UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

telecos BCN

Deep Learning for Earthquake detection

Degree Thesis
submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Sergi Mus

In partial fulfillment
of the requirements for the degree in
Enginyeria de Telecomunicacions amb Menció de Telemàtica

Advisor: Beatriz Otero (Universitat Politècnica de Catalunya)
Co-advisor: Otilio Rojas (Barcelona Supercomputing Center)
Barcelona, July 2021

Contents

List of Figures

List of Tables

List of Listings

Abbreviations

ANN	Artificial Neural Network(s)
CNN	Convolutional Neural Network(s)
DL	Deep Learning
FFNN	Feed Forward Neural Network(s)
FUNVISIS	Fundación Venezolana de Investigaciones Sismológicas
ML	Machine Learning
MLP	Multi Layer Perceptron
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descen
SLP	Single Layer Perceptron

Abstract

The creation of accurate ground motion models can only be achieved with the help of vast amounts of labelled data. The manual cataloging makes the processing of this data. The project proposes the automation of the labeling through real data and DL.

Previous study show that CNN are the best architecture for this type of problems. The project also implements FFNN, a simpler architecture, with the intent of achieving competitive results. To make this objective possible the project proposes a novel preprocessing.

The results show that CNN reach an accuracy of 98.2%, while FFNN achieves 91.2%. Moreover, the project includes an encoder based algorithm to approximate arrival times to the station. Finally, the project make use of meta-learning to detect seismic events providing from a single station.

Resumen

Modelos realistas para la detección automática de sismos solo se pueden conseguir analizando grandes cantidades de datos. Estos datos son difíciles de procesar por catalogado manual. Este proyecto plantea la automatización de este proceso usando datos reales y técnicas de DL.

Estudios anteriores demuestran que los mejores resultados, para resolver este tipo de problemas, se alcanzan usando CNN. A parte de estas redes, el proyecto implementa redes más simples, como las FFNN, esperando obtener resultados competitivos respecto a los obtenidos por las CNN. Para alcanzar este objetivo, el proyecto propone una técnica de preprocesamiento novedosa de trazas sísmicas.

Los resultados muestran que CNN llega a detectar el 98,2% de los eventos, mientras que FFNN lo realiza en un 91,2%. Por otra parte, el proyecto incluye un algoritmo basado en codificadores para aproximar el tiempo de arrivo del evento a la estación. Finalmente, el proyecto utiliza la metodología de meta-learning para realizar detección de sismos para los datos que provienen de una única estación.

Resum

Models realistes per la detecció automàtica de sismes només es poden aconseguir analitzant un gran volum de dades. Aquestes dades són difícils de processar amb un catalogat manual. Aquest projecte proposa la automatització d'aquest procés mistjançant el us de dades reals i tècniques de DL.

Estudis anteriors demostren que els millors resultats, per resoldre problemes d'aquest tipus, s'obtenen utilitzant CNN. A part d'aquestes xarxes, el projecte implement unes xarxes més simple, com les FFNN, esperant obtenir resultats competitius en comparació als obtinguts per les CNN. Per obtenir aquest objectiu, el projecte proposa una tècnica de preprocessament de traces sísmiques innovadora.

Els resultats mostren que les CNN arriben a detectar el 98,2% dels events, mentre que les FFNN ho fan en un 91,2%. Per un altra part, el projecte inclou un algoritme basat en codificadors per aproximar el temps de arribada del event a la estació. Finalment, el projecte utilitza la metodologia de meta-learning per realitzar la detecció de sismes per les dades que provenen exclusivament d'una única estació.

Revision history and approval record

Revision	Date	Purpose
0	13/04/2021	Document creation
1	26/04/2021	Document implementation
2	05/05/2021	Document implementation
3	11/05/2021	Document implementation
4	23/05/2021	Document implementation
5	07/06/2021	Document implementation
6	13/06/2021	Document first revision
7	18/06/2021	Document revision
8	20/06/2021	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Sergi Mus	sergi.mus@gmail.com
Beatriz Otero	botero@ac.upc.edu

Written by:		Reviewed and approved by:	
Date	13/04/2021	Date	20/06/2021
Name	Sergi Mus	Name	Beatriz Otero
Position	Project Author	Position	Project Supervisor

1 Introduction

Traditionally human analysts manually perform the processing of seismic records by detecting earthquakes within them and picking the wave phase. However, with the increase of recorded traces, due to the constant placement of new stations and instruments, the volume of data is outpacing the processing capabilities of the analysts. Furthermore, high noise conditions due to suboptimal station placement or human activity can lead to the loss of detection of small magnitude events. These facts call for efficient and robust interpretation tools that replace or supplement the labour of trained seismic analysts.

The project aims to use DL to lessen the burden on analysts who manually catalogue all the events on seismic traces. Automatic techniques already exist; some of them rely on DL, while others do not. Unfortunately, these automatic techniques fail to perform on challenging datasets with low signal to noise ratios or correlated noise (usually caused by human activity). One of those datasets is our dataset of the Venezuelan region, where due to accessibility reasons, the stations are not placed in ideal locations, and the seismic noise due to excavations is high. Another downside of our dataset is that the data is extracted from an actual sensor, meaning that it is not synthetic or generated by a scaled model. Only a tiny portion of the bibliography covers this kind of data, most of it preferring to focus on more controlled environments. This environment results in most of the tools being trained with synthetic data, which does not always translate to actual cases.

The project's main goal can be summed up as creating an optimal Neural Network to detect seismological event arrivals using DL methods.

1.1 Requirements

The requirements of the project exist in order to fulfil the stated purpose. The first requirement is to implement and train a ANN capable of detecting events within a trace. More specifically, the ANN should be able to classify a non-fixed size trace within two categories. The first contains all the traces with at least one event in them, while the second only contains noise samples. The second requirement comes as an extension on the first, which consists of locating the events within a trace. With the ability to locate the event, the picking job is fully automated and offloaded from the analysts.

Additional requirements appear with the attempt of making the created tools upgradable. Upgradable means that any new progress in the field of DL should be able to be integrated into our tools. For this reason, the software should allow different ANN architectures to be tested without any significant changes.

Another requirement is that adding new data to the training set should be straightforward for the user. It will only be possible if new data can be preprocessed independently and the software allows to fuse datasets. If this requirement is fulfilled, it will allow the user to add new events, which are constantly happening, with the training set, improving the ANN.

Finally, the last requirement is the need for the software to provide an interface that allows the user to take advantage of the features and perform their experiments without any deep knowledge of the field.

1.2 Methods and procedures

The starting point of this work is the survey written by Rojas et al. [2] where the theoretical basis for the problem was settled. The article was written in collaboration with the advisors to determine the latest ANN applications to the automatic interpretation of seismic data, focusing on earthquake detection. Here, we determined that the ConvNetQuake written by Perol et al. [3] would serve as the guide for our methodology.

However, it is essential to note that the mentioned paper works with synthetic data, making most of the conclusions not applicable to our problem where the data does not meet the requirement of the ideal condition. Nevertheless, the methodology and approach can be transferred to our problem.

The applied methodology in our project consists of using the ANN as an event detector within a trace. Unfortunately, our dataset is formed by continuous unlabelled traces too long to be fed to the ANN. For this reason, the first step in our methodology consists of preprocessing the data to make it suitable for DL. This transformation mainly converts the detection problem into a classification of slices of traces.

Once the data is preprocessed into a suitable format, we can start with the ANN. The first approach is to use the classical FFNN.

Regarding the implementation, the theory for ANN is presented and explained in this document in section 3.2. Instead of starting from scratch, the Tensorflow implementation was used for the experiments instead. The main reason behind this decision is the efficiency of Tensorflow's implementation, more concretely, the GPU acceleration for convolutional operations, which lowers execution times significantly.

After obtaining positive results and testing the validity of our methodology with FFNN, we will attempt to improve the results with CNN. The prediction accuracy for our particular problem with CNN is shown to be better, as shown in ConvNetQuake[3].

One of the main drawbacks of the approach is that the event arrival time is not estimated. Therefore, we will attempt to develop an algorithm that improves our detection uncertainty window by using our neural networks.

Finally, once the system can be tested, we will prove its flexibility and show a novel approach by integrating meta-learning. This novel approach will enable the solution to be better suited to the problem. For example, one of the problems previously presented was the constant addition of new stations and sensors. Meta-learning will allow the network to rapidly adapt to these new environments with tiny datasets (less than 100 samples).

1.3 Work plan

The work plan is split into five work packages. The first one focuses on preparing the dataset to be used. The following two work packages test two different DL architectures.

The fourth work package attempts to test a practical application of the Neural Networks. The final work package tests a new approach using meta-learning. The GANTT diagram for all the work packages can be seen in Figure 1.

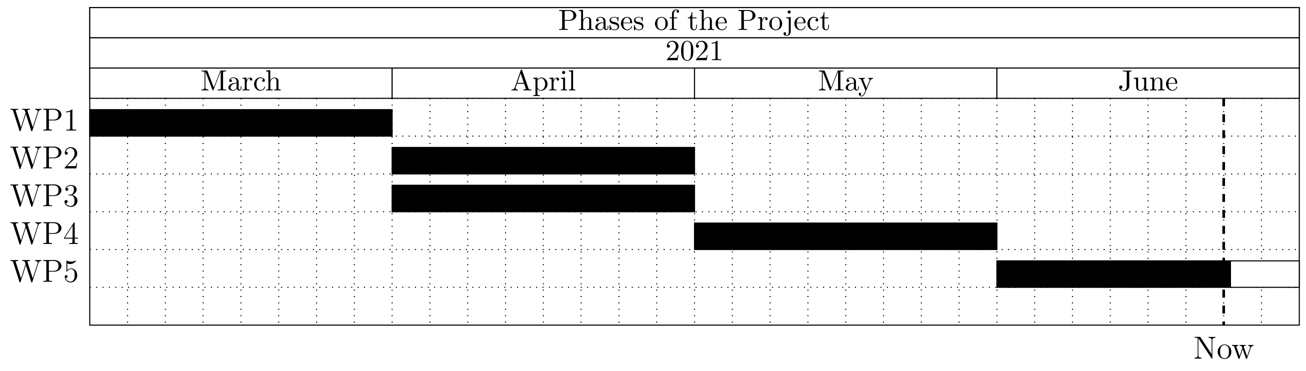


Figure 1: Gantt diagram of the project

The project documentation is not depicted in the Gantt diagram because it was developed continuously during each work package's run-time. Furthermore, a task decomposition for each work package can be found in Figure 2.

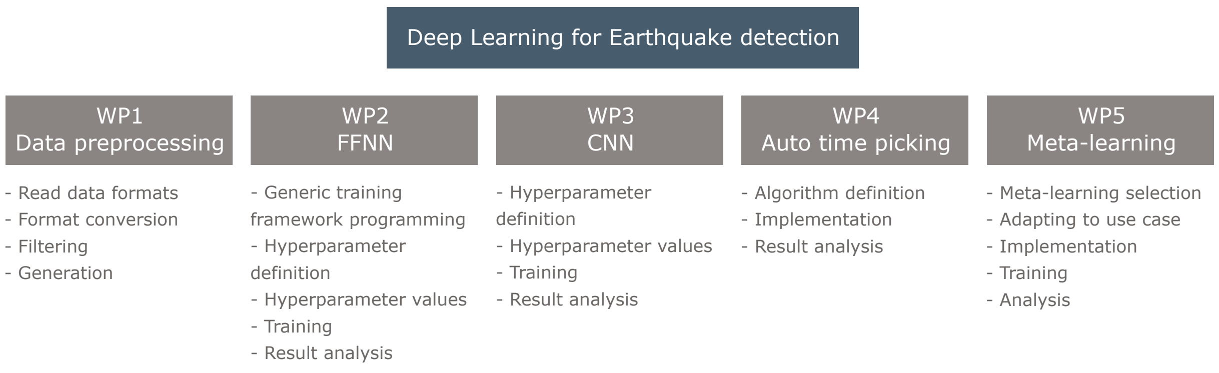


Figure 2: Work Breakdown Structure

It is important to note that initially, the second and third work packages were sequential. After realizing that the second task was going to extend into the third work package period, we decided to start the third simultaneously to meet the deadlines. Another deviation from the initial planning is the addition of the fifth work package due that it is a novelty never used in any of the papers studied in the survey.

2 State of the art

This section will focus on the current state of the technology used for earthquake location and the more relevant DL papers. Furthermore, it will attempt to establish the basic knowledge necessary for the understanding of the following sections.

2.1 Seismology

The consequences of high magnitude earthquake can be ground shaking, surface ruptures, landslides, liquefaction and Tsunamis, which cause loss of human life and significant economic damage. Although these high magnitude events are infrequent, low magnitude events can be used to extrapolate greater magnitude events and perform hazard assessment [4].

To study low magnitude events, seismologists focus on P-wave detection. P-waves is the abbreviation for Primary-waves or Pressure-waves, which correspond to the first wave that the seismograph receives when an event occurs. Additionally, P-waves stand in opposition to S-waves (Secondary-waves). On the one hand, P-waves travel in the propagation direction. In contrast, the oscillations of S-wave's particles travel perpendicularly to the propagation direction [5]. Figure 3 shows the particle motion directions for both waves.

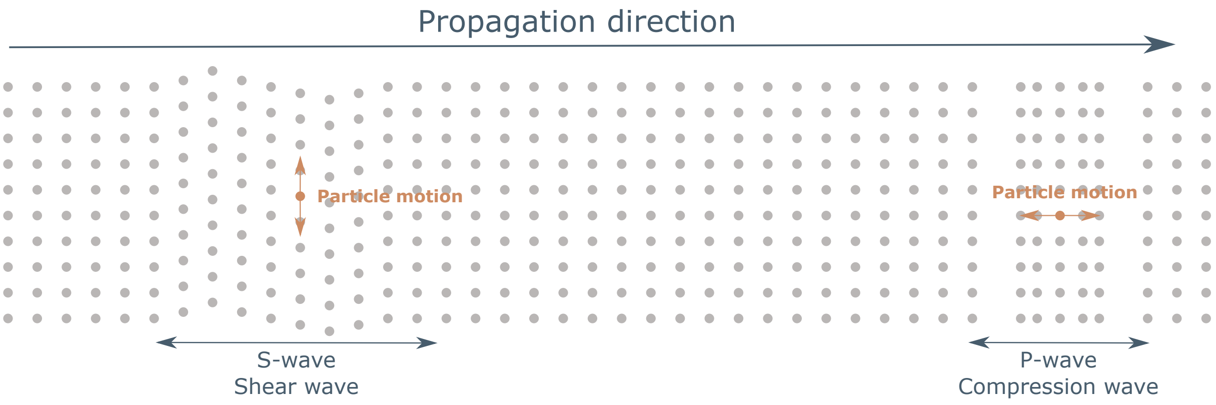


Figure 3: Particle motion directions for P- and S- wave

The problem we attempt to solve in the project is to estimate the arrival of the primary (P) waves in a trace recorded by an instrument at a seismic station. This process is known as time picking and allows subsequent analyses to estimate relevant event information (hypocenter location, focal source mechanism and important spectral properties). This information is later recorded in earthquake catalogues to allow seismologists to create ground motion models.

The most reliable system to date for detecting such waves is manual picking by a trained analyst. Some works, such as M. Leonard [6], describe the manual picking process and their comparison with automatic picking. The article states the importance of precision when detecting events, affirming that only an expert can correctly evaluate each event. Nevertheless, the main concern about manual picking is the clear processing throughput limitations. With the constant increase of seismic stations and sensors, the processing capabilities of the analysts are not enough to address all the existing data [7].

Hence, some classical algorithms for seismic detection exist to address the vast amount of data. One of these algorithms is the Short Time Average/Long Time Average, better known by its abbreviation STA/LTA. The algorithm performs the signal average periodically, dividing a small window of data averaged by a longer one. If the data contains an event, the value of the quotient increases. When the value surpasses a certain defined threshold, the event is detected. Some works such as Kumar et al. [8] study the algorithm's capabilities in seismic alert systems.

However, the STA/LTA method is challenging because the window size parameters depend on the seismic region. More importantly, a fair value of the threshold parameter is difficult to find. The main reason for this difficulty is that the event magnitude is inversely proportional to their frequency, making low-magnitude, frequent earthquakes easily missed when the threshold value is high. Moreover, if the threshold value is defined too low, the system will be very sensible to seismic noise, for example, produced by oil extraction.

Apart from the one depicted above, other algorithms such as Z-Detect [9], Baer picker [10] and AR picker [11] are used for time picking. The analysis of these classical algorithms is beyond this project's scope; however, numerous analysts have tested these algorithms with similar data as in this project, achieving underwhelming results.

2.2 Machine Learning Algorithms and Deep Neural Networks

ML is used in various fields and disciplines. We will focus on its applications to seismology or signal processing to state the relevant applications to this project.

The theory behind earthquakes is very complex and not fully developed. For this reason, Machine Learning can provide innovative insight. Therefore, many articles in the field of Machine Learning attempt to tackle different problems within geophysics [12] [13]. The vast majority of the articles focus on earthquake detection and phase picking (another problem in the geophysics that will not be analysed in this project), benefiting from the ability of Machine Learning algorithms to classify and regress data [14] [15].

Within Machine Learning the project focuses in DL. The classic ANN within DL are FFNN. They have been used with success in Dai and MacBeth [16], which uses single-component input seismograms to obtain a prediction accuracy of 80%. We will attempt to replicate these results in section 3.2, considering our modified preprocessing, which should allow obtaining better results.

Articles like Enescu [17] propose a bandpass filter for more accuracy in the P-wave arrival time prediction. To benefit from this article's contribution a filtering step is added to our preprocessing.

Other papers like Wang et al. [18] propose using trace features as input of the Neural Networks. They extracting relevant information used by analysts to detect events. The algorithm comes from an extension of STA/LTA, where the inputs are trace averages and some additional waveform parameters. With this approach, less burden is placed on the neural network, but gaining innovative insight from the networks is removed. Furthermore, this approach does not yield satisfying results in high noise environments like ours.

Even if FFNN are the classical approach in DL, CNN are better suited for our problem as proven by the literature. One of the most relevant examples is ConvNetQuake [3]. This article does not only present outstanding detection accuracy but also the epicenter location. However, the location performed is limited to classification into different event clusters. In our case, the location would fall outside the scope of the project. Nevertheless, most of the ideas of the paper are used in order to obtain similar results.

Recently, there has been an improvement in DL that allows for networks previously trained in generic environments to specialize in more concrete tasks. This approach has never been used in seismology to the best of the authors' knowledge. One of the flagship articles on the matter and the one used as inspiration for the project is MAML [1]. The paper uses meta-learning for parameter initialization to great success in a generic methodology that facilitates our particular application usage. This methodology has, however, never been adapted to the field of seismology. We will describe our attempt to use these advances in the seismological sector in Section 3.5.

3 Project development

In this chapter we will include all relevant methods(and their development) that were used during the project process. We will also present the experiments that were performed, and analyse the obtained results from those experiments.

3.1 Data preprocessing

This section will focus on the work conducted during WP1, where we took the raw data provided by FUNVISIS and transformed it into a suitable format for Deep Learning. The preprocessing step is essential in any project working in Neural Networks that wants to achieve competitive results.

3.1.1 Catalog parsing

One of the components of the dataset is a seismological catalogue generated by analysts. A catalogue is one or more files containing a collection of events, alongside details about those events like the epicentre location, the magnitude, phase and many others. The most important part of the catalogues are the picks attached to each event; those picks specify each event's station and time of arrival to the station.

The catalogue is in NORDIC file format [19], a catalogue file format generated by SEISAN [20], a widely used seismological analysis software. It has a nested structure where all the picks are included inside of events. Meaning, that all the picks of the same event are associated with the event as seen in Listing 1. Since the aim is to classify traces, the NORDIC format is not ideal for performing station and time based searching. For this reason, we convert it into a simple table that also drops event and pick details irrelecant to the problem. The only data apart from the pick time and stations preserved is the magnitude of the event. If neural networks underperformed, we could limit the problem to only events over a certain magnitude. Finally, the table is saved into a CSV file [21] to perform the following steps of preprocessing.

Listing 1 Initial lines of the catalog in NORDIC file format

```
2017  1 2 0607 47.7 L  10.418 -69.452  5.0F FUN  8 0.8 2.2WFUN             1
2017-01-02-0607-00S.MAN___174                                               6
SIQV HZ EP         6 7 54.81                            97   -1.1010 46.6 303
CURV HZ EP         6 8  0.19                            65    0.1210 71.6 231
BENV HZ EP         6 8 20.21                            43    0.2110  209 104
BAUV HZ EP         6 8 22.23                            43    0.4210  225 136
SOCV HZ EP         6 8 28.59                            43   -0.1410  282 213

2017  1 2 2137 57.4 L   9.958 -71.914 35.3  FUN  3 0.3 2.6WFUN             1
2017-01-02-2137-00S.MAN___174                                               6
CURV HZ EP        2138 27.59                            90    0.4310  214  88
SOCV HZ EP        2138 27.87                            90    0.2310  219 148

2017  1 3 0504 21.8 L  10.366 -69.724  5.0F FUN  9 0.9 2.5WFUN             1
2017-01-03-0503-00S.MAN___174                                               6
SIQV HZ EP      D  5 4 26.80                            99   -0.8110 32.6 344
CURV HZ EP         5 4 28.33                            97   -1.7610 46.9 214
BENV HZ EP         5 4 58.34                            43    0.8610  237 101
BAUV HZ EP         5 4 58.54                            43    0.4710  243 130
SOCV HZ EP         5 5  0.10                            43   -0.3010  262 208
```

3.1.2 Stream simplification

The other component of the dataset provided is the recording of the seismographs. The instrument data is provided in MiniSEED format, a binary format that stores the instrument displacement sample information for the North, East and Z components. It contains some metadata about the instrument and the recorded samples like the station where the instrument is located, the sampling rate, the start time of recording in UTC and the end time of the recording. The samples are stored in a stream structure that aggregates the traces of the instrument for each component. Generally, a stream is composed of three traces, each one representing a different component. An example of a trace representing the Z component of a stream can be seen in Figure 4a.

In our case, the streams provided come in with 3 component traces. However, analysts, to perform a suitable time pick for P-wave, will only use the Z component. Since we want to use ANN, reducing the data inputted by 66% allows us to decrease the number of parameters in the first layer, allowing deeper networks with the same parameter count. For this reason, only the Z components are kept, whilst the other components are discarded. Furthermore, due to the rarity of events, the dataset is pretty imbalanced. Hence, we discard all the traces not containing an event (negative traces). We can extract enough negative samples from the positive traces. It accelerates the preprocessing due to the reduced amount of traces to process.

3.1.3 Trace shaping

The traces can be filtered in order to simplify the ANN learning. Usually, before filtering, the instrument's response is subtracted from the trace to correct possible errors (instrument zeroing or instrument induced noise). However, we do not have such an instrument's response. In place of it, we can apply a bandpass filter between the frequencies of 0.5Hz and 10Hz. This filter will not only eliminate most of the instrument's response but will also reduce the power of the noise (example in Figure 4b). The main justification behind this filter can be found in [17].

In order to have faster ANN convergence, we will divide all the trace samples by the absolute maximum to be in a range between $[-1, 1]$ (example in Figure 4c).

3.1.4 Trace slicing

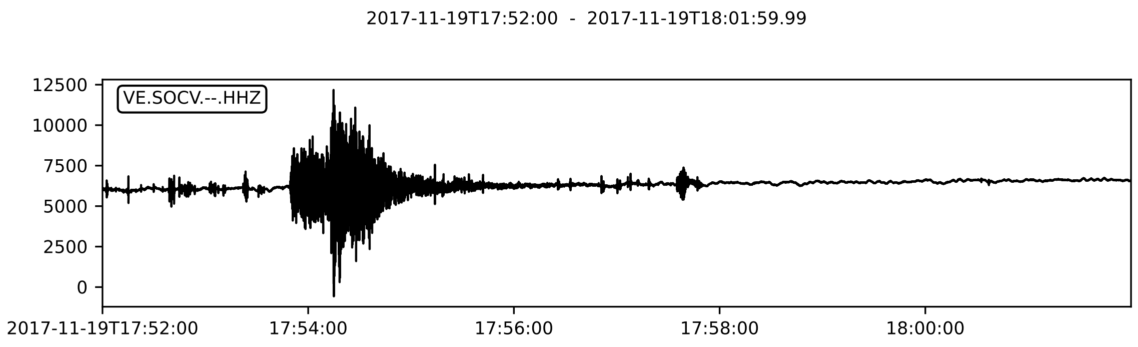
The final step on the preprocessing consists of slicing the traces into regular-sized windows, dividing the long traces into multiple shorter traces, some of them containing an event (positive window) others not containing it (negative window). Since event frequency is low, we focused on events when slicing and generating the negative slices from the remaining trace.

One crucial parameter when slicing the traces is the size of each slice. This parameter balances the Neural Network having context and having too much information to extract meaningful conclusions. With a large slice window size, the neural network can obtain more information before and after the arrival, allowing it to make better decisions. In contrast, with a smaller window, the number of parameters is lower, which allows us to experiment with more complex networks before they overfit. The size used to slice is 30 seconds or 3001 samples, which is close to the window of 5000 used by ConvNetQuake [3]. Different windows were tested with the dataset from 10 seconds to 100 seconds in 10 second increments. The best performance and stability was always with the 30 second windows.

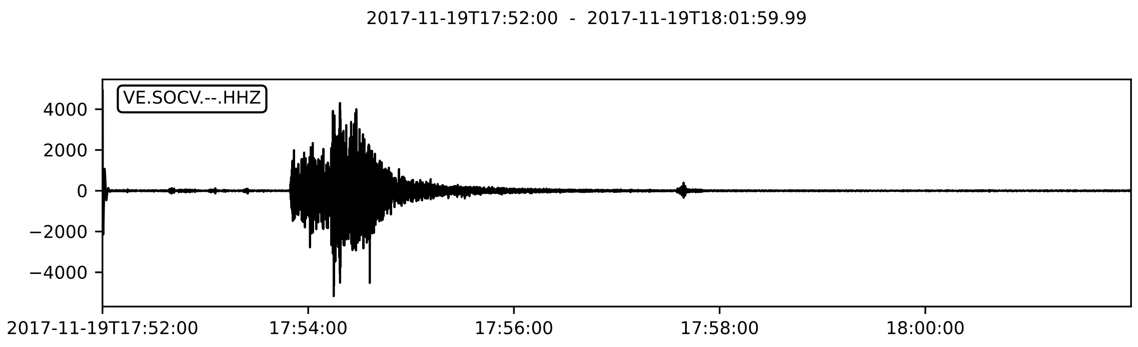
From each trace, we first find all the events in it denoted with a vertical blue line in Figure 5. The most straightforward approach would be to start slicing from the start of the trace and take the slice with the event (denoted in green) as seen in Figure 5a. This approach has two main problems. Firstly, the location of the event within the slice can differ from one slice to another. This alignment is a problem for classical perceptron based networks which are very sensitive to translation, but not for filter-based Neural Networks, which have been proven to be agnostic to translation. Secondly, the event can be located in the last samples of the slice, which leaves most of the slice containing noise (see Figure 5b). Hence, this approach was rejected.

Our solution to this problem is to align the windows according to the events. More specifically, we chose to align the slices with the 25% of the window because it is a good compromise between having most of the window be eventful signal and having the context of the noise before the arrival (see Figure 5c). A smaller and bigger alignment were tested with worse results. This approach gives us the slicing in Figure 5d.

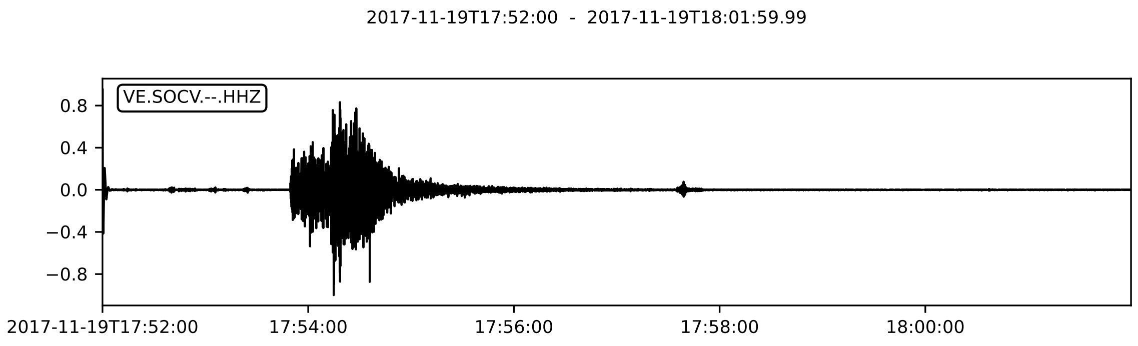
A problem arising from this alignment comes with the events placed at the end of a trace. If the slice end is greater than the end of the original trace, we cannot use the resulting window because it is smaller than the desired window size. In that case, we extend the original trace with zero padding to allow the slice to fit. This padding could be a problem if many samples needed to be added because it would be a non-real signal, but in our dataset, the maximum number of samples added were ten.



(a) Raw trace



(b) Filtered trace

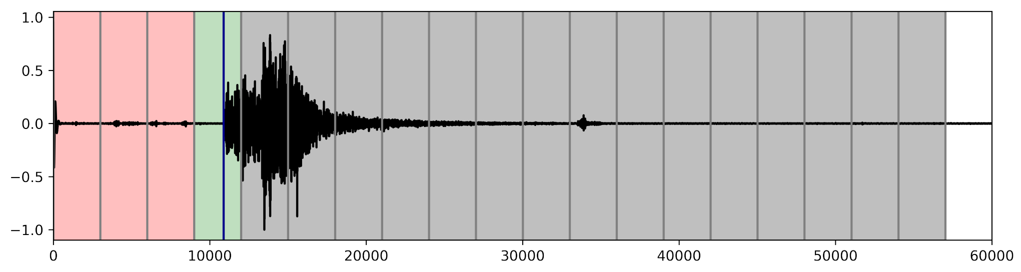


(c) Normalized trace

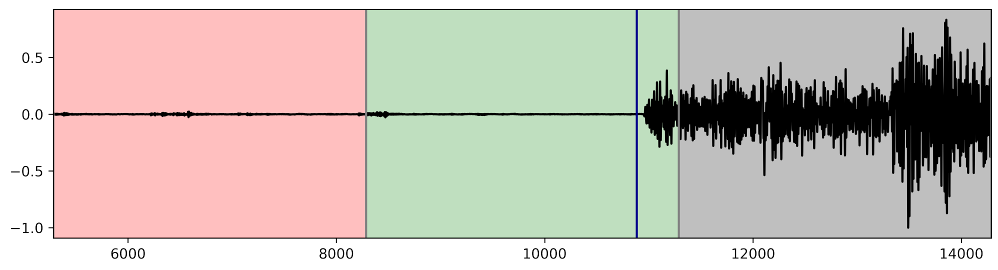
Figure 4: Trace preprocessing example

To extract noise slices from the signal, no slice after an event can be used. Since we only have the event arrivals in our catalogue, we do not have the event end times. For this reason, the slices after the positive window could still contain an event and can not be used as negative windows. Thus, we can only use the samples before the first event in a trace. These discarded windows do not suppose a problem because there is a significant imbalance between event and noise windows favouring the negative windows. The algorithm we followed to get the noise slices was: to take the start of the first event slice, and slice backwards until we reach the start of the trace.

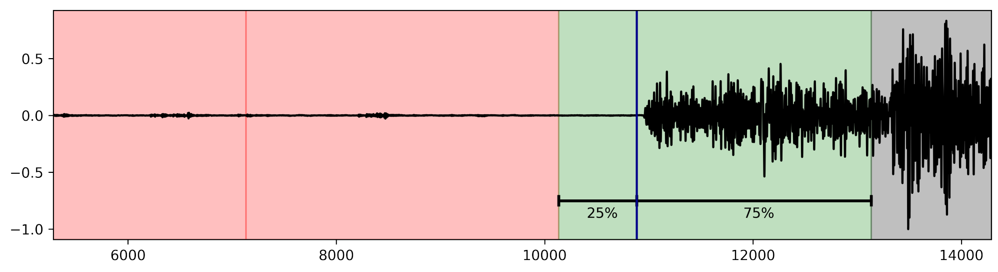
Overall, this section has depicted a way to transform the dataset features into a format that can be directly digested by the most common neural network architectures. In the following section the classical FFNN and CNN architectures are explained.



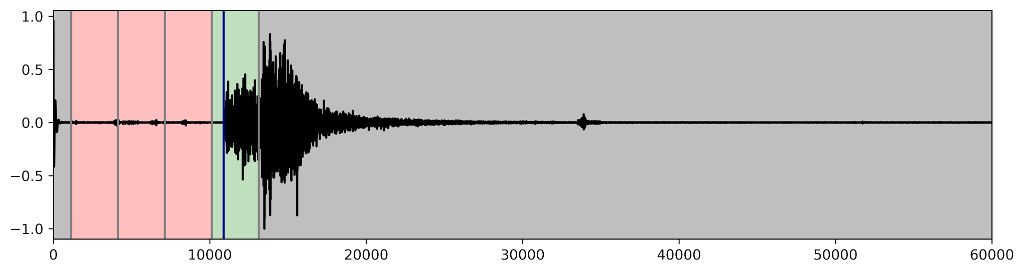
(a) Naive approach



(b) Naive approach problem



(c) Strict alignment



(d) Aligned approach

Figure 5: Trace slicing example

3.2 Feed Forward Neural Networks

The simplest form of FFNN is a SLP which maps a series of inputs $\boldsymbol{x}$ to some outputs $\boldsymbol{y}$ by multiplying the inputs by a matrix of parameters called weights $\boldsymbol{W}$ and adding to the result a vector of parameters called biases $\boldsymbol{b}$ resulting in Equation 1.

$$\boldsymbol{y} = \boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b} \tag{1}$$

We want our SLP to be able to accurately predict $\boldsymbol{y}$ for a given $\boldsymbol{x}$. In order to do so, we have to fine tune both weights and biases called together trainable parameters ($\boldsymbol{\theta}$). To train those parameters we need some examples for the network to be able to learn from. The different examples will be passed through the network. Then, the predictions given by the network will be compared to the expected results by using a loss function. Finally, $\boldsymbol{\theta}$ will be updated following a gradient decent algorithm.

The described methodology has however a major flaw, it can only model linear relationships between the input and output. For this reason, a function, called activation function, breaks the linearity between the inputs and outputs. The modified model is described in Equation 2 where f is the activation function.

$$\boldsymbol{y} = f(\boldsymbol{W} \cdot \boldsymbol{x}) + \boldsymbol{b} \tag{2}$$

A MLP extends the previously described idea by adding one or more layers between the input and the output, called hidden layers. In Equation 3 we can see the resulting operation for a MLP with one hidden layer.

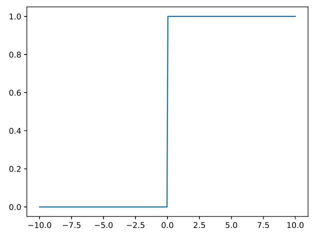
$$\boldsymbol{y} = f_2(\boldsymbol{W_2} \cdot [f_1(\boldsymbol{W_1} \cdot \boldsymbol{x}) + \boldsymbol{b_1}]) + \boldsymbol{b_2} \tag{3}$$

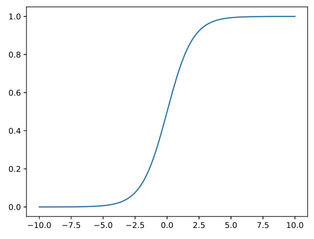
Previously, a loss function was mentioned. In our case, we use Binary Crossentropy, ideal for binary classification problems like ours. In Equation 4 we can see how to calculate Binary Crossentropy for a given prediction by the network y_p and the expected value y_t.

$$\mathcal{L}(y_t, y_p) = y_t \cdot \log(y_p) + (1 - y_t) \cdot \log(1 - y_p) \tag{4}$$

Another point mentioned previously is the usage of gradient decent algorithm for updating $\boldsymbol{\theta}$. The most popular, widely used and stable of this algorithms is Stochastic Gradient Descent (SGD) [22]. Furthermore, it is often referred to as the optimizer. Equation 5 describes the calculation used for updating the parameters, where $\boldsymbol{\theta}'$ are the updated parameters, $f_{\boldsymbol{\theta}}$ the application described by the neural network before the update of the parameters and α a hyperparameter known as the learning rate. The α hyperparameter can be fixed or change over time depending on the problem.

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(f_{\boldsymbol{\theta}}) \tag{5}$$


(a) Step


(b) Sigmoid

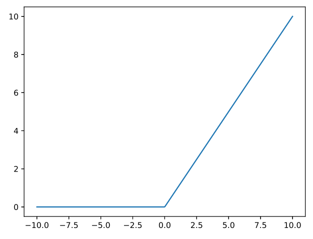

(c) Relu

Figure 6: Activation functions

In some of the use cases we use Adam [23], an upgraded version of SGD, that takes into account momentum and updates the parameter α depending on the previous gradient steps.

3.2.1 Experiments

In our particular case, we will have a MLP that uses the trace samples as inputs. We want it to output 1 if the trace contained an event or 0 otherwise. We can therefore establish that our networks will have 3001 input nodes connected to the first hidden layer, and a single perceptron as the output layer. As we want the output to be 0 or 1 we could use a step function (Figure 6a) as activation function. Nevertheless, for the output layer a sigmoid activation function was selected since it maps values between in the range $[0, 1]$, but has a smoother transition (Figure 6b).

We use Binary Crossentropy as our loss function, because it is the most suitable loss for a binary classification problem. The optimizer used in this case was Adam with a learning rate of 10^{-3}. The learning rate was determined experimentally with a procedure of trial and error.

For the hidden layers, we use a rectified linear (ReLU). This function is $y = x$ for positive values and 0 otherwise. The function can be seen in Figure 6c. However, the output layer uses a Sigmoid activation function.

In this architectures there are several hyperparameters that can be taken into consideration. We chose to focus on the two parameters which are more relevant to detection accuracy. The first hyperparameter is the number of hidden layers that our neural network uses, often called the depth of the network. The second one is the width of the network or the number of perceptrons in each hidden layer. The process of choosing the optimal parameters is a balance between the number of features and the complexity of those.

The number of features is controlled by the layer widths and their complexity is controlled by the depth of the network. Both parameters are bound by the number of trainable parameters. If it goes above a certain value the network will overfit.

For the width of the neural network the hyperparameter was tested with the values 16, 32, 64, 128 and 256. This values allow us to test a wide range of network combinations without generating a high computational cost. Furthermore, the values stop at 256 because the next logical step 512 caused the networks to overfit with very little iterations.

Finally, for the depth hyperparameter the values went from 1 to 4 hidden layers. This parameter was highly bound by the amount of neural networks we would have to train. For example, the network with 4 hidden layers we have a total of 625 networks to train. This gives us a total of 780 networks to train for all hyperparameter combinations.

3.2.2 Results

A summary of the obtained results can be seen in Table 1. This table presents the best networks according to validation accuracy. From it we can extract the following conclusions.

The first conclusion that can be extracted from the results in Table 1 is that most of the best performing networks have 4 layers and the outliers have 3 layers. We can conclude that deeper networks perform better. We can conjecture that the deeper networks are capable of modeling more complex relationships between inputs and outputs and therefore are capable of making better predictions and better generalization. Another reason justifying the great performance of deeper networks is the ReLU activation function that does not fade the back-propagation in deeper networks. Other activation functions like the hyperbolic tangent could be used. However, when we calculate the gradient for values smaller than -2 or larger than 2, the network to not converge. Therefore, we have a dead perceptron that never activates.

The architectures obtaining the best accuracies also have an autoencoder structure, by having a wide first layer that goes into more narrow layers and then expands into a wide layer again. The main idea behind this network architecture is that the information contained within the trace is condensed into its essential features within the narrow layers. The following wider layers use this essential information to extract a meaningful conclusion of the trace containing or not an event. If we turn our attention to the 4 best performing networks shown in Table 1 they follow the described pattern.

To conclude, the alignment of the events within the slice lets us obtain an accuracy of 91.2% close to the one obtained by articles, presented in related works, using CNN.

Ranking according to ACC	Hidden layer count	Units per layer				ACC
		1	2	3	4	
1	4	128	16	16	32	0.912
2	4	128	32	16	128	0.912
3	4	256	32	16	32	0.910
4	4	128	64	16	64	0.909
5	3	256	64	16	-	0.908
6	4	128	32	16	16	0.907
7	4	256	64	32	16	0.907
8	4	256	32	32	64	0.906
9	4	128	32	64	32	0.906
10	4	64	16	16	32	0.906
11	4	64	32	16	32	0.906
12	4	128	64	64	32	0.906
13	4	64	16	16	64	0.906
14	3	128	16	16	-	0.905
15	4	128	32	32	256	0.904

Table 1: Best 15 FFNN according to accuracy

3.3 Convolutional Neural Networks

The main idea behind CNN is to reduce the amount of connections between layers. For this reason, CNN use filters that can process all the inputs without the need to have a parameter for each of these inputs. Therefore, this reduces the number of parameters and reduces overfitting.

CNN are called convolutional because instead of multiplying the parameter matrix a convolution is performed. The parameters of a convolutional layer are the amount of filters or kernels and the size of the filters. The reason behind using 1D CNN is due to the input data being 1D. Having a 1-D convolutional network means that we only define the width of the filter (instead of both width and height). Note, do not confuse the width of the filter with the width of the layer (the amount of filters).

It is common practice to add pooling layers after the convolutional layers. This layers locally aggregate values, the two most popular are Average Pooling that takes the average and Maximum Pooling that takes the maximum of this local values.

3.3.1 Experiments

As we did previously with the fully connected networks we want to test a variety of architectures. For this reason, we are going to test multiple values for each hyperparameter. The three hyperparameters will be: layer width, filter width and network depth.

The parameter of layer width, the number of filters in each layer, will be treated the same way it was tested previously with the values 16, 32, 64, 128 and 256. In the case of CNN the previously mentioned problem of overfitting with the next step of 512 does not apply. However, by increasing the number of layer widths to test, the number of networks to train grows significantly, making infeasible to test all possibilities with the computational resources available, therefore the 512 layers were not tested.

The parameter of filter width is also limited by the increase in the number networks to test. Each filter width tested increases significantly the amount of networks to train. The chosen values were 3 and 12 samples filter width. The reasoning behind this selection is to test a smaller and a bigger filter, however a filter with more than 12 samples causes the networks to be to computationally expensive to train.

Finally, the last parameter is the depth of the networks (the amount of hidden layers). First of all, in opposition to the previous experiment, this parameter does not indicate the amount of convolutional layers but the amount of pair formed by a convolutional layer followed by a maximum pooling layer with width 3. For this reason, a value of 1 for this hyperparameter will make the network have 1 convolutional layer followed by a maximum pooling layer that is connected by the output perceptron. The values used for this hyperparameter were 1, 2 and 3. We would have liked to have the hyperparameter to also have the value 4. However, this increased the amount of networks to test from 1110 to 11110, which is not achievable with the available resources.

3.3.2 Results

As for the CNN, the best accuracies are similar to the ones achieved in the state of the art CNN but with the benefit of having lower parameter counts that allow the network to be trained with a smaller dataset.

Looking at the results in Table 2 we can see that all the networks are as deep as the hyperparameters allow them (3 layers). However, their width rarely takes the maximum allowed value of 256 filters. We can therefore assume that for the problem of event detection network depth is more important than width in order to achieve high accuracy.

Another important point to note is that the structure of the layer sizes in most of the cases (not all) resembles an autoencoder, like in the case of the MLP. For example, the first network goes from a wider layer of 64 filters to a narrower layer with 32 filters and goes back to a wide layer with 128.

The additional hyperparameter unique to CNN, the filter size. This parameter seams to follow a pattern, having a higher width in the first and last layers and coming back to a more narrow value for the central layer. This pattern is consistent with the encoder structure that compresses the information in the most internal layers. It then uses the layer closer to the input to compress the information and the ones closer to the output to decompress the information. Smaller filter implies a smaller count of trainable parameters due that for the same amount of filters the number of values of the filter is reduced.

Ranking according to ACC	Hidden layer count	Filter per layers			Kernel size per layer			ACC
		1	2	3	1	2	3	
1	3	64	32	128	12	3	12	0.982
2	3	64	16	256	12	3	12	0.982
3	3	16	32	256	12	12	12	0.981
4	3	16	16	128	12	3	12	0.981
5	3	256	32	64	12	3	12	0.981
6	3	128	16	128	12	3	12	0.981
7	3	32	16	64	12	3	12	0.981
8	3	64	16	128	12	3	12	0.980
9	3	128	32	64	12	3	12	0.980
10	3	32	32	64	12	3	12	0.980
11	3	32	16	64	12	12	12	0.980
12	3	64	16	128	3	3	12	0.980
13	3	32	64	128	12	3	12	0.980
14	3	64	32	256	12	3	12	0.980
15	3	16	64	64	12	3	12	0.980

Table 2: Best 15 CNN according to accuracy

3.4 Automatic time picking

Automatic time picking consists of automatically locating an event arrival. The goal is to achieve better temporal resolution than the 30 second window.

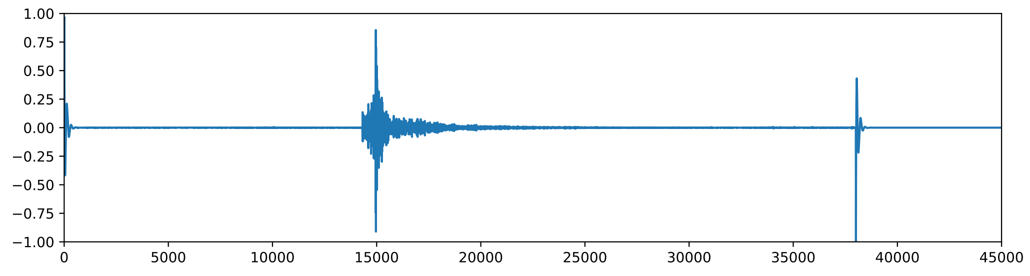
We want to use the previously trained networks to locate within a continuous trace where arrival times are. The main idea comes from using our neural networks as an encoder or filter and from the resulting signal locate the events. Once the events are located in the filtered trace, we can place the event on the input trace at the same location.

We consider the neural network as a function f_θ where θ are the trained parameters. The automatic time picking is performed by continuously applying the neural network to the stream. This procedure gives us a trace with values between 0 and 1 that can be interpreted as the likelihood of the particular window used to compute each value having an event.

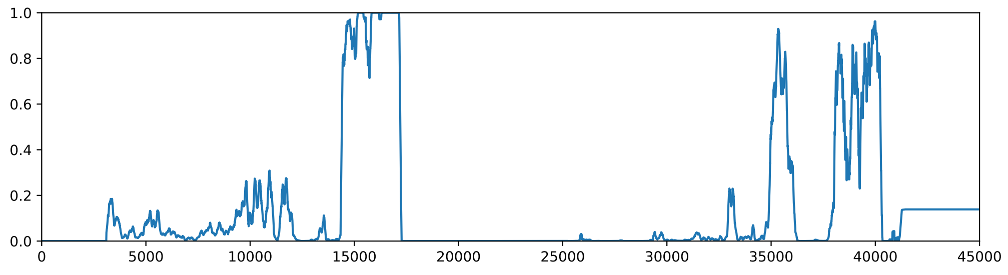
Figure 7 shows an example detailing all the steps in the process. Firstly, we create a set of pointers to the trace data with all the possible offsets for the sliding window. We apply the neural network to each of this created pointers. The equation form of the procedure can be is in Equation 6, where y is the resulting trace and x the seismograph trace preprocessed (see Figure 7a).

$$y[i] = f_\theta(x[i], x[i+1], ..., x[i+3000]) \tag{6}$$

The resulting y is often very noisy with multiple false positives, to reduce these noise



(a) Input trace



(b) Output trace

Figure 7: Auto picking example

we apply a square filter. In the case of Figure 7b, we have applied a square filter of 100 samples.

The outcome of the process can clearly be seen in Figure 8, where we draw the input trace with the color scheme determined by the output.

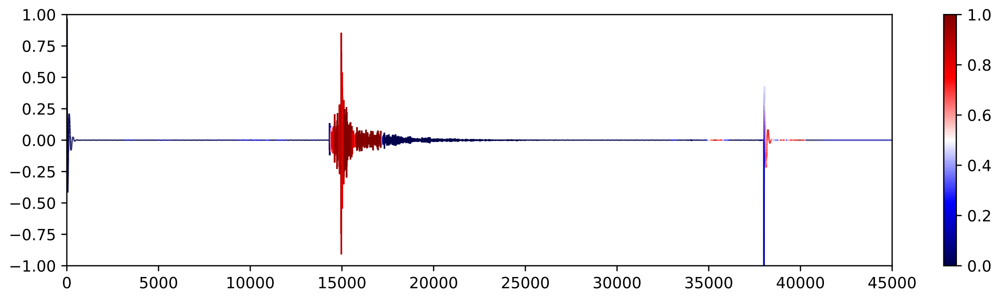


Figure 8: Superposed output to input

3.5 Meta-learning

Once the project has a working approach to address the seismic detection problem with FFNN and CNN, we attempt to transfer the meta-learning workflow to our framework.

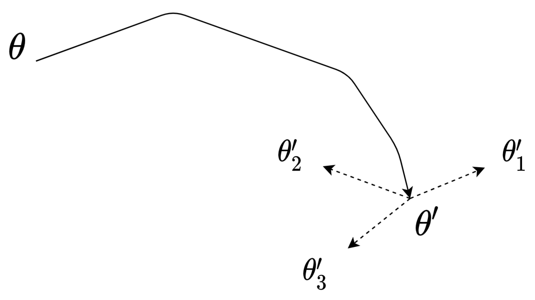


Figure 9: Parameter evolution in meta-learning example [1]

Meta-learning in the case of the MAML [1] article consists in creating a parameter initialization that allows the network to quickly adapt to a set of similar problems. These smaller problems are referred as tasks while the learning of the optimal starting point for the parameters is known as meta-learning. The meta-task is the common knowledge between all the sub-tasks. The example presented in the MAML article[1] is animal recognition. In it, the general problem or meta-task is recognising animals, the specific task is the detection of a specific animal.

A meta-learning approach allows the parameters to learn the generic features of the meta-task but being able to quickly adapt to the case specific sub-tasks. This idea is demonstrated in Figure 9, where the continuous arrow represents the meta-learning of the parameters and the dotted arrows the fine tuning that can be done after meta-learning. The initial parameters begin in θ and during the meta-learning process ideally the values would converge to θ'. Afterwards, they can be adapted and trained to their task specific values θ'_1, θ'_2 and θ'_3 in few iterations. In the MAML example [1], to detect a particular animal category the optimal parameters are the different θ'_i. And, the optimal meta-learning parameters is θ' since adapting the network to a new animal is easily performed. Thus, in our approach we would define the input data as θ, the different values of the stations as θ'_i, and the final training output would be θ'.

Nevertheless, a consequence is that the data used during the meta-learning results is an aggregation of all the sub-task datasets. In our case, each sub-task dataset is the streams from a specific station. This aggregation results in the usage of more complex models without overfitting.

The generic problem we want to solve is P-wave detection, however, commonly, the used approach is to train the neural network on single station traces because the event signature can change between stations. In our case, using a single station would not allow us to use deep learning due to insufficient data and events. However, having multiple stations allows us to apply it into a meta-learning context and use all the available data from each station. The final result achieved will be a meta-learned station generic model that can rapidly adapt to detect events in particular station.

The implementation consists in separating the dataset into single station datasets. Then we take one station for validation and use the others for meta-training. After applying the algorithm presented in [1], we get a 90% meta accuracy. After the meta-learning process, we attempt to fine tune the network with the station separated for validation at the beginning. Once fine tuning is performed during 5 epochs, the accuracy reaches 98%.

Overall, the main benefits of this approach is that the same results from the CNN are achieved with only 5 epochs. This allows the network to quickly be trained in the case a new station is incorporated. Furthermore, by the nature of meta-learning, new stations can be added to the meta-learning dataset seamlessly allowing it to continuously improve. A new station can always be added to the pool used for meta-learning.

4 Budget

The project being a research oriented project does not have any production related costs, however, it has numerous costs associated with the hardware and information required for performing the research.

The main cost of the project comes from the data gathering. For calculating the data gathering costs the industry standard metric for sample price was used. Some datasets have cheaper price per sample however because the used dataset needs specialized analysts to perform event picking we have considered the sample cost reasonable.

From the workforce standpoint we only need a single engineer to carry out the project (the author). Due to the author having previous experience in the field the minimum wage of 10€ is increased to 13€. On the other hand, to solve the issues derived from the geophysical domain, we have outsourced an expert in geophysics (80€ / h).

In order to carry out the project development a 1200€ computer is used, with depreciation period of 5 years and a residual value of 200€ giving us a yearly depreciation cost of 200€.

For execution, this computer is highly inefficient from an energetic standpoint and is sensible to blackouts, the cost to prevent this risks and inefficiencies is mitigated by outsourcing the computing. For the budget, a mean between the costs of similar machines in Amazon Web Services, Google Cloud and Azure was used. Several machines were used in parallel to increase throughput.

Details	**Count**	**Unitary**	**Amount**
Dataset	20000 samples	0.7€ / sample	14000€
Engineer salary	480h	13€ / h	6240€
Seismology consulting	10h	80€ / h	800€
Computer depreciation	4 months	200€ / year	67€
Cloud computing	6720h	0.122€ / h	820€
Total			**21,927 €**

Table 3: Project budget

5 Sustainability

5.1 Social impact

The outgoing impact of the project on a social aspect is straightforward; by improving the cataloguing throughput of the analysts, the project helps build a better understanding of faults and therefore having a better hazard assessment. This cycle will finally reduce the loss of human life during a high magnitude event.

Seismology as a science dramatically benefits from the detection capabilities of seismic events. The increase of seismological stations causes the manipulation of vast volumes of persistent and temporal data. In both cases, the application of algorithms that automate the detection process will ease most of the work of the analysts in wave detection concretely in low magnitude events that are more frequent. This process will create a positive impact on the generation of seismological catalogues used for seismic hazard assessment.

It also helps the analysts who no longer have to spend an extended portion of the time manually picking traces and building more accurate models.

Personally, this project has helped the author deepen his knowledge in Deep Learning and acquire a basic understanding of seismological signal processing and P wave detection.

5.2 Economical impact

The economic losses of a high magnitude event are countless most of them produced from the loss of infrastructure, causing second-hand losses in various economic sectors. However, the economic losses are not limited to the business sector and affect homeowners with the destruction of permanent housing. For these reasons, the prevention of such losses is of capital importance by building a proper risk assessment and building infrastructure and housing, taking into account updated seismological information. This project aims to improve the seismological models and give a better starting point to researchers aiming to work on high-magnitude event prediction systems.

An economic sector highly interested in the improvements provided by this paper is the oil industry. This industry relies on ground excavation for its activity which can results in catastrophic accidents without the proper safety measures. In addition, oil reserves are often situated in seismically active regions.

5.3 Environmental impact

Due to high magnitude event being a natural process, very rarely they are considered environmental hazard. However, if mixed with modern technology, these events can prove to have disastrous environmental consequences, as was proven by the 2011 Fukushima I accident, resulting in the leakage of radiation in the atmosphere, water, and local fauna.

On a smaller scale, high magnitude events can cause fires that can burn down acres of the forest, causing notable negative environmental impact.

As in the two previous cases, by deepening the seismic knowledge and having more accurate seismic models, we can prevent, if not negate, the environmental impact of earthquakes.

Finally, the availability of a complete seismological catalogue with sensor arrays will improve ground velocity models, understanding the detailed underground structures on a small and big scale. All this knowledge will positively impact the creation of more realistic models and their influence on propagating seismic waves. Thus, the response of the ground to high magnitude earthquakes could be simulated through software, especially in highly urbanized areas.

6 Conclusions

This project has proposed a novel preprocessing methodology highly specialized for P-wave detection that consists in having the same alignment of the event within a window. This alignment negates the negative effects of having different offsets of for the event within the window in FFNN without disturbing the performance of CNN.

The previously presented preprocessing allows improved performance of FFNN from around 80% to 91.2%. The simpler architectures can compete with filter based architectures.

However, the state of the art architecture for this problem, CNN, was also tested achieving the same results than flagship articles like ConvNetQuake with a multi-station environment. The top achieved classification accuracy is 98.2%.

Furthermore, we have improved the time resolution of our detection method by using our neural networks as auto-encoders. Another benefit from the proposed algorithm is that allows to perform detection in a similar procedure than used by classical algorithms, allowing a familiar approach to seismologists.

Finally, we have successfully used meta-learning in a multi-station environment. Therefore, proving that even with a multi-station dataset, station specific detection networks can be achieved by the use of meta learning.

The project purpose can therefore considered fulfilled, and all the requirements met.

7 Future Work

The project has proposed a new algorithm for reducing the time uncertainty for locating the pick time. However, an exhaustive study of the algorithm over the entire dataset was not performed. This analysis would allow to define with precision the time error of our picker. The first line of work is to optimise the neural network forward pass to allow lower execution times. Another possibility is to selectively apply the algorithms to the sections suspected to have the event.

In the meta-learning section we showed its possible uses in seismology. However, one of the best features of meta-learning is few-shot learning (allowing it to adapt with only 5 or less samples). This feature could not be taken advantage of due to dataset characteristics (we would need more station variety). Further work could be performed to collect a dataset suitable for this line of work allowing a more adaptable neural network.

References

[1] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In International Conference on Machine Learning, pages 1126–1135. PMLR, 2017.

[2] O. Rojas, B. Otero, L. Alvarado, S. Mus, and R. Tous. Artificial Neural Networks as Emerging Tools for Earthquake Detection. Computación y Sistemas, 23:335 – 350, 06 2019.

[3] M. Denolle, T. Perol, and M. Gharbi. ConvNetQuake: Convolutional Neural Network for Earthquake Detection and Location. In AGU Fall Meeting Abstracts, volume 2017, pages S41D–02, December 2017.

[4] R. Westaway. Extrapolation of populations of small earthquakes to predict consequences of low- probability high impact events: The pohang case study revisited. Geothermics, 92:102035, 2021.

[5] J. Milsom. Field Geophysics. John Wiley and Sons, 2003.

[6] M. Leonard. Comparison of Manual and Automatic Onset Time Picking. Bulletin of the Seismological Society of America, 90(6):1384–1390, 12 2000.

[7] H. Kanamori. Real-time seismology and earthquake damage mitigation. Annual Review of Earth and Planetary Sciences, 33(1):195–214, 2005.

[8] S. Kumar, R. Vig, and P. J. Kapur. Development of earthquake event detection technique based on STA/LTA algorithm for seismic alert system. Geol Soc India, 92:679–686, 2018.

[9] M. Withers, R. Aster, C. Young, J. Beiriger, M. Harris, S. Moore, and J. Trujillo. A comparison of select trigger algorithms for automated global seismic phase and event detection. Bulletin of the Seismological Society of America, 88(1):95–106, 02 1998.

[10] M. Baer and U. Kradolfer. An automatic phase picker for local and teleseismic events. Bulletin of the Seismological Society of America, 77(4):1437–1445, 08 1987.

[11] R. Sleeman and T. Van Eck. Robust automatic P-phase picking: an on-line implementation in the analysis of broadband seismogram recordings. Physics of the Earth and Planetary Interiors, 113(1):265–275, 1999.

[12] Y. Zhao and K. Takano. An artificial neural network approach for broadband seismic phase picking. Bulletin of the Seismological Society of America, 89(3):670–680, 1999.

[13] L. Zhu, E. Liu, and J. H. McClellan. Sparse-promoting full-waveform inversion based on online orthonormal dictionary learning. Geophysics, 82(2):R87–R107, 2017.

[14] T. Tiira. Detecting teleseismic events using artificial neural networks. Computers & Geosciences, 25(8):929–938, 1999.

[15] M. Titos, A. Bueno, L. Garcia, and C. Benitez. A deep neural networks approach to automatic recognition systems for volcano-seismic events. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 11(5):1533–1544, 2018.

[16] H. Dai and C. MacBeth. The application of back-propagation neural network to automatic picking seismic arrivals from single-component recordings. Journal of Geophysical Research: Solid Earth, 102(B7):15105–15113, 1997.

[17] N. Enescu et al. Seismic data processing using nonlinear prediction and neural networks. In IEEE NORSIG Symposium, Espoo, Finland, 1996.

[18] J. Wang and T. Teng. Artificial neural network-based seismic detector. Bulletin of the Seismological Society of America, 85(1):308–319, 1995.

[19] Nordic format documentation. `https://nordb.readthedocs.io/en/latest/nordic_desc.html`. Accessed: 2021-04-21.

[20] Seisan. `http://seisan.info/`.

[21] CSV format standard. `https://tools.ietf.org/html/rfc4180`. Accessed: 2021-04-21.

[22] N. Ketkar. Stochastic Gradient Descent, pages 113–132. Apress, Berkeley, CA, 2017.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.