PGCOMP - Programa de Pós-Graduação em Ciência da Computação
Universidade Federal da Bahia (UFBA)
Av. Milton Santos, s/n - Ondina
Salvador, BA, Brasil, 40170-110

https://pgcomp.ufba.br
pgcomp@ufba.br

The adoption of Deep Neural Network (DNN) methods to solve problems in real-world scenarios has been increasing as the data volume grows. Although such methods present impressive results in supervised learning, it is known that the occurrence of noises modifying the original data behavior can affect the model accuracies and, consequently, the generalization process, which is highly relevant in learning tasks. Several approaches have been proposed to reduce the impact of noise on the final model, varying since the application of preprocessing steps to the design of robust DNN layers. However, we have noticed that such approaches were not systematically assessed to understand how the noise influences have been propagated throughout the DNN architectures. This gap motivated us to design this work, which was focused on modeling noisy data with temporal dependencies, typically referred to as time series or signals. In summary, our main claim was to create a network capable of acting as a noise filter and being easily connected to existing networks. To reach this goal, we have defined a methodology, which was organized into four phases: (i) execution of a study about the application of DNNs to model signals collected from a real-world problem; (ii) investigation of different preprocessing tools to transform such signals and reduce noise influences; (iii) analysis about the impact of increasing/reducing the noise on the final model; and (iv) creation of a new DNN that can be embedded into DNN architectures and act as noise filtering layer to keep the overall performances. The first and second phases were achieved in collaboration with researchers from the Universidad de La Frontera, which provided a set of signals directly collected from the Llaima volcano in Chile. The modeling performed on such signals allowed the creation of a new architecture called SeismicNet. By knowing the behavior or such signals, we could create a controlled scenario with different additive noise levels and outputs produced by our original models, thus meeting the third phase of the described methodology. Next, we performed two new studies to understand the impact of noises in our scenario. Firstly, we used statistical tests to confirm the error variation when noise is added to the expected signals. Then, we used XAI (eXplainable Artificial Intelligence) to visually comprehend the noise propagation into the DNN layers. Finally, we were able to finish up the last phase and accomplish our primary goal: the design of a new neural network architecture with embedding noise filtering to suppress the preprocessing phase. Interpreting the obtained results, we understand that this novel approach learned the noisy features better and was capable of delivering stable results apart from the noise level on the signal.

Keywords: Deep Neural Networks, Explainable AI, Time Series Analysis.

DSC | 030 | 2022

On deep learning features for noisy time series classification

João Paulo Pereira
de Sá Canário

UFBA

On deep learning features for noisy time series classification

João Paulo Pereira de Sá Canário

Tese de Doutorado

Universidade Federal da Bahia

Programa de Pós-Graduação em
Ciência da Computação

Julho | 2022

Federal University of Bahia
Institute of Computing

Postgraduate Program in Computer Science

ON DEEP LEARNING FEATURES FOR NOISY TIME SERIES CLASSIFICATION

João Paulo Pereira de Sá Canário

PHD THESIS

Salvador
04 of July of 2022

JOÃO PAULO PEREIRA DE SÁ CANÁRIO

ON DEEP LEARNING FEATURES FOR NOISY TIME SERIES CLASSIFICATION

This doctoral thesis was presented to the Postgraduate Program in Computer Science at Federal University of Bahia as partial fulfillment of the degree requirements of a Ph.D. in Computer Science.

Advisor: Prof. Dr. Ricardo Araújo Rios

Salvador
04 of July of 2022

Ficha catalográfica elaborada pela Biblioteca Universitária de
Ciências e Tecnologias Prof. Omar Catunda, SIBI - UFBA.

C213 Canário, João Paulo Pereira de Sá

On deep learning features for noisy time series classification / João Paulo Pereira de Sá Canário. – Salvador, 2022.

85p.: il.

Orientador: Prof. Dr. Ricardo Araújo Rios

Tese (Doutorado) – Universidade Federal da Bahia. Instituto de Computação, 2022.

1. Computação. 2. Redes Neurais. 3. Análises de Series Temporais. I. Rios, Ricardo Araújo. II. Universidade Federal da Bahia. III. Título.

CDU 519.246

João Paulo Pereira de Sá Canário

On deep learning features for noisy time series classification

Esta tese foi julgada adequada à obtenção do título de Doutor em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da UFBA.

Salvador, 04 de julho de 2022

Ricardo Araújo Rios
Prof. Dr. Ricardo Araújo Rios
(Orientador - PGCOMP/UFBA)

Eduardo F. Simas Filho
Prof. Dr. Eduardo Furtado de Simas Filho
(UFBA)

Prof. Dr. Marcelo Keese Albertini
(UFU)

Prof. Dr. Rubisley de Paula Lemes
(UFBA)

Profa. Dra. Millaray Curilem Saldías
(Universidad de La Frontera)

To God, my family, my dog, and my friends.

ACKNOWLEDGEMENTS

During this journey, many people contributed to the viability of my work. Thus, I would like to thank everyone who helped and was by my side in this process. Everything you write here will be too little to extol the importance you had.

Among these are fellow researchers and professors from my research group. In particular, I would like to express my thanks to my advisor, Ricardo Rios, to whom I owe all my support and a friendly relationship.

I also would like to express my gratitude to my family, who continuously inspire and support me in all my decisions. My wife, Carolina, and my unborn daughter, Luiza. My parents, João and Terezinha. My sister, Camila, my brother-in-law, Hermes, and my nephews, Thiago and Arthur. Thank you for your trust, wisdom, and perseverance.

To my dog, Moreno, who, whenever I needed it, made me happy (even without saying a word) with all his love.

To my friends, present during this doctorate and offering encouragement and appreciation.

Thank you all for the support on this journey!

"Don't adventures ever have an end? I suppose not. Someone else always has to carry on the story."

—BILBO BAGGINS, THE FELLOWSHIP OF THE RING

ABSTRACT

The adoption of Deep Neural Network (DNN) methods to solve problems in real-world scenarios has been increasing as the data volume grows. Although such methods present impressive results in supervised learning, it is known that the occurrence of noises modifying the original data behavior can affect the model accuracies and, consequently, the generalization process, which is highly relevant in learning tasks. Several approaches have been proposed to reduce the impact of noise on the final model, varying since the application of preprocessing steps to the design of robust DNN layers. However, we have noticed that such approaches were not systematically assessed to understand how the noise influences have been propagated throughout the DNN architectures. This gap motivated us to design this work, which was focused on modeling noisy data with temporal dependencies, typically referred to as time series or signals. In summary, our main claim was to create a network capable of acting as a noise filter and being easily connected to existing networks. To reach this goal, we have defined a methodology, which was organized into four phases: (i) execution of a study about the application of DNNs to model signals collected from a real-world problem; (ii) investigation of different preprocessing tools to transform such signals and reduce noise influences; (iii) analysis about the impact of increasing/reducing the noise on the final model; and (iv) creation of a new DNN that can be embedded into DNN architectures and act as noise filtering layer to keep the overall performances. The first and second phases were achieved in collaboration with researchers from the Universidad de La Frontera, which provided a set of signals directly collected from the Llaima volcano in Chile. The modeling performed on such signals allowed the creation of a new architecture called SeismicNet. By knowing the behavior or such signals, we could create a controlled scenario with different additive noise levels and outputs produced by our original models, thus meeting the third phase of the described methodology. Next, we performed two new studies to understand the impact of noises in our scenario. Firstly, we used statistical tests to confirm the error variation when noise is added to the expected signals. Then, we used XAI (eXplainable Artificial Intelligence) to visually comprehend the noise propagation into the DNN layers. Finally, we were able to finish up the last phase and accomplish our primary goal: the design of new neural network architecture with embedding noise filtering to suppress the preprocessing phase. Interpreting the obtained results, we understand that this novel approach learned the noisy features better and was capable of delivering stable results apart from the noise level on the signal.

Keywords: Deep Neural Networks, Explainable AI, Time Series Analysis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AI	Artificial Intelligence	14
ANN	Artificial Neural Network	79
CEC	Constant Error Carousel	17
CWT	Continuous Wavelet Transform	32
CNN	Convolutional Neural Network	69
DBN	Deep Belief Network	22
DNN	Deep Neural Network	69
DWT	Discrete Wavelet Transform	70
EMD	Empirical Mode Decomposition	22
FP	False Positive	30
FN	False Negative	30
FT	Fourier Transform	7
FS	Fourier series	7
GB	Guided Backprop	20
IG	Integrated Gradients	20
IMF	Intrinsic Mode Function	22
LRP	Layer-wise Relevance Propagation	69
LSTM	Long Short-Term Memory	70
MAE	Mean Absolute Error	59
ML	Machine Learning	69
MLP	Multilayer Perceptron	39
OVDAS	Observatorio Vulcanológico de los Andes Sur	27
ReLU	Rectified Linear Unit	16
RBM	Restricted Boltzmann Machines	22
RNN	Recurrent Neural Network	17
SNR	Signal-to-Noise Ratio	79
STFT	Short-Time Fourier Transform	50
TS	Time Serie	5

TN	True Negative	30
TP	True Positive	30
XAI	eXplainable Artificial Intelligence	70
UFRO	Universidad de la Frontera	23

Chapter 1

INTRODUCTION

1.1 GENERAL CONTEXT AND MAIN PROBLEM

Time is a natural and important variable present in several applications, from the most conventional/usual activities (e.g., speaking, vision, and walking) to the most present-day ones (e.g., robotic tasks, disaster prevention, and fake news spread). In general, such applications deal with time by modeling events to support complex decisions such as, for example, a better understanding of current states and the identification of possible adverse effects that might affect a system in the future (FAMA, 1965). The increasing number of applications, in which the data is characterized by temporal relationships, motivated the development of important research areas as Time Series Analysis and Signal Processing. For the sake of clarity, in the context of our work, we will consider both areas in an interchangeable manner.

Time Serie (TS) (or Signal) can be defined as a sequence of data (observations) collected after monitoring a system during specific time intervals (BOX *et al.*, 2015)[1]. TS can also be defined as a way of organizing, over time, quantitative information of a given situation and, due to their natural temporal order, they are present in multiple tasks that require some human cognitive process. Several real-world systems, through the recording observations of numerous phenomena, yield outputs as TS (e.g., climate, stock market, and social media data), making it possible to model and, consequently, understand their behavior (GEOGHEGAN, 2006).

Thereby, with the increase in temporal data availability, researchers have been proposing various methods in the past years to better model TS (LÜTKEPOHL, 2005; ESLING; AGON, 2012; BAGNALL *et al.*, 2017). However, although the development of methods to analyze TS has remarkably increased (KEOGH; KASETTY, 2003; DIETTERICH, 2002), such a task has still been considered one of the most challenging problems in data

[1] At the time of writing this work, a particular type of time series has been widely studied, referred to as Data Stream, whose observations are characterized for being collected in an open-ended manner, i.e., a system is continuously monitored producing a data flow (GAMA *et al.*, 2014). However, such a time-series definition is not considered in this work.

mining as discussed by Yang and Wu (2006). Such challenge is especially explainable by the high dimensionality and the presence of noise, which jeopardize the estimation of models once the data dynamics become very complex or even unknown (SUTSKEVER; HINTON; TAYLOR, 2009). Therefore, the adoption of traditional methods, which only contain a reduced number of linear/nonlinear operations, fails to precisely model such complex data (SUTSKEVER; HINTON; TAYLOR, 2009).

In order to overcome this issue, several problems have been successfully modeled by using Deep Neural Networks (DNNs) that are capable of capturing data information, even presenting complex structures. DNN has been increasingly adopted to achieve state-of-the-art results across multiple sets of reference data, solving challenging Artificial Intelligence (AI) tasks. To illustrate this statement, consider PaLM, which is a Transformer language model densely activated, trained with 540 billion parameters in a computational infrastructure with 6144 TPU v4 chips (CHOWDHERY *et al.*, 2022). According to the Google Research team, PaLM is considered the state-of-the-art in its NLP (Natural Language Process) tasks providing the best results on 28 of 29 datasets usually considered in this area. Another impressive result is presented by Dall· 2 (RAMESH *et al.*, 2022), which is a new AI system capable of creating realistic images and art from a description in natural language. Dall· 2 (RAMESH *et al.*, 2022), designed by the OpenAI foundation[2], uses a 3.5 billion parameter model (plus a 1.5 billion parameter model to enhance the resolution of the produced images) to learn relationships between images and the text used to describe them. Recently, the DeepMind[3] company has published AlphaFold 2, which is an AI system that accurately predicts 3D models of protein structures and has the potential to accelerate research in every field of biology. Essentially, AlphaFold 2 is a novel machine learning approach that incorporates physical and biological knowledge about protein structures (JUMPER *et al.*, 2021). Besides those systems, 2022 has been marked by an important step towards achieving Artificial General Intelligence (AGI) with Gato (REED *et al.*, 2022), which is a single generalist agent trained with 1.2 billion parameters capable of performing multiple tasks, e.g., playing Atari, caption images, and stack blocks with a real robot arm. And, Flamingo (ALAYRAC *et al.*, 2022), a single visual language model, trained with 80 billion parameters, that sets a new state of the art on a wide range of open-ended multimodal tasks.

All those systems are characterized by requiring intensive computational specifications used to train huge models. In such scenarios, isolated noises are mitigated by the massive volume of data. However, errors or noises embedded into the data as a pattern may propagate accumulative errors to all layers, thus affecting the final models (RAMESH *et al.*, 2022). In deep learning solutions designed to deal with temporal data, such as TS classification, forecasting, and pattern recognition (LAPTEV *et al.*, 2017; GAMBOA, 2017), the presence of noise is even harmful because it does not affect isolated observations. In general, such a problem affects the whole system leading to the production of patterns known as, for example, additive or multiplicative noise.

In this sense, even presenting outstanding results in several practical problems

[2] https://openai.com/

[3] https://www.deepmind.com/

(CURILEM *et al.*, 2018b; CANÁRIO *et al.*, 2020; LEE *et al.*, 2009), the occurrence of noise, naturally present in real-world problems, affects the performance of DNN models. To overcome this issue, researchers are used to model noisy data by performing a two-step analysis: first, a filter is applied to remove noise and, then, models are adjusted on the resultant noise-free data. This process has called our attention and motivated us to investigate how the noise is propagated inside DNNs, as discussed in the following section.

1.2 HYPOTHESIS

Usually, TS collected from real-world systems present influences from stochastic[4] and deterministic[5] components, thus affecting the value of every single observation. As a consequence, high accuracy results depend on modeling those components as discussed in (HAN; LIU, 2009; RIOS; MELLO, 2016). In summary, by only modeling the deterministic component, one may obtain malformed attractors, whereas the individual modeling of the stochastic component tends to underestimate recurrent behavior.

By facing such a situation, researchers decompose noisy time series into stochastic and deterministic components. Then, two main approaches are taken into account (QIU *et al.*, 2017; YANG; CHEN, 2019): i) the stochastic component is considered as noise to be later removed from the analysis; or ii) stochasticity and determinism are combined in a hybrid model, that separately considers such components (RIOS; MELLO, 2013).

In this work, we are focused on understanding better the first approach, in which noise is just removed from the data to avoid affecting model accuracies. Therefore, by taking into account that this task is usually performed before setting a Neural Network, we defined the following hypothesis that guides our research:

> *"A filter created to work as an embedding layer in Long Short-Term Memory Networks can improve the modeling of noisy time series without requiring a previous decomposition/filtering step."*

Aiming at validating this hypothesis, we organized the methodology of our work into four main phases. Firstly, we started a collaborative study with researchers from the Universidad de La Frontera to model signals from a real-world system. In this phase, we have created a new DNN architecture that provided remarkable results to classify signals collected from the Llaima volcano according to their seismic sources. The second phase was planned to investigate different signal transformations usually performed as a pre-processing step. The third phase was structured to investigate the influence of noises during the training of the DNN models. Therefore, we created a controlled scenario to assess the impact of different signal-to-noise ratios on the final model. In summary, we evaluated the model degradation as higher noise was added to the signals. By knowing the impact of this operation, we have met our last methodological phase, which was performed in two parts. In the first one, we used eXplainable Artificial Intelligence (XAI) to visually understand and interpret the noise propagation through the DNN layers. Then, we have

[4] Current observations are influenced by random variables.

[5] Current observations only depend on past ones.

dedicated a great effort to designing a new neural network architecture with embedding noise filtering to suppress the pre-processing step by demonstrating the relevance of our hypothesis.

Finally, after finishing up all methodological phases, our contributions are three-fold: firstly, we created SeismicNet, a new DNN architecture devoted to model volcano signals. Secondly, the study performed with XAI tools allowed us to create a new approach to filter additive noise. Finally, we concluded this thesis with a new architecture that can be plugged into existing neural networks to automatically denoise signals during the training process.

1.3 CHAPTER MAP

This thesis heeds the following organization:

- Chapter 2 presents the research background and the essential issues to comprehend our proposed work and their relations;
- Chapter 3 discusses the research and shows the methodology of our work;
- Chapter 4 describes our experimental scenarios and the performed analysis;
- Chapter 5 presents the conclusion, discussion, and the future works;
- In the Appendix, we describe the path to reach the developed network architecture with embedding noise filtering.

Chapter 2

BACKGROUND

This chapter aims at providing a theoretical background necessary during the development of the research proposed in this project. We organized this chapter into three main sections: firstly, we present an overview about time series, which is the information processing paradigm adopted in this project; then, we show details on Artificial Neural Network, that is the Machine Learning method in which we are interested in investigating the influence of noise on the obtained models; finally, we depict some related researches published so far.

2.1 TIME SERIES

2.1.1 General Overview

Time Serie (TS) organizes sequence of observations over time, which allows modeling and analyzing systems behavior (BOX *et al.*, 2015). The importance of TS in analysis is observed in several areas, such as: Economy, Computing, Biology, Telecommunications, Medicine and Climatology (MORETTIN; TOLOI, 2004; SHUMWAY; STOFFER, 2006; CHATFIELD *et al.*, 2004). In economy, it is possible to use TS analysis to describe stock market behavior, which looks for to model fluctuations to predict product prices and stock values to avoid, for example, stock market failures (GUHATHAKURTA; BHATTACHARYA; CHOWDHURY, 2010; CHAN *et al.*, 1999; CHOI; KULICK; MAYER, 1999). Besides modeling economic systems, TS are widely used as decision making support to model disease symptoms, treatments evolution, tendencies of cancer in individuals, and heartbeat variations (TSCHACHER; KUPPER, 2002; SATO; HOSOKAWA; MAEDA, 2003; SUMMA *et al.*, 2007; ZHUANG *et al.*, 2008; PONOMARENKO *et al.*, 2005).

The use of TS for prediction aims to analyze n past observations ($\{x_{t-n-1}, ..., x_{t-2}, x_{t-1}, x_t\}$) to estimate l observations in future time ($\{x_{t+1}, x_{t+2}, ..., x_{t+l}\}$). The main purpose of this function is to reduce, to maximum extent, the difference between observed and predicted values. On the other hand, the use of TS to estimate transfer functions aims to understand system dynamic behavior, whose input values are represented by means of a TS. In contrast to the predictive approach, study and estimation of transfer functions generally aim to understand current behavior of an observation x_t based on a set of past observations $\{x_{t-k}, ..., x_{t-2}, x_{t-1}\}$. On analysis of effects in events intervention, TS can be used to understand impacts that one system has on another, such as the attempt to understand pollution effect on increase of planet temperature. Finally, the use of TS in control systems aims to monitor a particular process of interest in order to detect possible deviations from normal behavior and then adjust system output for it approaches to expected behavior.

These TS applications require models adjusted on the observation characteristics. Among them, we can cite the number of variables used to compose an observation and time interval between consecutive observations. Regarding the number of variables, TS can be classified as univariate or multivariate. Univariate series are composed of scalar values, sequentially collected while multivariate ones occur when k variables are observed at each time instant t (HAMILTON, 1994).

In relation to interval between collections, Morettin and Toloi (2004) subdivided TS into two classes:

(i) Discrete: analysis done on temporal domain, according to time intervals Δt, periodic and fixed in $\mathbb{N}$;

(ii) Continuous: analysis carried out in frequency domain (time in $\mathbb{R}_+$).

In addition to the number of variables that composes each observation and the interval time, the modeling process also involves the knowledge of components responsible for defining the TS behavior. In this sense, a given time series X_t can be denoted by the sum of three non-observable components $X_t = T_t + S_t + \varepsilon_t$, where T_t represents the trend, S_t the seasonality, and ε_t a random component (MORETTIN; TOLOI, 2004). These components, $\{T_t,\ S_t,\ \varepsilon_t\}$, are called unobservable because they are not collected directly from a system but inferred through temporal relations between observations. By understanding these components, it can be possible evaluate global aspects of TS, such as stochasticity, stationarity and linearity, which are important to provide more accurate TS models.

Stochastic series are composed of observations and random relations that follow probability density functions and can change over time, making it difficult to model their events. On the other hand, deterministic series predominantly present observations with strict dependence on past values. Stationary series are in a particular statistical equilibrium state (BOX *et al.*, 2015), that is, they develop around a constant average (MORETTIN; TOLOI, 2004). TS whose observations are modeled by stochastic processes, which do not satisfy the stationary condition, are denominated non-stationary.

Time series can also be classified by their linearity, a rule that defines their observations in linear and non-linear. Linear TS are those whose observations are composed of a linear combination of past occurrences and noises. Therefore, the linearity of a series is present in model, map, or process that originated it. In turn, non-linear series are formed by processes of non-linear combination of observations and past noises. After understanding these essential aspects of TS, one can select a subset of more appropriate techniques to model and understand their most representative behaviors. In this sense, the following presents some techniques widely used to decompose and transform TS section.

2.1.2 Time Series Decomposition

TS usually has a variety of patterns, and it is commonly valuable to deconstruct the TS data through statistical tasks into several components, each representing an underlying pattern type (HYNDMAN; ATHANASOPOULOS, 2018). Often the TS decomposition is done to help of the understanding a TS, but it can also be used to improve Machine Learning (ML) algorithms performance. In this section, we consider some common methods for extracting those components that we used during our research.

2.1.2.1 Fourier Transform

We can describe a Fourier Transform (FT) as a generalization of the complex Fourier series (FS) that transforms a function of time, $f(t)$, to a function of frequency, $F(\omega)$ (OSGOOD, 2002). Recall that for a general function $f(t)$ of period T the FS has the form

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{2\pi int/T}, \tag{2.1}$$

so that the frequencies are $0, \pm 1/T, \pm 2/T$, and so on, making the frequency terms more packed as the T increases. Accordingly, we describe the $n-th$ Fourier coefficient as

$$c_n = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} e^{-2\pi int/T} f(t)dt. \tag{2.2}$$

Then, in the limit as $T \to \infty$ we can replace n/T by s and, despite the several common conventions on definition of FT as an integrable function $f : \mathbb{R} \to \mathbb{C}$, formally define FT of a function $f(t)$ as

$$\hat{f}(s) = \int_{-\infty}^{\infty} e^{-2\pi ist} f(t)dt, \forall s \in \mathbb{R}. \tag{2.3}$$

As the FT of TS can not describe how the TS spectral content changes over time, which is critical in many non stationary signals, the time variable is introduced in the FT analysis to provide a proper description of how the spectral content changes as a function of time. Hence, the time-frequency transform could provide direct information about frequency components at instant time t (COHEN, 1995; GRÖCHENIG, 2001).

In our studies, we chose to work with the Short-Time Fourier Transform (STFT), an alternate form of FT, given its simplicity and results in TS processing analysis (CHIKKERUR; CARTWRIGHT; GOVINDARAJU, 2007). The STFT uses a sequence of FT of a windowed TS providing the local time-frequency information for TS local sections analysis as it changes over time. Thus, in the discrete domain, the STFT can be denoted by

$$SP[x(t)](n,k) = |\sum_{m=0}^{N-1} x(m) \cdot w(m-n) \cdot e^{i2\pi mk}|^2, \tag{2.4}$$

where $x(t)$, and $w(t)$ represent the TS and the STFT sliding window, respectively.

However, the fixed resolution of the STFT could be an issue. As the width of the windowing function describe how it represents the TS, it directly determines whether there is a good frequency or time resolution. Which denotes if the frequency components close together can be separated or the time at which frequencies change. Therefore, a wide and a narrower window gives better frequency resolution but poor time resolution and good time resolution but poor frequency resolution, respectively. Then, to overcome this issue, the wavelet transforms are chosen, which give good time resolution for high-frequency events and good frequency resolution for low-frequency events. In the next section, we further describe the wavelet transform.

2.1.2.2 Wavelet Transform

Wavelets are mathematical functions designed to decompose signals into different scale and resolution levels (GRAPS, 1995). They work quite similar to FT, that allows the study of signals in frequency domain. However, despite of its utility, the temporal information from signals is completely lost by the Fourier transform, making difficult to distinguish transient relations and identify when structural changes do occur over time.

Wavelets, on the other hand, have been proposed to estimate spectral characteristics from signals taking into account information from temporal and spacial scale. By using wavelets, signals are stretched into long wavelet functions to measure low frequency influences and are compressed into short wavelet functions to measure high frequency ones. Also, instead of using sine and cosine functions like Fourier, wavelets use wave forms as base functions to extract coefficients from signals. In summary, wavelet coefficients present two main patterns: the first one works as a smooth filter, whereas the second carries signal details. Such details represent fluctuations with high frequency which can be considered as noise. In our work, we studied discrete and continuous Wavelet transforms, as detailed in the following sections.

Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) is a wavelet transform that decomposes a TS into a number of wavelet resolutions, where each resolution is a TS of coefficients describing the time evolution of the signal in the corresponding frequency band (GRAPS, 1995). Through dilations and translations of a "mother function" $\Phi(x)$ it can be defined an orthogonal basis, such as

$$\Phi_{s,l}(x) = 2^{\frac{s}{2}}\Phi(2^{-s}x - l), \tag{2.5}$$

where s and l represent scale and dilation of the mother function Φ that generate the wavelets. Moreover, the scale indicates the wavelet width, whereas dilation gives its position. Similarly, we can notice in the Equation 2.5 that the mother functions are rescaled by power of two and translated by an integer. Consequently, this make the wavelet bases especially interesting because of the self-similarity generated by the scales and dilations. Thus, as we know about the mother's wavelet function, it possibly learn everything about the basis.

The analyzing wavelet is used in the scaling equation to span the data domain at different resolutions:

$$W(x) = \sum_{k=-1}^{N-2} (-1)^k c_{k+1} \Phi(2x + k), \tag{2.6}$$

where $W(x)$ is the scaling function of the mother function Φ and c_k represent the wavelets coefficients. Furthermore, to satisfy the linear and quadratic constraints, the wavelet coefficients follow

$$\sum_{k=0}^{N-2} c_k = 2, \tag{2.7}$$

$$\sum_{k=0}^{N-2} c_k c_{k+2l} = 2\delta_{l,0} \tag{2.8}$$

where δ is the delta function, and l is the location index.

The wavelets coefficients $\{c_0, \cdots, c_n\}$ work as a bank filter that is placed in a transformation matrix to apply in a raw data vector. The data is decomposed simultaneously through a low pass filter with impulse response g and a high-pass filter h that outputs, respectively, the detail coefficients and approximation coefficients. At each level of the decomposition process, expressed by the convolutional symbol $(*)$ at Equations 2.9 and 2.10, the filter output of the low-pass filter g is subsampled by two and further processed by passing it again through a new low-pass filter g and a high-pass filter h with half the cut-off frequency of the previous one as denoted by

$$y_{\text{low}} = (x * g) \downarrow 2, \tag{2.9}$$

$$y_{\text{high}} = (x * h) \downarrow 2. \tag{2.10}$$

In this work, we used a DWT from Daubechies family (DAUBECHIES, 1988) with a filter Least Asymmetric. Moreover, the length of wavelet and scaling filters was equals to 8. In order to easily understand the effect of such transformation on a signal, Figure 2.1 shows a TS $(x(t))$. The following figure $(V_1(t) - V_8(t))$ illustrate the wavelet coefficients varying the scale resolution from 1 to 9. As the resolution level increases, more details are removed from the signal. Consequently, the number of observations also reduces in a factor by 2 (see the x-axis scale in all plots). For instance, by comparing $V_1(t)$ to original signal $x(t)$, we notice the general behavior persists, but the number of observations reduces from $4,000$ to $2,000$. As the resolution grows, less observations and information are considered.

Continuous Wavelet Transform

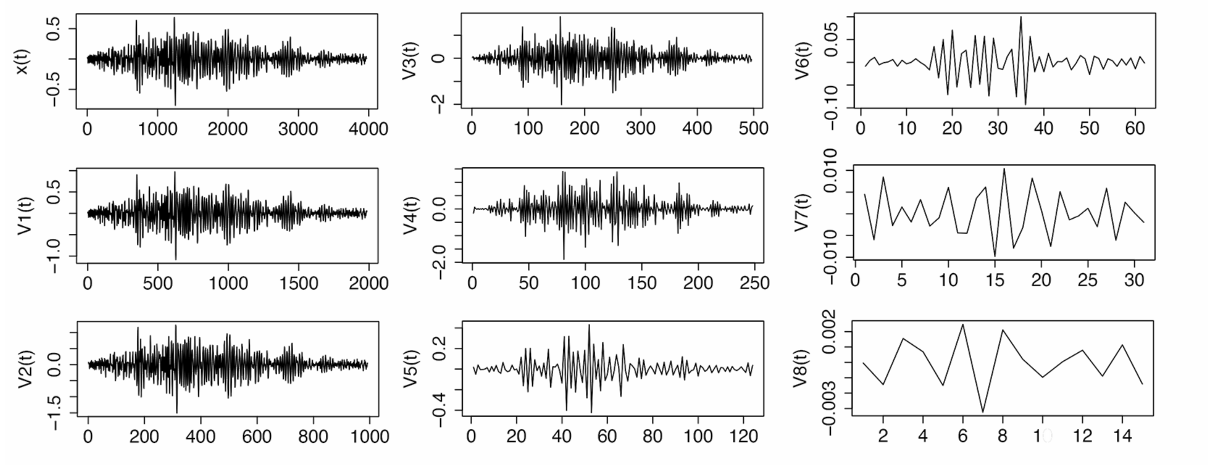


Figure 2.1 $x(t)$ is a TS and $V_1(t) - V_8(t)$ represent different wavelet resolutions.

The Continuous Wavelet Transform (CWT) is a formal tool that provides a complete representation of a signal by continuously translating and scaling wavelet functions. CWT is used to decompose a signal into wavelets and is considered strongly robust to the presence of outliers and noises (SLAVIČ; SIMONOVSKI; BOLTEŽAR, 2003).

The application of CWT on a signal $x(t)$ is expressed by

$$X_w(i,j) = \frac{1}{|i|^{1/2}} \int_{-\infty}^{\infty} x(t)\overline{\psi}\left(\frac{t-j}{i}\right) dt, \tag{2.11}$$

in which d represents a scale $(i > 0)$ $i \in \mathbb{R}^{+*}$, $j \in \mathbb{R}$ is a translational value, $\psi(t)$ is a continuous function in both time and frequency domain called the mother wavelet, and the $\overline{\psi}$ is its complex conjugate.

The main purpose of mother wavelet is to provide a source function to generate daughter wavelets which are simply translated and scaled versions of mother wavelet. In this work, the continuous wavelet transform was computed with the complex-valued Morlet wavelet as discussed in the next section.

Morlet Wavelets

The Morlet Wavelets belong to a one-parameter family of functions firstly introduced by Goupillaud, Grossman and Morlet (1984) and given by

$$\psi_{\omega_0}(t) = Ke^{i\omega_0 t}e^{-\frac{t^2}{2}}, \tag{2.12}$$

where ψ_{ω_0} is a complex sinusoid of the angular frequency ω_0 (damped by a Gaussian envelope).

However, such equation cannot be considered an actual wavelet once it fails to satisfy the admissibility condition as discussed in (CHATTERJEE, 1986). In order to fulfill this gap, a correction term can be added as shown in Equation 2.13.

$$\psi_{\omega_0}(t) = K(e^{i\omega_0 t} - e^{-\frac{\omega_0^2}{2}})e^{-\frac{t^2}{2}}. \tag{2.13}$$

Moreover, aiming at confirming $\psi_{w0}(t)$ respects the unit energy condition, the normalizing constant K must be chosen by considering Equation 2.14.

$$K = \pi^{-\frac{1}{4}}. \tag{2.14}$$

The Morlet Wavelet became the most popular complex-valued wavelets mainly due to its four properties. Firstly, it can be treated as an analytic wavelet. Secondly, the peak frequency $(\omega_{\cdot}^{P})$, energy frequency $(\omega_{\cdot}^{E})$, and central instantaneous frequency $(\omega_{\cdot}^{I})$ are given by Equation 2.15, which makes easier the conversion from scales to frequencies.

$$\omega_{\psi_{\omega_0}}^{P} = \omega_{\psi_{\omega_0}}^{E} = \omega_{\psi_{\omega_0}}^{I} = \omega_0. \tag{2.15}$$

Thirdly, the Heisenberg box area reaches its lower bound with this wavelet. In this sense, Morlet Wavelet has optimal joint time-frequency concentration. Lastly, time radius and frequency radius are defined by Equation 2.16, representing the best compromise between time and frequency concentration.

$$\sigma_t; \psi_{\omega_0} = \sigma_\omega; \psi_{\omega_0} = \frac{1}{\sqrt{2}}. \tag{2.16}$$

Figures 2.2 and 2.3 exemplify the output produced by the Morlet Wavelet on a single TS. As one can notice, by varying the number of octaves, it is possible to scale the obtained complex values. On the other hand, by changing the number of scales, we can control how smooth is the transform.

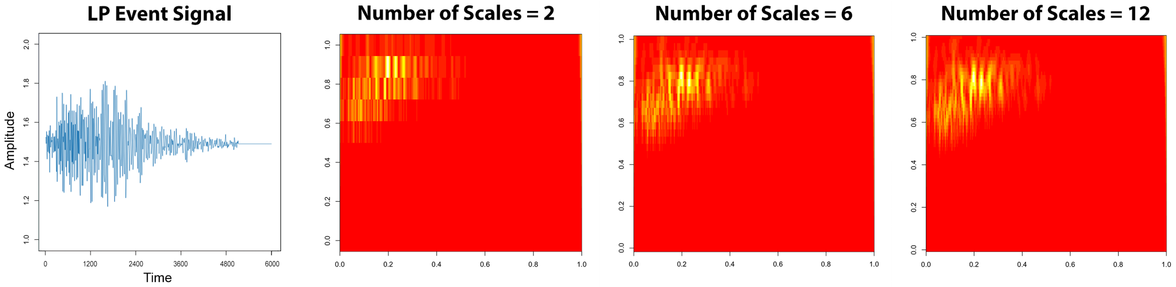


Figure 2.2 From left to right a TS followed by three CWT transformations using a Morlet Wavelet as mother wavelet with their respective number of scales in each octave.

As previously mentioned, the CWT output yields a set of complex values. Instead of just using the real part, we adopted their modules as defined in Equation 2.17, in which a and bi are the real and imaginary part, respectively.

$$Mod(\psi_{\omega_0}(t)) = \sqrt{a^2 + bi^2}. \tag{2.17}$$

The better understand the advantage of using this equation, Figure 2.4 shows the difference between the modulus and the real part calculated by CWT from a TS.

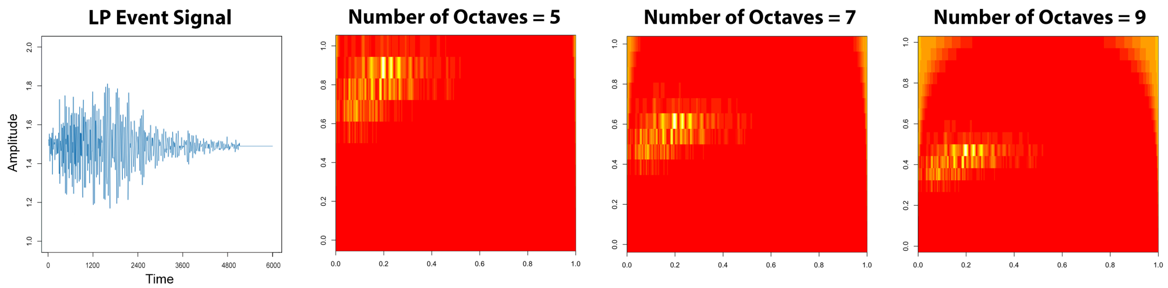


Figure 2.3 TS followed by three CWT transformations using a Morlet Wavelet as mother wavelet with changing only the number of octaves each.

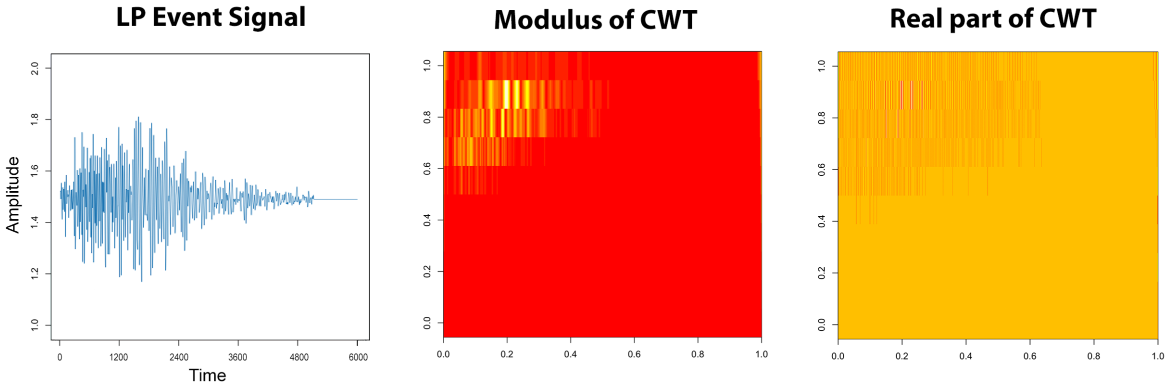


Figure 2.4 TS followed by the modulus and real part of CWT transformation.

2.2 ARTIFICIAL NEURAL NETWORKS

2.2.1 General Overview

Artificial Neural Network (ANN) are computing systems based on connected processing unities inspired in biological neurons (GERVEN; BOHTE, 2018). The first ANN, also known as perceptron, was developed by (ROSENBLATT, 1961), considering an earlier work published by (MCCULLOCH; PITTS, 1943). In summary, this network is composed of a single neuron designed to process inputs $(x_1, x_2, x_3, \dots, x_n)$ and produce a binary output y that can be specially used in linearly separable problems. Such output can also be transmitted to other neurons, simulating synapses in biological brains. In ANNs, a synapse is commonly determined by an activation function adopted to limit the neuron output as exemplified in Equation 2.18. This equation is a step function with 4 basic elements (HAYKIN; NETWORK, 2004): i) synaptic weights (w_i), that quantifies the importance of every input (x_i), by applying a dot product; ii) an adder working as a linear operator that combines all inputs and weights; iii) an external bias (b) used to apply an affine transformation on the adder output; and iv) a threshold (τ) to define the neuron output in terms of induced local field.

$$y = \begin{cases} 0 & if \quad v \leq \tau \\ 1 & if \quad v > \tau \end{cases} \quad \text{, such that} \quad v = \sum_i w_i \cdot x_i + b. \tag{2.18}$$

According to the literature, the main problem faced by this activation function is the all-or-none property which provides flipping results. An alternative, widely adopted in ANNs, is the sigmoid function, also referred to as s-shaped graph. Normally, a special case of sigmoid function, called logistic function, is considered due to its ability of providing values within the interval $[0, 1]$, as shown in Equation 2.19.

$$y = \frac{1}{1 + e^{-x}}. \tag{2.19}$$

The ANN learning process may be computed using a first-order iterative optimization that updates the parameters by considering a gradient descent algorithm. This algorithm looks for a network configuration in which some error measure achieves its global minimum. In summary, the algorithm is based on the assumption that if a multi-variable function $C(\cdot)$ is defined and differentiable in a neighborhood of a point w_i, then $C(w_i)$ decreases as it approaches w_i in the direction of the negative gradient of C at w_i, $-\nabla C(w_i)$, as presented in Equation 2.20.

$$w_i' = w_i - \eta \nabla C(w_i). \tag{2.20}$$

In this equation, if term $\eta \nabla C(w_i) \in \mathbb{R}_+$ is subtracted from w_i, then the learning optimization moves against the gradient towards the minimum as illustrated in Figure 2.5.

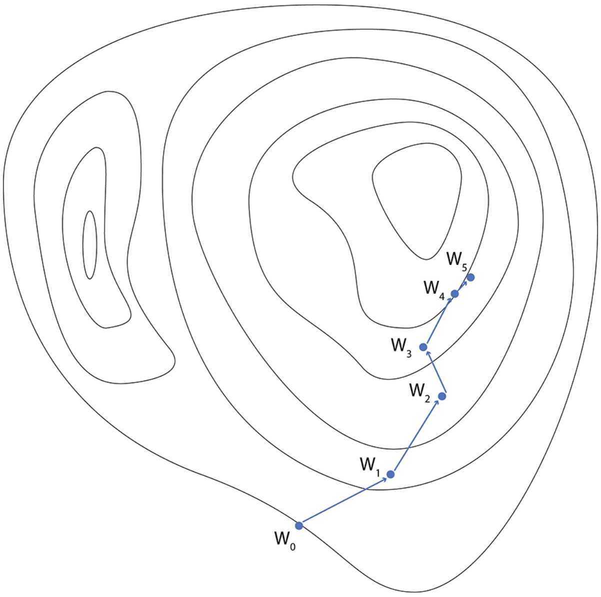


Figure 2.5 Example of gradient descent on a level sets series.

Considering the architecture, ANN can be composed of multiples neurons organized in layers, as shown in Figure 2.6, in which the left-most one is used to deal with inputs.

On the other hand, the right-most layer contains the output neurons. Finally, in the middle, neurons can be configured in one or multiple layers, also referred to as hidden layers, aiming at extracting implicit information from data.

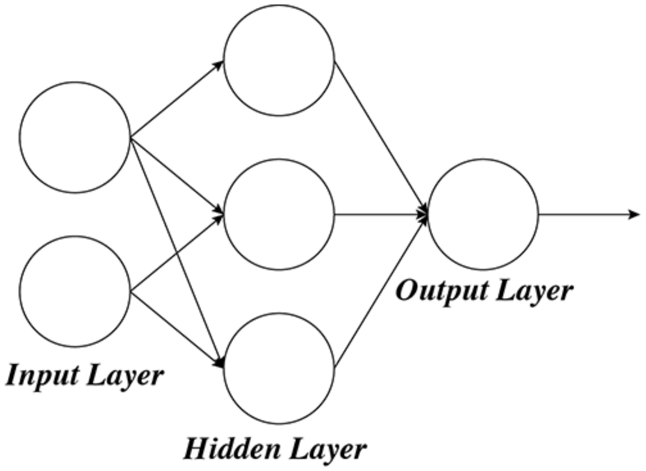


Figure 2.6 An Artificial Neural Network example composed by three layers. From left to right: the input, hidden and output layer.

Although the first motivation of creating ANN was somehow inspired in the strong Artificial Intelligence (AI) hypothesis, computers thinking as human brains, we noticed its main recent success is focused on solve specific problems such as computer vision (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), speech recognition (HINTON *et al.*, 2012), playing board games (SILVER *et al.*, 2016), and medical diagnosis (AMATO *et al.*, 2013). The following sections present the ANN architectures considered in our study.

2.2.2 Multilayer Perceptron

Multilayer Perceptron (MLP) is a well-known type of artificial neural network composed of, at least, three layers: input, hidden, and output. The neurons adopted in such network use non-linear activation functions (as shown in Equation 2.19), whose outputs are connected to neurons from the following layer in a nonlinear mapping. Another important aspect related to MLP is the complexity of defining the number of neurons and hidden layers. In this sense, Stathakis (2009) present a discussion on heuristics that can be adopted to overcome this issue.

MLP is also viewed as a logistic regression classifier, that propagates the input data through layers in a non-linear manner. In summary, this step transforms the data into a space where they are linearly separable. Depending on the task, a single hidden layer is sufficient to be used as universal approximator. An example of a MLP is shown on Figure 2.7. Later, in Section 4.1, we present a set of experiments using MLP to classify the TS.

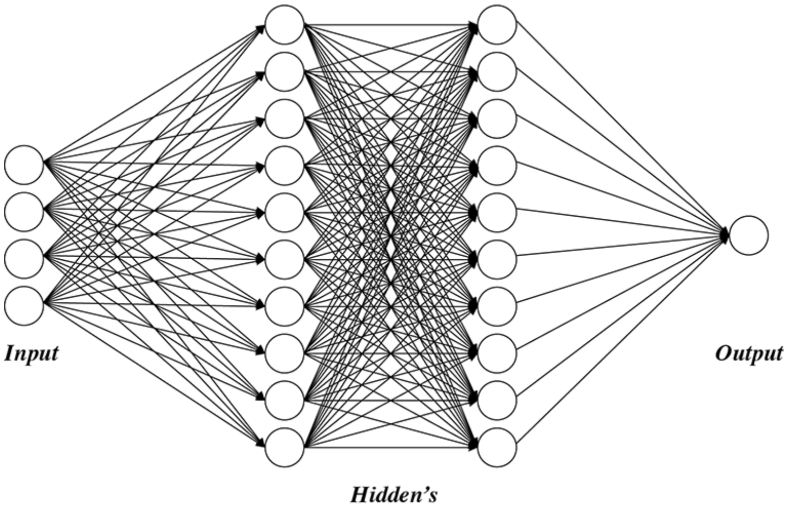


Figure 2.7 A MLP example. This MLP is composed as follow: the leftmost and rightmost are the input and output layers, respectively, and in the middle of network there are two hidden layers.

2.2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) is very similar to ordinary Neural Network described in previous section. In general, CNN is a multilayer network capable of recognizing patterns with extreme variability, distortions, and geometric transformations (LECUN *et al.*, 1998). Its architecture is built up using neurons with adaptable weights and biases, following the same learning steps previously presented: neurons receive data as input, apply an adder on inputs, weights and biases, and calculate outputs using activation functions. The main difference is related to the hidden layers, which can be organized by considering four different architectures (KRIZHEVSKY; SUTSKEVER; HINTON, 2012): i) convolutional; ii) activation; iii) pooling; and iv) full connection.

The convolutional layer is the core building block of CNNs and requires the heaviest computational load during the learning process. The layer parameters consist of a set of learnable filters spatially small, designed to deal with information in 3 dimensions: width, height, and depth. During a forward pass, each filter is convolved across the width and height of the input volume by computing its dot product at any position. This convolutional step produces a 2-dimensional activation map, providing responses in every spatial position. Intuitively, the learning process in this layer updates filters to recognize features (e.g. edges, colors, and patterns) and produce output volumes. Figure 2.8 shows the described operations of convolutional layer.

In addition to filters, convolutional layers use two other hyperparameters for controlling the output volume size. Firstly, the stride which defines the way the filter will be slided on inputs from the convolutional layer. The greater this parameter is, the smaller output volume is. Secondly, the zero-padding parameter fulfills the input volume with zeros around the border.

Activation layers apply a function on neuron outputs to avoid learning problems as, for

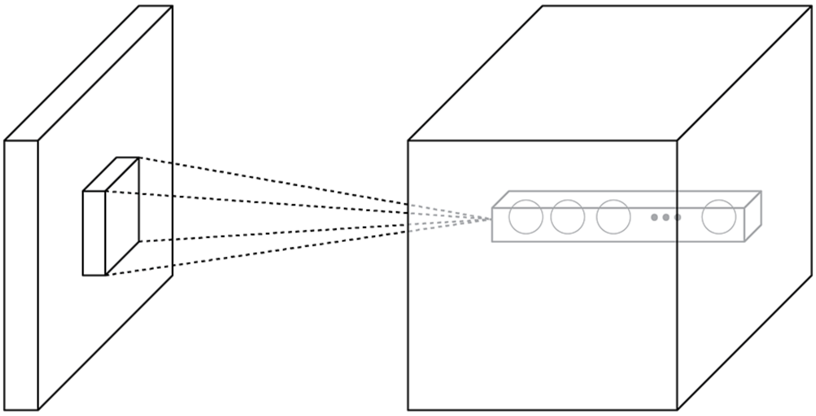
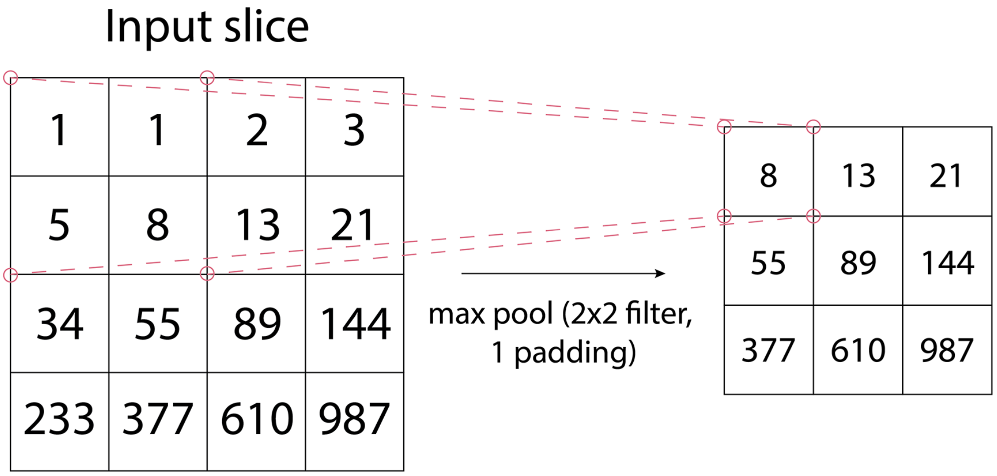


Figure 2.8 On left, the input layer followed by an convolutional layer. As it is observed, multiple neurons along the depth are connected at the same region of input layer.

instance, getting stuck near zero or indefinitely growing up. Rectified Linear Unit (ReLU) is a typical example of activation layer, which sets all negative values in a matrix or vector to zero while all other values are constantly kept.

Pooling layers perform a downsampling operation along spatial dimensions. These layers operate on every depth slice of input, spatially resizing it by using, for instance, a MAX operation.

All these layers are usually set considering 2 × 2 filters and stride of 2 downsampling every depth slice by shifting two units at a time and disregarding 75% of activations. In this case, MAX operation gets the maximum value of 4 numbers from a 2 × 2 region in some depth slice. Figure 2.9 presents an example of a max pooling operation over a matrix input data.



1	1	2	3
5	8	13	21
34	55	89	144
233	377	610	987

8	13	21
55	89	144
377	610	987

Figure 2.9 On left, an input matrix and, on right, the max pooling operation result. On this example, each max operation was taken over 4 numbers that was formed by a $2x2$ filters size.

Finally, the fully connected layers, also referred to as dense layers, are used to compute the class scores. They work as ordinary MLP, in which every neuron in a layer is completely connected to all neurons in the previous and following ones. In this work, we used multiple types of CNNs architectures that will be detailed through the Chapter 4.

2.2.4 Recurrent Neural Networks

Recurrent Neural Network (RNN) is a special type of ANN with feedback loop, in which outputs are also used as neuron inputs, allowing to persist some information during the training process. Figure 2.10 summarizes this network showing that neurons in a given layer receive inputs and yield outputs that are transmitted to the following layer and used again to feed such neurons along with the next inputs.

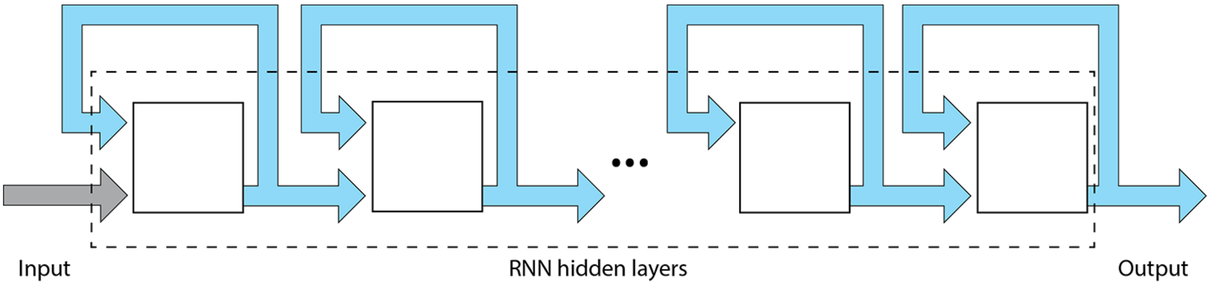


Figure 2.10 A summary of RNN, where each hidden layer output yields to the same hidden layer input and the next one as well.

RNNs have been widely adopted in different tasks such as speech recognition, language modeling, and signals (time series) prediction. The most used RNN architecture is the Long Short-Term Memory (LSTM) network, which was designed to deal with long-term dependency issues.

LSTM was introduced by (HOCHREITER; SCHMIDHUBER, 1997) and widely adopted to deal with huge entries in several remarkable researches such as (GERS; SCHMIDHUBER; CUMMINS, 1999; GRAVES; SCHMIDHUBER, 2005; GREFF *et al.*, 2017).

Similarly to standard RNNs, LSTM has a repeating block, also known as memory cell, that is created by using a different composition: i) input, forget, and output gates; ii) block input; iii) single Constant Error Carousel (CEC); iv) output activation function; v) peephole connections; and vi) the block output recurrently connected back to the block input and all gates. The described arrangement is show in Figure 2.11.

To better explain this network, let x^t be the input vector and y^{t-1} the output vector at time t. Then, the block input it can be written as shown in Equation 2.21, in which $\mathbf{W}_z, \mathbf{R}_z$, and $\mathbf{b}_z$ are the input, recurrent, and bias weights, respectively. In such equation, $g(\cdot)$ is a *hyperbolic tangent* activation function defined as $g(x) = \tanh(x)$.

$$\mathbf{z}^t = g\left(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z\right). \tag{2.21}$$

The input gate with its input, recurrent, peephole, and bias weights ($\mathbf{W}_i, \mathbf{R}_i$, $\mathbf{p}_i$ and $\mathbf{b}_i$, respectively) is described in Equation 2.22, such that $\sigma(\cdot)$ is the *logistic sigmoid* ($\sigma(x) = \frac{1}{1+e^{-x}}$) activation function.

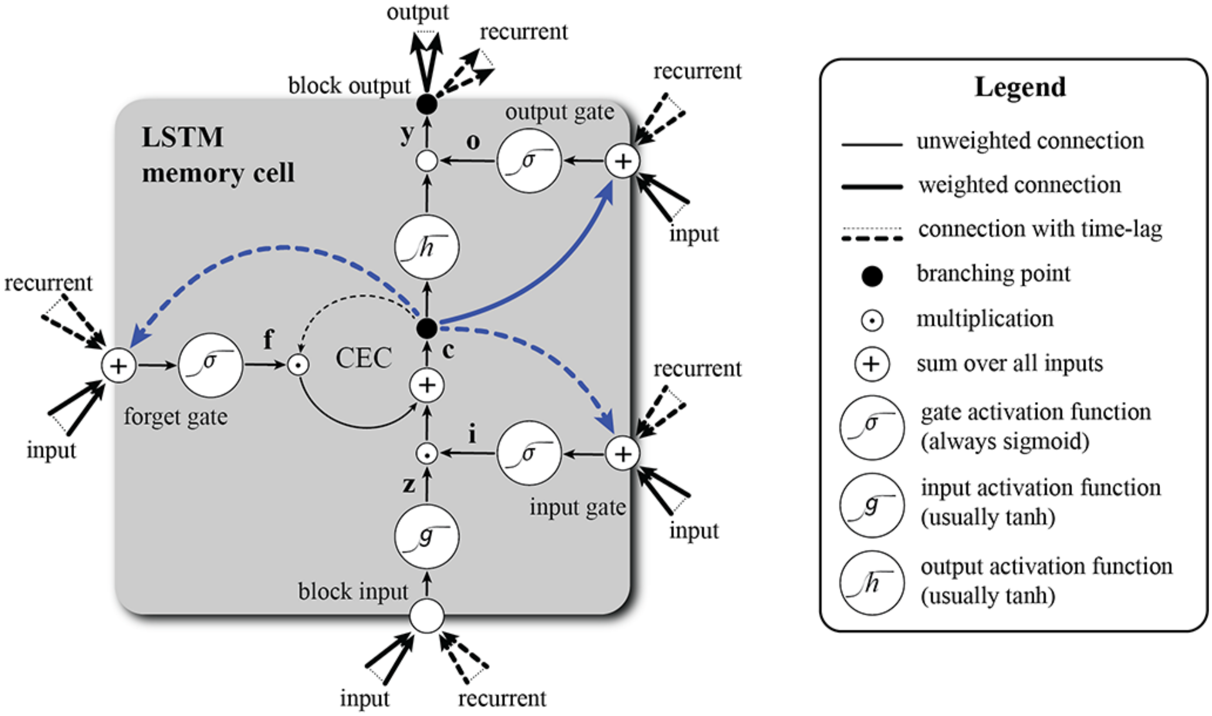


Figure 2.11 LSTM memory cell (adapted from Greff *et al.* (2017).

$$\mathbf{i}^t = \sigma\left(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i\right). \tag{2.22}$$

The forget gate is defined in Equation 2.23, in which $\mathbf{W}_f, \mathbf{R}_f$, $\mathbf{p}_f$, and $\mathbf{b}_f$ stand for the input, recurrent, peephole, and bias weights, respectively, as well as the $\odot$ denotes the element-wise product.

$$\mathbf{f}^t = \sigma\left(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f\right). \tag{2.23}$$

Similarly, the output gate is presented in Equation 2.24, being $\mathbf{W}_o, \mathbf{R}_o$, $\mathbf{p}_o$, and $\mathbf{b}_o$ the input, recurrent, peephole and bias weights, respectively.

$$\mathbf{o}^t = \sigma\left(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o\right) \qquad , \tag{2.24}$$

On the other hand, we can use Equation 2.25 to compute CEC, taking into account $\odot$ as a point-wise multiplication of two vectors.

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t \qquad , \tag{2.25}$$

Finally, the memory cell output it computed by Equation 2.26 such that $h(\cdot)$ is also a *hyperbolic tangent* ($h(x) = tanh(x)$) activation function.

$$\mathbf{y}^t = h\left(\mathbf{c}^t\right) \odot \mathbf{o}^t \qquad , \tag{2.26}$$

Then, the deltas for the LSTM memory cell are calculated by Equation 2.27, such than Δt is a vector of deltas received from the higher layer.

$$
\begin{array}{rcl}
\delta \mathbf{y}^t & = & \Delta^t + \mathbf{R}_z^T \delta \mathbf{z}^{t+1} + \mathbf{R}_i^T \delta \mathbf{i}^{t+1} + \mathbf{R}_f^T \delta \mathbf{f}^{t+1} + \mathbf{R}_o^T \delta \mathbf{o}^{t+1} \\
\delta \overline{\mathbf{o}}^t & = & \delta \mathbf{y}^t \odot h\left(\mathbf{c}^t\right) \odot \sigma'\left(\overline{\mathbf{o}}^t\right) \\
\delta \mathbf{c}^t & = & \delta \mathbf{y}^t \odot \mathbf{o}^t \odot h'\left(\mathbf{c}^t\right) + \mathbf{p}_o \odot \overline{\mathbf{o}}^t + \mathbf{p}_i \odot \delta \mathbf{i}^{t+1} \\
 & & +\mathbf{p}_f \odot \delta \overline{\mathbf{f}}^{t+1} + \delta \mathbf{c}^{t+1} \odot \mathbf{f}^{t+1} \\
\delta \overline{\mathbf{f}}^t & = & \delta \mathbf{c}^t \odot \mathbf{c}^{t-1} \odot \sigma'\left(\overline{\mathbf{f}}^t\right) \\
\delta \overline{\mathbf{i}}^t & = & \delta \mathbf{c}^t \odot \mathbf{z}^t \odot \sigma'\left(\overline{\mathbf{i}}^t\right) \\
\delta \overline{\mathbf{z}}^t & = & \delta \mathbf{c}^t \odot \mathbf{i}^t \odot g'\left(\overline{\mathbf{z}}^t\right).
\end{array}
\tag{2.27}
$$

The loss function E formally corresponds to $\frac{\partial E}{\partial \mathbf{y}^t}$ without including recurrent dependencies. Moreover, input deltas are only necessary when there is a layer below that requires to be trained. Such deltas can be calculated by using Equation 2.28.

$$
\delta \mathbf{x}^t = \mathbf{W}_z^T \delta \overline{\mathbf{z}}^t + \mathbf{W}_i^T \delta \overline{\mathbf{i}}^t + \mathbf{W}_f^T \delta \overline{\mathbf{f}}^t + \mathbf{W}_o^T \delta \overline{\mathbf{o}}^t \qquad . \tag{2.28}
$$

Finally, the weight gradients are computed by Equations 2.29, 2.30, 2.31, 2.32, 2.33, and 2.34, in which $\star$ can be any of $\{\overline{\mathbf{z}}, \overline{\mathbf{i}}, \overline{\mathbf{f}}, \overline{\sigma}\}$ and $\langle \star^{\cdot}, \star^{\cdot} \rangle$ represents the outer product between vectors.

$$
\delta \mathbf{p}_i = \sum_{t=0}^{T-1} \mathbf{c}^t \odot \delta \overline{\mathbf{i}}^{t+1}, \tag{2.29}
$$

$$
\delta \mathbf{p}_f = \sum_{t=0}^{T-1} \mathbf{c}^t \odot \delta \overline{\mathbf{f}}^{t+1}, \tag{2.30}
$$

$$
\delta \mathbf{p}_o = \sum_{t=0}^{T} \mathbf{c}^t \odot \delta \overline{\mathbf{o}}^t, \tag{2.31}
$$

$$
\delta \mathbf{W}_\star = \sum_{t=0}^{T} \left\langle \delta \star^t, \mathbf{x}^t \right\rangle, \tag{2.32}
$$

$$
\delta \mathbf{R}_\star = \sum_{t=0}^{T-1} \left\langle \delta \star^{t+1}, \mathbf{y}^t \right\rangle, \tag{2.33}
$$

$$
\delta \mathbf{b}_\star = \sum_{t=0}^{T} \delta \star^t . \tag{2.34}
$$

2.3 INTERPRETABILITY

2.3.1 General Overview

This section briefly presents an overview of interpretability techniques designed explicitly for ANNs. Such techniques have been considered a hot topic in the AI area once the current requirements of real-world systems depend on the general performance and the steps taken to perform a prediction. In summary, interpretability uses a set of visualization tools to explain how these black-box methods execute classification tasks. Consequently, the decision process is better detailed, thus improving models and detecting patterns that influence their executions.

Generally, AI experts use such explanation mechanisms to, for example, understand how layers of a deep network respond to specific input data or to debug/validate a model. In contrast, non-experts often use them to comprehend how some model outputs are produced to ensure trustworthiness, no bias, or compliance with the regulation. In summary, the object of explainability methods is often determined according to the users' goals (TOMSETT *et al.*, 2018).

The recent ANN literature on eXplainable Artificial Intelligence (XAI) is based on different mechanisms as, for instance, perturbation analyses (RIBEIRO; SINGH; GUESTRIN, 2016), backward propagation (BACH *et al.*, 2015), proxy models (RIBEIRO; SINGH; GUESTRIN, 2018), and activation optimization (MONTAVON; SAMEK; MÜLLER, 2018).

To investigate ANNs models, authors widely use backward propagation methodologies such as DeepLIFT (SHRIKUMAR; GREENSIDE; KUNDAJE, 2017), SmoothGrad (SMILKOV *et al.*, 2017), Integrated Gradients (IG) (SUNDARARAJAN; TALY; YAN, 2016), Guided Backprop (GB) (SPRINGENBERG *et al.*, 2014), and Layer-wise Relevance Propagation (LRP) (BACH *et al.*, 2015). The backward propagation mechanism works layer by layer from the output to the input layer, thus estimating the contribution of all neurons to the probability yielded by the logit function used in the last layer (e.g., softmax).

The methods mentioned above mainly differ on the calculation of probability contributions in the last layer. LRP and DeepLIFT compute such contributions by decomposing target neuron activation values into constituent values coming from previous layers using a decomposition approach. Alternatively, the GB and SmoothGrad methods estimate the contributions based on partial gradients of a target neuron activation concerning previous layer neurons through a sensitivity analysis approach. Finally, the IG method achieves the contribution estimation multiplying sensitivity-based contributions by activating previous layer neurons. In this work, we decided to use LRP due to its relevant results in interpreting a non-linear classifier, thus increasing the trust in predictions and identifying potential data selection biases. The following section presents an overview of the chosen method.

2.3.2 Layer-wise Relevance Propagation

In a nutshell, LRP is a visualization method used to identify the importance of pixels through a backward pass in Deep Neural Network (DNN), in which neurons that contribute the most to the higher layer receive greater relevance (BACH *et al.*, 2015; KOHLBRENNER *et al.*, 2020). Thus, using the LRP, it is possible to visually identify where the DNN is looking at. Also, it is worth emphasizing that the propagation rules used by LRP can be understood, for many architectures, as a Deep Taylor Decomposition of the prediction. Figure 2.12 illustrates the LRP procedure.

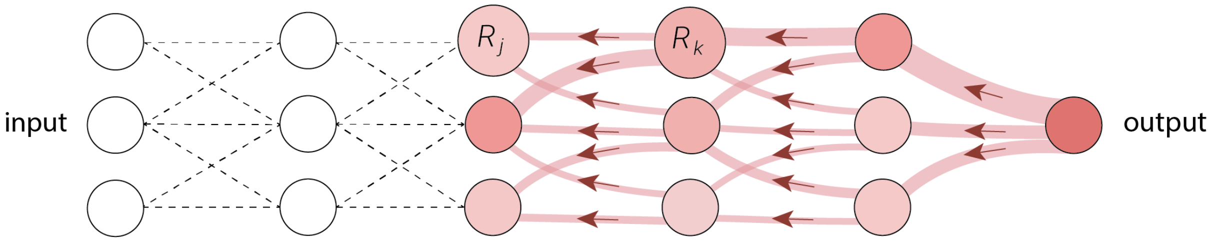


Figure 2.12 The LRP procedure (adapted from Montavon *et al.* (2019)).

As shown in this figure, the principal LRP procedure implements from the right to the left of the illustrated network. In the first step, it creates a list to store relevance scores at each layer. Top-layer activations are first multiplied by a label indicator in order to retain only the evidence for the actual class. Thus, it can then be propagated backward in the network by applying propagation rules at each layer. Also, as the layers are composed of a collection of 2D feature maps, their relevance scores can be visualized as a 2D heatmap.

As discussed in Jung, Han and Choi (2021) , LRP can be explained by considering an output value z_c^L for a given class c produced by an ANN with L layers. A possible rule to calculate the LRP scores is defined by Equation 2.35.

$$R_n^{(l)} = \sum_m \frac{\left(x_n^{(l)} w_{nm}^{(l,l+1)}\right)^+}{\sum_{n'} \left(x_{n'}^{(l)} w_{n'm}^{(l,l+1)}\right)^+} R_m^{(l+1)} \tag{2.35}$$

In this equation, nodes in layers l and $l+1$ are represented by $\{1,\cdots,n,\cdots,N\}$ and $\{1,\cdots,m,\cdots,M\}$, respectively. $R_n^{(l)}$ and $x_n^{(l)}$ are the relevance score and the input value for the n-*th* node in layer l, respectively. $w_{nm}^{(l,l+1)}$ means the weight connecting layers l and $l+1$, and $+$ represents that only positive values are used. The initial relevance score is defined as $R_n^{(L)} = z_c^L$, if $n = c$, or $R_n^{(L)} = 0$, otherwise. From this score, LRP calculates the scores for all layers, in a backward phase, until finding $R^{(1)}$.

2.4 RELATED WORK ON MODELING NOISY TIME-SERIES

Following a broader line of research, Kalapanidas *et al.* (2003) demonstrated in their study, a variety of results of ML algorithms tested on artificially-noisy datasets. They show a existing concern about to form a general guidelines useful to select the best machine learning algorithm for modeling or predicting noisy data. Furthermore, Nettleton, Orriols-Puig and Fornells (2010) also conducted a research, by analyzing the effect of different types of noise on the precision of supervised learning techniques. In that work, the authors are specifically concerned about the classic classifiers performance, thus analyzing how four supervised learners are affected by different degrees of noise. On the other hand, Seltzer, Yu and Wang (2013) conducted an investigation on the noise robustness of DNN based acoustic models for speech recognition. Moreover, they introduced two methods to further improve their DNN model that overcome the previously best published results on their research area.

In order to deal with noise effect, some researchers use a decomposition strategy. Sáez *et al.* (2014) attenuated the presence of noise in multi-class problems by using the one-vs-one scheme. Qiu *et al.* (2016) present an ensemble method composed of Empirical Mode Decomposition (EMD) algorithm and deep learning approach. In their work, the data is firstly decomposed into several Intrinsic Mode Functions (IMFs) to be latter modeled by a Deep Belief Network (DBN) including two Restricted Boltzmann Machiness (RBMs). By using this way, the authors affirm the tendencies of each IMFs can be accurately predicted. Lastly, Yang and Cheng (2019) developed a hybrid deep learning and empirical mode decomposition model for multi-step ahead forecasting.

Finally, it is worth emphasizing most of the related studies that transform the signal in the frequency domain were focused on performing a fast convolution as discussed by Bengio and Lecun (2007) , Mathieu, Henaff and Lecun (2013) , and Vasilache *et al.* (2014) . In contrast, Rippel, Snoek and Adams (2015) have shown that, besides presenting a faster convolution, the analysis on frequency domain can also provides a good representation in CNNs modeling and training.

The related works presented in this section emphasizes our project is relevant to model real-world systems, specially in situation in which the analyzed data is affected by noise. However, we understand the manuscripts listed in this chapter show a limited view on how the scientific community has been dealing with this situation in specific problems, without performing a study on cause-effect among noise rate and classification performance. In the next chapter, we discuss our research in-depth and explain the methodology used in the present work.

Chapter 3

EXPERIMENTAL METHODOLOGY

3.1 RESEARCH CONTEXT

This research project analyzes the influence of noise on Deep Neural Network (DNN) models. Instead of just applying noise reduction filters before modeling some data, as usually performed in Machine Learning (ML), we are interested in investigating how the noise affects the final models and is propagated/smoothed through the network layers.

After understanding those points better, we intend to demonstrate the importance of our hypothesis, which states a noise filter, created to work as an embedding Convolutional Neural Network (CNN) in an Long Short-Term Memory (LSTM), can improve the modeling of real-world time series without requiring a decomposition step. Such a demonstration was conducted by the methodology presented in the following section.

3.2 METHODOLOGY

The main goal of this work is to assess our hypothesis, demonstrating its importance in modeling noisy time series. The methodology designed to reach this goal was based on four steps. Firstly, we decided to concentrate our investigation on 1-dimensional signals. In this sense, we started collaborative work with researchers from the Universidad de la Frontera (UFRO) and the Observatorio Vulcanológico de los Andes Sur (OVDAS) to model and classify seismic signals from the Llaima Volcano in Chile (see Section 4.1.2). From this collaboration, the first contribution of this thesis was published in the following conference:

- Curilem, M., Canário, J. P., Franco, L., Rios, R. A. (2018, July). Using CNN To classify spectrograms of seismic events from llaima volcano (Chile). In 2018 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

The results presented in this manuscript motivated us to go further on this topic. In this sense, we developed a deeper analysis about the usage of DNN models on this application, supporting the development of a new architecture called SeismicNet, which was published in the following journals:

- Canário, J. P., Mello, R. F., Curilem, M., Huenupan, F., Rios, R. A. (2020). In-depth comparison of deep artificial neural network architectures on seismic events classification. Journal of Volcanology and Geothermal Research, 401, 106881.
- Canário, J. P., Mello, R. F., Curilem, M., Huenupan, F., Rios, R. A. (2020). Llaima volcano dataset: In-depth comparison of deep artificial neural network architectures on seismic events classification. Data in brief, 30, 105627.

In our second step, we have created a controlled scenario in which different synthetic white noise influences were added to the seismic signals to measure the impact on the final models. Our experiments (see Section 4.2) confirmed by using statistical tests that the noise influence plays an essential role in the classification performances. As a direct consequence of this result, we have conducted the third step of our methodology, which was devoted to understanding the effect of noises in classifiers not only in the final classification task but also during the training phase.

In this step, we have used the most recently published tools from the eXplainable Artificial Intelligence (XAI) area, aiming at visually understanding the noise propagation inside our DNN architectures. The results have emphasized how signals and noise affect class prediction differently (see Section 4.3.3). By knowing this information, we were able to build up the second contribution of this research which is a new approach to reconstruct signals highlighting how every observation is seen during the classification process, e.g., noise or signal. The main advantage of this contribution is the possibility of being used as a filter bank, i.e., depending on the region where the noise is shown, bandpass filters can be used to separate stochastic from the deterministic behavior. The results of this contribution are under review in the following journal:

- Canário, J. P., Ribeiro, O., Rios, R. A. (2022). Using eXplainable AI to understand noise effects in CNN: a real case study on Llaima volcano signals. Neurocomputing (under review).

After knowing the importance of noisy influences in our classification models, we have driven our efforts to the fourth step with the final and most challenging part of this work: the designing of a new LSTM with a SeismicNet-based CNN embedded in the memory cells. This embedded CNN acts as a noise filter, suppressing the preprocessing step usually considered in ML. In summary, it is responsible for learning noisy features and delivering more stable results regardless of the noise level added to the seismic signal. The evaluation of our proposal was performed by using a straightforward experiment, illustrated in Figure 3.1, in which we have sent noisy signals to both LSTM architectures (traditional one and our proposal). Then, the obtained results are analyzed by using well-defined validation criteria widely considered in classification tasks (see Section 4.1.3).

With results obtained in this last step (see Section 4.4.1), we have achieved our main goal: to formulate a theoretical contribution to demonstrate our hypothesis, i.e., the development of a robust embedding layer in Long Short-Term Memory Networks, thus improving the classification process in the presence of noisy data. The final manuscript with the full description and results yielded from this contribution will be submitted after

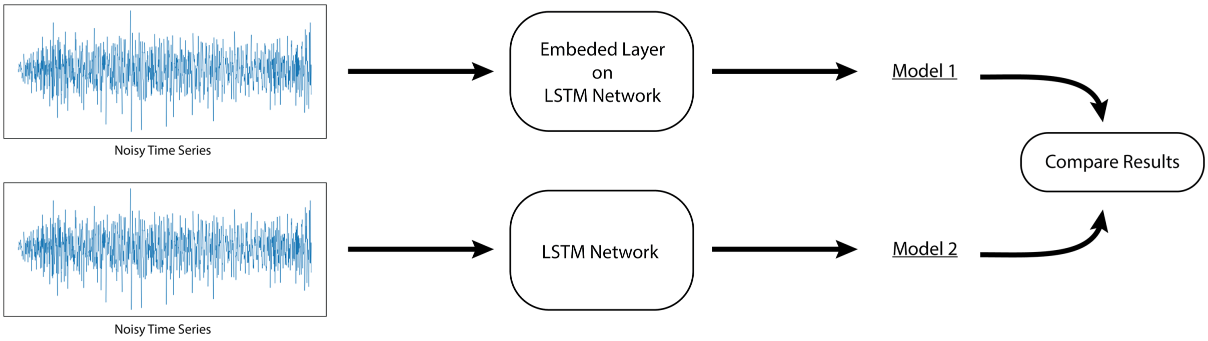


Figure 3.1 The top-most figure shows our proposal that will be created to implicitly attenuate the influence of noise on DNN. Just to exemplify, we represent a DNN with an wavelet-embedded filter layer. In the lower pipeline, we plan to use traditional DNN, e.g SeismicNet, without the filter layer to quantify their differences in relation to the proposed approach.

the thesis defense. Due to the volume of experiments presented in this work, we have provided more details about each step preceding their respective results.

3.3 SOCIAL CONTRIBUTION

Although not directly related to our main investigation topic, two important events have drastically affected the scientific community during the period spent developing this thesis. The first event was related to the coronavirus disease 2019 (COVID-19), which is a severe acute respiratory syndrome caused by the coronavirus 2 (SARS-CoV-2), deeply affecting not only the health systems but also the global economy and politics. Aiming at dealing with such a very aggressive virus, scientists have been dedicating efforts in two main directions. The first one is the development of a new vaccine specifically designed to immunize the population. The second direction is related to non-pharmacological strategies, which include, for example, social distancing, quarantine, isolation, and the adoption of alcohol-based hand sanitizers and face masks.

Among all possible non-pharmacological strategies, the adoption of face masks has been recommended by the WHO and scientists, that have been conducting researches to understand their influence to reduce the person-to-person spread through close contact. In Brazil, where the high number of confirmed cases and deaths caused by coronavirus was noticed, the public transportation, widely used by the population in general, is an important monitoring point. The relaxation of the mask adoption in such an environment can be responsible for increasing the reproduction number (R) of SARS-CoV-2. For example, before starting the pandemic outbreak, more than 1,1 million passengers used to take buses per day in Salvador – Brazil (average calculated between 2016 and 2019). In this context, the restriction of minimum distance among passengers cannot be assured, especially during rush hours. Additionally, the monitoring of people wearing face masks by human supervisors is neither efficient, due to the huge volume of passengers, nor economically possible.

Therefore, the implementation of an Artificial Intelligence (AI) system, devoted to identify whether or not every passenger is respecting the mask recommendation, is essential for, at least, two main reasons: (i) to support the development of more effective policies, once the main awareness campaigns can be dedicated to regions where the rate of the facemask-wearing adoption is getting lower; and (ii) to present lines “safer” to users where face masks have been more adopted.

Aiming at solving this problem, we have developed an AI solution based on Deep Neural Network (DNN) and Visualization to monitor all bus passengers in Salvador (Brazil) and detect whether they wear face masks. The final results were published in the following journal:

- Canário, J.P., Ferreira, M.V.,Freire, J., Carvalho, M., Rios, R. A. (2022) A face detection ensemble to monitor the adoption of face masks inside the public transportation during the COVID-19 pandemic. Multimedia Tools and Applications.

The second important event was related to the risk of deploying DNN systems to recognize faces. In summary, face detection approaches have been widely adopted in several applications to create biometric markers. The advances of DNNs, especially with the Convolutional Neural Networks (CNN), have improved the face detection performances and made its usage more usual in many real-world scenarios.

Despite the theoretical and scientific advances, scientists are concerned about the fairness in AI models, which is a current hot topic in computer vision, aiming at better understand the robustness of such AI models across important human features as, for example, age, gender, and race. Due to ethical issues, some commercial systems designed by important companies (e.g. IBM, Microsoft, and Clarifai) were discontinued due to the high error rates in specific groups as black women, possibly presenting a racist bias. Recently, in New York, scientists have asked to interrupt the usage of face recognition in some situations due to errors associated with gender, race, and ethnicity.

In our context, the problem came up during the development of an AI-based system to detect fraud in public transportation in Salvador (Brazil). By considering Salvador is the Brazilian city with the highest percentage of black people (about 80%), any error may affect a significant number of users, leading to a high number of false positives. In our scenario, due to the absence of images with appropriate labels to describe users by gender and race, we used pre-trained face detectors, published by the original authors. Thus, we created an empirical setup to assess whether the detectors have gender and race biases. Aiming at reaching this goal, we firstly created a specialist committee to label our images. Then, we analyzed the errors produced by every detector by taking into account different groups of genders and races. As discussed in the following manuscript, we have indeed found a race bias in our environment:

- Ferreira, M.V., Almeida, A., Canario, J.P., Souza, M. Nogueira, T., Rios, R. (2021). Ethics of AI: Do the Face Detection Models Act with Prejudice? Brazilian Conference on Intelligent Systems, 89-103.

Chapter 4

EXPERIMENTS

This chapter, organized in four sections, presents a set of experiments and analyses conducted to assess every step of the proposed research. Firstly, we present a study of Deep Neural Network (DNN) models to classify seismic events of Llaima Volcano in Chile, which resulted in a new DNN architecture referred to as SeismicNet.

Next, we describe experiments designed to evaluate the influence of synthetic noises on the performance of the obtained models. Such experiments motivated a deeper study on such influences, thus leading us to apply an eXplainable Artificial Intelligence (XAI) approach to visually understand the noise behavior better.

Finally, all those experiments naturally paved the way to our last contribution, which was a new Artificial Neural Network (ANN) architecture that acts as an embedding layer in Long Short-Term Memory Networks to filter noise from signals without requiring a great effort in preprocessing tasks.

4.1 SEISMICNET: STUDYING THE LLAIMA VOLCANO

4.1.1 The Llaima Dataset

The seismic signals used in our experiments were collected from the Llaima volcano, which is one of the most dangerous volcanoes in South America. Llaima is located in Chile, more specifically, in Araucania Region (S 38°41′ - W 71°44′), on the western edge of Andes. Several researchers have studied its seismicity to design automatic classifiers for its events (CURILEM *et al.*, 2016; BHATTI *et al.*, 2016; CURILEM *et al.*, 2017). Due to its location, Llaima is considered a touristic attraction surrounded by villages, whose productive activity is mainly farming and livestock. Aiming at providing some security level for the people living in the neighborhood, the state agency Observatorio Vulcanológico de los Andes Sur (OVDAS) monitors not only the Llaima but also other 42 volcanoes over the whole country. In particular, for Llaima, OVDAS performs constant surveillance with 9 stations, illustrated in Figure 4.1, that continuously gather seismic activity with a 24/7 monitoring service.

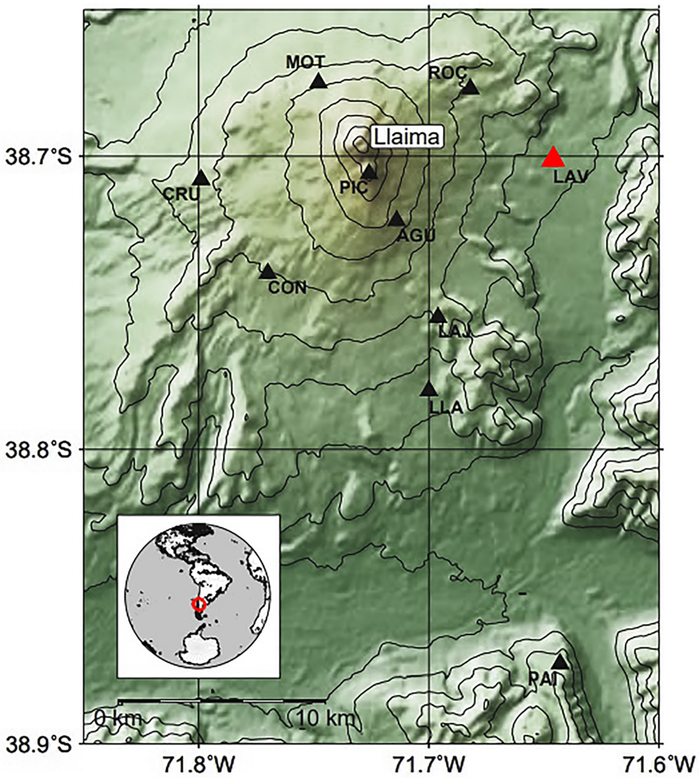


Figure 4.1 Seismic stations surrounding the Llaima Volcano. All stations are shown as black triangles but LAV, in red, that was considered in our study.

This study employs the signals collected from the LAV station, highlighted in red in Figure 4.1. Those signals were recorded in terms of the Z-vertical component from 2010 to 2016, sampled at 100 Hz and filtered using a 10th-order Butterworth bandpass filter in the range [1, 10] Hz, therefore preserving the bandwidth containing the range of interest for detecting seismic events (CURILEM *et al.*, 2016; CURILEM *et al.*, 2018a) . Then, signals were normalized by their maximum value and organized into four classes: Long Period (LP), Tremor (TR), Volcano-Tectonic (VT), and Tectonic (TC). Table 4.1 details the number of each seismic events class. Moreover, it is important to emphasize that all this process and the resultant signals were individually reviewed, segmented, and labeled by experts from OVDAS using criteria defined in Lahr *et al.* (1994).

Table 4.1 Number of events by class.

LP	TR	VT	TC	Total
1310	490	304	1488	**3592**

The first event class presented in this table is labeled as LP events are related to the transit of magmatic and hydrothermal fluids inside the volcanic conduits (CHOUET, 1996). The spectral pattern of LP is mainly bounded in the range [0.5, 5] Hz.

TR is a class of events that generally present a longer duration than LP. Also, the source of tremor is understood as a sustained pressure disturbance over magmatic and hydrothermal fluids, which may be continuous or caused by a sequence of transient signals similar to those generated by LP. Their broadband spectrum is typically in [0.5, 3.0] Hz, being slowly attenuated at the end of the event.

The VT which is associated with rock fracturing inside the volcanic building. VT events present a frequency pattern with a broadband spectrum that may reach 10 Hz.

TC events are not related to volcanic activities but they are one of the most common results of the dynamics of geological faults (CHOUET, 1996). Such events may be local, regional, or even distant according to the epicenter location. A TC event detected by a station located in a region far away from the epicenter has lower frequencies than closer ones. Depending on the source proximity, TC could be confused with LP or VT events. However, TC events generally carry more energy (signal amplitude). To illustrate the differences among such events, Figure 4.2 shows one example of each aforementioned normalized seismic class.

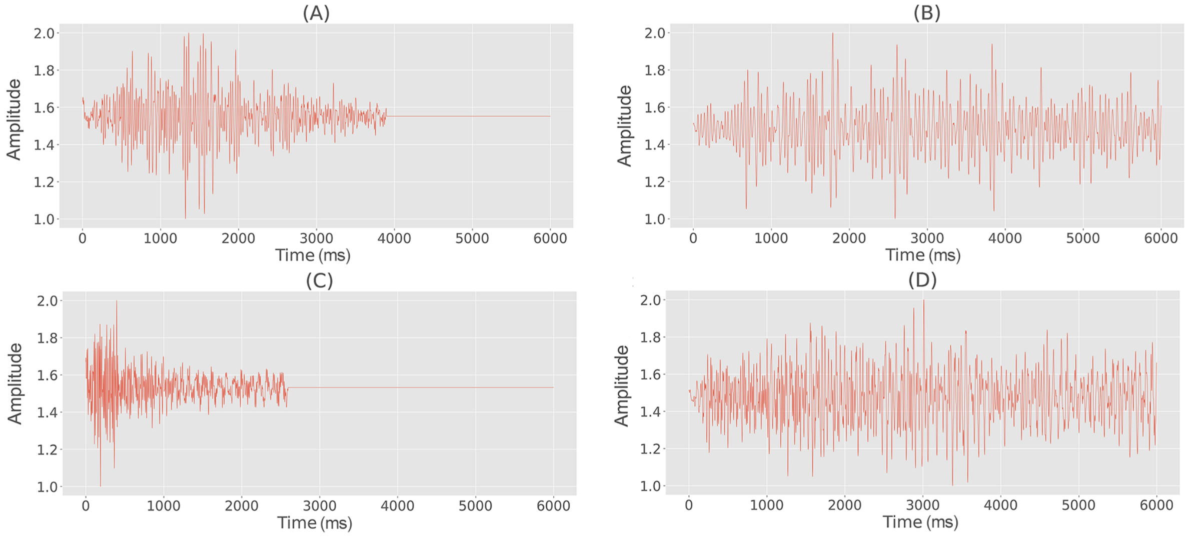


Figure 4.2 Seismic event classes: (A) LP; (B) TR; (C) VT; and (D) TC.

4.1.2 Seismic Event Processing

4.1.2.1 Seismic Spectrograms

For applying STFT to seismic signals, our experiments used experts' specifications with sliding windows equal to 1 second with 95% of overlapping. The signals were all set to 6000 samples (60 seconds) filling with zeros when necessary. Although the signals were filtered from 1 to 10 Hz, the frequency interval considered to calculate the spectrograms was from 1 to 20 Hz, to show a better visualization of their shapes (CURILEM *et al.*, 2018a; CURILEM *et al.*, 2018b; CANÁRIO *et al.*, 2020). With these specifications, all the spectrograms had the same dimensions in time and frequency. The frequency components of the spectrograms were smoothed by using a moving average filter to obtain their envelopes. The moving average was calculated by Equation 4.1, in which Sf is the number of the averaged samples (Smooth factor).

$$SmoothedSP[x(t)](n,k) = \frac{1}{Sf}\sum_{j=0}^{Sf-1} SP(n,k+j), \tag{4.1}$$

Different values for Sf were tested and the best results were obtained with 150 samples. As is illustrated by Figure 4.3, the LP spectrogram has a non-impulsive beginning with a slow decay. TR is characterized by very narrow, regular, and long shapes. The VT is very characteristic, presenting an impulsive at the beginning and exponential decay. Similar to VT, the spectral content of TC is characterized by impulsive beginning and exponential decay.

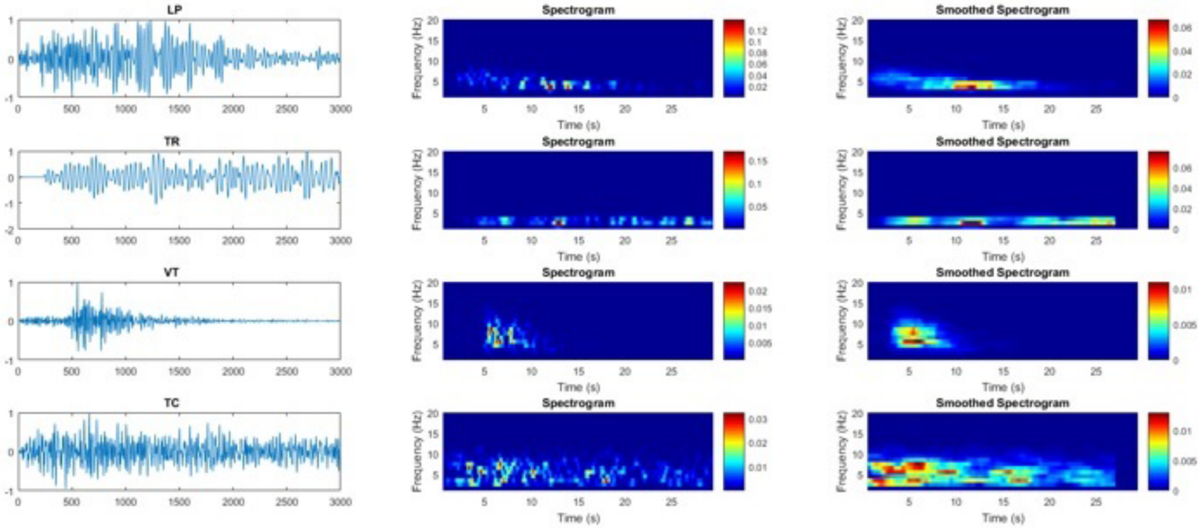


Figure 4.3 Normalized seismic events and their normal and smoothed spectrograms.

Finally, all the spectrograms were transformed into RGB images of 20 x 20 pixels and used as input of developed DNNs. Figure 4.4 shows the shape of the spectrograms for the different classes.

4.1.3 Performance Measures

The obtained results were evaluated by using measures traditionally computed to assess supervised learning tasks. Firstly, the generalization capability of our predictive models was studied by sampling the original dataset by using a 10-fold cross-validation strategy. Then, the results were organized into contingency matrices containing the number of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). Based on such matrices, we calculated four measures: (i) Accuracy (Equation 4.2); (ii) Error (Equation 4.3); (iii) Specificity (Equation 4.4); and (iv) F1-score (Equation 4.5).

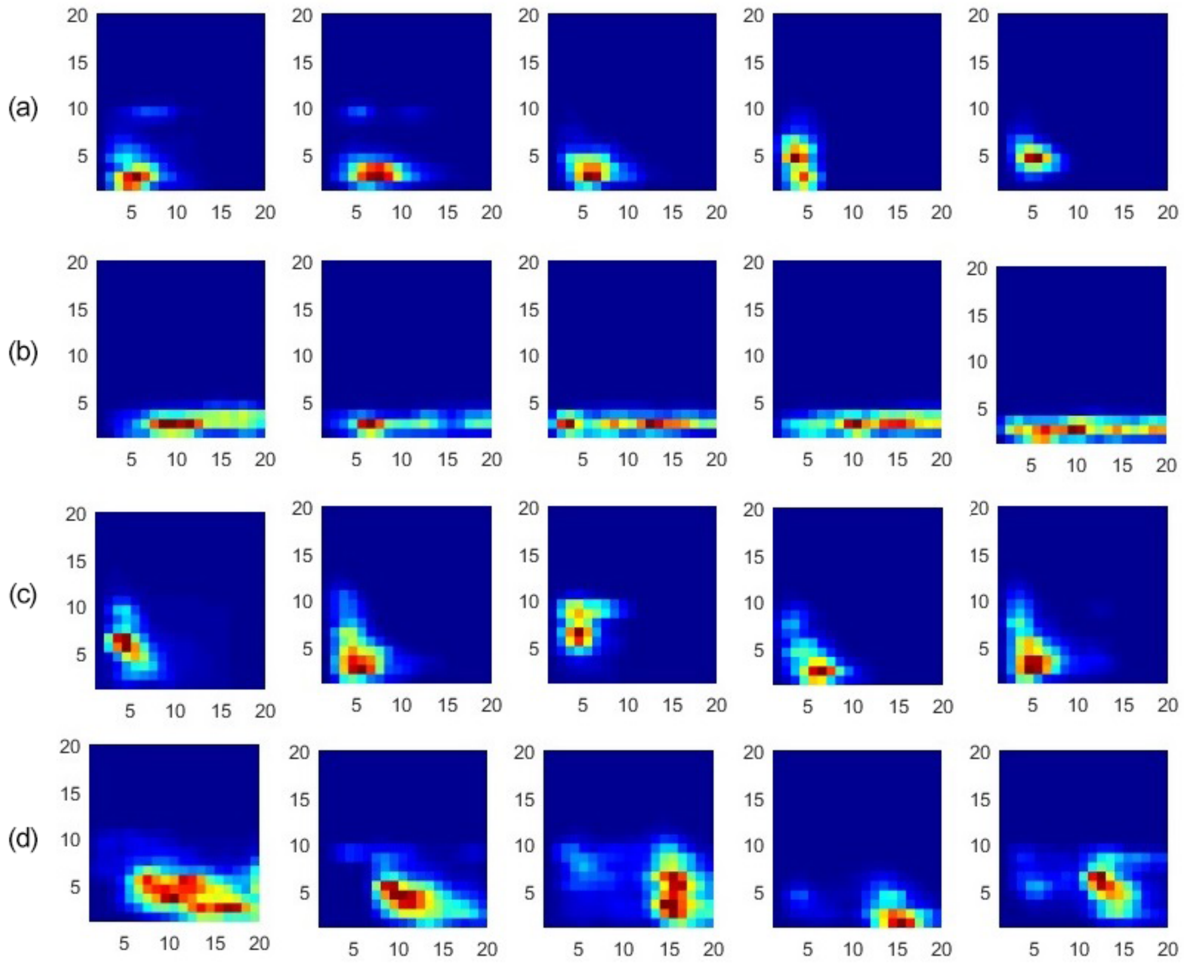


Figure 4.4 Smoothed spectrograms of the events: each line presents five examples of the smoothed spectrograms for the (a) LP, (b) TR, (c) VT and (d) TC classes.

$$\texttt{Accuracy} = \frac{(TP + TN)}{n} \tag{4.2}$$

$$\texttt{Error} = \frac{(FP + FN)}{n} \tag{4.3}$$

$$\texttt{Specificity} = \frac{TN}{(TN + FP)} \tag{4.4}$$

$$\texttt{F1-score} = \frac{2 \times (\texttt{Recall} \times \texttt{Precision})}{(\texttt{Recall} + \texttt{Precision})} \tag{4.5}$$

In such equations, n represents the total number of classified signals, *Recall* corresponds to the true positive rate (Equation 4.6), and *Precision* takes into account the number of seismic signals correctly classified as a specific event and signals wrongly classified as the same event (Equation 4.7).

$$\texttt{Recall} = \frac{TP}{(TP + FN)} \tag{4.6}$$

$$\texttt{Precision} = \frac{TP}{(TP + FP)} \tag{4.7}$$

In addition to these indices, the Kappa coefficient (WITTEN *et al.*, 2016) was also used to measure the general agreement between our classification system and experts, emphasizing the results were not obtained by chance. This coefficient is based on the difference between "observed" agreement (P_o) and agreement that would be expected to occur by chance also referred to as "expected" one (P_e), as shown in Equation 4.8.

$$P_o = \sum_{i=1}^{C} p_{ii}, \quad P_e = \sum_{i=1}^{C} p_{i.}p_{.i} \tag{4.8}$$

This equation considers p_{ii} as the joint proportion of agreement (diagonal), $p_{.i}$ and $p_{i.}$ corresponds to the sum of joint proportions of our classifier (column) and experts (rows) for every class, while C is the number of classes. Finally, the Kappa (K) coefficient is obtained using Equation 4.9, in which the better the agreement is, the closer to 1 such coefficient is.

$$K = \frac{P_o - P_e}{1 - P_e} \tag{4.9}$$

4.1.4 Results

It is important to recall that this work discusses a series of contributions to the state-of-the-art in seismic signal processing. Our contribution, referred to as SeismicNet, is based on an extension of SoundNet, which is a 1D Convolutional Neural Network (CNN) configured to classify raw signals. The main advantage of using this CNN is to perform a one-dimensional analysis, directly applied to the seismic signals, instead of transforming them into a set of features before applying machine learning algorithms. Secondly, we assessed the advantage of employing wavelets to improve the training stage of SeismicNet. In this sense, we applied a discrete wavelet transform on signals and just considered the first level, thus downsampling them by a factor of 2 and, as a consequence, reducing the spatial and temporal complexities. Thirdly, we extended our previous analyses using Continuous Wavelet Transform (CWT), which provided the bests results by considering the numbers of scales and octaves equal to 2 and 5, respectively. It is worth emphasizing that we performed a deep analysis on the Llaima volcano signals training more than 50 different DNN architectures. In this section, we only present the final architectures that provided the best results, assuring the reproducibility of our achievements.

4.1.4.1 Our First Attempt: Spectrograms plus Convolutional Neural Network

At the time of writing this work, the state-of-the-art results obtained after analyzing the Llaima volcano were previously published by our research group in (CURILEM *et al.*, 2018a) and (CURILEM *et al.*, 2018b) . In Table 4.2, we summarized the final results published in (CURILEM *et al.*, 2018b) , showing the mean Accuracy and F1-score, along with their standard deviation, by analyzing 4 signal classes: LP, TR, VT, and TC. The classification was performed in two steps. Firstly, the signals were transformed into a set of spectrograms using Fourier Transform. Then, we implemented a 2D-CNN to classify the spectrograms according to the 4 classes.

Table 4.2 The state-of-the-art accuracies and f1-scores. We used a 2D-CNN + spectrograms (CURILEM *et al.*, 2018b) according to the 10-fold cross-validation strategy.

Spectrograms + 2D-CNN		
	Accuracy	**F1-score**
Mean:	**97.08%**	**95.84%**
Std:	**0.78%**	**1.27%**

Table 4.3 details all performance indices obtained during this classification task, including the Kappa coefficient. It is important to highlight the design of our experiments were conducted considering a 10-fold cross-validation strategy to reduce the probability of considering overfitted results.

Table 4.3 Spectrograms plus convolutional neural network performance indices (in percentages).

Performance Indices					
	LP	**TR**	**VT**	**TC**	**Mean**
Recall	97.86%	96.53%	91.78%	97.65%	95.95%
Specificity	98.86%	99.52%	99.24%	98.15%	98.94%
Accuracy	98.50%	99.11%	98.61%	97.94%	98.54%
Error	1.50%	0.89%	1.39%	2.06%	1.46%
Kappa: 0.956					

The results have shown this CNN structure was capable of discriminating the four classes from the spectrograms achieving the best performance so far. However, it is worth mentioning results are obtained using a laborious preprocessing phase performed to transform all signals into spectrogram images, also requiring a high training time.

Architecture using spectrograms plus a 2D CNN

Considering that CNNs were not commonly used on spectrograms from seismic activities, we started analyzing the signals by using a standard architecture proposed by

LeCun *et al.* (1998). In summary, we performed the following analyses: i) Changing the number of fully-connected layers or convolutional layers by adding or removing them; ii) Adding a dropout layer following the convolutional layers; iii) Using dropout layers following the fully-connected layers to avoid overfitting; iv) Changing the optimization algorithm from stochastic gradient descent to Adam (KINGMA; BA, 2014); v) Changing the hyperparameters of some layers, such as the number of filters on convolutional layers, dropout rate, number of neurons on fully-connected layers. The considered architecture, illustrated in Figure 4.5, was selected based on the best performance measures. Next, every layer is presented in detail:

1. An input layer denoted by $I(h, w, c)$, where h is the height, w is the width and c is the number of channels provided by the input data;

2. Two convolutional layers $C(K, F, S, P)$, where K is the number of filters, F is their spatial extent, S is the stride and P is the amount of zero padding;

3. Two max-pooling layers, denoted as $P(F, S)$, in which F is the size of the square pooling regions and S is the stride;

4. Two fully connected layers $F(u)$, where u is the number of units or neurons in the layer;

5. At the output of each convolutional layer and the first fully-connected layer, the ReLU activation function $f(x) = max(0, x)$ was applied, where x is the output of each neuron from the previous layer;

6. Two dropout layers $D(dr)$, in which dr is the dropout rate;

7. Finally, on top of our CNN a softmax activation function was applied.

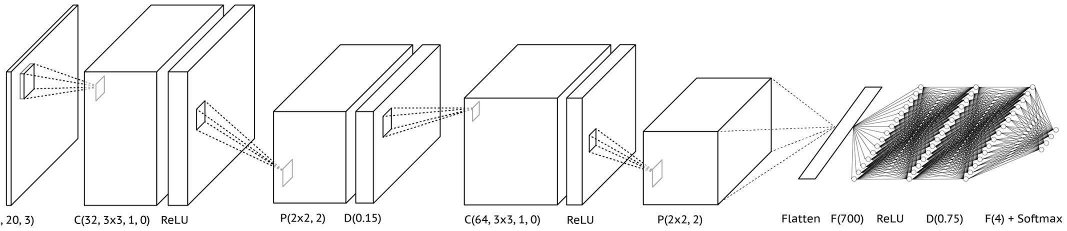


Figure 4.5 2D-CNN summary: conv. 2D, ReLU, max pool., dropout, conv. 2D, ReLU, max pool., fully-connected, ReLU, dropout, fully-connected and output layer with softmax activation function.

4.1.4.2 Our Second Attempt: CWT plus Convolutional Neural Network

The following experiments were conducted using a 2D-CNN architecture, whose input seismic signals were pre-processed using the Morlet complex wavelet transform (here referred simply to as CWT), as discussed in Section 2.1.2.2, into a set of new images. Table 4.4 summarizes accuracies and F1-scores along all 10 folds of the cross-validation strategy. As one may notice, this CNN architecture has confirmed very good overall results, to mention, greater than 96% and 94%. Moreover, the overall standard deviation was lower than 0.9 and 1.3%, respectively, corroborating the network stability.

Table 4.4 Our Second Attempt: CWT + 2D CNN performance indices and confusion matrix (in percentages).

CWT + 2D Convolutional Neural Network		
	Accuracy	**F1-score**
Mean:	**96.21%**	**94.66%**
Std:	**0.84%**	**1.25%**
Fold 1:	95.83%	93.78%
Fold 2:	95.83%	94.62%
Fold 3:	97.78%	96.66%
Fold 4:	96.67%	95.45%
Fold 5:	96.38%	94.21%
Fold 6:	95.54%	93.53%
Fold 7:	96.66%	95.98%
Fold 8:	97.21%	96.13%
Fold 9:	95.25%	93.27%
Fold 10:	94.97%	92.98%

In Table 4.5, we listed the performance indices as well as the confusion matrices and Kappa values. By individually analyzing such matrices, one may notice that the overall classification performance using CWT was very close to one obtained in the state-of-the-art (Section 4.1.4.1). This is also confirmed by the Kappa coefficient equals to 0.950. Observe the use of CWT has kept the good results along with all classes.

CWT plus 2D CNN Architecture

The 2D-CNN architecture employed in this set of experiments is a simplification of the CNN presented in our first attempt (CURILEM *et al.*, 2018b) , in which a sequence of 2D-convolution layers was proposed to classify 2D-spectrogram images of seismic events. In this paper, we only used one convolutional layer along with one max pooling and one fully-connected layer to process and classify the continuous wavelet transformation applied to

Table 4.5 Our Second Attempt: CWT + 2D CNN performance indices and confusion matrix (in percentages).

Performance Indices					
	LP	**TR**	**VT**	**TC**	**Mean**
Recall	97.33%	95.71%	88.82%	96.91%	94.69%
Specificity	98.42%	99.19%	99.00%	98.00%	98.65%
Accuracy	98.02%	98.72%	98.13%	97.55%	98.11%
Error	1.98%	1.28%	1.87%	2.45%	1.89%

Confusion Matrix					
		Predicted Labels			
		LP	**TR**	**VT**	**TC**
True Labels	**LP**	**1275**	3	18	14
	TR	4	**469**	0	17
	VT	22	1	**270**	11
	TC	10	21	15	**1442**
Kappa: 0.950					

seismic signals. Finally, as the aforementioned setups, we followed the same strategy to test and estimate the best configurations for this CNN architecture. The chosen architecture, illustrated in Figure 4.6, was also selected based on the best performance measures. Next, every layer is presented in detail:

1. An input layer $I(w, h, c)$, in which w is its width, h is its height and c is the number of channels provided the input data;
2. One convolutional layer $C(K, F, S, P)$, having K is the number of filters, F as the spatial extent, S is the stride, and P is the zero-padding length;
3. One max-pooling layer, denoted as $P(F, S)$, in which F is the size of the square pooling region, and S is its stride;
4. One fully-connected layer expressed as $F(u)$, having u as the number of units or neurons at such a layer;
5. After the output of the convolutional and the fully-connected layers, we added two batch normalization layers to finally apply the CReLU activation function;
6. After the output of feature extraction layers, the flatten layer is responsible for collapsing all data into a single-dimensional vector;
7. Two dropout layers $D(dr)$ were added, in which dr is the dropout rate;
8. Finally, a softmax activation function was applied.

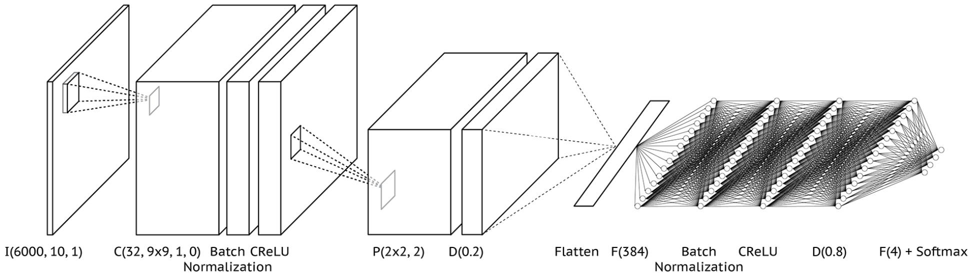


Figure 4.6 CNN summary: input, conv. 2D, batch normalization, CReLU, max pool. 2D, dropout, flatten, fully-connected, batch normalization, CReLU, dropout and output layer with softmax activation function.

4.1.4.3 Our Third Attempt: CWT plus Long Short-Term Memory

The next set of experiments was conducted after replacing the 2D-CNN by a Long Short-Term Memory (LSTM) neural network on the CWT data obtained after pre-processing all seismic signals, as performed in the previous section. Table 4.6 summarizes all accuracies and F1-scores for all 10 folds with the cross-validation strategy, which allowed to confirm that the LSTM in conjunction with the CWT was capable of producing a great performance as well, although the standard deviation was slightly higher.

Table 4.6 Our Second Attempt: LSTM accuracies and F1-scores (in percentages).

Long Short-Term Memory		
	Accuracy	**F1-Score**
Mean:	**95.82%**	**94.08%**
Std:	**1.28%**	**1.94%**
Fold 1:	94.44%	90.82%
Fold 2:	94.72%	93.67%
Fold 3:	98.06%	97.28%
Fold 4:	95.56%	93.80%
Fold 5:	97.21%	96.31%
Fold 6:	96.38%	94.57%
Fold 7:	94.15%	91.59%
Fold 8:	94.71%	92.89%
Fold 9:	97.21%	95.94%
Fold 10:	95.81%	93.90%

Table 4.7 complements our analyses by presenting the performance indices, confusion matrix, and Kappa value. According to the individual results presented in such a table as well as the overall performance of the mean accuracies and f1-scores, we notice the

LSTM architecture also achieved a great performance, something confirmed by the Kappa coefficient 0.945.

Table 4.7 Our Second Attempt: Long Short-Term Memory performance indices and confusion matrix (in percentages).

Performance Indices					
	LP	**TR**	**VT**	**TC**	**Mean**
Recall	97.71%	93.88%	87.83%	96.44%	93.96%
Specificity	97.98%	99.23%	98.94%	97.86%	98.50%
Accuracy	97.88%	98.50%	98.00%	97.27%	97.91%
Error	2.12%	1.50%	2.00%	2.73%	2.09%

Confusion Matrix					
		Predicted Labels			
		LP	**TR**	**VT**	**TC**
True Labels	**LP**	**1280**	3	20	7
	TR	3	**460**	0	27
	VT	26	0	**267**	11
	TC	17	21	15	**1435**

Kappa: 0.945

CWT plus LSTM Architecture

As performed in the previous experiment, the design of each LSTM layer was individually assessed until obtaining the following best architecture (Figure 4.7):

1. An input layer $I(w, h)$, having w as its width and h as its height;

2. One LSTM layer $LSTM(Un, RS)$, in which Un is the output space dimensionality and RS is the return sequence flag to ensure that LSTM cells return all of outputs from the unrolled LSTM cell through time. If this argument is left out, the LSTM cell will simply provide the output from the previous time step;

3. One fully-connected layer expressed as $F(u)$, having u as the number of units or neurons at such a layer;

4. After the output of the LSTM and the fully-connected layers, we added a batch normalization layer followed by a CReLU activation function layer;

5. After the output of feature extraction layers, a flatten layer was added to collapse data into a single-dimensional vector of features;

6. Two dropout layers $D(dr)$ were added, in which dr is the dropout rate;
7. Finally, a softmax activation function was applied on top of our neural network.

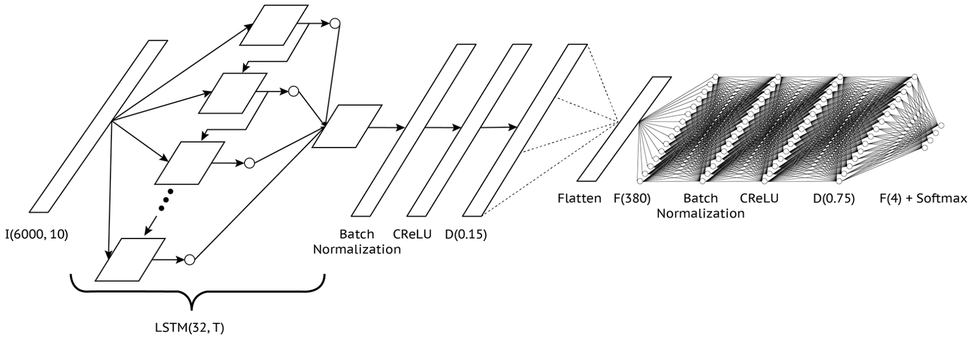


Figure 4.7 LSTM summary: input, LSTM, batch normalization, CReLU, dropout, flatten, fully-connected, batch normalization, CReLU, dropout and output layer with softmax activation function.

4.1.4.4 Our Fourth Attempt: CWT plus Multilayer Perceptron

After analyzing the previous results, we decided to compare the DNN architectures assessed so far to a baseline composed of the same CWT input data followed by a Multilayer Perceptron (MLP) architecture. Neurons and layers were empirically added using the same validation criterion adopted to build up the other neural networks. The results obtained with the best MLP architecture are summarized in Table 4.8, which confirm great performance indices as well, allowing to conclude that the DNN architectures evaluated so far could not significantly outperform this classical and simple approach.

Complementary, Table 4.9 shows the performance indices for every class as well as the confusion matrix and the Kappa value. As one may notice, an MLP containing a reduced number of layers and hyper-parameters achieved similar results when compared to other architectures.

CWT plus MLP Architecture

The MLP setup followed the simple structure briefly presented in Section 2.2.2. As aforementioned, the best configuration was built up taking into account the following step-by-step approach (Figure 4.8):

- Fully-connected layers were added or removed, as well as having their number of neurons changed along with assessment iterations;
- Hyper-parameters at each layer were adapted, such as the dropout rate and the activation functions;

Table 4.8 Our Third Attempt: MLP accuracies and F1-scores in percentages.

Multi-Layer Perceptron		
	Accuracy	**F1-score**
Mean:	**96.74%**	**95.47%**
Std:	**0.76%**	**1.18%**
Fold 1:	96.11%	94.09%
Fold 2:	96.94%	96.33%
Fold 3:	97.50%	96.19%
Fold 4:	96.39%	95.56%
Fold 5:	97.49%	96.39%
Fold 6:	95.82%	94.04%
Fold 7:	96.38%	94.73%
Fold 8:	97.49%	96.64%
Fold 9:	97.77%	97.07%
Fold 10:	95.53%	93.62%

- Batch normalization layers were inserted and removed along the evaluation process;

- A dropout layer was added after the fully-connected layer to avoid overfitting;

Each MLP architecture found with those steps was individually evaluated and the best MLP architecture obtained was composed of:

1. An input layer $I(w, h, c)$, having w as its width, h as its height, and c as the number of channels used from input data;

2. A flatten layer to collapse all features into a single-dimensional vector to represent the input data;

3. One fully-connected layer expressed as $F(u)$, in which u is the number of units or neurons;

4. One batch normalization layer to increase the stability of this MLP architecture;

5. At the output of batch normalization layer, we applied the Concatenated ReLU (CReLU) (SHANG *et al.*, 2016) activation function;

6. One dropout layer $D(dr)$, in which dr is the dropout rate;

7. Finally, a softmax activation function was applied on top of our MLP to deal with the multi-class classification problem.

Table 4.9 Our Third Attempt: Multi-Layer Perceptron performance indices and confusion matrix (in percentages).

Performance Indices

	LP	**TR**	**VT**	**TC**	**Mean**
Recall	97.56%	95.71%	90.79%	97.58%	95.41%
Specificity	98.38%	99.58%	99.09%	98.24%	98.82%
Accuracy	98.08%	99.05%	98.39%	97.97%	98.37%
Error	1.92%	0.95%	1.61%	2.03%	1.63%

Confusion Matrix

		Predicted Labels			
		LP	**TR**	**VT**	**TC**
True Labels	**LP**	**1278**	0	19	13
	TR	4	**469**	0	17
	VT	21	0	**276**	7
	TC	12	13	11	**1452**

Kappa: 0.957

4.1.4.5 Our Proposed DNN Architecture: SeismicNet

This section shows the results obtained with our proposed DNN architecture referred to as SeismicNet. Table 4.10 summarizes the model accuracies and F1-scores after being directly applied on the raw seismic signals and their wavelet samples along with the 10-fold cross-validation strategy. As one may notice, our CNN architecture has also shown great results, presenting a mean accuracy greater than 0.95 (95%) and an F1-score greater than 0.91 (91%). Those results are also very close to the ones obtained by the state-of-the-art. By analyzing our network applied on the first wavelet level, we noticed an improvement in the accuracy and the F1-score values, even using a considerably smaller number of analyzed observations.

Table 4.11 shows the corresponding confusion matrices and the Kappa values for every model. By analyzing the individual values, one may notice the overall classification was improved after using the wavelet transform. This observation is also confirmed by Kappa coefficients which were greater than 0.93, highlighting the outstanding performance of both classifiers.

In Table 4.12, we list all results of our classifiers for every class, instead of presenting their overall performance. Similarly, the use of wavelets improved results for all classes, but VT whose rates were slightly lower.

In contrast with previous work and the first attempt, in which intensive feature extraction steps or signals to image transformations were adopted during the preprocessing stage, our architecture directly uses seismic signals. By downsampling the signal obser-

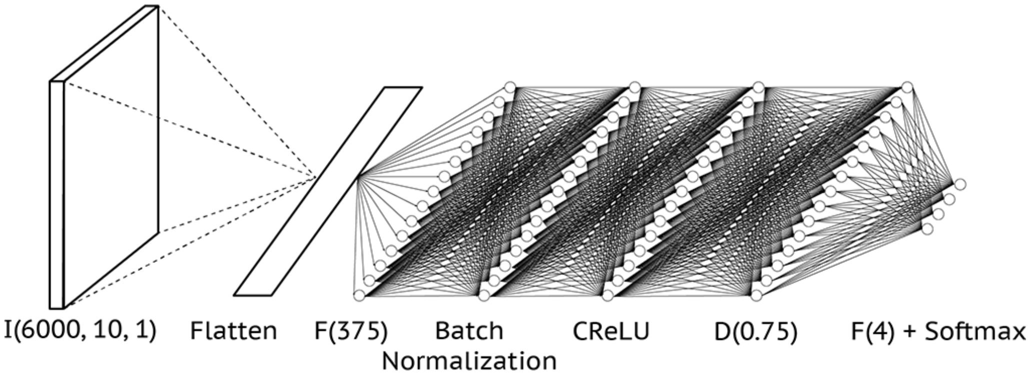


Figure 4.8 MLP summary: input layer, flatten, fully-connected, batch normalization, CReLU, dropout and output layer with softmax activation function.

Table 4.10 Our Proposed DNN Architecture: SeismicNet accuracies and F1-scores in percentages.

	Raw Signal		**Wavelets (V1)**	
	Accuracy	**F1-Score**	**Accuracy**	**F1-Score**
Mean:	**95.05%**	**91.75%**	**96.07%**	**93.02%**
Std:	**1.61%**	**2.89%**	**1.54%**	**3.97%**
Fold 1:	95.79%	93.01%	98.13%	97.16%
Fold 2:	92.06%	85.87%	97.66%	97.60%
Fold 3:	93.46%	86.93%	92.99%	85.15%
Fold 4:	94.39%	91.90%	94.86%	92.41%
Fold 5:	97.66%	95.99%	97.66%	95.99%
Fold 6:	93.93%	91.95%	96.26%	94.62%
Fold 7:	95.79%	93.11%	95.79%	90.42%
Fold 8:	95.33%	92.74%	95.33%	92.05%
Fold 9:	94.86%	92.06%	97.20%	96.74%
Fold 10:	97.18%	93.40%	94.84%	88.10%

vations using the first wavelet level ($V1$) of a Daubechies discrete wavelet transform (here simply referred to as DWT), we not only improved the classification measures but also reduced the spatial and temporal complexities. By analyzing Figure 2.1, we notice this first level keeps the general signal behavior and reduces the observations to $N/2$, such that N is the total of observations from the raw seismic signal. As a new level (e.g. $V2$ or $V4$) is analyzed, the representation of the signal behavior reduces in proportion to the number of observations. As a consequence, the accuracy of our learning model is directly affected as expected, but still provides important results, e.g. the mean accuracy for $V1$, $V2$, $V3$, and $V4$ were equal to 98.10%, 97.87%, 97.31%, and 95.34%, respectively.

Table 4.11 Our Proposed DNN Architecture: SeismicNet confusion matrices.

		Raw Signal				**Wavelets (V1)**				
		Predicted					**Predicted**			
		LP	**TR**	**VT**	**TC**		**LP**	**TR**	**VT**	**TC**
True	**LP**	**603**	5	5	16	**LP**	**608**	4	11	6
	TR	0	**125**	0	13	**TR**	1	**134**	0	3
	VT	22	0	**125**	3	**VT**	22	0	**127**	1
	TC	19	13	10	**1180**	**TC**	18	3	15	**1186**
		Kappa: 0.935				**Kappa: 0.948**				

Table 4.12 SeismicNet performance indices (in percentages).

Raw Signal					
	LP	**TR**	**VT**	**TC**	**Mean**
Recall	95.87%	90.58%	83.33%	96.56%	91.59%
Specificity	97.28%	99.10%	99.25%	96.51%	98.04%
Accuracy	96.87%	98.55%	98.13%	96.54%	97.52%
Error	3.13%	1.45%	1.87%	3.46%	2.48%
Wavelets (V1)					
	LP	**TR**	**VT**	**TC**	**Mean**
Recall	96.66%	97.10%	84.67%	97.05%	93.87%
Specificity	97.28%	99.65%	98.69%	98.91%	98.63%
Accuracy	97.10%	99.49%	97.71%	97.85%	98.04%
Error	2.90%	0.51%	2.29%	2.15%	1.96%

SeismicNet Architecture

The proposal of SeismicNet is based on a deep analysis of the DNN architecture proposed by Aytar, Vondrick and Torralba (2016) (AYTAR; VONDRICK; TORRALBA, 2016). Such architecture, referred to as SoundNet, directly extracts features from raw audio waveforms and classifies them using a next stage with a Support Vector Machine. The authors proposed the use of a sequence of 1D convolutions followed by nonlinear operators to process sound signals. They also tested two deep CNN architectures to represent such signals: (i) the first CNN is composed of 8 convolutional layers, followed by 3 pooling layers; and (ii) the second had 5 convolutional layers with 3 next pooling layers. After several experiments considering those two architectures, we selected the 5-layer SoundNet to compose the feature extraction process of our SeismicNet architecture, then we added other layers according to the following process:

- Adding or removing extra fully-connected layers;

- Testing different configurations of layer hyper-parameters, such as the dropout rate and the number of neurons at each dense layer;
- Adding dropout layers after convolutional layers and dense layers to avoid overfitting;

Every adaptation was individually evaluated and the final SeismicNet architecture (illustrated in Figure 4.9) is composed of:

1. An input layer $I(ln)$, in which ln is the input data length;
2. Five convolutional layers $C(K, F, S, P)$, having K is the number of filters, F as the spatial extent, S is the stride, and P is the zero-padding length;
3. Three max-pooling layers denoted as $P(F, S)$, in which F is the size of the square pooling region, and S is its stride;
4. One dense layer expressed as $F(u)$, having u as the number of units or neurons at such a layer;
5. In the output of every convolutional and dense layer, we applied the ReLU activation function $f(x) = max(0, x)$, so that x is the output from neuron layers;
6. Three dropout layers $D(dr)$, in which dr is the dropout rate;
7. Finally, a softmax activation function was applied on the CNN output.

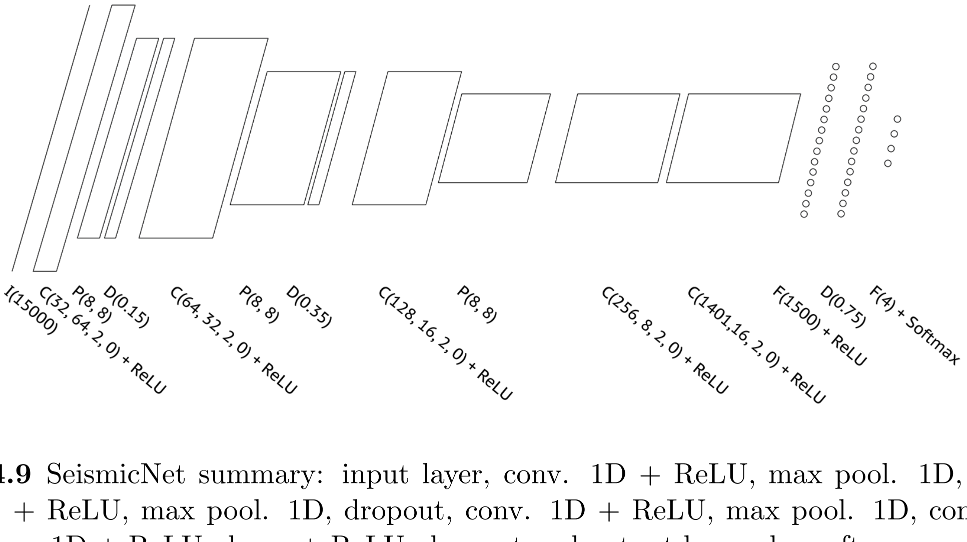


Figure 4.9 SeismicNet summary: input layer, conv. 1D + ReLU, max pool. 1D, dropout, conv. 1D + ReLU, max pool. 1D, dropout, conv. 1D + ReLU, max pool. 1D, conv. 1D + ReLU, conv. 1D + ReLU, dense + ReLU, dropout and output layer plus softmax as activation function.

4.1.5 Studying The Llaima Volcano: Results Summary

In this experiment, we performed a broad set of analyses to explore the DNN performance while discriminating seismic activities, discussing a series of contributions to state-of-the-art seismic signal processing. We evaluate such architectures using traditional measures to assess supervised learning tasks. Firstly, we tested our predictive models' generalization capability by sampling the original dataset using the k-fold cross-validation strategy.

Through the obtained results of the k-fold cross-validation, we could analyze that each architecture presents, respectively, over 95% and 90% mean performance of the computed measures: accuracy or f1-score. Figures 4.10 and 4.11 shown mean and standard deviations of cited measures for each model.

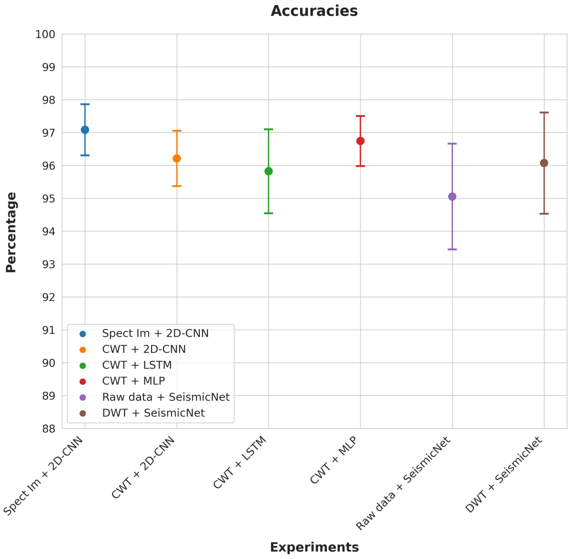


Figure 4.10 Model architectures accuracies (k-fold cross-validation mean and standard deviation).

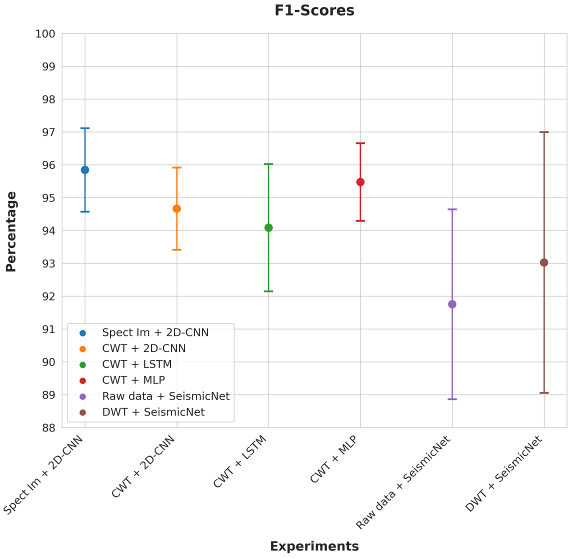


Figure 4.11 Model architectures f1-scores (k-fold cross-validation mean and standard deviation).

Similarly, performance measures computed by class presented equivalent results for all architectures: accuracy (Figure 4.12), recall (Figure 4.13), specificity (Figure 4.14, and error (Figure 4.15).

Finally, aside from those indices, we compute the Kappa coefficient to measure the general agreement between our classification system and experts. Figure 4.16 illustrates the coefficient overview, in which we can analyze that all models continue to deliver equivalent results. Moreover, as observed, each model reached over 0.93 on the Kappa coefficient, indicating almost a perfect agreement.

4.2 NOISE INFLUENCE EXPERIMENT

This section describes an initial experiment designed to analyze the white noise influence on the seismic signals. Results, assessed by a set of statistical tests, suggest our hypothesis is relevant, i.e. the presence of the noise in seismic signals affects the classification

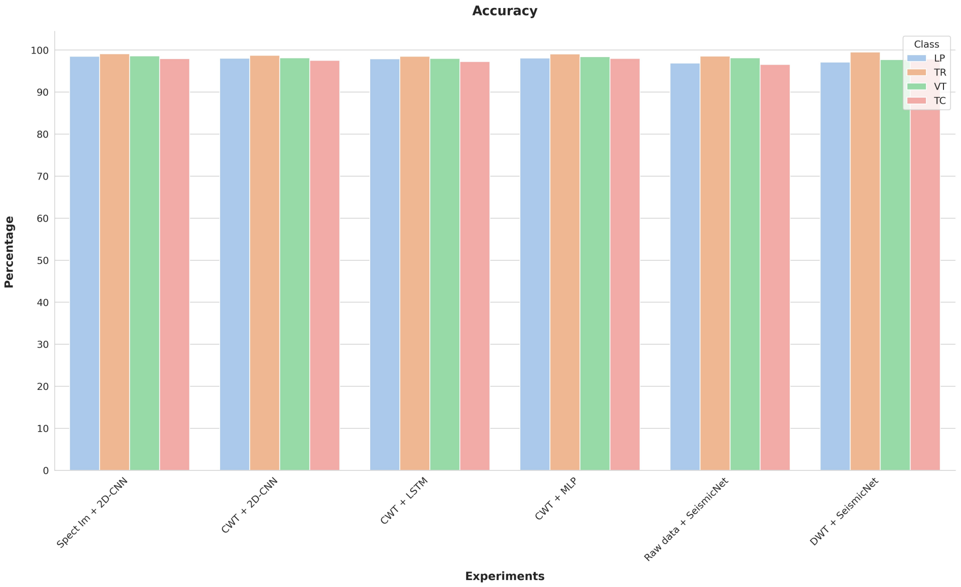


Figure 4.12 Architectures' accuracy by class.

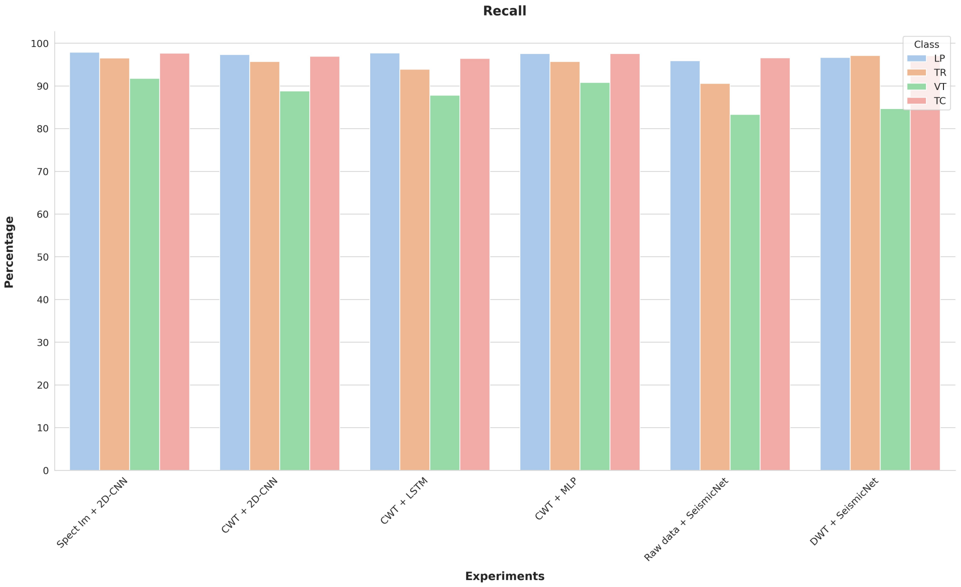


Figure 4.13 Architectures' recall by class.

performance of our DNN models, as discussed next.

4.2.1 Noise Influence Results

As well as in our previous experiments, we used the 10-fold cross-validation strategy to compute the accuracy of our SeismicNet. Then, to verify whether or not our results were affected by noise, we considered the following statistical tests: i) Shapiro-Wilk; ii) Bartlett's; iii) Wilcoxon; and iv) Student's t-distribution (t-Student).

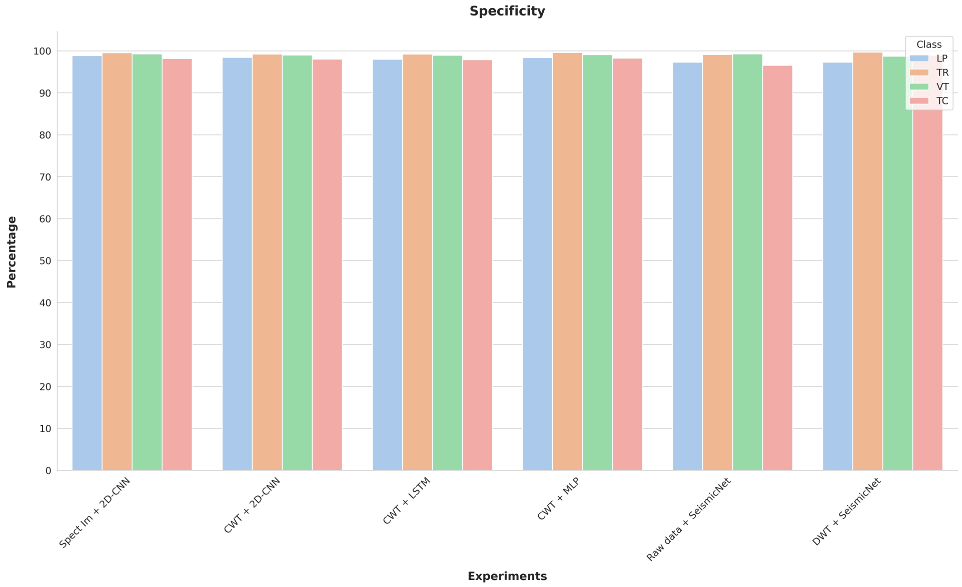


Figure 4.14 Architectures' specificity by class.

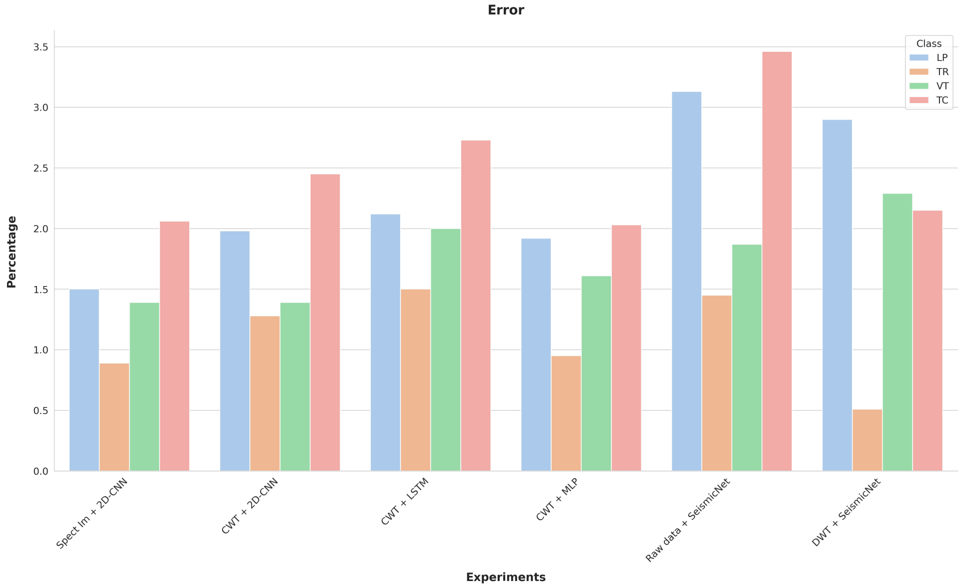


Figure 4.15 Architectures' error by class.

In summary, we added different levels of random noise on the seismic signals (Equation 4.10) by changing the standard deviation of the normal distribution function (Equation 4.11), just to simulate different Signal-to-Noise Ratio (SNR).

$$y_t = x_t + f(x_t \mid \mu, \sigma^2) \tag{4.10}$$

$$f(x_t \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_t-\mu)^2}{2\sigma^2}} \tag{4.11}$$

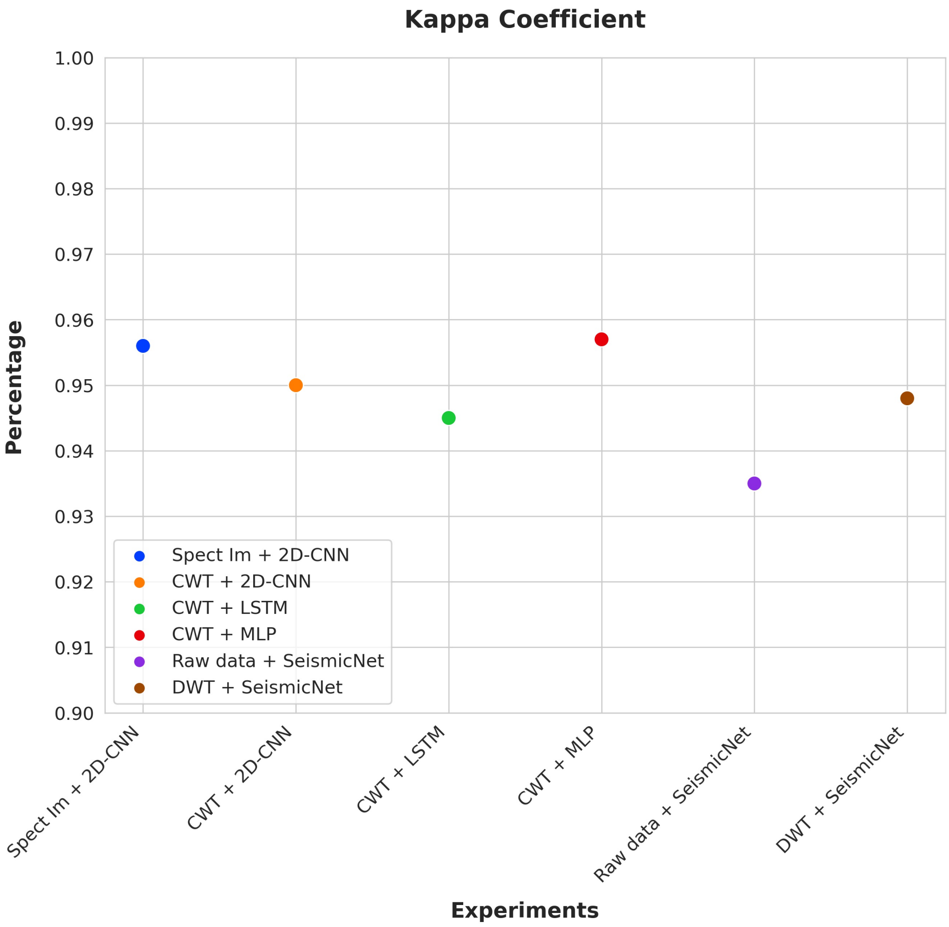


Figure 4.16 The Kappa coefficients results overview.

In Equation 4.10, x_t and y_t are, respectively, the input and the output in a time instant t. Also, $f(x \mid \mu, \sigma^2)$ represents the normal distribution function where μ is the signal mean or expectation of the distribution (and also its median and mode), σ is the standard deviation, and σ^2 is the signal variance. In this experiment, we fixed the mean to zero.

For every resultant dataset, we tested the differences between the data with/without noise. Firstly, we performed a Shapiro-Wilk normality test in order to analyze if there is sufficient evidence that the results came from a normal distribution (p-value $> 5\%$). Secondly, we used the Bartlett's test to confirm that the variances are homogeneous (p-value $< 5\%$). Finally, we apply the Student's t-distribution test to confirm that there is no significant difference at 5%. In case of the Shapiro-Wilk test fails, we check if the results are significantly different using the Wilcoxon test, which emphasizes that the results are truly different. Table 4.13 lists the results of the Shapiro-Wilk test for each noisy seismic data that was generated.

By analyzing the individual values, one may notice that as the noise level increases, the p-value for the Shapiro-Wilk test decays. It is important to recall that the Shapiro-

Table 4.13 Statical tests for each noise level applied on the seismic data.

Noise level	Shapiro-Wilk	Wilcoxon	Bartlett's	t-Student
1.5%	0.9770	-	0.3241	0.816
2%	0.9770	-	0.3241	0.816
5%	0.7611	-	0.0557	0.9982
10%	0.1417	-	4.9803×10^{-5}	0.552
15%	1.3734×10^{-6}	0.5201	-	-
20%	1.7657×10^{-4}	0.0232	-	-
25%	2.0168×10^{-5}	0.0038	-	-
35%	6.2018×10^{-6}	0.4139×10^{-3}	-	-
40%	5.6608×10^{-5}	0.5609×10^{-3}	-	-
50%	6.3144×10^{-6}	0.1688×10^{-3}	-	-

Wilk test will fail if the probability value is lower than 0.05, which occurred when the noise level reached 15%. Also, for those datasets where the test failed, the Wilcoxon test demonstrates that the signals with/without noise are indeed different.

On the datasets that passed on the Shapiro-Wilk test, which in our analyses were those with noise levels lower than 15%, the Bartlett's and Student's t-distribution tests were applied, as shown in respective columns of Table 4.13. As one can observe, the variance was homogeneous (p-value < 0.05) only when the noise level was equal to 10%. Besides that, results demonstrated that was no significant difference for SNR=5% (p-value > 0.05) for these tested datasets.

As a conclusion, these initial results have suggested our hypothesis is feasible and our proposed project is worth being investigated. Finally, we emphasize these tests show some relation between noise and modeling. However, aiming at understanding better the noise influence on the DNN models, we decided to use XAI as shown in the following section.

4.3 MODEL INTERPRETABILITY EXPERIMENT

This section details the experimental setup designed to better understand the effects of noises on signals collected from the Llaima volcano. Firstly, we describe the process of transforming signals from 1D to 2D before using the classification method. Secondly, we show the performance measures considered to quantify the classification differences in our study. Next, we present our CNN architecture and, finally, we discuss the obtained results, including the overall measures and the interpretable features extracted by the Layer-wise Relevance Propagation (LRP) approach, the XAI approach considered in our study.

4.3.1 Seismic Spectrograms

Before starting the classification interpretability, we repeated most of steps of the experiment described in Section 4.1, preprocessing all 1D signals by using the Short-Time Fourier Transform (STFT) method (detailed in Equation 4.12), in which x is a seismic signal and w is the sliding window size.

$$SP[x(t)](n,k) = \left| \sum_{m=0}^{N-1} x(m) \cdot w(m-n) \cdot e^{-i2\pi mk} \right|^2 \tag{4.12}$$

We also used the preprocessing specifications provided by experts from OVDAS, previously described in Section 4.1.2.1. To better evidence the noise influence on the spectrograms, no filter was applied to smooth the outputs. Finally, all spectrograms were transformed to RGB images with dimensions of 20 × 20 pixels. Figure 4.17 illustrates the shape of the spectrograms for the different classes. We randomly selected four signals from every source to exemplify their behavior and support the readers' understanding of the classification tasks.

4.3.2 Performance Measures

Our results were evaluated by using the same measures described in Section 4.1, which are traditionally considered to assess supervised learning tasks. Firstly, the generalization capability of our predictive models was studied by sampling the original dataset with a 10-fold cross-validation strategy. Then, the results were organized into contingency matrices containing the number of true positive (TP), true negative (TN), false positive (FP), and false-negative (FN). Based on such matrices, we calculated the measures: i) Accuracy (Equation 4.2); and ii) F1-score (Equation 4.5). Reinforcing that, in such equations, n represents the total number of classified signals, *Recall* corresponds to the true positive rate (Equation 4.6), and *Precision* takes into account the number of seismic signals correctly classified as a specific event and signals wrongly classified as the same event (Equation 4.7).

4.3.3 Interpretability Results

For this analysis, we decided only to use the architecture with the best results so far, described on the Section 4.1.4.1: a 2D CNN architecture illustrated in Figure 4.5 and published by Curilem *et al.* (2018b) .

Spectrogram Images

Aiming to proceed with our experimental analyses about the noise influence on the classification performance, we split the dataset into three parts: training, validation, and testing. To simulate different SNR, we have modified the original seismic signals by using different additive noise as described in Section 4.2.

In our experiments, signals with different SNR were obtained by only changing the

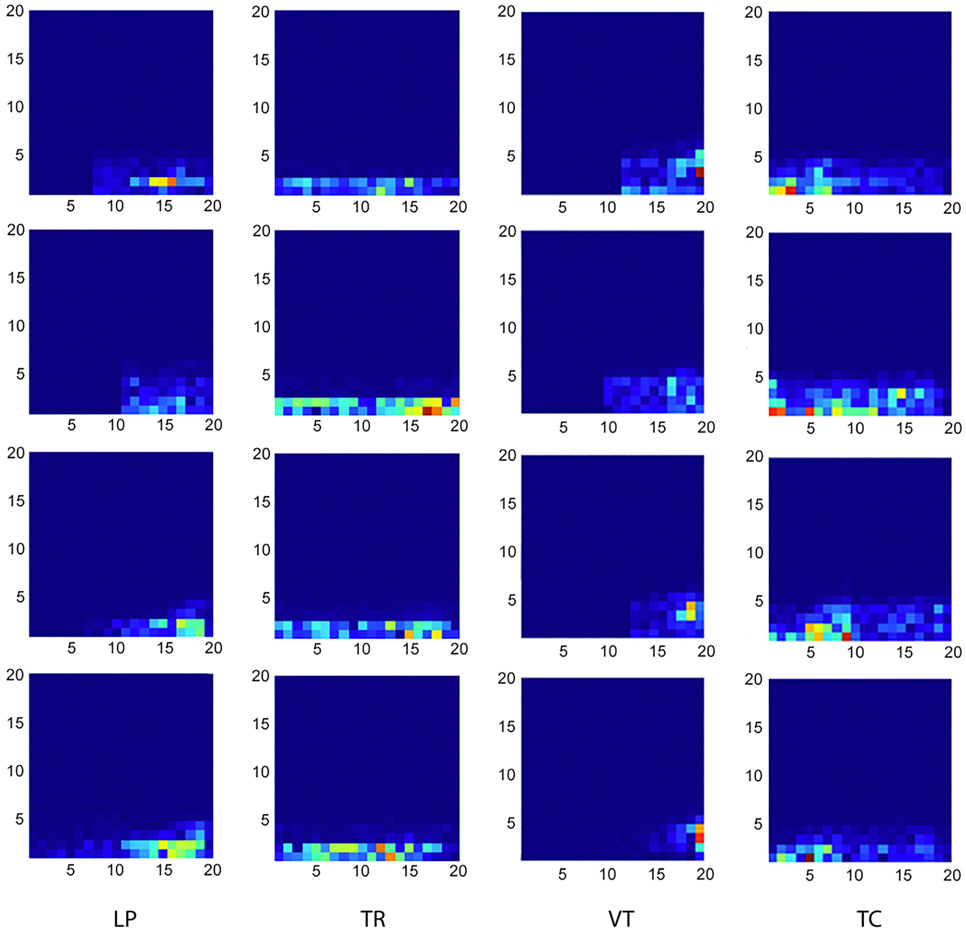


Figure 4.17 Samples of the LP, TR, VT, and TC seismic events converted into 20 x 20 spectrograms images.

standard deviation values. Then, we performed the same classification experiment previously performed on the original signals. Table 4.14 summarizes, in every row, the final results by SNR. In this analysis, we show the classification results obtained from the test fold. As the reader might notice, the greater the SNR influence, the lower the accuracies and F1-scores. With SNR equals to or greater than 20%, the performances approaches to the random classifier. Such observation is confirmed by the correlation tests presented in the last three rows. The tests were performed considering the Pearson method and the R coefficients are closer to -1, indicating a negative correlation, i.e., the accuracy and F1-scores have an inverse behavior when compared to the noise level. The significance of our tests is confirmed by the p-values and the confidence intervals with 95%.

Before proceeding with the interpretability study, we have also analyzed the classification performance by class as shown in Table 4.15. Such analyses were fundamental to understand better whether the noise influences equally affect signals from different sources (classes). We only present the results with accuracy to avoid being redundant once the F1-score showed similar behavior.

Table 4.14 Classification performances by considering different noise levels (SNR).

Noise Level	Accuracy	F1-score
0%	95.6%	94.2%
1%	95.5%	94.3%
2%	94.7%	92.8%
3%	93.5%	91.6%
4%	92.3%	90.0%
5%	90.0%	87.3%
10%	74.7%	69.9%
15%	62.3%	55.1%
20%	55.4%	48.1%
25%	53.2%	45.5%
30%	48.2%	40.7%
R	-0.979	-0.979
p-value	< 0.00001	< 0.00001
CI (95%)	(-0.995, -0.920)	(-0.995, -0.920)

Table 4.15 Individual classification performances per class by considering different noise levels (SNR).

	Accuracy per Seismic Event			
Noise Level	LP	TR	VT	TC
0%	97.13%	98.65%	98.06%	97.39%
1%	97.22%	98.48%	98.23%	97.13%
2%	97.13%	98.23%	97.47%	96.71%
3%	97.05%	97.47%	97.30%	95.36%
4%	96.71%	96.29%	97.13%	94.52%
5%	95.36%	95.28%	96.37%	93.09%
10%	90.05%	85.83%	92.83%	80.86%
15%	83.98%	79.51%	90.05%	71.08%
20%	77.23%	78.25%	88.11%	67.37%
25%	72.43%	81.20%	88.28%	64.50%
30%	67.20%	83.14%	86.42%	59.70%
R	-0.994	-0.876	-0.978	-0.982
p-value	< 0.00001	0.00039	< 0.00001	< 0.00001
CI (95%)	(-0.999, -0.978)	(-0.968, -0.584)	(-0.995, -0.916)	(-0.995, -0.929)

According to these accuracy results, we noticed that, although all classes were affected by the increasing SNR (see accuracies when SNR is 30%), signals from LP and TC presented the worse results. After knowing such classification particularities, we visually interpreted the patterns of the additive noises on the seismic signals. For a more in-depth comparison, Table 4.16 presents the confusion matrix for datasets with 0% noise and 20% noise.

Table 4.16 Confusion matrix obtained on evaluation of CNN with 0% and 20%.

	Noise Level 0%				**Noise Level 20%**			
	LP	**TR**	**VT**	**TC**	**LP**	**TR**	**VT**	**TC**
LP	**413**	1	11	8	**244**	76	33	80
TR	3	**151**	0	8	7	**137**	4	14
VT	6	0	**94**	0	18	3	**17**	62
TC	5	4	6	**476**	56	154	21	**260**

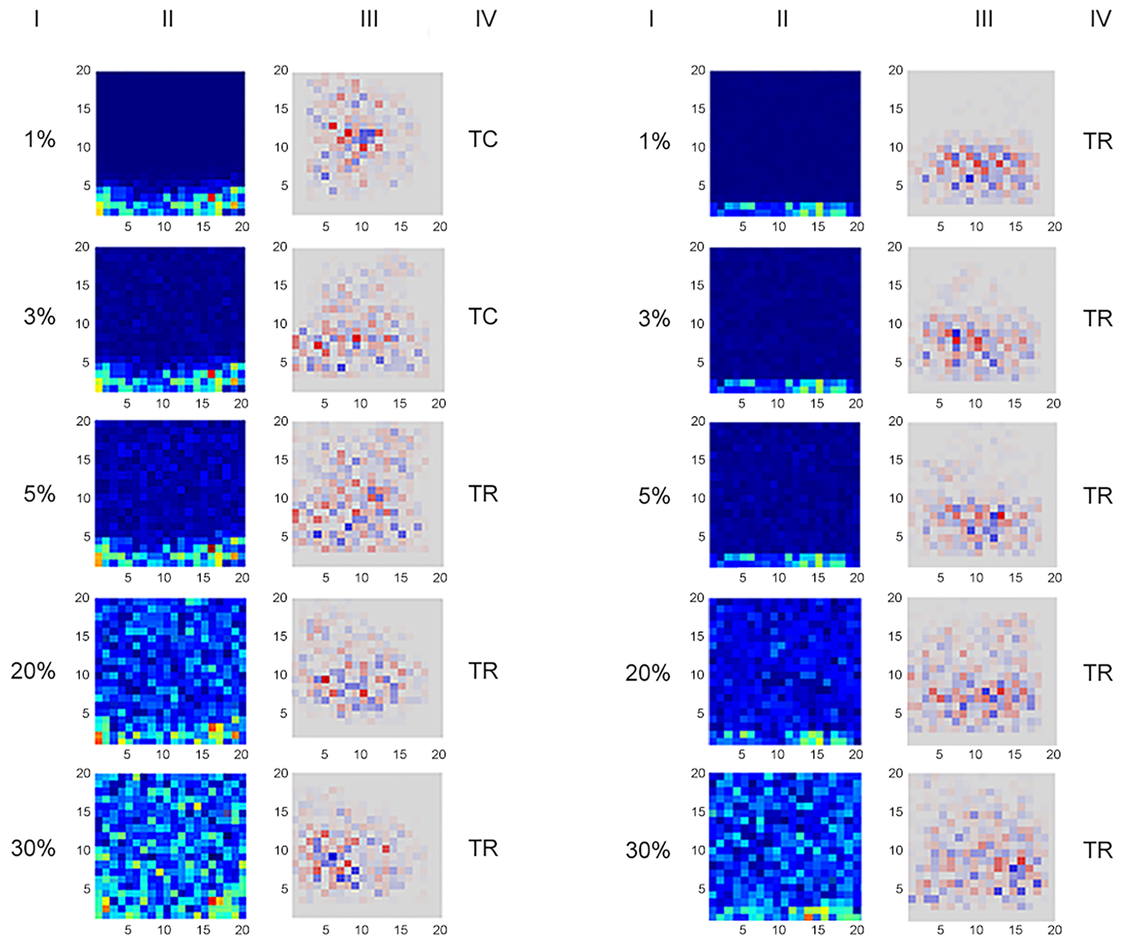
The visual interpretation was conducted using the LRP approach, whose main advantage is the possibility of identifying regions on the seismic spectrograms that lead the classifiers towards specific classes. In summary, the approach highlights the pixels in the spectrograms that our CNN was focused during the classification process. Pixels in red were considered relevant during this process. In turn, pixels that negatively affected the classification are shown in blue. Using such a procedure, we have identified how natural noise might affect the signals collected from Llaima. To illustrate our findings, Figure 4.18a exemplifies a few LRP visualizations from a TC signal by considering different noises.

In this figure, Column I shows the noise percentages. Columns II and III present the spectrogram and LRP images by noise. Finally, in Column IV, we see the final class provided by our CNN. As one may notice, LRP is strongly important to identify relevant pixels as the noise increases. In this example, as the noise is greater than 3%, the patterns of the red (positive influence) and blue (negative influence) pixels change, inducing the classifier to select a class TR for the signal.

Aiming to understand the transition between those two classes better, we also plotted (Figure 4.18b) the relation between spectrograms and LRP images for a signal TR correctly classified regardless of the noise level. Knowing the region where red and blue pixels are placed, we can find the relationship between those two classes (TR and TC), thus leading to a misclassification when the noise level is more significant. Finally, by using the inverse STFT, we can reconstruct the signal and locate the red and blue influences on the raw signal.

Spectrograms

In the following experiments, we decided to directly work on the spectrograms outputs, instead of using their resultant images, i.e., we modified our approach to process the spectrograms outputs, without saving them as 20×20 images in prior step to avoid losing information. To proceed with this modification, we computed the spectrogram module as



(a) TC seismic event.

(b) TR seismic event.

Figure 4.18 LRP visual interpretation to the pixel influence analysis on our CNN during the classification process. Labels (I - IV) on top explain the meanings of the plot columns: I) percentage of noise applied on the seismic event; II) seismic spectrograms; III) LRP visualizations; and IV) predicted classes.

defined in Equation 4.13, in which a and b are the real and imaginary parts, respectively.

$$Mod(SP[x(t)](n,k)) = \sqrt{a^2 + bi^2}. \tag{4.13}$$

Table 4.17 summarizes, in every row, the final results by SNR. As previously noticed with spectrogram images, the greater the SNR influence, the lower the accuracies and F1-scores. However, differently from the spectrogram image classification, the modules of spectrograms classification demonstrate an improvement of the noise tolerance, in which the classifier performance only approaches the random classifier when the noise ratio is equal to or greater than 25%. The tests were performed considering the Pearson method and the R coefficients are closer to -1, indicating a negative correlation, i.e., the accuracy and F1-scores have an inverse behavior when compared to the noise level. The significance of our tests is confirmed by the p-values and the confidence intervals with 95%.

Table 4.17 Spectrograms classification performances by considering different noise levels (SNR).

Noise Level	Accuracy	F1-score
0%	94.34%	92.93%
1%	94.25%	92.29%
2%	94.81%	93.35%
3%	94.62%	92.87%
4%	93.51%	91.98%
5%	93.32%	91.40%
10%	85.44%	78.49%
15%	73.10%	60.11%
20%	62.06%	44.90%
25%	56.22%	40.50%
30%	51.11%	32.60%
R	-0.979	-0.979
p-value	< 0.00001	< 0.00001
CI (95%)	(-0.995, -0.920)	(-0.995, -0.920)

As presented in Section 4.3.3, we also analyzed the classification performance by class as shown in Table 4.18. This fundamental analysis allowed us to identify that the overall classification performance is mainly reduced by the TC class, which is the seismic target most impacted by the SNR influence.

Compared with the accuracy results described in Table 4.15, we noticed a slight improvement on the performance of the four classes. Figure 4.19 exemplifies a few LRP visualizations for a TR signal by considering different noises.

Similarly to the previous analysis, in Figure 4.20, we illustrate the visualizations produced by LRP after analyzing an LP signal modified with different noise levels.

Another advantage of using the spectrogram outputs, instead of their transformed images, is the possibility of applying an inverse transformation to reconstruct the signal from the frequency domain to the time one. As as consequence, we can identify how each observation influences the classification process. By using this transformation, we have noticed that the VT and TR classes are strongly influenced by frequency, the LP class by time, and TC is influenced by both frequency and time. Thus, to understand those relationships better, we cropped the LP signals at 25 seconds and compared them to the original ones, whose the results are shown in Table 4.19.

Another important application observed from our study with inverse transformation is the possibility of using the explanation approach as a filter to remove noise from signals, after visualizing parts of the signal that was affecting positively and negatively the classification process.

Table 4.18 Individual classification performances per class by considering different noise levels (SNR).

	Accuracy per Seismic Event			
Noise Level	**LP**	**TR**	**VT**	**TC**
0%	96.38%	98.33%	98.24%	96.85%
1%	96.38%	98.33%	98.24%	96.85%
2%	96.29%	98.24%	98.24%	96.85%
3%	95.92%	98.14%	98.14%	96.66%
4%	96.20%	98.33%	98.33%	96.75%
5%	96.38%	98.24%	98.52%	96.85%
10%	97.12%	97.12%	98.24%	95.64%
15%	95.92%	89.89%	97.03%	87.11%
20%	91.19%	86.36%	93.04%	73.38%
25%	77.74%	86.36%	91.74%	56.22%
30%	66.33%	86.36%	91.56%	44.62%

Table 4.19 Hit rate of cropped LP x original LP class considering different noise levels (SNR).

	Hit rate	
Noise Level	**LP (cropped)**	**LP (original)**
0%	98.47%	97.96%
1%	98.73%	96.69%
2%	98.47%	96.69%
3%	97.46%	96.18%
4%	97.46%	95.17%
5%	97.71%	95.17%
10%	92.37%	90.08%
15%	75.57%	72.26%
20%	55.98%	51.40%
25%	33.59%	32.06%
30%	21.88%	26.21%

Essentially, LRP has the same format as the network input, meaning that if we are classifying the spectrograms, we can accurately tell which frequencies and samples in time have positive and negative influence in the model performance. By analyzing the LRP output of TR and VT signals with 5% of additive noise, one may notice a clear separation around the 5Hz mark, as shown in the top-most plots of Figure 4.21. In this figure, the frequencies above that mark have mainly a negative impact for the TR class and a positive influence for the VT one.

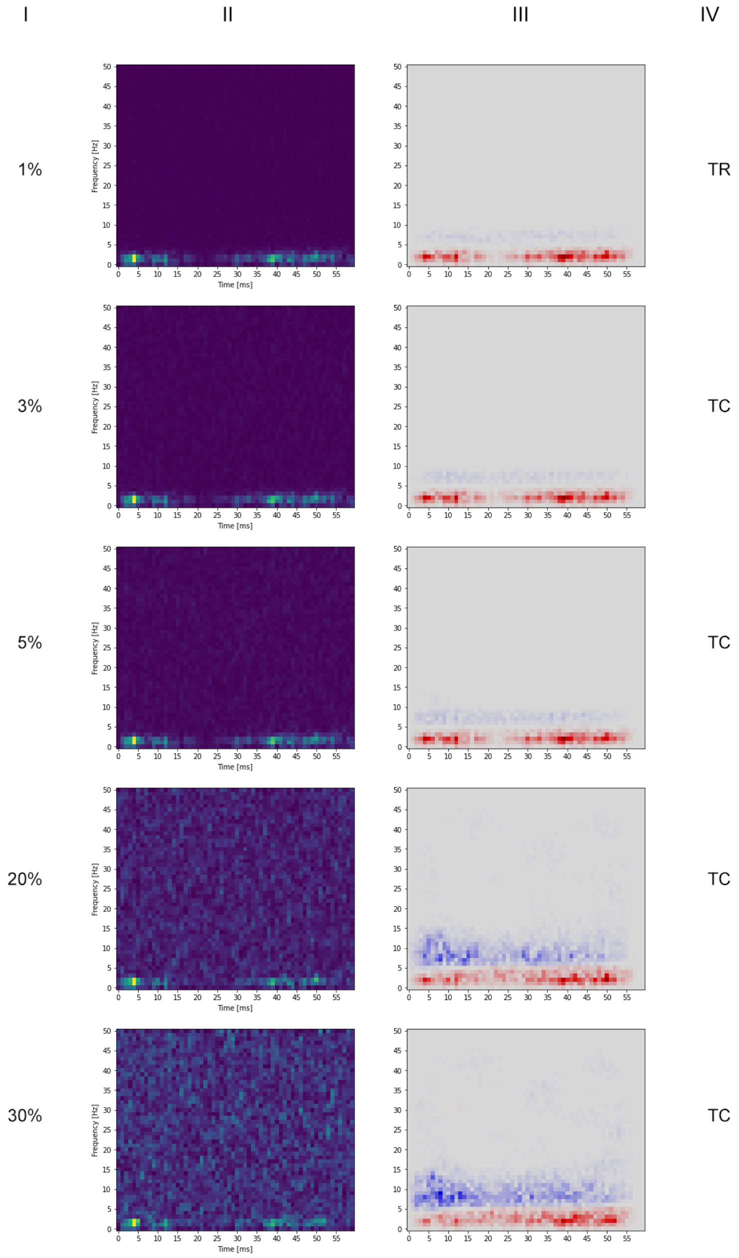


Figure 4.19 Labels (I - IV) on top explain the meanings of the plot columns: I) percentage of noise applied on an TC seismic event; II) seismic spectrograms; III) LRP visualizations; and IV) predicted classes.

When a network classifies a given spectrogram, it analyzes features in both frequency and time domains. Therefore, we decided to apply a low and a high pass filter with 5Hz to the signal, to create two different views of it. We also split LRP at the same frequency and summed each section over the y-axis, thus creating 1D arrays over time representing noise and signals. Samples with negative, positive, and neutral impacts are represented

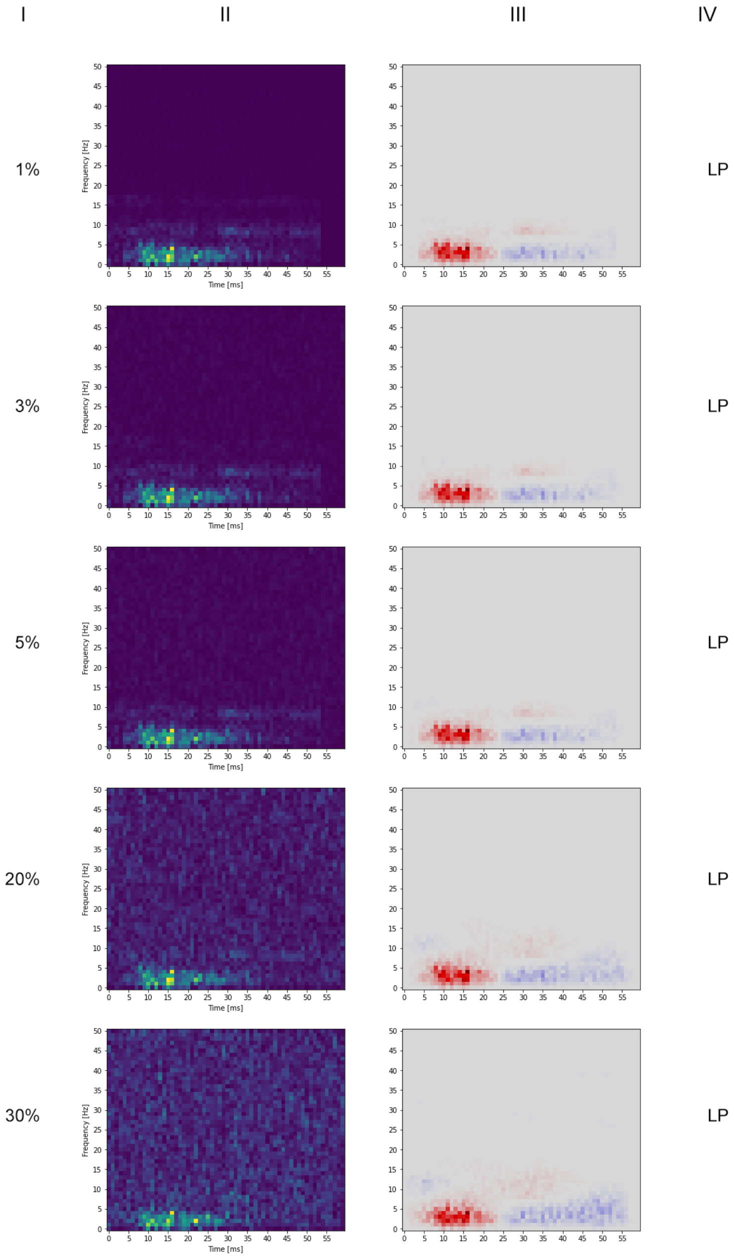


Figure 4.20 Labels (I - IV) on top explain the meanings of the plot columns: I) percentage of noise applied on an LP seismic event; II) seismic spectrograms; III) LRP visualizations; and IV) predicted classes.

by blue, red, and gray colors (Figures 4.21 C-F).

In Figure 4.21, the middle plots (C and D) are then low-pass filtered signals for both TR and VT classes, while the bottom plots (E and F) are the high-pass filtered signals for both classes. The low-pass filter extracted positive and negative impacts for the TR and VT classes, respectively. The opposite happened when we applied a high-pass filter.

Then, we can conclude that the TR class would benefit from low-pass filters to pre-process the signals, while the VT class would benefit from high-pass filters.

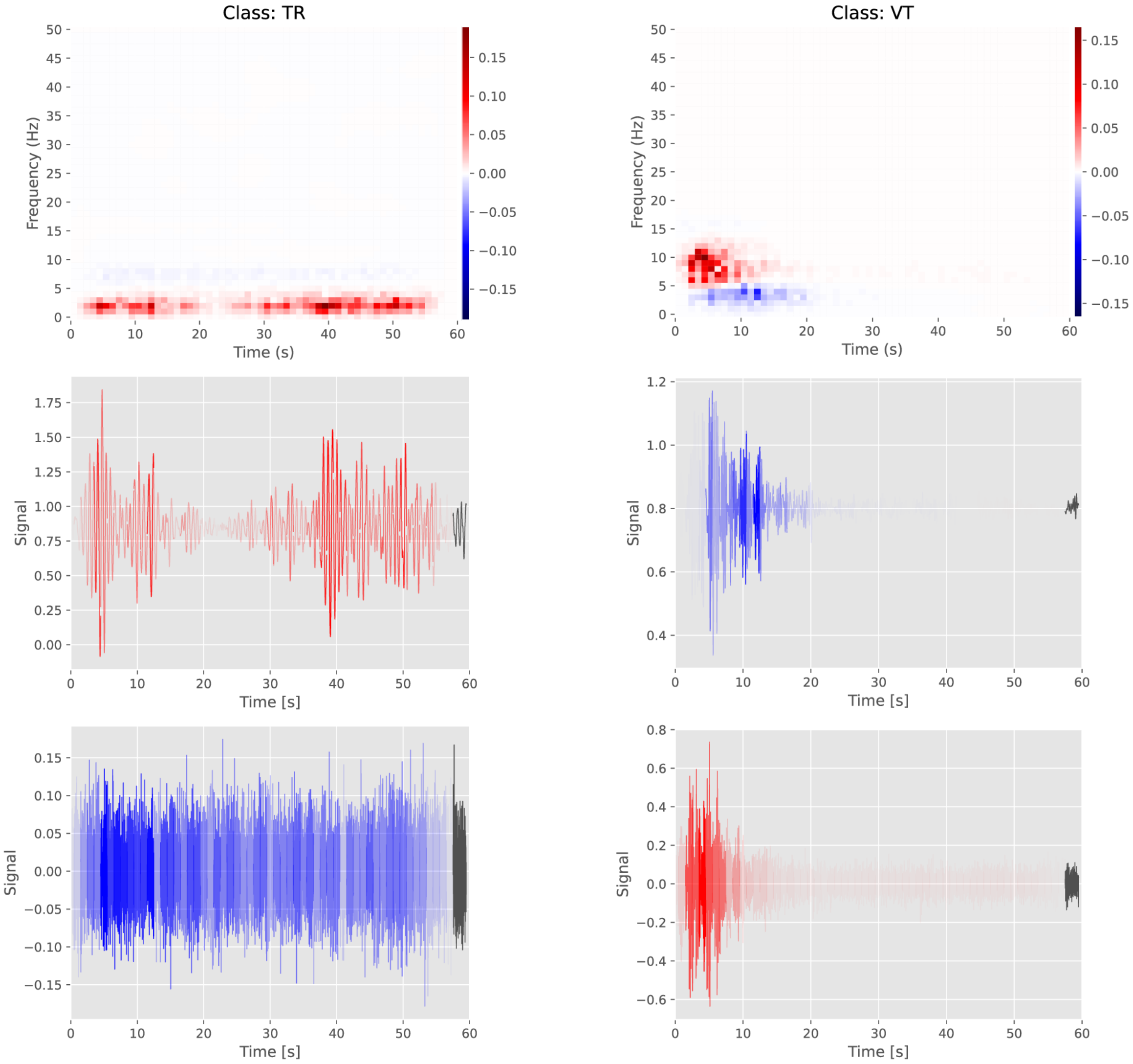


Figure 4.21 Example of a TR class on the left and a VT class on the right. The top plots are the LRP outputs, which have samples with positive impact displayed in red and negative in blue. The middle and bottom plots correspond to the signals after a low a high-pass filter respectively. Again, the samples shown in red have a positive effect on the network, the samples in blue have a negative impact, and the ones in gray have no effect.

Finally, we performed a batch experiment to assess the general behavior of the filter process. In summary, we considered the Mean Absolute Error (MAE) to calculate the error between the original signals against the noisy and filtered ones. The violin plots shown in Figure 4.22 shows the general error reduces when the filtered signal is considered, highlighting the importance of this process.

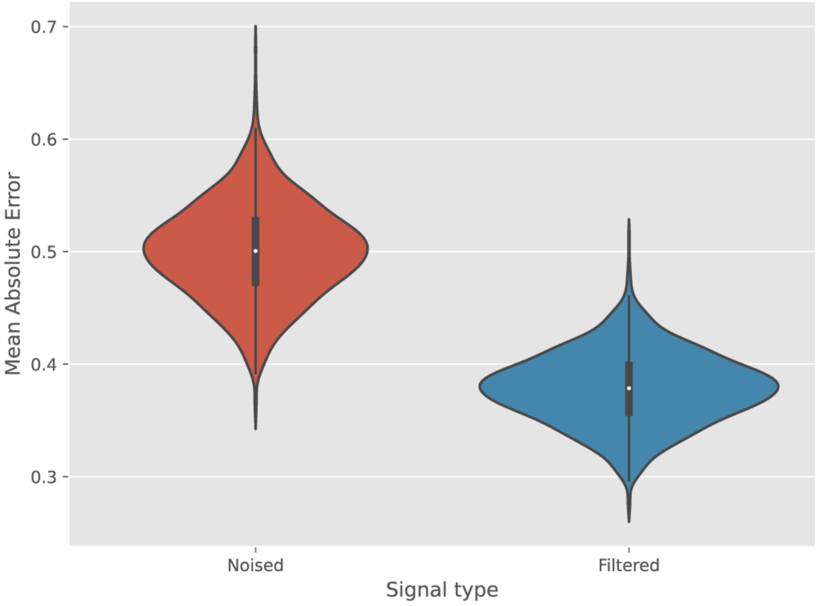


Figure 4.22 Violin plot of the mean absolute error of the noised and filtered signals the original signal.

4.4 DEEP LEARNING FEATURES FOR NOISY TOLERANT TIME SERIES CLASSIFICATION

The experiments presented in previous sections were very relevant to emphasize the importance of noise in DNN models. Based on conclusions drawn from the interpretability process, we were able to design a new architecture that can be used as LSTM memory cells to create models more robust to noise. As a consequence, our contribution modifies the memory cells, thus making them act as noise filters and suppressing the pre-processing step.

The evaluation of our architecture was conducted by using the same measures considered so far, described in Section 4.1. The generalization process of our predictive models was also studied by sampling the original dataset with a 10-fold cross-validation approach. Moreover, we compared the obtained results using an LSTM neural network with and without our cell. Next, we discuss the proposed architecture and the obtained results.

4.4.1 Our Proposed Artificial Neural Network Architecture

Firstly, we develop a LSTM neural network to classify raw 1D seismic signals using the same strategies described in Section 4.1, summarized in the following steps:

- Adding or removing extra fully-connected layers;

- Testing different configurations of layer hyper-parameters, such as the dropout rate and the number of neurons at each dense layer;

- Adding dropout layers after convolutional layers and dense layers to avoid overfitting.

We evaluated every adaptation and ended up with the following LSTM composition:

1. One LSTM layer $LSTM(Un, RS)$, in which Un is the output space dimensionality and RS is the return sequence flag to ensure that LSTM cells return all of outputs from the unrolled LSTM memory cell through time.

2. One dense layer expressed as $F(u)$, having u as the number of units or neurons at such a layer;

3. In the output of both LSTM and dense layers, we applied the ReLU activation function $f(x) = max(0, x)$, so that x is the output from neuron layers;

4. Two dropout layers $D(dr)$, in which dr is the dropout rate, after apply the ReLU activation function;

5. Then, a softmax activation function was applied on the output of our network.

Next, to make a network architecture capable of filtering noises present in the raw data, we firstly embed a convolutional layer in the memory cell of the LSTM layer used by the DNN previously described. As shown in Figure 4.23, the convolutional layer is based on SeismicNet (CANÁRIO *et al.*, 2020).

Our final LSTM memory cell is illustrated in Figure 4.24, in which the symbol $X \hookrightarrow Y$ show the places where the original LSTM cell was modified. As one may notice, our contribution directly extracts features from the sum of the overall inputs in the LSTM memory cell. The extracted features are, then, delivered to the LSTM gating mechanism and, in the next stage, classified by the LSTM dense layer.

Another positive aspect of our cell is related to its usage in practical scenarios. The user can train its parameters to learn better the noisy behavior of their data. Another possibility is to consider frozen models to support, for example, one-shot learning processes.

Finally, we have created final neural network, referred to as SeismicLSMTNet (Figure 4.25), putting together all architectures presented in this section:

1. An input layer $I(w, h)$, having w as its width and h as its height;

2. One LSTM layer $LSTM(Un)$, in which Un is the output space dimensionality, and an embed CNN in it. This CNN is finally composed of:

   (a) An input layer $I(ch)$, in which ch is the number of channels of the input data;

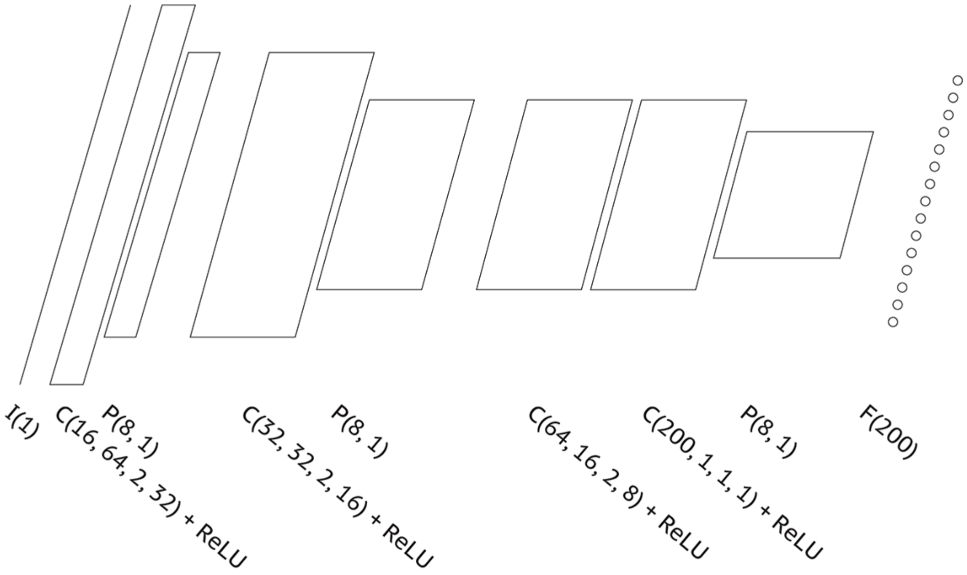


Figure 4.23 Embed CNN summary: input layer, conv. 1D + ReLU, max pool. 1D, conv. 1D + ReLU, max pool. 1D, conv. 1D + ReLU, conv. 1D + ReLU, max pool. 1D, and dense layer.

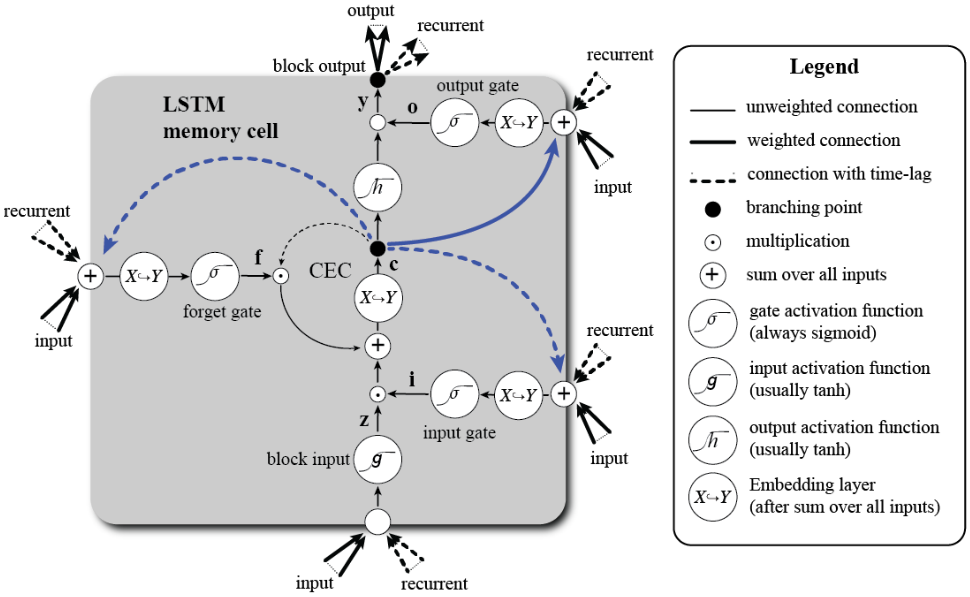


Figure 4.24 LSTM memory cell with the embed layers extracting features from the sum of the overall inputs.

(b) Four convolutional layers $C(K, F, S, P)$, having K as the number of filters, F as the spatial extent, S as the stride, and P as the padding length;

(c) Three max pooling layers denoted as $P(F, S)$, in which F is the size of the square pooling region, and S is its stride;

(d) One dense layer expressed as $F(u)$, having u as the number of units or neurons at such a layer;

(e) In the output of every convolutional, we applied the ReLU activation function $f(x) = max(0, x)$, so that x is the output from neuron layers;

3. One dense layer expressed as $F(u)$, having u as the number of units or neurons at such a layer;
4. In the output of both LSTM and dense layers, we applied the ReLU activation function $f(x) = max(0, x)$, so that x is the output from neuron layers;
5. Two dropout layers $D(dr)$, in which dr is the dropout rate, after apply the ReLU activation function;
6. Then, a softmax activation function was applied on the output of our network.

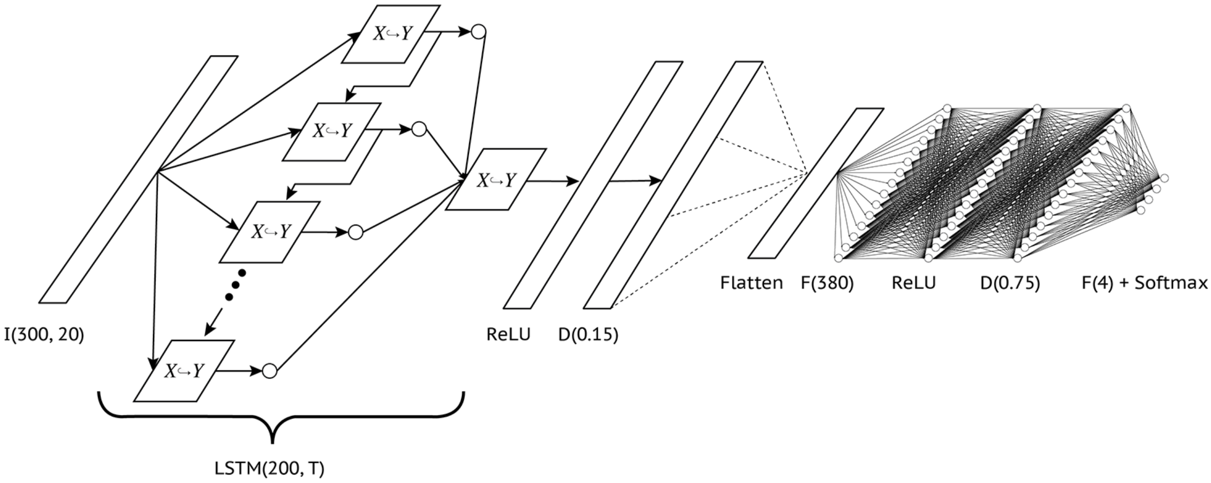


Figure 4.25 SeismicLSMTNet summary: input, LSTM with embed CNN, ReLU, dropout, flatten, dense, ReLU, dropout and output layer with softmax activation function.

4.4.2 Results

This section shows the results obtained with our proposed DNN architecture referred to as SeismicLSMTNet. Table 4.20 summarizes the F1-scores for the LSTM Network without our memory cell and SeismicLSMTNet, obtained after running a 10-fold cross-validation strategy. As one may notice, the LSTM architecture shows great results, presenting a mean F1-score greater than 91%. It is worth mentioning that such results are also very close to the ones obtained in previous experiments. However, despite the SeismicLSMTNet presenting in the F1-score values slightly lower, we notice that was caused only by the poor performance in one of the folds.

Tables 4.21 and 4.22 illustrates, for each noise level added to the raw seismic signals, the results calculated from both classifiers per class instead of just presenting their overall performances.

Table 4.20 K-fold cross-validation comparison: SeismicLSMTNet x LSTM Network.

	LSTM Network	**SeismicLSMTNet**
	F1-Score	**F1-Score**
Mean:	**91.75%**	**88.05%**
Std:	**3.12%**	**15.07%**
Fold 1:	83.61%	42.94%
Fold 2:	90.88%	90.90%
Fold 3:	92.64%	92.32%
Fold 4:	92.85%	93.12%
Fold 5:	93.32%	91.77%
Fold 6:	93.09%	93.17%
Fold 7:	94.16%	94.31%
Fold 8:	94.61%	94.48%
Fold 9:	94.35%	94.19%
Fold 10:	94.56%	93.36%

Table 4.21 LSTM Network individual classification performances per class by considering different noise levels (SNR).

	F1-Score per Seismic Event			
Noise Level	**LP**	**TR**	**VT**	**TC**
0%	96.46%	97.22%	87.50%	97.33%
1%	79.20%	94.29%	29.06%	80.54%
2%	79.69%	91.97%	28.57%	80.32%
3%	77.83%	91.97%	25.00%	79.11%
4%	76.06%	90.37%	28.10%	79.04%
5%	77.59%	89.55%	20.56%	78.48%
10%	63.90%	80.65%	6.19%	72.40%
15%	43.03%	69.33%	0.00%	66.82%
20%	20.14%	44.44%	0.00%	62.13%
25%	7.35%	22.89%	0.00%	59.96%
30%	3.98%	9.09%	0.00%	59.24%

By analyzing the individual values for each class, side by side, one may notice the overall performance for the SeismicLSMTNet architecture keeps stable even increasing the noise levels (SNR) in the raw seismic signals. In the LP event, illustrated in Figure 4.26 at the noise level of 5%, the LSTM Network begins to decay very quickly in contrast to SeismicLSTMNet.

On the TR event, the LSTM Network presents better performance than Seismi-

Table 4.22 SeismicLSMTNet individual classification performances per class by considering different noise levels (SNR).

	F1-Score per Seismic Event			
Noise Level	**LP**	**TR**	**VT**	**TC**
0%	96.11%	95.83%	89.47%	96.03%
1%	80.47%	65.77%	25.93%	77.04%
2%	80.06%	63.93%	25.93%	76.74%
3%	80.00%	66.37%	22.64%	76.91%
4%	80.00%	65.77%	20.95%	76.78%
5%	80.24%	62.67%	20.95%	76.64%
10%	81.12%	64.55%	2.17%	76.33%
15%	79.10%	65.77%	0.00%	75.38%
20%	72.67%	64.25%	0.00%	73.33%
25%	64.04%	69.23%	0.00%	70.89%
30%	53.97%	65.10%	0.00%	68.56%

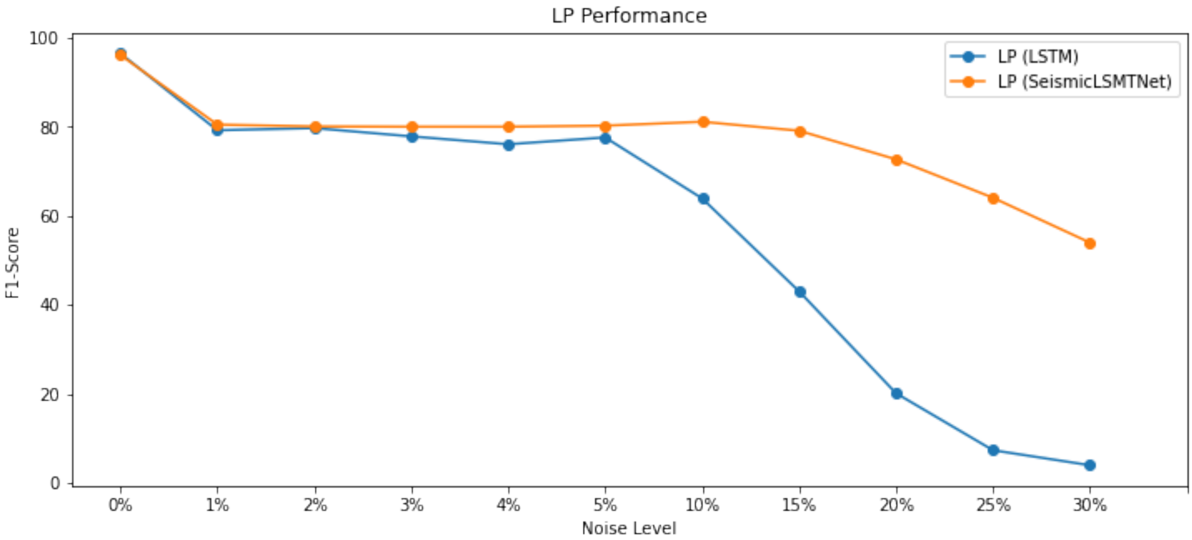


Figure 4.26 LP classification performance of LSTM Network and SeismicLSTMNet by considering different SNR.

cLSMTNet for the noise level below 15%. After this point, the performances are inverted and the LSTM Network decays until reaches a score closes to zero. Figure 4.27 shows the TR classification performance of both architectures.

In the case of the VT and TC rates, both LSTM Network and SeismicLSMTNet presented a similar behavior, as are illustrated by Figures 4.28 and 4.29. However, in the TC event, after the seismic events reached a noise level of 5%, SeismicLSMTNet showed a lower decaying when compared with the LSTM Network.

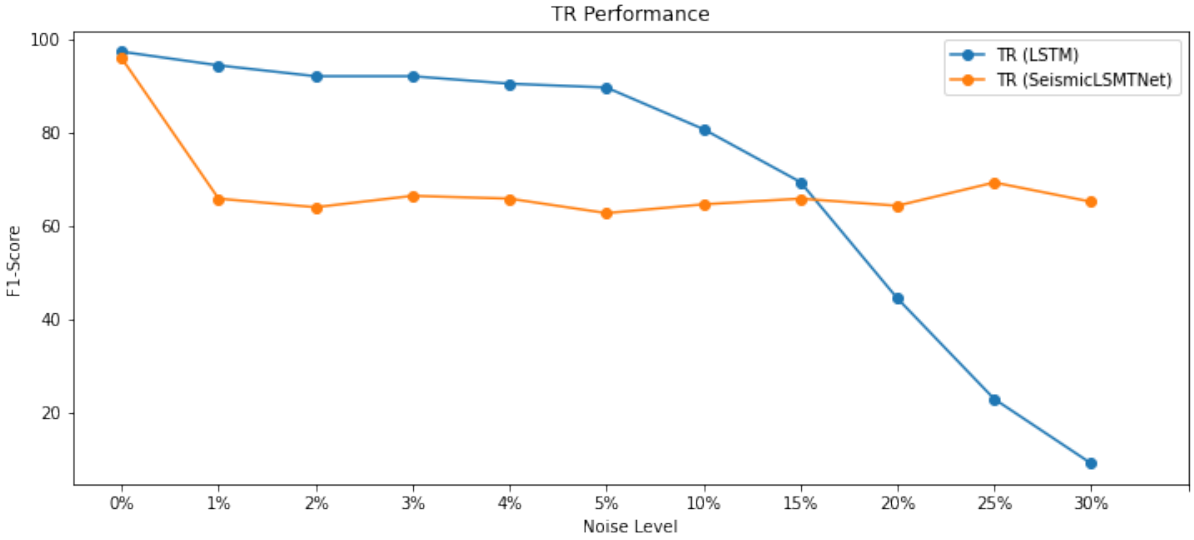


Figure 4.27 TR classification performance of LSTM Network and SeismicLSTMNet by considering different SNR.

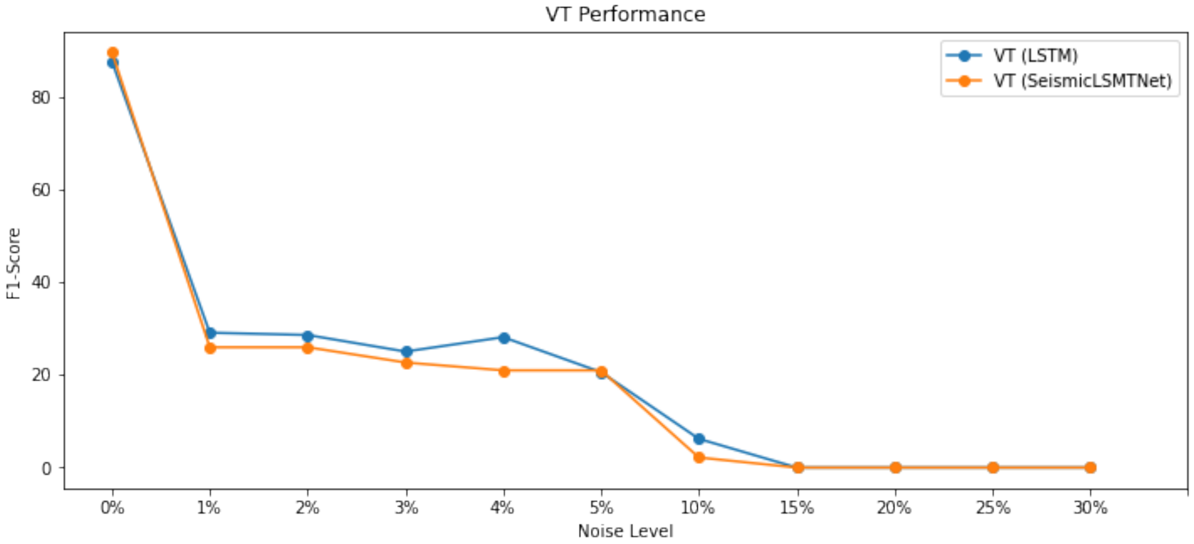


Figure 4.28 VT classification performance of LSTM Network and SeismicLSTMNet by considering different SNR.

In contrast with most machine learning studies, in which intensive feature extraction steps and data transformations are usually adopted during the preprocessing stage, our SeismicLSTMNet architecture directly uses seismic signals. As a consequence of using our CNN embedded into LSTM layers, we have created an architecture with a stable performance despite the addition of noise in the analyzed signals. In summary, our final architecture was able to directly model one-dimensional signals, without transforming them to be later used by machine learning algorithms. Moreover, SeismicLSMTNet pre-

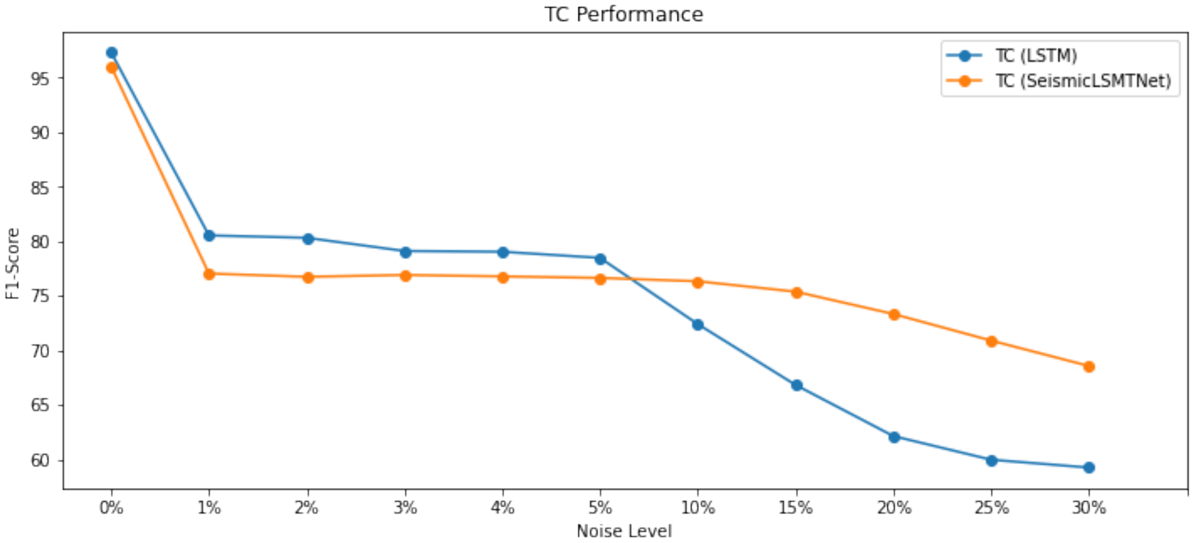


Figure 4.29 TC classification performance of LSTM Network and SeismicLSTMNet by considering different SNR.

sented a very stable behavior by facing higher noise levels. Finally, tt is worth emphasizing that we performed a deep analysis on the Llaima signals training multiple variations of the SeismicLSMTNet. In this section, we only present the architecture that provided the best results, but we fully describe the path to the development of SeismicLSTMNet in Appendix A.

Chapter

5

CONCLUSION

In this work, we investigated the influence of noises on Deep Neural Network (DNN) models and present a novel deep learning approach with an embedding noise filter capable of suppressing the preprocessing stage. Instead of just applying noise reduction filters before modeling some data, as usually performed in the Machine Learning (ML) area, our proposal relies on the capability of learning noisy patterns, thus delivering more stable results regardless of the noise level added to signals and improving the modeling of real-world systems without requiring a decomposition/filtering stage.

The development of this approach demonstrated the validity of our initial hypothesis that stated a filter created to work as an embedding layer in Long Short-Term Memory Networks improves the modeling of noisy time series without requiring a previous decomposition/filtering step.

At the begging of our journey to demonstrate it, we started modeling a real-world application using DNN. Essentially, we collaborated with researchers from the Universidad de La Frontera to model signals collected from the Llaima volcano. Firstly, we developed a 2D Convolutional Neural Network (CNN) to recognize spectrograms from seismic events. This CNN architecture could classify spectrograms without the need for a high number of feature extractors. The obtained results motivated us to conduct an intensive study on the use of DNNs to classify seismic signals. The consequence of this study was the development of a new DNN architecture, named SeismicNet, which provides comparable results to the literature and demands no extra effort from the end-user to explicitly preprocess seismic signals. In summary, all learning stages were encapsulated in a single piece of software.

Next, we created a controlled scenario in which different synthetic noise influences were added to signals to measure the impact on SeismicNet. Then, we extended this experiment in two directions. Firstly, we analyzed the 2D CNN architecture with the best classification results and confirmed that the presence of noise significantly affects the final results. Secondly, we have focused on understanding the propagation of different noise levels on the DNN models. In this sense, we have used the Layer-wise Relevance Propagation (LRP) approach to visualize the impact of noises on the signal predictions.

By incrementally varying the noise levels in our experiments, LRP highlighted the spectrogram parts that negatively and positively contributed to classification errors.

The results obtained with LRP allowed us to produce another important contribution that was a study about using DNN and eXplainable Artificial Intelligence (XAI) as a filter bank, thus separating stochastic influences from noisy signals.

After performing such a comprehensive investigation to comprehend better the noise effects on DNN models, we have created the main contribution of our work: a new memory cell to make Long Short-Term Memory (LSTM) models more robust. In summary, we have embedded a CNN into the LSTM memory cell to act as a noise filter suppressing the preprocessing stage. Analyzing the obtained results, we noticed that this novel approach learned the noisy features better and could deliver stable results apart from the noise level added to the seismic signal.

Although our contributions have provided motivating results, there are some limitations that are worth investigating. Firstly, we understand that the study of noise influences with XAI has come up with a new research topic related to adopting tools to separate noise from signals. Despite demonstrating how low- and high-pass filters can be considered in such a scenario, it is necessary to dedicate more effort to analyzing other filters and their applicability in more complex signals.

Moreover, our final contribution has created an environment that allows the implementation of different preprocessing functions inside LSTM cells. For example, instead of using complex DNN models, one can implement traditional signal processing tools as, for example, the Wavelet transform and plug it in our memory cell. According to preliminary experiments, we have noticed that wavelets can improve the classification results by only using the first component levels extracted from the Discrete Wavelet Transform (DWT). The improvement is explained by removing details from components, which are usually considered stochastic behavior, i.e., noise. Therefore, by creating a learnable filter in this way, the model result can be improved due to the lower complexity obtained by downsampling observations while keeping the general signal behavior.

Finally, the application considered in this dissertation was completely focused on modeling seismic signals. However, the contributions yielded from our results can be applied to signals collected from any system. Therefore, in another future work, we intend to apply them to neuroscience signals, which are also investigated by our research group.

BIBLIOGRAPHY

ALAYRAC, J.-B. *et al. Flamingo: a Visual Language Model for Few-Shot Learning.* arXiv, 2022. Disponível em: ⟨https://arxiv.org/abs/2204.14198⟩.

AMATO, F. *et al. Artificial neural networks in medical diagnosis.* [S.l.]: Elsevier, 2013.

AYTAR, Y.; VONDRICK, C.; TORRALBA, A. Soundnet: Learning sound representations from unlabeled video. In: *Advances in Neural Information Processing Systems.* [S.l.: s.n.], 2016. p. 892–900.

BACH, S. *et al.* On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, Public Library of Science, v. 10, n. 7, p. e0130140, 2015.

BAGNALL, A. *et al.* The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery*, Springer, v. 31, n. 3, p. 606–660, 2017.

BENGIO, Y.; LECUN, Y. Scaling learning algorithms towards ai. *Large-scale kernel machines*, v. 34, n. 5, p. 1–41, 2007.

BHATTI, S. M. *et al.* Automatic detection of volcano-seismic events by modeling state and event duration in hidden markov models. *Journal of Volcanology and Geothermal Research*, v. 324, p. 134–143, 2016.

BOX, G. E. *et al. Time series analysis: forecasting and control.* [S.l.]: John Wiley & Sons, 2015.

CANÁRIO, J. P. *et al.* In-depth comparison of deep artificial neural network architectures on seismic events classification. *Journal of Volcanology and Geothermal Research*, Elsevier, v. 401, p. 106881, 2020.

CHAN, N. T. *et al.* Agent-based models of financial markets: A comparison with experimental markets. *Unpublished Working Paper, MIT Artificial Markets Project, MIT, MA*, 1999.

CHATFIELD, D. C. *et al.* The bullwhip effect—impact of stochastic lead time, information quality, and information sharing: a simulation study. *Production and Operations Management*, Wiley Online Library, v. 13, n. 4, p. 340–353, 2004.

CHATTERJEE, S. Bootstrapping arma models: Some simulations. *IEEE transactions on systems, man, and cybernetics*, IEEE, v. 16, n. 2, p. 294–299, 1986.

CHIKKERUR, S.; CARTWRIGHT, A. N.; GOVINDARAJU, V. Fingerprint enhancement using stft analysis. *Pattern recognition*, Elsevier, v. 40, n. 1, p. 198–211, 2007.

CHOI, N. G.; KULICK, D. B.; MAYER, J. Financial exploitation of elders: Analysis of risk factors based on county adult protective services data. *Journal of Elder Abuse & Neglect*, Taylor & Francis, v. 10, n. 3-4, p. 39–62, 1999.

CHOUET, B. A. Long-period volcano seismicity: Its source and use in eruption forecasting. *Nature*, v. 380, n. 6572, p. 309–316, 1996.

CHOWDHERY, A. *et al. PaLM: Scaling Language Modeling with Pathways.* arXiv, 2022. Disponível em: ⟨https://arxiv.org/abs/2204.02311⟩.

COHEN, L. *Time-frequency analysis.* [S.l.]: Prentice hall New Jersey, 1995.

CURILEM, M. *et al.* Using cnn to classify spectrograms of seismic events from llaima volcano (chile). In: IEEE. *2018 International Joint Conference on Neural Networks (IJCNN).* [S.l.], 2018b. p. 1–8.

CURILEM, M. *et al.* Pattern recognition applied to seismic signals of llaima volcano (chile): An evaluation of station-dependent classifiers. *Journal of Volcanology and Geothermal Research*, v. 315, p. 15–27, 2016.

CURILEM, M. *et al.* Discriminating seismic events of the llaima volcano (chile) based on spectrogram cross-correlations. *Journal of Volcanology and Geothermal Research*, v. 367, p. 63 – 78, 2018a. ISSN 0377-0273.

CURILEM, M. *et al.* Improving the classification of volcanic seismic events extracting new seismic and speech features. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. CIARP 2017. Lecture Notes in Computer Science.* [S.l.]: Springer, Cham, 2017. (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 10657), p. 177–185.

DAUBECHIES, I. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, Wiley Online Library, v. 41, n. 7, p. 909–996, 1988.

DIETTERICH, T. Machine learning for sequential data: A review. *Structural, syntactic, and statistical pattern recognition*, Springer, p. 227–246, 2002.

ESLING, P.; AGON, C. Time-series data mining. *ACM Computing Surveys (CSUR)*, ACM, v. 45, n. 1, p. 12, 2012.

FAMA, E. F. The behavior of stock-market prices. *The journal of Business*, JSTOR, v. 38, n. 1, p. 34–105, 1965.

GAMA, J. *et al.* A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 46, n. 4, p. 1–37, 2014.

GAMBOA, J. C. B. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

GEOGHEGAN, R. Time series analysis and its applications: with r examples. Springer, 2006.

GERS, F. A.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: Continual prediction with lstm. In: *9th International Conference on Artificial Neural Networks: ICANN '99*. [S.l.: s.n.], 1999. p. 850–855.

GERVEN, M. van; BOHTE, S. *Artificial neural networks as models of neural information processing.* [S.l.]: Frontiers Media SA, 2018.

GOUPILLAUD, P.; GROSSMANN, A.; MORLET, J. Cycle-octave and related transforms in seismic signal analysis. *Geoexploration*, Elsevier, v. 23, n. 1, p. 85–102, 1984.

GRAPS, A. An introduction to wavelets. *IEEE Computational Science and Engineering*, v. 2, p. 50–61, 1995.

GRAVES, A.; SCHMIDHUBER, J. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, Elsevier, v. 18, n. 5-6, p. 602–610, 2005.

GREFF, K. *et al.* Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, IEEE, v. 28, n. 10, p. 2222–2232, 2017.

GRÖCHENIG, K. *Foundations of time-frequency analysis.* [S.l.]: Springer Science & Business Media, 2001.

GUHATHAKURTA, K.; BHATTACHARYA, B.; CHOWDHURY, A. R. Using recurrence plot analysis to distinguish between endogenous and exogenous stock market crashes. *Physica A: Statistical Mechanics and its Applications*, Elsevier, v. 389, n. 9, p. 1874–1882, 2010.

HAMILTON, J. D. *Time series analysis.* [S.l.]: Princeton university press Princeton, 1994.

HAN, M.; LIU, Y. Noise reduction method for chaotic signals based on dual-wavelet and spatial correlation. *Expert Systems with Applications*, Elsevier, v. 36, n. 6, p. 10060–10067, 2009.

HAYKIN, S.; NETWORK, N. A comprehensive foundation. *Neural networks*, v. 2, n. 2004, p. 41, 2004.

HINTON, G. *et al.* Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, IEEE, v. 29, n. 6, p. 82–97, 2012.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HYNDMAN, R. J.; ATHANASOPOULOS, G. *Forecasting: principles and practice*. [S.l.]: OTexts, 2018.

JUMPER, J. *et al.* Highly accurate protein structure prediction with alphafold. *Nature*, Nature Publishing Group, v. 596, n. 7873, p. 583–589, 2021.

JUNG, Y.-J.; HAN, S.-H.; CHOI, H.-J. Explaining cnn and rnn using selective layer-wise relevance propagation. *IEEE Access*, v. 9, p. 18670–18681, 2021.

KALAPANIDAS, E. *et al.* Machine learning algorithms: a study on noise sensitivity. In: *Proc. 1st Balcan Conference in Informatics*. [S.l.: s.n.], 2003. p. 356–365.

KEOGH, E.; KASETTY, S. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and knowledge discovery*, Springer, v. 7, n. 4, p. 349–371, 2003.

KINGMA, D.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KOHLBRENNER, M. *et al.* Towards best practice in explaining neural network decisions with lrp. In: IEEE. *2020 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2020. p. 1–7.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2012. v. 2, p. 1097–1105.

LAHR, J. *et al.* Earthquake classification, location, and error analysis in a volcanic environment: implications for the magmatic system of the 1989–1990 eruptions at redoubt volcano, alaska. *Journal of Volcanology and Geothermal Research*, v. 62, n. 1, p. 137 – 151, 1994. ISSN 0377-0273. The 1989-1990 Eruptions of Redoubt Volcano, Alaska. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/0377027394900310⟩.

LAPTEV, N. *et al.* Time-series extreme event forecasting with neural networks at uber. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2017. v. 34, p. 1–5.

LECUN, Y. *et al.* Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2323, 1998.

LEE, H. *et al.* Unsupervised feature learning for audio classification using convolutional deep belief networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2009. p. 1096–1104.

LÜTKEPOHL, H. *New introduction to multiple time series analysis*. [S.l.]: Springer Science & Business Media, 2005.

MATHIEU, M.; HENAFF, M.; LECUN, Y. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

MONTAVON, G. *et al.* Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, Springer, p. 193–209, 2019.

MONTAVON, G.; SAMEK, W.; MÜLLER, K.-R. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, Elsevier, v. 73, p. 1–15, 2018.

MORETTIN, P. A.; TOLOI, C. M. d. C. Análise de séries temporais. 2004.

NETTLETON, D. F.; ORRIOLS-PUIG, A.; FORNELLS, A. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, Springer, v. 33, n. 4, p. 275–306, 2010.

OSGOOD, B. Lecture notes for ee 261 the fourier transform and its applications. In: . [S.l.: s.n.], 2002.

PONOMARENKO, V. *et al.* Deriving main rhythms of the human cardiovascular system from the heartbeat time series and detecting their synchronization. *Chaos, Solitons & Fractals*, Elsevier, v. 23, n. 4, p. 1429–1438, 2005.

QIU, X. *et al.* Empirical mode decomposition based ensemble deep learning for load demand time series forecasting. *Applied Soft Computing*, v. 54, p. 246 – 255, 2017. ISSN 1568-4946. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S1568494617300273⟩.

RAMESH, A. *et al. Hierarchical Text-Conditional Image Generation with CLIP Latents.* arXiv, 2022. Disponível em: ⟨https://arxiv.org/abs/2204.06125⟩.

REED, S. *et al. A Generalist Agent.* arXiv, 2022. Disponível em: ⟨https://arxiv.org/abs/2205.06175⟩.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. ”why should i trust you?” explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* [S.l.: s.n.], 2016. p. 1135–1144.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. Anchors: High-precision model-agnostic explanations. In: *Proceedings of the AAAI Conference on Artificial Intelligence.* [S.l.: s.n.], 2018. v. 32, p. 1.

RIOS, R. A.; MELLO, R. F. D. Improving time series modeling by decomposing and analyzing stochastic and deterministic influences. *Signal Processing*, Elsevier, v. 93, n. 11, p. 3001–3013, 2013.

RIOS, R. A.; MELLO, R. F. de. Applying empirical mode decomposition and mutual information to separate stochastic and deterministic influences embedded in signals. *Signal Processing*, Elsevier, v. 118, p. 159–176, 2016.

RIPPEL, O.; SNOEK, J.; ADAMS, R. P. Spectral representations for convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 2449–2457.

ROSENBLATT, F. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. [S.l.], 1961.

SÁEZ, J. A. *et al.* Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. *Knowledge and information systems*, Springer, v. 38, n. 1, p. 179–206, 2014.

SATO, K.; HOSOKAWA, K.; MAEDA, M. Rapid aggregation of gold nanoparticles induced by non-cross-linking dna hybridization. *Journal of the American Chemical Society*, ACS Publications, v. 125, n. 27, p. 8102–8103, 2003.

SELTZER, M. L.; YU, D.; WANG, Y. An investigation of deep neural networks for noise robust speech recognition. In: IEEE. *2013 IEEE international conference on acoustics, speech and signal processing*. [S.l.], 2013. p. 7398–7402.

SHANG, W. *et al.* Understanding and improving convolutional neural networks via concatenated rectified linear units. *CoRR*, abs/1603.05201, 2016. Disponível em: ⟨http://arxiv.org/abs/1603.05201⟩.

SHRIKUMAR, A.; GREENSIDE, P.; KUNDAJE, A. Learning important features through propagating activation differences. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2017. p. 3145–3153.

SHUMWAY, R. H.; STOFFER, D. S. Time series regression and exploratory data analysis. *Time Series Analysis and Its Applications: With R Examples*, Springer, p. 48–83, 2006.

SILVER, D. *et al.* Mastering the game of go with deep neural networks and tree search. *nature*, Nature Publishing Group, v. 529, n. 7587, p. 484, 2016.

SLAVIČ, J.; SIMONOVSKI, I.; BOLTEŽAR, M. Damping identification using a continuous wavelet transform: application to real data. *Journal of Sound and Vibration*, Elsevier, v. 262, n. 2, p. 291–307, 2003.

SMILKOV, D. *et al.* Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

SPRINGENBERG, J. T. *et al.* Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

STATHAKIS, D. How many hidden layers and nodes? *International Journal of Remote Sensing*, Taylor & Francis, v. 30, n. 8, p. 2133–2147, 2009.

SUMMA, M. G. *et al.* A new clustering method for time series to discover geographical cancer trends from 1960 to 2000. *Annals of epidemiology*, Elsevier, v. 17, n. 9, p. 744, 2007.

SUNDARARAJAN, M.; TALY, A.; YAN, Q. Gradients of counterfactuals. *arXiv preprint arXiv:1611.02639*, 2016.

SUTSKEVER, I.; HINTON, G. E.; TAYLOR, G. W. The recurrent temporal restricted boltzmann machine. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2009. p. 1601–1608.

TOMSETT, R. *et al.* Interpretable to whom? a role-based model for analyzing interpretable machine learning systems. *arXiv preprint arXiv:1806.07552*, 2018.

TSCHACHER, W.; KUPPER, Z. Time series models of symptoms in schizophrenia. *Psychiatry Research*, Elsevier, v. 113, n. 1, p. 127–137, 2002.

VASILACHE, N. *et al.* Fast convolutional nets with fbfft: A gpu performance evaluation. *arXiv preprint arXiv:1412.7580*, 2014.

WITTEN, I. H. *et al.* Data mining: Practical machine learning tools and techniques. In: _____. [S.l.]: Morgan Kaufmann, 2016. p. 1–621.

YANG, H.-F.; CHEN, Y.-P. P. Hybrid deep learning and empirical mode decomposition model for time series applications. *Expert Systems with Applications*, v. 120, p. 128 – 138, 2019. ISSN 0957-4174. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0957417418307395⟩.

YANG, Q.; WU, X. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, World Scientific, v. 5, n. 04, p. 597–604, 2006.

ZHUANG, X. *et al.* Robust registration between cardiac mri images and atlas for segmentation propagation. In: *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*. [S.l.: s.n.], 2008. p. 691408.

Appendix

A

APPENDIX - PATH TO THE SEISMICLSTMNET

This appendix details the entire analysis process to reach the final architecture of SeismicLSTMNet and the designed experiments to deal with noise effects on the seismic signals collected from Llaima. Next, we describe the design process of each Artificial Neural Network (ANN) architecture assembled to deliver more stable results regardless of the noise level added to the signals.

The evaluation of each model was conducted by using the same measures considered in previous experiments, illustrated in Section 4.1. The generalization process of our models was also studied by sampling the original dataset with a 10-fold cross-validation approach. Moreover, we compared the obtained results using an LSTM neural network with and without our cell.

A.0.1 First Stage: Embedding Position

Firstly, we conducted a test between three methods to analyze which one could perform better on attenuation of the noise effect on our dataset: a Gaussian filter, a smooth average filter, and a convolutional layer.

Table A.1 summarizes the F1-scores per seismic classes considering different noise levels (Signal-to-Noise Ratio (SNR)) and obtained after running a 10-fold cross-validation strategy. As one may notice, the convolutional layer shows more promising results, presenting better noise attenuation.

Next, we reviewed the most suitable position to set the convolutional layer: before the LSTM cell inputs or after the sum of the overall inputs. Figure A.1 illustrates the position review, in which the symbols $X \hookrightarrow Y$ and $X \hookrightarrow Y$ reveal the modification locations on the original LSTM cell.

Table A.2 illustrates for each noise level added to the raw seismic signals the convolutional layer position results. As it was noticed, the layer placed after the sum of the overall inputs reached better results. However, we noticed that the layer alone could not deliver the expected results. Consequently, we presumed that a neural network could yield those results.

Table A.1 Noise filters: individual classification performances per class by considering different noise levels (SNR).

		F1-Score per Seismic Event			
	Noise Level	**LP**	**TR**	**VT**	**TC**
Gaussian Filter	0%	93.45%	92.78%	81.53%	95.48%
	1%	40.62%	23.95%	6.38%	64.50%
	2%	40.32%	20.73%	6.38%	64.08%
	3%	38.08%	16.25%	6.38%	63.58%
	4%	38.97%	13.92%	2.17%	63.58%
	5%	37.90%	11.54%	4.30%	63.08%
	10%	16.63%	2.68%	0.00%	60.28%
	15%	4.95%	0.00%	0.00%	59.05%
	20%	0.00%	0.00%	0.00%	58.62%
	25%	0.00%	0.00%	0.00%	58.62%
	30%	0.00%	0.00%	0.00%	58.62%
Smooth Average Filter	0%	93.11%	94.12%	78.95%	95.95%
	1%	25.49%	11.54%	4.30%	61.57%
	2%	24.67%	10.32%	2.17%	61.44%
	3%	22.72%	11.54%	2.17%	61.27%
	4%	20.72%	11.54%	0.00%	61.02%
	5%	18.76%	7.84%	4.30%	60.69%
	10%	10.10%	1.35%	0.00%	59.56%
	15%	1.01%	0.00%	0.00%	58.70%
	20%	0.51%	0.00%	0.00%	58.66%
	25%	0.00%	0.00%	0.00%	58.62%
	30%	0.00%	0.00%	0.00%	58.62%
Convolutional Layer	0%	93.71%	93.99%	83.23%	96.12%
	1%	57.14%	13.92%	14.29%	68.20%
	2%	57.14%	13.92%	8.42%	67.89%
	3%	55.84%	15.09%	8.42%	67.89%
	4%	53.91%	16.25%	14.29%	66.97%
	5%	52.98%	10.32%	16.16%	66.52%
	10%	29.83%	0.00%	0.00%	62.00%
	15%	10.50%	0.00%	0.00%	59.51%
	20%	2.01%	0.00%	0.00%	58.82%
	25%	0.00%	0.00%	0.00%	58.62%
	30%	0.00%	0.00%	0.00%	58.62%

A.0.2 Second Stage: Network Complexity

Based on previous stage results, we conduct our tests on three new architectures: two SeismicNet variations (CANÁRIO *et al.*, 2020) and a simpler CNN. The first embed

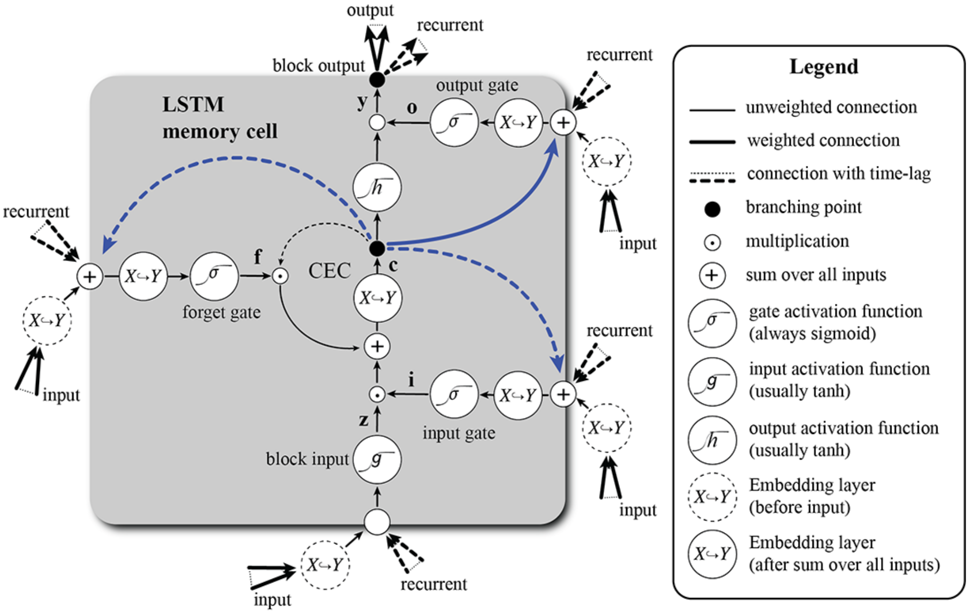


Figure A.1 Interventions in the LSTM memory cell to set the convolutional layer's best position.

Table A.2 K-fold cross-validation comparison: Embed Layer (Before inputs) x Embed Layer (After sum of the overall inputs).

	Embed Layer (Before inputs)	**Embed Layer (After sum of the overall inputs)**
	F1-Score	**F1-Score**
Mean:	**52.14%**	**87.93%**
Std:	**28.67%**	**1.81%**
Fold 1:	34.71%	90.65%
Fold 2:	14.66%	86.08%
Fold 3:	47.59%	88.22%
Fold 4:	83.54%	86.09%
Fold 5:	14.66%	85.75%
Fold 6:	78.36%	87.09%
Fold 7:	73.35%	89.72%
Fold 8:	82.29%	89.63%
Fold 9:	14.66%	86.20%
Fold 10:	77.58%	89.91%

architecture (Figure A.2) has the following network composition:

1. An input layer $I(ch)$, in which ch is the number of channels of the input data;

2. Four convolutional layers $C(K, F, S, P)$, having K is the number of filters, F as the spatial extent, S is the stride, and P is the zero-padding length;

3. Three max-pooling layers denoted as $P(F, S)$, in which F is the size of the square pooling region, and S is its stride;

4. In the output of the second, third, and fourth convolutional layers, we applied the ReLU activation function $f(x) = max(0, x)$, so that x is the output from neuron layers;

5. Finally, at the end of the network, we flatten the resultant features.

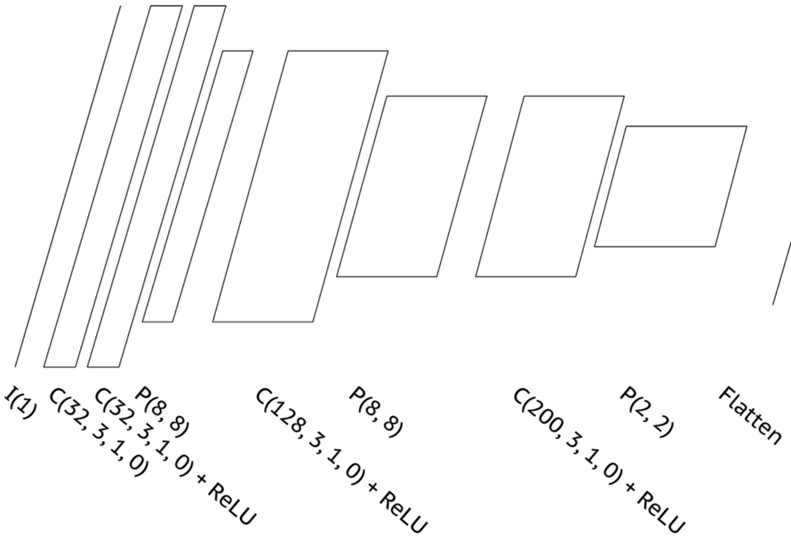


Figure A.2 Embed CNN (version 1) summary: input layer, conv. 1D, conv. 1D + ReLU, max pool. 1D, conv. 1D + ReLU, max pool. 1D, conv. 1D + ReLU, max pool. 1D, and a flatten layer.

The second architecture, illustrated by Figure A.3, was a more complex version of our first trial, and it has the following network:

1. An input layer $I(ch)$, in which ch is the number of channels of the input data;

2. Six convolutional layers $C(K, F, S, P)$, having K is the number of filters, F as the spatial extent, S is the stride, and P is the zero-padding length;

3. Three max-pooling layers denoted as $P(F, S)$, in which F is the size of the square pooling region, and S is its stride;

4. In the output of each convolutional layers, we applied the ReLU activation function $f(x) = max(0, x)$, so that x is the output from neuron layers;

5. Finally, at the end of the network, we flatten the resultant features.

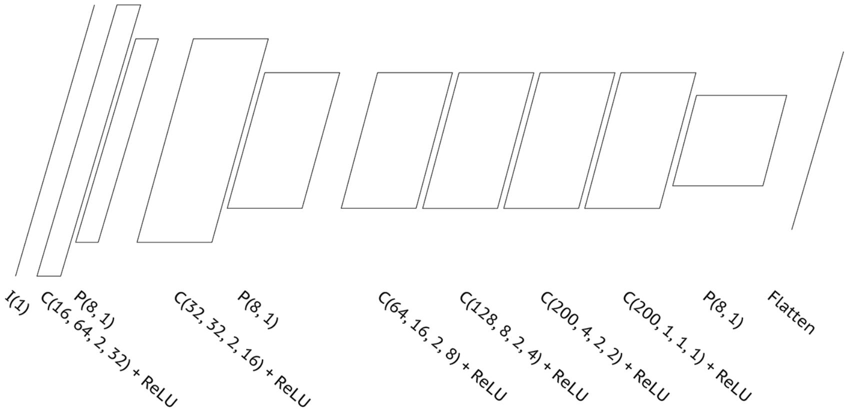


Figure A.3 Embed CNN (version 2) summary: input layer, conv. 1D + ReLU, max pool. 1D, conv. 1D + ReLU, max pool. 1D, conv. 1D + ReLU, conv. 1D + ReLU, conv. 1D + ReLU, conv. 1D + ReLU, max pool. 1D, and a flatten layer.

The third created architecture is the simpler CNN that we tested. This architecture (Figure A.4) is composed as follows:

1. One convolutional layer $C(K, F, S, P)$, having K is the number of filters, F as the spatial extent, S is the stride, and P is the zero-padding length;
2. One max-pooling layer denoted as $P(F, S)$, in which F is the size of the square pooling region, and S is its stride;
3. In the output of the convolutional layer, we applied the ReLU activation function $f(x) = max(0, x)$, so that x is the output from neuron layers;
4. One dense layer expressed as $F(u)$, having u as the number of units or neurons at such a layer.

Therefore, we compared each architecture performance against the LSTM Network without any embed network in the memory cell. Table A.3 and Table A.4 synthesizes the obtained results of the LSTM Network and each architecture, respectively.

After carefully analyzing the results obtained from each model, we perceived that the most possible track to reach our goal is some ANN architecture derived from the second architecture. Therefore, we develop the SeismicLSTMNet using the same strategies described in Section 4.1, summarized in the following steps:

- Testing different configurations of layer hyper-parameters, such as the dropout rate, the kernel size, the number of filters in convolutional layers and the number of neurons at each fully-connected layer;
- Adding dropout layers after convolutional layers and fully-connected layers to avoid overfitting.

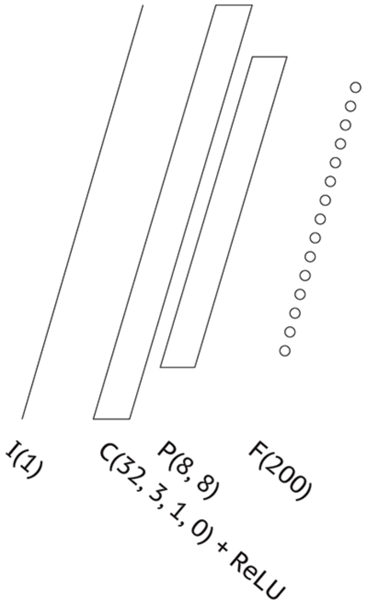


Figure A.4 Embed CNN (version 3) summary: input layer, conv. 1D + ReLU, max pool. 1D, and a dense layer.

Table A.3 LSTM Network: individual classification performances per class by considering different noise levels (SNR).

		F1-Score per Seismic Event			
	Noise Level	**LP**	**TR**	**VT**	**TC**
LSTM Network	0%	93.45%	92.78%	81.53%	95.48%
	1%	40.62%	23.95%	6.38%	64.50%
	2%	40.32%	20.73%	6.38%	64.08%
	3%	38.08%	16.25%	6.38%	63.58%
	4%	38.97%	13.92%	2.17%	63.58%
	5%	37.90%	11.54%	4.30%	63.08%
	10%	16.63%	2.68%	0.00%	60.28%
	15%	4.95%	0.00%	0.00%	59.05%
	20%	0.00%	0.00%	0.00%	58.62%
	25%	0.00%	0.00%	0.00%	58.62%
	30%	0.00%	0.00%	0.00%	58.62%

Table A.4 CNN Architectures: individual classification performances per class by considering different noise levels (SNR).

		F1-Score per Seismic Event			
	Noise Level	**LP**	**TR**	**VT**	**TC**
Architecture 1	0%	97.47%	96.22%	91.01%	96.77%
	1%	0.00%	24.44%	0.00%	3.41%
	2%	0.00%	24.40%	0.00%	3.00%
	3%	0.00%	24.36%	0.00%	3.01%
	4%	0.00%	24.34%	0.00%	3.02%
	5%	0.00%	24.28%	0.00%	3.47%
	10%	0.00%	24.28%	0.00%	3.90%
	15%	0.00%	24.15%	0.00%	3.89%
	20%	0.00%	24.33%	0.00%	5.51%
	25%	0.00%	24.34%	0.00%	8.59%
	30%	0.00%	24.91%	0.00%	17.29%
Architecture 2	0%	96.21%	95.77%	91.53%	96.79%
	1%	27.13%	0.00%	0.00%	61.05%
	2%	24.44%	0.00%	0.00%	60.76%
	3%	21.72%	0.00%	0.00%	60.43%
	4%	16.78%	0.00%	0.00%	60.04%
	5%	12.41%	0.00%	0.00%	59.64%
	10%	0.51%	0.00%	0.00%	58.66%
	15%	0.00%	0.00%	0.00%	58.62%
	20%	0.00%	0.00%	0.00%	58.62%
	25%	0.00%	0.00%	0.00%	58.62%
	30%	0.00%	0.00%	0.00%	58.62%
Architecture 3	0%	96.04%	93.95%	87.70%	96.13%
	1%	5.93%	0.00%	17.13%	59.42%
	2%	4.48%	0.00%	16.37%	58.81%
	3%	3.01%	0.00%	14.07%	58.14%
	4%	1.52%	0.00%	13.88%	58.02%
	5%	1.01%	0.00%	13.91%	57.82%
	10%	0.51%	0.00%	14.42%	57.89%
	15%	0.00%	0.00%	8.11%	58.12%
	20%	0.00%	0.00%	1.80%	57.79%
	25%	0.00%	0.00%	0.00%	57.94%
	30%	0.00%	4.79%	0.00%	58.34%