24].

The primary function of pooling is to reduce the size of the representation in order to reduce the amounts of parameters and computations in the network. Pooling works in a similar way to convolution. A moving window is again overlayed on the input and either the maximum or the average of the input in the window is retrieved (Figure 3.6). Note that the input here is the output of the convolutions as opposed to the output of the previous layer as it was the case for the convolutions. Moreover, the pooling layer has no trainable parameter has it only applies a fixed operation, but it however has the same hyperparameters as the convolution layer (size and stride).

3.2.3 Recurrent neural networks

Recurrent neural networks (RNNs) are a family of neural networks specialized in dealing with sequential data. That is, they are networks that are specifically designed to process a sequence of values $x^1 \dots x^t \dots x^\tau$.

Recurrent networks are not feedforward, they contain loops (recurrent connections, Figure 3.7a). Each unit of a recurrent layer keeps a real-valued **state**, which can be interpreted as the memory of the unit. In RNNs, each unit in a layer does not only receive input from the previous layer at time t, but also the vector of states of this same layer from the **previous time step** $t-1$. By unrolling the network (Figure 3.7b), it is easier to visualize how, despite *advancing* in the input sequence, the information computed at the previous time steps persists.

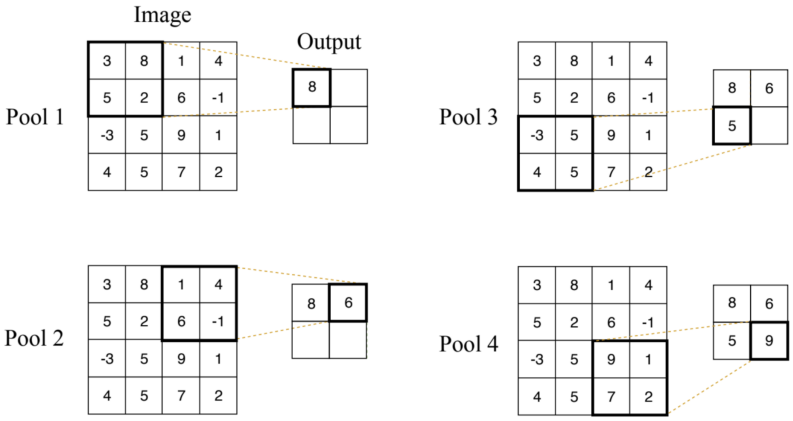
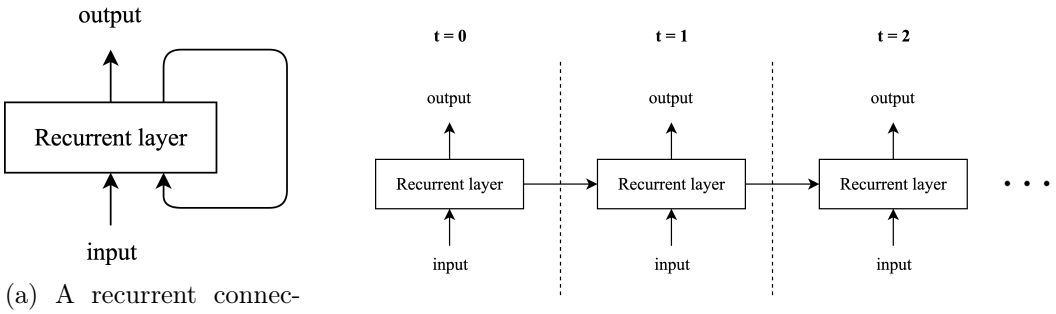


Figure 3.6: Max-Pooling with a 2×2 window size and a stride of two [13].



(a) A recurrent connection.

(b) An unrolled RNN.

Figure 3.7: Recurrent neural networks.

In practice, long-term dependencies in the sequence are difficult to handle and learn for the standard recurrent architecture because the state of each unit is more affected by data seen *recently* (this is because, at each time step t, the hidden state is always updated with the input at this time t). Hence, input vectors observed much earlier are *forgotten* over time [13].

LSTMs

Long Short-Term Memory (LSTM) networks are capable of learning long-term dependencies. Not only do LSTMs keep a hidden state internally (and use it as output), but they also store a memory cell c_t. A LSTM unit (module) makes decisions about what should be stored in this memory cell and when to allow reading, writing and erasing this cell. Said decisions are learned from data and take the form of **gates** (Figure 3.8) :

- the **forget gate** handles erasure of the memory cell by multiplying it by values in $[0, 1]$. Multiplying a memory cell value by 0 will obviously completely erase this value;
- the **input gate** is used alongside the candidate memory (output of the tanh layer) to decide how much of the current input needs to be stored in the memory

cell. Hence, the role of the input gate is to update the memory cell with new information;

- the **output gate** chooses how much of the memory cell needs to be retrieved to be used alongside the hidden state in the output.

Note that these gates are made of fully connected layers. Hence learning adequate weights for these layers will enable the module to decide when it is best to erase, update and read its memory cell.

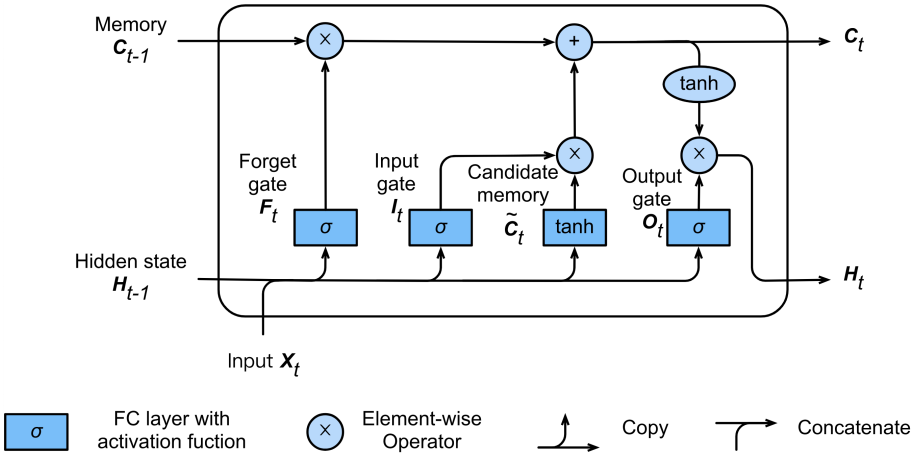


Figure 3.8: A LSTM module [2].

3.3 Assessing performance of the model

Once we obtain a model trained by our learning algorithm, one question yet remains: how good is our model ? Since the learnable parameters of the model were fit to the training set and its hyperparameters were chosen depending on the model performance on the validation set, we cannot truly assert its capabilities on these sets. That is why instead of simply dividing our original dataset into a training set and a validation set, we further partition it into a third set, called the **test set**. This test set is thus composed of data points, for which we know the true label since they are part of our dataset, but that the model has never seen before. Assessing the performance of the model is thus performed on the test set to indicate how well the model **generalizes** to new, unseen data. This will hence give us some hint as to how well the model would perform in practice.

Many different metrics asserting performance of the model on the test set exist. Among these metrics the most popular one is the **accuracy** which is the percentage of examples correctly classified by our model. Accuracy can also be expressed in terms of **true positives**, **true negatives**, **false positives**, and **false negatives**. To explain these notions we will use the example of a model that tries to predict if a given patient is ill or not. Then, when our model is given some patient information as input, the prediction made by the model can be categorized :

- if the patient is truly ill and the model classified the patient as being ill, the prediction is a **true positive** (TP);

- if the patient is not ill and the model classified the patient as not being ill, the prediction is a **true negative** (TN);
- if the patient is truly ill and the model classified the patient as not being ill, the prediction is a **false negative** (FN);
- if the patient is not ill and the model classified the patient as being ill, the prediction is a **false positive** (FP).

We can hence rewrite the accuracy metric in terms of these introduced notions :

$$\text{accuracy} = \frac{\#\text{TP} + \#\text{TN}}{\#\text{TP} + \#\text{TN} + \#\text{FP} + \#\text{FN}}.$$

Suppose now our test set is extremely imbalanced : 99% of the patients are labeled as not being ill whereas only 1% are labeled as being ill. If we create a trivial model that always predicts a patient as not being ill, then the accuracy of our model on the test set will probably be close to 99% ! When the dataset is imbalanced in terms of class representation, it is said that we have **skewed classes**. As we intuitively described, asserting performance of a model with the accuracy metric when the problem at hand has skewed classes do not effectively evaluate the model.

Fortunately, different metrics can be used to counter this problem : **sensitivity** (or recall), **specificity** and **precision** :

$$\begin{aligned}
\text{sensitivity} &= \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}}; \\
\text{specificity} &= \frac{\#\text{TN}}{\#\text{TN} + \#\text{FP}}; \\
\text{precision} &= \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}}.
\end{aligned}$$

In our example, sensitivity can be expressed as the percentage of truly ill patients that were effectively classified as being ill. The trivial model that always predicts a patient as not being ill would therefore have a sensitivity of 0%.

Chapter 4

Features for forecasting the onset of AF

4.1 Deriving features from Coumel's Triangle

As we discussed in Section 2.3.3, Coumel's Triangle [18] gives some insights regarding the physiological processes responsible for AF initiation.

Given that we have at our disposal the RR intervals preceding AF events, how can we then **alleviate this physiological knowledge to derive corresponding discriminating features** from said RR intervals ?

In this section, we introduce features that can be obtained from RR intervals signals and that are directly related to the following Coumel's Triangle factors (although with some restrictions, which we will discuss) :

1. the trigger factor and premature atrial complexes (PACs);
2. the role of the ANS modulation as facilitator.

4.1.1 Premature atrial contractions

Ectopic beats are irregular premature heart beats consisting of either a premature contraction of the atrium (PAC) or a premature contraction of the ventricle (PVC). Since approximately 93% of AF events are triggered by the former [31], PACs are the type of ectopic beat that interests us. However, as we only have access to the RR intervals in this work, it is very difficult to distinct a PAC from a PVC (we would need to access the raw ECG to analyse the morphology of the QRS complex). That is why we will make no distinction between PACs and PVCs and rather work with ectopic beats more generally.

Hence, the question becomes: how can we identify ectopic beats in RR intervals ? There exists actually a few recommended different methods to achieve this, but we will focus on the approach suggested by Malik et al. [34]. According to Malik et al. two successive RR intervals should not differ by more than 20%. Hence, we can count ectopic beats in a RR intervals episode by simply labelling any RR interval that does not respect the following rule as an ectopic beat:

$$\texttt{is-ectopic}(RR_{i+1}) \equiv |RR_i - RR_{i+1}| > 0.2 \times RR_i \tag{4.1}$$

NN intervals

When ectopic beats get removed from a RR interval signal, we obtain **Normal-to-Normal (NN) intervals** which correspond to intervals between adjacent QRS complexes resulting from the sinus node depolarization only [38]. As ectopic beats are often seen as artifacts, NN intervals are sometimes preferred over RR intervals to ensure valid data.

4.1.2 The ANS and frequency-domain features

The ANS is composed of two branches : the sympathetic nervous system and the parasympathetic (or vagal) nervous system. The former is often considered the *flight or fight* system while the latter is more associated with the *rest and digest* system.

One can evaluate the activity of these autonomic nervous systems by computing the Power Spectral Density (PSD) of the RR signal. PSD analysis informs on how power distributes as a function of frequency and its calculation is generally achieved via the Fast Fourier Transform (FFT) algorithm [38].

As FFT requires an evenly sampled and detrended signal, it is necessary to choose:

- a resampling frequency (often 7Hz);
- the interpolation method (e.g. linear, cubic);
- the detrending method (e.g. mean RR);

before applying FFT and extracting the PSD. Once the PSD is obtained, we can distinguish three main spectral components in specific frequency ranges (Figure 4.1) :

- $0 - 0.04$Hz $\rightarrow$ **Very Low Frequency (VLF)**;
- 0.04Hz $- 0.15$Hz $\rightarrow$ **Low Frequency (LF)**;
- 0.15Hz $- 0.4$Hz $\rightarrow$ **High Frequency (HF)**.

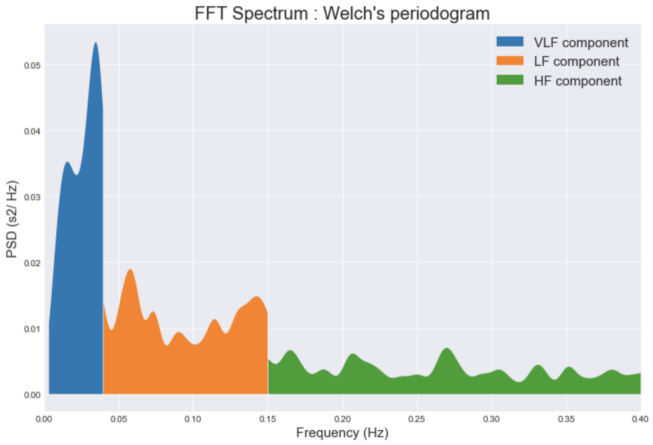


Figure 4.1: Example PSD with its spectral components.

Vagal activity is the major contributor to the HF component but studies disagree regarding the LF metric. Some argue that LF only reflects sympathetic modulation whereas others view LF as a marker of both vagal and sympathetic activity [38].

Two other metrics are also often derived from the values of VLF, HF and LF :

- the **LF/HF** ratio to characterize sympatho-vagal balance;
- the total power (power in all frequency bins minus VLF) of the PSD (**TOTAL-POWER**).

4.2 Other standard features

Many other features can help in forecasting AF. We hereby list the most commonly used in the literature.

4.2.1 Time-domain features

From a series of RR or NN intervals, statistical time-domain measures can be calculated:

- **MeanRR**, mean value of RR intervals;

$$\text{MeanRR} = \frac{1}{N}\sum_{i=0}^{N} RR_i \tag{4.2}$$

- **SDRR**, standard deviation of RR intervals;

$$\text{SDRR} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(RR_i - MeanRR)^2} \tag{4.3}$$

- **RMSSD**, square root of the mean of the sum of the squares of successive differences;

$$\text{RMSSD} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(RR_{i+1} - RR_i)^2} \tag{4.4}$$

- **pRR50**, percentage of adjacent RR intervals differing by more than 50ms.

$$\frac{\#[RR_{i+1} - RR_i > 50ms]}{N} \tag{4.5}$$

4.2.2 Bispectral features

The power spectrum provides the signal's power as a function of its frequency components but fails to give information about the phase relations between the frequencies. This can, however, be obtained with Higher Order Spectra (HOS) and the bispectrum [42].

The bispectrum $B(f_1, f_2)$ is the 2D Fourier transform of the third order cumulant and is, in most of the literature, calculated via `Matlab`'s Higher Order Spectral Analysis Toolbox [44].

Figure 4.2 shows an example bispectrum as well as its region of interest (ROI). The ROI is divided into 3 subband regions: LF-LF (LL), LF-HF (LH), and HF-HF (HH) each covering a specific range of frequencies.

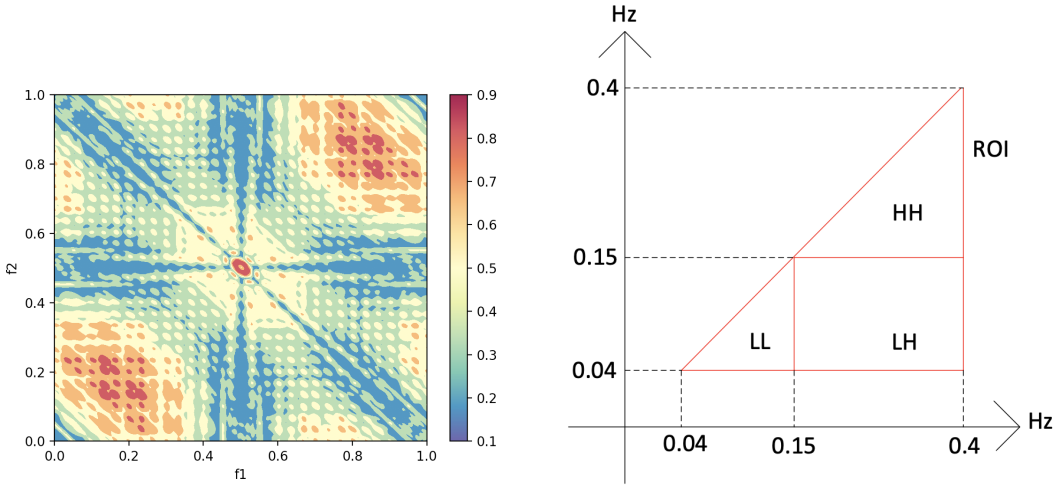


Figure 4.2: Example of bispectrum (left) and subband regions (right).

Many features can be extracted from each subband region and ROI and we use formulas in Yu et al. [48] and Zhou et al. [49] to compute them. With Ω representing one of the subband regions (and total ROI) in Figure 4.2 and N is the number of points within Ω we derive the following features:

- $\mathbf{M}_{avg}$, the average magnitude;

$$\mathrm{M}_{avg} = \frac{1}{N}\sum_{\Omega}|B(f_1, f_2)| \tag{4.6}$$

- $\mathbf{P}_{avg}$, the average power;

$$\mathrm{P}_{avg} = \frac{1}{N}\sum_{\Omega}|B(f_1, f_2)|^2 \tag{4.7}$$

- $\mathbf{E}_1$, the normalized bispectrum entropy;

$$\mathrm{E}_1 = -\sum_{m} E_i \log E_i \tag{4.8}$$

where

$$E_i = \frac{|B(f_1, f_2)|}{\sum_{\Omega}|B(f_1, f_2)|}$$

- $\mathbf{E}_2$, the normalized bispectrum square entropy;

$$\mathrm{E}_2 = -\sum_{m} E_m \log E_m \tag{4.9}$$

where

$$E_m = \frac{|B(f_1, f_2)|^2}{\sum_{\Omega}|B(f_1, f_2)|^2}$$

- $\mathbf{H}_1$, the sum of the logarithmic magnitudes;

$$\mathrm{H}_1 = \sum_{\Omega} \log|B(f_1, f_2)| \tag{4.10}$$

- $\mathbf{H}_2$, the sum of the logarithmic magnitudes of the diagonal elements;

$$\mathrm{H}_2 = \sum_{D} \log|B(f_k, f_k)| \tag{4.11}$$

  with D the diagonal elements of a region Ω.

- $\mathbf{H}_3$, the first-order spectral moment of the magnitudes of diagonal elements;

$$\mathrm{H}_3 = \sum_{k=1}^{D} k \log|B(f_k, f_k)| \tag{4.12}$$

- $\mathbf{H}_4$, the second-order spectral moment of the magnitudes of diagonal elements;

$$\mathrm{H}_4 = \sum_{k=1}^{D} (k - \mathrm{H}_3)^2 \log|B(f_k, f_k)| \tag{4.13}$$

4.2.3 Non-linear features

By plotting each RR interval against the next RR interval, one can obtain the Poincaré plot (Figure 4.3). From this plot we can extract the following features:

- **SD1**, standard deviation of the distances of RR intervals from line-of-identity ($y = x$);
- **SD2**, standard deviation of the distances of RR intervals from $y = -x + 2MeanRR$
- **SD1/SD2**, ratio of SD1 and SD2.

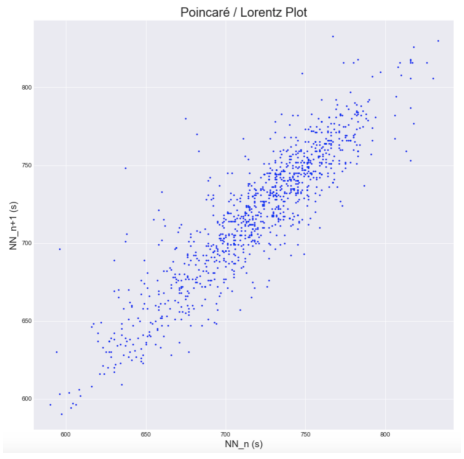


Figure 4.3: Example Poincaré plot.

Finally, some studies use sample entropy (**SamEn**) to characterize the regularity of time series data; which can hence be applied to NN intervals. Details about its calculation can be accessed at [36].

Chapter 5

State of the art

Before introducing state of the art methods and results for the forecasting of atrial fibrillation in ECG signals, let us describe precisely what is meant by AF **forecast** by contrasting it with AF **detection**.

5.1 Detection vs. forecasting

The **detection** of atrial fibrillation can be formulated as the following question :

> Given an ECG episode, is this ECG **currently** in atrial fibrillation ?

Atrial fibrillation forecast, on the other hand, can be formulated as :

> Given an ECG episode in **Normal Sinus Rhythm** (NSR), will the patient (from which this ECG was sampled) **enter atrial fibrillation in the foreseeable future** ?

AF detection is thus concerned with flagging an AF episode while it is happening (*a posteriori*) while AF forecast consists of detecting the onset of AF in ECGs preceding the AF episodes (*a priori*).

5.2 Atrial fibrillation detection

Detecting whether or not a given ECG signal is in AF is an extensively studied problem in the literature. One could go back to 1991 to find the first application of neural networks for the atrial fibrillation detection task. Indeed, Artis et al. [8] used a simple multilayer perceptron to obtain beat-by-beat classification of the ECG signal between AF and not AF. By using some clever preprocessing and postproceessing steps, the model already performed very well with a sensitivity of 92.86%.

5.2.1 Binary classification with RR intervals

Recent works tend to use deeper architecture to solve this problem. For example, in 2018, Faust et al. [22] fed series of 100 RR intervals to a model made of two LSTMs layers followed by two fully-connected layers and achieved an overall accuracy of 99.77%. The same year, Andersen et al. [7] proposed an architecture composed of two convolutional layers followed by a single LSTM layer. RR intervals were again

used as inputs to the network. This model achieved a sensitivity of 98.17% and a specificity of 96.29%.

In his master thesis [23], C. Gilon also performed binary classification (AF/not AF) of 300 RR intervals windows. Two different models were analyzed : the first one was composed of a bidirectional gated recurrent unit (BiGRU, a variation of LSTM) layer and two FCs layers; the second one inserted two convolutional layers prior to the BiGRU layer and dropped a FC layer. Both models performed well and achieved a sensitivity and specificity of 97.90%, 99.58% and 94.91%, 99.1% respectively.

As these results show, **binary *a posteriori* classification** of atrial fibrillation using deep neural networks with **RR intervals** has been a **great success**.

5.2.2 End-to-end multi-class arrhythmias detection

However, the methods presented in the previous section limit their problem to binary classification (AF/not AF). Screening different kinds of arrhythmias is thus not possible with these models. Moreover, by only analyzing RR intervals, informations present in the raw ECG signal but absent in said RR intervals are inherently lost and hence cannot be used by the model.

Given this loss of information, recent works have instead used an **end-to-end** deep learning setting. In such a setting, the feature extraction process from the raw ECG signal is completely performed by the deep model (as opposed to being done manually) and the raw ECG signal is directly used as the input feature vector of the network.

In this context, Acharya et al. [6] designed a CNN consisting of 4 convolutional layers (convolution operation and max pooling) followed by 2 fully-connected layers to categorize a raw ECG segment into four different classes : normal sinus rhythm, atrial fibrillation and two other arrhythmias types (atrial flutter and ventricular fibrillation). Denoising was applied to the ECG segments before feeding them to the model but no hand-crafted features were retrieved from the ECGs; the convolutional layers used the denoised ECGs to extract features by themselves. The sensitivity and specificity on ECG segments of two seconds and five seconds was the following: 98.09%, 93.13% in the former case and 99.13%, 81.44% in the latter case.

In the same year, Hannun et al. [27], used a 34 layers CNN taking as input a 30 seconds-long raw ECG signal. The network is designed to classify 12 different arrhythmia types. Because of the very deep nature of the network, the architecture of the CNN model is composed of skip connections, dropout layers and batch normalization. Moreover, to make the training of such a network possible, a big private dataset of ECG signals (64.000 ECG records from 29.000 patients) was assembled. Unfortunately this dataset was not made publicaly available. The model achieved an overall average sensitivity of 82.7%. It is shown that the model outperforms the average cardiologist performance.

These results clearly indicate that an **end-to-end deep learning setting** is an approach that, given **enough available data**, can pave the way towards **complete multi-arrhythmia automated screening**.

Group	Desc.	# pairs in learning set	# pairs in testing set
A	PAF patient	25	28
N	Normal patient	25	22

Table 5.1: Data repartition of the AFPDB.

5.3 Atrial fibrillation forecasting

Although AF screening has been the subject of many publications in the last decade, the literature considering the problem of AF forecasting is much narrower. Let us thus make an exhaustive review of the previous works tackling AF forecasting. We order the presented articles by publication date. Table 5.3 summarizes the known published results.

5.3.1 The Physionet challenge

In 2001, Physionet [3] and Computers in Cardiology (CinC) [1] offered a challenge to develop a fully automated method to predict the onset of paroxysmal atrial fibrillation based on the ECG prior to the event [5]. The dataset, the PAF Prediction Challenge Database (AFPDB) [4] consists of 100 pairs of 30 minutes ECG data. Each pair is derived from a 24h ECG recording from a single subject. Subjects are divided in two groups (see Figure 5.1):

1. Group **A** consists of subjects experiencing PAF. For these subjects, one ECG episode in the pair ends just prior to the onset of PAF, while the other ECG episode is distant to any PAF episode (at least 45 mins).

2. Group **N** does not experience PAF. Hence, for these subjects, both the ECG episodes in the pair were chosen randomly in the 24h recording period.

Given such pairs of ECGs, the challenge was defined in two parts:

1. distinguish subjects in group **A** from subjects in group **N**;

2. if the pair is in group **A**, which ECG episode among the two is **the one that ends just prior to the onset of PAF** ?

To implement their strategy and develop a model, a learning set composed of 25 pairs in group **A** and 25 pairs in group **N** was made available to the participants during the challenge. The performance of the participants' submissions was then assessed on a **separate test dataset**, composed of 28 pairs in group **A** and 22 pairs in group **N**. See table 5.1 for a summary of the data repartition among the learning and test sets in the AFPDB.

Given that we are interested in PAF forecasting, it is clear that we will only **focus on the second part of the challenge**, i.e. given ECGs of a patient that is at risk of PAF, determine when an ECG is just prior the onset of PAF.

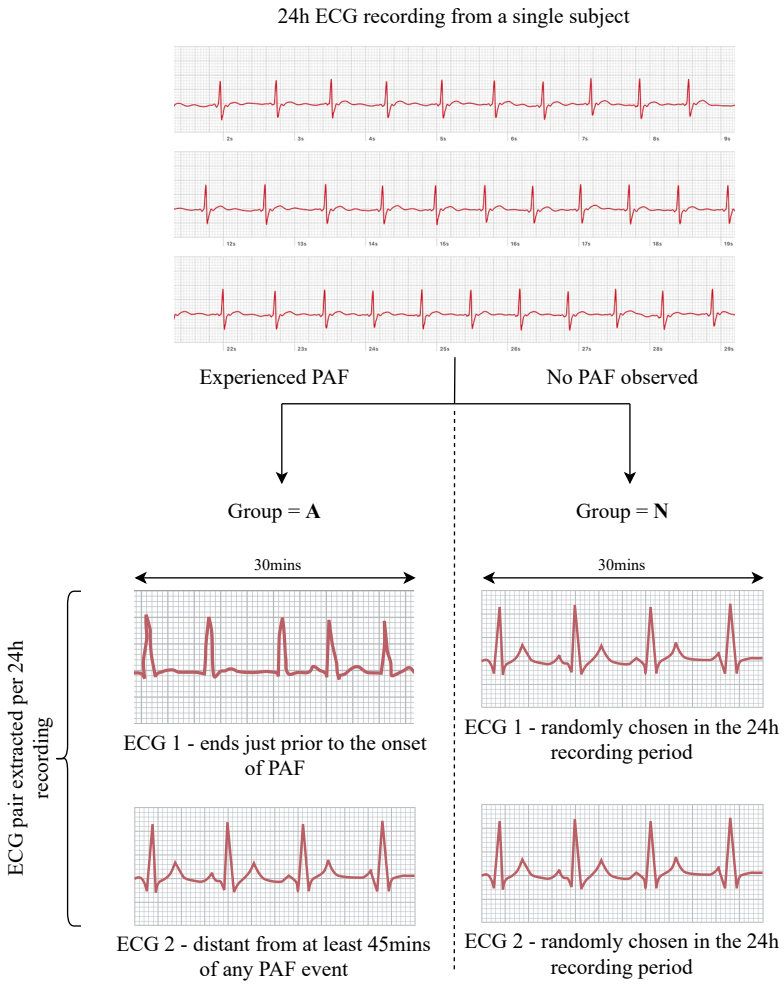


Figure 5.1: Specification of the ECG pairs for the AFPDB.

5.3.2 High performing submissions of the Physionet challenge

Zong et al.

The winning submission, given by Zong et al. [50], used the premature atrial complex (PAC) count. They identified PACs as any beat that is at least 15% premature (in respect of a moving average of RR intervals).

In table 5.3, the submission of their model is marked as Expert System (ES). In reality, however, their algorithm was extremely simple and made full use of the fact that the ECGs were coming in **pairs**. They also knew that, out of the two episodes in the pair, **exactly one is prior to the PAF event**. Hence, in such a setting, participants could make **comparisons between the episodes in the pair** to **help with inference**.

Indeed, in the strategy adopted by Zong et al., they simply **counted the number of PACs** in each ECG and marked the episode with the **highest count** as the one **prior to the PAF event**. Despite being so simple, this algorithm was the submission with the highest accuracy of 79% (22/28).

Chazal and Heneghan

The submission of Chazal and Heneghan [15] is also interesting to take a look at. It is one of the few studies that reported their results on shorter ECG episodes, rather than the full 30mins segments. Moreover, they also compared results with different features groups. Note that for every approach, they used linear discriminant analysis (LDA) as the classifier model.

The best result, an accuracy of 68% (19/28), was obtained with the PSD of the RR intervals sequences of the last 10mins. Instead of deriving LF and HF estimates from the PSD, they combined adjacent frequency bins to result in a 16-point PSD estimate and used the magnitude of theses bins as features. Note that the mean and standard deviation of the 10min RR sequences were also included. Although the result on the test set was not communicated, the 30mins version of this feature set performed worse than the 10mins version on their validation set.

Two other feature sets were derived from the P-wave. The first one directly used the amplitudes of the P-wave area whereas the second feature set was formed by computing the PSD of the P-wave area. These feature sets were best performing on 5mins windows but showed very poor results, with accuracies around 39% (11/28) only.

5.3.3 Early post-challenge publications

After the challenge, the complete dataset (AFPDB) that was assembled was made available at [4]. It has since then been adopted in the literature as a **common evaluation standard** to assert and compare methods trying to forecast the onset of atrial fibrillation.

Hickey and Heneghan

The first paper that appeared after the Physionet challenge took place was the one written by Hickey and Heneghan [28] in 2002. This study was the first to not consider the ECGs by pairs and classified each segment independently. For each pair of segments in the AFPDB, they removed the distant record from the dataset. They then constructed their learning data (resp. test) as follows :

1. 25 (28) pre-AF episodes from PAF patients;
2. 50 (44) normal episodes from normal patients.

Since the model has to discriminate segments belonging to PAF patients from segments belonging to normal patients, it is clear that this study **concerns PAF screening much more than PAF forecasting**. However, as this study is unfortunately often referenced by other studies on PAF forecasting ([37, 36, 12, 11, 20]), we had to mention said study, despite it being wrongly categorized as a PAF forecasting publication.

Thong et al.

Thong et al. [45] improved the work published by Zong et al. [50] during the challenge by analyzing isolated PACs not followed by regular RR intervals. Indeed, instead of only considering the PAC in itself, they further investigated what was

happening after the detected PAC. They constructed an ES with a set of rules derived from analyzing those PACs and obtained an accuracy of 89% (25/28).

5.3.4 Classical ML approaches with RR intervals

Most of the more recent work on AF forecast focuses on extracting features from RR intervals and feeding these features to a classical ML model after some feature selection. Figure 5.2 depicts the typical approach of the more recent studies :

1. RR intervals are first extracted from the raw ECG signal by using the Hamilton and Tompkins algorithm [40];
2. PACs are removed from the RR signal to create NN intervals which are then used to infer time-domain and non-linear features;
3. in order to perform spectral and bispectral analysis on the signal, NN intervals are resampled (usually around 7hz), interpolated and the signal is detrended;
4. feature selection is applied, typically with filter methods;
5. the selected features are fed into a classical ML classifier (e.g. SVM).

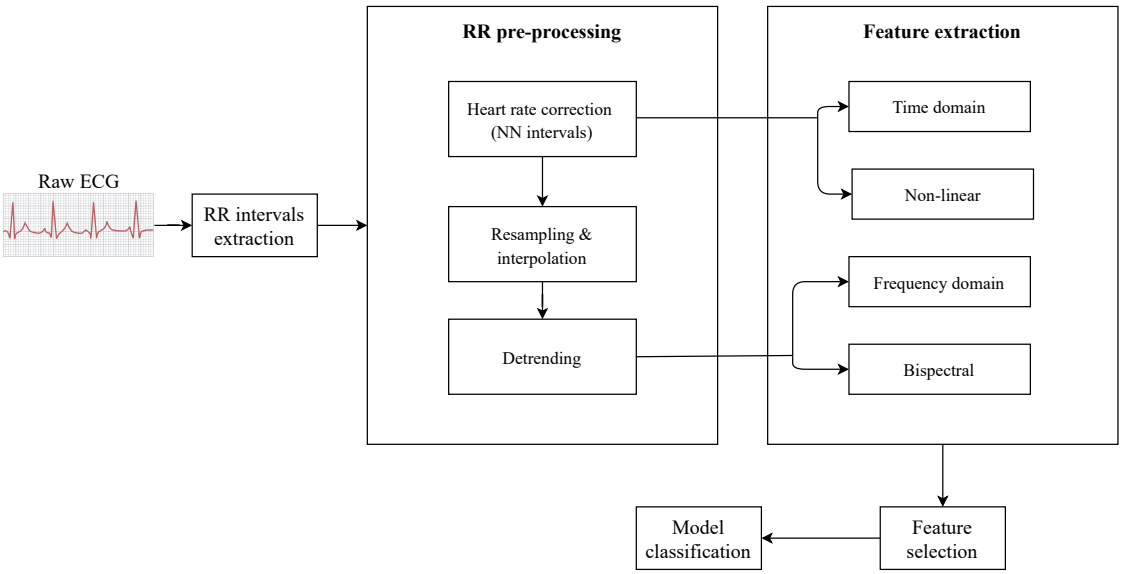


Figure 5.2: Typical setup for the AF forecasting problem.

Chesnokov

In 2008, Chesnokov [16] attempted to classify ECGs in group **A** as distant or prior to the AF event. He used the entire 30mins segments and extracted VLF, LF, HF and total power from the PSD as well as the signal's sample and approximate entropies. The trained SVM obtained an accuracy score of 80% and the simple 3-layer MLP achieved an accuracy of 82% (both on the separate test set).

On top of that, Chesnokov also highlights in his study the statistical differences by means of error bars (mean $\pm$ std) between the spectral features (as well as the sample entropy) in episodes distant and prior the AF event. Figure 5.3 reports his findings which clearly indicate the **discriminating power** of these features.

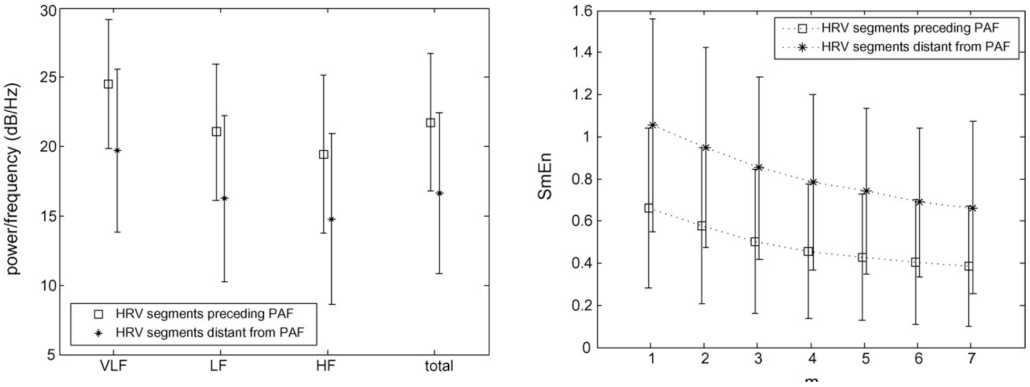


Figure 5.3: Chesnokov's [16] error bars analysis of VLF, LF, HF total power(left) and sample entropy (right) of the 30mins records distant from PAF and immediately prior the PAF onset.

Feature	***p*-value**
Bispectral	≤ 0.0001
Non-linear	≤ 0.001
LF and HF power	≤ 0.001
LF frequency peak	0.13
HF frequency peak	0.2

Table 5.2: Mohebbi and al. [36] ANOVA test results.

Mohebbi et al.

Results obtained in 2012 by Mohebbi et al. [36] on the Physionet AFPDB are, to this day, still state of the art. No other publication was able to reach the same level of performance. It is the first study to compute **bispectral features from the RR signal**. They combine these bispectral features with standard spectral and non-linear features. In total, 12 features are extracted from the 30mins RR intervals. They train and validate a SVM model on the AFPDB learning set. The performance of the SVM on the withheld AFPDB test set is then reported: they obtained 96% sensitivity and 93% specificity.

The improvement in the model performance in Mohebbi et al. [36] compared to Chesnokov [16] suggests an overwhelming added value of the bispectral features extracted from RR intervals (the only difference between the two approaches is the use of the bispectrum by Mohebbi and al.). This was confirmed in the study by running analysis of variance (ANOVA) tests on the learning set, as the lowest p-value was obtained by the bispectrum-based features (see Table 5.2).

Boon et al.

The 2016 paper of Boon et al. [12] is the paper that uses the most features. In total, 55 features are extracted using time domain, frequency domain and non-linear analysis. Genetic Algorithm (GA) is applied as wrapper feature selection method. The 10-fold cross-validated accuracy metric of a SVM is used in the fitness function of the GA. Moreover, the authors motivate the need to repeat the optimization process 10 times, in order to reduce the probability of getting results that converge to local optima of the fitness function. Surprisingly, the optimized cross-validated accuracy is also used for final model evaluation... The authors did not keep a separate test set to assess model performance. Nevertheless, they obtained accuracies of 80%, 79% and 70% for segments of 30, 15 and 10mins respectively.

In 2018, Boon et al. extended their 2016 work by (1) using another Genetic Algorithm and (2) instead of optimizing only the SVM hyperparameters and the feature selection with GA, they also optimized the RR feature extraction process. This means that, for example, the GA optimization decided to enable or not heart rate correction when extracting time domain features. As it was the case for their 2016 work, the reported model evaluation are the GA-optimized cross-validated metrics (86.8% sensitivity and 88.7% specificity).

Narin et al.

Using a GA for feature selection was also the strategy adopted by Narin et al. in 2018 [37]. They restricted their feature set to time domain and frequency domain features and used a k-Nearest Neighbours (kNN) classifier. The performance metrics optimized by GA are the 10-fold cross-validated average numbers of TN and FP. They tune the number of neighbours k by repeating the experience for values of $k \in [1, 3, 5 \dots 19]$. Final model evaluation is reported on the same 10-fold cross-validation.

They obtain an accuracy of 90% for a subset of one time-domain and three frequency-domain features ($k = 3$). Surprisingly, they also report an accuracy of 84% when selecting only two time-domain features with $k = 1$.

5.3.5 Deep learning and AF forecasting

Gilon's masters thesis

In his 2019 masters thesis [23], Gilon worked with a dataset of ECGs assembled (and annotated) by Dr. Jean-Marie Grégoire. This dataset is composed of 140 ECG records presenting PAF. In total, 301 different PAF episodes are observed among these records. It is the first work to report results on a different dataset than the AFPDB.

To the best of our knowledge, it is also the first study to apply DL models on the AF forecast task. No features were derived from the RR intervals and Gilon let the DNN learn relevant features by itself. Hence, the raw RR signal was fed as input vector to the network.

Gilon's best performing network is composed of two 1-D CNN layers followed by a bidirectional GRU layer and a final FC layer. The network performances for different window sizes of RR intervals that precede AF onset were reported (Figure

5.4). Windows of size 60, 300 and 900 were tested (corresponding to segments of resp. approx. 1min, 5mins and 15mins) and it is observed that the 300RR window is the most promising (0.72 AUC, 68% sensitivity and specificity).

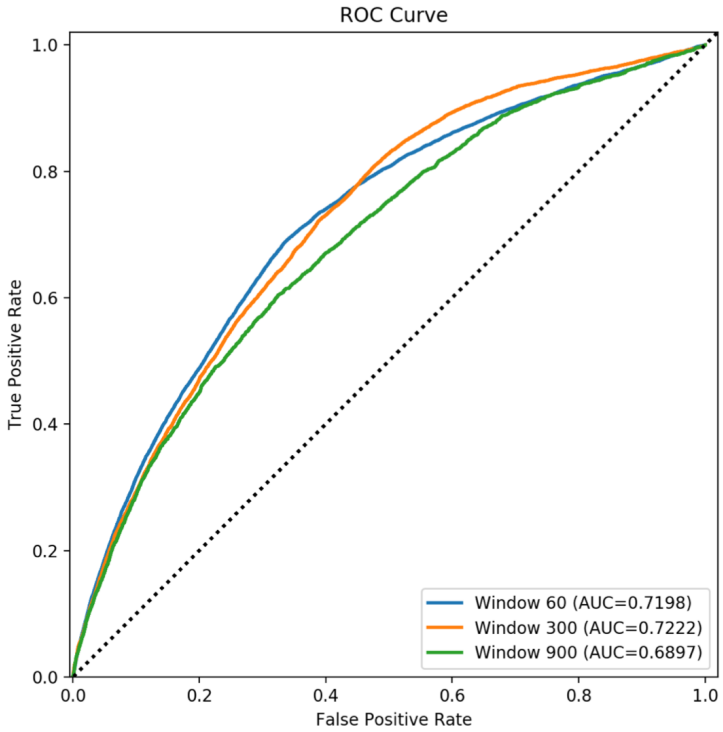


Figure 5.4: Gilon's DNN results on different window sizes.

Author	Year	Length (min)	Features	Model	Sens. (%)	Spec. (%)	Acc. (%)
Zong [50]	2001	30	PACs	ES	-	-	79*
Chazal [15]	2001	10	PSD	LDA	-	-	68*
Hickey [28]	2002	30	PACs PSD	LDA	79	72	75
		5	PSD	LDA	51	79	68
Thong [45]	2004	30	PACs	ES	-	-	89*
Chesnokov [16]	2008	30	PSD	SVM	83.72	76.47	80.51
			Non-linear	MLP	68.18	100	82
Mohebbi [36]	2012	30	PSD Bispectrum Non-linear	SVM	96.2	93.1	-
Costin [17]	2013	30	Time-domain PSD Morphology	t-test	89.44	89.29	90
Boon [12]	2016	30	Time-domain PSD Bispectrum Non-linear	SVM	81.1	79.3	80.2
		15			77.4	81.1	79.3
		10			58.5	81.1	69.8
Boon [11]	2018	5			86.8	88.7	87.7
Narin [37]	2018	5	Time-domain	kNN	84	84	84
			Time-domain PSD		92	88	90
Gilon [23]	2019	5	Raw RRs	BiGRU + CNN	68	68.7	-

Table 5.3: Published works on AF forecasting. Results marked with a star (*) indicate that these results were obtained by considering the ECG episodes by pairs, rather than classifying each episode independently.

Chapter 6

Confirming Coumel's arrhythmogenic factors

6.1 Objectives

We presented in Section 2.3.3 how Coumel's Triangle formalised the necessary components for AF initiation. In Section 4.1, we investigated how we could map these factors to computable features from RR intervals. Let us remind ourselves said factors and features :

1. the trigger factor can be represented by PACs;
2. the ANS modulation is associated with LF and HF values from the PSD.

It now naturally comes to confirm on our own datasets that indeed these **features exhibit some discriminating behaviour towards ECG segments prior the onset of AF**.

6.2 Data

We use the 140 long-term ECG recordings assembled and annotated by Dr. Jean-Marie Grégoire. 308 PAF events were identified in those recordings.

6.3 Methods

For each PAF event in the dataset, we analyzed the recording starting from the event until 1h before the event. The analysis consisted of assessing the evolution of the aforementioned features before the onset of PAF. To realize this, we implemented the following procedure:

1. set $t =$ `time-of-paf-event`;
2. set *window* = `5mins-RR-intervals-before` t;
3. compute the number of PACs from *window*;
4. compute the PSD from *window* and extract LF, HF and LF/HF;
5. step backwards in time by setting $t = t - 10s$;

6. repeat 2 until t is distant for more than 1h from the PAF event.

Then, for each time t iterated, we computed the average of the derived PACs, LF, HF and LF/HF across the windows for said time t and plotted the evolution of said average for each feature from 1h before the PAF event until the onset of the event.

6.3.1 Counting PACs

We identified and counted PACs in each episode by respecting the condition presented in Section 4.1.1 : if two successive RR intervals differ by more than 20%, mark the second one as an ectopic beat.

6.3.2 Pre-processing and computing spectral features

We first removed ectopic beats to obtain NN intervals (using the same rule as above). Remember that using NN intervals provides better estimation of ANS modulation, as ectopic beats are artifacts. We interpolated the removed ectopic beats with linear interpolation.

Then, to obtain an evenly sampled and detrended signal we:

- resampled the signal at 7Hz;
- interpolated missing RRs with linear interpolation;
- used constant detrending to detrend the signal.

We finally estimated the PSD using Welch's method (`NFFT` = 4096, Hann's window) and extracted LF, HF and LF/HF from their respective frequency ranges.

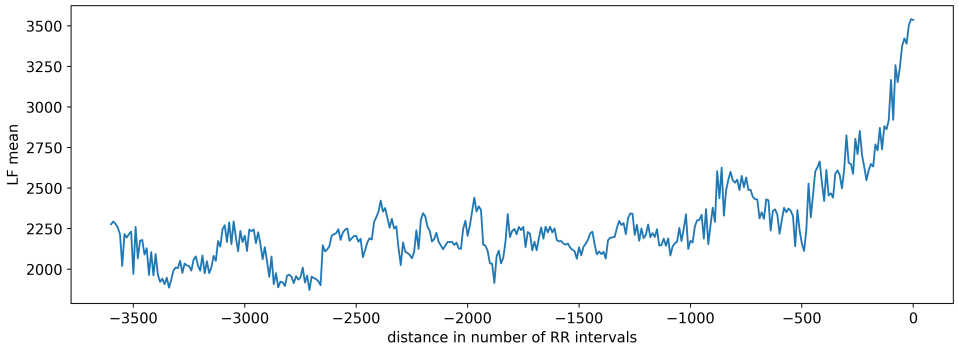
6.4 Results

Figure 6.1 reports results on Dr. Grégoire's data. It is directly clear that the heavy shift in LF and HF in the last few minutes preceding the onset of PAF **completely confirm** the role of **ANS modulation in facilitating the onset of PAF**. In both LF and HF cases, this shift seems to start approximately 500RR intervals before the start of the event.

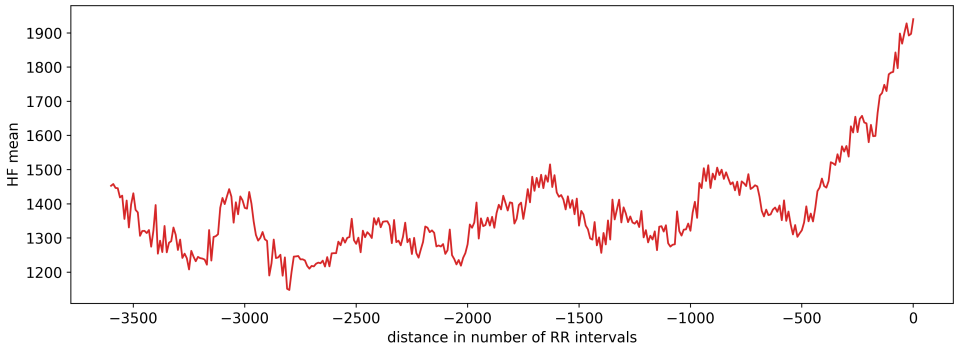
It is equally evident from Figure 6.1d that **ectopic beats play a trigger role in PAF initiation**, as the number of PACs seem to grow exponentially in the final moments before the onset of PAF.

6.5 Further validation with data from CHU Ambroise Paré

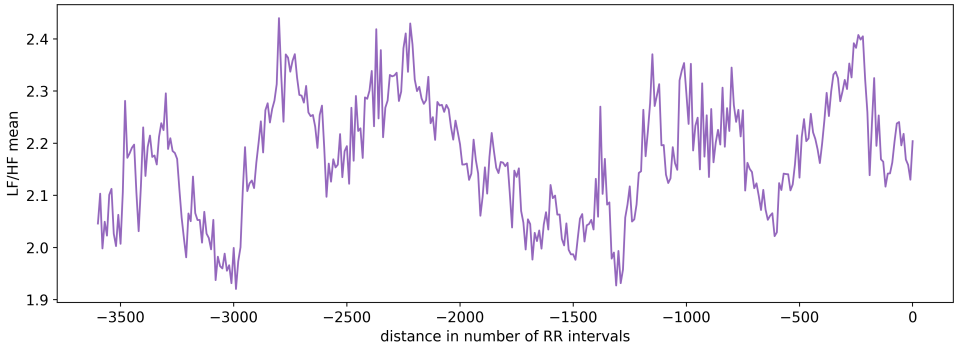
As we also had access to a dataset of 69 long-term ECG recordings (100 PAF events) originating from CHU Ambroise Paré, we ran the same experiments on this dataset to further validate Coumel's factors, as well as the data (it goes both ways). Figure 6.2 reports our findings on CHU Ambroise Paré's dataset. The reader will observe that exactly the same conclusion can be drawn as for Dr. Grégoire dataset, which indicate a certain consistency and robustness to Coumel's Triangle and its derived features.



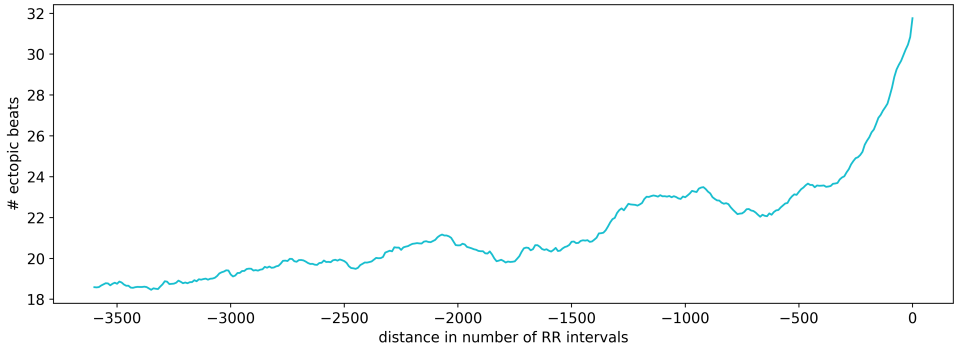
(a) LF means



(b) HF means

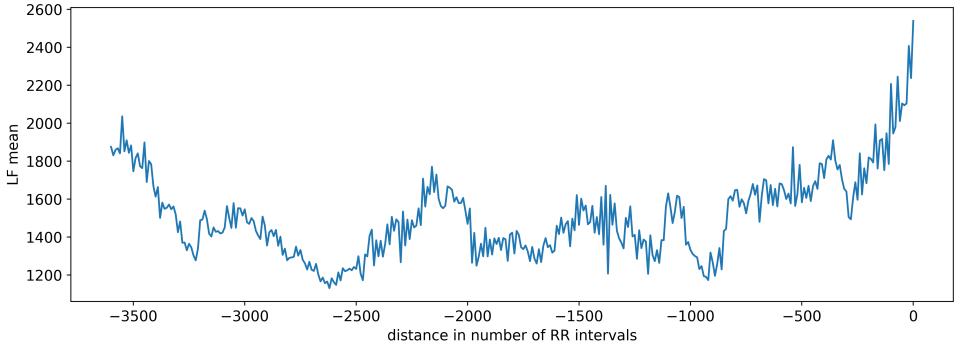


(c) LF/HF ratio means

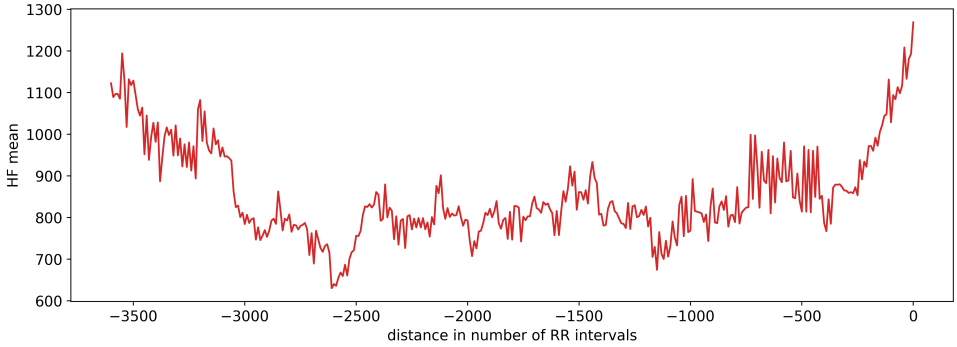


(d) Mean number of ectopic beats

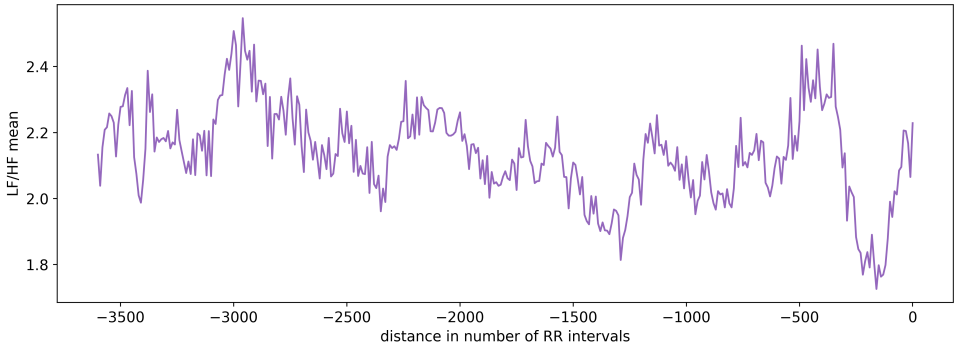
Figure 6.1: Evolution of Coumel's Triangle factors in the hour preceding AF events (Dr. Grégoire's data). Distance 0 marks the onset of AF.



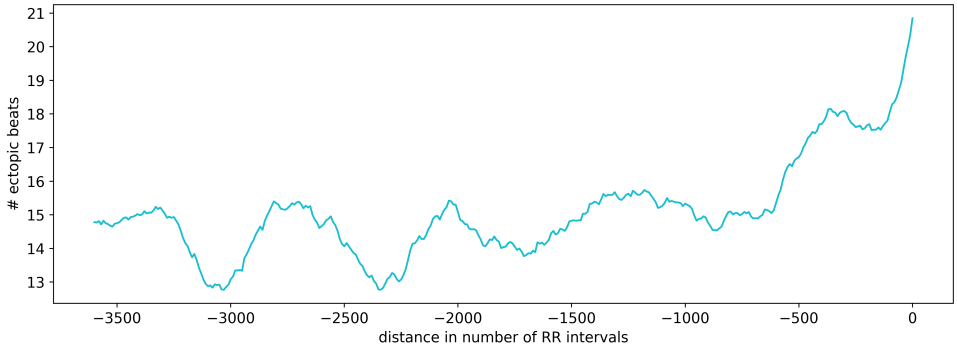
(a) LF means



(b) HF means



(c) LF/HF ratio means



(d) Mean number of ectopic beats

Figure 6.2: Evolution of Coumel's Triangle factors in the hour preceding AF events (CHU Ambroise Paré's data). Distance 0 marks the onset of AF.

Chapter 7

Injecting Coumel's features into DNNs

7.1 Procedure so far

Let us recap with Figure 7.1 what we have done so far:

1. We started by understanding what physiological phenomenons influenced AF initiation via Coumel's Triangle.
2. We mapped these influencing factors into features that we can extract from RR intervals:
   (a) the trigger factor is represented by PACs;
   (b) the ANS modulation is characterized by LF, HF and LF/HF values.
3. We confirmed the derived features by visualizing their evolution during a 1h window before AF events on two datasets.

7.2 Next objectives

We thus now have features that we can extract from RR intervals which were proved to be discriminating. Our next goals are to (also illustrated in Figure 7.1):

1. Develop a deep model to try and **compete** with the **best one implemented by Gilon** in his masters thesis [23].
2. Augment the competing deep model and Gilon's best model by **injecting** them with **Coumel's features**.
3. Report results for the features injections when injecting the models with **different subsets** of these features.

7.3 Methods

All the reported results in this chapter share the same methods:

- We used the dataset of Dr. Grégoire.

- We ran each experiment on 50 different splits of this dataset (as there is variance in the difficulty of a random split). Each split is made of:
  1. a training set to optimize the model's parameters;
  2. a validation set for early stopping;
  3. a testing set for final evaluation.
- Performance of each model is assessed by averaging the results on the testing set for the 50 different splits.
- Our deep models shared the same hyperparameters:
  1. learning rate $\alpha = 0.0001$;
  2. patience = 50;
  3. number of epochs = 1000;
  4. batch size = 512;
  5. the loss function is the binary crossentropy.

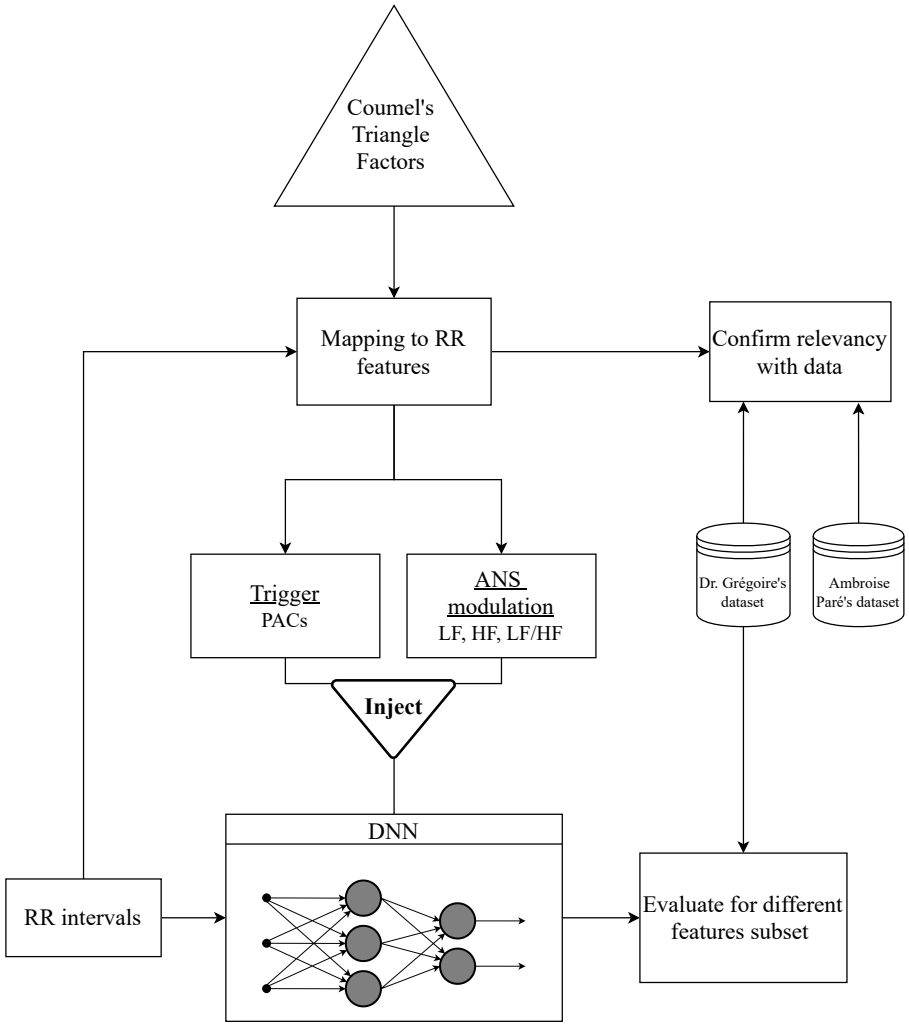


Figure 7.1: Previous steps leading us to the next objective: injecting Coumel's Triangle derived features into a DNN.

7.4 Baseline model: Gilon's network

Gilon's best model, called BiGRU + CNN (abbreviated GRU in figures), is a 4-layers deep neural network composed of two 1-D CNN layers followed by a bidirectional GRU layer and a final FC layer (Figure 7.2).

The reported network AUC is 0.72. This was obtained with a tolerance of 30 to augment the dataset. We, however, use a tolerance of 0 in order to make computations faster. As the reader will observe in the ROC curves further below, we obtain a slightly (expected) worse AUC of 0.69 when testing Gilon's model with a tolerance of 0.

We will also use a tolerance of 0 for the competing model. Since we only want to compare the performances of the models and since we respect the same testing conditions for all models, having a slightly worse baseline model is not relevant.

Layer	Type	Parameters	Output shape
1	Input layer		300 x 1
2	1D convolution	filers:100, kernel size: 3	298 x 100
3	1D convolution	filers:100, kernel size: 3	296 x 100
4	Global max pooling		100
5	Reshape		100 x 1
6	Bidirectional GRU	units: 100	200
7	Fully connected	units: 1, activation: sigmoid	1
	Total number of parameters	91 901	

Figure 7.2: Baseline model: Gilon's DNN architecture for AF forecast.

7.5 Competing model: ResNet for time series

As the BiGRU + CNN model uses a recurrent construct (GRU) and is rather conservative regarding its depth, the idea is to implement a competing model which:

1. has potentially more depth;
2. does not use recurrent constructs.

7.5.1 Residual Network

The Residual Network (ResNet) for time series described by Wang et al. [46] fulfills both desired requirements. Figure 7.3 depicts the network. As the figure illustrates, a ResNet is composed of an arbitrary number of **repeatable residual blocks**.

Each block mainly applies three successive convolutions on its input. A **shortcut connection** (arrow marked with a "+" in Figure 7.3) **sums** the **input** of the block with the **result of its three convolutions** and the sum then serves as **output of the block**. Shortcut connections thus **skip layers** and allow the **gradients to backpropagate easier**, which enables the block to be **repeated**.

7.5.2 Tuning the repetitions of residual blocks

We implemented the suggested ResNet of Wang et al. [46] as it is depicted in Figure 7.3. We assessed the performances of ResNets with varying numbers of residual

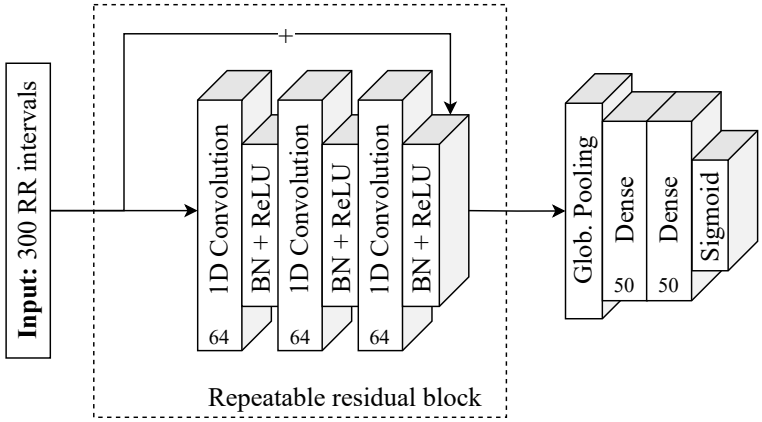


Figure 7.3: Competing model: the ResNet architecture for time series. [46]

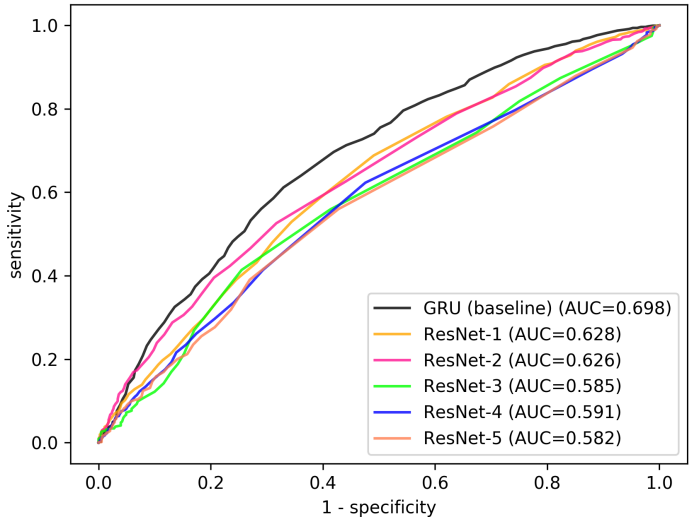


Figure 7.4: ROC curves for ResNets with varying numbers of repetitions of the residual block (with ResNet-k a ResNet with k residual blocks).

blocks in order to build our final competing model. We report the results for each ResNet-k with $k \in [1, 2, 3, 4, 5]$ the number of residual blocks.

Figure 7.4 reports our findings. Let us first note that **none of the ResNets-k performed better than the baseline** (Gilon's network). It is difficult to explain precisely why but the lack of recurrent construct may be a reason.

It is unclear whether the ResNet with 1 residual block (ResNet-1) is better than the ResNet with 2 residual blocks (ResNet-2) as the AUCs are very comparable (resp. 0.628 and 0.626). It appears, however, that the ResNet architecture does not benefit (in our case) from added residual blocks when $k > 2$, as ResNet-3, ResNet-4 and ResNet-5 all obtain an AUC $<= 0.59$. We suppose that this is the case because our **dataset is too small**, making it more difficult for **deeper** networks to avoid **overfitting**.

As ResNet-1 exhibits (marginally) more promising results, it is the ResNet model we chose to compete with the BiGRU + CNN model in the next section.

7.6 Injecting Coumel's features

In an attempt to improve performance on the baseline (BiGRU + CNN) and competing (ResNet-1) models, we augment the models by supplying them with different subsets of the 4 features that we derived from Coumel's work: LF, HF, LF/HF and PACs. For each model, we assess performance on the following features subsets injections (with *model* = BiGRU + CNN or ResNet-1):

1. PACs (*model*-PAC);
2. LF, HF, LF/HF (*model*-Freq);
3. PACs, LF, HF, LF/HF (*model*-all).

7.6.1 Architecture modifications

We decided to inject the features as input of the two final Fully Connected (a.k.a. Dense) layers, which were present in both networks. Said layers thus receive as input a concatenation of the DNN outputs and the hand-crafted features (see Figure 7.5). In essence, one can interpret the DNN as feature extractor, the hand-crafted features as the *human helping hand* and the Dense layers as final classifier collecting both sets of features.

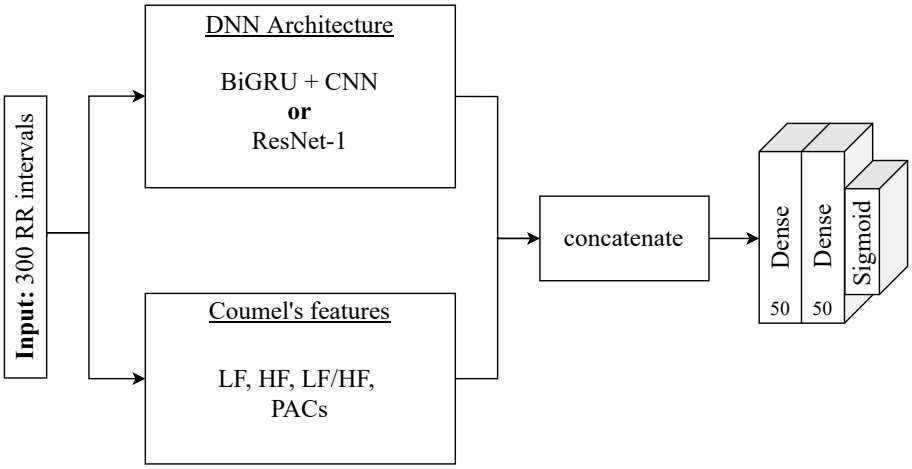


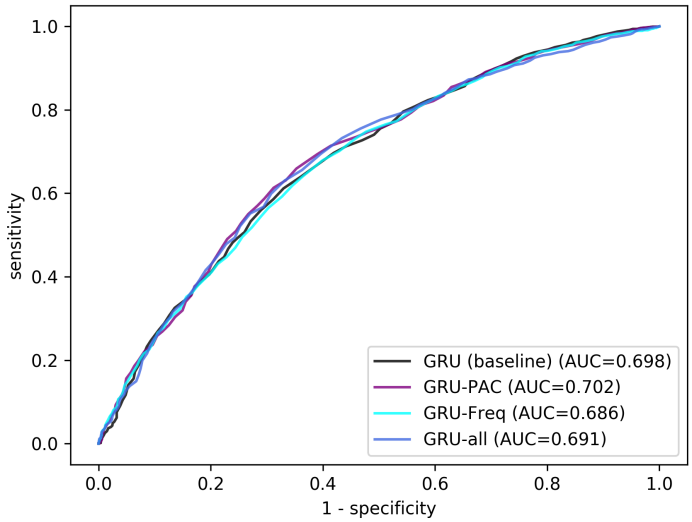
Figure 7.5: Architecture template when adding LF, HF, LF/HF and PACs features to the deep models.

7.6.2 Results

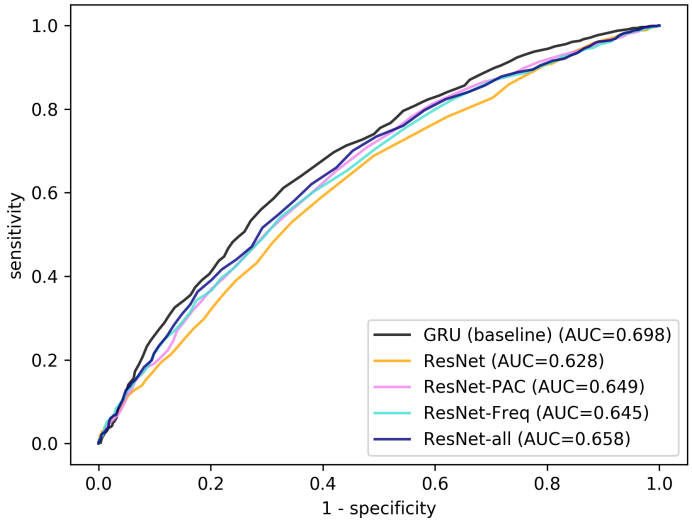
Figure 7.6 reports the averaged ROC curves and AUC over the test sets of the 50 distinct dataset splits for the two models and the different features subsets.

It is quite apparent on Figure 7.6a that **none of the subsets** of injected hand-crafted features **help** the **BiGRU + CNN** model in achieving higher performance. We have trouble explaining exactly why this is the case. It could be that the final Dense layers **do not use** at all the **added features**, and further investigation of the network's weights would therefore be required. It is also conceivable that there exists heavy **overlapping** between the **injected features** and the **features learned** by the DNN. Again, closer inspection of the network would here be necessary to draw any tangible conclusion.

For the **ResNet-1** model, however, the performances are **slightly increased** (Figure 7.6b). When supplying all the hand-crafted features (ResNet-all), an improvement of up to 3% is observed compared to the raw model without added features (ResNet). Still, these improvements are **insufficient** to **beat** the baseline **BiGRU + CNN** model as the latter surpasses every augmented ResNet model by at least 5%.



(a) BiGRU + CNN model



(b) ResNet-1 model

Figure 7.6: ROC curves for the models with the different features subsets.

Chapter 8

Reproducibility of AF forecast research

8.1 Motivations and objectives

The last chapter presented our unsuccessful attempts at improving the results obtained by Gilon [23]. Not only did the suggested competing network perform much worse, but augmenting both networks with Coumel's features have also been proved to be fruitless.

When comparing model performances reported in the AF forecast literature, it is apparent that our own DNN models are not up to the challenge. In this context, we decided to try and reproduce state of the art methods for AF forecast. The objective is two-fold.

1. Understand the applied ML techniques and pipelines to improve our work.
2. Confirm that the state of the art AF forecast results are sound and reliable.

8.2 The state of reproducibility in ML literature

Reproducing published works is, however, far from an easy and straightforward task. Indeed, faithful reproduction requires the original paper to be exhaustive and rigorous when describing the employed methods, which is rarely the case. Ambiguities, oversights and usage of private datasets are all a thorn in the side of reproducibility in the Machine Learning literature.

So much so, that many recent published papers discuss the problem of reproduction of existing ML research to the point of calling the current state of affairs a reproducibility crisis [41, 10, 35, 39]. A few months ago, an entire book, dedicated to the reproducibility of existing works in the field of Credit Card Fraud detection, was even (partly) released online [32], which further stresses the current emphasis put by ML practitioners on publishing reproducible work.

Fully aware of the difficulty of the task ahead, let us dive into our reproductions attempts of existing AF forecast systems.

8.3 Narin et al. [37]

8.3.1 Methods

The first AF forecast system that we aim to reproduce in detail is the one proposed by Narin et al. [37]. Figure 8.1 summarizes the methods of the study.

Data

They use the learning set of the AFPDB [4]. They thus dispose of 25 30mins ECGs segments just before a PAF event and 25 30mins ECGs segments distant from at least 45mins of any PAF event. These recordings were collected on 25 different PAF patients (1 pair of (prior episode, distant episode) per patient). The 28 pairs of the AFPDB testing set are overlooked by this study.

Pre-processing

RR intervals of 5mins ECG segments are extracted from the 50 episodes. As the FFT to compute the PSD requires evenly sampled data that is free of non-stationary components, the RR signals are re-sampled at 7Hz (with cubic spline interpolation) and detrended with Smoothness Prior Analysis [47].

Features

The features derived from the RR signals are the following :

- 8 time-domain features (AVNN, SDNN, SDSD, RMSSD, NN50, NN20, pNN50, pNN20);
- 3 frequency-domain features (VLF, LF, HF, PSD-TOTAL).

Model

The k-NN algorithm with Euclidean distance is chosen as classifier. The study employs GA as feature selection procedure. Note that, as it is shown in the bottom right of Figure 8.1, the number of neighbours k is not optimized by GA. Instead, the GA optimization is performed for 10 different values of k with $k \in [1, 3, 5, \ldots, 19]$.

8.3.2 GA for feature selection and overfitting

Let us delve deeper into how exactly GA is employed by Narin et al. as feature selection wrapper method. We adapt the work presented by Huang and Wang [29] to illustrate the GA feature selection process of Narin et al. in Figure 8.2:

1. initially, a random population of chromosomes is generated. Each chromosome is a binary string where the i-th bit is 1 if feature i is selected, else it is 0;
2. the dataset is split into 10 folds (i.e. groups);
3. for each chromosome in the current generation, select features according to its genotype;
4. then, for each fold:

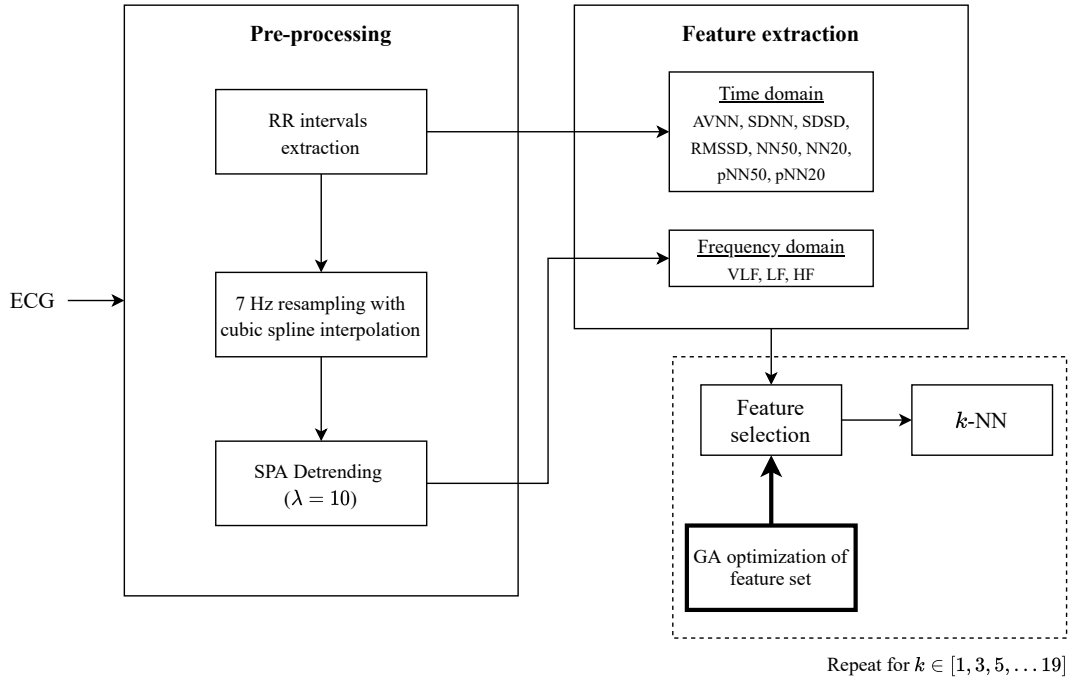


Figure 8.1: AF forecast system as proposed by Narin et al. [37].

(a) take $fold_i$ as validation data and the 9 other folds as training data;

(b) fit the k-NN with the selected features on the training data;

(c) compute performance metrics on the validation data $fold_i$;

5. compute the **fitness function as the average of the computed performance metrics** (average of FN + FP);

6. if a termination criterion is true (max. number of generations, thresholded fitness improvement, etc.), **report the latest best validation result as final model evaluation**, else apply classical genetic operations to produce the next generation and repeat from 3.

This step by step rendition of the GA optimization implementation in Narin et al. for feature selection clearly highlights an **overfitting problem**.

GA is a search algorithm that explores a large portion of the feature subset space and is, in this implementation, completely **guided by the validation sets performance**. Hence, the GA could **choose a bad feature subset with high validation sets performance but poor generalization**. Moreover, since there are so many candidate feature subsets, it is likely that one of them leads to high performance on the validation sets but has, again, mediocre generalization. This is discussed thoroughly in [14].

To avoid these overly optimistic final performance measures, there is a very strong emphasis in [14, 30, 43] that experiments that use cross-validation to guide the feature subset search must report final performance on a **separate test set** that was never used during the optimization process (Figure 8.3).

Let us also note that the train-validation folds are not communicated in the study. Yet, since the dataset is so small (50 data points), there is an inherent high variance

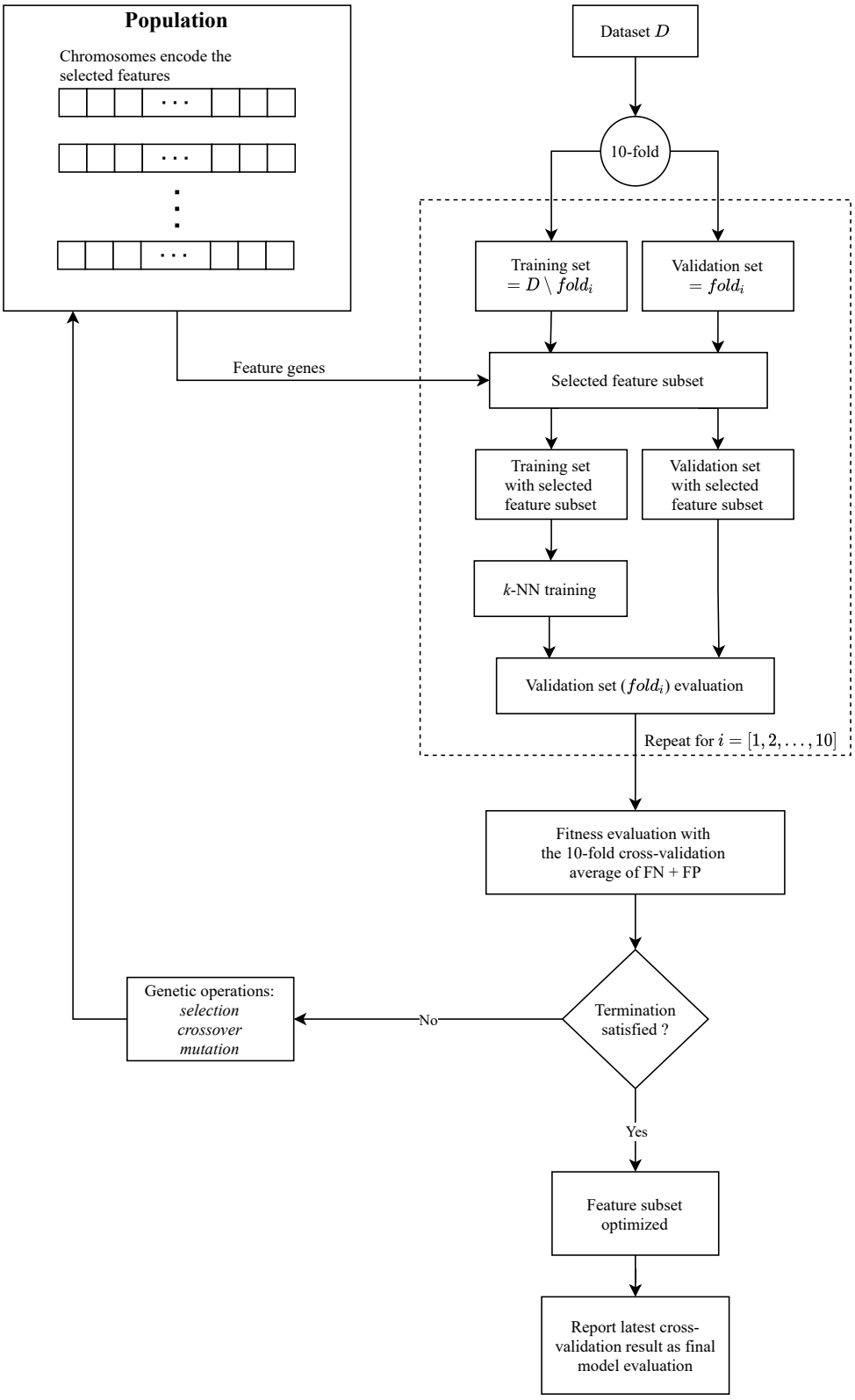


Figure 8.2: GA optimization for feature selection as implemented by Narin et al. (inspired by [29]).

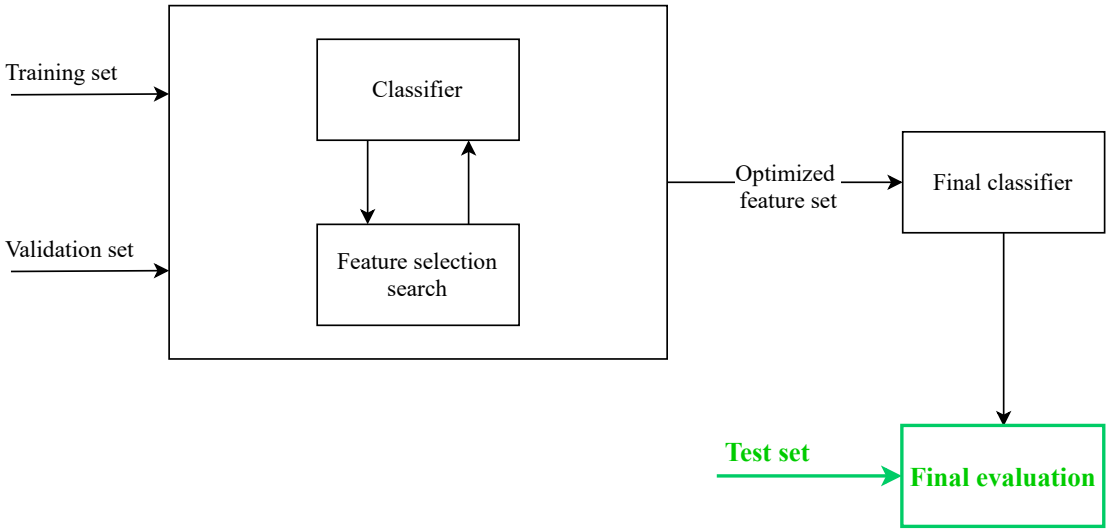


Figure 8.3: Outline of wrapper method with correct final evaluation (inspired by [43]). Note that the test set used for final evaluation is not used during the feature subset optimization procedure.

in the **difficulty of the split** and a favorable easy split can make the **validation performance skyrocket**. On the other hand, if, for example, the only 3 examples manifesting a specific phenomenon in the dataset are grouped into the same fold, the k-NN classifier will fail to find adequate neighbours in the other folds and inference performance will therefore plummet.

8.3.3 Reproducibility

Ambiguities and oversights

Unfortunately, the article fails to deliver an exhaustive and thorough description of their methods. We hereby list some oversights as well as some hypotheses we were forced to make when trying to reproduce this study.

- As the authors do not mention RR intervals transformation to NN intervals, we used RR intervals to compute all features, despite the fact that some features (eg. NN50) should require NN intervals to be computed.
- ECGs episodes of 5mins are used but it is not specified which segment of the distant ECG (in the ECG pair) are used as negative examples (not preceding AF). We opted for the first 5mins of each distant ECG as they are most likely the farthest away from any AF event.
- 5mins RR segments are often approximated by counting 300RR intervals. As the study does not specify whether they use RR counts or seconds by cumulating RRs, we ran the experiments for both cases.
- The train data split is not said to be a patient-level split; hence we ran experiments for patient-level splits (both ECGs in each pair must belong to the same fold) as well as for unrestricted splits.

- Finally, as we mentioned above, the train-validation split is not reported i.e. the exact content of each fold is unknown. We thus ran experiments for 1000 different random dataset splits.

Objectives

We try to reproduce the results obtained by the two following models:

1. **model A**: k-NN with $k = 3$ and feature subset {RMSSD, VLF, LF, PSD-TOTAL} which obtained 92% sensitivity, 88% specificity and 90% accuracy;
2. **model B**: k-NN with $k = 1$ and feature subset {SDNN, NN50} which obtained 84% sensitivity, specificity and accuracy.

Methods

We use the 25 ECG pairs from the AFPDB training set and for each of the two aforementioned models we run experiments with the following configurations:

1. patient-level split and 300RR windows;
2. patient-level split and 5mins windows;
3. unrestricted split and 300RR windows;
4. unrestricted split and 5mins windows.

For each configuration, we make random training-validation splits on the dataset 1000 times (w.r.t. patient-level or unrestricted) and for each split, we compute the 10-fold cross-validation sensitivity, specificity and accuracy (remember that the article reports 10-fold cross-validation performances as final model evaluation).

To exhibit the overfitting problem discussed above, we also report both models performances on the withheld AFPDB test set (28 ECG pairs).

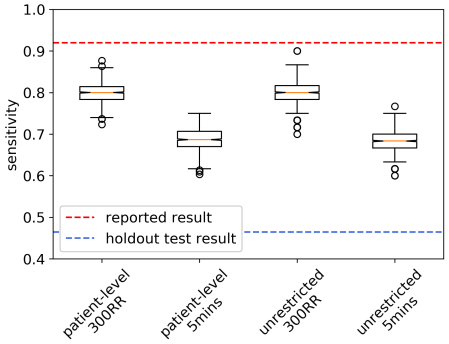
Results

Model A We report performances on 10-fold cross-validation for 1000 train-validation splits for the above configurations in Figure 8.4.

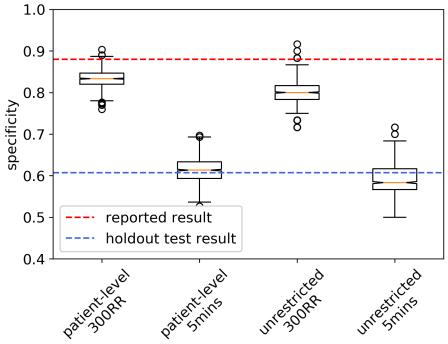
Let us first note that the diameter of the scattering completely validates our previous point regarding the high variance in the model performances for different train-validation splits. Some dataset splits are simply *easier* than others.

It appears that 300RR were the opted choice in the article, as it is quite clear that this configuration out-performs the 5mins implementation for all metrics. Surprisingly, however, there seems to be no significant difference between patient-level and unrestricted splits.

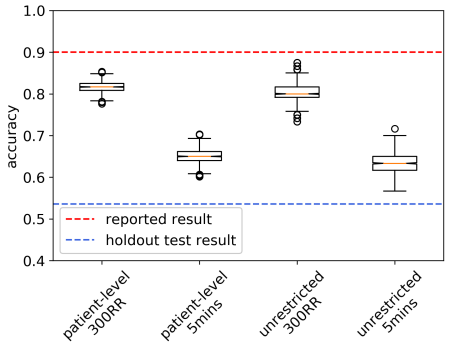
When metrics are taken independently, it seems like some dataset splits perform as well cross-validation wise as the reported results in the paper. When considering the sensitivity and specificity performances of a single split at the same time (Figure 8.4d) however, **no split is able to reproduce exactly the same performance as was reported in the paper**. As it is evident that some splits come close to these results though, we suppose that **there exists a split of the dataset that would make model A perform as well as the study suggests on the validation set**.



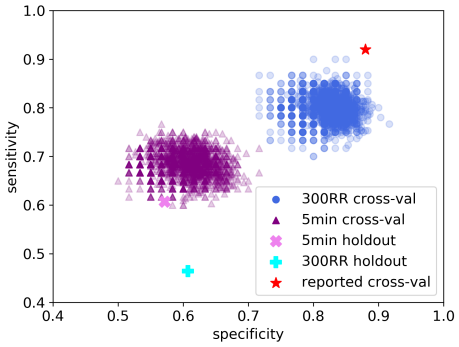
(a) Sensitivities



(b) Specificities



(c) Accuracies



(d) Combined sens. and spec. results

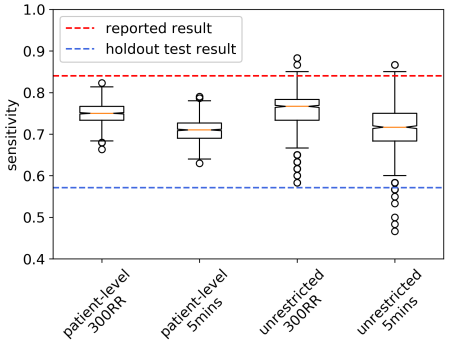
Figure 8.4: Reproduction results of **model A** (the article reported 92% sensitivity, 88% specificity and 90% accuracy, these values are shown in red). In total, 1000 different training-validation splits were evaluated. Note that all metrics are computed on the validation set except otherwise specified (i.e. holdout test results).

Although, in rare cases, the cross-validation performances are somewhat close to the reported values, the **overfitting problem that we highlighted above is here plainly demonstrated**. Indeed, the results on a separate withheld test set are much poorer than the 10-fold cross-validation results. This hence indicates that the model overfitted the validation set during feature selection GA optimization.

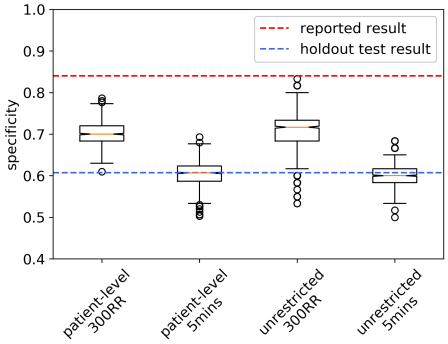
Model B Figure 8.5 presents the reproduced results for Model B. As for the Model A, the 300RR configuration performs better than the 5mins one, even though the difference is less significant. There is also no clear cut difference between patient-level and unrestricted splits.

When considering the reproduction results on cross-validation, we, again, make the same observation as for the other model: we fail to reproduce a result performing as well as mentioned in the article.

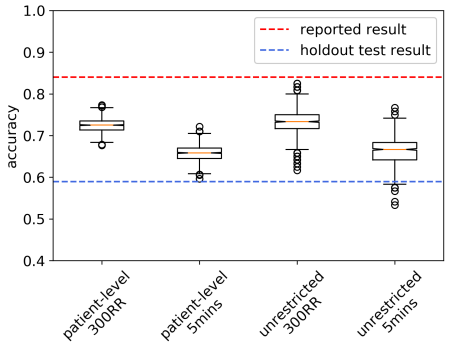
Overfitting is once more revealed as the model performs much worse on a separate test set (Figure 8.5d)



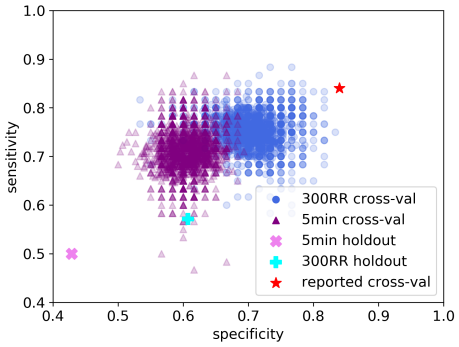
(a) Sensitivities



(b) Specificities



(c) Accuracies



(d) Combined sens. and spec. results

Figure 8.5: Reproduction results of **model B** (the article reported 84% sensitivity, specificity and accuracy).

8.4 Mohebbi et al. [36]

8.4.1 Methods

The second article we try to reproduce is a publication by Mohebbi et al. [36]. This study is of particular interest, as their reported results are the best in the AF forecast literature. We depict the implemented methods of the research in Figure 8.6.

Data

The entire AFPDB in group **A** (PAF patients) is used. The authors thus possess in total 106 ECGs segments of 30mins (53 pairs, 25 from the AFPDB learning set and 28 from the AFPDB testing set).

Pre-processing

The entire 30mins segments are kept and the study does not explore shorter episodes. RR intervals are extracted from these segments and no further pre-processing is reported in the article.

Features

In total, 12 features are derived from the RR intervals:

- 2 frequency-domain features (LF, HF);
- 6 bispectral features ($\mathrm{E}_1, \mathrm{E}_2, \mathrm{H}_1, \mathrm{H}_2, \mathrm{H}_3, \mathrm{H}_4$);
- 4 non-linear features (SamEn, SD1, SD2, SD1/SD2).

Model

A SVM with Radial Basis Function (RBF) kernel is trained on the AFPDB learning set. The hyperparameters σ and C are tuned with a holdout validation set which they subtract from their training set. The reported optimal values are 3.6 and 10 for σ and C, respectively. Final performance is assessed on the withheld AFPDB test set.

Note that no feature selection is applied, and the entire feature set is supplied to the SVM.

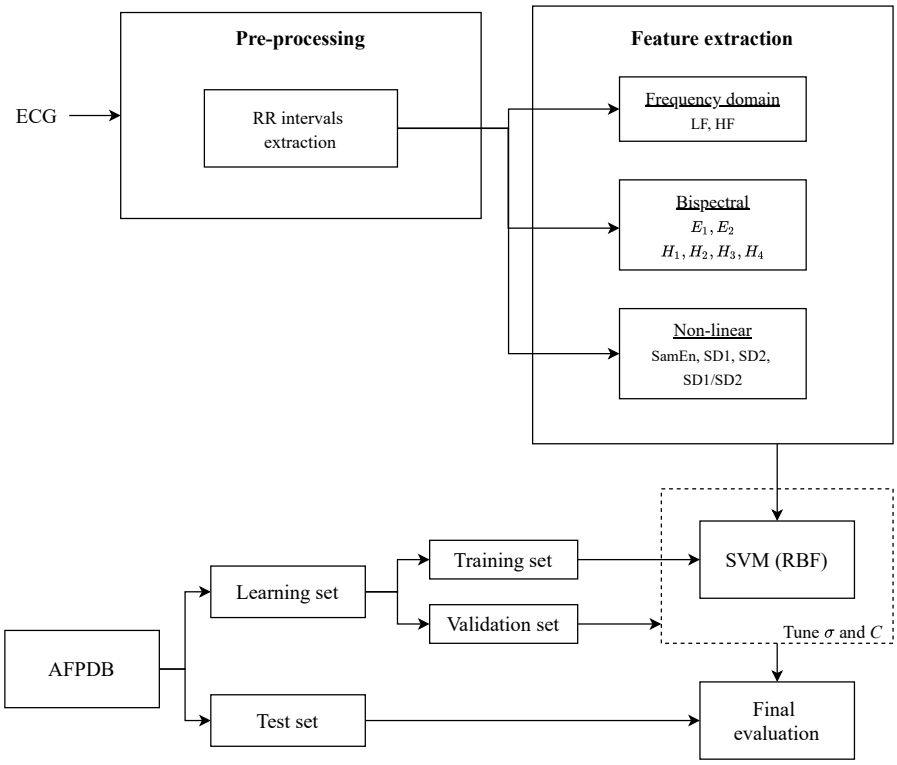


Figure 8.6: Methods of AF forecast system by Mohebbi et al. [36].

8.4.2 Reproducibility

Ambiguities and oversights

Similarly to Narin et al., the study lacks rigor in reporting their exact methods:

- The parameters of the algorithm they used for bispectral estimation are not communicated. We discuss this in next section.

- It is unclear whether some scaling and/or normalization is applied on the bispectrum before computing the bispectral features.
- It is common in SVM models to use normalization of the inputs but the authors did not indicate doing so. So, we had to implement models with and without normalization.
- Lastly, they split the AFPDB learning set into a training set and a validation set. The details of this dataset split are unfortunately not reported (not even the validation size). Hence, in the same way as our re-implementation of Narin et al.'s work, we ran experiments for different dataset splits (note that we did not need to split the test set though, as the authors withheld it for final model evaluation).

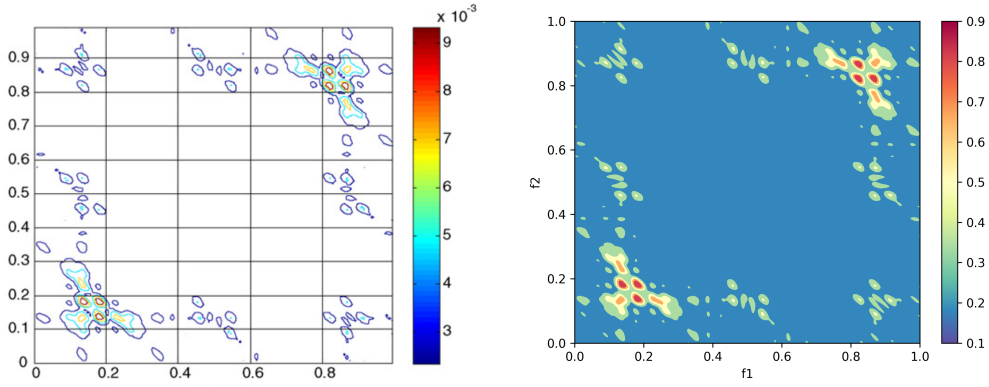
Bispectrum calculation

As the article does not give the parameters of the algorithm used for bispectrum extraction, we had to search for said parameters via some *trial-and-error* reproductions by using the two bispectra that the paper displays.

We ended up finding the correct values for the segment `p16`, as Figure 8.7 illustrates (using MATLAB's HOSA [44] with `nlag`=64, `nsamp`=64, `overlap`=0, `nfft`=128, `window`=`default`).

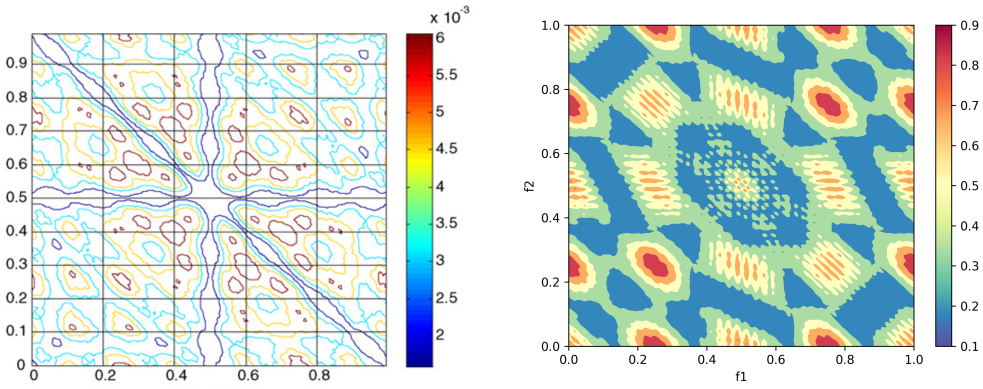
We then used the same parameters in order to reproduce the second bispectrum of the article, which is supposedly segment `p03` (Figure 8.8a), and obtained a completely different result from the original (Figure 8.8b).

By computing the bispectrum for each segment in the AFPDB, we found one that closely matches the second bispectrum of the article (Figure 8.9). It is hence likely that the authors wrongly reported the segment number for the second bispectrum.



(a) Original (b) Reproduction

Figure 8.7: Successful reproduction of bispectrum found in the study for AFPDB segment `p16` (after *some* trial and error).



(a) Original (b) Reproduction

Figure 8.8: Failure to reproduce the other bispectrum found in the article, despite using the same bispectrum parameters as in Figure 8.7 (a faithful reproduction). This is segment `p03` of the AFPDB according to the article.

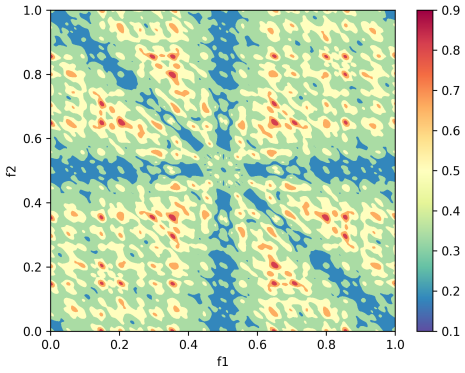


Figure 8.9: Segment `p19` of the AFPDB. We suppose that this is likely the real segment from which the authors derived the bispectrum in Figure 8.8a.

Objectives

We try to reproduce the result obtained by a SVM with RBF kernel ($\sigma = 3.6$, $C = 10$) and the following 12 input features on 30mins segments:

- frequency-domain (2): (LF, HF);
- bispectral (6): (E_1, E_2, H_1, H_2, H_3, H_4);
- non-linear (4): (SamEn, SD1, SD2, SD1/SD2).

It is reported that this model obtains 96.3% sensitivity and 93.1% specificity on the AFPDB test set.

Methods

Since neither the contents of the train-validation split nor the size of the validation set were reported, we made 1000 splits of the 50 30mins segments from the AFPDB

learning set (group **A**) for 5 different validation set sizes (10%, 15%, 20%, 25% and 30% of the learning set) for a total of 5000 splits.

For bispectrum extraction, we use MATLAB's HOSA [44] with the parameters found when we tried to reproduce the bispectra displayed by the study (`nlag`=64, `nsamp`=64, `overlap`=0, `nfft`=128, `window`=`default`).

We report result on the AFPDB test set (56 withheld segments) a SVM with and without input normalization (2 configurations), as usage of normalization was not clarified by the authors.

Results

Figure 8.10 clearly shows that we are **far from able to reproduce the reported results** by Mohebbi et al, even though we tried for many dataset splits.

We acknowledge that this could come from a lack of scaling and/or normalization of the bispectrum in our implementation.

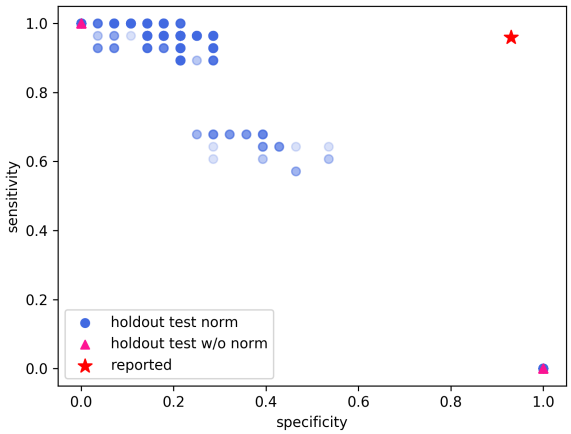


Figure 8.10: Reproduction results on the AFPDB test set for a SVM with and without input normalization.

8.5 Boon et al. [11]

8.5.1 Methods

The final and third study that we reproduce is a work from Boon et al. [11]. Note that the AF forecast system is very similar to the one implemented by Narin et al. with some adjustments.

- GA is used to optimize feature selection **as well as** the pre-processing **for each** of the features categories.
- Instead of repeating the GA optimization for different values of k, GA optimization is repeated exactly 5 times.
- A SVM classifier is used instead of a k-NN.
- Bispectral and non-linear features are additionally extracted.

Hence the reader is invited to consult Figure 8.1 to have a general representation of the methods employed in this study.

Data

Similarly to Mohebbi et al., the entire AFPDB in group **A** is used. A total of 106 segments is hence available.

Pre-processing

The work uses windows of 5mins RR intervals. The applied pre-processing and its parameters (e.g. method of interpolation) are tuned via GA for each features categories.

Features

The authors use almost all the features used in the PAF forecast literature. They obviously apply feature selection with GA on these features. In total, 53 features are extracted from time-domain, frequency domain, bispectral domain (for each subband region) and non-linear analysis.

Model

A SVM with RBF kernel is the classifier. Its hyperparameters γ and C are optimized by GA. Hence, GA is used for tuning:

1. the pre-processing for each features categories;
2. the feature selection;
3. the hyperparameters of the classifier.

8.5.2 Reproducibility

Ambiguities and oversights

- The authors report final evaluation on the validation set (similarly to Narin et al.) but GA optimization is guided by 10-fold cross-validation metrics. As GA is repeated 5 times and optimized the entire pipeline of the system, there is some risk of overfitting. By not reporting results on a separate test set, overfitting cannot be assessed.
- Unfortunately, as it seems to often be the case, train-validation splits are not reported.
- Although hyperparameters are tuned by GA, their final optimized values are not reported.
- We encounter the same problem as for the Mohebbi et al. implementation, as no bispectrum post-processing is reported.

Objectives

We try to reproduce the results for the feature selection (NN50, pNN50, SamEn, SD2, AR-LF, ROI-WCOB(f_2), LL-H_1). This optimized selection obtained a sensitivity of 86.8%, a specificity of 88.7% and an accuracy of 87.7% on 10-fold cross-validation.

Methods

We apply the optimized pre-processing steps described in the last page of the article before extracting the selected features.

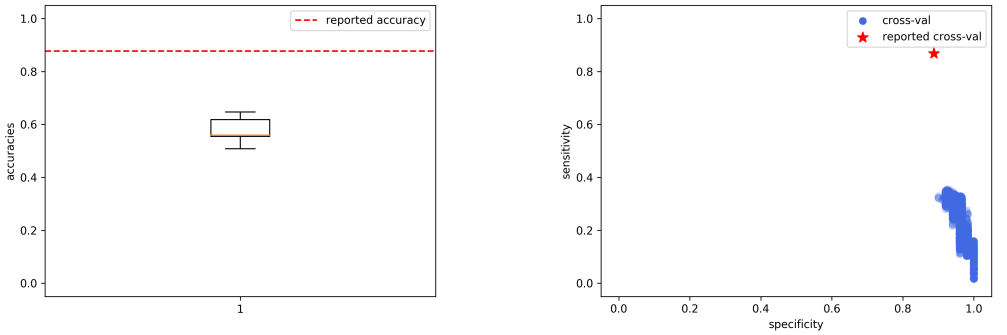
As neither the hyperparameters of the SVM nor the train-validation split are communicated, we make 20 different dataset splits and for each one we sample γ 50 times and for each of the γ values we sample C 50 times. The samples are taken from an exponential distribution with scale 0.1 and 100 for γ and C respectively.

We then report sensitivity and specificity on the 10-fold cross-validation for the current dataset split.

Results

At first glance Figure 8.11b may suggest that, by varying the decision threshold of the classifier, the results could come closer to the reported cross-validation values. However, this is denied by Figure 8.11a as the accuracies we obtained are indeed much lower than the expected results.

Hence, as it was the case for the Mohebbi et al. re-implementation, we are unsuccessful at reproducing the results reported by Boon et al.



(a) Accuracies

(b) Combined sens. and spec. results

Figure 8.11: Reproduction results of Boon et al. [11] for 20 different dataset splits, with each split being experimented for 50×50 different hyperparameters values of the SVM (γ and C).

8.6 Summary

In this final section, we summarize the reproduction results for the re-implemented articles in a few key points. Note that we **contacted the authors** of the studies that we were unsuccessful at reproducing. **None of them responded**.

Narin et al.

1. Final evaluation on validation set.
2. GA overfits the validation set.
   (a) Fitness function depends on validation set performance.
   (b) GA feature selection is run multiple times (tuning k).
   (c) Dataset is small.
3. Train-validation split content not reported.
4. Reproduction results close from reported on the validation set (for some rare splits only).
   (a) Highlights the variance in the difficulty of the splits.
   (b) **When evaluated on the AFPDB test set, performances are much poorer**.
   (c) This confirms the **validation set overfitting**.

Mohebbi et al.

1. Missing parameters for bispectral calculation.
2. Unclear post-processing of bispectra (scaling/normalization).
3. Normalization of SVM not reported (but likely).
4. Train-validation split content and size not reported.
5. **Reproduction results very far away from reported result**.

Boon et al.

1. Final evaluation on validation set.
2. GA at risk of overfitting.
   (a) GA guided by validation set results.
   (b) GA tunes the entire pipeline.
   (c) GA is run 5 times.
   (d) As no results are reported on a withheld test set, it is not possible to assess whether or not overfitting was avoided (and it would be useless to do it ourselves as we cannot reproduce the cross-validation results).
3. Train-validation split not reported.
4. SVM hyperparameters not reported.
5. **Reproduction results on cross-validation do not at all reflect the reported results**.

Chapter 9

Conclusion

9.1 Summary

We introduced this work by giving some important medical knowledge regarding AF and in particular the factors influencing its initiation, which were formalised by Coumel's Triangle [18].

From this medical expertise, we derived computable features from RR intervals that could be used in a typical inference model:

1. the trigger factor can be represented by PACs;
2. the ANS modulation is associated with LF and HF values from the PSD.

We assessed the evolution of these features on 1h windows before AF events and observed that said features indeed had discriminating power regarding the onset of AF. This was achieved on both Dr. Grégoire's dataset and CHU Ambroise Paré's dataset. We also gave a comprehensive list of the rest of the standard features used in AF forecast systems.

We then provided the reader with a throughout review of the AF forecast literature, and observed that state of the art methods used rather classical ML algorithms and showcased spectacular results on the AFPDB (e.g. Mohebbi et al [36]).

Our procedure in trying to surpass the performance of the AF forecast system described by Gilon [23] was a two-step process.

Firstly, we implemented a ResNet model inspired from Wang et al. [46]. We evaluated the results for varying depths (i.e. numbers of residual blocks) of the ResNet, and observed that the best networks had the smallest amount of layers. We suspected this was caused by the limited dataset used for training. Moreover, none of the DNNs proved to be as good as Gilon's model.

Secondly, we injected Gilon's RNN and a shallow ResNet with the features we derived earlier from Coumel's Triangle. No performance increase was achieved with the recurrent network but minor improvements were observed on the ResNet. The ResNet improvements were nevertheless still insufficient to outperform the metrics obtained by Gilon's model.

Lastly, we aimed to reproduce existing AF forecast research, in an attempt to (1) understand the used ML techniques to improve our own future work and (2) confirm that the reported results are sound and reliable.

After carefully presenting our (mainly) unsuccessful reproduction attempts, we concluded that the state of the art AF forecast systems were unfortunately not reproducible and we highlighted some recurring patterns in these works.

1. Reporting final evaluation on validation sets.
2. Overfitting the validation set.
3. Not reporting the train-validation split of the dataset.
4. Not providing the exact parameters used for an algorithm.

9.2 Future work

This work could be continued in various ways. Firstly, one could try to understand why augmenting the DNNs with Coumel's features was fruitless. This would require in-depth analysis of the networks' weights and learned features, which concerns the field of Interpretability.

Secondly, given the small size of the publicly available dataset (the AFPDB, 106 recordings) and our own datasets (less than 200 recordings) we acknowledge that using Deep Learning is maybe not the way to go for this problem, as it is known to require sizeable datasets. Hence, a more classical ML approach using various feature selection methods could be undertaken.

Finally, as it was also concluded by Gilon's previous work [23], it is important to remember that this work only had access to RR intervals of ECGs. As it is known that some information regarding AF initiation can be found in the morphology of the P waves, making use of the complete raw ECG signal could also be a future goal.

Bibliography

[1] Computers in cardiology website. http://www.cinc.org/. Accessed: 2021-05-17.

[2] Long short term memory (lstm). https://www.d2l.ai/chapter_recurrent-modern/lstm.html. Accessed: 2021-05-16.

[3] Physionet website. https://physionet.org. Accessed: 2021-05-17.

[4] Paf prediction challenge database. https://physionet.org/content/afpdb/1.0.0/, 2001. Accessed: 2021-05-17.

[5] Predicting paroxysmal atrial fibrillation/flutter - the physionet computing in cardiology challenge 2001. https://physionet.org/content/challenge-2001/1.0.0/, 2001. Accessed: 2021-05-17.

[6] U.R. Acharya, H. Fujita, O.S. Liha, Y. Hagiwaraa, J.H. Tana, and M. Adam. Automated detection of arrhythmias using different intervals of tachycardia ecg segments with convolutional neural network. *Information Sciences*, 405:81–90, 2017.

[7] R.S. Andersen, A. Peimankar, and S. Puthusserypady. A deep learning approach for real-time detection of atrial fibrillation. 115:465–473, 2018.

[8] S.G. Artis, R.G. Mark, and G.B. Moody. Detection of atrial fibrillation using artificial neural networks. In *Proceedings Computers in Cardiology*, pages 173–176, 1991.

[9] Kim E. Barrett, Scott Boitano, Susan M. Barman, and Heddwen L. Brooks. *Ganong's Review of Medical Physiology*. MIT Press, 24th edition, 2012.

[10] Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. A systematic review of reproducibility research in natural language processing, 2021.

[11] K.H. Boon, M. Khalil-Hani, and M.B. Malarvili. Paroxysmal atrial fibrillation prediction based on hrv analysis and non-dominated sorting genetic algorithm iii. *Computer methods and Programs in Biomedicine*, 153:171–184, 2018.

[12] K.H. Boon, M. Khalil-Hani, M.B. Malarvili, and C.W. Sia. Paroxysmal atrial fibrillation prediction method with shorter hrv sequences. *Comput. Methods Programs Biomed.*, 134:187–196, 2016.

[13] Andriy Burkov. *The Hundred-Page Machine Learning Book*. 2019.

[14] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1):16–28, 2014. 40th-year commemorative issue.

[15] P. Chazal and C. Heneghan. Automated assessment of atrial fibrillation. *Computers in Cardiology*, 28:117–120, 2001.

[16] Y.V. Chesnokov. Complexity and spectral analysis of the heart rate variability dynamics for distant prediction of paroxysmal atrial fibrillation with artificial intelligence methods. *Artificial Intelligence in Medicine*, 43:151–165, 2008.

[17] H. Costin, C. Rotaariu, and A. Pasarica. Atrial fibrillation onset prediction using variability of ecg signals. *Advanced Topics in Electrical Engineering*, pages 1–4, 2013.

[18] P. Coumel. The management of clinical arrhythmias. an overview on invasive versus non-invasive electrophysiology. *European Heart Journal*, 8(2):92–99, 1987.

[19] C. Dallet. *Caractérisation locale de la propagation de l'onde d'activation cardiaque pour l'aide au diagnostic des tachycardies atriales et ventriculaires : application a l'imagerie électrocardiographique non-invasive.* PhD thesis, Université de Bordeaux, 2017.

[20] E. Ebrahimzadeh, M. Kalantari, M. Joulani, R.S. Shahraki, F. Fayaz, and F. Ahmadi. Prediction of paroxysmal atrial fibrillation: A machine learning based approach using combined feature vector and mixture of expert classification on hrv signal. *Computer Methods and Programs in Biomedicine*, 165:53–67, 2018.

[21] U. Erdenebayar, H. Kim, J.U. Park, D. Kang, and K.J. Lee. Automatic prediction of atrial fibrillation based on convolutional neural network using a short-term normal electrocardiogram signal. *Journal of Korean Medical Science*, 34(64), 2019.

[22] O. Faust, A. Shenfield, M. Kareem, T.R. San, H. Fujita, and U.R. Acharya. Automated detection of atrial fibrillation using long short-term memory network with rr interval signals. *Computers in Biology and Medicine*, 102:327–335, 2018.

[23] C. Gilon. Paroxysmal atrial fibrillation diagnosis and forecast with deep neural networks, 2019.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* The MIT Press, 2016.

[25] Arthur C. Guyton and John E. Hall. *Textbook of Medical Physiology.* 11th edition, 2006.

[26] Yuki Hagiwara, Hamido Fujita, Shu Lih Oh, Jen Hong Tan, Ru San Tan, Edward J Ciaccio, and U Rajendra Acharya. Computer-aided diagnosis of atrial fibrillation based on ecg signals: A review. *Information Sciences*, 467:99–114, 2018.

[27] A.Y. Hannun, P. Rajpurkar, M. Haghpanahi, C. Bourn, and A.Y. Ng. Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836*, 2017.

[28] B. Hickey and C. Heneghan. Screening for paroxysmal atrial fibrillation using atrial premature contractions and spectral measures. *Computers in Cardiology*, pages 217–220, 2002.

[29] Cheng-Lung Huang and Chieh-Jen Wang. A ga-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with Applications*, 31(2):231–240, 2006.

[30] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324, 1997. Relevance.

[31] C. Kolb, S. Nurnberger, G. Ndrepepa, B. Zrenner, A. Schomic, and C. Schmitt. Modes of initiation of paroxysmal atrial fibrillation from analysis of spontaneously occurring episodes using a 12-lead holter moni- toring system. *Amer. J. Cardiol.*, 88(8):853–857, 2001.

[32] Yann-Aël Le Borgne and Gianluca Bontempi. *Machine Learning for Credit Card Fraud Detection - Practical Handbook*. Université Libre de Bruxelles, 2021.

[33] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[34] Malik M. Geometric methods for heart rate variability assessment. In Futura Pub. Co. Inc., editor, *Heart Rate Variability*.

[35] Marcia McNutt. Journals unite for reproducibility. *Science*, 346(6210):679–679, 2014.

[36] M. Mohebbi and H. Ghassemian. Prediction of paroxysmal atrial fibrillation based on non-linear analysis and spectrum and bispectrum features of the heart rate variability signal. *Computer methods and Programs in Biomedicine*, 105:40–49, 2012.

[37] A. Narin, Y. Isler, M. Ozer, and M. Perc. Early prediction of paaroxysmal atrial fibrillation based on short term heart rate variability. *Physica A*, 509:56–65, 2018.

[38] Task Force of the European Society of Cardiology the North American Society of Pacing Electrophysiology. Heart rate variability: Standards of measurement, physiological interpretation, and clinical use. *Circulation*, 93(5), 1996.

[39] Babatunde Kazeem Olorisade, Pearl Brereton, and Peter Andras. Reproducibility of studies on text mining for citation screening in systematic reviews: Evaluation and checklist. *Journal of Biomedical Informatics*, 73:1–13, 2017.

[40] J. Pan and W.J. Tompkins. A real time qrs detection algorithm. *IEEE Transactions on Biomedical Engineering*, BME-32(3):230–236, 1985.

[41] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Lariviere, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program), 2020.

[42] I. Pinhas, E. Toledo, D. Aravot, and S. Akselrod. Bicoherence analysis of new cardiovascular spectral components observed in heart-transplant patients: statistical approach for bicoherence thresholding. *IEEE Trans. Biomed. Eng.*, 51(10):1774–1783, 2004.

[43] Salcedo-Sanz S., Prado-Cumplido M., Perez-Cruz F., and Bousono-Calzon C. Feature selection via genetic optimization. In *Proceedings of the ICANN international conference on artificial neural networks*, pages 547–552, 2002.

[44] Ananthram Swami. Hosa - higher order spectral analysis toolbox. `https://nl.mathworks.com/matlabcentral/fileexchange/3013-hosa-higher-order-spectral-analysis-toolbox`. Accessed: 2021-05-22.

[45] T. Thong, J. McNames, M. Aboy, and B. Goldstein. Prediction of paroxysmal atrial fibrillation by analysis of atrial premature complexes. *IEEE Transactions on Biomedical Engineering*, 51:561–569, 2004.

[46] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline, 2016.

[47] van der Molen MW Weber EJ, Molenaar PC. A nonstationarity test for the spectral analysis of physiological time series with an application to respiratory sinus arrhythmia. *Psychophysiology*, 29(1):55–65, 1992.

[48] S.N. Yu and M.Y. Lee. Bispectral analysis and genetic algorithm for congestive heart failure recognition based on heart rate variability. *Comput. Biol. Med.*, 42(8):816–825, 2012.

[49] S.-M. Zhou, J.Q. Gan, and F. Sepulveda. Classifying mental tasks based on features of higher-order statistics from eeg signals in brain-computer interface. *Inf. Sci.*, 178(6):1629–1640, 2008.

[50] W. Zong, R. Mukkamala, and R.G. Mark. A methodology for predicting paroxysmal atrial fibrillation based on ecg arrhythmia feature analysis. *Computers in Cardiology*, 28:125–128, 2001.