POLITECNICO DI TORINO

Master's Degree in Mathematical Engineering

Master's Degree Thesis

Synthesis of realistic human heart-beats with time-series forecasting techniques and arrhythmia classification using Wavelet Scattering Transform

Supervisors
Prof. Lamberto Rondoni
Dr. Davide Carbone

Candidate
Matteo Scanu

Academic Year 2024-2025

Contents

Introduction

Cardiovascular diseases are the first cause of death, globally [22]. In 2019 there has been more than 18 million people that died due to disease of this kind, and that more than half of them (around ten millions) had high-blood pressure, which causes human hearth to pump blood in a different and irregular way: this last phenomenon is called arrhythmia.

Despite that death rates for this kind of diseases have dramatically decreased in the last decades due to the technology advancement, and a very popular and effective improvement is tied to the use of computational tools to try and find different classes of arrhythmias. Cardiologist are formed for around ten years of their life in order to understand and be able to distinguish one type of arrhythmia from another just by reading the electrocardiogram (ECG) of the hearth-beat considered. ECG is a pivotal technique in medicine, which allows for a graphical representation of an hearth-beat in the domain of time. Each point describes the electric potential of the cardiac muscle. The signal is sampled by a certain frequency, so that each point represent one single instant; for example, if the signal is sampled at 360 Hz, it means that each point represents $\frac{1}{360} \approx 0.0028$ seconds, so that we need 360 sample points in the graph in order to have the graphic representation of one second of said heart-beat.

It is clear from this description that through ECG it is possible to have access for an unbelievable amount of data describing just a minute of an heart beating; consequently, working on big amount of data con become very strenuous even for the hardest-working cardiologist, and due to human nature, this increases the probability of mistakes in a field where even the smallest one can cause irreparable damage. This has been clear as the sheer computational power of calculators improved over the decades, with the realization that using a calculator would have increased the accuracy of the prediction, with a percentage error very small with respect to the one we could have had from the doctors not because of their faulty formation, but because of other exogenous motives (tiredness, distractions, etcetera).

This represents a classic supervised classification problem: a mathematical classifier has to label correctly data that hasn't seen after having been trained on

labelled data. This process rarely can be carried out with raw data, because some features needs to be extracted: this is the reason why in a lot of similar works a data transformation is used, and this is no exception. However, in order to achieve this objective, it is necessary to go one step at the time.

Part I of this work is all about that: transformation of signals using a mathematical transform in order to effectively extract features. It is a derivation of Fourier transform (so it is useful in order to work on the frequency-domain of the signal), but has a lot of good mathematical properties which allow the signal to be invariant to translations and scaling: its name is Wavelet Scattering Transform, and all its components will be analysed piece by piece. Fourier transform will be taken as the starting point in chapter 1, and from that a new transform, based on wavelet, will be not only defined, but deeply analysed in order to search for all its mathematical features and properties: all of that in chapter 2. In chapter 3 another integral operator will be considered: the scattering propagator, which will become scattering transform: that is the key that will allow, in chapter 4, to create the Wavelet Scattering Transform, with all its good mathematical properties proved. Finally, in chapter 5, all mathematical transform will be confronted, in order to assure WST superiority in this field.

Another problem when analysing cardiac data is the dramatic shortage of it, especially when treating abnormal beats with some kind of arrhythmia. Acquiring heart-beats is very expensive both economically and computationally, so it would be ideal to find a new way that allow to produce great amounts of abnormal beats in a relatively short amount of time. The idea is then to use a very known technique, time-series forecasting, and create beats by using it. Time-series forecasting is usually used to predict what is going to happen in the future, assuming that all previous hypothesis remain true. Since heart-beats are pretty regular for a certain person, the idea is to use this technique in order to create new type of a certain beat, recreating abnormalities and features of that kind of cardiac disease; at the same type these beats have to be original, in order that no privacy concerns emerge.

This is possible because every kind of signal can be seen as a time series, and ECGs represent no exception. As a consequence, it is possible to train a mathematical model on some data, learning the characteristics of each kind of heart-beat, and after that, it is possible to test it on data that wasn't used in the training, so that the model has to label each heart-beat itself. In the end we can compute the percentage error of the model using the real labels of the test part of the dataset: this is, in short, a statistical learning process.

Part II of this work analyses every important aspect of time-series and time-series forecasting: to be more precise, these matters are tackled in chapter 6 and 7, respectively. In the latter classic forecasting techniques will be described, such as ETS and ARIMA: they will be used to create new heart-beats. Then, in chapter 8 new models are examinated, that permit to enter the world of machine learning

and deep learning: in particular, two famous and very popular neural models, Long-Short Term Memory (LSTM) and Recurrent Neural Network (RNN), will be used as an attempt to improve results, and show how effective and powerful they can be in such complex tasks.

Finally, in part III the experiment is described in all its detail, especially in chapter 9, while in chapter 10 all results obtained are gathered, rearranged and showed through tables and images.

Code used for this work is available at the following link: https://github.com/MatteoScanuPolito/WST_Classification_Forecasting_ECG_Generation.

Part I

Wavelet Scattering Transform

The main core of this thesis work is discussing a tool that is becoming more and more popular in signal processing tasks: it is called Wavelet Scattering Transform (WST). The name comes from the fusion of the ideas of wavelet transform and scattering propagator, and it has a number of very good properties that allows for an excellent feature extraction when working on time-dependent data, like ECG. The idea of wavelet transform is definitely not new, and has been formally introduced by Grossman and Morlet in 1984 [33]. Morlet was working on the research of hydrocarbons through reflection seismology, when he realized that the modulated pulses sent underground have such a long duration high frequencies that it is impossible to separate the returns of close layers. Instead of emitting pulses of equal duration, he thought of sending shorter waveforms at high frequencies. Such waveforms are simply obtained by scaling a single function called mother wavelet. Grossmann noticed how close Morlet's idea was to his own work, and they started collaborating with each-other, triggering a cascade of scientists that started working on it too.

This part is organized as follows. The main point is to show all the properties of WST, in particular the fact that it is one of the few transform that can represent a signal in the frequency domain and that is invariant for translations and deformations; note that it is not necessary to be invariant to all kind of deformations, but only to certain ones, which allow a transform to be Lipschitz continuous; more on that later.

In order to do that, it was decided to take a long and winding road from the beginning of signal analysis, starting with a short recap on Fourier transform in chapter 1. Note that only stuff strictly related to this work was considered, so there is no consideration of Fourier series or the discrete version of Fourier transform; a lot of other books treat those sections in a deepened fashion. Besides the only focus of this work is to consider the case if signal f is continuous on its temporal and frequency domain.

At the end of the chapter there is a proof that show some of the problems of Fourier transform when it comes to modify the original signal by translating it or by deform it in any way, but some interesting alternatives start to be proposed, like Fourier transform with a modulus. The idea results too weak, but it is an interesting idea that will be recovered later on.

Chapter number 2 is the one which introduce and study a very unique type of functions, called wavelets because of the affinity of their graph to a little wave. They are created by translating and deforming one function which is positive, zero-average and respects another existence condition.

Its convolution with the signal gives the Wavelet transform, which solves one of our problems, as it is invariant to deformations, but not to translations; it has a similar property, as it is shift-invariant, which does not resolve the problem still.

More analysis was considered to be important, as wavelets can create orthonormal basis which yield the existence of a scaling function, that will become fundamental in later chapters; for this reason it was decided to comment on some properties for wavelet basis, as it is also the main tool that allow to approximate the signal when transformation is on.

Finally, some comments were made on its regularity and on its behaviour when interval considered were very small. This was decided because when signal are analysed, most of the informations are concentrated in singularity points with irregular structures; from this point of view, ECG are a hugely important example, and it has to be contemplated.

In chapter 3 a new landmark is defined: the scattering propagator, introduced by Mallat in [43], is the main tool that allows to define wavelet scattering transform. It is basically just a cascade of convolutions and modulus, and by itself is neither translation invariant nor Lipschitz-continuous. However, by adding the integral on its domain and averaging with a scaling function, both of those properties are respected, so that the WST can be defined and its properties demonstrated in chapter 4.

Chapter 5 is just a short recap on the properties of each transform analysed, with table 5.1 to keep everything clear.

Chapter 1

Fourier transform

1.1 Definition and properties

Wavelet Scattering Transforms find their roots from the Fourier transform, probably the most important concept in time-series analysis. It allows to convert a time-domain function $f(t)$ into its corresponding frequency-domain one $\widehat{f}(\omega)$; in other words, while the first one give information on the value of f in each time instant, the second one measures how much oscillations at each frequency ω there is in f.

Definition 1.1.1. The **Fourier transform** of a time-domain function $f(t)$ is:

$$\widehat{f}(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-i\omega t} dt. \tag{1.1}$$

Before going ahead, let's list some of the main properties of Fourier transform, as it will be widely used throughout this work.

1.1.1 Base properties

Property 1.1.2. Consider a function $f \in L^2(\mathbb{R})$. Then the following are valid:

1. scaling (or dilation) property:

$$\widehat{f}(at) = \frac{1}{|a|} \widehat{f}\left(\frac{\omega}{a}\right), a \neq 0; \tag{1.2}$$

2. time shift, which yields frequency modulation:

$$\widehat{f}(t-b) = e^{-i\omega b} \widehat{f}(\omega); \tag{1.3}$$

3. exponential modulation, which yields frequency shift:

$$\widehat{e^{iat}f(t)} = \widehat{f}(\omega - a). \tag{1.4}$$

Proof. These properties are quite easy to prove.

1. Putting $u = at$:

$$\begin{aligned}\widehat{f}(at) &= \int_{-\infty}^{+\infty} f(at)e^{-i\omega t}dt \\ &= \frac{1}{|a|}\int_{-\infty}^{+\infty} f(u)\exp\left(-i\omega\frac{u}{a}\right)du \\ &= \frac{1}{|a|}\widehat{f}\left(\frac{\omega}{a}\right).\end{aligned}$$

2. Putting $t - b = u$:

$$\begin{aligned}\widehat{f}(t-b) &= \int_{-\infty}^{+\infty} f(t-b)e^{-i\omega t}dt \\ &= \int_{-\infty}^{+\infty} f(u)e^{-i\omega(u+b)}du \\ &= \int_{-\infty}^{+\infty} f(u)e^{-i\omega t}e^{-i\omega b}dt \\ &= e^{-i\omega b}\int_{-\infty}^{+\infty} f(u)e^{-i\omega t}dt \\ &= e^{-i\omega b}\widehat{f}(\omega).\end{aligned}$$

3. We just have to compute the integral explicitly to show that:

$$\begin{aligned}\widehat{e^{iat}f(t)} &= \int_{-\infty}^{+\infty} f(t)e^{iat}e^{-i\omega t}dt \\ &= \int_{-\infty}^{+\infty} f(t)e^{-i(\omega-a)t}dt \\ &= \widehat{f}(\omega - a).\end{aligned}$$

□

In the context of our analysis it's important to deeply analyse this transform, as it will be fundamental to understand and to explain why it was necessary to introduce the scattering transform and why does the latter improves the results obtained with Fourier. For this reasons, it has to be determined if Fourier transform is a bounded operator, because that would mean that it is continuous too. These two concepts are equivalent for operators, as stated by equation (A.1.4), whose proof and statement are in appendix A.

Now it is clear, because of the properties of integrals, that Fourier transform is a linear operator: but is it bounded? In order to show this, it is necessary to state a massively important results in functional analysis.

1.1.2 Generalized Riemann-Lebesgue Lemma and its validity for Fourier transform

Lemma 1.1.3. Consider a function $f \in L^1[a,b], -\infty \leq a \leq b \leq +\infty$ and a bounded function h defined on $\mathbb{R}$; if the following averaging condition holds

$$\lim_{|c|\to+\infty} \frac{1}{c}\int_0^c h(t)dt = 0, \tag{1.5}$$

then the **generalized Riemann-Lebesgue Lemma** states that:

$$\lim_{\omega\to+\infty} \int_a^b f(t)h(\omega t)dt = 0. \tag{1.6}$$

Proof. Consider $f(t) = \mathbb{1}_{[a,b]}$ the indicator function of set $[c,d] \subset [0,+\infty)$; then:

$$\int_0^{+\infty} \mathbb{1}_{[a,b]}h(\omega t)dt = \int_a^b h(\omega t)dt$$

and after substituting $x = \omega t$ we have:

$$\int_a^b h(\omega t)dt = \frac{1}{\omega}\left(\int_0^{b\omega} h(x)dx - \int_0^{a\omega} h(x)dx\right).$$

Because of the averaging condition this quantity goes to 0 if $\omega \to +\infty$, which means that for linearity the following holds for any step function g:

$$\lim_{\omega\to+\infty} \int_0^{+\infty} g(t)h(\omega t)dt = 0.$$

Since step function are dense in $L^1(\mathbb{R})$, if $|h| \leq C$, then for every $f \in L^1(\mathbb{R})$ and $\forall\ \varepsilon \in \mathbb{R}^+$, then exists a step function g such that

$$\int_0^{+\infty} |f(t) - g(t)|dt \leq \|f - g\|_1 < \frac{\varepsilon}{2C}.$$

If ω is sufficiently large the following is obtained

$$\left|\int_0^{+\infty} g(t)h(\omega t)dt\right| < \frac{\varepsilon}{2},$$

which yields:

$$\begin{aligned}\left|\int_0^{+\infty} f(t)h(\omega t)dt\right| &\leq \int_0^{+\infty} |f(t) - g(t)||h(\omega t)| + \left|\int_0^{+\infty} g(t)h(\omega t)dt\right| \\ &\leq \frac{\varepsilon}{2C}C + \frac{\varepsilon}{2} = \varepsilon.\end{aligned}$$

The same can be applied if integration is done in the set $(-\infty, 0]$. □

Considering our case, since:

$$\lim_{|c|\to+\infty} \frac{1}{c}\int_0^c e^{-it}dt = \lim_{|c|\to+\infty} \frac{1}{c}(1-e^{-ic}),$$

from which it is possible to have an estimate of the modulus:

$$\begin{aligned}\lim_{|c|\to+\infty} \frac{1}{|c|}|1-e^{-ic}| &= \lim_{|c|\to+\infty} \frac{1}{|c|}|1+e^{-ic}| \\ &\leq \lim_{|c|\to+\infty} \frac{1}{|c|}(|1|+|e^{-ic}|) \\ &= \lim_{|c|\to+\infty} \frac{2}{|c|} = 0,\end{aligned}$$

because e^{-it} is a point in the circumference of radius $r = 1$ in the complex plane $\forall\ t \in \mathbb{R}$, which means that $|e^{-it}| = 1$, so our point is proven.

That means that the Fourier transform is bounded, and so it is continuous. This will be a very useful fact in the continuation of this work.

1.1.3 Invertibility of Fourier transform

Observation 1.1.4. If $f \in L^1(\mathbb{R})$:

$$|\widehat{f}(\omega)| < \int_{-\infty}^{+\infty} |f(t)|dt < +\infty, \tag{1.7}$$

which means that $f \in L^1(\mathbb{R})$, too.

Definition 1.1.5. Since it will be widely used in this work, let's remind of the concept of **convolution** of two function $f, g \in L^2(\mathbb{R})$:

$$f \star g(t) = \int_{-\infty}^{+\infty} f(t-x)g(x)dx = \int_{-\infty}^{+\infty} f(x)g(t-x)dx.$$

Because of linearity of the integral, convolutions of functions are linear, too.

Observation 1.1.6. Thanks to Fubini's theorem, if $f, g \in L^1(\mathbb{R}) \Rightarrow f \star g \in L^1(\mathbb{R})$. This is easy to prove:

$$\begin{aligned}\|f \star g\|_1 &= \int_{-\infty}^{+\infty} |f \star g\ (t)|dt \\ &= \int_{-\infty}^{+\infty} \left|\int_{-\infty}^{+\infty} f(t-x)g(x)dx\right| dt \\ &\leq \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} |f(t-x)g(x)|dxdt \\ &= \int_{-\infty}^{+\infty} |g(x)| \int_{-\infty}^{+\infty} |f(t-x)|dtdx \\ &= \|g\|_1\|f\|_1 < +\infty.\end{aligned}$$

Theorem 1.1.7. If $f, \widehat{f} \in L^1(\mathbb{R})$, then the **inverse Fourier theorem** affirms the following:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \widehat{f}(\omega) e^{i\omega t} d\omega. \tag{1.8}$$

Equivalently, it can be said that if hypothesis are verified, then Fourier transformation is invertible, and the inverse is continuous.

Proof. Given $t > 0, x \in \mathbb{R}^n, \phi(\omega) = \exp(i\omega x - \pi t^2 |\omega|^2)$, its Fourier transform is the following:

$$\widehat{\phi}(y) = t^{-n} \exp\left(\frac{-\pi |x-y|^2}{t^2}\right) =: g_t(x-y),$$

with $g_t(x) = t^{-n} \exp(\frac{-\pi|x|^2}{t^2})$. Then thanks to a lemma in [31]:

$$\int e^{-\pi t^2 |\omega|^2} e^{i\omega x} \widehat{f}(\omega) d\omega = \int \widehat{f} g_t = \int f \widehat{g}_t = f \star g_t(x).$$

Since

$$\int e^{-\pi |x|^2} dx = 1$$

and because f is bounded and uniformly continuous, then $f \star g_t \underset{t \to 0}{\longrightarrow} f$. Meanwhile $\widehat{f} \in L^1$, and for the dominated convergence theorem:

$$\lim_{t \to 0} \int e^{-\pi t^2 |\omega|^2} e^{i\omega x} \widehat{f}(\omega) d\omega = \int e^{i\omega x} \widehat{f}(\omega) d\omega = \mathcal{F}(\mathcal{F}(f)),$$

with $\mathcal{F}(f) = \widehat{f}$ being another notation to denote the Fourier transform. So it was proved that $f = \mathcal{F}(\mathcal{F}(f))$, and they are both continuous: so the proof is complete. □

1.2 Sensitivity to translations and deformations

One of the main points in signal analysis is predicting the behaviour of its transformed version; for example, in this work it will be fundamentally important to understand if a certain transform is able to make a certain signal invariant to translation.

Definition 1.2.1. Let's define $L_c f(t)$ as a translation operator, which means simply that $L_c f(t) = f(t-c), c \in \mathbb{R}$. Then, an operator $\phi \in L^2(\mathbb{R}^n)$ is **translation invariant** if

$$\phi(L_c f) = \phi(f), \ \forall \ f \in L^2(\mathbb{R}^n). \tag{1.9}$$

Fourier transform are not translation invariant (it's trivial), but if one adds a modulus to it, then it is: more on that on chapter 3. In fact, if $\widehat{x}(\omega)$ is the Fourier transform of a signal $x(t) \in L^2(\mathbb{R}^d)$, then:

$$|\widehat{x}(\omega)| = |\widehat{L_c x}(\omega)|, \ \forall \ c \in \mathbb{R}; \tag{1.10}$$

this will be one of the tools that will be used in order to introduce scattering transform in 3.

At the same time, it is quite clear that Fourier transform, despite being fundamental for opening a new field of mathematical study, have another very big downside: they are not invariant to deformations (even the smaller one) at high frequencies.

To show it, consider the following linear operator $\tau(x(t)) = sx(t), |s| < 1$ and then consider $x(t) = e^{i\omega t}\theta(t)$, which is the modulated version of a low-pass filter $\theta(t)$. Then, if we consider the dilation $x_\tau(t) := x(t - ts)$, frequencies are moved from ω to $(1 - s)\omega$; in other words, the frequency center of the transform is translated.

Then, if we define:

$$\sigma_\theta^2 = \int |\omega|^2 |\theta(\omega)|^2 d\omega$$

as the frequency spread of function θ, then the following is valid:

$$\sigma_x^2 = \int |\omega - \xi|^2 ||\widehat{x}(\omega)|^2 d\omega = \sigma_\theta^2$$

and

$$\begin{aligned}
\sigma_{x_\tau}^2 &= \frac{1}{(s-1)^d} \int (\omega - (1-s)\xi)^2 \left| \widehat{x}\left(\frac{\omega}{1-s}\right) \right|^2 d\omega \\
&= \int |(1-s)(\omega - \xi)|^2 |\widehat{x}(\omega)|^2 d\omega \\
&= (1-s)^2 \sigma_x^2,
\end{aligned}$$

which yields that the distances between the central frequencies of x and x_τ, equal to $s\xi$, is large compared to their frequency spread; this means that $\widehat{x}$ and $\widehat{x}_\tau$ have disjoint supports and that

$$\| |\widehat{x}_\tau| - |\widehat{x}| \| \approx \|x\| \approx |s||\xi|\|\theta\|,$$

which can be arbitrarily big because of ξ, which means that Fourier transform are not Lipschitz continuous (i.e. stable) to deformation, as said before.

The importance of Fourier transform can not be underestimated: most of the modern techniques for analysing signals and performing feature extraction comes from that idea, and the wavelet scattering transform is no exception.

Despite that, it is clear that in order to work precisely with signals, it is necessary that another mathematical operator is introduced: this is one of the reasons wavelet transforms were first introduced.

Chapter 2

Wavelet transform

2.1 Definition and existence conditions

First of all: what is a wavelet? In order to define it, let's consider a zero-average function $\psi \in L^2(\mathbb{R})$, called the mother wavelet; so a function such that

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0. \tag{2.1}$$

Definition 2.1.1. Starting from it, we call **wavelets** all the function obtained by translating and scaling of the mother wavelet ψ;

$$\psi_{a,b}(t) = |a|^{-\frac{1}{p}} \psi\left(\frac{t-b}{a}\right), p > 0,\ a, b \in \mathbb{R},\ a \neq 0. \tag{2.2}$$

Note that in this work only one case will be covered, with $p = 2$.

Observation 2.1.2. The mother function is normalized, so $\|\psi\| = 1$; this yields that each function generated from that is normalized as well: $\|\psi_{a,b}\| = 1$.

Definition 2.1.3. If $f \in L^2(\mathbb{R})$, its **wavelet transform** is defined as follows:

$$Wf(a,b) = \langle f, \psi_{a,b} \rangle = \int_{-\infty}^{+\infty} f(t) \frac{1}{\sqrt{a}} \psi^* \left(\frac{t-b}{a}\right) dt, \tag{2.3}$$

where $\psi^*(t)$ is the conjugate complex of $\psi(t)$.

Basically the wavelet transform take as input a function $f(t)$ and filters it with a scaled and translated function $\psi_{a,b}(t)$. The above definition can be interpreted as a convolutions of two functions, since $Wf(a,b) = f \star \bar{\psi}_a(b)$, with:

$$\bar{\psi}_a(b) = \frac{1}{\sqrt{a}} \psi^* \left(\frac{-b}{a}\right).$$

So it is clear that Fourier transform and the wavelet transform have a lot in common, and have both the same aim: measuring the time evolution of each frequency. This is accomplished by using a complex analytic wavelet, that is able to separate amplitude and phase components, while real wavelets $Wf(a,b)$ measure the variation of f near a point b, and can be used to isolate peaks or discontinuities; the size of the neighbourhood is proportional to a.

In particular, the bigger the value of a the bigger the time interval that is analysed; this comes at the cost that high-frequencies are basically not considered. On the other hand, when $a \to 0$ the time interval is very small, and higher frequencies can be seen by the filter: this case will be analysed in section 2.4.

Not every function in $L^2(\mathbb{R})$ with zero average can be used to define a family of wavelets; there has to be at least one more condition, thanks to which the following theorem is proven right.

Theorem 2.1.4. Let $\psi \in L^2(\mathbb{R})$ be a real function such that the following condition, called the **wavelet admissibility condition**,

$$C_\psi = \int_0^{+\infty} \frac{|\widehat{\psi}(\omega)|^2}{\omega} d\omega < +\infty \tag{2.4}$$

is valid. Then any $f \in L^2(\mathbb{R})$ satisfies the following equation:

$$f(t) = \frac{1}{C_\psi} \int_0^{+\infty} \left(\int_{-\infty}^{+\infty} \frac{1}{a^2} Wf(a,b) \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) db \right) da; \tag{2.5}$$

Proof. Let us define

$$\begin{aligned} w(t) :&= \frac{1}{C_\psi} \int_0^{+\infty} \frac{1}{a^2} Wf(a,\cdot) \star \psi_a(t) da; \\ &= \frac{1}{C_\psi} \int_0^{+\infty} \frac{1}{a^2} f \star \bar{\psi_a} \star \psi_a(t) da \end{aligned}$$

to prove our point it is sufficient to show that $w = f$ by showing that their Fourier transform are equal (since each function has its unique Fourier transform).

So:

$$\begin{aligned} \hat{w}(\omega) &= \frac{1}{C_\psi} \int_0^{+\infty} \frac{1}{a^2} \hat{f}(\omega) \sqrt{a} \hat{\psi}^*(a\omega) \sqrt{a} \hat{\psi}^*(a\omega) da \\ &= \frac{1}{C_\psi} \hat{f}(\omega) \int_0^{+\infty} \frac{1}{a} |\hat{\psi}(a\omega)|^2 da. \end{aligned}$$

Changing variable to $\xi = a\omega$ gives:

$$\hat{w}(\omega) = \frac{1}{C_\psi} \hat{f}(\omega) \int_0^{+\infty} \frac{|\hat{\psi}(\xi)|^2}{\omega} d\omega = \hat{f}(\omega).$$

so the thesis has been proven. □

Theorem 2.5 has a very important meaning; it guarantees that, if Wf is know, it is possible (even if very hard from an analytic point of view) to recover the original function f from it. In other words, wavelet transform is invertible, a fact that can be very exciting for the future prospect of research in signal analysis and generation.

Imposition for mother wavelet to be a zero average function was done so that $\widehat{\psi}(0) = 0$ in order to guarantee that this integral is finite. Besides, if $\widehat{\psi}(0) = 0$ and $\widehat{\psi}(\omega)$ is continuously differentiable then the admissibility condition is satisfied.

Theorem 2.1.5. If $f, g \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, then the **Parseval formula** affirms that:

$$\int_{-\infty}^{+\infty} f(t)g^*(t)dt = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \widehat{f}(\omega)\widehat{g}^*(\omega)d\omega \tag{2.6}$$

Observation 2.1.6. Because equation (2.6) holds, the following is also valid:

$$Wf(a,b) = \int_{-\infty}^{+\infty} f(t)\psi^*_{a,b}(t)dt = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \widehat{f}(\omega)\widehat{\psi}^*_{a,b}(\omega)d\omega, \tag{2.7}$$

so the same coefficient can be obtained through time or frequency integration.

If $f = g$, (2.6) changes into the Plancharel formula.

Theorem 2.1.7. If $f \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, then the **Plancherel formula** affirms the following:

$$\int_{-\infty}^{+\infty} |f(t)|^2 dt = \frac{1}{2\pi}\int_{-\infty}^{+\infty} |\widehat{f}(\omega)|^2 d\omega. \tag{2.8}$$

The scaling function ϕ can be seen as the response of a low-pass filter. The low-frequency approximation of f at the scale a is:

$$\left\langle f(t), \frac{1}{\sqrt{a}}\phi\left(\frac{t-b}{a}\right)\right\rangle = f \star \bar{\phi}_a(b). \tag{2.9}$$

That is basically the main operation that is made when transforming a signal, and results may vary a lot dependently to the scaling function: for example we are using the Gabor wavelets, which uses a Gaussian function as the low-pass filter (which is the scaling function).

2.2 Core properties

The whole point of introducing wavelet transform is to take advantage of the good mathematical features of it, which allows, in signal analysis, to extract information in a more efficient and effective way, improving classification results while taking less time doing it. We are now going to talk about them, using the same notation

as in [63]: from now on consider a wavelet transform $Wf(a,b)$ of a time-domain function $f(t)$.

First of all, $Wf(a,b)$ is linear, and the proof is trivial, because the inner product between two functions is linear. Besides that, it is useful to compute now the Fourier transform of a wavelet, since it will be used later in some proofs.

Property 2.2.1. Consider a wavelet $\psi_{a,b}(t)$. Then its Fourier transform is equal to:

$$\widehat{\psi}_{a,b}(\omega) = e^{-i\omega b}\sqrt{a}\widehat{\psi}(a\omega), \tag{2.10}$$

with of course $\widehat{\psi}$ being the Fourier transform of the mother wavelet ψ.

Property 2.2.2. Consider a function $f'(t)$ defined as a translation of $f(t)$, i.e. $f'(t) = f(t-b'), b' \in \mathbb{R}$; then

$$Wf'(a,b) = Wf(a, b-b') \tag{2.11}$$

This is called the **shift property**. Proving it is quite simple because:

$$\begin{aligned} Wf'(a,b) &= \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} f'(t)\psi^*\left(\frac{t-b}{a}\right)dt \\ &= \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} f(t-b')\psi^*\left(\frac{t-b}{a}\right)dt \\ &= \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} f(t')\psi^*\left(\frac{t'+b'-b}{a}\right)dt' = Wf(a,b-b'). \qquad \square \end{aligned}$$

It is important to note that this property is not equal and does not imply translation invariance for any operator; in fact it is clear that when translating a signal $f(t)$ by a certain amount b', its wavelet transform does change (in fact parameter b changes to $b-b'$). This will be the most important reason that brought to the introduction of scattering transform.

Property 2.2.3. Consider this time a function $f'(t)$ defined as dilation of f; mathematically $f'(t) = \frac{1}{\sqrt{s}}f\left(\frac{t}{s}\right)$. In that case, the **scaling property** is true:

$$Wf'(a,b) = Wf\left(\frac{a}{s}, \frac{b}{s}\right) \tag{2.12}$$

In fact:

$$\begin{aligned} Wf'(a,b) &= \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} f'(t)\psi^*\left(\frac{t-b}{a}\right)dt \\ &= \frac{1}{\sqrt{sa}}\int_{-\infty}^{+\infty} f\left(\frac{t}{s}\right)\psi^*\left(\frac{t-b}{a}\right)dt \\ &= \sqrt{\frac{s}{a}}\int_{-\infty}^{+\infty} f(t')\psi^*\left(\frac{st'-b}{a}\right)dt' = Wf\left(\frac{a}{s}, \frac{b}{s}\right). \qquad \square \end{aligned}$$

The two properties above are the fundamental ones, and they basically allow wavelet transform (and WST too) to be so interesting when analysing data because it means that the transform of a signal will be the same even if it was scaled or shifted beforehand. These are the first big advantages that WST holds versus Fourier transform, since the latter are very sensible to any kind of deformation, as explained in chapter 1.

Wavelet transform also has an energy conservation property, which will be enounced with the following proposition.

Proposition 2.2.4. Given $f \in L^2(\mathbb{R})$ and its wavelet transform $Wf(a,b)$, the following holds:

$$\int_{-\infty}^{+\infty} |f(t)|^2 dt = \frac{1}{C_\psi} \int_0^{+\infty} \int_{-\infty}^{+\infty} \frac{|Wf(a,b)|^2}{a^2} da\ db. \tag{2.13}$$

Proof. Let's start by applying Parseval formula to Wavelet transform, and by this is what we obtain:

$$\int_{\mathbb{R}^2} \frac{1}{a^2} |Wf(a,b)|^2 dadb = \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} \left| \frac{\sqrt{a}}{2\pi} \int_{-\infty}^{+\infty} \widehat{f}(\omega) \widehat{\psi}^*_{a,b}(a\omega) e^{ib\omega} d\omega \right| db \right) \frac{da}{a^2}.$$

By calling

$$P(\omega) = \widehat{f}(\omega)\widehat{\psi}^*_{a,b}(a\omega),\ p(b) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} P(\omega) e^{ib\omega} d\omega$$

we obtain that:

$$\begin{aligned} \int_{\mathbb{R}^2} \frac{1}{a^2} |Wf(a,b)|^2 dadb &= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} |p(b)|^2 db \right) \frac{da}{|a|} \\ &= \int_{-\infty}^{+\infty} \left(\frac{1}{2\pi} \int_{-\infty}^{+\infty} |P(\omega)|^2 d\omega \right) \frac{da}{|a|}, \end{aligned} \tag{2.14}$$

since $|e^{ib\omega}| = 1$, as already stated.

By using Parseval formula once again, we have:

$$\int_{-\infty}^{+\infty} \left(\frac{1}{2\pi} \int_{-\infty}^{+\infty} |\widehat{f}(\omega)|^2 |\widehat{\psi}^*_{a,b}| d\omega \right) \frac{da}{|a|} = \frac{1}{2\pi} \int_{-\infty}^{+\infty} |\widehat{f}(\omega)|^2 \int_{-\infty}^{+\infty} \frac{|\psi(a\omega)|^2}{|a|} dad\omega.$$

Evidently from (2.4),

$$\int_{-\infty}^{+\infty} \frac{|\psi(a\omega)|^2}{|a|} da = C_\psi.$$

So:

$$\begin{aligned} \frac{1}{C_\psi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \frac{Wf(a,b)}{a^2} dadb &= \frac{1}{C_\psi} \frac{C_\psi}{2\pi} \int_{-\infty}^{+\infty} |\widehat{f}(\omega)|^2 d\omega \\ &= \int_{-\infty}^{+\infty} |f(t)|^2 dt, \end{aligned}$$

which proves the theorem. □

It was already stated that working on sharp time structures was one of the main goals of wavelet transforms, since it can help on time localization of signal. In that sense, it is useful to compute the continuous transform of a Dirac's delta centered in t_0 is:

$$W\delta(a,b) = \frac{1}{\sqrt{a}} \int \psi\left(\frac{t-b}{a}\right) \delta(t-t_0)dt = \frac{1}{\sqrt{a}}\psi\left(\frac{t_0-b}{a}\right) \tag{2.15}$$

It is clear that for small values of a, wavelet transforms allows for a very good localization.

2.3 Wavelet basis

Wavelet transform is a two dimensional representation of a one dimensional signal $f(t)$. This could lead to some redundancy as values for both parameters are modified; consider only a sample of values might be a solution, and completely eliminating redundancy is equivalent to build a basis of the space where signal has to be represented; for more info, check appendix B.

In a certain sense wavelets already form a basis, because starting from the mother wavelet and through all its possible dilations and translation, it is possible to obtain every other function in $L^2(\mathbb{R})$; there are surely alternatives anyway, since wavelets do not guarantee orthogonality (they all have unitary norm; so if orthogonality holds, then the basis is surely orthonormal).

In order to define wavelet basis in general, it is necessary to define what a multiresolution analysis is; reasons for this definition will be given afterwards.

Definition 2.3.1. Consider a sequence of closed subspaces $(V_j)_{j\in\mathbb{Z}}$ s.t. $V_j \subset L^2(\mathbb{R})\ \forall\ j \in \mathbb{Z}$; this is called **multiresolution analysis** (MRA) if the following properties hold:

1. $\cdots \subset V_{-1} \subset V_0 \subset V_1 \subset \ldots$;
2. $\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R})$;
3. $\bigcup_{j\in\mathbb{Z}} V_j = \{0\}$;
4. $f(t) \in V_j \Leftrightarrow f(2t) \in V_{j+1}$;
5. $\exists\ \phi \in V_0$ such that the set of the functions obtained by integer translations of it $\{\phi(t-n) \mid n \in \mathbb{Z}\}$ are an orthonormal base for V_0; ϕ is the scaling function we already talked about.

Theorem 2.3.2. Consider an MRA (V_j, ϕ). It generates the following:

a. a direct sum decomposition of $L^2(\mathbb{R})$ through orthogonal spaces of V_j;

b. a mother wavelet $\psi(t)$ such that its wavelets $\{\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k)\}_{j,k\in\mathbb{Z}}$ compose an orthonormal basis for $L^2(\mathbb{R})$.

Proof. Starting with point a., thanks to theorem C.2.1 and because $V_0 \subset V_1$, then $V_1 = V_0 \oplus V_0^\perp$. Analogously

$$V_2 = V_1 \oplus V_1^\perp = (V_0 \oplus V_0^\perp) \oplus V_1^\perp$$

and so it is possible to generalize everything:

$$V_{n+1} = V_n \oplus V_n^\perp = \cdots = V_0 \oplus V_0^\perp \oplus \cdots \oplus V_n^\perp, n \in \mathbb{N}.$$

All these subspaces are orthogonal with each other by construction, which means that their orthogonal direct sum is $L^2(\mathbb{R})$:

$$V_0 \oplus \left(\bigoplus_{n=0}^{+\infty} V_n^\perp\right) = L^2(\mathbb{R}). \tag{2.16}$$

Let's try to write V_0 in another way. Thanks again to theorem C.2.1,

$$V_0 = V_{-1} \oplus V_{-1}^\perp, \qquad V_{-1} = V_{-2} \oplus V_{-2}^\perp, \tag{2.17}$$

so we have

$$V_0 = (V_{-2} \oplus V_{-2}^\perp) \oplus V_{-1}^\perp,$$

and generalizing:

$$V_0 = V_{-n}^\perp \oplus \cdots \oplus V_{-1}^\perp.$$

At this point the main goal is to see that

$$V_0 = \bigoplus_{n=1}^{+\infty} V_{-n}^\perp. \tag{2.18}$$

In order to do that it is needed to show that $\left[\bigoplus_{n=1}^{+\infty} V_{-n}^\perp\right]^\perp = \{0\}$.

Supposing $g \in V_0 = \left[\bigoplus_{n=1}^{+\infty} V_{-n}^\perp\right]^\perp \Rightarrow g \in (V_{-n}^\perp)^\perp \ \forall\ n \in \mathbb{N}$, which yields that:

$$g \in \bigcap_{n=0}^{+\infty} V_{-n}.$$

So, since V_j are increasing, for property 3. of 2.3.1:

$$\bigcap_{n=0}^{+\infty} V_{-n} = \{0\},$$

which proves that (2.18) is valid.

Finally, thanks to that, considering (2.16) we have that:

$$L^2(\mathbb{R}) = \bigoplus_{n\in\mathbb{Z}} V_n^{\perp}, \tag{2.19}$$

proving point a.

For point b., let's consider a mother wavelet ψ for MRA (V_n, ϕ); in this context it means that all possible integer translations of ψ are an orthonormal basis for $V_0^{\perp}$.

Considering the same isomorphism defined in C.2, since

$$I(f(t)) = \sqrt{2}f(2t), \qquad I(\psi(t)) = \sqrt{2}\psi(2t),$$

and

$$\int_{-\infty}^{+\infty} f(t)\overline{\psi(t-n)}dt = \frac{1}{2}\int_{-\infty}^{+\infty} f(2t)\overline{\psi(2t-n)}dt = 0, n \in \mathbb{Z}, f \in V_0$$

then from property 4. in 2.3.1 $\psi(2t-n) \in V_1^{\perp}\ \forall\ n \in \mathbb{Z}$ which yields that $\{\sqrt{2}\phi(2t-n) \mid n \in \mathbb{Z}\}$ is an orthonormal basis for $V_1^{\perp}$. Following the same reasoning we have that in general $\{2^{j/2}\psi(2^j t - n) \mid n \in \mathbb{Z}\}$ is an orthonormal basis for the generic $V_j^{\perp}, j \in \mathbb{Z}$. It means that, because of (2.19), wavelets generated from mother wavelet ψ form an orthonormal basis for $L^2(\mathbb{R})$.

□

A very famous example of orthonormal basis of space $L^2(\mathbb{R})$ built with wavelets is the one realized by Haar in 1910:

$$\left\{\psi_{j,n}(t) = \frac{1}{2^{j/2}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\}_{j,n\in\mathbb{Z}}, \tag{2.20}$$

where

$$\psi(t) = \begin{cases} 1 & \text{if } 0 \le t \le \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} \le t \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

So dilations and translations of ψ generate an orthonormal basis of function space $L^2(\mathbb{R})$.

Using function basis to work on a signal is a very important and appreciated technique, since it allows for a linear approximation of the signal itself. In fact, given a signal $f(t)$ and an orthonormal basis of functions $\mathcal{B} = \{g_m\}_{m\in\mathbb{N}}$, it is possible to project this signal over M given functions chosen from the orthonormal basis, so that the following:

$$f_M = \sum_{m=0}^{M-1} \langle f, g_m \rangle g_m \tag{2.22}$$

represents the linear approximation of signal f using M functions from basis $\mathcal{B}$.

Of course this type of operations have to be smart: one doesn't want to build an approximation too far from the original one (maybe cause of a low value of M) and at the same time using M too large will lead to an ineffective use of this approximation, since it will be too large to compute it in a efficient way. The main point is to minimize the approximation error, which can be computed as the sum of the remaining squared:

$$\varepsilon[M] = \|f - f_M\|^2 = \sum_{m=M}^{+\infty} |\langle f, g_m \rangle|^2.$$

This approximation can be improved making it non-linear if one considers random functions and doesn't choose them a priori; in that case we have:

$$f_M = \sum_{m\in I_M} \langle f, g_m \rangle g_m, \tag{2.23}$$

and the approximation error is simply computed as:

$$\varepsilon[M] = \|f - f_M\|^2 = \sum_{m\notin I_M} |\langle f, g_m \rangle|^2.$$

2.4 Wavelet regularity

2.4.1 α-Lipschitz functions and vanishing moments

In this section the main point will be to analyse what happens when the interval considered becomes very small (so $s \to 0$). It is a known fact that singularity points or irregular structures carry more informations when analysing signals, so in those cases it becomes very important to know what happens from an analytical point of view.

Definition 2.4.1. A function f is said to be **pointwise α-Lipschitz**, with $\alpha \geq 0$ in point ν if $\exists\ K \in \mathbb{R}^+$ such that:

$$|f(t) - p_\nu(t)| \leq K|t - \nu|^\alpha,\ \forall\ t \in \mathbb{R}, \tag{2.24}$$

where p_ν is the Taylor approximation of f in ν of degree $m = \lfloor \alpha \rfloor$:

$$p_\nu(t) = \sum_{k=0}^{m-1} \frac{f(k)(\nu)}{k!}(t-\nu)^k. \tag{2.25}$$

It is important to remember that:

- for the existence of p_ν, $f \in C^m$;
- at each point ν, p_ν is uniquely defined.

Of course, if f is pointwise α-Lipschitz $\forall\ \nu \in [a,b], a, b \in \mathbb{R}$, then f is said to be **uniformly α-Lipschitz** over $[a,b]$.

Observation 2.4.2. From definitions above, we have the following:

- if $0 \leq \alpha < 1$ then equation (2.24) becomes:

$$|f(t) - f(\nu)| \leq K|t-\nu|^\alpha,\ \forall\ t \in \mathbb{R}; \tag{2.26}$$

- for each function f approximated in ν by its Taylor representation p_ν, it is possible to define the approximation error as:

$$e_\nu(t) := f(t) - p_\nu(t). \tag{2.27}$$

  This error is always bounded following this relation:

$$|e_\nu(t)| \leq K|t-\nu|^\alpha; \tag{2.28}$$

- if f is bounded but discontinuous in ν then it means that $\alpha = 0$;
- if $\alpha < 1$ in ν, then f is not differentiable in ν.

In order to enunciate properties related to local regularity of a signal, it's of capital importance to use wavelets with vanishing moments; this characteristic of them yield a first relation between differentiability of f and decay of its wavelet transform.

Definition 2.4.3. A wavelet ψ has $n > \alpha$ **vanishing moments** if:

$$\int_{-\infty}^{+\infty} t^k \psi(t) dt = 0, 0 \leq k < n. \tag{2.29}$$

In this case, α is referred to equation (2.27).

Observation 2.4.4. The followings derives from the above definition:

1. a wavelet with n vanishing moments is orthogonal to every possible polynomial of degree $n-1$;
2. since $\alpha < n$, then degree of p_v is at most equal to $n-1$.

As a consequence of point 1. in 2.4.4, we have:

$$Wp_\nu(u,s) = 0 \tag{2.30}$$

and since $f = p_\nu + e_\nu$, for linearity of wavelet transform:

$$Wf(u,s) = We_\nu(u,s). \tag{2.31}$$

Definition 2.4.5. Consider a wavelet $\psi \in C^n$ with n vanishing moments; then it is said to have a **fast decay of order n** if $\forall\ m \in \mathbb{N}\ \exists\ C_m \in \mathbb{R}$ such that the following is true with:

$$|\psi^{(k)}(t)| \leq \frac{C_m}{1+|t|^m},\ \forall\ t \in \mathbb{R}, 0 \leq k \leq n. \tag{2.32}$$

This definition means that if wavelet ψ has n vanishing moments, not only itself but also its first n derivatives has a fast decay. This hypothesis will be taken for granted in the next two theorems, as it will be important for their demonstration.

2.4.2 Regularity theorems

Definition 2.4.5 is really important, as it allow to enounce the following theorem.

Theorem 2.4.6. Consider a function $f \in L^2(\mathbb{R})$ which is $\alpha \leq n$-Lipschitz in point ν; then the **Jaffard theorem** affirms that exists $A \in \mathbb{R}^+$ such that

$$|Wf(u,s)| \leq As^{\alpha+\frac{1}{2}}\left(1+\left|\frac{u-v}{s}\right|^\alpha\right),\ \forall\ (u,s) \in \mathbb{R}\times\mathbb{R}^+. \tag{2.33}$$

Proof. Since f is α-Lipschitz in ν, it means that its Taylor approximation in ν allows for (2.24) to be valid. Since ψ has n vanishing moments, it has been already stated that $Wp_\nu(u,s) = 0$, so:

$$\begin{aligned}
|Wf(u,s)| &= \left|\int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right)dt\right| \\
&= \left|\int_{-\infty}^{+\infty} (f(t)-p_\nu(t))\frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right)dt\right| \\
&\leq \int_{-\infty}^{+\infty} |f(t)-p_\nu(t)|\frac{1}{\sqrt{s}}\left|\psi\left(\frac{t-u}{s}\right)\right|dt \\
&\leq \int_{-\infty}^{+\infty} K|t-\nu|^\alpha\frac{1}{\sqrt{s}}\left|\psi\left(\frac{t-u}{s}\right)\right|dt.
\end{aligned}$$

Let's change the variable $x = \frac{t-u}{s}$:

$$|Wf(u,s)| \leq \sqrt{s} \int_{-\infty}^{+\infty} K|sx + u - \nu|^{\alpha} |\psi(x)| dx.$$

In order to conclude our proof, consider that $|a+b|^{\alpha} \leq |a|^{\alpha} + |b|^{\alpha} \leq 2^{\alpha}(|a|^{\alpha} + |b|^{\alpha})$ and that $\int_{-\infty}^{+\infty} |x|^{\alpha} |\psi(x)| dx = C$, so:

$$\begin{aligned} |Wf(u,s)| &\leq K 2^{\alpha} \sqrt{s} \left(s^{\alpha} \int_{-\infty}^{+\infty} |x|^{\alpha} |\psi(x)| dx + |u-\nu|^{\alpha} \int_{-\infty}^{+\infty} |\psi(x)| dx \right) \\ &\leq K 2^{\alpha} s^{\alpha + \frac{1}{2}} \left(\int_{-\infty}^{+\infty} |x|^{\alpha} |\psi(x)| dx + \left| \frac{u-\nu}{s} \right|^{\alpha} \int_{-\infty}^{+\infty} |\psi(x)| dx \right) \end{aligned}$$

Finally, putting

$$A := K 2^{\alpha} \max \left\{ \int_{-\infty}^{+\infty} |x|^{\alpha} |\psi(x)| dx, \int_{-\infty}^{+\infty} |\psi(x)| dx \right\}$$

proof is completed. □

The latter is useful per sè, but it is very helpful with as it allows to prove the following, which is similar but a little bit stronger.

Theorem 2.4.7. If $f \in L^2(\mathbb{R})$ is α-Lipschitz ($\alpha \leq n$) over $[a,b]$, then it exists $A \in \mathbb{R}^+$ so that:

$$|Wf(u,s)| \leq A s^{\alpha + \frac{1}{2}}, \ \forall \ (u,s) \in [a,b] \times \mathbb{R}^+ \tag{2.34}$$

Proof will not be reported because it is trivial after having proved theorem 2.33: only difference is to put $u = \nu$, since $\nu \in [a,b]$.

This closes this section and this chapter, as behaviour of wavelet transform when $s \to 0$ have been studied: thanks to the fast decay of wavelets and their derivatives, it was demonstrated that modulus of wavelet transform is always controlled by a power of s.

Still, problem has not been solved, as this transform is not invariant to translation. In the next chapter some fundamental tools will be defined, and they will improve chances of succeeding.

Chapter 3

Finite scattering paths

3.1 Aim and reasoning behind

Despite wavelets being regular and localized function, they are not invariant to translation; in section 2.2 it was proved that wavelet transform is a mathematical operator invariant to shift of the function taken in input, but it is not itself translation invariant.

On the other hand, in chapter 1 it was discussed how Fourier transform is not translation invariant, but adding a modulus to the integral would make it so; since it became clear that properties given by wavelet transform were too convenient to come back to Fourier transform, in 2012 Stéphane Mallat introduced a new concept called scattering transform [43]: its work was then expanded with one of his PhD students, Joan Bruna [9], [10], [11].

This is the situation right now:

- Fourier transform is not invariant to neither translations nor deformations;
- Fourier transform with a modulus is invariant to translations, but not to deformations;
- wavelet transform is not invariant to translations, but it invariant to deformations.

The goal now is to create a mathematical transform both translations and deformations invariant, bounding together wavelet transform and modulus.

Before doing that, it is necessary to be more precise about deformations invariance: from now on, a new definition, more formal, will be used.

Definition 3.1.1. Denote with $L_\tau f(x)$ the following C^2-diffeomorphism:

$$L_\tau f(x) = f(x - \tau(x)), x \in \mathbb{R}^d, f \in L^2(\mathbb{R}), \tag{3.1}$$

with $\tau(x) \in \mathbb{R}^d$ called displacement field. Then the following distance can be defined over any $\Omega \subset \mathbb{R}^d$:

$$d_\Omega(\mathbb{1}, \mathbb{1} - \tau) := \sup_{x \in \Omega} \|\tau(x)\|_2 + \sup_{x \in \Omega} |\nabla \tau(x)| + \sup_{x \in \Omega} |H\tau(x)|, \tag{3.2}$$

with:

- $\mathbb{1}$ notation for the unitary vector;
- $\|\tau(x)\|_2$ is the euclidean norm in $\mathbb{R}^d$ of τ;
- $\nabla \tau(x)$ is its Jacobian matrix;
- $H\tau(x)$ is its Hessian matrix.

Thanks to that, it is possible to define precisely what type of deformations we want transforms to be invariant to.

Definition 3.1.2. A translation-invariant operator ϕ defined over an Hilbert space $\mathbb{H}$ is said to be **Lipschitz-continuous** to the action of a C^2-diffeomorphism if, $\forall\ \Omega \subset \mathbb{R}^d\ \exists\ C \in \mathbb{R}$ such that $\forall\ f \in L^2(\mathbb{R}^d)$ with support in Ω and $\forall\ \tau \in C^2(\mathbb{R}^d)$, the following holds:

$$\|\phi(f) - \phi(L_\tau f)\|_{\mathbb{H}} \leq C\|f\| \left(\sup_{x \in \mathbb{R}^d} |\nabla \tau(x)| + \sup_{x \in \mathbb{R}^d} |H\tau(x)| \right). \tag{3.3}$$

Another way to say that for any operator equation (3.3) is valid is that *it does not commutes with the action of a diffeomorphism.*

Now, obviously, if ϕ is Lipschitz-continuous, not only that operator is translation invariant, but it would also be deformation invariant in such a way that even high-frequency instabilities can be avoided; this can be obtained by using a wavelet transform, but at this moment this is not possible, since it is not invariant to translation.

As already stated, it's time to modify it so that (1.9) holds for a modified version of wavelet transform.

In general, given $Q \in L^2(\mathbb{R})$ an operator non-invariant for translations, it yields that

$$\int Qx(u)du \tag{3.4}$$

is translation invariant. Then it is possible to define a simple operator invariant to translations, like $Qx = x \star \psi_\lambda$, which yields:

$$\int x \star \psi_\lambda(u)du = 0, \tag{3.5}$$

because

$$\int \psi_\lambda(u)du = 0. \tag{3.6}$$

This is a problem for our aim, as any linear transformation of Qx would be equal to zero. So it is necessary to add another pooling operator which has to be nonlinear, since a linear one would make the integral vanish still [11].

The operator chosen is a modulus:

$$My(u) := |y(u)| = \sqrt{|y_r(u)|^2 + |y_i(u)|^2}, \tag{3.7}$$

where $y_r(u)$ and $y_i(u)$ are respectively the real and complex part of signal $y(u) = y_r(u) + iy_i(u)$: this allows to remove the complex phase of the signal. Operator M has the following properties:

- it is non-expansive, in order to preserve stability, which means that $\|Mx - My\| \leq \|x - y\|$. This is trivial and will not be proved;
- as proved in [9], if $\Phi \in L^2(\mathbb{R}^n)$ is a non-expansive operator and commutes with the action of diffeomorphism, then M is a pointwise operator which means that $My(u)$ only depends on the value of $y(u)$ itself.

Finally we obtained the following:

$$\int M(x \star \psi_\lambda) = \int |x \star \psi_\lambda(u)|du = \|x \star \psi_\lambda\|_1,$$

which means that the resulting translation invariant coefficients are $L^1(\mathbb{R}^2)$ norms.

3.2 Definition of scattering transform and basic properties

3.2.1 Definition of scattering propagator

It is important to state that from now on a different notation will be used for wavelets; for any $\lambda \in \mathbb{R}^+, \psi_\lambda(t) = \lambda\psi(\lambda t)$ and so its Fourier transform is $\widehat{\psi}_\lambda(\omega) = \widehat{\psi}\left(\frac{\omega}{\lambda}\right)$. This means that energy of ψ is concentrated in $t = 0$, while $\omega = \lambda$ represents the center frequency of $\widehat{\psi}$, and it is the point where most of its energy is concentrated as well.

Scattering coefficients will be commented later on; for this moment, just consider that if q is the number of wavelets per octave, then $\lambda = \frac{2^k}{q}, k \in \mathbb{Z}$.

Changing the value for λ would allow to obtain a first order of scattering coefficients:

$$\{|x \star \psi_\lambda|\}_\lambda, \tag{3.8}$$

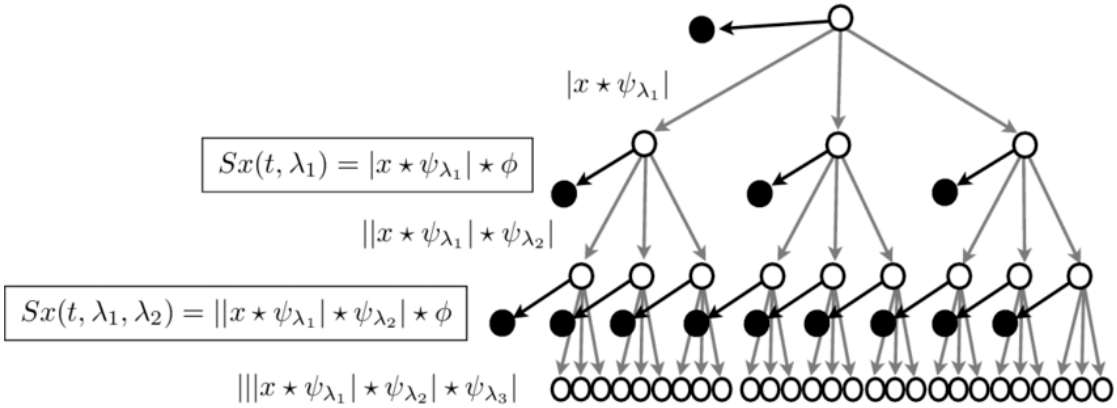

Figure 3.1. What a cascade of modulus and convolutions really mean: each node represents a coefficient. Note how the number of coefficients increases when increasing the order, one of the reasons why usually coefficients after the second order are not computed.

which measures the sparsity of wavelet coefficients, at the cost of a loss of information due to integration (and not because of the presence of modulus): to be precise, all non zero frequencies are lost. In order to recover part of the informations lost in the process, it is possible to reiterate the process, and compute the second order coefficients as follows:

$$\{|x \star \psi_{\lambda_1}| \star \psi_{\lambda_2}\}_{\lambda_2}. \tag{3.9}$$

It basically means that we are computing the wavelet coefficients for $|x \star \psi_{\lambda_1}|$, so that we have a much larger family of invariant coefficients.

This process can be iterated further, allowing for a more general definition.

Definition 3.2.1. Consider a finite list of numbers $p = (\lambda_1, \dots, \lambda_n)$; it is called **scattering path**. All the possible scattering paths of length n are denoted as $\mathcal{P}_n$. The set of all admissible scattering paths of non finite length is $\mathcal{P}_\infty$. It is a sequence of frequencies, and the output of the scattering transform will basically be a function of it.

Definition 3.2.2. Given $p \in \mathcal{P}_n$, the following operator is called **scattering propagator**:

$$\begin{aligned} U_p x &:= U_{\lambda_n} \dots U_{\lambda_1} x \\ &= | \, |x \star \psi_{\lambda_1}| \dots | \star \psi_{\lambda_n}|, \end{aligned} \tag{3.10}$$

It means that operator U_p is basically a cascade of convolutions and modulus that allows to compute scattering coefficients up to order n; each U_λ maps the frequencies covered by $\widehat{\psi}_\lambda$ to lower frequencies with modulus; if $p = \emptyset \Rightarrow U_p x = x$.

3.2.2 Basic properties

This operator is well-defined and continuous in $L^2(\mathbb{R}^d)$ since $\|U_\lambda f\| \leq \|\psi_\lambda\|_1 \|f\|$ for any choice of λ.

Property 3.2.3. Consider two scattering paths

$$p = (\lambda_1, \ldots, \lambda_n), p' = (\lambda'_1, \ldots, \lambda'_m),$$

with $n \neq m$ in general. Then, denoted a new scattering path

$$p + p' = (\lambda_1, \ldots, \lambda_n, \lambda'_1, \ldots, \lambda'_m),$$

the following holds:

$$U_{p+p'} = U_p U_{p'} \tag{3.11}$$

If needed the scattering propagator can be defined for "negative" frequencies too, analogously to what happens with Fourier transform.

Property 3.2.4. Consider a scattering propagator U_λ. Then, $\forall\ f \in \mathbb{R}^d$ the following is valid:

$$U_{-\lambda} f = U_\lambda f, \tag{3.12}$$

and denoting $-p = (-\lambda_1, \ldots, -\lambda_n)$ as a negative path, it is possible to generalize (3.12) as follows:

$$U_{-p} f = U_p f. \tag{3.13}$$

Now, sets of function ψ_λ not necessarily have to be wavelets in general; this means that a scattering propagator (which yields a scattering transform) for another set of function can be built. However, in the original formulation of scattering transform the only type of function used was wavelets, so the general case will not be explained (also because it would fall outside the scope of this work).

In any case, the fact that ψ are wavelets is really important for the definition of scattering propagator; for example 3.2.4 would not be valid if other functions would have been used; in fact, since f is a real function, $Wf(\lambda, \cdot) = Wf(-\lambda, \cdot)$, which yields (3.12), proving the point. So for simplicity only real functions with positive paths (i.e. paths with all positive values) will be considered, as the same properties holds for complex functions and scattering propagators that use paths including negative values.

Chapter 4

From scattering transform to Wavelet Scattering Transform

4.1 Localized Scattering Transform

Scattering propagator was the last tool needed to define the scattering transform.

Definition 4.1.1. Consider a function $f \in L^1(\mathbb{R}^d)$, then the following is defined as the **scattering transform** of function f along the path $p \in \mathcal{P}_\infty$:

$$Sf(p) := \frac{1}{\mu_p} \int U_p f(x) dx \tag{4.1}$$

$$= \frac{1}{\mu_p} \int |\, |f \star \psi_{\lambda_1}| \ldots | \star \psi_{\lambda_n}(x)| dx \tag{4.2}$$

where

$$\mu_p = \int U_p \delta(u) du, \tag{4.3}$$

and $\delta(x)$ is a Dirac delta distribution.

It's immediate to note that this new integral operator has two properties inherited by the scattering propagator.

Property 4.1.2. Consider a scattering path p and its associated scattering transform S_p; then the following are valid:

1. if f is a real function, then $Sf(-p) = Sf(p)$;

2. if f is a complex function, Sf can be defined for negative paths, but not in such a way that property 1. holds;

3. extending this idea we get that $S\mu f(p) = |\mu| Sf(p), \forall\ \mu \in \mathbb{R}$;

One last step to be done before introducing WST is to consider a localized version of the scattering transform called windowed scattering transform, which will be used to show that scattering transforms have all the properties required (translation invariance, diffeomorphism stability). Most importantly, it will be showed that its limit for $J \to +\infty$ defines the wavelet scattering transform, and it will be showed explicitly that it is exactly coincident to one operator that has been already defined.

Definition 4.1.3. Let $J \in \mathbb{Z}$ and $p = (\lambda_1, \dots, \lambda_n)$ a scattering path, with $|\lambda_k| = 2^{jk} > 2^{-J}$. Then a **windowed scattering transform** of a function $f \in L^1(\mathbb{R}^d)$ is defined for all admissible paths p as the following:

$$\begin{aligned} S_J f(p) :&= U_p f \star \phi_{2^J}(x) \\ &= |f \star \psi_{\lambda_1}| \star \psi_{\lambda_2}| \dots | \star \psi_{\lambda_n}| \star \phi_{2^J}(x) \\ &= \int U_p f(u) \phi_{2^J}(x-u) du, \end{aligned} \tag{4.4}$$

with $\phi_{2^J}(x) = 2^{-dJ}\phi(2^{-J}x)$ being the scaling function, one of the key points of this operator; for now it localizes scattering transforms over spatial domains of size proportional to 2^J.

We now have two goals:

- defining the scattering transform as a generalization of (4.4) when $J \to +\infty$;
- expanding the amount of functions admissible from L^1 to L^2.

These two goals are both accomplished thanks to the following theorem.

Theorem 4.1.4. If $f \in L^1(\mathbb{R}^d)$, then

$$\lim_{J \to +\infty} S_J f(p) = Sf(p) = \frac{1}{\mu_p} \int U_p f(x) dx. \tag{4.5}$$

for any admissible path p.

Proof. Let's prove that

$$\lim_{J \to +\infty} \sqrt{2^{dJ}} \|S_J f(p)\| = \|\phi\| \int U_p f(x) dx. \tag{4.6}$$

Now, since $S_J f(p) = U_p f \star \phi_{2^J}$, then for the Plancherel formula we have the following:

$$2^{dJ}\|S_J f(p)\|^2 = 2^{dJ}(2\pi)^{-d}\int |\widehat{U_p f}(\omega)|^2 |\widehat{\phi}(2^J\omega)|^2 d\omega. \tag{4.7}$$

Now, thanks to the following:

- $\phi \in L^1(\mathbb{R}^d)$, and so are its derivatives, so $\widehat{\phi}(\omega) = O((1+|\omega|)^{-1})$, and then

$$\lim_{J\to+\infty}(2\pi)^{-d}2^{dJ}|\widehat{\phi}(2^J\omega)|^2 = \|\phi\|^2\delta(\omega); \tag{4.8}$$

- if $f \in L^1(\mathbb{R}^d) \Rightarrow U_p f \in L^1(\mathbb{R}^d) \Rightarrow \widehat{U_p f}(\omega)$ is continuous in $\omega = 0$,

it is possible to affirm that (4.7) can be re-written as:

$$\lim_{J\to+\infty} 2^{dJ}\|S_J f(p)\|^2 = |\widehat{U_p f}(0)|^2\|\phi\|^2,$$

which proves that (4.6) is true. Coming up to the conclusion, thanks to proposition 3.1 in [43], $U_p\delta \neq 0$, so from (4.6) we have:

$$\lim_{J\to+\infty}\frac{\|S_J f(p)\|}{\|S_J\delta(p)\|} = \frac{\int U_p f(x)dx}{\int U_p\delta(x)dx}, \tag{4.9}$$

and that means (4.5) has been proved. □

The only thing that is missing to have the complete definition of a Wavelet Scattering Transform is the scaling function; in fact it is present in (4.4), but seems to be missing in (4.1); does it mean that scattering transform and wavelet scattering transform are that different?

As a matter of fact, it is just a consequence of the notation used by Mallat; in fact scaling function is already there in (4.1), it just hasn't been explicated. In order to do that (and to finally define Wavelet Scattering Transform) it is needed to start from (4.4), remembering that $\phi_{2^J}(x) = 2^{-dJ}\phi(2^{-J}x)$. At this point it is basically done, because thanks to (4.7) and (4.8):

$$\widehat{U_p f} \star \widehat{\phi}_{2^J} \underset{J\to+\infty}{\longrightarrow} \widehat{U_p f}$$

which implies thanks to Plancherel formula

$$U_p f \star \phi_{2^J} \underset{J\to+\infty}{\longrightarrow} U_p f;$$

this proof can be found in [31].

It is important to note that Folland does that for parameter $t \to 0$, but $\widehat{\phi}_{2^J}$ has the same asymptotic behaviour when $J \to +\infty$.

So this means that in (4.1) the convolution with a scaling function was already considered, it just wasn't explicit because $\|\phi\| = 1$ for definition.

4.2 Definition of Wavelet Scattering Transform

Now it is possible to define the Wavelet Scattering Transform.

Definition 4.2.1. Considering a scattering path $p = (\lambda_1, \dots, \lambda_n)$ and a scaling function ϕ, the **wavelet scattering transform** of the generic signal f is defined as the following integral operator:

$$\begin{aligned} WSf(p) :&= \frac{1}{\mu_p} \int U_p f \star \phi(u) du \\ &= \frac{1}{\mu_p} \int | \, |f \star \psi_{\lambda_1}| \dots | \star \psi_{\lambda_n}| \star \phi(u) du. \end{aligned} \tag{4.10}$$

In order to close this part of the work, it is necessary to show that WST is an operator that preserves the norm (so it is non-expansive), translation-invariant and does not commute to the action of a diffeomorphism.

4.3 Properties of WST

Before starting, it is important to note that everything we have seen for scattering transform and WST is valid for functions in $L^1(\mathbb{R}^n)$; this is true, but fortunately everything can be expanded for $L^2(\mathbb{R}^n)$. This will not be demonstrated in this work, but can be found in [43].

Most of these properties will be proven starting from the windowed scattering transform S_J; this is a little easier and in any case the wavelet scattering transform inherits all properties of it, representing its limit when $J \to +\infty$. When necessary, some properties of scattering propagator U_p will be stated as well.

4.3.1 Non-expansion

Proposition 4.3.1. WST is a non-expansive operator, which means that:

$$\|WSf(p) - WSh(p)\| \leq \|f - h\|, \ \forall \ p \in \mathcal{P}_J, \ \forall f, h \in L^2(\mathbb{R}^d). \tag{4.11}$$

Proof. Let's denote

$$U_J f = \{A_J f, (U_\lambda f)_{\lambda \in \mathbb{Z}}\},$$

with $A_J f = f \star \phi_{2^J}$ and $U_\lambda f = |f \star \psi_\lambda|$ the classic scattering propagator, with ψ a generic mother wavelet. This is a non-expansive operator since we are working with wavelets, and we know that wavelet transform is unitary; besides the modulus is non-expansive. In fact:

$$\begin{aligned} \|U_J f - U_J h\|^2 &= \|A_J f - A_J h\|^2 + \sum_\lambda \| |U_\lambda f| - |U_\lambda h| \|^2 \\ &\leq \|Wf - Wh\|^2 \leq \|f - h\|^2. \end{aligned}$$

Since:

- $U_\lambda U_p = U_{\lambda+p}$ because of 3.2.3;
- $A_J U_p = S_J(p)$, for the definition of $S_J(p)$ itself;

then:

$$U_J U_p f = \{S_J f(p), (U_{p+\lambda} f)_{\lambda \in \mathbb{Z}}\},\ p \in P_J. \tag{4.12}$$

Now, since U_J is non-expansive, the following is valid:

$$\begin{aligned} \|U_p f - U_p h\|^2 &\geq \|U_J U_p f - U_J U_p h\|^2 \\ &= \|S_J f(p) - S_J h(p)\|^2 + \|U_{\lambda+p} f - U_{\lambda+p} h\|^2 \end{aligned}$$

and summing those for all admissible paths p:

$$\sum_{p \in \mathcal{P}_\infty} \|S_J f(p) - S_J h(p)\|^2 \leq \|f - h\|^2.$$

Since S_J is non expansive, WST is non expansive as well when $J \to +\infty$, so proof is done. □

4.3.2 Translation invariance

In order to prove that WST is a translation-invariant, it is necessary to enounce the following.

Theorem 4.3.2. Consider the translation operator $L_c f(x) = f(x - c)$; then for every $f \in L^2(\mathbb{R}^d), \forall\ c \in \mathbb{R}^d$ and $\forall\ p \in \mathcal{P}_\infty$:

$$\lim_{J \to +\infty} \|S_J(p) f - S_J L_c f(p)\| = 0. \tag{4.13}$$

Proof. Since $S_J L_c f(p) = L_c S_J f(p)$, we can easily prove that

$$\lim_{J \to +\infty} \|L_c S_J f(p) - S_J f(p)\| = 0\ \forall\ f \in L^2(\mathbb{R}^d). \tag{4.14}$$

This is because every $f \in L^2(\mathbb{R}^d)$ can be written as a limit for the sequence $\{f_n\}_{n \in \mathbb{N}}$; since S_J is non-expansive as seen before, and L_c is unitary by definition, it is possible to verify to:

$$\begin{aligned} \|L_c S_J f(p) - S_J f(p)\| \leq & \|L_c S_J f_n(p) - S_J f_n(p)\| + 2\|f - f_n\| \\ & \underset{n \to +\infty}{\longrightarrow} \|L_c f - f\| + \|f - f\| = 0. \end{aligned}$$

Which proves our point. □

4.3.3 Lipschitz-continuous

Theorem 4.3.3. Consider $f \in L^2(\mathbb{R}^n)$ with $\|U_p f\|_1 < +\infty$ and $\tau \in C^2(\mathbb{R}^n)$ with $\|\nabla\tau\|_\infty \leq \frac{1}{2}$. Then the following relation holds:

$$\|S_J L_\tau f(p) - S_J f(p)\| \leq C\|U_p f\|_1 K(\tau), \tag{4.15}$$

with

$$K(\tau) := 2^{-J}\|\tau\|_\infty + \|\nabla\tau\|_\infty \max\left\{\log\left(\frac{\|\Delta\tau\|_\infty}{\|\nabla\tau\|_\infty}\right), 1\right\} + \|H\tau\|_\infty, \tag{4.16}$$

where $\|\Delta\tau\|_\infty := \sup_{x,u\in\mathbb{R}^d} |\tau(x) - \tau(u)|$.
Demonstration is really long: should I prove it?

Chapter 5

Final considerations

5.1 Recap on Fourier, Wavelet and Wavelet Scattering Transform

Fourier transform is considered as one of the main landmarks in signal analysis, as it was one of the first transform that allow to represent functions in their respective frequency domain, as opposed to the classic time domain representations. Because of that, it became extremely popular. It is defined as:

$$\widehat{f}(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}dt$$

Unfortunately some disadvantages of it have become quite clear as signal analysis tools were getting better and better; in fact Fourier transform is not:

- translation-invariant;
- invariant for deformation.

More on that on chapter 1.

These two properties are fundamental for an effective and efficient signal analysis, especially today: while performing Machine Learning and Deep Learning you want models to be able to transform signals without losing its original meaning or properties: this is something Fourier transform is not good at.

Despite that, it is clear that putting the modulus to the Fourier transform

$$|\widehat{f}(\omega)| = \left|\int_{-\infty}^{+\infty} f(t)e^{-i\omega t}dt\right|$$

makes it translation invariant; this is due to basic properties of the integral, as the area under a graph does not change if function is translated. This will be

Transform	Non-expansive	Translation	Lipschitz
Fourier	✓	✗	✗
Fourier with modulus	✓	✓	✗
Wavelet	✓	✗	✓
Scattering propagator	✓	✗	✗
Wavelet Scattering	✓	✓	✓

Table 5.1. Recap on the properties for the main transforms analysed.

very important later on, and will be one of the main ideas that allowed to go from wavelet transform to the wavelet scattering transform.

In order to solve the deformation-invariance problem, wavelet transform was introduced, and it allowed to have a non-expansive operator invariant to deformation; the main disadvantage of it is that it is not invariant to translations, so despite a good step in the right direction, main goal is not reached yet.

Let's remind ourself of the definition. Given a mother wavelet ψ , wavelet transform is defined as:

$$Wf(a,b) = \int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{a}}\psi^*\left(\frac{t-b}{a}\right)dt.$$

Main properties for wavelet transform can be found in chapter 2.

Ideas were put down, so it was just a matter of putting them together. For now we have:

- Fourier transform, which is sensible to translations and deformations;
- Fourier transform with modulus, which is only invariant to translations but not to deformations;
- wavelet transform, which is sufficiently invariant to deformations (that condition is called Lipschitz-continuous; it is defined in chapter 3) but not to translations.

Now it is time to put those two together.

In order to do that, a new idea opened a new chapter in this field of knowledge: not only putting the modulus to the convolution, but using a cascade of modulus and convolutions in order to recover more information on the signal. To do that it was necessary to define a new operator, called scattering propagator:

$$U_p x = |\,|x \star \psi_{\lambda_1}| \ldots | \star \psi_{\lambda_n}|.$$

p is a sequence of frequencies called scattering path. More on that on 3.2.

Thanks to that it was possible to define a non-expansive, translation-invariant and Lipschitz-continuous integral operator. It is the Wavelet Scattering Transform:

$$WSf(p) := \frac{1}{\mu_p} \int U_p f \star \phi(u) du.$$

At the end of the day it just is a scattering propagator averaged by a scaling function ϕ; the latter is yield by the same MRA that itself was used to define the family of wavelets generated by the mother wavelet ψ, as seen in 2.3.

In this work it was also proved explicitly that equation (4.10) is perfectly equal to (4.1): missing of scaling function was only due to the notations used by Mallat in [43]. This was done in section 4.

Finally, in order to close this chapter and this part of the work altogether, all properties studied for each mathematical operator defined are summarised in table 5.1.

Part II

Time-series analysis and forecasting

Transforming the signal in order to study its spectrum is undoubtedly an incredible achievement from a mathematical perspective, which probably allowed science as a whole to come as far as it did.

But that is not the only point of this work.

In this second part theoretical bases and properties for time-series will be taken on, analysed deeply, and demonstrated if necessary. The need of doing that is evident in the structure itself of ECGs, as they are modelled as time-series.

Time-series is one of the most important concepts in data analysis, since it allows to model in a simple and intuitive way a lot of different processes, related to health, economy, chemistry, and so on. Besides that, it can help in forecasting the future, especially when talking about processes that repeat themselves in a redundant way as time goes on.

This is probably the main feature of time-series: it allows to take a certain quantity that change with time (for example the number of passengers that travel with a certain airplane company), values taken with regular intervals, and extract some information about them; the airplane example is very didactic and will be widely used while explaining all the basis of time-series and its analysis.

This part serves as a theoretical foundation for the second aim of this work. In a modern world where everything can be saved and written in some kind of tabular data, these informations become at the same time very important, as they describe basically everything about us, and also very fragile, as they can be stolen very easily and very frequently, too: in the last few years news of data robbery can be the main focus for a couple of days.

It becomes particularly delicate when talking about data regarding personal health, as clinical reports or ECGs, since a lot of personal and private information are hidden in there: all these issues makes collecting ECGs so much harder because of all the privacy issues that were listed before and because of the time needed to collect a sufficient amount of data.

These problems would be overcome if only a new technique allowed to create realistic human beats, without actually having to use an electrocardiograph: this aim will be achieved thanks to the time-series forecasting and new methods of performing it, including some neural methods: besides, a comparison between them and some classical methods (like ETS and ARIMA) will be performed.

This part will be organized in a fairly simple way. Chapter 6 will talk about everything that is needed to be known when computing time series: its definition, its properties, its characteristics and a little bit of theory behind its concept too. As mentioned beforehand, a very known dataset (called airline) will be introduced and used in order to show some of the properties in a very intuitive way.

Forecasting is the most interesting thing about this part, and it will be divided in two chapters. Chapter 7 will be about all things regarding forecast for time-series using classic time-series techniques, in particular ETS and ARIMA: they

don't require any training model, that will be used later on. Forecasting is the main core of time-series applications: in fact it is possible to use them in order to forecast behaviours of a lot of different processes for a period which represent the future, with a certain degree of certainty. Of course, the furthest the time of the prediction, the bigger the uncertainty.

After having introduced this issue, all the forecasting method will be analysed and described in a deepened fashion, and each one of them will be used in order to create artificial beats, which will be used for train a classifier that will be able to distinguish regular beats from irregular ones: of course results will be compared.

Finally, in chapter 8 the last frontier of time-series forecasting will be taken on, as time-series forecasting models based on deep-learning neural neural network will be analysed. These models are becoming more and more popular as time goes by, and their potential is far to be reached yet.

Despite being quite heavy from a computational point of view, their training process (i.e. the research of the optimal values for all their parameters) become feasible thanks to new hardware technologies such as GPUs: the expectation are high, as it is expected from them to dramatically reduce mistakes and creating very realistic human beats: thanks to it scores for the classification process will be far more solid than a lot of other research works.

Chapter 6

Time-series analysis

6.1 Theoretical foundation

Time-series analysis is one of the main derivation of one of the main breach of mathematics: probability and statistics. All of its theoretical foundation will come from it, and the main features will be listed here.

Let's start reminding some of the basic definitions.

Definition 6.1.1. In this work a **time-series** will be defined as a sequence of random variables $(X_{t_i})_{i\in\mathcal{I}}$, $\mathcal{I}$ a generic set of indexes, $\mathcal{T}$ set of instants and $t_i \in \mathcal{T}$ time instant in which value X_{t_i} has been taken; remember that this happens with regularity (so each time interval is the same, i.e. $t_{i+1} - t_i = c, i \in \mathcal{I}, c \in \mathbb{R}$ constant value).

Consider that in this work every time-series will be of finite length, so from now on $I = \{1 \dots, n\} \subset \mathbb{N}$ and if we have to consider the distribution function of it we have the following notation:

$$F_{X_{t_1},\dots,X_{t_n}}(x_1, \dots, x_n) = \mathbb{P}(X_{t_1} \leq x_1, \dots, X_{t_n} \leq x_n),$$

which is basically the same notation as the one used for indicating the cumulative distribution function of a random vector of n random variables.

Definition 6.1.2. A time-series is said to be **first-order stationary** if the following holds:

$$F_{X_{t_i}}(x) = F_{X_{t_i+k}}(x),\ \forall\ x \in \Omega,\ \forall\ t_i, k \in \mathcal{T}, \tag{6.1}$$

with Ω being the set for every possible value admissible for x.

Definition 6.1.3. A time-series is said to be **strongly stationary** if

$$F_{X_{t_1},\dots X_{t_n}}(x_1, \dots, x_n) = F_{X_{t_1+k},\dots X_{t_n+k}}(x_1, \dots, x_n),\ \forall\ x_i \in \Omega,\ \forall\ t_i, k \in I. \tag{6.2}$$

Equation (6.2) is also known as strong stationarity condition.

Definition 6.1.4. A time-series is said to be **weakly stationary of order** $s \in \mathbb{N}$ if

$$\mathbb{E}(X_{t_1} \dots X_{t_s}) = \mathbb{E}(X_{t_1+h} \dots X_{t_s+h}) < +\infty, \ \forall \ k \text{ s.t. } t_i + k \in \mathcal{T}, \ \forall \ i \in I. \quad (6.3)$$

It basically means that the first s moments, which are defined as:

$$m_s = \int x^s f(x) dx, \quad (6.4)$$

with $f(x)$ probability density function of random variable considered, are the same for each random variable forming the time-series.

Observation 6.1.5. In case of a weakly stationary time-series of order one it means that each random variable has the same mean as all the others, i.e.:

$$\mathbb{E}(X_{t_i}) = \mathbb{E}(X_{t_i+k}), \forall \ t_i, k \in \mathcal{T}, i \in I.$$

If the time series is weakly stationary of order two the following is true:

$$\begin{aligned} \mathbb{E}(X_{t_i}) &= \mathbb{E}(X_{t_i+k}), \\ \mathbb{E}(X_{t_i} X_{t_j}) &= \mathbb{E}(X_{t_i+k} X_{t_j+k}), \forall \ t_i, t_j, k \in \mathcal{T}, i, j \in I, \end{aligned}$$

which means that not only the mean, but also the variance remains the same for all random variables. In that case, if $X_i \sim \mathcal{N}(\mu, \sigma^2) \ \forall \ i \in \mathcal{I} \Rightarrow$ considered time-series is also strongly stationary.

Concepts, definitions and properties for mean and variance of a random variable and of a time-series will be taken for granted; more importance will be put on correlation function, since it will be necessary in order to talk about autocorrelation of elements of a time-series, as from its value a lot of properties are deducted.

Definition 6.1.6. Consider two time-series X_{t_i} and X_{t_j}, which have means μ_i and μ_j and variances σ_i^2 and σ_j^2 respectively; the following value is called **covariance** between random variables:

$$\gamma(t_i, t_j) = \mathbb{E}\left[(X_i - \mu_i)(X_j - \mu_j)\right]. \quad (6.5)$$

The definition above is useful in order to define the following.

Definition 6.1.7. Consider the same random variables as above. The following function is called **correlation** between X_{t_i} and X_{t_j}:

$$\rho(t_i, t_j) = \frac{\gamma(t_i, t_j)}{\sqrt{\sigma_i^2 \sigma_j^2}}. \quad (6.6)$$

These quantities can be defined for strongly stationary time series. Let's consider a time-series $(X_{t_i})_{i\in\mathcal{I}}$, which has to be strongly stationary; among all other things, it implies that all variables have the same mean and the same variance. Then, for any t_i, t_j, k we have:

$$\gamma(t_i, t_j) = \gamma(t_i + k, t_j + k)$$
$$\rho(t_i, t_j) = \rho(t_i + k, t_j + k).$$

If we also put $t_i = t - k, t_j = t$ we have:

$$\gamma(t_i, t_j) = \gamma(t - k, t) = \gamma(t, t + k) = \gamma_k$$
$$\rho(t_i, t_j) = \rho(t - k, t) = \rho(t, t + k) = \rho_k,$$

which means that for these kind of processes covariance and correlation functions only depends on lag k.

These allows for the following definitions.

Definition 6.1.8. Consider a strongly stationary time-series $(X_{t_i})_{i\in\mathcal{I}}$. Then the following is called **autocovariance function**

$$\gamma_k = \mathrm{Cov}(X_t, X_{t+k}) \tag{6.7}$$

and the following is called **autocorrelation function** (ACF)

$$\rho_k = \frac{\mathrm{Cov}(X_t, X_{t+k})}{\sqrt{\mathbb{V}(X_t)\mathbb{V}(X_t + k)}} = \frac{\gamma_k}{\gamma_0}. \tag{6.8}$$

They both depend only on lag k.

In some cases considering each variable can be not really useful: in those cases the following is defined.

Definition 6.1.9. The **partial autocorrelation function** (PACF) is defined, for a stationary time-series $(X_t)_{t\in\mathbb{N}}$ as:

$$p_k = \frac{\mathrm{Cov}(X_t, X_{t+k}|X_{t+1}, \ldots, X_{t+k-1})}{\mathbb{V}(X_t|X_{t+1}, \ldots, X_{t+k-1})}. \tag{6.9}$$

It can be defined, in a different way, for non stationary time-series.

As the standard autocorrelation function, it is dependent only on value of lag k and it is represents the autocorrelation between a generic X_t and X_{t+k}, but without the effect of the variables between them $(X_{t+1}, \ldots, X_{t+k-1})$.

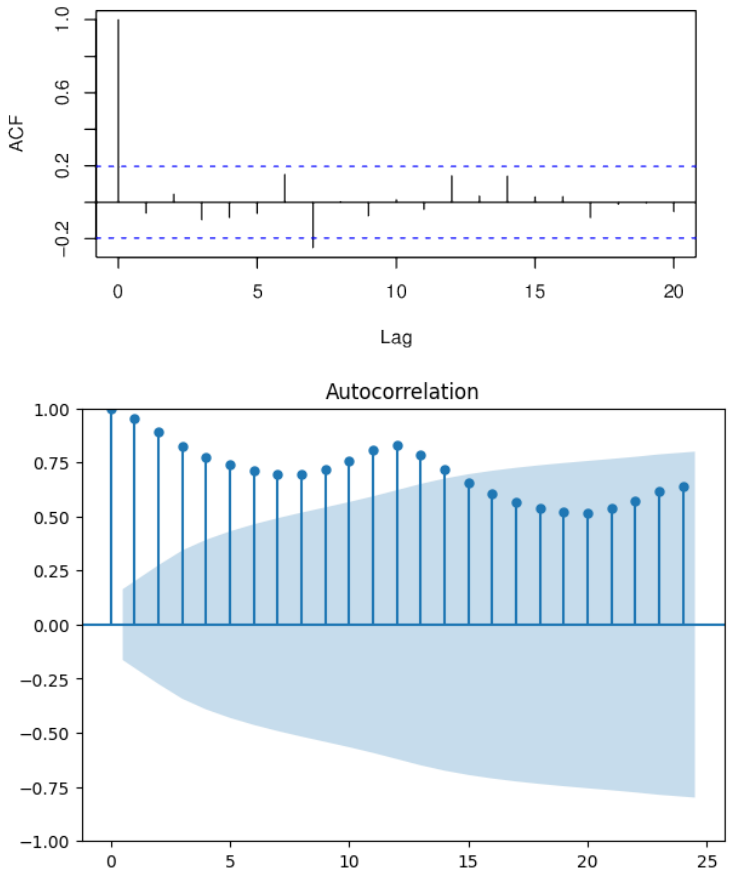

Figure 6.1. Correlograms for white noise process (above) and raw airline dataset (below). It can be noted how the first just rely on elements of lag $k = 1$, while for the second also elements with lag $k = 12$ is more relevant then its neighbours, as expected because of the seasonality component.

Note that autocovariance of lag $k = 0$ is just variance of random variable X_t, which is the same for each random variable; mathematically,

$$\gamma_0 = \mathbb{V}(X_t) = \mathbb{V}(X_{t+k}).$$

Autocorrelation is a very important part of time-series analysis, as it gives a lot of informations on just how much each elements depend on all the other. For example, if autocorrelation is very high for $k = 1$ and very low for all the others, we can think of that process like it's a white noise; on the other hand, if it is very low for every element and very high for $k = 4$ or $k = 12$, it means that a seasonal component is very strong, and it has to be considered.

The best way to visualize autocorrelation is to use a graph where on the x-axis lags are put, while on the y-axis values of ACF for the corresponding lag are being put; these values are represented by little bars, and these graphs are called correlograms. Correlograms on figures 6.1 allow to visualize some of the properties that were listed before.

6.2 Time-series elements

Thanks to time series it is possible to study the changing of processes that can be described numerically as time passes. They carry an incredible amount of informations that needs to be extracted, in order to understand what is happening. For this reason it is mandatory to model time-series in such a way that allow to decompose them: each component will carry some kind of information, and each of them will be necessary in order to understand the whole model.

Usually, when decomposing time-series, three components are found and used:

- the first one is called **trend** (b), and it is used to denote a constant change of the value independent to every other components. It is often described with just a linear increase or decrease, as it is sufficient for most of the times;
- when a certain pattern repeat itself periodically it is described as a **seasonal component** (s_t), and it is probably the most telling component; while in some subjects, like macro-economics, it can be used a lot less, it is absolutely fundamental when talking about processes related to tourism or climate. A very interesting example of those two component bounded together is the airplane dataset, shown in figure 6.2. More on that later;
- any other effect that can not be described as a trend effect or a seasonal component is generically labelled as an error or **residual** component (r_t).

It is important to put some assumption on residual particularly, since it is a random component and as such needs to be modelled adequately:

- residuals need to be uncorrelated with each other, and their average needs to be zero;
- usually residuals are assumed to have constant variance and normal distribution. So basically:

$$r_t \sim \mathcal{N}(0, \sigma^2), \tag{6.10}$$

  σ^2 dependent from the model used. It might as time passes on.

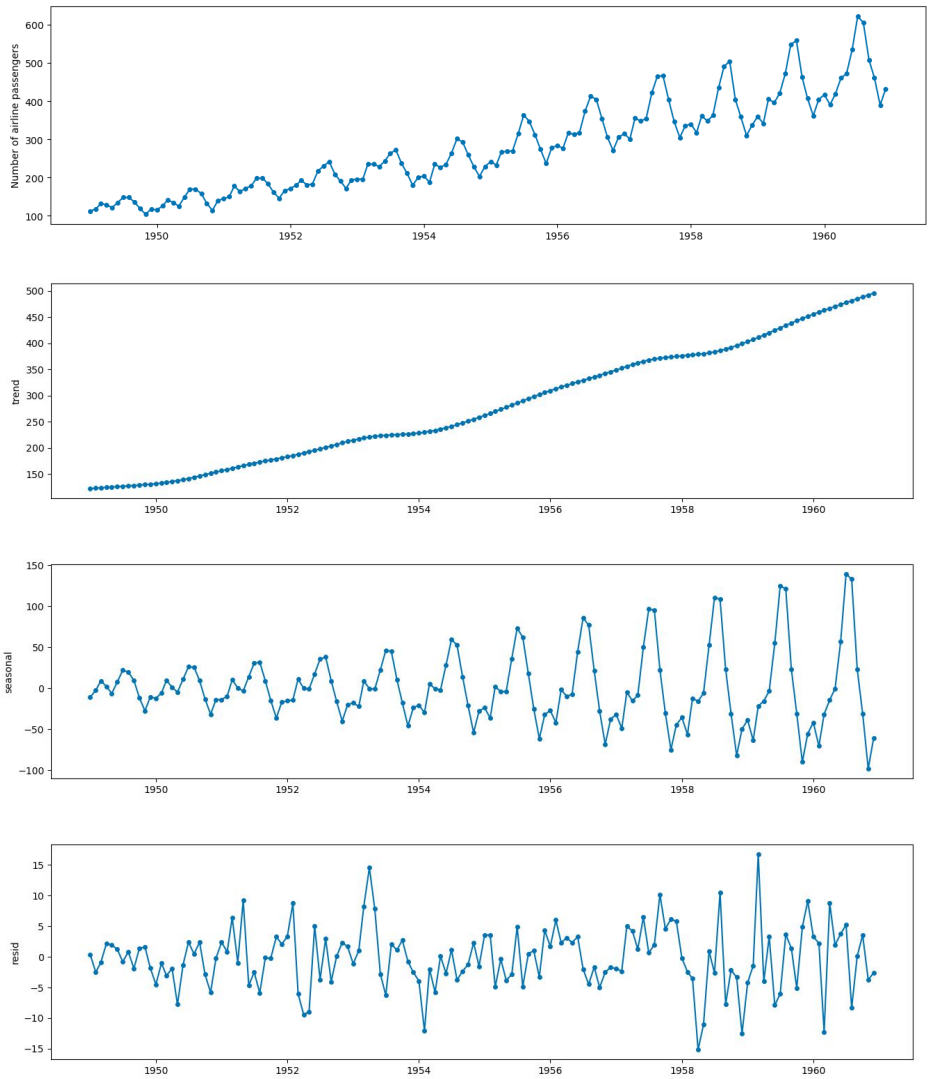

Figure 6.2. Time-series components for famous dataset Airline: this decomposition has been evaluated using STL.

The most popular decomposition method for time-series is the additive one, where:

$$x_t = a + bt + s_t + r_t, \tag{6.11}$$

with $a, b \in \mathbb{R}$ constant obtained by estimating the time series by using a linear model. In this case the seasonal effect is obtained as

$$s_t = x_t - a - bt + e_t. \tag{6.12}$$

If the seasonal effect grows as trend value increases, then a multiplicative model

might be more appropriate:

$$x_t = a + bts_t + r_t. \tag{6.13}$$

Sometimes a more precise estimation of the trend is required; a popular choice is the moving average, which considers some elements before x_t, some elements after and compute an average among all of them. If for example $l \in \mathbb{N}$ elements before and after time t has to be considered, we obtain the following:

$$\widehat{b}_t = \frac{1}{N} \sum_{k=-l}^{l} x_{t+k}, \ N = 2l + 1. \tag{6.14}$$

In cases like this, seasonal component is computed as:

$$s_t = x_t - \widehat{b}_t + e_t. \tag{6.15}$$

Decomposing a time-series is one of the main operations that needs to be done when analysing them, before making any kind of forecast. In fact it is usual to have some phenomenon masked behind a trend or a seasonal component, and it underlines the importance of separating them.

There lots of example available: the unemployment rate is one of the most studied seasonal adjusted series, since there are some periods of the year where it goes up and down that are only due to the season: this however can masks some interesting considerations or trending components, that might have nothing to do with any seasonal component.

On the other hand, considering only the trend might not be enough in order to understand some seasonal phenomenon, like in figure 6.2. It is clear that the number of airplane passengers is increasing significantly every year, but this new peak value is reached approximately every summer, while the amount of passengers remain relatively low during the winter holidays, even if it grows every year as well.

Chapter 7

Classic time-series forecasting

Time-series are an exquisitely useful tool when analysing particular phenomenon, especially if a trend or seasonal component is present. In those cases time-series can be utilized in order to try and predict the future, as long as the conditions considered do not change. This is what this chapter will be all about: forecasting with time-series. There is a big quantity of models one can choose from, and all of them can be useful depending what process modelled by a time-series is considered.

This chapter will consider two different classes of forecasters: classic ones (especially ETS and ARIMA, used in this work) and neural ones (LSTM and RNN: there will be some comments about neural networks in general and those two in particular), as they yield some significant improvements with respect to older forecaster thanks to the raw computational power granted by the training process and the notable flexibility of the structure of a neural network.

7.1 Basic notions of the forecasting problem

Model fitting and model forecasting are two faces of the same medal: once a fitting model for a certain database has been found, using it to forecast the future is a natural consequence.

In order to understand that, let's make an example. Consider a dataset formed by couples: $(x_t, y_t)_{t=1,\dots T}$; the simplest way to fit those data is to use a linear model, generically described by the following equation:

$$\widehat{y}_t = \alpha + \beta x_t, \tag{7.1}$$

and the main point of the regression problem is to find parameters of α and β that

allows for the minimization of the mean square error

$$\text{MSE} = \frac{1}{T}\sum_{t=1}^{T}(y_t - \widehat{y}_t)^2, \tag{7.2}$$

where y_t is the real value and $\widehat{y}_t$ is its approximation computed with the linear model.

Once α and β have been found, then it is possible to use them in order to find values of $y_{T+1} = \alpha + \beta x_{T+1}$ and in general to find values for $y_t, t > T$ that were not studied yet or that are not available yet, since those instants represent the future. When this procedure is put in place, it is called forecasting with time-series.

Of course there are lots of different models that can be used in order to do that, and a few were implemented in this work; they will be described with no shortage of details.

7.2 Forecasting with Exponential Time Smoothing

Linear regression model can be very useful in order to understand the idea behind forecasting, but it is not as powerful when fitting an actual model; this is because, factually, it can only manage to model a trend, but it becomes useless when any seasonal component comes into play.

For this reason it is urgent to introduce new and more complicated models, in order to increase the degrees of freedom and manage to fit better the data from the considered phenomenon.

The first model analysed is called **exponential time smoothing** (ETS), which rests on a recursive procedure that allows to compute forecasting value as time instant advances.

The model itself is pretty simple. In order to find the predicted value for $\widehat{y}_{T+1}$ a weighted average between true value y_T and predicted value $\widehat{y}_T$ is taken in consideration, obtaining the following equation:

$$\widehat{y}_{T+1} = \alpha y_T + (1-\alpha)\widehat{y}_T, \tag{7.3}$$

with α smoothing parameter. Of course the predicted value $\widehat{y}_T$ is computed considering the true and the predicted value of y in time instant $T-1$, and so on.

Iterating this process would give the following equation:

$$\widehat{y}_{t+1} = \alpha y_t + (1-\alpha)\widehat{y}_t. \tag{7.4}$$

It is important to note that, in those cases, $\widehat{y}_0$ has to be taken in an arbitrary way, since it is the very first value and can not be computed by using this recursive

method. A very popular choice is to take the average of all known values:

$$\widehat{y}_0 = \frac{1}{T}\sum_{t=0}^{T} y_t. \tag{7.5}$$

This method is a good start, but it appears to be very naive, and it does not take in consideration any trend nor seasonal components.

This is possible, and this version of the ETS model is called the Holt-Winters' additive model, that will be used in this work. Lots of other versions have been developed, especially some multiplicative versions: to know them better, check on [37].

Definition 7.2.1. Considering a time-series $\{(x_t, y_t)\}_{t\in\mathcal{T}}$, the **Holt-Winters' additive method** can be described by the following equations:

$$\widehat{y}_{t+h} = l_t + hb_t + s_{t+1-m} \tag{7.6}$$
$$l_t = \alpha(y_t - s_{t-m}) + (1+\alpha)(l_{t-1} + b_{t-1}) \tag{7.7}$$
$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{7.8}$$
$$s_t = \gamma(y_t - l_{t-i} - b_{t-1}) + (1-\gamma)s_{t-m}, \tag{7.9}$$

where:

- $\widehat{y}_{t+h}$ is the predicted value with lag h to actual time t;
- l_t is called the level component of the smoothing model and it is a more useful version of (7.3). Its smoothing parameter is $0 \leq \alpha \leq 1$;
- b_t represents the trend component, with smoothing parameter $0 \leq \beta \leq 1$;
- s_t is the seasonal component with smoothing parameter $0 \leq \gamma \leq 1$;
- m is the seasonality period (for example $m = 12$ for monthly data).

This model represent a good improvement from the linear model, and the amount of components that has to be managed represent a good indicator: as a matter of facts this model has a trend and a seasonality component, and besides that every element is computed considering the previous one, so that it has the same logic of the time-series itself.

Of course this is a quite simple model, and it still has its difficulties when, for example, trend value changes or when contribution from residuals is higher. For this reason another model has been implemented: it is very famous because it has into them three simple models, that will be analysed in depth, considering conditions of their existence and some interesting properties about how each of them works and has to be used together with the others.

7.3 Forecasting with ARIMA models

ARIMA models are by far some of the most popular and studied models in statistics, thanks to their simplicity from a theoretical point of view and their implementation too.

They are the result of the bounding of three different models, each of them very basic and intuitive:

- the autoregressive part, denoted with AR(p);
- the moving average part, denoted with MA(q);
- the differencing process, denoted with I(d),

where p, q are called order of the autoregressive and moving average parts and q is the degree of differencing.

Before analysing everything, define the following random variable $w_t \sim \mathcal{N}(0, \sigma^2)$ and call it white noise; besides, it is important to define a new operator that will be fundamental in this context.

Definition 7.3.1. Consider a time-series $\{x_t\}_{t\in\mathcal{T}}$. Then the **backshift operator** is defined as follows:

$$Bx_t = x_{t-1}, \quad B^p x_t = x_{t-p}. \tag{7.10}$$

It will be very important in order to study properties of these models.

Now it is possible to analyse each one of them.

7.3.1 Autoregressive model

Consider a time series $\{x_t\}_{1\leq t\leq T}$ and a parameter $p < T$. The main idea of the autoregressive model of order p is to write value of x in instant t as a weighted average of all previous p values for x; in this way the following definition can be explicited.

Definition 7.3.2. The **autoregressive model of order** p is defined as:

$$\begin{aligned} x_t &= \alpha_1 x_{t-1} + \cdots + \alpha_p x_{t-p} + w_t \\ &= \sum_{i=1}^{p} \alpha_i x_{t-i}. \end{aligned} \tag{7.11}$$

It is possible to write this model using the backshift operator:

$$\begin{aligned} x_t &= (\alpha_1 B + \cdots + \alpha_p B^p) x_t + w_t \\ &= \left(\sum_{i=1}^{p} \alpha_i B^i \right) x_t + w_t. \end{aligned} \tag{7.12}$$

Another way to write it is to isolate the white noise:

$$(1 - \alpha_1 B - \cdots - \alpha_p B^p) x_t = w_t, \tag{7.13}$$

and thanks to this notation it is possible to introduce the following operator:

$$\theta_p(B) = 1 - \alpha_1 B - \cdots - \alpha_p B^p, \tag{7.14}$$

such that

$$\theta_p(B) x_t = w_t. \tag{7.15}$$

It will be very important in order to have a lighter notation when the final ARIMA model will be introduced, but even right now it results very useful indeed. In fact one can consider the characteristic polynomial of this model $\theta_p(B)$, with B treated as a real or complex number and find its roots, i.e. values for B such that:

$$\theta_p(B) = 0. \tag{7.16}$$

Then the following can be stated.

Property 7.3.3. Consider an autoregressive process of order p, described by the following characteristic polynomial $\theta_p(B)$. Then consider its roots z; if $|z| > 1$ for each one of them, then the autoregressive model is stationary of the second order. This means that its mean and its average are constant.

This helps a lot when using these models, because in those cases they are very stochastically very stable and allow to model lots of different phenomenon: a very common example is a random walk, which is basically an AR(1) model with $\alpha = 1$; in fact it is described by the following equation:

$$x_t = x_{t-1} + w_t. \tag{7.17}$$

For what concerns forecasting using an autoregressive model, the following holds:

$$x_{t+1} \sim N\left(\sum_{i=1}^{p} \alpha_i x_{t+1-i}, \sigma^2\right). \tag{7.18}$$

This comes as an obvious consequence of the fact that, when arrived at time t, the only stochastic component is given by white noise w_t, which is a random normal variable with null mean and variance equal to σ^2. Since at time t value of $\sum_{i=1}^{p} \alpha_i x_{t+1-i}$ is deterministic, then x_{t+1} can be modelled as in equation (7.18).

7.3.2 Moving average and differencing models

The second part of the ARIMA model is the moving average, denoted with MA(q).

Definition 7.3.4. Consider a time-series $\{x_t\}_{t\in\mathcal{T}}$. Then the following is defined as the **moving average model of order** q:

$$\begin{aligned} x_t &= w_t + \beta_1 w_{t-1} + \cdots + \beta_q w_{t-q} \\ &= (1 + \beta_1 B + \cdots + \beta_q B^q) w_t \\ &= \left(1 + \sum_{i=1}^{q} \beta_i B^i\right) w_t. \end{aligned} \tag{7.19}$$

Defining the characteristic polynomial as

$$\phi_q(B) = (1 + \beta_1 B + \cdots + \beta_q B^q), \tag{7.20}$$

then the moving average model can be written with the following notation:

$$x_t = \phi_q(B) w_t. \tag{7.21}$$

The most important condition of existence for this model is the following.

Proposition 7.3.5. Consider a moving average model MA(q) and its characteristic polynomial $\phi_q(B)$. Then the model is identifiable if $\phi_q(B)$ is invertible, which happens if and only if every root z of $\phi_q(B)$ satisfy the following condition:

$$|z| > 1. \tag{7.22}$$

The last ingredient missing in order to define an actual ARIMA model is the so called integrated or differencing model.

Definition 7.3.6. Consider a time-series $\{x_t\}_{t\in\mathcal{T}}$. The **integrated or differencing model or order** d is defined as:

$$(1 - B)^d x_t = \underbrace{(1 - B) \dots (1 - B)}_{d \text{ times}} x_t = w_t. \tag{7.23}$$

So basically an integrated model is used in order to consider a differentiation of order d such that the time-series itself collapses and becomes at best a sequence of white noises (so ultimately a white noise). It is a very useful tool, especially if any trend component is found while working on the time-series considered.

7.3.3 Adding a seasonal element and bounding everything together: SARIMA models

After having defined every component of the model, it is possible to give the following.

Definition 7.3.7. Consider a time-series x_t, an autoregressive model AR(p), a moving average model MA(q) and a differencing model I(d). Then it is possible to define a model called **ARIMA**(p, d, q) as the following:

$$\theta_p(B)(1 - B)^d x_t = \phi_q(B)w_t. \tag{7.24}$$

It is important to note that before apply the moving average and the autoregressive parts, it is necessary to differentiate: so it can be said that an ARIMA model is just an ARMA on the differences.

As already saw, time-series are very useful not only to analyse phenomenons with a constant growth (so with a trending), but also to fit seasonal data: ARIMA models can be used to do that too, and in this work is it of capital importance that a seasonal component is considered, because of the characteristics of ECGs.

Definition 7.3.8. Consider the same time-series and the same ARIMA model as before. If a seasonal component can be found, then we can define a **SARIMA** model by applying a very subtle change to equation (7.24):

$$\theta_p(B^s)(1 - B^s)^d x_t = \phi_q(B^s)w_t, \tag{7.25}$$

where s represents the seasonal parameter. For example, for data repeating themselves each twelve periods of time, we have $s = 12$: this can be the case for monthly data with annual seasonality.

These models have been fundamental when they were developed, and to this day are very important not only for forecasting itself, but also in order to easily learn about how to apply these principles.

However, now new techniques have been available for some time now, and they are pushing the boundaries of performances in ways that were not considered before: form now on deep-learning methods for forecasting will be considering, especially recurrent neural networks, which are, from both a practical and philosophical point of view, the ideal tool in that sense.

Chapter 8

Forecasting with neural methods

The methods seen until now are classic mathematical methods, which fit data and then can be used in order to forecasting. However in the last few years neural networks have entered the scene, and are showing unbelievable potential in a lot of fields, and forecasting with time-series makes no exceptions: one of the aims of this work is to compare average performances and average computational time for forecasting using classic methods and neural techniques.

Before talking in detail about that, it is necessary to introduce neural network, and understand their structure: in particular recurrent neural network will be considered, as they will be used in every method that will be seen in generation of beats.

8.1 Neural networks

Neural networks are, in this moment of time, one of the most exciting prospects in science: comparing with classic mathematical methods, they are able to use a way bigger amount of data and produce results that, on average, are a lot more effective. However using them has a very big cost, as they need to be trained in a recurrent way for a very long time: this increases computational costs in such a dramatic way that it results to be feasible by only using a more efficient and expensive hardware component, called GPU.

8.1.1 Basic structures

The name used to indicate them seems a little bit off from a neurological point of view, but it does make more sense considering its logical structure: let's take some

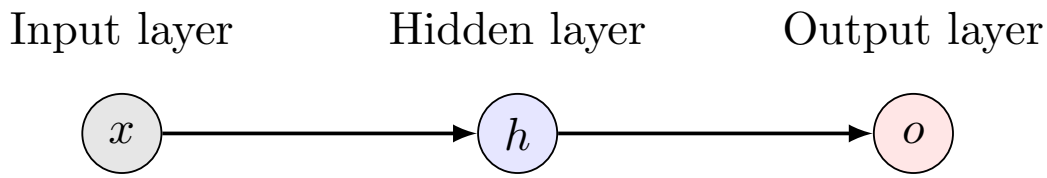

Figure 8.1. A neural network with an hidden layer composed by one single neuron, with $h = f(xw + b)$ and $o = h$.

input data x, usually a vector but it can also be just a number. If the considered neural network just had one neuron, we would have a classic equation:

$$o = f(wx + b), \tag{8.1}$$

where:

- w is the value used to multiply the input data. It is called **weight** and it usually is a matrix;
- b is usually a vector, it is added to wx and is called **bias**;
- function f is needed in order to introduce some sort of non-linearity. Popular

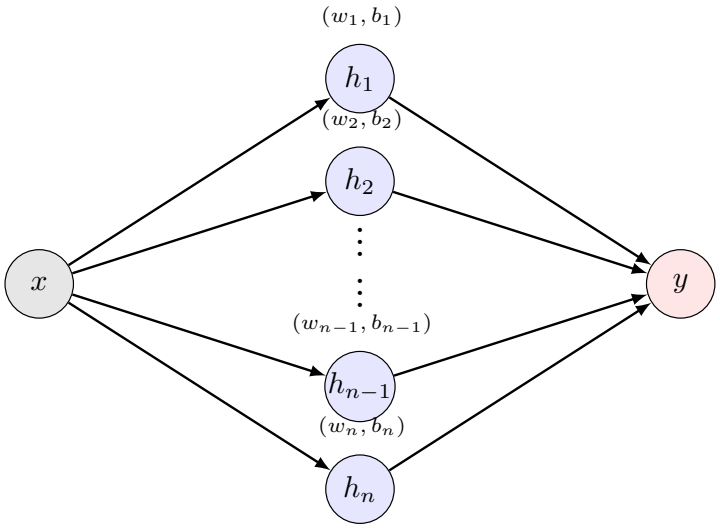

Figure 8.2. Neural network with n neurons in the hidden layer. It can be noted how any neuron has its own values of weight w_i and b_i.

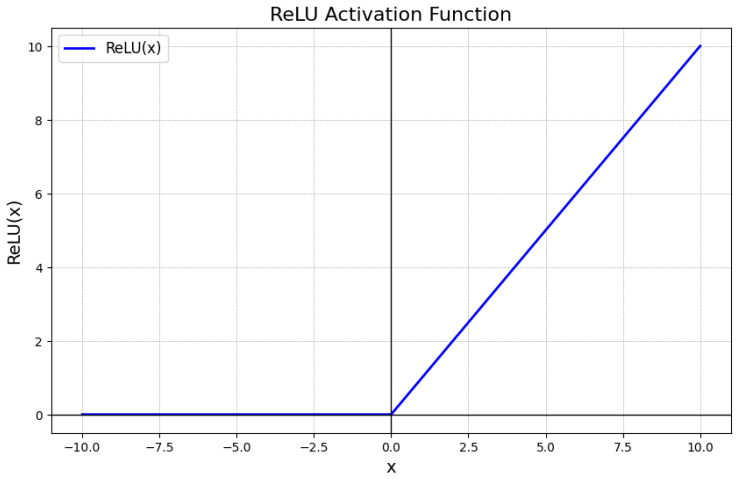

Figure 8.3. Plot for ReLU function.

choices are tanh and function $\text{ReLU}(x) = \max\{0, x\}$, that can be seen plotted in image 8.3;

- o is just the output of the operation.

This simple idea of neural network can be seen in image 8.1.

Having cleared that, now it results easier to understand the case with more than one neuron and more than one hidden layer. In fact one of the main pros of these structures is their complexity, given by the very high amount of neurons which usually composes a neural network.

Considering a neural network with an hidden layer with n neurons, each of them with weight w_i and bias b_i, input data x will be given to all those neurons and then processed differently depending on the values of the weight and the bias for each neuron; then the output can be obtained as a weighted average of all the values obtained in each neuron. A similar model can be seen in image 8.2

8.1.2 From input to output: forward process and loss function

This process can be iterated, creating neural networks with more than one hidden layer, each of them with a certain amount of neurons. Besides input layer and output layer can have more than one neuron each.

The process that allow data to flow from input to output is called **forward propagation**, and it is fundamentally important for neural network to work. The point is that if only forward propagation is used, the whole process might be

quite static; besides that, weights and biases might be far from their optimal values, and this would basically mean that the whole network is useless. Since it is impossible to find a priori the optimal values for w_i and b_i, it is necessary to change them dynamically as new data flows inside the network: this process is called **training** and it represents one of the main differences when talking about classic mathematical methods and AI-based techniques.

Consider for simplicity a neural network like the one in image 8.1: the same reasoning can be iterated when using more neurons and more layers. Main aim of this training part is to find ideal values for biases and weights in order that when new data are taken into the neural network, the value in output $\widehat{y}$ will be as close as possible to the real value y.

First thing needed is a measure that assumes low values when y and $\widehat{y}$ are close, and very high values if those two quantities are very different: this is called **loss function** and it can be defined in lots of different ways depending on the task taken on. For example, for regression task two very popular choices are mean absolute error and mean squared error:

$$\text{MAE}(= y - \widehat{y}), \qquad \text{MSE} = (y - \widehat{y})^2, \tag{8.2}$$

while for classification tasks like the one tackled in this work a 0-1 loss function is usually deployed:

$$\mathcal{L}(y, \widehat{y}) = \mathbb{1}_{\{y \neq \widehat{y}\}}. \tag{8.3}$$

8.1.3 Improving the results: back-propagation and optimization algorithms

Having defined that, it is now necessary to understand how the new values for the hyper-parameters of the neural network (weights and biases) are computed. Every algorithm of this type is composed by two main operations:

- the **backward** or **back-propagation** process, during which the value that will be used to update actual weights is computed;
- the **optimization** process, when the value computed in the backward process is used to update the hyper-parameters of the neural networks.

In this work only one was used: the Adam optimizer [38]. It is one of the most efficient, effective and famous optimization algorithms, and it will be analysed in depth in the following pages. There are lots of other possible algorithms, depending on the structure of the neural network and the assignment considered: if interested in studying some of them, check [32].

The algorithm is based on the computations of two moment vectors m_0, v_0, initialized with null vectors, and it is basically composed of the following steps.

After having initialized parameters $\beta_1, \beta_2, \varepsilon$, the learning rate α and the stochastic objective function f with hyper-parameters θ:

- backward process: compute the gradient with respect to $\theta : g_t = \nabla_\theta f_t(\theta_{t-1})$;
- update moments:

$$m_t = \frac{1}{1-\beta_1^t}\beta_1 m_{t-1} + (1-\beta_1)g_t, \qquad v_t = \frac{1}{1-\beta_2^t}\beta_2 m_{t-1} + (1-\beta_2)g_t; \quad (8.4)$$

- update parameter vector:

$$\theta_t = \theta_{t-1} - \alpha\frac{m_t}{\sqrt{v_t}+\varepsilon}. \quad (8.5)$$

The last two steps make the optimization process.

An interesting topic is regarding possible choices for learning rate. For simple algorithms a fixed value is preferred, but there are lots of methods that allow to change it adaptively: in this work a fixed value (standard choice for Adam algorithm implemented in sktime) for learning rate is used.

8.2 Forecasting with Recurrent Neural Networks

A general knowledge of neural networks and how they work have now been exposed, and it is necessary in order to understand how forecasting with those actually work. There are lots of different possible models that can be used in order to achieve success in a task like that, but one in particular seems to be perfect not only in its physical structure, but in its concept too: the Recurrent Neural Network (in short, RNN).

They are a particular type of structure that allow to make us of sequential information. As a matter of fact, in original neural networks every input is considered to be independent for each other, but when considering tasks like generating words to compose phrases or forecasting with time-series you have to consider what has come before in order to predict what is coming next.

What differentiate them from all other neural network is that they are time-driven, so basically everything (inputs, outputs and hidden states) are labelled with the time instant during which that value has been computed. So basically one can think of RNNs as memory-equipped neural networks that not only takes in consideration new inputs as time goes on, but "remembers" the hidden state computed before and uses it as in order to compute a new output, and so on.

From a mathematical point of view, these are the operations that characterize them:

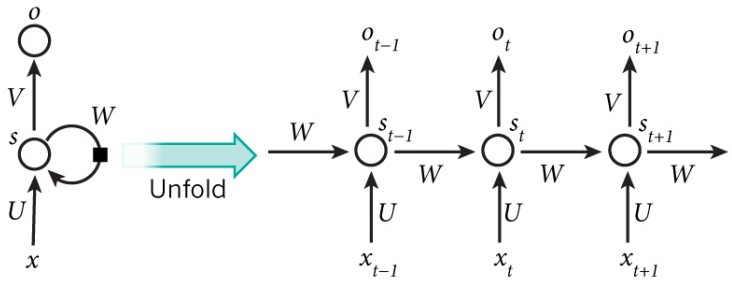

Figure 8.4. A schematic summary on how a recurrent neural network works.

- x_t is the input taken at time t. In the context of this work, it can be seen as an element of a time-series;
- s_t is the hidden state value computed at time step t, and it represent the part that is memorized and re-used. Considering for each neuron weights U, W and ignoring biases value, in general it is computed as follows:

$$s_t = f(Ux_t + Ws_{t-1}), \tag{8.6}$$

  with f activation function. If this process is not clear enough, refer to image 8.4;
- o_t is the output at time t. Usually is strongly related to the hidden state s_t but not always exactly equal: a popular choice is to put

$$o_t = \sigma(Vs_t), \tag{8.7}$$

  where V is a vector weight and the σ is known as the softmax function, defined as:

$$\sigma : \mathbb{R}^K \to \left\{ z \in \mathbb{R}^n | z_i > 0, \sum_{i=1}^{K} z_i = 1 \right\}, \quad \sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}. \tag{8.8}$$

  So the latter is a multi-dimensional function; in fact hidden state s_t can easily be a vector: in this work everything is far easier as only numerical values will be considered.

It is important to note that RNNs are more efficient than most of other neural networks; this is due to the fact that basically no training is necessary as time goes on, as the values of the weights U, V and W are fixed for all the steps. This dramatically reduces the number of parameters learned and allow the whole

process to be way faster. However, this does not mean that any training process is required: more on that later.

One last thing that needs to be considered is the loss function. This is obvious, since one has to consider that the whole optimization process have one very important aim: find the ideal values for hyper-parameters in order that the value for loss function is as low as possible. In a RNN structure this part of the process is defined through another algorithm that allow to train the neural network: it will be analysed in this next section.

8.2.1 Training a Recurrent Neural Network

Despite what was written above, training for a RNN is necessary as it is for all other neural network: this process is however a little bit different from the one studied until now, and it is perpetrated with a different algorithm, called Back-Propagation Through Time (BPTT). This is the algorithm that is used by RNNs to compute gradients, which is one of the most important parts when training a neural network. Besides that, this algorithm is very important in order to avoid the vanishing gradient problem, discovered in 1991 [35] and soon become one of the main problems to solve when approaching any deep learning model.

Considering the process described before, as already stated only the optimization process for the loss function $\mathcal{L}$ is missing: this is important in order to compute the gradient and to optimize the parameters. It has to be remembered that the optimization process means that gradient for $\mathcal{L}$ with respect to parameters U, V and W has to be computed, and then summed up for each time-step (remember to check image 8.4.

Consider from now on $\mathcal{L}_t := \mathcal{L}(y_t, \widehat{y}_t)$. Mathematically all of that means:

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_t \frac{\partial \mathcal{L}_t}{\partial W}. \tag{8.9}$$

By using the chain rule we obtain, for any time j:

$$\begin{aligned} \frac{\partial \mathcal{L}_j}{\partial W} &= \frac{\partial \mathcal{L}_j}{\partial \widehat{y}_j} \frac{\partial \widehat{y}_j}{\partial W} \\ &= \frac{\partial \mathcal{L}_j}{\partial \widehat{y}_j} \frac{\partial \widehat{y}_j}{\partial s_j} \frac{\partial s_j}{\partial W}, \end{aligned}$$

with $s_j = f(Ux_t + Ws_{j-1})$.

This has a very big consequence: it is needed to unroll the RNN in order to complete the training process, and to do that it is necessary to go back to time $t = 0$:

$$\frac{\partial \mathcal{L}_j}{\partial W} = \sum_{t=0}^{j} \frac{\partial \mathcal{L}_j}{\partial \widehat{y}_j} \frac{\partial \widehat{y}_j}{\partial s_j} \frac{\partial s_j}{\partial s_t} \frac{\partial s_t}{\partial W}. \tag{8.10}$$

So in the end the training process for a RNN is the same as a normal training process for a classic neural network. The only differences are that:

- RNN needs to be unrolled before proceeding, so that it can be treated as a normal neural network;
- values for weights U, V, W do not change as t changes, but they are modified globally, making the process itself not as efficient as it may first appeared.

Despite everything, the fact that updating weights requires a back-propagation through time should not surprise as a concept: in fact this is due to the fact that for time-series, in order to predict the next step, you need to know time-step you are in, even more if you have to consider a seasonal process.

8.2.2 The Vanishing Gradient Problem

Equation (8.10) defined before might not be enough to understand the amount of computations needed in order to compute the gradient. Rewriting it as:

$$J := \frac{\partial s_j}{\partial s_k} = \left(\prod_{i=k+1}^{j} \frac{\partial s_i}{\partial s_{i-1}} \right) \frac{\partial s_k}{\partial W} \tag{8.11}$$

gives a better idea of the huge amount of computation one needs to tackle in order to compute matrix J.

Fortunately, in [52] it was proved that

$$\|J\|_2 \leq 1, \tag{8.12}$$

and this is obvious considering that $-1 \leq \tanh(x) \leq 1 \ \forall \ x \in \mathbb{R}$.

This becomes a big problem as $x \to \pm\infty$, because in those cases $\tanh(x) \to 0$ and so the whole gradient vanishes: this means that gradient contribution from far away steps vanishes, becoming zero and not allowing the system to learn any long-range dependencies.

This was a very big problem when RNNs were discovered and started to be popular (they are not the only deep learning models that have this problem: any deep feed-forward NN has it), but now there are lots of ways to tackle it: a very popular one is to use ReLU function as activation, which allows to better control gradient value at each instant, since derivative can only be 0 or 1. This will allow to use a recurrent neural network as described in this section when creating artificial heart beats.

Another idea that had important results is the application of a Long Short-Term Memory (usually called with their acronym LSTM) architecture. It is another recurrent neural network, but it allows to partly remember informations taken a long time ago.

8.2.3 Forecasting with Long Short-Term Memory

It should now be clear how neural networks can be very complicated just considering their structure alone. Then one has to analyse what is happening in each neuron, adding to the complexity of the whole process. This is what make these architectures quite complex, but it also allow them to be very effective when tackling certain tasks.

What happens inside each neuron is the main difference between RNNs and LSTMs: the main goal of the latter is in fact to combat vanishing gradients and try to keep memory of long term informations. This idea was developed thanks to a sort of gating system that can be found in each neuron. Considering σ as the sigmoid function and the $\odot$ element-wise matrix product (also known as the Hadamard product), we have the following set of equations:

$$\begin{aligned} i &= \sigma(x_t U_i + s_{t-1} W_i) \\ f &= \sigma(x_t U_f + s_{t-1} W_f) \\ o &= \sigma(x_t U_o + s_{t-1} W_o) \\ g &= \tanh(x_t U_g + s_{t-1} W_g) \\ c_t &= c_{t-1} \odot f + g \odot i \\ s_t &= \tanh(c_t) \odot o. \end{aligned}$$

Here we have:

- i, f and o that represents the input, forget and output gates for each neuron. They are called gates because thanks to function σ all elements these vectors are compressed between [0,1] $\subset \mathbb{R}$, defining how much of each one will be considered. In particular, i defines the part of the new input that will be taken, f defines the part c_{t-1} that will be remembered and o how much the output will be affected by the newly computed c_t;
- g can be seen as a candidate hidden state that will be multiplied with input state i and then summed with $c_{t-1} \odot f$;
- c_t represents memory state of the architecture at time t, and it represents a combination of what we want to remember from previous states and the newly computed one;
- s_t is the output hidden state.

Through this procedure we have a newly architecture which allow to remember (partly) what was happening even a long time ago, solving for the major part the vanishing gradient problem.

Part III

Experiment and results

Chapter 9

Experimental experience

9.1 Pipeline description

In this experiment we wanted to classify beats using the k-nearest neighbours classifiers in order to state if a patient has some sort of arrhythmia. To improve results, WST are used, so that dimensionality can be reduced and features can be extracted; in that way the classification process is more efficient and effective. Due to privacy issues and the limited amount of data available, two different data augmentation processes were implemented and results among them compared. The pipeline will now be explained in a more detailed way:

- firstly the MIT-BIH dataset was considered: all its characteristics will be talked in depth in section 9.2;
- we will consider only the lead that allows us to extract more information, which means the one with the higher peaks: in our case, it will be lead number one. Signals are centred on the R-peak, with 99 sample points taken before it and 100 after. Since data is sampled at 360 Hz and that each signal is 200 sample points long, it means that each beat represent 0.56 seconds: in figure 9.1 an example of ECGs that are going to be considered.
- raw data was used to train a model able to classify each beat to its class;
- the original split class is displayed in table 9.1. It is clear that the amount of beats classified as Normal (N) is way higher than the number of beats for all the other classes, even combined. This a very known in statistical and machine learning: an imbalanced dataset can in fact lead, in training phase, to a biased classifier, which will tend to wrongly classify elements of other classes as elements of the most represented class (in our case, we are talking about class N).

In order to sort this out it is needed to make a data augmentation process, in order to balance the classes out: before that, as in [26], beats classified as unrecognizable (Q) were dropped. The number of beats classified as Q, as shown in 9.1, is in fact so low that the class itself is not suitable for any data augmentation process; also a number of beats of class N was dropped.

The first process of data augmentation implemented a very naive data augmentation process: each data point of signals for classes S, V and F was perturbed with random white noise $w_t \sim \mathcal{N}(0, 0.05)$; distribution of classes after data augmentation is shown in 9.2. Having done that, it was decided to implement a more advanced data augmentation process, using known signals to train models that were able to generate, through time series forecasting, new realistic beats. Many methods were implemented, with different results; all beats creation process has been evaluated using different metrics, in order to show that these new techniques allowed to have beats that still had the same features as the real ones, while being original. This topic is discussed in section 9.4;

- we made classification using only KNN classifier without transforming data using WST to have a comparison;
- data was passed to WST in order to extract features;
- another classification process is then performed, in order to observe the improvements made after having transformed the dataset.

9.2 MIT-BIH Dataset

The impact of the MIT-BIH dataset can not be underestimated: it was in fact the first dataset available for everyone who wanted to train a model of for evaluation of arrhythmia detectors, and since the first release back in 1980 it has been used for this purpose by at least 500 universities [50]. This dataset is the result of an unbelievable work done by researchers George B. Moody and Roger G. Mark: both of them spent four years, from 1975 to 1979, not only acquiring and digitizing the data, but also annotating every information that could be extracted by each beat. Their work arrived to this kind of completeness because they wanted to make the recordings available for everyone, in order to stimulate the research in this field.

Each row of the dataset represent an half-hour record of an ECG taken to different patients in the Arrhythmia Laboratory of Boston's Beth Israel Hospital (BIH): there are 48 records, taken from 47 different patients (which means that two records are from the same patient), distributed as follows:

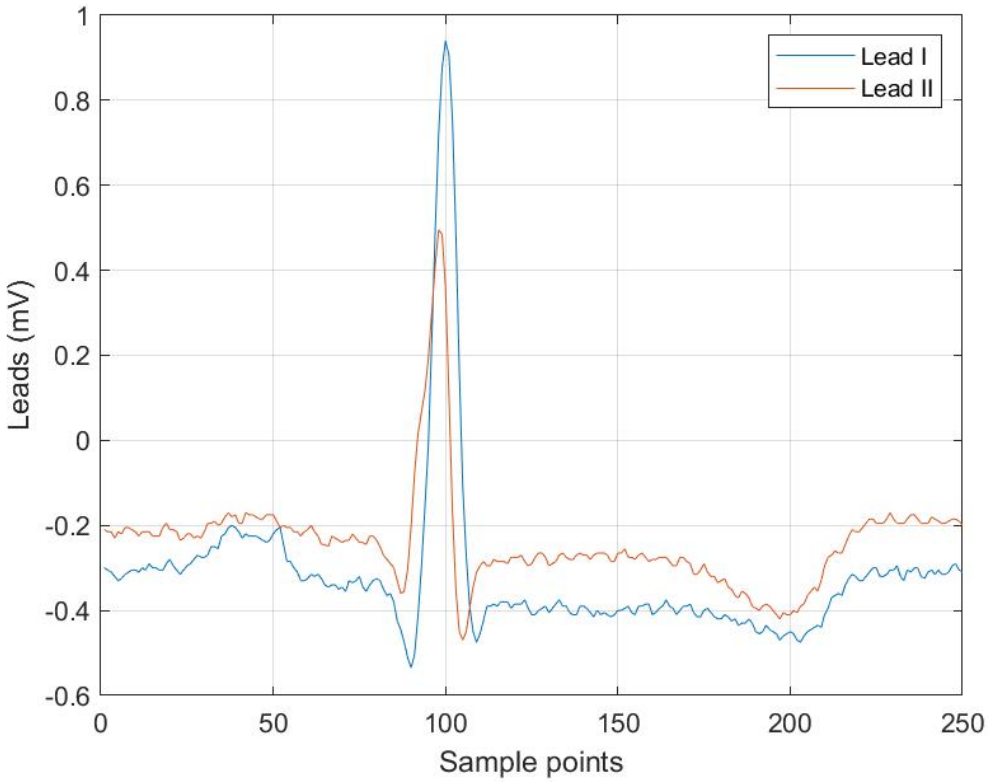

Figure 9.1. Third beat of file 100

- the first 23 were chosen randomly from a collection of 4000 patients with no notable cardiac problems;
- the last 25 were selected in order to include uncommon but clinically important type of arrhythmias: among them, there were four patients with pacemaker, so their beats were artificially paced. These kind of beats are not well represented because pacemakers usually work at frequencies in the kilohertz range, far above the passband of the recorders, which was set between 0.1 to 100 Hz, which worked really well for the major part of the recordings.

For each record, two leads were saved: one channel is usually the modified limb lead II (MLII), obtainable by placing the electrodes on the chest, while V1 is usually saved on the other channel (also V2, V4 and V5 are used, depending on the patient.

The recordings were digitized at 360 Hz (samples per second in this context; it means that each record is 650000 sample points long) with an 11-bit resolution over

a 10 mV range. This stage took the longest time by far, due to the tools available at the time, and it had to be done with some compromises: for example, the digitization rate was chosen to be 360 Hz because data could not be written faster, as the RAM available at that time was just 24 kilobyte, which made impossible the possibility of storing the data.

Once everything was done, each half-hour tape was printed in a 46 meters paper chart, which included both leads. Each chart was then given by two cardiologist, who began annotating each beat independently and then worked together in order to resolving discrepancies by consensus: only 33 beats (concentrated in six files) remained unclassified because an agreement could not be found in those cases.

By the end of this monumental work, approximately 109 000 beats were saved and annotated, originally divided in 21 classes. In the entirety of the registrations, a loss of signal so significant that both channels could not be recorded happened only only seven times, and all these cases combined have a total duration of ten seconds: so it can be stated that the impact of these episodes was absolutely negligible.

9.3 The Wavelet Toolbox™ in MATLAB®

Parameters tuning for WST is absolutely fundamental in order to maximize feature extraction and, with that, the effectiveness of the classification scores. The Wavelet Toolbox™ [61] implemented in MATLAB® (version 2018b) was used to effectively transform the data using Wavelet Scattering Transform.

The main reason why this toolbox was chosen is because it allows the user to choose parameters that other packages just compute automatically: so this toolbox can be a little more complicated to set up than others, but way more precise, and especially way less unpredictable and more adjustable by an expert user. The most important function is the one that physically creates the transform, which depends on the following parameters:

- signal length L: simply the length of the signal in sample points. In our case obviously $L = 200$ sample points was used;
- sampling frequency f: frequency of the signal, which is the number of sample points needed in order to cover one second. As already stated, data is sampled at $f = 360\ Hz$, so that value is passed to the function;
- quality factors Q: represents the amount of wavelet per octave. The maximum amount acceptable is 32, and of course this value must be greater than one and has to be a integer. This can also be a vector, used in case there is filter with more than one order: in that case, number of wavelet has to be decreasing. Standard choice is $Q = [8, 1]$, which means that eight wavelet are

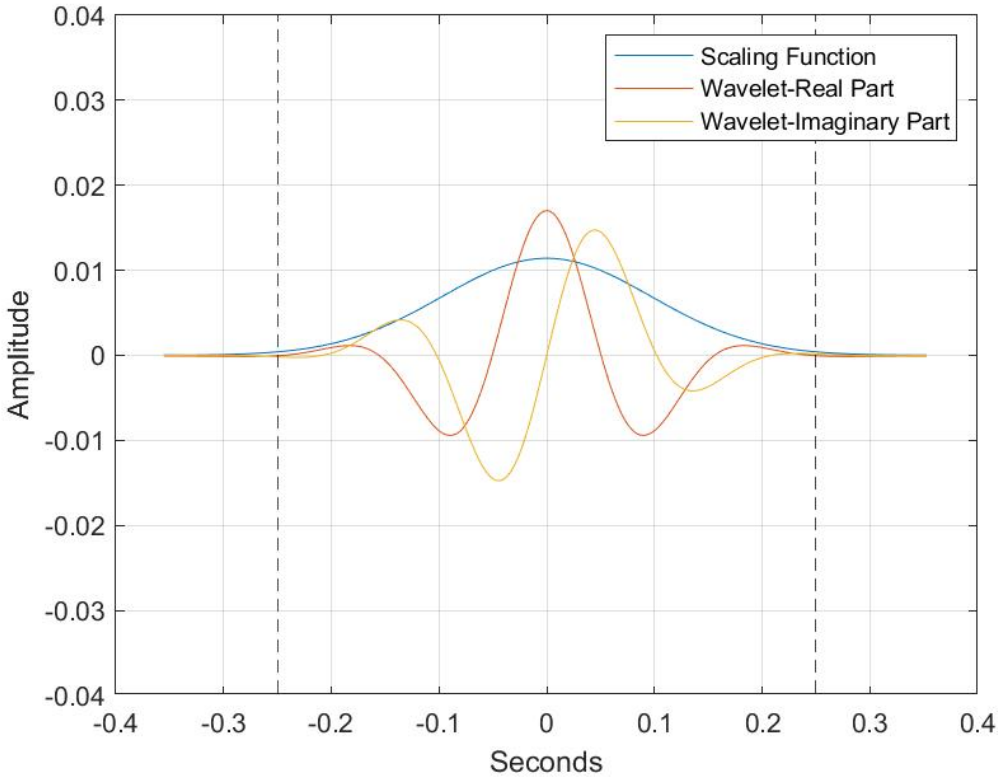

Figure 9.2. Plot for scaling and real and imaginary first order functions

used for the first order filter, and just one for the second order: their plots are displayed in 9.2 and **??**;

- invariance scale I: specifies the translation invariance of the scattering transform and it is measured in units of seconds. It is particularly important because the network is constructed to be invariant to translations (the main property of the wavelet transformations, as already stated multiple times) up to the invariance scale; in 9.2 the invariance window is represented by the two black-traced vertical lines. This value has an upper-bound dependently to the sampling frequency f and the length of the signal processed L: both these values have to be passed in order to compute this upper-bound, which is: $I < \frac{L}{f}$. In our case $I = 0.5\ s < \frac{L}{f} = \frac{200}{360\ Hz} \approx 0.56\ s$ so the upper bound is satisfied.

Natively the Wavelet Toolbox™ implements the Gabor wavelet, defined in 1947 by physicist Dennis Gabor [44]. The idea is quite simple: in place of considering the

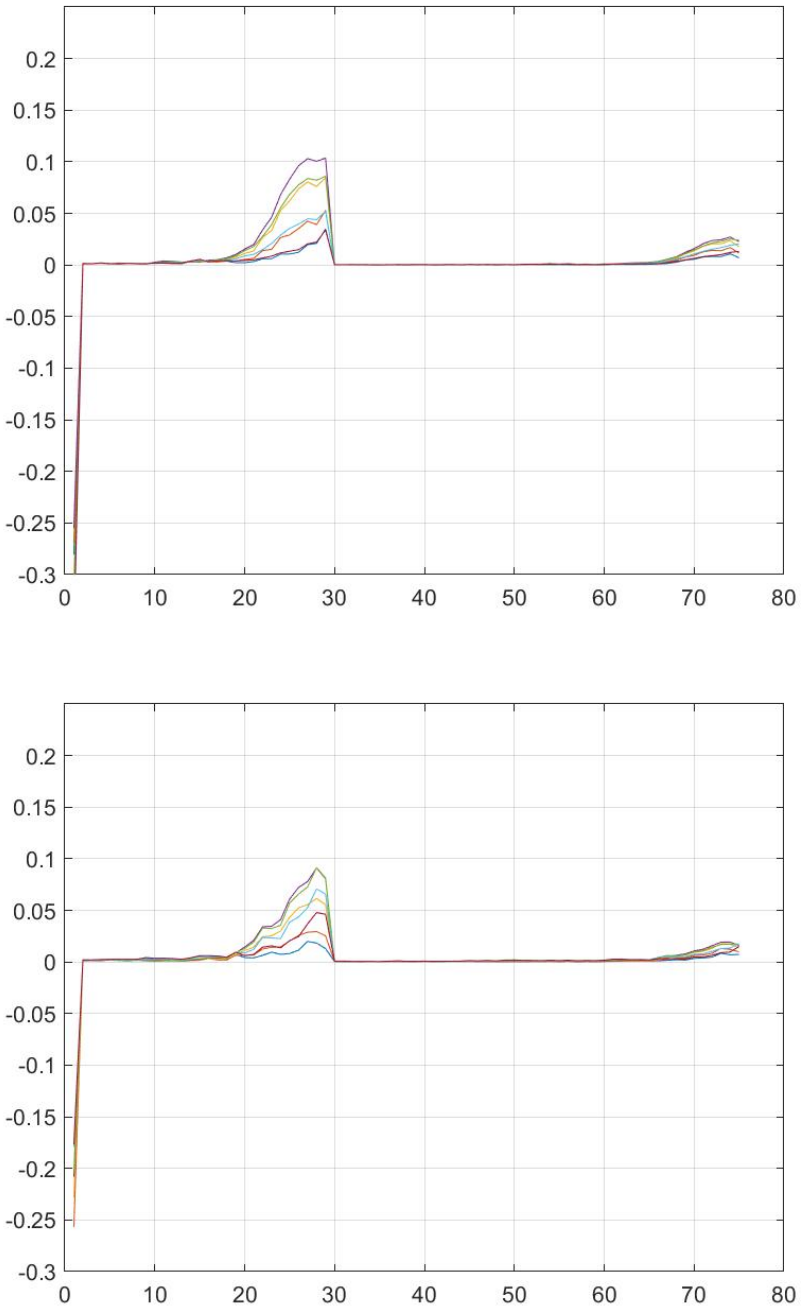

Figure 9.3. Plot for a transformed beat of type N (above) and F (below).

entire domain of the function, its transformation is computed only in an interval of size σ_t (it depends on the invariance window) centred in a certain point u. So we don't consider just the function g but:

$$g_{u,\xi}(t) = g(t-u)e^{i\xi t}, \tag{9.1}$$

where ξ is the frequency of the signal. Therefore, its Fourier transform is defined as followed:

$$\widehat{g}_{u,\xi}(\omega) = \widehat{g}(\omega-\xi)e^{-iu(\omega-\xi)} \tag{9.2}$$

At this point it is possible to define the window Fourier transform defined by Gabor:

$$Sf(u,\xi) = \int_{-\infty}^{+\infty} f(t)g(t-u)e^{-i\xi t}dt. \tag{9.3}$$

Finally, if we put

$$g(t) = \frac{1}{(\pi\sigma^2)^{\frac{1}{4}}} \exp\left(-\frac{t^2}{2\sigma^2}\right), \tag{9.4}$$

it is possible to define the Gabor wavelet:

$$\psi(t) = g(t)e^{-i\xi t} \tag{9.5}$$

$$= \frac{1}{(\pi\sigma^2)^{\frac{1}{4}}} e^{-\frac{t^2}{2\sigma^2}} e^{-i\xi t}. \tag{9.6}$$

It is important to note that in general it is not particularly clever to consider any order of scattering coefficients after the second: this is due to the fast decay of these coefficients. In fact, as said in [10], more than 98% of the energy of a wavelet is carried by coefficients of order zero, one and two: in order to improve efficiency of the computation process, Wavelet Toolbox™ automatically computes only these orders for wavelet coefficients.

After transforming the beats a new decomposed signal has been obtained: it is composed by 7 time windows and 75 paths each, as it can be seen in image 9.3, so that from a computational point of view it needs to be represented as a three order tensor (which is a three-dimension matrix) of dimension: number of beats considered, number of path, number of time windows.

In order to apply some data reduction and also make KNN useful in this scenario, only one time window has been considered: for each experiment only the time window that guaranteed on average the highest maximum among all was considered, and for all cases it was time window number four.

9.4 Data augmentation and the sktime package

For every machine learning task, data preprocessing is the real key in order to compute it successfully, and to do it well a deep investigation on the data that is going to be used is necessary.

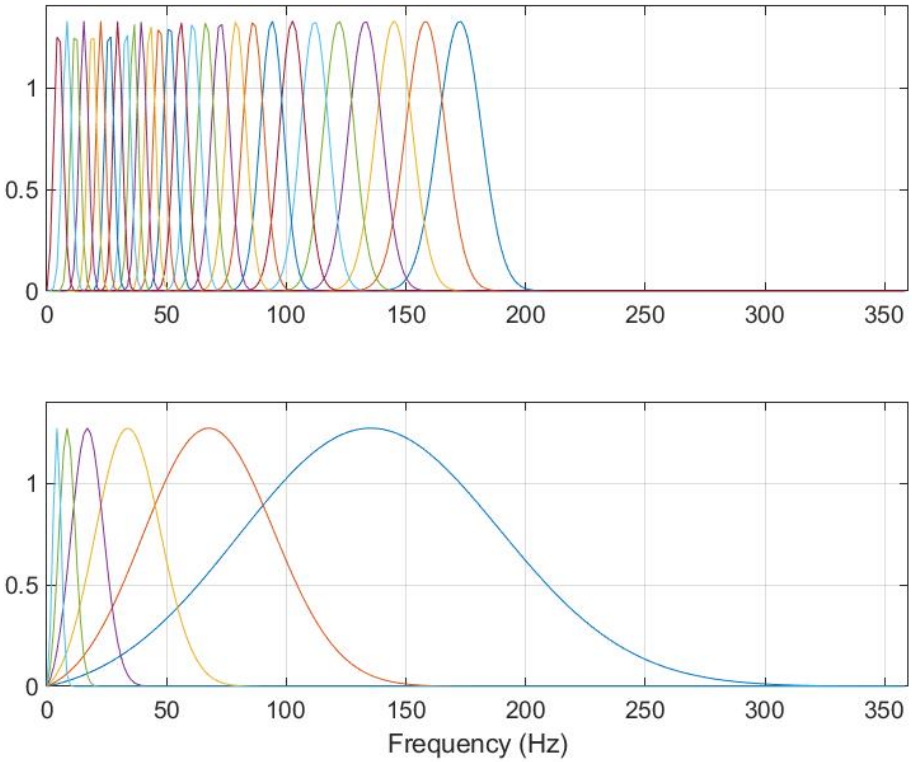

Figure 9.4. Filter-banks for first and second order with $Q = [8, 1]$

The MIT-BIH dataset has a relative shortcoming: it is very much unbalanced, and almost 75% of all the beats in the original classes distribution are labelled as normal (N). Following indication in [26], it was decided to group together a lot of the original classes, since all of them had a number of elements negligible with respect to the number of elements of class N. So it was decided to work with five classes: F, N, V, S and Q, following the ANSI/AAMI ECCE:1998 standard [58]; however, as shown in 9.1, the amount of unclassifiable beats (Q) is very low, so it was decided to drop them altogether.

After that, now almost 90% of all the beats saved are labelled as normal; this huge skewness make the dataset unusable, since a model fitted with these data would be so used to see normal beats that it would label abnormal beats as normal ones when it sees them. This is a very well known problem in the machine learning community, and there are ways to solve it: one of them is data augmentation. Two different concept of data augmentation will be described and confronted, stating also the difference between a more naive approach and a more complex but effective

Class	Number of beats	Percentage
F	802	0.7964%
N	90097	89.4681%
Q	15	0.0149%
S	2781	2.7616%
V	7008	6.9590%

Table 9.1. Distribution of beats classes before data augmentation.

one.

The first process implemented is pretty simple: each point of a certain beat is perturbed with a random noise, modelled as a random normal variable $w \sim \mathcal{N}(0, 0.05)$. This process was repeated for classes V, F and S until each category could count as much as 90 000 beats, as shown in table 9.2; in order to have a more robust model, beats were chosen randomly and uniformly before being perturbed, so that there were no bias on our side during this phase: in figure 9.5 a natural beat and its perturbed version are shown.

Class	Number of beats	Percentage
F	90000	25.0000%
N	90000	25.0000%
S	90000	25.0000%
V	90000	25.0000%

Table 9.2. Distribution of beats classes after the naive and ETS data augmentation.

The second process is very different, and far more complex: it is based on the idea of time series forecasting. ECGs and signals in general can be seen as time series; for each sample point (which are basically moments in time) it is possible to find a value describing electrical activity for cardiac muscles. This opens up a very interesting possibility: using a certain number of beats to train a model capable of creating a new, artificial beat very similar to the natural ones, thanks to the properties of the forecasting methods for time series. Reasons for that are very heterogenous:

- first of all, having more realistic beats in place of simple perturbations of the original one allow us to train a more robust model, and it will be way more used to work with realistic ECGs. Even if scores for the classifier with this data augmentation process would be lower, results would probably be a lot more significant just because model training is done with a realistic dataset,

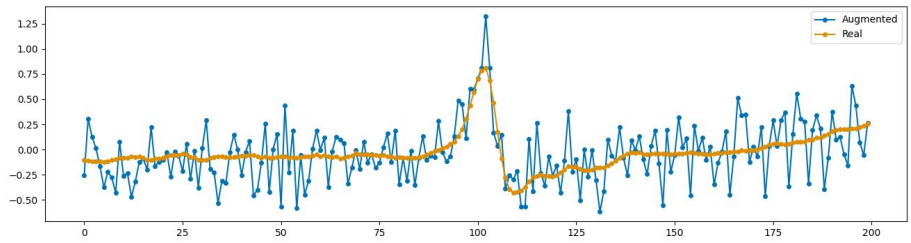

Figure 9.5. A real beat of class S and the one obtained by adding a noise $w_t \sim \mathcal{N}(0, \sigma^2), \sigma^2 = 0.05$.

giving a lot more robustness to the model itself when different data were tested there;

- in secundis, and this is another general motivation for this data augmentation process, there are a lot of concerns about privacy. We are talking about very sensible data, that even if taken anonymously are subject to risk of theft from databases of hospitals: this is a very widespread practice already;
- finally, there is a enormous lack of data in this field. For lots of reasons, one of them being the second point of this dotted list, collecting this kind of data is not only difficult, but also very time-consuming both for the collecting part but also because it is necessary to digitalize it, as explained in 9.2. It is out of question that now this process is definitely more feasible than in 1980, but a lot of concerns and difficulties are still there.

For all these reasons, having the possibility to create easily and in relative short time great quantities of artificial but realistic beats can surely help in this very unique context. Just to clear things up, beats obtained with the first method will be called *augmented*, while the ones obtained with the second method will be called *created*.

In order to implement some useful forecasting methods for time series, it was decided to use the sktime library, which is an open source Python framework for machine learning and artificial intelligence with time series.

The need of creating a new Python package for time series was created because of the lack of an unified framework that grouped all functions and methods that are unbelievably useful when working with all types of time series: we are not talking only about forecasting, but also for analysis of the series, like its seasonality, stationarity and its autocorrelated function. The framework can also be used in order to solve classification and regression problems with time series.

The framework was used to successfully implement some forecasting techniques, not only among the classic ones but also using modern approaches, based on deep

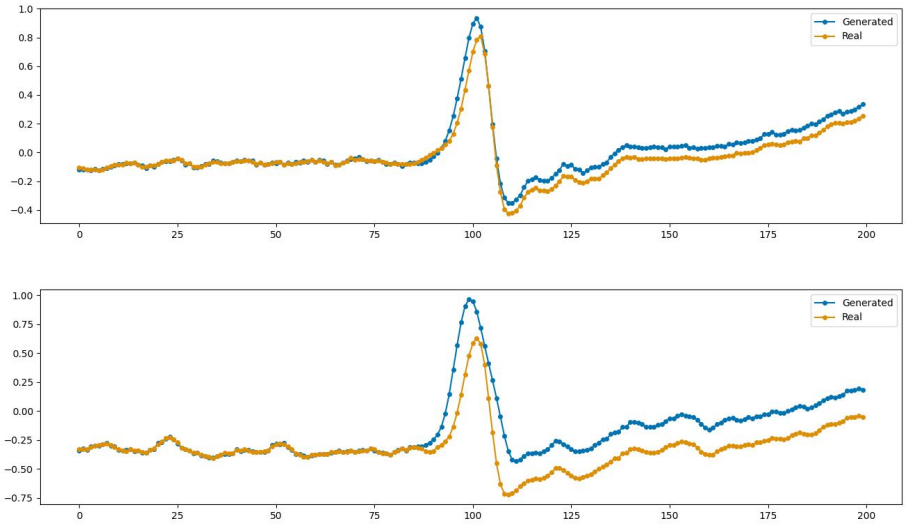

Figure 9.6. A real beat of class S with one generated with ETS (above) and with ARIMA (below).

learning: they were described in chapters 7 and 8. The entire list of used methods follows, together with the ideal parameters for each of them:

- exponential time smoothing (ETS) is implemented in sktime in such a way that all ideal parameters are automatically computed after having passed the training set. Eventually the `AutoETS` function can be set up manually: particularly, it was specified that the model would have been the additive one. Moreover, for the nature itself of ECGs, no trend has been added on the model;

- ARIMA is another classic forecasting technique that is implemented by using an automated function that allows to compute the ideal parameters in order to fit the beats that form the training set;

- neural forecaster LSTM and RNN were considered and used to create new beats, too.

It is necessary to specify that in time-series forecasting one of the main tasks is to recover a distribution for the data in the training-set, in order to evaluate an estimation for the test-set. Doing that for ECGs proved to be difficult, since one can divide beats in two different moments: peaks and the rest. These two moments are very different from a mathematical point of view, and they give very different informations.

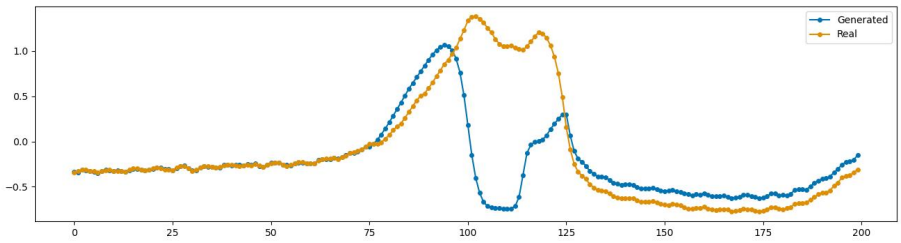

Figure 9.7. A real beat of class V generated with LSTM.

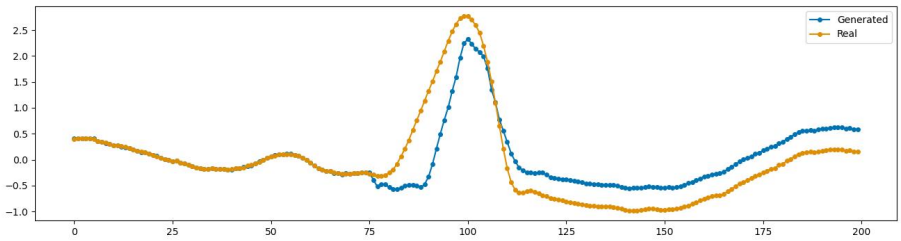

Figure 9.8. A real beat of class F generated with RNN.

For this reason, in all the models considered, it was decided to create only the peaks for each beat; for what concerns the rest of the beat, it was decided to just perturb it with a Gaussian noise with a far smaller variance then the one used in the augmented beats: in particular in this case $w_t \sim \mathcal{N}(0, 0.005)$.

This process is very successful for both classic and neural forecaster, and it can be seen in figures 9.6, **??** and 9.8. All beats created are not only realistic, but they are original too. This is a very important goal of this whole work, since it is now possible to create a great amount of realistic beats using a relatively small quantity of resources, and without any risk of breaching patients privacy.

From an efficiency point of view, there is a wide spread of results:

- ETS model is the fastest of them all, as it manages to create a beat in less than a second: to be more precise, on average, ETS model needed 0.656 seconds in order to create one single beat: this result can be improved using a more efficient processor as the one inside my PC. For this reason it was decided to create 90 000 beats per class, for a grand total of 270 000 beats;

- ARIMA model is the slowest since it needs almost a minute to create each beat: to be precise, 46.491 seconds are needed on average per beat. As a consequence only 1 400 beats per class have been considered;

- LSTM and RNN are very similar in terms of performances, since the both need two to five seconds in order to create a beat: to be more precise, the average creation time resulted to be equal to 4.93 seconds for LSTM and 2.727 seconds for RNN. Thanks to HPC staff in PoliTo, it was possible for me to have access to one A40 GPU with an incredibly high computational power, which allowed the training of these models to be so efficient.

9.5 Classification

Finally, it is necessary to remember that this experiment is basically a supervised classification problem; from a mathematical point of view it means that each signal can be represented as a point in a space $\mathbb{R}^n, n$ equal to the length of the signal (so in this case $n = 200$). A group of points relatively close to each other creates a cluster and, usually, they have the same label, which is known in the dataset: in those cases the problem is said to be supervised.

The point of this task is to train a model with a portion of the data, called the training set, that is able to recognize the right label for data that the model has not been trained on and without knowing labels: the latter is called the test set. There are many different popular models that get this task done, some of them are conceptually very simple, like KNN, which is the only one that was used in this work. Of course other choices. usually more complicated but sometimes also more effective, are available, like SVM or neural classifiers.

For what concerns k-Nearest Neighbors (KNN), it is a popular and very simple method to implement for classification tasks. Each signal is represented as a point in a space $\mathbb{R}^n$, and when points of the test set are given to the model, label is given only depending on the label of the k-nearest points: when more than one label is present, label is chosen with a majority voting system, with each vote having an inverse weight with respect of distance to the point considered. This means that the further the point, the less its vote will count; in our case, $k = 4$ was chosen.

Chapter 10

Results and discussion

10.1 Beats creation

10.1.1 Metrics used

First of all different metrics have been used in order to show how the new beats were very different from the original ones, while preserving the main features for each class of beats. In particular, for each beat created, it was compared with one real beat, that at that moment composed the test set for that creation process. Here the entire list.

Definition 10.1.1. Consider a metric space S, two curves A, B and a parametrization of them $\alpha, \beta : [0,1] \to [0,1]$ such that these are both continuous, non-decreasing, surjective. Then the **Fréchet distance** between A and B, which we can denote as $F(A, B)$, is defined as:

$$F(A, B) := \inf_{\alpha,\beta} \max_{t\in[0,1]} \{d\left(A(\alpha(t)), B(\beta(t))\right)\}, \tag{10.1}$$

where d is the distance metric of space S.

From a less formal point of view, it can be described as a metric which compares two curves, describing how much they are similar to each other: it also takes into account the location and the ordering of each point in the curve. In our case d can be seen as the Euclidean distance, since we are modelling the ECGs as they exists in a $\mathbb{R}^n$ euclidean space, where n is the signal length.

Definition 10.1.2. Consider two vectors A, B, each one composed of n components. Then their **cosine similarity** $S_C(A, B)$ is defined as follows:

$$S_C(A, B) := \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{10.2}$$

where of course $A \cdot B$ represents the scalar product between those two vectors and $\|\cdot\|$ compute its norm.

It is interesting to note that this metric exists in an interval of $[-1, 1]$. This means that:

- $S_C(A, B) = 1 \Rightarrow A = B$;
- $S_C(A, B) = 0 \Rightarrow A \perp B$;
- $S_C(A, B) = -1 \Rightarrow A = -B$.

This is the only measure among the ones considered that has bigger values when signals are closer to each other.

Definition 10.1.3. Consider two random variable u, v with different distributions. Then it is possible to define the **Wasserstein-1 distance** between them, defining it as:

$$l_1(u, v) = \inf_{\pi \in \Gamma(u,v)} \int_{\mathbb{R}\times\mathbb{R}} |x - y| d\pi(x, y), \tag{10.3}$$

where $\Gamma(x, y)$ is the set of probability distributions on whose marginals are and on the first and second factors respectively. For a given value x, $u(x)$ gives the probability of u at position x, and the same for $v(y)$.

The motivation behind the decision of including it is quite explicit: aim of time-series forecasting is to recover a probability distribution, in order to allow the forecasting for future values. Computing this distance allow to understand better if a certain beat is original (if its distance is bigger than the one computed by the naive augmentation process) or not.

Notice that a generic Wasserstein-p distance can be defined and implemented, but it this case the one implemented in SciPy was used [29]. For this reason only the case $p = 1$ is considered.

Finally, there is the Dynamic Time Warping (DTW) distance, which is a metric used to compare the similarity of two time series by measuring the minimum distance between them after aligning their features. Opposite to the ones defined until now, there is no definition, as it is an heuristic algorithm used to compare two different time-series. To analyse it deeper, check [56].

10.1.2 Beats similarity

Results are summarized in table 10.1.

Without any surprise, results show that the naive augmentation method creates the more similar beats. This means that this method might be useful to have lots of beats in a very short time, but it fails to create original and realistic beats

altogether. All the other methods allow that, managing to preserve characteristic for each class of beats while creating new ones: this might be very useful if one needs loads of new beats, considering how much resources are needed in order to acquire them.

Method	DTW	Wasserstein	Cosine	Fréchet
Naive	**3.23899**	**0.09107**	**0.87219**	**3.53172**
ETS	4.28596	0.26573	0.75039	6.06445
ARIMA	7.22731	0.43715	0.52765	9.39777
LSTM	6.24350	0.37540	0.53392	9.11308
RNN	5.92522	0.35729	0.56174	8.80935

Table 10.1. Average metrics value for each augmentation method. Bolded values indicates closer beats to the real ones according to each metric.

10.2 Classification with WST

Classification problems are one of the most popular kind of tasks in machine learning and deep learning. It consists in two phases:

- training, during which a consistent portion of available data is used in fit the data and find the optimal parameters for the classifier that is going to be used;
- testing, during which the rest of the data is labelled by the classifier.

In our case, since during the training phase the classifier can see true labels, it is called supervised classification.

10.2.1 Metrics used

In order to have a measure of the results obtained, some classical metrics, often used in machine learning classification problems, have been implemented. In order to define and understand them, consider a generic class A and its element; after the training and the testing processes, every element of said class can be labelled as:

- true positive (TP), when an element of class A has been labelled correctly;
- false positive (FP), if an element which is not in class A has been mistakenly labelled as A;

- false negative (FN), an element of class A which was not recognized as one;
- true negative (TN), an element which is not in class A and was correctly labelled outside that set.

Now the following can be defined.

Definition 10.2.1. Consider a supervised classification problem, and consider also a generic class of elements, called A. Then the **accuracy** of said class is defined as:

$$\mathrm{ACC}_A = \frac{TP + TN}{TP + TN + FP + FN}. \tag{10.4}$$

It basically count the number of correct labelled elements, and then divides everything for the number of all elements considered.

Definition 10.2.2. Same as above. The **precision** metric is defined as

$$\mathrm{PRE}_A = \frac{TP}{TP + FP}. \tag{10.5}$$

It gives an idea of the precision of the classifier with respect of a certain class.

Definition 10.2.3. Considering the same problem, the **sensitivity** metric is defined as

$$\mathrm{SEN}_A = \frac{TP}{TP + FN}, \tag{10.6}$$

while the **specificity** metric is defined as

$$\mathrm{SPE}_A = \frac{TN}{TN + FP}. \tag{10.7}$$

10.2.2 Results of the creation and classification processes using naive augmentation

These results were obtained after having crafted 270 000 abnormal beats by using the naive augmentation technique, which means that a random noise $w_t \sim \mathcal{N}(0, 0.05)$ was added to each sample point. For each class (V, S, F) 90 000 were considered, and 90 000 normal beats (N) were randomly sampled among the ones already available.

Results are displayed in table 10.2 and image 10.1.

It is clear that this method is pretty good in classification, even if it is not perfect neither in classify normal nor abnormal beats. Besides, after this process has finished all one has is a dataset full of unrealistic and noised beats.

Metric	N	F	S	V	Averages
Accuracy	0.95054	**0.96671**	0.93679	0.94370	0.9494
Precision	**0.97241**	0.92056	0.82973	0.89234	0.9038
Sensitivity	0.82559	**0.94871**	0.94007	0.88112	0.8989
Specificity	**0.99219**	0.97271	0.93570	0.96456	0.9663

Table 10.2. Average metrics value for each class using naive generated beats. Bolded values indicates highest class value for each metric.

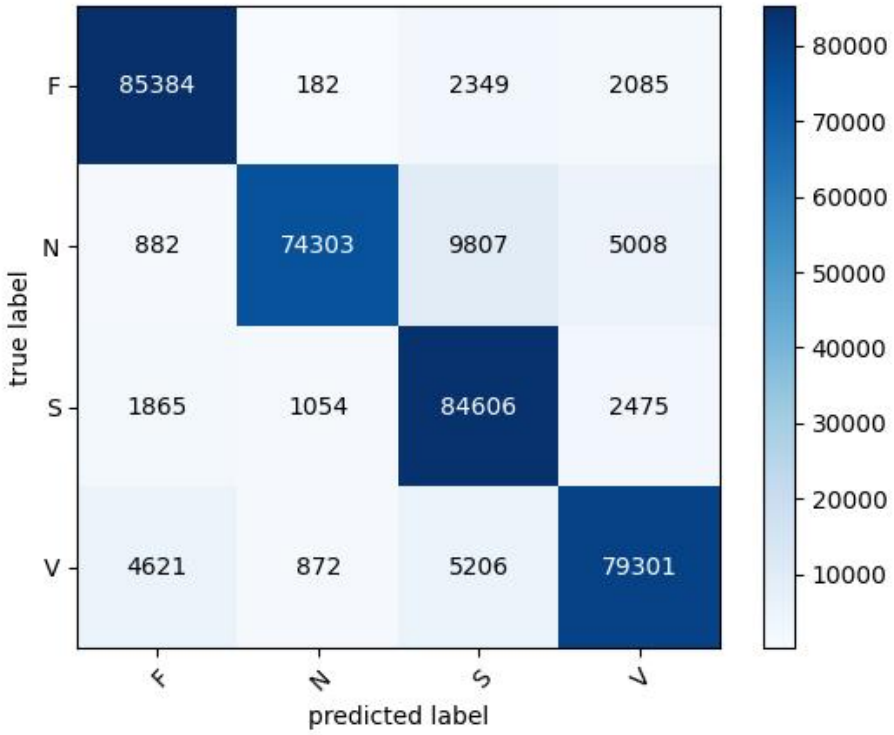

Figure 10.1. Confusion Matrix for naive generated beats.

10.2.3 Results of the creation and classification processes using ETS technique

These results were obtained after having crafted 270 000 abnormal beats using the exponential time smoothing (ETS) technique. As before, 90 000 beats for each class (V, S, F) were considered, with 90 000 normal beats randomly sampled

among the ones already available.

Metric	N	F	S	V	Averages
Accuracy	0.88257	**0.95995**	0.94512	0.95046	0.9345
Precision	**0.89608**	0.87375	0.83450	0.88534	0.8724
Sensitivity	0.59449	**0.98164**	0.97356	0.92116	0.8691
Specificity	**0.97681**	0.95272	0.93564	0.96023	0.9564

Table 10.3. Average metrics value for each class using ETS generated beats. Bolded values indicates highest class value for each metric.

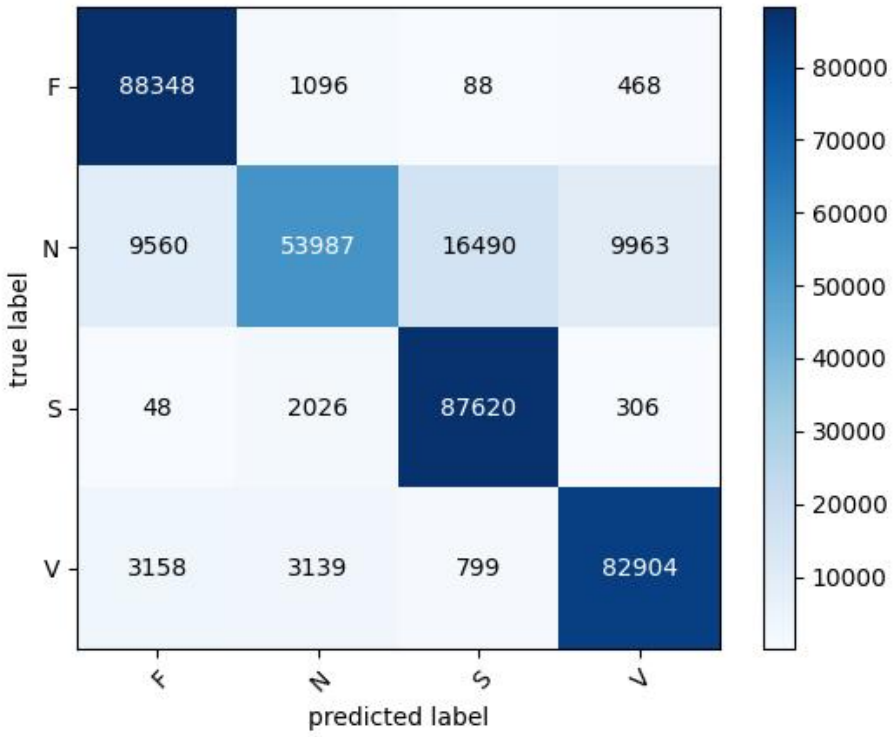

Figure 10.2. Confusion Matrix for ETS generated beats.

In table 10.3 and figure 10.2 all results referring to ETS crafted beats have been summarized.

First thing that stands out is the very low sensitivity on normal (N) beats. This might be as a consequence of the fact that for all other classes only created beats

Metric	F	S	V	Averages
Accuracy	0.98299	**0.99253**	0.97899	0.9848
Precision	0.95835	0.98652	**0.98793**	0.9776
Sensitivity	**0.99208**	0.99112	0.94856	0.9773
Specificity	0.97844	0.99323	**0.99421**	0.9886

Table 10.4. Average metrics value only considering the three classes with arrhythmia using ETS generated beats. Bolded values indicates highest class value for each metric.

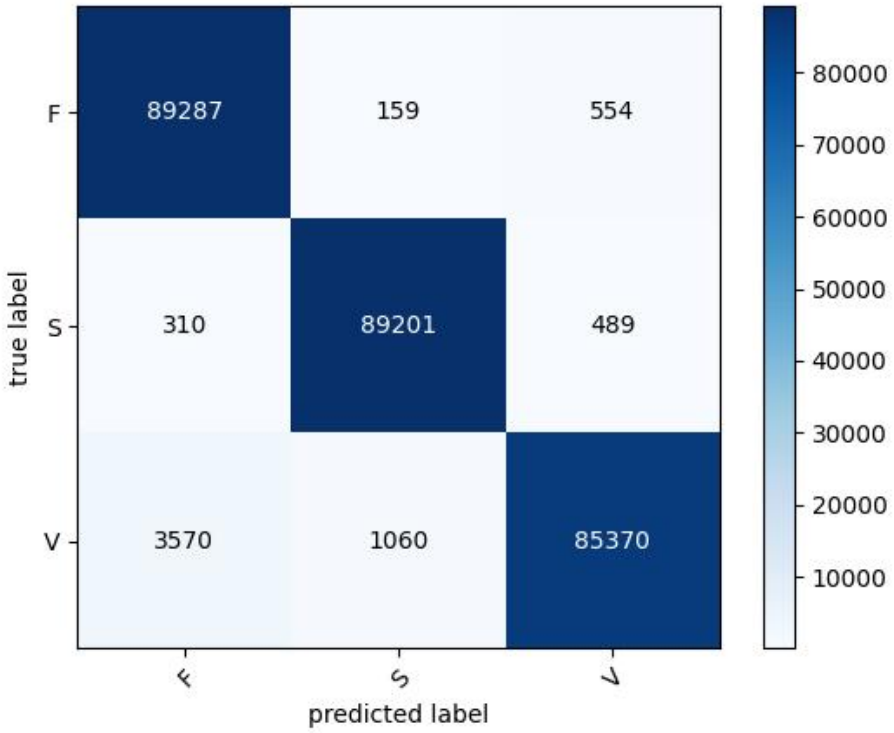

Figure 10.3. Confusion Matrix for ETS generated beats, using only the three classes with arrhythmia.

have been considered, while all normal beats were natural ones. This has another major consequence: it basically means that the number of false negative is very high for class N, and it can be said that our KNN classifier is very cautious, as it labels lots of normal beats as abnormal.

In the real world, this would lead to a significant growth in cardiac visits, which

would ultimately carry to a reductive number of abrupt failures and problems in the long term. On the other hand, a significant increase in medical expenses would be required by the national health service.

Finally, it can be noted that overall results are lower than the one obtained with augmented beats: this should not cancel the significant pros of this new augmentation process, since it allows to have a dataset full of realistic heart beats, all having some kind of abnormalities. This might me a key result in future research works, since it is known how difficult it is to acquire such data. Besides, results are lower mainly because of the low results for class N; this is far from optimal, but has another interesting consequence, as this pipeline allow KNN to be very precise when it has to distinguish beats with abnormal features. This can be an interesting idea for future works, since this simple model already allow to have very good performances once an irregularity has been found, since it can help to understand with a very good precision what type of cardiac disease is displayed and, consequently, how it should be treated: classification itself was done after transforming more realistic data, while the first one was done on non-realistic beats obtained by perturbing the original ones: this should be taken into account as well.

Lastly, because of the results obtained, it was decided to show what this model can do if it just has to determine what kind of cardiac disease is revealed by ECG. Basically, a Bayesian approach is used, and the results in 10.4 and 10.3 are referred to the type of beats this model can correctly recognize, after knowing that an arrhythmia has been diagnosed.

As shown, results are very good, and this model might be an important tool for cardiologist in order to lower their work load, since it can distinguish very effectively different kind of cardiac disease, besides of generating realistic beats of said diseases too.

10.2.4 Results of the creation and classification processes with ARIMA

Results obtained with ARIMA data augmentation process are compromised by the low amount of beats that were created: in fact 1 400 beats per class has been considered, for a total of just 5 600 beats, which is a very low number in this field. This is a consequence of the inefficiency of ARIMA model with respect to all other models used. It has to be remembered that ARIMA needs almost a minute in order to create one single beat, while ETS just needs 0.656 seconds and both RNN and LSTM require just two and five seconds per beat, respectively: this is probably due to the fact that `AutoETS` function perform an automatic search of the ideal parameters for ETS model, which is very light with respect of ARIMA; function used to implement the latter, `AutoARIMA`, does basically the same, but

needs a lot of time in order to find the ideal parameters.

Despite everything, a classification process was carried out and unfortunately the number of beats are not enough in order to have significant results. Classification is still better for abnormal classes, but in general results do not satisfy minimum expectations for tasks like this one. If this process could be made more efficient, then an higher number of beats would be created, and results would be far better than the one shown in table 10.5.

Metric	N	F	S	V	Averages
Accuracy	0.77679	0.86464	0.81893	**0.87214**	0.8331
Precision	0.56168	0.70240	0.62468	**0.77360**	0.6656
Sensitivity	0.48786	**0.79571**	0.69071	0.69071	0.6663
Specificity	0.87310	0.88762	0.86167	**0.93262**	0.8888

Table 10.5. Average metrics value for each class using ARIMA generated beats. Bolded values indicates highest class value for each metric.

It needs to be noted that in order to improve results several for parameter k of classifier KNN were tried, but no improvement was observed even putting $k = 1,2,3$. So the decision was to come back to the standard value $k = 4$.

10.2.5 Results of the creation and classification processes using neural techniques

Classification with beats created with LSTM and RNN do suffer from the low amount of data obtained, and results are pretty much comparable for both methods considered: even by increasing parameter $k = 10$ there is no visible improvement, so it was decided again to leave it as $k = 4$. Results for LSTM are shown in table 10.6 and figure 10.2.5, while results obtained with RNN are shown in table 10.7 and figure 10.2.5. These results prove how important it is to work with big quantities of data in order to have success when dealing with deep learning tasks.

Metric	N	F	S	V	Averages
Accuracy	0.77550	**0.84369**	0.79025	0.83537	0.8112
Precision	0.55849	**0.68870**	0.56772	0.68045	0.6238
Sensitivity	0.48700	**0.68388**	0.67488	0.64387	0.6224
Specificity	0.87167	0.89696	0.82871	**0.89921**	0.8741

Table 10.6. Average metrics value for each class using LSTM generated beats. Bolded values indicates highest class value for each metric.

Metric	N	F	S	V	Averages
Accuracy	0.78733	**0.83544**	0.78547	0.83186	0.8100
Precision	0.58342	0.67457	0.55725	**0.68463**	0.6250
Sensitivity	0.52222	0.66033	**0.69056**	0.60711	0.6201
Specificity	0.87570	0.89381	0.81711	**0.90678**	0.8734

Table 10.7. Average metrics value for each class using RNN generated beats. Bolded values indicates highest class value for each metric.

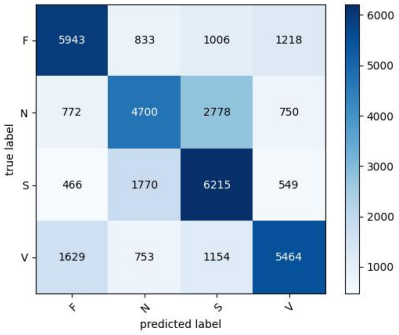

Figure 10.4. Confusion Matrix for RNN method.

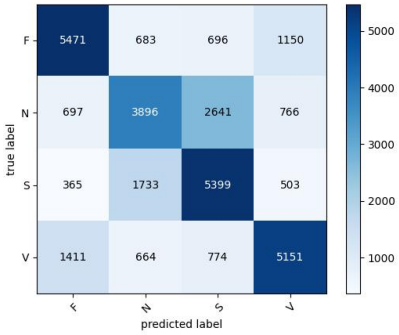

Figure 10.5. Confusion Matrix for LSTM method.

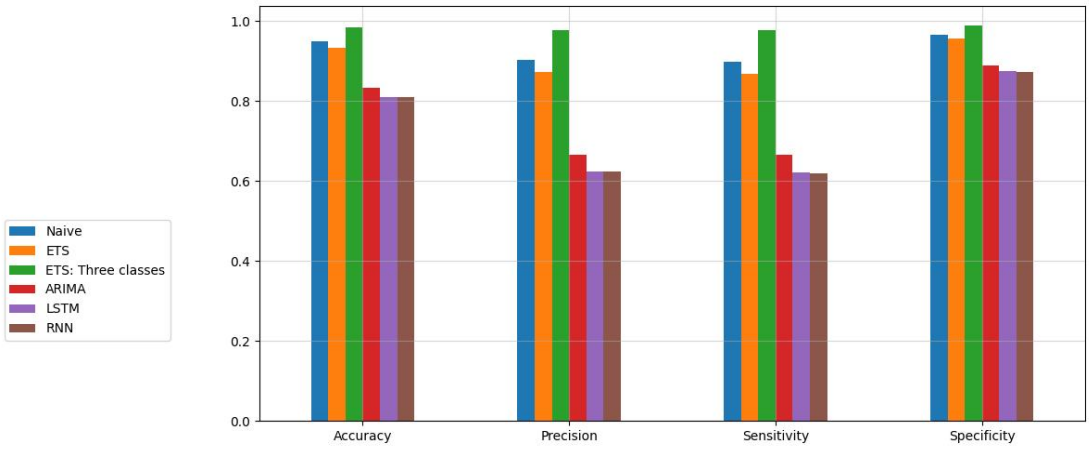

Figure 10.6. Histogram with all average results for all augmentation techniques seen in this work.

Besides that, it has to be considered that ETS was implemented by using an automatic function, which allow the process to be as efficient as possible. In this

case this was not possible, since LSTM and RNN do not have such possibilities and have to be manually set-up. This model allow to have some original beats, but fails to be as efficient as it should in order to have the needed amount in order for the beats to be classified in a correct way.

Lastly, in image 10.6 there is a full recap on the average score for each metric and for each augmentation technique that was used in this work. It is interesting how both accuracy and sensitivity scores do not go below 80% even for the worse methods, so it does show how there is promise for all of them; on the other hand, only ETS method seems to be sufficiently precise, as shown from precision and sensibility scores. In order to understand these results, it has to be remembered that the number of beats is different for each method, as vastly discussed in this chapter, and results shown in this image do vary a lot because of it.

Conclusions and future works

In this work, a much needed analysis on Wavelet Scattering Transform has been performed, starting from Fourier transform. Despite being fundamental from an historical point of view, since it allowed signals to be analysed by using their frequency domain for the first time, its properties are not suited for all signals: in fact, its sensibility of any kind of deformations results in a lot of difficulties when analysing determined signals, like ECGs.

For this reason a new mathematical transform has been defined: the Wavelet Scattering Transform, which bounds together the wavelet transform and a scattering propagator, creating a brand new transform, invariant to translation and scaling: these properties will be very useful, especially during the classification process.

Another problem rose up just before taking on the classification problem: is it possible to create a great amount of realistic human beats using mathematical techniques, and in particular time-series forecasting? That is the point of the whole second part of the work, since ECGs are really difficult to acquire and consume a very high amount of both time and resources: all of that without even considering all the privacy problems and the rarity of some syndromes, which means that available data is limited.

For this reason a new procedure has been evaluated: nine random beats are sampled and then, using different methods (ETS, ARIMA, LSTM, RNN) a new one is created: this process has been repeated until almost three hundred thousand new beats have been created. This allow for a greater variability of beats, which results to be very original and significantly different from the real ones and from the augmented ones, without losing their characteristics.

Finally it was possible to classify those new beats using WST.

In my future works I hope to improve even more the methods used to create beats, so that it can create even more realistic beats and hopefully do that in a more efficient way.

Appendices

Appendix A

Fundamentals of functional analysis

A.1 Basic notions on bounded operators

Definition A.1.1. Consider a set M. Then the function $d : M \times M \to \mathbb{R}$ is called **metric** if the following properties are satisfied:

1. $d(x, y) \geq 0$;
2. $d(x, y) = 0 \Leftrightarrow x = y$;
3. $d(x, y) = d(y, x)$ (symmetry property);
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

If d is a metric on M, then the pair (M, d) is called **metric space**.

Definition A.1.2. Let X be a vector space over a field $\mathbb{F}$. It is called **norm** on X a function $\|\cdot\| : X \to \mathbb{R}$ such that the following are valid $\forall\, x, y \in X$ and $\alpha \in \mathbb{F}$:

1. $\|x\| \geq 0$;
2. $\|x\| = 0 \Leftrightarrow x = 0$;
3. $\|\alpha x\| = |\alpha| \|x\|$;
4. $\|x + y\| \leq \|x\| + \|y\|$.

A generic vector space X equipped with a norm is called a **normed space**.

Definition A.1.3. Let X, Y two normed spaces defined on a numeric field $\mathbb{F}$; a transformation (or operator) $T : X \to Y$ is said to be **linear** if:

$$T(\alpha x + \beta y) = \alpha T(x) + \beta T(y), \forall\ x, y \in X, \alpha, \beta \in \mathbb{F}. \tag{A.1}$$

Proposition A.1.4. Let X, Y two normed spaces and $T : X \to Y$ a linear transformation. The following are equivalent:

1. T is uniformly continuous;

2. T is continuous;

3. T is continuous in 0;

4. $\exists\ k > 0\ s.t.\ \|T(x)\| \leq k$ if $x \in X, \|x\| \leq 1$;

5. $\exists\ k > 0\ s.t.\ \|T(x)\| \leq k\|x\|\ \forall\ x \in X$.

Proof. Implications 1. $\Rightarrow$ 2. and 2. $\Rightarrow$ 3. are taken for granted. Only the last three ones (3. $\Rightarrow$ 4., 4. $\Rightarrow$ 5. and 5. $\Rightarrow$ 1.) will be shown.

3. $\Rightarrow$ 4. If T is continuous in 0, it has to be so in a neighborhood of 0, which means that if $\epsilon = 1 \Rightarrow \exists\ \delta > 0$ s.t. $\|T(x)\| < 1$ if $\|x\| < \delta, x \in X$. Taken $\omega \in X$ with $\|\omega\| \leq 1$, it yields that $\|\frac{\delta\omega}{2}\| = \frac{\delta}{2}\|\omega\| \leq \frac{\delta}{2} \leq \delta$, and so:

$$\left\|T\left(\frac{\delta\omega}{2}\right)\right\| < 1 \Rightarrow T\left(\frac{\delta\omega}{2}\right) = \frac{\delta}{2}T(\omega)$$

because T is linear. So: $\frac{\delta}{2}\|T(\omega)\| < 1 \Rightarrow \|T(\omega)\| < \frac{2}{\delta}$.

4. $\Rightarrow$ 5. If k is defined in such a way that $\|T(x)\| \leq k$, with $x \in X, \|x\| = 1$. Since $T(0) = 0 \Rightarrow T(0) \leq k\|0\|$, so let's consider $y \in X, y \neq 0$; then if we normalize by the norm $\left\|\frac{y}{\|y\|}\right\| = 1 \Rightarrow \left\|T\left(\frac{y}{\|y\|}\right)\right\| \leq k$. Then again, because T is a linear transformation:

$$\frac{1}{\|y\|}\|T(y)\| = \left\|\frac{T(y)}{\|y\|}\right\| = \left\|T\left(\frac{y}{\|y\|}\right)\right\| \leq k,$$

so $\|T(y)\| \leq k\|y\|$, which proofs the point.

5. $\Rightarrow$ 1. Since T is a linear transformation:

$$\|T(x) - T(y)\| = \|T(x - y)\| \leq k\|x - y\|, \forall\ x, y, \in X.$$

Putting $\epsilon > 0, \delta = \frac{\epsilon}{k}$, then if $\|x - y\| < \delta$, we have:

$$\|T(x) - T(y)\| \leq k\|x - y\| < k\left(\frac{\epsilon}{k}\right) = \epsilon,$$

which means that T is uniformly continuous. □

That means that linear bounded operators are continuous, and viceversa.

Definition A.1.5. Consider a measure space (X, μ). Once the following norm is defined:

$$\|f\|_p = \left(\int_X |f|^p d\mu \right)^{\frac{1}{p}}, \ 1 \leq p < +\infty, \tag{A.2}$$

it is called **space** L^p the set which includes all functions $f : X \to \mathbb{R}$ such that the following property is verified:

$$\|f\|_p < +\infty. \tag{A.3}$$

Cases for $p = 1{,}2$ are of particular importance, as the first part of this work pointed out.

Definition A.1.6. Consider two functions f, g defined in $\mathbb{R}$. Then the following function is defined as **convolution** of f and g:

$$(f \star g)(t) := \int_{-\infty}^{+\infty} f(\tau) g(t - \tau) d\tau = \int_{-\infty}^{+\infty} f(t - \tau) g(\tau) d\tau \tag{A.4}$$

Definition A.1.7. Consider a function $f : \mathbb{R}^n \to \mathbb{R}^m$, and denote each component of $f(x) = (f_1, \dots \ f_m)$ Then the following matrix with m rows and n columns is defined as the **Jacobian matrix** of f in the generic point $x \in \text{dom}(f) \subset \mathbb{R}^n$:

$$Jf(x) = (a_{ij})_{1 \leq i \leq m, \ 1 \leq j \leq n} \quad a_{ij} = \left(\frac{\partial f_i}{\partial x_j}(x) \right) \tag{A.5}$$

For the same function it is possible to define the **Hessian function** as well:

$$Hf(x) = (h_{ij})_{1 \leq i,j \leq n}, \quad h_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}(x) \tag{A.6}$$

Definition A.1.8. The following generalized function (also called distribution) is called **Dirac's distribution**:

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.7}$$

Observation A.1.9. It is possible to note that, following its definition:

$$\int_{-\infty}^{+\infty} \delta(x) dx = 1$$

It is clearly useful in cases where you just want to consider only a point of a certain function; in fact for every continuous function the following holds:

$$\int_{-\infty}^{+\infty} \delta(x - x_0) \phi(x) dx = \phi(x_0).$$

Definition A.1.10. Given $f : X \to Y, X, Y$ normed spaces, f is said to be a **Lipschitz operator** if:

$$\|f(x) - f(y)\|_Y \leq k\|x - y\|_X, \forall\ x \in X, y \in Y. \tag{A.8}$$

A.8 is called **Lipschitz property**. Basically, f is a Lipschitz operator if its variations in Y is always bounded by a multiple of the variations happened in X; k is called **Lipschitz constant**. Besides, a Lipschitz operator is said:

- **contractive** if $k < 1$;
- **nonexpansive** if $k \leq 1$;
- **expansive** if $k > 1$.

A.2 Definition of Hilbert spaces

Together with the properties for linear operators, it is necessary to briefly review some concepts of functional analysis, starting from the definition of a Hilbert space and going through everything that supports this definition.

Definition A.2.1. The sequence $\{x_n\}_{n\in\mathbb{N}}$ is called a *Cauchy sequence* if $\forall\ \varepsilon > 0\ \exists\ N(\varepsilon) > 0$ such that $d(x_n, x_m) < \varepsilon\ \forall\ n, m > N(\varepsilon)$.

Noting that d is a metric, the following are recalled:

- in a metric space (that is, a vector space equipped with a metric) any convergent sequence is a Cauchy sequence;
- a metric space is said to be complete if all Cauchy sequences belonging to it are convergent.

At this point, it is possible to give the definition of a Hilbert space.

Definition A.2.2. A real or complex vector space H on which an inner product $\langle\cdot,\cdot\rangle$ is defined is called a Hilbert space $\mathbb{H} = (H, \langle\cdot,\cdot\rangle)$ if the metric space (H, d) turns out to be complete, where d is the metric induced by the inner product on H.

Note that for $1 \leq p < +\infty$, L^p are Banach spaces; only L^2 is an Hilbert space.

Definition A.2.3. Consider an Hilbert space $\mathbb{H}$ and its scalar product

$$\langle\cdot,\cdot\rangle : \mathbb{H} \times \mathbb{H} \to \mathbb{R}, \tag{A.9}$$

and let it respects all properties from A.1.2; then if the following function:

$$d(x, y) = \|x - y\| \tag{A.10}$$

is a metric, then (A.10) is called norm induct by scalar product $\langle\cdot,\cdot\rangle$, and it is denoted by $\|\cdot\|_{\mathbb{H}}$

Appendix B

Topology and algebra revise

B.1 Basic notions

Notions of topological space and open and closed set will be taken for granted.

Definition B.1.1. Given B subset of a topological space X, it is called **closing** of set B the following:

$$\overline{B} = \bigcap \{C | B \subset C \subset X\} \tag{B.1}$$

Definition B.1.2. A subset A of a topological space X is called **dense** in said topological space if $\overline{A} = X$.

Definition B.1.3. Consider a subset S of a vector space V over a numeric field $\mathbb{K}$; then it is possible to define the following:

$$span(S) = \{\lambda_1 v_1 + \dots \lambda_n v_n \mid n \in \mathbb{N}, v_1, \dots, v_n \in S, \lambda_1, \dots, \lambda_n \in \mathbb{K}\}.$$

This sum is not necessarily finite.

Definition B.1.4. A sequence $\{g_n\}_{n\in\mathbb{N}}$ of elements in an Hilbert space $\mathbb{H}$ is said to be a **basis** of it if the following condition is satisfied:

$$\overline{span(g_n)} = \mathbb{H}, \tag{B.2}$$

which means that the linear space spanned by those elements is dense in the Hilbert space considered.

If $\langle g_m, g_n \rangle = 0 \ \forall \ m \neq n$, which means that all the elements of the basis are linearly independent from each other, then the basis is called **orthogonal**; besides, if $\|g_n\| = 1 \ \forall \ n \in \mathbb{R}$ is valid together with all other properties, then $\{g_n\}_{n\in\mathbb{N}}$ is said to be an **orthonormal basis**.

Corollary B.1.5. Let $\mathcal{B} = \{g_n\}_{n\in\mathbb{N}}$ an orthonormal basis of an Hilbert space $\mathbb{H}$; then for every $u \in \mathbb{H}$ we have:

$$u = \sum_{k=1}^{+\infty} \langle u, g_k \rangle g_k \tag{B.3}$$

and

$$|u|^2 = \sum_{k=1}^{+\infty} |\langle u, e_k \rangle|^2. \tag{B.4}$$

Thanks to B.1.3 it is clear that every element of an Hilbert space $\mathbb{H}$ can be written as a sum using each element of $\mathcal{B}$; it can be said that elements of $\mathcal{B}$ generates $\mathbb{H}$.

Appendix C

Useful concepts for proving theorem 2.3.2

C.1 Definitions and lemmas

Definition C.1.1. Consider a countable family $(M_n)_{n\in\mathbb{N}}$ of closed subspaces of an Hilbert space X; then the **internal orthogonal direct sum** of said M_n is defined such that:

$$\bigoplus_{n\in\mathbb{N}} M_n = \overline{\bigcup_{n\in\mathbb{N}} M_n} \tag{C.1}$$

In other words, if $\{x_n\}$ is an orthonormal base for an Hilbert space X, then $X = \bigoplus_{n\in\mathbb{N}} x_n$.

Lemma C.1.2. Considering an MRA $V_j, j \in \mathbb{Z}$, an orthonormal base for it is given by $\{2^{j/2}\phi(2^j t - n) \mid n \in \mathbb{Z}\}$

Proof. Changing variable $u = 2t$, we have an isomorphism I such that:

$$I(V_0) = V_1, \qquad I(f(t)) = \sqrt{2}f(2t). \tag{C.2}$$

So $\{\sqrt{2}\phi(2t-n) \mid n \in \mathbb{Z}\}$ is an orthonormal base for V_1 and $\{2^{j/2}\phi(2^j t-n) \mid n \in \mathbb{Z}\}$ is an orthonormal base for V_j □

C.2 Theorems with relative proofs

Theorem C.2.1. Consider a space X with an inner product $\langle\cdot,\cdot\rangle$ and a complete subspace $M \subset X$; then the following are valid:

a. $X = M \oplus M^{\perp}$;

b. $M = (M^\perp)^\perp$

Proof. Point a. Clearly $M \cap M^\perp = \{0\}$; consider $x \in X$ and define $m_0 \in M$ as its projection in M. Since $x - m_0 \in M^\perp$ and $x = m_0 + (x - m_0)$, then M and $M^\perp$ are complementary.

Besides:

$$\|x - x_i\|^2 = \|m - m_i\|^2 + \|n - n_i\|^2,$$

which means that, if we define $x_i = m_i + n_i \in M \oplus M^\perp$, we obtain that

$$x_i \to x = m + n \Rightarrow m_i \to m \wedge n_i \to n,$$

and because of arbitrariness of x, it means that

$$X = M \oplus M^\perp.$$

Point b. Since $M \subset M^{\perp\perp}$ (this will be taken for granted; proof in [4]), thanks to point a. the following can be considered: $x = m + m' \in M^{\perp\perp}$.

Now, since $x, m \in M^{\perp\perp} \Rightarrow m' = x - m \in M^{\perp\perp}$ as well. Finally, since $m' \in M^\perp \cap M^{\perp\perp}$, it means that $m' = 0$, so $M^{\perp\perp} \subset M$ and $M = M^{\perp\perp}$. □

Theorem C.2.2. For any subspace M of an Hilbert space X, the following hold true:

a. $M^{\perp\perp} = \overline{M}$;

b. $M^\perp = \{0\} \Leftrightarrow \overline{M} = X$.

Proof. To prove point a., remember that $M \subset M^{\perp\perp}$, and M is closed, then necessarily $\overline{M} \subset M^{\perp\perp}$. Since $\overline{M}$ is complete, then Hilbert space $M^{\perp\perp}$ can be decomposed as:

$$M^{\perp\perp} = \overline{M} \oplus \overline{M}^\perp.$$

Since $M \subset \overline{M}$, then $\overline{M}^\perp \subset M^\perp$. Finally, considering $x \in (\overline{M})^\perp$, then $x \in M^\perp \cap M^{\perp\perp} = \{0\}$ and $M^{\perp\perp} = \overline{M} \oplus (\overline{M})^\perp = \overline{M}$

Point b., ($\Leftarrow$): let $x \in M^\perp$; then for any $r > 0$, since M is dense in X, there exists $y \in M$ such that $\|x - y\| < r$. Then:

$$r^2 > \|x - y\|^2 = \|x\|^2 + \|y\|^2 \geq \|x\|^2.$$

Since r is arbitrary, then $x = 0$.

Point b., ($\Rightarrow$): simply, $M^\perp = \{0\} \Rightarrow \overline{M} = M^{\perp\perp} = 0^\perp = X$. □

Ringraziamenti

Contro ogni convenzione e superstizione, ho iniziato a scrivere questi ringraziamenti a novembre 2023, ovvero nel periodo, da un punto di vista accademico, più complicato. Dopo essere stato bocciato in due esami a settembre e aver preso un voto basso in un progetto su cui avevo speso i precedenti sei mesi della mia vita, mi sono ritrovato senza alcuna borsa di studio, con ancora sette esami da dare, una media bassissima e avevo accumulato già un anno di ritardo rispetto alla naturale conclusione degli studi.

Quasi due anni dopo la media non si è alzata, il ritardo si è aggravato eppure mi riempie di gioia leggere queste righe ora che il percorso si è formalmente concluso: e a questo giro vi beccate i ringraziamenti di una vita.

Ringrazio il mio relatore, il professor Lamberto Rondoni, e il dottor Davide Carbone per il sostegno e l'aiuto indispensabile offertomi in questi mesi al fine di concludere il mio lavoro.

In particolare, ci tengo a ringraziare il professore per la sua incredibile passione per le materie che insegna, che mi ha permesso di riaccendere la mia per questo mondo strano ed infinito che è la matematica.

Inoltre ringrazio infinitamente Davide per avermi letteralmente accompagnato, nel corso dell'ultimo anno, attraverso questa ondivaga esperienza che è stata scrivere la tesi: il tuo sostegno e le tue indicazioni sono state fondamentali al fine di concludere questo lavoro, e te ne sono grato.

Ringrazio Alessandro, in arte Sandrino, non solo perché condividere con te le passioni per la musica, lo sport e per tutto ciò che riguarda la matematica mi apre sempre nuovi orizzonti, ma soprattutto perché sei una persona onesta e generosa. Le parole difficilmente rendono giustizia di quanto ogni istante passato con te mi abbia fatto sentire bene: ogni chiamata ricevuta in tutti questi anni è stata la dimostrazione della purezza della nostra amicizia, e spero di poter continuare a condividere con te le mie passioni come ho potuto fare fino a questo momento.

Ringrazio Claudia e Veronica, coinquiline e amiche, perché ciò che abbiamo condiviso rimarrà per sempre una parte fondamentale della mia vita.

Claudia, il modo naturale e spontaneo che hai di migliorare l'umore di chi sta attorno a te semplicemente con la tua presenza e con la tua capacità di prendertene

cura con tranquillità, sincerità e serenità è stato per me fondamentale nei tanti momenti bui dei due anni in cui abbiamo condiviso l'appartamento e tante altre sfighe.

Veronica, la tua gentilezza, il tuo modo naturale di far sentire importanti e a loro agio le persone sono qualità speciali e rare, e dalle quali prendo ispirazione ogni giorno della mia vita. Grazie per la tua generosità d'animo, per tutte le volte che mi hai fatto ridere e soprattutto per tutte le volte in cui hai fatto i piatti senza chiedermi nulla in cambio.

Qualunque cosa accadrà l'appartamento di Corso Unione Sovietica 91 rimarrà per sempre il primo luogo a cui ripenserò ricordando questi anni, e voi con esso. (Che poi sia una cosa positiva o meno, vedremo...)

Ringrazio Nicola, che per me è stato come un secondo fratello dal primo momento in cui l'ho incontrato ad ottobre del 2021 a Villa Claretta. Posso solo provare a trascrivere l'importanza che hanno avuto tutte le gare viste insieme, gli spritz bevuti alle panche, i consigli scambiati nei momenti difficili e le risate condivise nei momenti belli. Ti ringrazio soprattutto per la spontaneità con cui vivi, che mi ha insegnato quanto sia bello essere sè stessi, senza pensare al giudizio altrui, e di come questa cosa possa anche non andare bene a tutte le persone che si incontreranno, senza che ciò abbia importanza: tutto ciò che conta alla fine è potersi guardare allo specchio a fine giornata, e poter sorridere per essere riusciti ad esprimerci nella maniera più naturale possibile.

Ringrazio i miei fantastici amici di Carbonia: Alessia, Camilla, Davide A., Davide E., Fabio, Giulia, Lorenzo F., Lorenzo P., Lorenzo S., Luca M., Luca O., Silvia. Potrei scrivere un libro su ciascuno di voi ed elencare le cazzate che abbiamo fatto, le risate, le giornate al mare e i campeggi in cui ci siamo avvicinati e attraverso i quali abbiamo solidificato rapporti già molto profondi e coinvolgenti. Vi ringrazio soprattutto per avermi dato una prova tangibile di una cosa di cui sono sempre stato profondamente convinto: che l'amicizia non conosca confini e distanze, e che anzi sia sempre in grado di travalicarle e andare oltre. Negli ultimi tre anni tantissimi rapporti su cui avevo speso tempo, denaro, lacrime e sanità mentale si sono definitivamente interrotti per motivi diversi, mentre il nostro ha resistito al tempo, alle insidie, alle incomprensioni, e allo spazio che ci ha separati. Mi auguro possa essere così ancora per molto, moltissimo tempo.

Ringrazio Niccolò, il mio fratellino. Anche se talvolta non condivido alcuni tuoi atteggiamenti, io penso che tu abbia un potenziale immenso: sei una persona determinata, che quando si appassiona a qualcosa ha la capacità di comprenderla e applicarla nella maniera più precisa possibile. Proprio per questo volevo ringraziarti: perché negli ultimi anni abbiamo scoperto di avere tante passioni in comune, e poterle condividere con te è stata un'emozione nuova, abituato come sono a considerare le mie passioni come dei guilty pleasures e a nasconderle, come se fossero cose di cui vergognarsi. Invece le passioni sono ciò che più di ogni altra

cosa ci fa capire quanto la vita meriti di essere vissuta. Ti auguro di spiccare il volo come meriti, e come già stai dimostrando di poter fare: che questo sia solo l'inizio anche per te.

Ultimi, ma primi per importanza, ringrazio Patrizia e Pierpaolo, ovvero la mia mamma e il mio papà. Non solo per il vostro sostegno (economico, morale, spirituale), incrollabile come solo la fede sa essere anche nei momenti più bui e miseri, ma soprattutto vi ringrazio perché, nel corso degli anni, mi avete insegnato cosa significhi davvero amare incondizionatamente. In questo momento della mia vita penso che farò scelte diverse da quelle che avete fatto voi, e probabilmente lo faccio solo per egoismo e per voglia di una effimeria ed illusoria libertà. Proprio per questo mi rendo conto di quanto preziosi siate stati voi che, nel creare la nostra famiglia, avete accettato tutto ciò che questa scelta comportava, senza mai farmi mancare nulla e senza mai privarmi dell'affetto e del sostegno necessari per proseguire e imparare a camminare con le mie gambe, in un mondo sempre più complicato e complesso.

Anche se oggi si conclude una lunga fase della mia vita, e se ne apre una con ancora maggiori responsabilità, so che potrò sempre contare sempre sulla mia mamma e sul mio papà.

Questo traguardo è per voi, per tutte le persone meravigliose che ho menzionato prima e per tutte le altre che non ho menzionato, ma che essendo presenti qui oggi hanno deciso di condividere con me questo sogno che si realizza: grazie, perché questi ricordi e questi momenti saranno per sempre nostri, che in fondo è ciò che conta di più.

A voi tutti e tutte voglio dire che vi voglio bene, che siete la parte migliore della mia vita, e vi amo.

Bibliography

[1] A. Karpathy. The unreasonable effectiveness of recurrent neural networks, 2015. https://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[2] J. Anden and S. Mallat. Multiscale scattering for audio classification. *2011 International Society for Music Information Retrieval*, 2011.

[3] M. Artin. *Algebra*. Pretience Hall, 1991.

[4] G. Bachman, L. Narici, and E. Beckenstein. *Fourier and Wavelet analysis*. Springer, 2002.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[6] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis - Forecasting and Control*. Prentice-Hall International, 1994.

[7] H. Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer, 2011.

[8] D. Brillinger. *Time Series - Data Analysis and Theory*. SIAM, 2001.

[9] J. Bruna. *Scattering Representations for Recognition*. PhD thesis, Ecole Polytechnique, 2012.

[10] J. Bruna and S. Mallat. Classification with Scattering Operators, 2013. https://arxiv.org/abs/1011.3023.

[11] J. Bruna and S. Mallat. Invariant Scattering Convolution Networks. *IEEE Transaction on pattern analysis and machine intelligence*, 2013.

[12] C. Olah. Understanding lstm networks, 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[13] P. Cannarsa and T. D'Aprile. *Introduzione alla teoria della misura e all'analisi funzionale*. Springer, 2008.

[14] D. Carbone and A. Licciardi. Wavelet Scattering Transform for Bioacustics: Application to Watkins Marine Mammal Sound Database. *arXiv e-prints*, pages arXiv–2402, 2024.

[15] C. Chatfield. *The analysis of time series. An introduction*. Springer, 1984.

[16] S. Cobzas, R. Miculescu, and A. Nicolae. *Lipschitz Functions*. Springer, 2019.

[17] P. Cowpertwait and A. Metcalfe. *Introductory Time Series with R*. Springer, 2009.

[18] D. Britz. Recurrent neural network tutorial, part 4 – implementing a gru and lstm rnn with python and theano, 2015. https://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-4/.

[19] D. Britz. Recurrent neural networks tutorial, part 1 – introduction to rnns, 2015. https://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-1/.

[20] D. Britz. Recurrent neural networks tutorial, part 3 – backpropagation through time and vanishing gradients, 2015. https://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-3/.

[21] M. Kumar Das and S. Ari. ECG Beats Classification using mixture of features. *International Scholarly Research Notices*, 2014.

[22] S. Dattani, F. Spooner, H. Ritchie, and M. Roser. Causes of death. *Our World in Data*, 2023. https://ourworldindata.org/causes-of-death.

[23] M. Deza and E. Deza. *Encyclopedia of Distances*. Springer, 2009.

[24] H. Ni et al. Time Series Modeling for Heart Rate Prediction: from ARIMA to Transformers. *Proceedings of the 2024 6th International Conference on Electronic Engineering and Informatics*, 2024.

[25] K. Benidis et al. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 2022.

[26] Liu Z. et al. Wavelet Scattering Transform for ECG Beat Classification. *Computational and Mathematical Methods in Medicine*, 2020.

[27] M. Löning et al. sktime version 0.36.0. https://github.com/sktime/sktime.

[28] M. Löning et al. 33rd conference on neural information processing systems. In *sktime: A Unified Interface for Machine Learning with Time Series*, Vancouver, Canada, 2019.

[29] P. Virtanen et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[30] R. Acharya et al. A Deep Convolutional Neural Network Model to Classify Heartbeats. *Computers in Biology and Medicine*, 2017.

[31] G. Folland. *Real analysis. Modern techniques and their applications*. Wiley-Interscience, 1999.

[32] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[33] A. Grossmann and J. Morlet. Decomposition of Hardy functions into square integrable wavelets of costant shape. *Society for Industrial and Applied Mathematics*, 1984.

[34] A. Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 1910.

[35] S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. PhD thesis, Institut fur Informatik, Technische Universitat, Munchen, 1991.

[36] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[37] R. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2021. OTexts.com/fpp3.

[38] D. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[39] N. Lanchier. *Stochastic Modeling*. Springer, 2010.

[40] A. Licciardi. Wavelet scattering transform. mathematical analysis and applications to virgo gravitational waves data. Master's thesis, Politecnico di Torino, 2023.

[41] A. Licciardi, D. Carbone, and L. Rondoni. Wavelet Scattering Operators for Multiscale Processes: The Case Study of Marine Mammal Vocalizations. In *International Conference on Nonlinear Dynamics and Applications*, pages 173–191. Springer, 2024.

[42] A. Licciardi, D. Carbone, L. Rondoni, and A. Nagar. Wavelet Scattering Transform for gravitational wave analysis: An application to glitch characterization. *Physical Review D*, 111(8):084044, 2025.

[43] S. Mallat. Group Invariant Scattering. *Communications on Pure and Applied Mathematics*, 2011.

[44] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 2011.

[45] S. Mallat. Understanding Deep Convolutional Networks. *A Philosophical Transactions*, 2016.

[46] M. Manetti. *Topologia*. Springer, 2014.

[47] H. Marzog and H. Abd. ECG-signal Classification using efficient Machine Learning approach. *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications*, 2022.

[48] T. Mills. *Applied Time Series Analysis*. Academic Press, 2019.

[49] D. Montgomery, C. Jennings, and M. Kulahci. *Introduction to Time Series Analysis and Forecasting*. Wiley, 2008.

[50] G. Moody and R. Mark. The Impact of the MIT-BIH Arrhythmia Dataset. *IEEE Engineering in Medicine and Biology*, 2001.

[51] J. Pan and W. Tompkins. A real-time QRS detection algorithm. *IEEE Transaction on Biomedical Engineering*, 1985.

[52] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks, 2013.

[53] M. Reed and B. Simon. *Functional Analysis*. Academic Press, 1980.

[54] G. Van Rossum and F. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[55] B. Rynne and M. Youngson. *Linear Functional Analysis*. Springer, 2008.

[56] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[57] Staff HPC@Polito. Guida introduttiva e regole per l'uso dei cluster hpc, 2022.

[58] American National Standard. Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms. *Association for the Advancement of Medical Instrumentation*, 2001.

[59] The MathWorks Inc. MATLAB version: 9.5 (R2018b), 2018. https://www.mathworks.com/matlabcentral/answers/593449-r2018b-matlab-9-5.

[60] The MathWorks Inc. MATLAB version: 9.5 (R2018b), 2018. https://www.mathworks.com/matlabcentral/answers/593449-r2018b-matlab-9-5.

[61] The MathWorks Inc. Wavelet Toolbox version: 5.1 (R2018b), 2018. https://www.mathworks.com/products/wavelet.html.

[62] C. Torrence and G. Compo. A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, 1997.

[63] M. Vetterli and J. Kovacevic. *Wavelets and subband coding*. Prentice Hall PTR, 1995.

[64] W. Wei. *Time Series Analysis*. Pearson Education, 2006.