

N° d'ordre :

THESE

En vue de l'obtention du : **DOCTORAT**

Structure de Recherche : Equipe Informatique, Mathématique Appliquée,
Intelligence Artificielle et Reconnaissance de Formes

Discipline : Sciences de l'ingénieur
Spécialité : Informatique et Intelligence Artificielle

Présentée et soutenue le : 14/12/2021 par:

Youness CHOUBIK

Fully Convolutional Neural Networks for Seismic Time Series Analysis in the Big Data Context

JURY

Bouchta EL FELLAH	PES , Institut Scientifique, UM5, Rabat	Président
Khalid MINAOUI	PES, Faculté des Sciences, UM5, Rabat	Rapporteur / Examineur
Mohamed MASTERE	PES, Institut Scientifique, UM5, Rabat	Rapporteur / Examineur
Mohammed KHALIL	PH, Faculté des Sciences et Techniques, UH2, Mohammedia	Rapporteur / Examineur
Mohammed Majid HIMMI	PES, Faculté des Sciences, UM5, Rabat	Directeur de thèse
Abdelhak MAHMOUDI	PH, Ecole Normale Supérieure, UM5, Rabat	Co-encadrant

Année Universitaire : 2020/2021

Avant-propos

The research work presented in this thesis has been performed in the "Laboratoire d'Informatique Mathématiques appliquées Intelligence Artificielle et Reconnaissance de Formes" (LIMIARF) at the Faculty of Sciences of Mohammed V University, Rabat, Morocco, as part of the "vulnérabilité et gestion des risques naturels dans le rif; - risque sismique - risque de mouvements de terrain - risque d'inondation", funded by National Center for Scientific and Technical Research (CNRST).

First of all, it is with immense gratitude that I recognize all the support and advice of my supervisors Professor Mohamed Majid HIMMI and Professor Abdelhak MAHMOUDI. Their genuinely scientific intuition has made them a source of ideas and passions in science, which inspire and exceptionally enrich my growth as a student, researcher, professor and scientist. I especially want to thank them for giving me a lot of autonomy while providing appropriate advice. I would like to thank Professor Bouchta EL FELLAH (Scientific Institute, Mohammed V University) for having accepted to preside the jury for my dissertation. I am very grateful to my reviewers, Professor Khalid MINAOUI (Faculty of Sciences, Mohammed V University), Professor Mohamed MASTERE (Scientific Institute, Mohammed V University) and Professor Mohammed KHALIL (Faculty of Sciences and Techniques, Hassan II University) for their constructive comments on this thesis. I want to express my deep gratitude for the honor they do me by agreeing to review my work. I am grateful that despite all their activities, they have agreed to review my work.

I thank the wonderful members of LIMIARF Laboratory for the consultations and moral support. LIMIARF laboratory provided me with the support and equipment I needed to carry out and complete my thesis. I would also like to thank my colleagues from the LIMIARF Laboratory for the joyful and pleasant working atmosphere. In addition, I am also very grateful to Professor Mohamed Majid HIMMI for getting involved as a Temporary Assistant Professor, this opportunity provided me with an excellent environment to teach and develop new teaching techniques. More importantly, words alone cannot express the thanks I owe my family for believing and loving me. They have

always filled my life with generous love, unconditional support and prayers. I dedicate this thesis to them.

Dedication

To my father

I miss you more and more every day

To my mother, my brother, my sisters

For your love, your trust, your patience,

I love you.

Youness Choubik



Abstract

Short Term Average to Long Term Average (STA/LTA) is the most commonly used algorithm for earthquake detection, and it involves signal processing using filters in different time frames to identify amplitude outliers and hence seismic phases. Because automated identifications frequently involve crucial errors, this method is generally corrected further by experts in seismic signal analysis. In this thesis, we implemented STA/LTA trigger algorithm using Hadoop MapReduce. This implementation allowed us to find out how effective is Hadoop in this type of tasks as well as to accelerate the detection process by reducing the processing time. We tested our implementation on a seismological dataset of broadband seismic stations implemented in Morocco and compared it with the traditional one.

On the other hand, automatic earthquake detection is widely studied to replace manual detection, however, most of the existing methods are sensitive to seismic noise. Hence, the need for deep learning has become more and more significant to process seismic time series. We proposed an automatic earthquake detection model based on a Fully Convolutional Networks (FCN) classifier. To test the efficiency of our classifier, we applied it to real continuous seismic data from XB Network from Morocco.

Keywords: *Deep Learning, Big Data, Time Series, Earthquake Detection*



Résumé

Short Term Average to Long Term Average (STA/LTA) est l'algorithme le plus couramment utilisé pour la détection des tremblements de terre. Étant donné que les algorithmes de détection automatique impliquent fréquemment des erreurs cruciales. Dans cette thèse, nous avons implémenté l'algorithme de déclenchement STA/LTA en utilisant Hadoop MapReduce. Cette implémentation nous a permis de connaître son efficacité dans ce type de tâches ainsi que d'accélérer le processus de détection en réduisant le temps de traitement. Nous avons testé notre implémentation sur des données sismologiques issue de stations sismiques à large bande installées au Maroc.

D'autre part, la détection automatique des séismes est largement étudiée pour remplacer la détection manuelle. Par conséquent, le besoin d'apprentissage profond est devenu de plus en plus important pour le traitement des séries chronologiques. Nous avons proposé un modèle de détection automatique des tremblements de terre basé sur le classifieur FCN. Pour tester l'efficacité de notre classificateur, nous l'avons appliqué à des données sismiques continues réelles du réseau XB du Maroc et comparé les résultats à notre catalogue.

Mots clé : *Apprentissage Profond, Donnée Massive, Séries Chronologiques, Détection de Séisme*



Contents

Abstract	5
Résumé	6
Contents	8
List of Figures	12
List of Tables	15
1 Introduction	17
2 Background	22
2.1 Introduction	22
2.2 Deep Learning	23
2.2.1 Supervised and unsupervised learning	23
2.2.2 Artificial Neuron	24
2.2.3 Activation Function	25

2.2.4	Artificial Neural Networks	26
2.2.5	Training a Neural Network	28
	Forward propagation	28
	Backward propagation	29
	Gradient-Based Learning	30
2.2.6	Parameters and Hyperparameters of the network	32
2.2.7	Convolutional Neural Network	33
	Convolution layer	34
	Fully connected layer	35
	pooling layer	36
2.2.8	Generative Adversarial Networks	37
2.3	Big Data Tools	40
2.3.1	Apache Hadoop	41
	Hadoop 1.x	42
	Hadoop 2.x	45
	Hadoop 3.x	47
2.3.2	Apache Spark	47
	Spark Core	48
	Spark SQL	48
	Spark Streaming	49
	MLlib	49
	GraphX	49
2.3.3	Cloud Computing	49
	Public cloud	50
	Private cloud	50
	Hybrid cloud	51
	Infrastructure As A Service	51

PPlatform As A Service	52
Software As A Service	52
virtualization	52
Google Cloud Platform	53
Amazon Web Services	54
Microsoft Azure	54
2.4 Conclusion	54
3 Big Data tools for earthquake detection	55
3.1 Introduction	55
3.2 Seismology overview	56
3.2.1 Structure of the Earth	57
3.2.2 Seismic waves	63
3.2.3 Earthquake detection	65
3.2.4 Measuring instruments	68
3.2.5 Seismic data structure	71
3.3 Seismic detection using Hadoop MapReduce	72
3.3.1 Short Term Average to Long Term Average	73
3.3.2 Method	74
3.3.3 Results and Discussion	80
3.4 Conclusion	81
4 Deep Learning for Earthquake Detection	83
4.1 Introduction	83
4.2 Earthquake detection using Convolution Neural Networks	84
4.2.1 Data description	86
4.2.2 Training with the Fully Convolutional Networks	87
4.2.3 Results and discussions	91

CONTENTS	11
4.3 Seismic data Augmentation using Deep Learning	95
4.3.1 Generative Adversarial Networks	98
4.3.2 Earthquake dataset	99
4.3.3 Model Training	100
4.3.4 Results and discussion	101
4.4 Conclusion	104
5 Conclusion and perspectives	105
Bibliography	107



List of Figures

1	Biological Neuron	24
2	Artificial Neuron	24
3	An Artificial Neural Network	27
4	Gradient Descent	31
5	Convolutional Neural Network Architecture	33
6	A Convolution example	34
7	An Artificial Neural Network	36
8	A max pooling example	37
9	A Generative Adversarial Networks Architecture	38
10	Hadoop Architecture	43
11	hdfs Architecture	44
12	MapReduce Architecture	45
13	Spark Architecture	48
14	The internal composition of the earth	58
15	The earth's tectonic plates	59

16	Normal fault (a) and reverse fault (b)	61
17	Left lateral (a) and right lateral strike-slip (b)	62
18	Focal mechanism and related fault types	62
19	P-waves (a) and S-waves (b) in an elastic 1D element	64
20	Rayleigh waves (a) and Love waves (b)	65
21	The seismic waves origin	67
22	A classic seismograph	69
23	STA/LTA algorithm	74
24	Size of Avro files of the 14 PM Stations	75
25	XB seismic network	76
26	Number of seismic events detected per Station	81
27	Distribution of seismic stations used to record earthquakes (a) and noise (b).	87
28	XB seismic network installed in both Morocco and Spain, it contains 93 seismic stations, from which 19 stations are used in this study and are colored in red.	88
29	The FCN architecture consists of 4 convolutional layers followed by Batch- Normalization (BN) then a Rectified Linear Units (ReLU) activation func- tion, finally global-pooling and softmax layers.	89
30	Standardization effect for the same two signals, in top an earthquake sig- nal while in bottom a noise one. Figure (a) shows raw signals, in (b) both signals are standardized per feature, (c) shows the sample-standardization version for both signals.	93
31	Model loss per batch size.	93
32	Training and validation subsets accuracies per batch size. Large batches tend to have lower accuracies for both subsets.	94

33	The mean accuracy per batch size. The best accuracies in train/validation/test subsets are obtained by using a 16 batch size.	94
34	representation of GANs architecture	98
35	class samples from the training set	100
36	GAN architecture	101
37	FCN architecture	102
38	GANs generated sample compared to real sample	102
39	Experiments on batch size and data distribution	103



List of Tables

3.1 Hadoop cluster's configuration	76
4.1 Train/Validation/Test subsets distribution used in our study.	89
4.2 The accuracy for different feature normalization methods.	92
4.3 The accuracies for different per-sample normalization methods.	93

Introduction

Earthquakes are one of the most serious natural disasters in modern urbanized zones, where buildings are frequently tall and crowded. Because the devastating effects that big earthquakes may have are well recognized, monitoring and detecting earthquakes is a critical task. It is often carried out by national or worldwide seismological networks, which spread seismometers across specific regions of interest and transmit real-time ground movement data to a central base for processing.

Manual detection of earthquakes is performed by experts in seismic signal analysis by identifying mainly the arrival time of the primary seismic wave (P phase) and the secondary wave (S phase), then the localization of the earthquake is carried out by using arrival times in at least three seismological stations. Nowadays, software for earthquake detection performs the automatic detection and identification of these phases (mainly the P phase, which is easier to distinguish because it is the first to arrive). The most widely used methods have significant flaws that can lead to earthquakes being missed and false alarms being generated. Manual detection is an optimal and reliable method for identifying earthquake arrival time, however,

with the large and growing datasets of seismic recordings, manually performing the task of earthquake detection is not only difficult but also time-consuming.

To accomplish reliable automatic detection, a broad range of conventional algorithms have been developed. Among all these methods, Short Term Average to Long Term Average (STA/LTA) [1] is the most commonly used algorithm for this task, and it involves signal processing using filters in different time frames to identify amplitude outliers and hence seismic phases. Because automated identification frequently involve crucial errors, this method is generally corrected further by experts in seismic signal analysis [2].

A big challenge for earthquake detection over the coming decade remains the huge quantity of seismic data in the world which still grow more and more [3]. In our first contribution [4], we applied STA/LTA trigger algorithm to the three-component records of the XB seismic network installed in Morocco between 2009 and 2013. Due to the large amounts of seismic data that we worked on, we used Hadoop MapReduce to benefit from its power of processing data in a distributed fashion.

While traditional automatic detection methods are valuable, their accuracy is lower than that of manual detection, making them ineffective in seismic detection. Traditional automatic detection may not be accurate enough, especially when dealing with noisy data. The most significant issue of STA/LTA algorithm is its inability to identify earthquakes with a low signal-to-noise ratio and the ease with which it might create false detection because of the signal noise [5]. Another issue with this method is that the filters used must be adjusted on a regular basis because of variations in seismological station noise, which makes this method a time consuming task. Thus, the development of new deep learning techniques for handling and learning from large data sets is required.

Deep learning algorithms, trained with manually identified earthquakes [6], are

used to detect earthquakes. It's thought that with training deep learning algorithms using seismic time series data, we might identify earthquakes with better accuracy. Earthquake detection has recently attracted a lot of interest due to the usage of deep learning algorithms along with the availability of large volume of seismic data. A large number of articles in this field of study were published, including Convolutional Neural Networks (CNN) for earthquake detection [7; 8; 9; 10], CNN for identifying the arrival of the P wave and its polarity [11], and CNN for earthquake detection and identification of arrival times [12].

In general, research on earthquake detection issue focuses on addressing the challenge as a classification issue, with neural networks being the most commonly employed approach [8]. This might be because of the fact that neural networks have demonstrated that they can handle time series classification issues as well as or better than human specialists in other domains. In our second contribution [13], we developed a Fully Convolutional Networks (FCN) architecture for earthquake detection and conducted multiple experiments using a balanced subset of Stanford EArthquake Dataset (STEAD) to train and validate our model. Each sample from the subset is re-sampled then normalized. We investigated different and widely used, normalization methods. To test the efficiency of our classifier, we applied it to real continuous seismic data from Morocco's XB Network and compared the results to an earthquakes catalog of Morocco region, obtained from the National Geographic Institute (IGN) of Spain.

In our third contribution [14], we developed a data augmentation technique to enhance the performance of the earthquake dataset classification problem. We used Generative Adversarial Networks (GANs) to augment the imbalanced Earthquakes dataset, which is obtained from the University of California Riverside (UCR) Time Series Classification Archive [15], which included less positive classes than negative, then we concatenated the training set with the generated data from GANs to create

our new training set that will be used for classification, while the test set remains untouched. We developed a Fully Convolutional Networks architecture as a classifier and conducted multiple experiments on multiple parameters such as batch size and data distribution. The results show that the best accuracy on the test set is achieved when the training set is balanced. Since our best accuracy reached 77.69%, we get the best accuracy for the earthquakes dataset compared to other works, where the same dataset has been used [16; 17].

This thesis consists of five chapters, each of which is briefly described as follows:

In chapter 2, deep learning is introduced and the major concepts are presented. We first begin by defining some terminologies generally used in deep learning. Also, we present the learning process including the forward and backward propagation. Then we define some parameters needed for the learning process. Finally, we present the two deep learning algorithms used in this thesis, Convolutional Neural Networks and Generative Adversarial Networks. Also, we introduces some big data concepts and technologies, such as Apache Hadoop and Spark. Then, we present an overview of cloud computing, its mechanism, along with its different services.

In chapter 3, the most important principles in seismology were briefly covered. Then, we show the usability of Hadoop MapReduce for seismic detection, particularly using STA/LTA algorithm. We compare MapReduce implementation performance with that of a traditional implementation. This chapter is based on the research previously published in the following paper:

- Choubik, Youness & Mahmoudi, Abdelhak & Himmi, Mohammed & El Moud-nib, Lahcen. (2020). STA/LTA trigger algorithm implementation on a seismological dataset using Hadoop MapReduce. IAES International Journal of Artificial Intelligence (IJ-AI). 9. 269. 10.11591/ijai.v9.i2.pp269-275.

In chapter 4, we present a seismic detection model, based on a Fully Convolutional Networks classifier which is trained on STanford EArthquake Dataset (STEAD)

and tested on real continuous seismic data. We apply our classifier to real continuous data from Morocco's XB network and compare the detected earthquake with our catalog. Chapter 4 is based on the research previously published in the following papers:

- Choubik, Youness & Mahmoudi, Abdelhak & Majid, Mohammed. (2021). Fully Convolutional Networks for Local Earthquake Detection. International Journal of Advanced Computer Science and Applications. 12. 10.14569/I-JACSA.2021.0120286.
- Choubik, Youness & Mahmoudi, Abdelhak & Majid, Mohammed. Seismic data Augmentation using Deep Learning. Submitted to CESER International Journal of Artificial Intelligence.

Finally, in chapter 5, we conclude and present ideas for future work.

Background

2.1 Introduction

Deep learning is an artificial intelligence subfield that lets a computer learn tasks performed by humans, such as identification and prediction. The second section of this chapter gives a background information on both deep learning and its benefits. We introduce deep learning and present the major building block of this field. We first begin by defining some terminologies generally used in deep learning. Also, we present the learning process including the forward and backward propagation. Then we define some parameters needed for the learning process. Finally we present the two deep learning algorithms used in this thesis, Convolutional Neural Networks and Generative Adversarial Networks.

The term big data has become a hot topic. Big data is a collection of a huge amount of data that is difficult to manage and analyze with conventional technologies such as database management systems. In the third section we introduce some big data concepts and technologies, such as Apache Hadoop and Spark. Then, we

present an overview of cloud computing, the mechanism, advantages, disadvantages, and service examples.

2.2 Deep Learning

Machine learning as a subfield of artificial intelligence is an attempt to teach computers to learn on their own from large amounts of data instead of following imposed rules. This is possible thanks to the processing power of modern computers, which can easily process large datasets. Neural Networks are a well known algorithm in the machine learning area. The creators of the first neural network were inspired by the structure of the cerebral cortex. The basic component of the network is the perceptron, it is essentially the mathematical analogue of a biological neuron, and like the brain, perceptrons are linked with each others.

If we think of machine learning as a variant of artificial intelligence, then deep learning is a specialized type of such variant. Deep learning is essentially a neural network training technique that uses many layers to solve complex problems. In the past decades, most neural networks were single-layer due to the high cost and limited capabilities of the computers. Deep learning is just a more convenient name for artificial neural networks. "Deep" in this phrase means the degree of complexity (depth) of the neural network, which can often be very superficial.

2.2.1 Supervised and unsupervised learning

Supervised learning involves the use of labeled datasets containing inputs and expected outputs. When you train a neural network with supervised learning, you give both inputs and expected outputs. If the result generated by the neural network is incorrect, it will adjust its calculations. It is an iterative process that ends when the network stops making mistakes.

Unsupervised learning is machine learning using datasets with no specific structure. When you train a neural network out of control, it independently performs logical data classification. In this case, the network is not trained on the labeled data. Instead, it classifies the input data on its own.

2.2.2 Artificial Neuron

The artificial neuron is inspired from the biological neuron which receives an input from the neural dendrites then produces an output transmitted along its axon (Figure 1). Analogically, The artificial neuron is a mathematical function that receives an input vector x and associate to it a weight vector w , and each element of x is weighted, then the sum of all the weighted element is added to a bias b and passed through an activation function f (Figure 2).

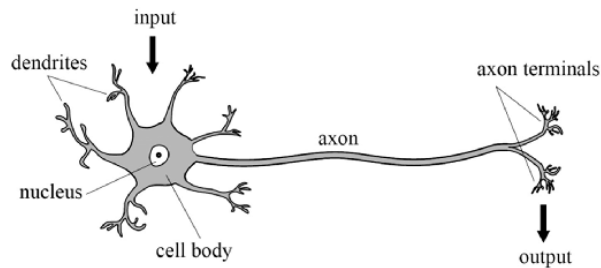


Figure 1 – Biological Neuron

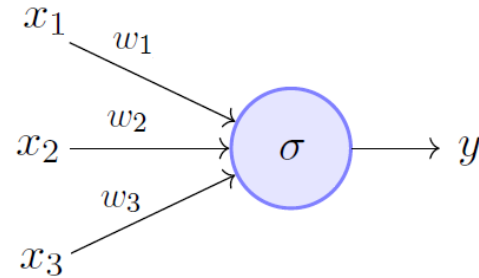


Figure 2 – Artificial Neuron

Given an input vector x with n element ($x \in \mathbb{R}^n$), The weight vector w has the same dimension as x , so $w \in \mathbb{R}^n$, while the parameter b is a real number ($b \in \mathbb{R}$). The neuron apply the following formula to the vector x , in order to calculate the intermediate output z :

$$z = w^T x + b = \sum_{i=1}^n w_i x_i + b \quad (2.1)$$

The second operation done by the neuron is the activation function f which takes the intermediate output z and return the final output a also know as activation.

$$a = f(z) \quad (2.2)$$

2.2.3 Activation Function

The activation function is the key component of the artificial neuron which introduces non-linearity to the summed weighted input. Multiple activation function exist, but in this section we are presenting only the most known like sigmoid, Tanh and Relu functions.

Sigmoid

The sigmoid activation function is very popular, and for a long time, it was the default activation used on neural networks. The sigmoid function takes a Real input $z \in [-\infty +\infty]$ and returns a value in the range $[0 \ 1]$. It is defined as follow:

$$f(z) = \frac{1}{1 + \exp(-z)} \quad (2.3)$$

Since the sigmoid output is in the range $[0 \ 1]$, it is considered as a probability and often used in the output of the neural network.

Tanh

The hyperbolic tangent or tanh for short is a scaled and shifted version of the sigmoid function, it returns a value in the range $[-1 \ 1]$ and defined as follow:

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2.4)$$

In general, the tanh function performs better, and is preferred over the sigmoid function.

ReLU

The Rectified Linear Unit or ReLU is a simple function that returns the same input if it is positive, otherwise it returns zero. It is defined with a simple max function as follow:

$$f(z) = \max(0, z) \quad (2.5)$$

The ReLU function does not require the use of the exponential calculation unlike the sigmoid and tanh functions, and become the default activation function for most neural networks architectures.

2.2.4 Artificial Neural Networks

An Artificial Neural Network, as its name indicates, is a network of artificial neurons assembled in a certain manner. The artificial neurons are stacked vertically in layers, and those layers are stacked horizontally to build the Artificial Neural Network (ANN), see Figure 3.

A Neural Network of L layers is comprised of an input layer, which only transmits the input X as output $A^{[0]}$ to the first hidden layer, followed by $L-1$ hidden layers with each contains $n^{[l]}$ neurons (with $l \in [1, L]$), and finally an output layer which returns the final prediction output of the network $\hat{Y} = A^{[L]}$, which is the prediction of the real desired output Y .

Excepting neurons of the input layer ($l=0$), which only transmit the input X to the first hidden layer, each neuron in a layer l takes the input from the previous layer

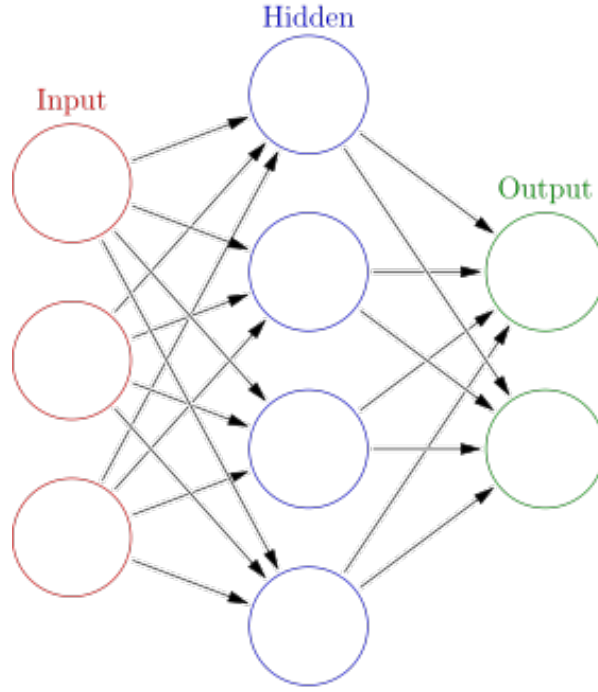


Figure 3 – An Artificial Neural Network

l-1 and calculate the summed weighted input as an intermediate output $Z^{[l]}$, then apply an activation function to that intermediate output to return the activation $A^{[l]}$ of the layer l.

$$Z^{[l]} = W^{[l]T} A^{[l-1]} + b^{[l]} \quad \text{with } l \in [1L] \quad (2.6)$$

$$A^{[l]} = f(Z^{[l]}) \quad \text{with } f \text{ is the activation function} \quad (2.7)$$

Where : X is the input matrix of size (n,m) , which contains all the inputs $x(i)$ with $i \in [1,m]$ and m is the number of input vectors. While $W^{[l]T}$ is a concatenation of all the weights of the layer's neurons, with a size of $(n^{[l]}, n^{[l-1]})$ with $n^{[0]} = n$.

2.2.5 Training a Neural Network

The training is the process of finding the best weights and biases that gives the best output $\hat{Y}$, which is the nearest possible value to the desired output Y . To train a neural network two steps are required; the first is the forward propagation and the second is backward propagation.

Forward propagation

The forward propagation start by taking the input $A^{[0]} = X$ and calculate the activation outputs for each layer l by applying the two formula 2.6 and 2.7 respectively for each layer l in the network, so, we end up by calculating the predicted output $\hat{Y}$. To quantify the difference between the predicted output $\hat{Y}$ and the real expected output Y , we compute the loss function J , which is an essential part of the forward propagation. There are many loss functions, but the most used ones are the following:

Cross-Entropy

$$J(W, b) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \quad (2.8)$$

Mean Absolute Error

$$J(W, b) = \frac{1}{m} \sum_{i=1}^m |\hat{y}^{(i)} - y^{(i)}| \quad (2.9)$$

Mean Squared Error

$$J(W, b) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 \quad (2.10)$$

Backward propagation

The network error is determined by its outputs, i.e. related to the output layer of the network. To determine the error for neurons on hidden layers, we needed a procedure for transferring the error from the output layer to the input layer of the network. Therefore, this method is called the Backward propagation method, or back-propagation for short.

The Backward propagation is the process of updating the weights and biases of the neural network. This process start by calculating the derivatives of the loss function with respect to every parameter in the network. The derivative of a function f measures how changing the input variable effect the output value. So, in order to update the weights and biases, which are the input variables of the loss function, we should calculate the derivatives of the loss function with respect to each variable, then update the variables using the calculated derivatives.

The basic function behind back-propagation is called gradient descent, which is based on updating the parameters of the network with respect to the derivatives of the loss function. In order to minimize the loss function, we should change the parameters of the network, and this should not be done randomly, for that reason, the derivatives of the loss function with respect to every parameter are calculated. by using the derivatives of the loss function we can properly update the parameters of the network as follows:

$$W = W - \alpha \frac{\delta J}{\delta W} \quad b = b - \alpha \frac{\delta J}{\delta b} \quad (2.11)$$

Where α is the learning rate. It is very crucial to the convergence of the back-propagation process. The parameter α should not be very large, otherwise, the

Gradient Descent will never converge to the optimal minimum, neither should be very small, otherwise, the updates will be very small, and the Gradient Descent will be very slow.

Gradient-Based Learning

The core function behind ANNs is the Gradient Descent which allows the network to adjust the parameters in order to minimize the cost function. The essence of the method is to slightly change the weights after each iteration . By calculating the derivative (or gradient) of the loss function for a certain set of weights, it is possible to determine in which direction the minimum is. To minimize the loss function, you need to iterate over the data many times. The weights are updated using gradient descent automatically.

Gradient descent is the most used algorithm in almost every machine learning model and without the gradient descent, machine learning would not exist. gradient descent technique, is widely used for training deep neural networks. Gradient descent is a method for finding the minimum value of a loss function J , it is used to control error in predictions of a machine learning model. Finding the minimum means getting the smallest possible error and improving the accuracy of the model. We minimize the loss function by iterating over the training dataset while adjusting the parameters of our model (weights and biases).

The essence of the algorithm is the process of obtaining the smallest error value. Further, in order to find the lowest error in the loss function, we need to adjust the model parameters. The slope of the graph of a function is the derivative of the function with respect to the variable and this slope always indicates the nearest minimum. The derivative of the loss function with respect to one weight, is the direction in which we need to move in order to reach local minima. As we iterate over

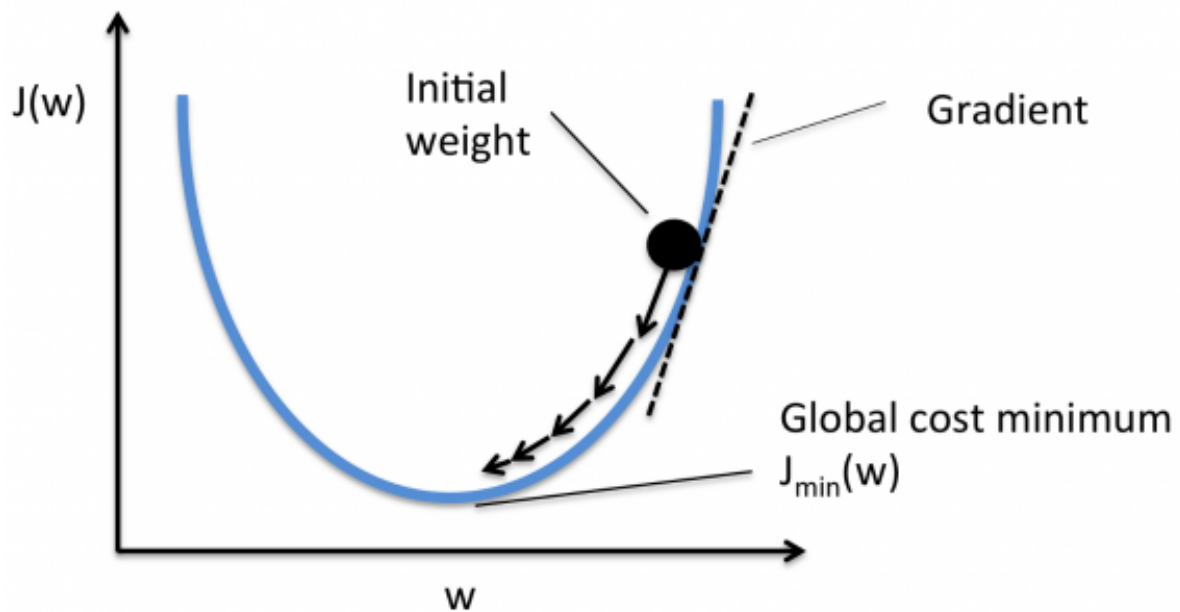


Figure 4 – Gradient Descent

all the training data, we keep adding the derivative of the loss function with respect to the weights (dJ / dw) for each weight. Then we divide the collected values by the number of training examples to get the average. We then use that average (of each weight) to set that weight. dJ / dw depends on your choice of loss function. The most common loss function is the root mean square error. Three types of the gradient descent exist:

Batch Gradient Descent

The batch gradient descent estimates the error for all the training samples and updates the model upon evaluation of all the training samples. The training epoch is a single pass through the full training samples.

Mini-batch Gradient Descent

Instead of iterating over all training examples and with each iteration performing

calculations on only one training example, we process n training examples at once. This choice is good for very large datasets.

Stochastic Gradient Descent

Instead of using the entire training dataset, we just use one sample to calculate the gradient. Since only one sample is used, the path to local minimums will be very noisy.

2.2.6 Parameters and Hyperparameters of the network

A Neural Network contains many parameters, and by tuning those parameters the network can learn the model representations. Below, we present some of the Neural Network parameters.

Weights and Biases

The weights and the biases form the model of the Artificial Neural Network, and by finding the best weights and the biases the network achieves the best performance. The Weights are randomly initialized and represent the strength of the connection, I.e. how much influence does the input have on the output.

Learning Rates

The learning rate is a parameter of the gradient descent algorithm that helps us control the change in weights in our network. Training the network will progress very slowly if we keep the learning rate low, as it will introduce small changes to the weights in the network and can also lead to overfitting. Whereas, when we set a high learning rate, the loss function tends to bounce chaotically, skipping the local optimum, causing the learning to diverge. The Learning Rate Schedule helps

us adjust the learning rate during the learning process by changing the learning rate according to a predefined schedule. There are three options for learning rate Schedule such as decay over time, exponential decay, and step decay.

Batch size

The batch size refers to the number of training samples feed to the input of the network. Some documents recommend using a larger batch size that can be supported by the system's memory. Some also suggest changing the packet size rather than the learning rate. However, a larger batch size allows a higher learning rate to be used. Large lot sizes tend to have low early learning losses, while final loss values are low when lot size is reduced.

2.2.7 Convolutional Neural Network

Convolutional Neural Network (CNN) is an architecture of Neural Networks proposed by Yan Lecun [18] known to be very robust in many research areas specially in computer vision. A two-dimensional input is taken into account by this architecture. At the same time, convolutional networks are resistant to small displacements, changes in scale and rotation of objects in the input.

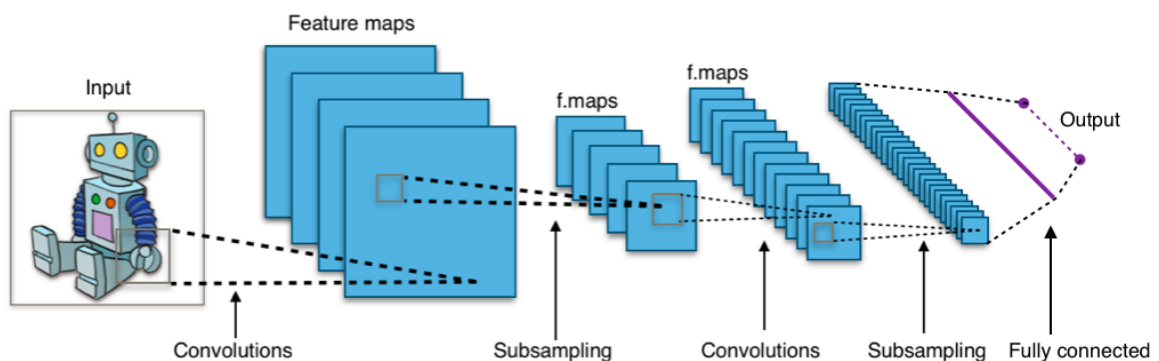


Figure 5 – Convolutional Neural Network Architecture

The main feature of CNNs is that they work with multi-dimensional inputs, while multilayer perceptrons work with one dimension inputs, and therefore it makes no difference for them whether some points are near or at opposite ends. CNNs are made up of basic blocks, so that they can be assembled as a constructor. The main building blocks of convolutional neural networks are convolutional layers, pooling layers and fully connected layers.

Convolution layer

The convolution layer is the most important layer of the network. Its main purpose is to highlight features in the input and generate a feature map. In order for the layer to be able to distinguish features, it contains the so-called filters (or kernels). Kernels are just a collection of tensors. These tensors are of the same size, and their number determines the depth of the output 3D array. In order to form a feature map from the input, the input is convolved with each of the filters. Convolution is the operation of calculating a new value for the selected pixel, taking into account the values of the surrounding pixels (figure 6).

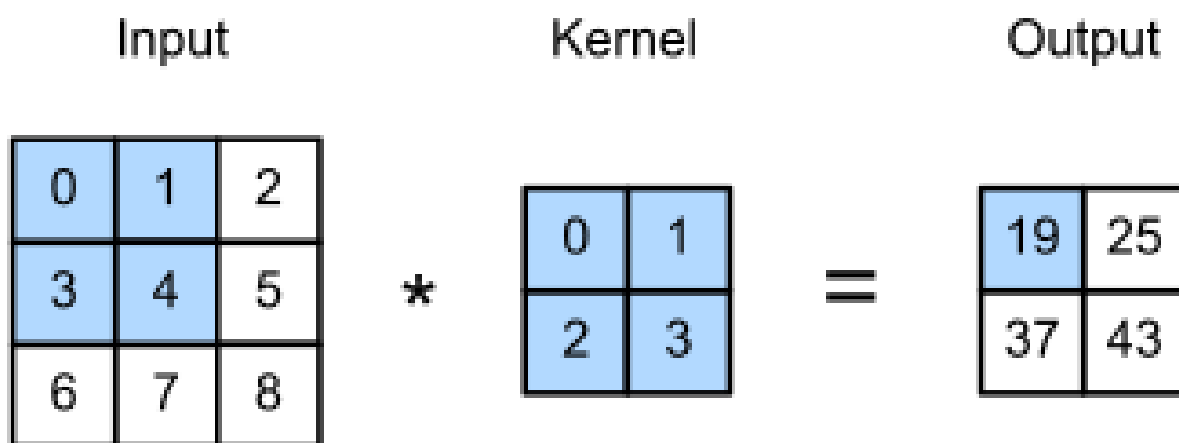


Figure 6 – A Convolution example

The filter is applied to the upper left part of the input and the filter values and the image values are multiplied componentwise, after which the filter moves further along the image until all its sections are processed in the same way. Then the numbers of the obtained matrices are summed up into a single matrix. After that, the bias value is added to each value of the matrix. The resulting matrix constitutes one channel of the output feature map. After the channels for each of the filters are obtained, the matrices are combined into a single output tensor.

The Convolutional layer parameters are the number of filters (fc) in the layer, the filter size (fs) which is the height and width of the filters tensor (usually an odd number), the stride (S) which is the number of values by which the filter matrix is moved across the input. Finally, a zero padding (P) which is the number of pixels that are added from each edge of the input to avoids reducing the input by the size of the filter). Thus, if the input parameters of the convolutional layer are a tensor of size $W_1 \times H_1 \times D_1$, the output parameter of the layer will be a tensor of size $W_2 \times H_2 \times D_2$, where :

$$W_2 = (W_1 - fs + 2P)/S + 1, \quad H_2 = (H_1 - fs + 2P)/S + 1, \quad D_2 = fc \quad (2.12)$$

Fully connected layer

The Fully connected layer is a layer whose input neurons are connected to all output neurons. The main task of a fully connected layer is to model a complex nonlinear function most often used for classification. This function is optimized during the training of the network. It can be considered that the layers going up to fully connected are means of preprocessing, and further classification is performed by a conventional feedforward network. That is, everything that goes to the fully

connected layer is used to highlight various features, which are then fed to the classifier.

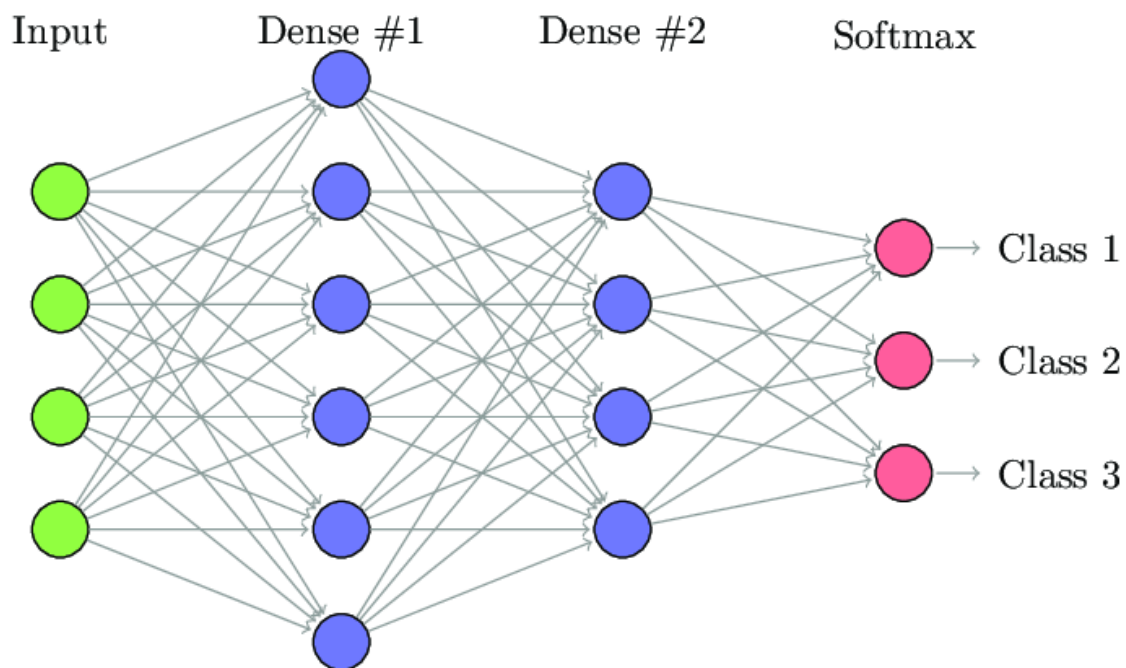


Figure 7 – An Artificial Neural Network

When compared to convolutional layers, fully connected layers perform worse and aren't as good at feature extraction. They also have a larger number of weights to train, resulting in a longer training time, whereas convolutional layers are trained with fewer parameters.

pooling layer

The main task of the pooling/subsampling layer is to reduce the dimension of the input, which makes it possible to reduce the number of further calculations. In

addition, this reduction in dimensionality allows the network to be more resistant to small variation in the input.

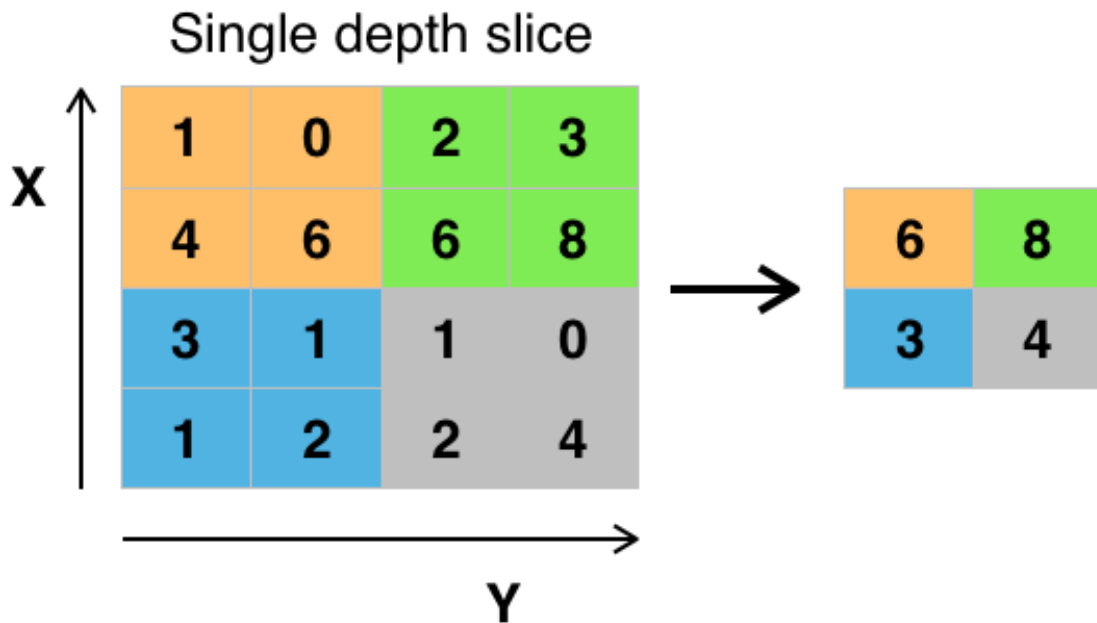


Figure 8 – A max pooling example

There can be a lot of ways to reduce the dimension of the feature map space, by selecting the maximum values (max pooling), averaging the values (average pooling), taking the sum (sum pooling), or taking the minimum (min pooling). Most often in convolutional networks, max pooling is used in the pooling layer.

2.2.8 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were invented by Ian Goodfellow, Yoshua Bengio and other researchers at the University of Montreal in 2014. The idea behind generative adversarial networks is to form a quality criterion for the generator using another neural network (Figure 9). In GANs, there are two neural

networks working in parallel:

the generator: by feeding noise as input, it generates new instance of a given class, for example, a photo of a human face or handwritten digit like those found in the MNIST.

The discriminator: evaluates the quality of the generator's work and decides whether the received data instance belongs to the training dataset (real) or was created by the generator. The discriminator is trained on real samples, so during training real data are fed into the input.

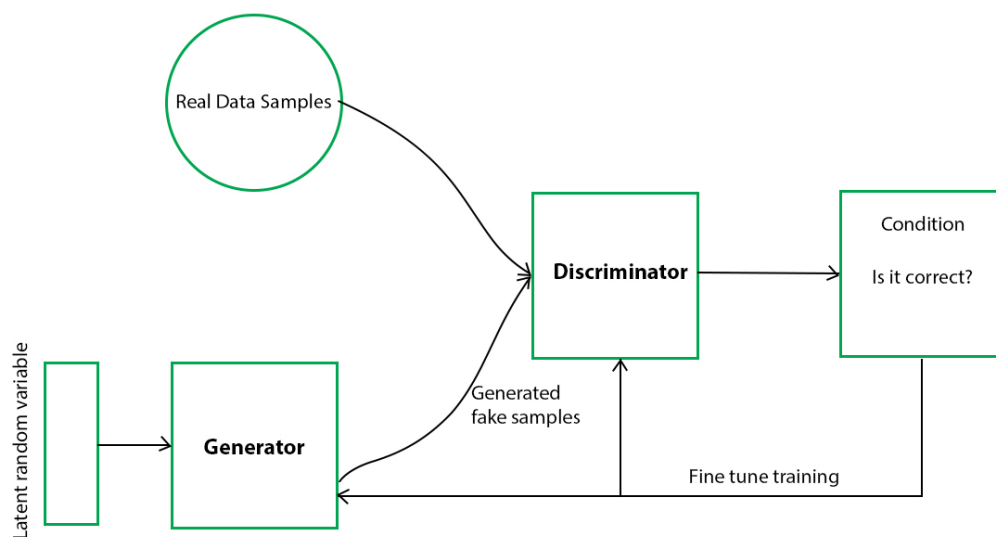


Figure 9 – A Generative Adversarial Networks Architecture

The generator start by receiving input noise and passes it through the network weights to form a generated sample. Then, the generated sample is fed to the discriminator along with a stream of samples taken from the dataset. Finally, The

discriminator returns probabilities which vary from 0 to 1 and can be perceived as the degree of confidence of the discriminator that the generated sample is real or fake. A probability close to 1 indicating that the sample is exactly from the training dataset and close to 0 indicating generated by the generator:

$$\text{real} \implies 1, \quad \text{fake} \implies 0 \quad (2.13)$$

Since we make a distinction between two classes (real and fake), it is advisable to take binary cross-entropy as an indicator of quality:

$$\begin{cases} \text{loss}_{\text{real}} = -t_R \log(\text{real}) - (1 - t_R) \log(1 - \text{real}) \\ \text{loss}_{\text{fake}} = -t_F \log(\text{fake}) - (1 - t_F) \log(1 - \text{fake}) \end{cases} \quad (2.14)$$

where $t_R = 1, t_F = 0$ are the desired network responses. Simplifying and summing up these losses, we get:

$$\text{loss} = -\log(\text{real}) - \log(1 - \text{fake}) \quad (2.15)$$

It can be seen from this formula that the indicator should be minimum for real samples and maximum for generated ones. The discriminator acts as a quality criterion for the generator. That is, the generator must generate the sample in such a way that the discriminator cannot distinguish it from the real one:

This means that in the training mode of the generator in binary cross-entropy, the required response $t_F = 1$. We get the quality criterion:

$$\begin{cases} \text{loss}_{\text{gen}} = -t_F \log(\text{fake}) - (1 - t_F) \log(1 - \text{fake}) \\ \text{loss}_{\text{gen}} = -\log(\text{fake}) \end{cases} \quad (2.16)$$

The discriminator and the generator compete with each other and are trained so that the discriminator understands the inputs from the generator, and the generator, on the contrary, tries to disguise its generated samples as real samples. As a result of this competition, the discriminator is better and better at distinguishing between real and fake samples, and the generator is more and more accurate in creating realistic samples. Ideally, the generator should form indistinguishable sample from real ones and the discriminator will no longer be able to distinguish them.

2.3 Big Data Tools

Big data is a modern technological direction associated with the processing of large amounts of data that are constantly growing. The prospects that Big Data can bring are interesting for business, marketing and science. Big data is data, so large that it is difficult for it to be processed with traditional tools. It can be structured, and unstructured. To come up with a means of processing growing data, it is necessary to create new, innovative solutions. Thus, Algorithms and software tools are being developed for Big Data.

The main goal of working with big data is to analyze discover new opportunities and information hidden in the data. Humans has produces huge amounts of data, and there are still problems with their management and analysis. Big data makes services more convenient and profitable by optimizing production resources to make production more efficient businesses. To reduce the ambiguity of definitions in the

field of Big Data, features have been developed that they must comply with. They all start with the letter V, so the system is called VVV:

Volume: The amount of information is measurable.

Velocity (speed): The volume of information is constantly growing, and processing tools must take this into account.

Variety: The information does not have to be in the same format. It can be structured or unstructured.

To these three principles, as the industry develops, additional V.s are added. For example, veracity (reliability), value, or viability. But the first three are enough for understanding: big data is measurable, incremental and heterogeneous.

2.3.1 Apache Hadoop

Apache Hadoop is one of the earliest and most influential open source tools for storing and processing large amounts of data. It is originated from a project called Nutch that tried to find the best way to crawl open source web pages. The most attractive feature of Apache Hadoop is that it is open source. This means that Hadoop is free and anyone can download and use it personally or professionally. Hadoop runs on Commodity hardware, and is designed for scalable, fault-tolerant, and distributed computing. It can provide fast and reliable analysis of both structured and unstructured data.

Data exists everywhere and When it is collected, there is so much information that it does not fit on one computer. In fact, search companies faced this particular problem when web content exploded. In 2003, Google released their influential paper, The Google File System [19], describing how their software handles storing massive amounts of data processed by their search engine. This was followed by the

MapReduce paper [20], simplifying data processing on large clusters and detailing how they made it easier to process such large amounts of data. These two articles have greatly influenced the architecture of Hadoop.

Big data systems had to accept this data would be distributed. It was unavoidable to keep the dataset on a cluster of computers in pieces. If clusters become the backbone of storage, software would have to account for hardware failure, which is unavoidable when dealing with hundreds or thousands of devices in a cluster.

In data computing, we are used to a particular machine, usually identified by an IP address or hostname, sending specific data to another specific machine. This is explicit communication had to be replaced with implicit communication, where one machine tells another machine that it must process certain data. Otherwise, programmers will face a validation problem at least as big as the data processing problem itself.

Machines will need a new way to interact with one another when they fail. A computer is generally defined by an IP address or hostname, transmitting specific data to another specific computer. Explicit communication had to be replaced by implicit communication, in which one computer instructs another to handle certain data. Thus, computing would need to go to the data and process it on the distributed machines instead of moving large amounts of data across the cluster.

Hadoop 1.x

Hadoop version 1.0 was Released in 2007, as Java-based programming environment. Its consisted of two components:

HDFS: Hadoop Distributed File System responsible for storing data across multiple machines. It is the distributed storage layer that Hadoop uses to distribute data and ensure it is properly stored for high availability.

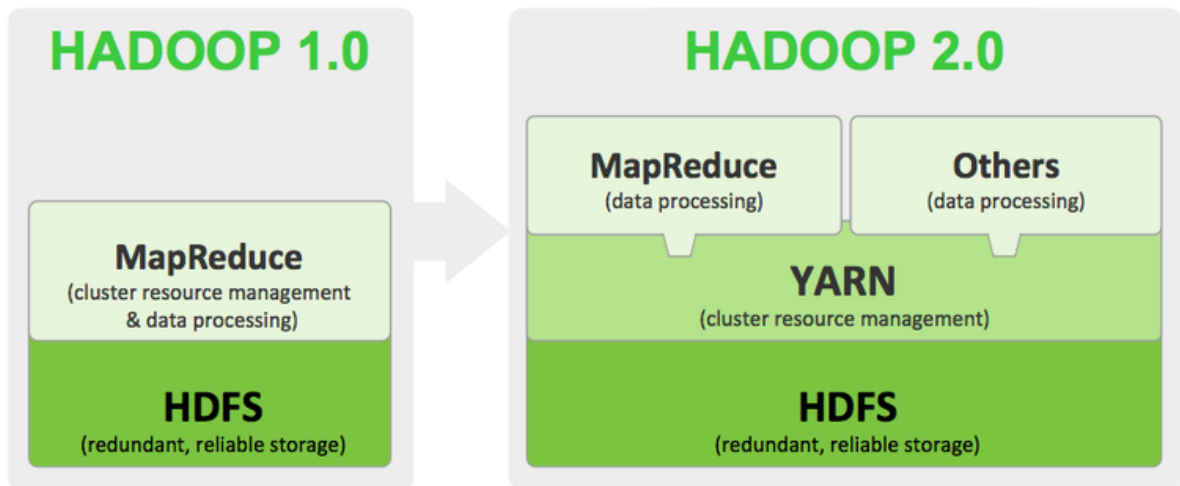


Figure 10 – Hadoop Architecture

MapReduce: A software environment for processing data in place and in parallel on each machine, as well as for scheduling tasks, and monitoring them.

HDFS 1.0

HDFS uses block replication to reliably store very large files across multiple computers with two different pieces of software: the NameNode server, which manages the file system namespace and client access, and the data nodes, which are responsible for serving read, write, deletion and replication requests, and the create block. HDFS stores each file as a sequence of blocks of the same size. By default, blocks are replicated three times, but both the block size and the number of replicas can be configured for each file separately.

When a new block is created, HDFS places the first replica on the node where the writer is running. The second replica is written on a randomly selected node. The third copy is then placed on a randomly selected machine in any rack except the rack in which the first replica was written. If the configuration specifies more

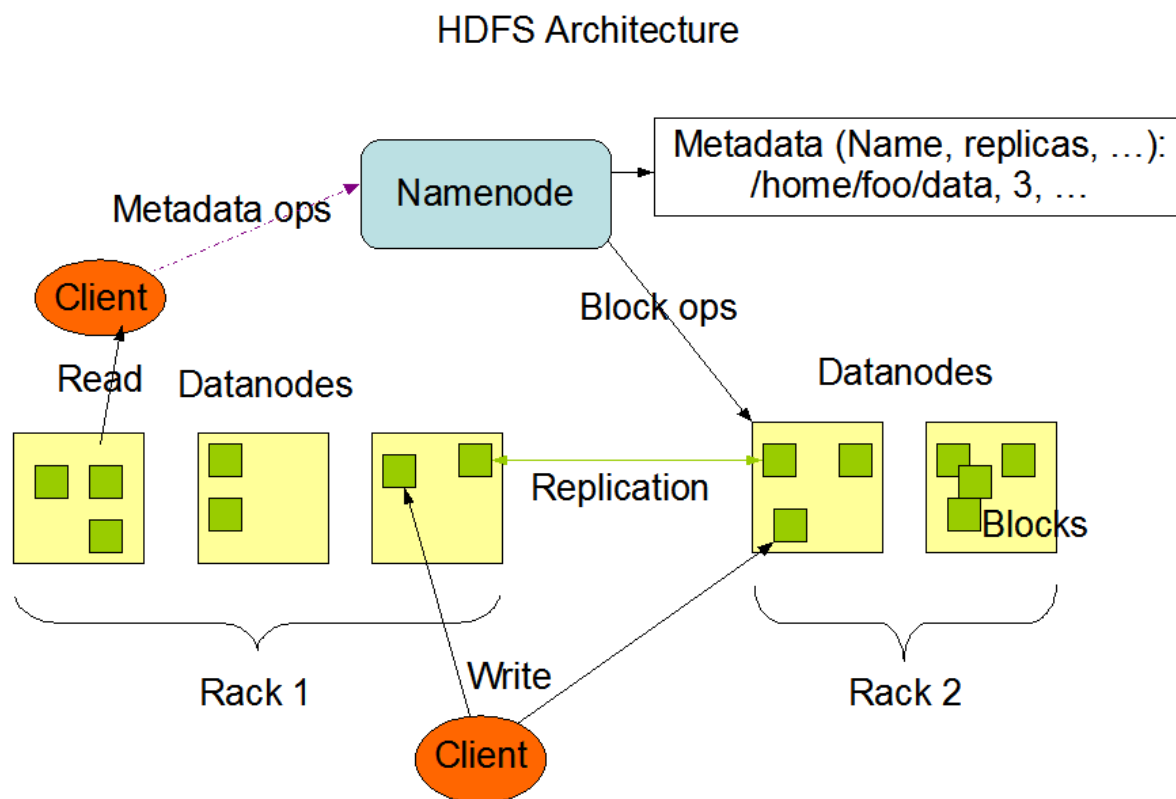


Figure 11 – hdfs Architecture

than three default replicas, the remaining replicas are randomly allocated, with the restriction that no more than one replica is located on one node and no more than two replicas are placed in one rack.

MapReduce 1.x

MapReduce, is responsible for batch processing data stored in HDFS. Hadoop's implementation of the Google MapReduce programming model allows the developer to use the resources provided by HDFS without having to work with parallel and distributed systems. Higher-level components can plug into the MapReduce layer to provide additional functionality. For example, Apache Pig provides a language for developers to write data analysis programs by abstracting the Java MapReduce

idioms to a higher level, similar to what SQL does for relational databases. Apache Hive supports data analysis and reporting using a SQL-like interface for HDFS. It abstracts Java API MapReduce requests to provide high-level request functionality for developers. Many additional components are available for Hadoop 1.x, but the ecosystem was limited by some key limitations in MapReduce.

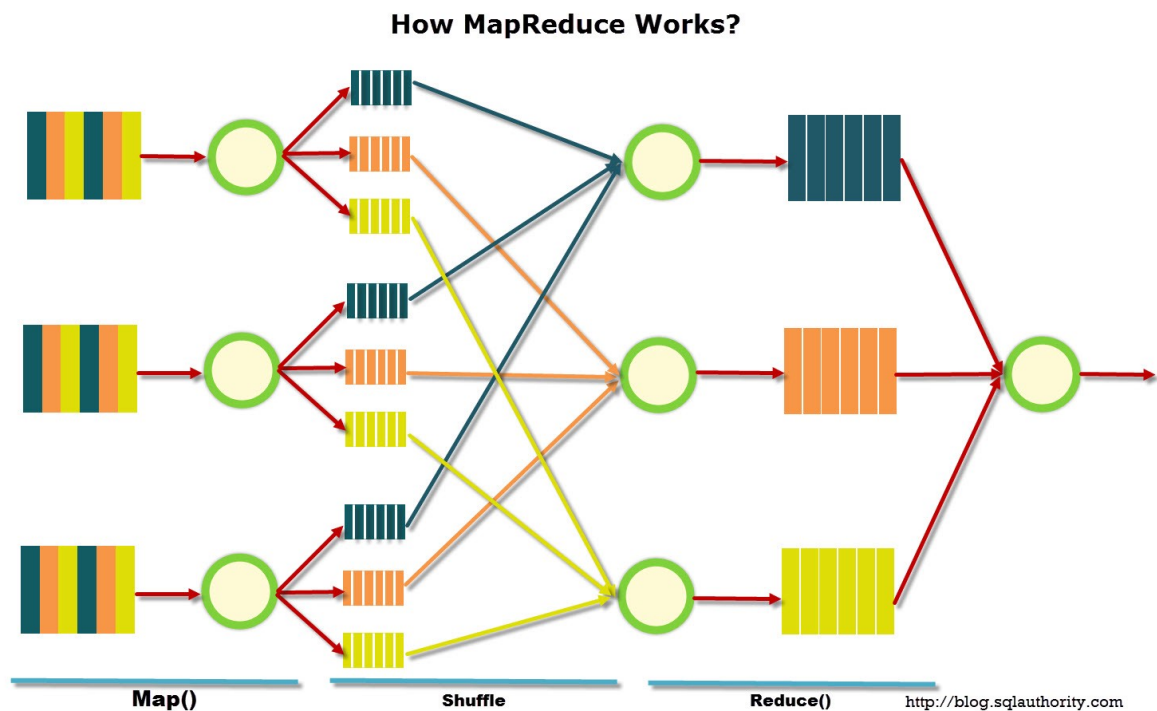


Figure 12 – MapReduce Architecture

Hadoop 2.x

Hadoop's 2.x branch, released in December 2011, introduced four major improvements and fixed major version 1 limitations. Hadoop 2.0 introduced:

HDFS Federation

HDFS federation provides a clear separation between namespace and storage, making it possible to use multiple namespaces in a cluster. Federated name nodes manage the file system namespace. They work independently and do not coordinate with each other. Instead, the data nodes in the cluster register with each name node by sending heartbeats and blocking reports and processing incoming commands from the name node.

NameNode High Availability

In addition to the improved scalability, performance, and isolation provided by the introduction of NameNode federation, Hadoop 2.0 also introduced high availability for the NameNode. Prior to Hadoop 2.0, in the event of a NameNode failure, the entire cluster was unavailable until it was restarted or started on a new machine. To prevent this, Hadoop 2.0 implements an active / passive configuration for fast failover.

YARN

YARN (Yet Another Resource Negotiator) is a large-scale distributed operating system for big data applications that makes Hadoop suitable for both MapReduce and other applications that cannot wait for batch processing to complete. YARN eliminated the need to run high-latency MapReduce I/O infrastructure to enable new HDFS processing models. Decoupled MapReduce remained dedicated to performing the task for which it was intended: batch processing.

ResourceManager High Availability

When it was first released, YARN had its bottleneck: the ResourceManager. The only JobTracker in MapReduce 1.x provides resource management, job scheduling,

and job monitoring. Early releases of YARN really improved this by dividing responsibilities between the global ResourceManager and ApplicationMaster for each application. The ResourceManager monitored cluster resources and scheduling applications such as MapReduce Jobs, but was the only point of failure until 2.4, which introduced the Active / Standby architecture.

Hadoop 3.x

Hadoop 3.x provide improvements such as HDFS erasure coding to save disk space, YARN timeline service enhancements to improve scalability, reliability and usability, support for more than two hostnames, an intranode balancer, and more. To find out more, visit the major changes overview.

2.3.2 Apache Spark

Apache Spark is a distributed processing framework for cluster computing. Originally developed at the AMP Lab at the University of California, Berkeley in 2009 and is currently being managed and developed by Apache Software Foundation. Apache Spark is an open source framework for processing huge data at high speed, with APIs for programming languages such as Java, Scala, and Python. It is used for tasks that require a large amount of data, such as weather / disaster prediction, gene / SNS real-time analysis, and behavior analysis of users who visit the site.

Apache Spark achieves 10 to 100 times faster speed than Hadoop by storing data in memory. In addition to being able to support real-time aggregation, it has a wide range of data storage location options. The downside of Hadoop is that it doesn't make good use of the memory of individual computers. Doing the same data multiple times, If you handle the same data many times, access storage will occur each time. In that regard, Apache Spark can run in-memory.

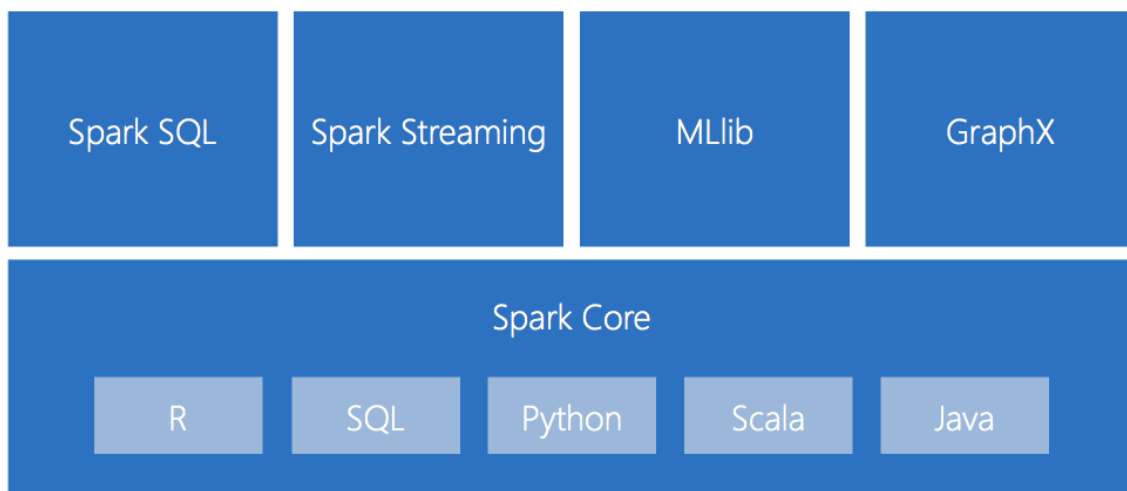


Figure 13 – Spark Architecture

Apache Spark has been attracting attention as a processing platform for big data in recent years. Apache Spark is composed of multiple components, it is divided into a parallel distributed processing engine "Spark Core" and a group of libraries. Here are five components of Apache Spark:

Spark Core

Spark Core is a platform that is a parallel distributed processing engine that provides the basic functions of Apache Spark. All other features are built on Spark Core. Spark Core provides in-memory computing capabilities, execution models that support applications, and Java, Scala, and Python APIs that facilitate development.

Spark SQL

Spark SQL is a library that provides an API for using SQL for structural data. A module for structured data processing that can be used as a distributed SQL query engine or even read data from an existing Hive installation. In addition, you can now process unmodified Hadoop Hive queries with existing data up to 100 times faster.

Spark Streaming

Spark Streaming is a library that provides the function of processing stream data. It processes stream data using the microbatch method, and realizes an analysis application with high interactivity in both streaming data and historical data while inheriting the advantages of Spark such as ease of use and fault tolerance. In addition, it can be easily integrated with various data sources running in Spark.

MLlib

MLlib is a machine learning library that provides APIs for using machine learning algorithms. MLlib is a scalable library that offers both high quality algorithms and up to 100 times faster processing than MapReduce. MLlib can be used in Java, Python and Scala as part of a Spark application.

GraphX

GraphX is a graph calculation library that provides the API required for processing graph structure data. GraphX is an engine for graph and graph parallel computing built on Spark, and by using GraphX, it is possible to interactively build a large amount of graph structure data, make changes and make decisions. It can be used in combination with Spark's API and other libraries.

2.3.3 Cloud Computing

Cloud computing means using computing services (servers, storage, networks, databases, software, etc.) in a cloud environment (Internet environment), and companies can benefit from cloud computing technology. In the past, it was necessary to prepare physical servers and networks to build a system. As a result, the problems of enormous initial costs and the time it takes for the system to become available have become apparent. However, by utilizing cloud computing, it has become pos-

sible to provide all of these via the Internet. As a result, companies were able to minimize initial investment and reduce times to system releases.

Cloud is a word that means that rather than storing data on terminals such as PCs and servers, storing it in a servers existing on the Internet and operating it. In addition, since it is not necessary to prepare a server in-house, it is possible to smoothly build the optimum environment for the company by freely increasing/decreasing or combining server resources. It is also a nice point that it leads to cost optimization because you can use resources as much as you need. In this way, cloud computing allows to build systems in a short period of time without cost and effort, can be said to be an indispensable technology for companies.

Cloud computing can be divided into three types:

Public cloud

Cloud computing services are fully hosted and managed by third-party providers, providing users with reasonably priced computing resource access services. Users do not need to purchase hardware, software or supporting infrastructure, but only pay for the resources they use. Public cloud users do not need to pay hardware bandwidth fees, and the investment cost is low, but the data security is lower than that of private clouds. The public cloud is a form of open cloud service that anyone can use. Google, Yahoo, Amazon, etc. are typical examples of public clouds, and the feature is that the service can be shared by an unspecified number of people.

Private cloud

The enterprise purchases its own infrastructure, builds a cloud platform, and develops application cloud services on top of it. Private cloud can fully guarantee the security of virtualized private network, but the investment cost is higher than that of public cloud. A private cloud is a form of cloud service in which a company

or individual builds a dedicated environment. IBM and NEC are typical examples of private clouds, and they are characterized by their ability to customize and control service content.

Hybrid cloud

Generally created by users, and management and operation and maintenance responsibilities are shared by users and cloud computing providers. It uses private clouds as the foundation and combines the service strategies of public clouds. Users can follow the privacy of the business. Different levels of autonomy switch between public cloud and private cloud.

Large domestic enterprises prefer the private cloud model rather than the public cloud model. The core difference between the private cloud and the public cloud model is whether customers using cloud services have their own corresponding cloud infrastructure. The advantages of flexible configuration and low cost of the public cloud model are welcomed by small and medium-sized enterprises, while large enterprises pay more attention to the pertinence of solutions and information security, and are relatively insensitive to costs. At the same time, the deployment of public clouds in banking, power and other industries is also regulated. The limitation of the private cloud model has been adopted by large domestic enterprises in many places. Government departments that are more sensitive to data security will use private cloud as the main deployment model.

Cloud computing generally classified into IaaS, PaaS and SaaS according to the service model:

Infrastructure As A Service

Users can rent a complete computer infrastructure layer (resources such as computing, storage and network bandwidth) through the Internet. Users don't care

about the underlying infrastructure of the cloud system, they can run software, store data, and publish programs on it. Such as IBM's Blue Cloud and Amazon's EC2.

PPlatform As A Service

The software development platform is provided to users as a service (the deployment, allocation, monitoring and security management of resources in the system, as well as distributed concurrency control, etc.). On the PaaS platform, enterprise users can quickly develop applications without worrying about the resources required for program operation, and third-party software providers can also quickly develop customized applications suitable for enterprises.

Software As A Service

Provide users with services such as cloud software application services and user interaction interfaces through the Internet. These software applications are only installed in the cloud, which saves the cost of purchasing software.

virtualization

In order to provide the wide range of cloud services described previously, cloud providers need to accurately understand the needs of users and prepare the necessary computing resources without omission. Also, of course, each user has different requirements, so we must provide them as independent environments. In this way, the mechanism called "virtualization" realizes cloud computing that requires high flexibility. The simplest way to describe virtualization is to build multiple virtual servers on a physical server.

In virtualization, there is only one physical server, so there is only one CPU, memory, etc. However, by utilizing virtualization technology, it is possible to allo-

cate CPU processing amount and memory capacity to each virtual server, and it is possible to operate each virtual server as if it were an independent server. Virtualization technology is behind cloud computing, and virtualization is now an integral part of many enterprises.

Cloud computing is a mechanism for storing data in a virtual environment, it is possible to access information regardless of location or device. As a result, you can get the data you need anytime, anywhere, and as a result, accelerate your business. Flexible scaling according to the situation Scaling is to increase or decrease the resources (usage) of the service you are using. If you have your own server, changing resources is not easy, and you may need to review your contracted server or purchase new equipment.

Since cloud computing is a form that uses completed services, it is not good for individual customization for each company. There are some services that can be customized, but there are limits to what can be done, and it is not possible to realize the details. Therefore, there are cases where cloud computing is not suitable if you want to make detailed customizations according to your company's requirements.

There is a risk that the service will end during use. In the case of cloud computing, there is a risk that the services used by the company will suddenly end. In the extreme case, if the service provider goes bankrupt, the service will naturally become unusable.

Examples of services using cloud computing:

Google Cloud Platform

Google Cloud is a cloud service provided by Google. It is also used as the basis for operating services provided by Google, such as Gmail, YouTube, and Google Maps, and the point is that it has a solid track record of operating services.

Amazon Web Services

Amazon Web Services (AWS) is a service provided by Amazon and has the longest history of cloud services. One of its features is its rich functionality, and the long history of the service itself has made it a criterion for selecting various cloud services, and many companies are considering AWS first.

Microsoft Azure

Microsoft Azure is a cloud service provided by Microsoft. The service was launched in 2010, and sales are steadily increasing due to its affinity with various Microsoft products such as Windows Server and Microsoft Office.

2.4 Conclusion

In this chapter, we have introduced deep learning, as well some basic concept needed for the rest of this thesis. Then we looked at some big data technologies, such as Apache Hadoop and Spark. Finally we have introduced cloud computing services.

Big Data tools for earthquake detection

3.1 Introduction

Earthquake detection is an important task in seismology, it allows for identifying seismic events, which permits deductions about the interior of the Earth, identify areas with high seismic activity and adding more seismic stations to better understand the seismicity in these areas. There are many seismic event detection algorithms [21]. In this chapter we are interested in applying the Short Term Average to Long Term Average trigger algorithm (STA/LTA) [22], which is widely used in seismic detection. We applied STA/LTA to the three-component records of the XB seismic network installed in Morocco between 2009 and 2013. Due to the large amounts of seismic data that we will be working on, we used Hadoop MapReduce to benefit from its power of processing data in a distributed fashion.

3.2 Seismology overview

Seismology is a branch of geophysics that studies earthquakes, their causes, nature and consequences. The main carriers of seismological information are seismic waves, the interpretation of seismograms of which, along with the study of manifestations of seismicity, makes it possible to study the deep structure, physical properties and dynamics of the interior of the Earth and other planets. Seismology received wide applied development in the methods of seismic exploration of minerals, in earthquake-resistant construction, for defense purposes when registering explosions, including nuclear ones, as well as in other fields related to vibrational and shock effects.

Seismology as a modern scientific discipline began at the turn of the 19th – 20th centuries, when the first seismographs were designed and installed. One of the most important breakthroughs in the early stage of seismology was the invention of the electromagnetic seismograph by the Russian scientist, Prince Boris Borisovich Golitsyn in 1906. The data recorded by these seismographs accumulated over decades, and their analysis led to such fundamental scientific discoveries as understanding the internal structure of the Earth and the physical mechanism that leads to earthquakes, and was also a major contribution to the formation of the concept of plate tectonics - a modern geodynamic theory that explains motion and deformation of the Earth's upper shell and the origin of seismicity and volcanism.

Since the beginning of the 70s of the 20 century, a new direction has been developing in seismology - the physics of the earthquake source and the development of methods for predicting earthquakes. The search for earthquake precursors continues. In laboratory conditions, all kinds of physical processes are simulated that may be related to seismic sources. Modern seismology has a wide global seismic network. The stations record earthquakes according to a single time (Greenwich Mean Time) and uniform instructions. Information through communication chan-

nels is collected in data processing centers and is the source for world seismological research.

Each earthquake has its own magnitude M . This is a value that characterizes the strength (size of the rupture, energy, etc.) in the source. The closer to the epicenter, the more intensely the Earth's surface vibrates, the higher the level of seismic effects, and moving away from the epicenter, seismic signals attenuate. This is a fundamental property inherent in all signals in a continuous medium. Obviously, it is not the magnitude that destroys the building, but the peak acceleration of the soil, when it exceeds a certain level.

The introduction of modern digital and communication technologies, begun in the 1990s, has completely transformed seismology. Fast data transfer and efficient computer algorithms make real-time seismic monitoring possible. In addition to this, the quality of seismic records has significantly improved and their number has increased. Today, thousands of high-quality seismographs are installed around the world that record data in a continuous mode and transmit it in real time to data processing and storage centers, the main ones of which are located in the USA, Europe and Japan. The devices of the world seismological network register more than 200 thousand earthquakes a year.

3.2.1 Structure of the Earth

The earth is comprised of five layers (Figure 14, the first of which is the crust, which is made of resistant material and has a thickness of 25–40 km underneath the continents and 5 km beneath the oceans. The higher and lower mantle layers make up the mantle layer, which extends for 2850 kilometers. When subjected to impulsive stress, the materials of the mantle behave like a solid, but when subjected to stagnant stress, they behave like a fluid [23].

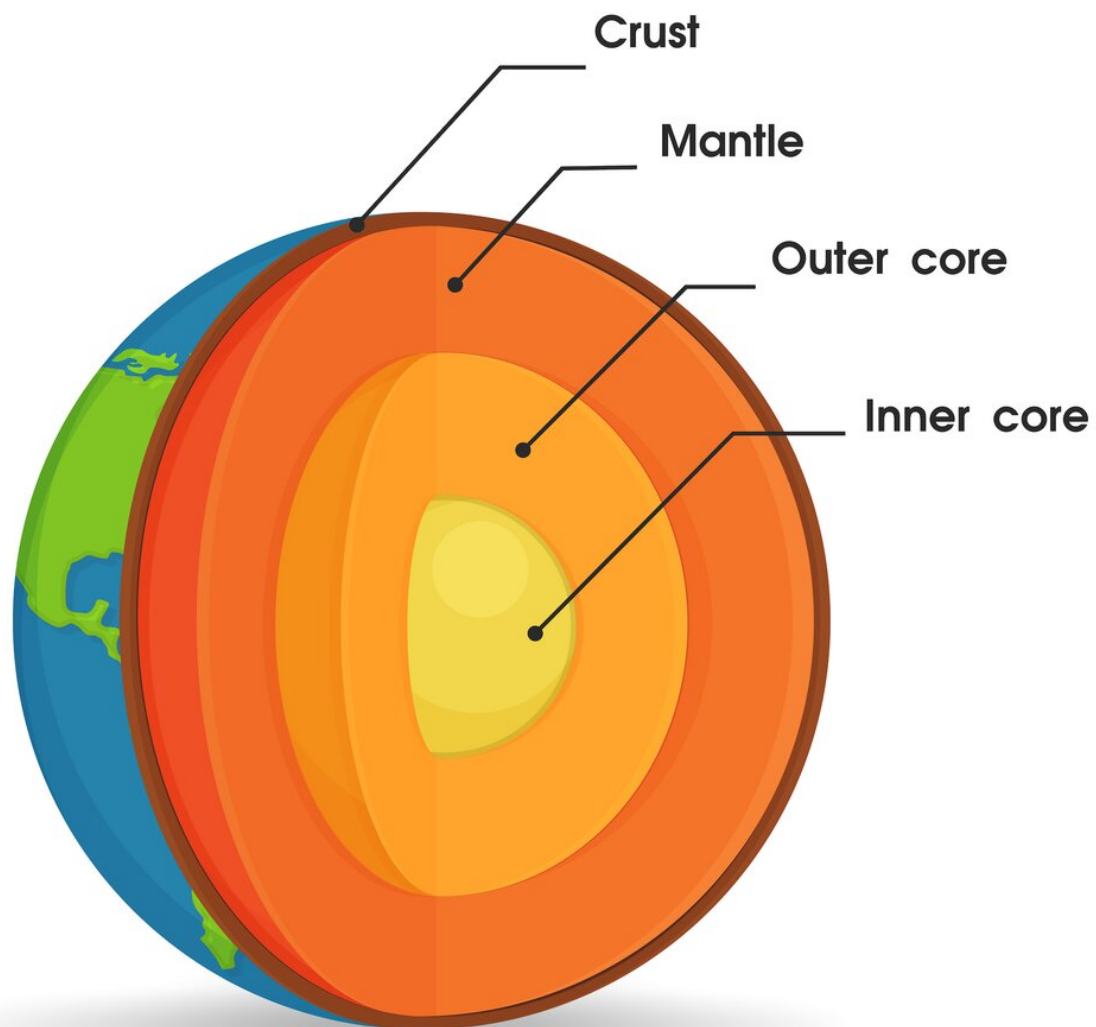


Figure 14 – The internal composition of the earth

The continental drift theory was introduced by Alfred Wegener according to which the earth had only one large continent called Pangea, then broke into different pieces that slowly drifted until the current continental configuration. The theory of continental drift was widely accepted by scientists and led to a modern theory of plate tectonics, according to which the earth's surface is divided into six major tectonic

plates and the fourteen subcontinental plates, that move relatively to one another (Figure 15).

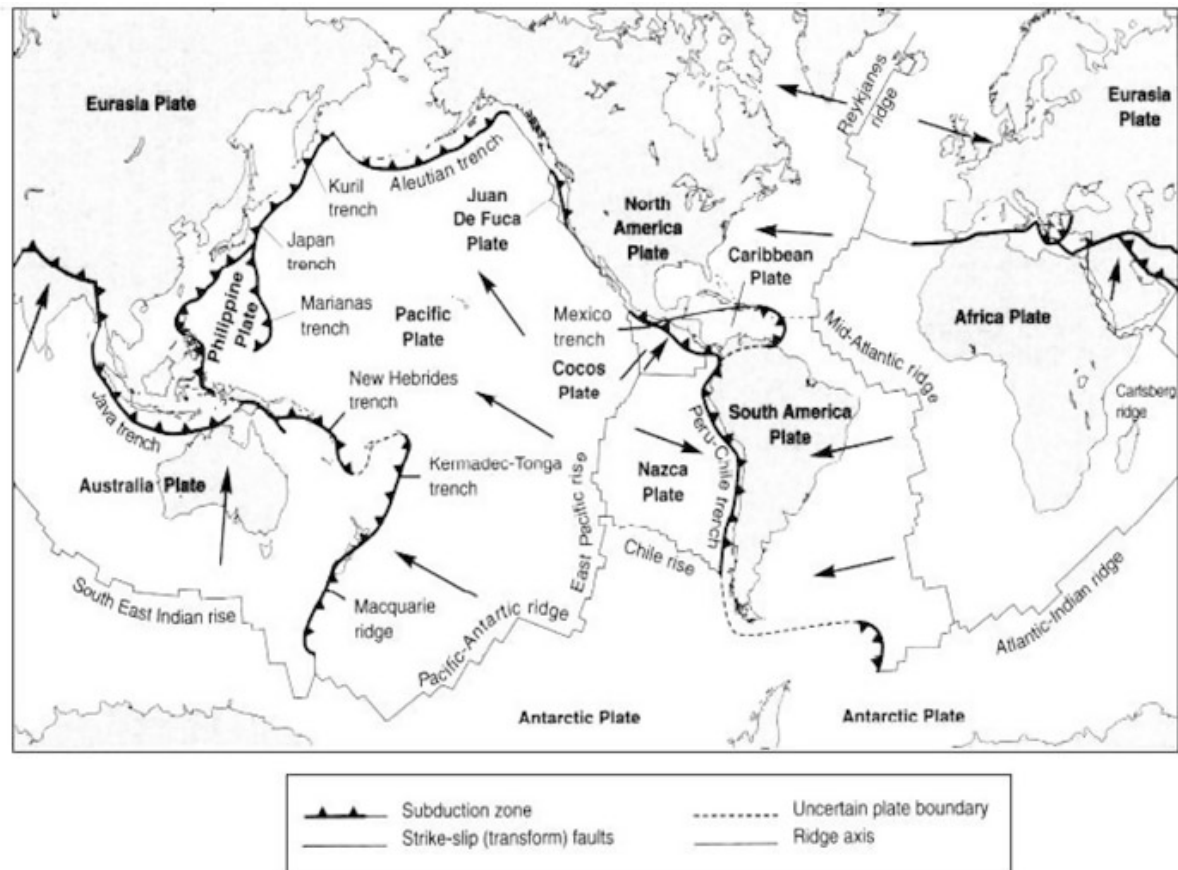


Figure 15 – The earth's tectonic plates

The upper mantle is in contact with the cooler portion of the crust while the lower mantle is located near to the hot outer core. The differences in temperature inside the mantle makes the denser material sinks due to the gravity, while the less dense material rise up. This movement occurs continuously and leads to the movement of the plate tectonics.

The nature of earthquakes is determined by the plate boundaries, which is divided into three types:

Spreading ridges: Because of the temperature difference between two neighboring plates moving in opposing directions, molten rock from the upper mantle comes to the surface and solidifies. As a result, the solidified mantle material accumulates over time and produces the spreading ridge.

Subduction zones: The consumption of plate material, known as subduction zones, must balance the formation of new plate material at spreading ridges. One plate slides below the other, in this section of the earth. Because the oceanic crust is colder and denser than the continental crust, it tends to sink underneath it.

Transform faults (strike-slip): The adjacent plate boundary is called a transform fault when the plates move together without producing or consuming crust. Earthquakes are primarily produced by tangential displacement along the transform fault. They are identified by depressions or scarps and are positioned orthogonally to two oceanic ridges.

The prediction of the time when the earth's surface starts to shake is a major challenge for seismologists all around the world. Science can identify the geographic areas on the verge of trembling, where some of the biggest cities in the world like Los Angeles, Tokyo and Istanbul are at risk.

The fault is the broken part of the crust along which action takes place. A fault's length can range from a few meters to hundreds of kilometers, and its depth can reach ten kilometers. The most famous criterion for determining whether a fault is active or inactive is the one suggested by [24], which measures an active fault by referencing the time after the most recent fault movement. The likelihood of an earthquake occurring on a given fault section is proportional to the time after the last earthquake. In addition, the emitted energy is a critical parameter in predicting the frequency of an earthquake. It is possible to predict gaps in seismic activity for

the studied fault section by considering fault movement and previous earthquakes along the fault.

The orientation (strike) and slope of the fault-plane (dip), as well as the direction of the sliding, are geometric parameters that characterize the fault (slip). The direction of the strike and dip determines the form of displacement that occurs on a fault, which is divided into:

Dip-slip movement: The main movement is further subdivided based on the movement's trajectory and dip angle. When the foot wall rolls over the hanging wall and a tensile force is produced in the crust, it is called a normal fault. On the other hand, a reverse fault is found as the foot wall passes under the hanging wall (Figure 16).

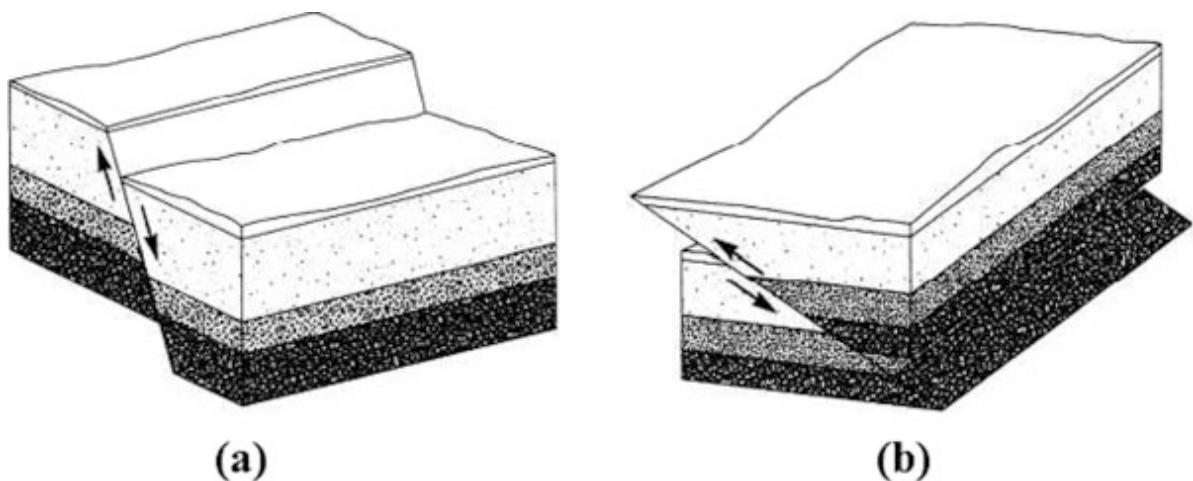


Figure 16 – Normal fault (a) and reverse fault (b)

Strike slip movement: The displacement is parallel to the impact, and the resulting strike-slip fault plane is usually vertical. A right lateral or left strike-slip fault may occur depending on the relative direction of movement (Figure 17).

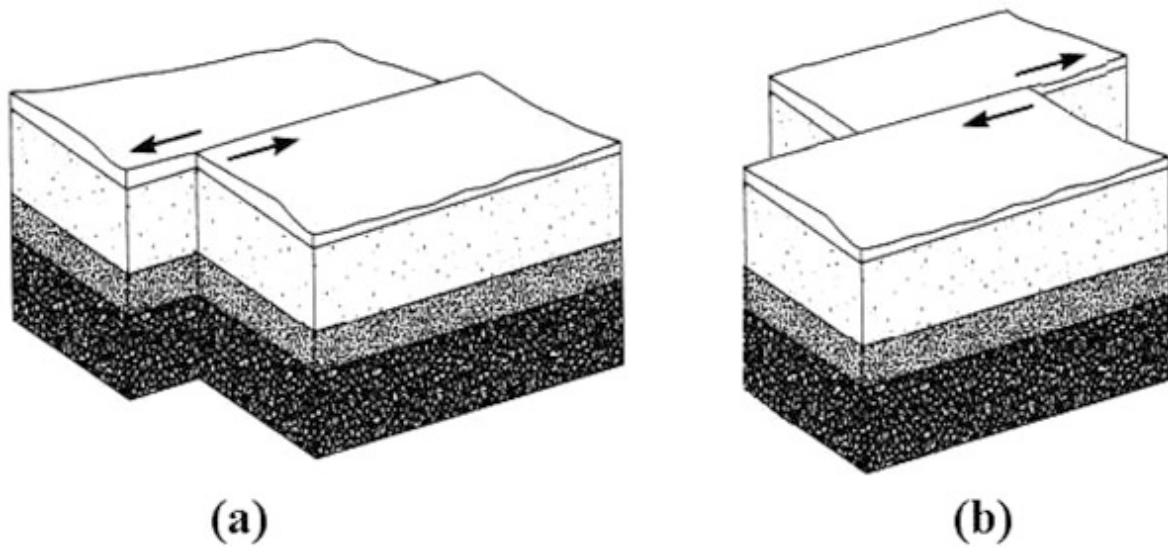


Figure 17 – Left lateral (a) and right lateral strike-slip (b)

Since a fault has both dip-slip and strike-slip elements, it is called oblique fault movement. Strike slip (left-lateral or right-lateral), natural, reverse, and oblique reverse are the four fault mechanisms that can be established (Figure 18).

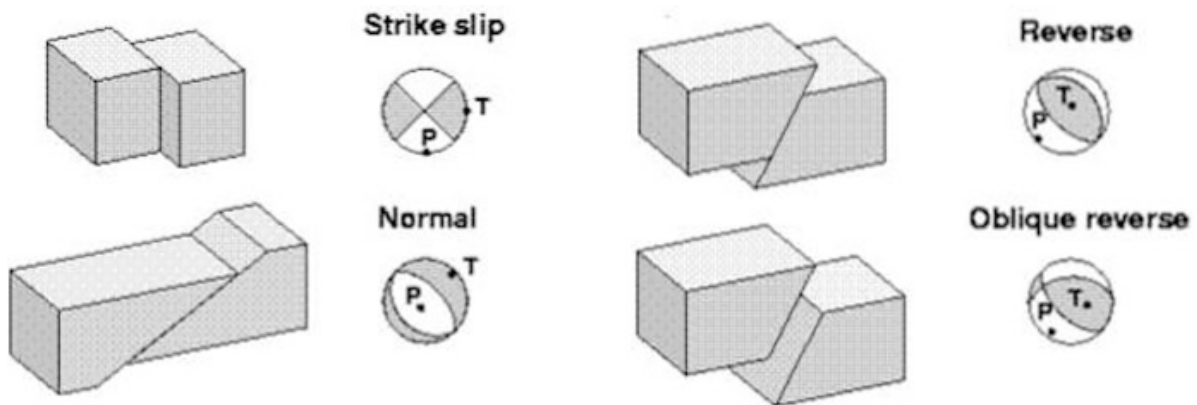


Figure 18 – Focal mechanism and related fault types

Elastic strain energy is retained in the materials as shear stresses on the fault plane that separates the plates during the moving process. The cumulative strain

energy is released as shear stress exceeds the fault material's shear power, resulting in material failure. The key shock is caused by the release of energy. Aftershocks occur as the stress on the fault surface gradually changes to the current uniform distribution.

3.2.2 Seismic waves

Seismic waves are elastic vibrations propagating in the Earth from sources of earthquakes, explosions and other seismic sources. Near the sources of strong earthquakes, seismic waves have a destructive force, and at considerable distances from the epicenters they are recorded by seismographs. The main types of waves are longitudinal, shear and surface. Longitudinal waves (P) transfer changes in volume in the medium (compression and tension). Oscillations in them are made in the direction of propagation. Shear waves (S) do not form volumetric changes in the medium and are oscillations of particles that occur perpendicular to the directions of wave propagation. Shear waves propagate only in a solid medium, they are absent in liquid media. Waves in appearance resemble waves on the surface of water, however, they have a different nature, representing a superposition of longitudinal and transverse waves. They are diverse (Rayleigh, Love, Stoneley waves, etc.). The propagation speed of seismic waves increases with depth.

The elastic wave propagation in the soil controls the motion on the earth's surface caused by a rapid release of energy from a seismic source. Only the energy is transmitted across the elastic body through volume or surface waves in this form of physical system. There are two different kinds of volume waves:

P-waves (compression waves): By compression and expansion phases, wave propagation induces particle oscillation in the same direction. The P-waves cause the elastic body's volume to change but not its shape (Figure 19 (a)).

S-waves (distortion waves): are a form of wave that occurs while The particles in the body fluctuate in a direction that is perpendicular to the wave direction. The volume of the body remained unchanged in this situation, but the shape changed (Figure 19 (b)).

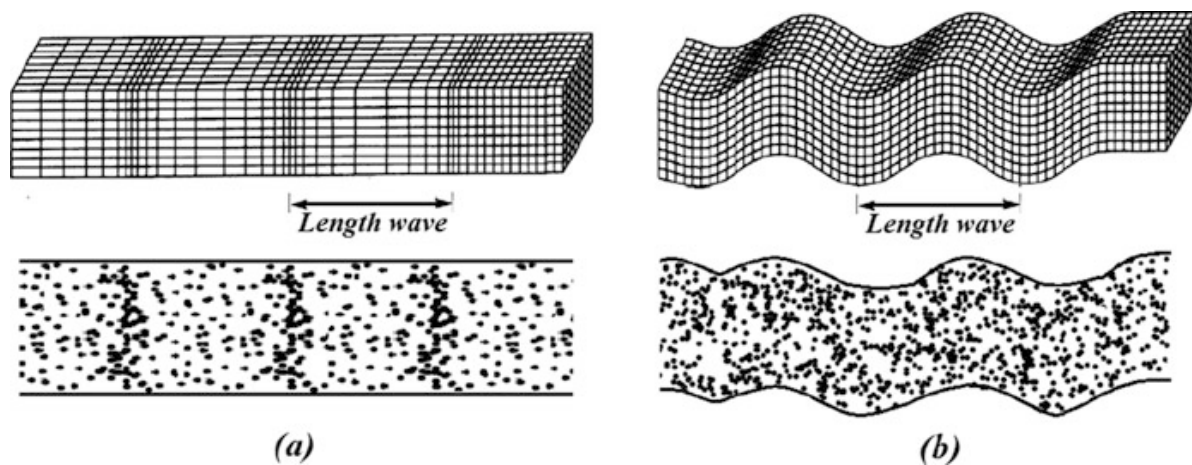


Figure 19 – P-waves (a) and S-waves (b) in an elastic 1D element

Since P and S waves spread on the Earth's surface, they generate a third form of wave known as surface waves. Surface waves are weak and stand out from the S waves that precede them on the recordings of local earthquakes. On the other hand, Surface waves, usually occur as the most dominant waves on long duration seismograms signaling distant earthquakes since they attenuate much more slowly than P or S waves.

The distribution of volume waves occurs in the earth's crust various surface layers. The refraction and reflection of waves is caused by the distinct stiffness properties of two neighboring soil layers. This effect causes people to pay attention mainly to S-waves on the surface, which cause the structures to shake horizontally. The earth's surface can be used as a wave propagation irregularity. The elastic waves are converted into surface waves as they reach the surface. Surface waves

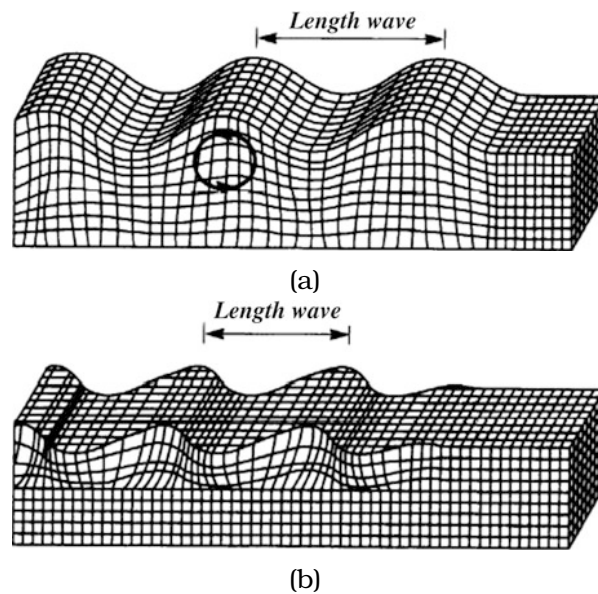


Figure 20 – Rayleigh waves (a) and Love waves (b)

are divided into two categories.

Rayleigh waves: The particle oscillation takes place in a vertical plane that contains the propagation vector. The elastic element's particles are all subjected to elliptical motion (Figure 20 (a)).

Love waves: The ions oscillate in a plane perpendicular to the surface plane. The perturbation velocity is dependent on the excitation frequency, but it can be estimated to be 90% of the S-waves velocity (Figure 20 (b)).

3.2.3 Earthquake detection

The International Monitoring System (IMS) is designed to monitor seismically active regions. Most of the information is recorded at single seismic stations. The more informative the recorded signal, the more various parameters can be deter-

mined and more accurately. Groups of seismic stations have begun to be used for observation. The most widely used are small-aperture groups with a diameter. due to the fact that in this case it is possible to neglect the signal distortions arising from the inhomogeneity of the earth's surface. The reason for using seismic groups is also that with this method of observation it is possible to apply algorithms for complex processing of the recorded multichannel seismogram, which provide a better quality assessment of the parameters of the recorded information, in comparison with single seismic stations.

One of the many problems arising in regional monitoring is the problem of identifying the type of seismic source or the problem of classifying seismic signals. It consists in determining the cause of the recorded event from the seismogram, i.e. distinguish between an explosion and an earthquake. Its solution involves the development of a certain method (decision rule), which, with a certain probability, could classify the recorded event into one of two classes. To solve this problem, various analytical methods from the theory of statistical analysis are currently used, which make it possible to correctly classify data with a high probability. As a rule, for a specific region, there is a database of recorded events. It includes an example of seismograms characterizing both earthquakes and explosions that have occurred in this region. All existing identification methods use this database as a training set, thereby, capturing subtle differences characteristic of a given region, adjust their parameters in a certain way and, as a result, learn to classify the entire training set for belonging to one of two classes.

As seismic waves pass across the earth's crust to meet the surface, they cause ground shaking. The strength and length of shaking at a specific location during an earthquake are largely determined by three factors:

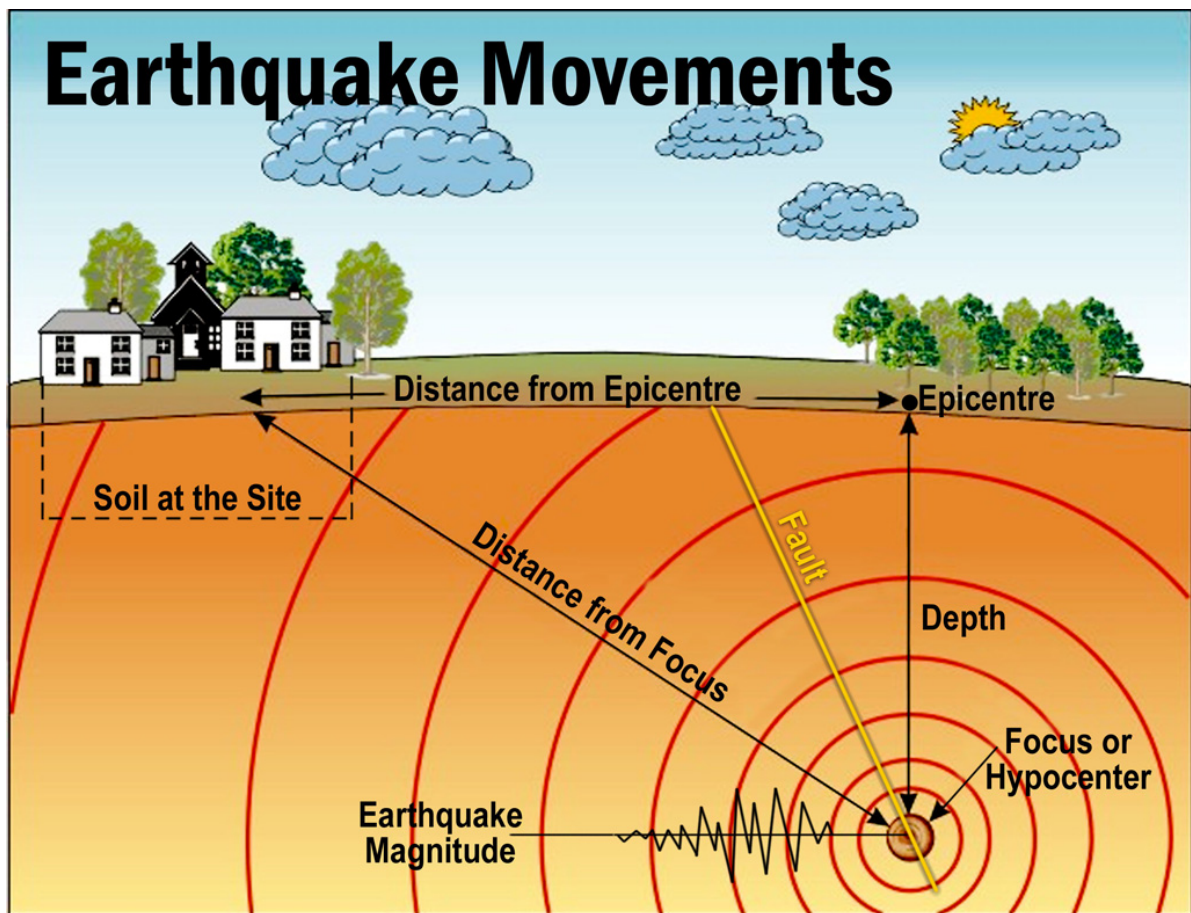


Figure 21 – The seismic waves origin

The magnitude of the earthquake: bigger earthquakes produce more shaking. A powerful earthquake can cause widespread ground shaking, impacting a large number of buildings at once. Also minor earthquakes can cause severe damage to a large number of structures.

Location: The louder the shaking, the closer you are to the epicenter. Structures near the epicenter of a powerful earthquake may sustain significant damage, including partial or complete collapse (Figure 21).

The elements that make up the structure's subsurface: Soft soils amplify the

trembling, while rocks do not. Any soils serve as seismic wave buffers, inhibiting some frequencies while amplifying others.

The intensity of a seismic occurrence is one of the most critical parameters to consider. It may be dependent on seismic effects (qualitative approach) or instrumental measurements (quantitative approach). The seismic strength is calculated based on structural destruction, human life loss, geomorphological changes, and other factors. Mercalli, Cancani, and Sieberg proposed one of the most widely used macro-seismic scales [25]. It's built on a 12-intensity-degree definition that describes a standard registered influence.

The seismic strength can be described using a parameter called magnitude, which takes into account the geometric and energetic characteristics of the earthquake source. The earthquake magnitude M is a quantity that represents the magnitude of the earthquake itself, and is different from the seismic intensity that represents the degree of shaking at a certain point. Historically, the magnitude was taken from the common logarithm of the maximum amplitude recorded by a standard seismograph 100 km from the epicenter. It is determined by using the correction formula for the distance from the epicenter and the type of seismograph and the measurement method of ground shaking (displacement or velocity).

3.2.4 Measuring instruments

Seismographs are instruments that monitor and quantify earthquakes. Vibrations caused by the cracking of the earth's crust are emitted from the point of fracturing during an earthquake. Seismographs pick up and register these vibrations, which are then observed. Land vibrations that are too small to be sensed by humans may be detected by these sensitive instruments. Seismographs create a graphic drawing called a seismogram (Figure 22). The seismograph amplifier can be located at a distance and communicate with the recording unit via radio or tele-

phone. After that, signals from networks of remote stations may be telemetered to a central location for continuous recording and study.

During earthquakes, the rotation of the Earth is determined in reference to any object that is not affected by the ground movement. Electromagnetic seismographs make up the majority of seismographs. The ground is a large magnet, and the outer box holds several rolls of wire. Weak electrical signals are generated in the wire rolls as the box moves relative to the magnet. These signals are either electronically amplified and sent to a paper sheet recording unit (older systems) or sent in real time to processing centers by satellite, telephone line, or radio.

If they migrate across the Earth, high frequency seismic waves weaken. As a result, seismographs used to monitor distant earthquakes must be alert to a different frequency of ground movement than those used to observe local earthquakes. Local earthquakes are recorded using short-period seismographs, in which the waves hitting the seismographs are both quite quick and near together. Long-period seismographs detect and monitor distant earthquakes by responding to lower frequency waves. Both functions are performed by modern broadband seismographs.

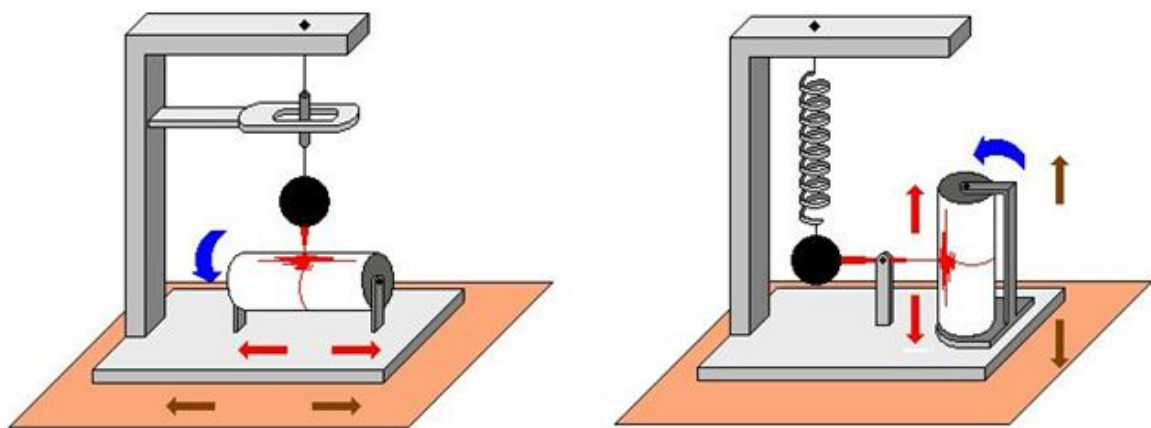


Figure 22 – A classic seismograph

Three perpendicular directions must be determined to achieve a complete picture of the Earth's movement. As a result, seismographs are often deployed in threes. Each seismograph tracks one of the ground motion elements, such as North-South, East-West, or vertical motion.

Seismograms are used to detect and measure the frequency of earthquakes. Seismologists can measure the distance between the seismograph and the earthquake's origin by calculating the time interval between the presence of classes of P and S waves. From the amplitude of the waves on the seismogram and the distance between the earthquake and the seismograph, they will calculate the size. The location of an earthquake's source can be calculated when three or more stations register it. A network of seismographs is set up in seismically active areas in order to detect even very mild tremors.

A special code of three to five letters identifies each station. A three-letter code is applied as a suffix to the station code on seismic plots to denote the features of the seismograph signal to the seismologist. ".BHZ" denotes a high gain wideband seismograph aligned along the Z axis, measuring up and down movements of the Earth. The code ".EHZ" denotes a seismograph with a very short duration and high gain that is also aligned along the Z axis. Since the speed at which signals pass across the crust is known, we can use trigonometric relationships to calculate the precise position of the earthquake based on variations in the arrival time of the same signal at many stations.

Seismographs are sensitive enough to sense even the tiniest waves in the Earth. Wind, moving cars, big vehicles, and fires from mining and manufacturing operations, for example, can all be detected by stations. The majority of the time, these are very weak signals picked up by just a few stations in the region. On the other hand, big blasts such as those caused by nuclear testing, will be observed by a large range of stations.

The seismic stations signals are sent to the Earthquakes data centers through different electronic means. The data recorded by its seismogram is mainly transmitted by one or more satellites, the terrestrial microwaves, UHF radio, telephone modem and internet to each station via its own computer system.

Starting with the P waves, which vibrate at a higher frequency and in the same direction as the energy path, they are more visible in vertical traces ("V") and east-west directions ("E"). The S waves move slower than the P waves, so they arrive after the P waves. These waves vibrate in a direction that is perpendicular to the energy's arrival path.

3.2.5 Seismic data structure

A SEED file is a data file saved in the Standard for the Exchange of Earthquake Data (SEED) format, a common data format used in the field of seismology. It contains time series values, instrument metadata, or both. Time series values are a collection of waveform time intervals recorded by a seismic instrument. Instrument metadata includes information about the response of the instrument and the physical location of the station. SEED files are created by seismic instruments used to monitor seismic movements on Earth. The SEED format is supported by the Federation of Digital Seismograph Networks (FDSN).

SEED files containing time series values and metadata are divided into two parts. The part containing the time series values is known as "miniSEED". The part containing the metadata is called "Dataless SEED". If a SEED file contains only time series values, it is called a miniSEED file. If a SEED file contains only metadata, it is known as a SEAT Dataless file.

MiniSEED is a format used for storing time series. It is a subset of the SEED format used by IRIS and does not contain additional metadata. Mseed files are

composed of a sequence of records, typically 512 or 4096 bytes long, each containing part of a continuous series of measurements.

3.3 Seismic detection using Hadoop MapReduce

In recent years, processing data using traditional tools has become a difficult task due to the huge amount of available data. Hence the need of new tools and frameworks that facilitate and accelerate data processing. Big Data tools have become widely used in many fields, including Seismology. Where Given the Flexibility, Scalability and the ability of the Big Data tools to accelerate parallel processing of massive amounts of data, geophysicists became more and more dependent on these tools. For example, [26] cross correlated a global dataset consisting of over 300 million seismograms. This required 42 days using a conventional distributed cluster. By re-architecting the system to run as a series of MapReduce jobs on a Hadoop cluster they achieved a factor of 19 performance increase on a test dataset. [3] used Hadoop and Spark to perform a large-scale calculation of seismic waveform quality metrics, and compared their performance with that of a traditional distributed implementation. They found that both Spark and MapReduce were about 15 times faster than the traditional distributed implementation. On the other hand, based on processing 43 TB using MapReduce and Spark, they predicted that for a dataset of 350 TB, Spark running on a 100 node cluster would be about 265 times faster than traditional implementation.

[27] have developed a big data platform, which is built upon Apache Spark, to facilitate the work of geophysicists in interpreting and analyzing large volumes of seismic data with scalable performance. [28] tried to answer the question that if Apache Spark is scalable to process seismic data with its in-memory computation and data locality features by using typical seismic data processing algorithms. [29] has adopted Hadoop platform to manage, store, and to analyze seismic data, and by

using MapReduce to integrate the data in the HDFS to ensure that they are analyzed and processed quickly. [30] explored earthquake prediction using Spark distributed computing framework and H₂O library for cluster computing. In order to predict the maximum magnitude in the coming seven days, the authors applied regressors to a 1 GB dataset catalog.

3.3.1 Short Term Average to Long Term Average

The Short Term Average to Long Term Average (STA/LTA) is a trigger algorithm used for seismic detection. It is widely used in many processing software of the weak-motion seismic networks, as well as in portable seismic recorders, it may also be useful in many strong motion applications. It improves the detection of weak earthquakes and decreases the number of false detections triggered by seismic noise. A decreased number of false triggers and trigger's selectivity minimize the work of analysts [5].

The STA/LTA algorithm calculates the average of the absolute amplitude of a seismic signal in two consecutive moving-time windows (Figure 23). The short time average (STA) represents the current average of a short duration over which an event could occur, while the long time average (LTA) represents the previous average of a longest duration to assess the seismic noise.

The STA/LTA trigger algorithm requires three main parameters, the short term duration, the long term duration and the trigger threshold value α . When the current average is greater than the previous average, the ratio STA/LTA exceeds a pre-set threshold value α (equation 3.1), and an event is declared. An other parameter can be added, which is the dettrigger threshold. When the ratio STA/LTA falls below the dettrigger threshold β (equation 3.2), the end of the seismic event is pointed out.

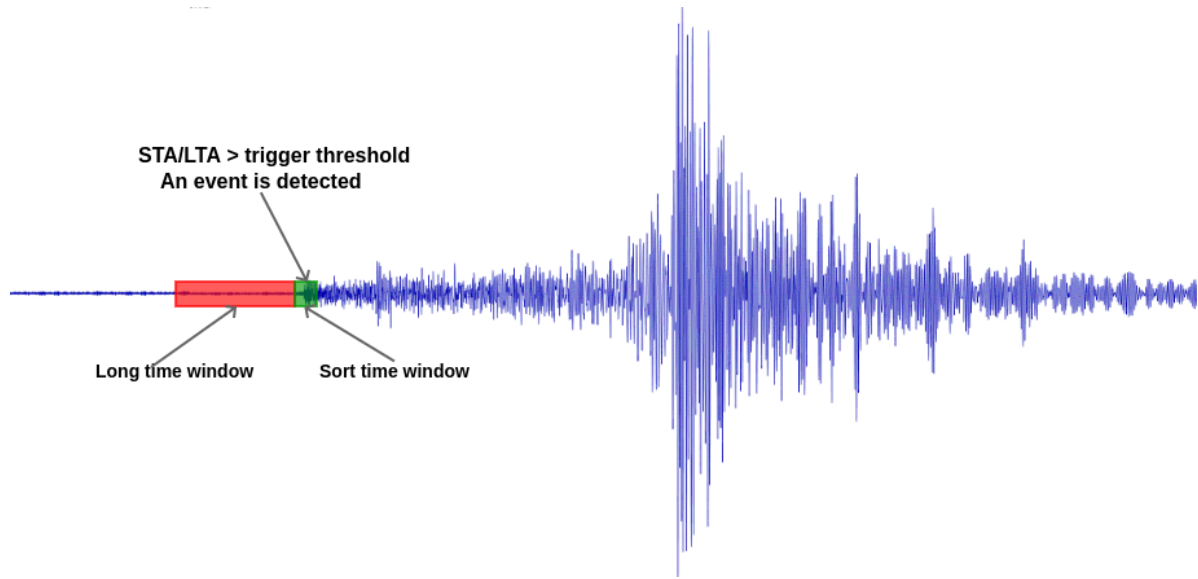


Figure 23 – STA/LTA algorithm

$$STA/LTA > \alpha \quad (3.1)$$

$$STA/LTA < \beta \quad (3.2)$$

The STA/LTA trigger algorithm misses some true seismic events and detects some false events. However, it is suitable for detecting local, micro and distant earthquakes.

3.3.2 Method

The dataset used in this work is from a seismological network called XB, a code assigned by the Federation of Digital Seismographic Networks (FDSN) archive to provide uniqueness to seismological data streams [31]. The XB seismological network was deployed in both Morocco and Spain, in the frame of the Project to Investigate

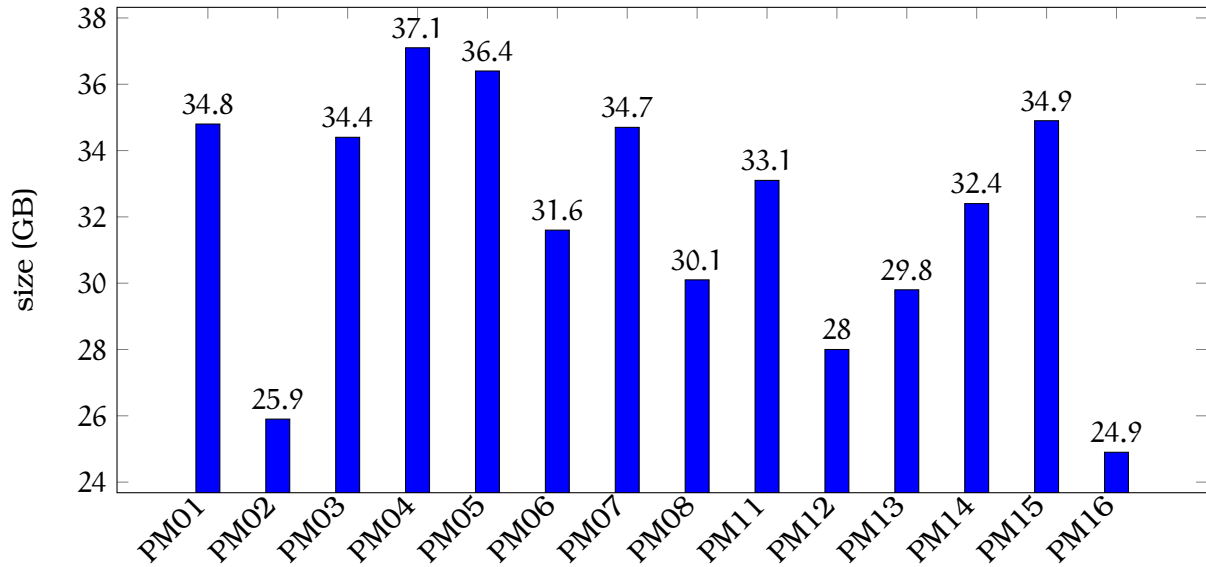


Figure 24 – Size of Avro files of the 14 PM Stations

Convective Alboran Sea System Overturn (PICASSO), between the time period 2009 to 2013. It contained 93 seismic stations (labeled as PICASSO Spain (PS) and PICASSO Morocco (PM), of which 44 seismic stations were installed in Morocco. Figure 25 shows the positions of the XB network.

The acquired time series data are generally of good quality due to the excellent seismological instrumentation deployed in a careful site location. The seismic data are SEED files (The Standard for the Exchange of Earthquake Data), obtained from the Incorporated Research Institutions for Seismology Data Management Center (IRIS DMC) and offering a large amounts of a continuous seismic data corresponding to many seismic stations around the world. SEED format is measured at one point in space and at equal intervals of time[32].

In order to use the downloaded SEED files in the Hadoop platform, we chose to use Apache Avro data serialization system which provides a compact, fast and

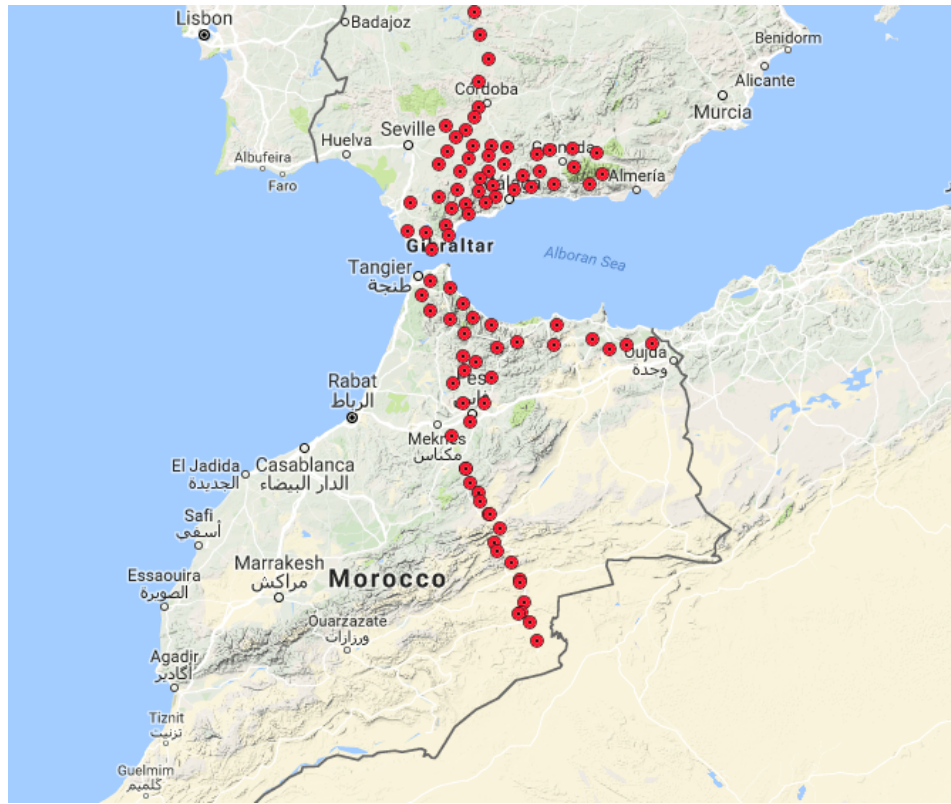


Figure 25 – XB seismic network

binary data format. According to [3], AVRO provided the best API and portability across Big Data tools, as well as other features such as compact serialized size.

To test our implementation, we used a dataset of 14 stations from PM01 to PM16. The stations PM09 and PM10 were excluded from our study because of downloading problems. Figure 24 shows the size of the Avro files corresponding to every seismic station.

	master nodes	worker nodes
Count	1	5
Storage	100 GB	900 GB
Memory	5 GB	25 GB
Cores	2	5

Table 3.1 – Hadoop cluster's configuration

We used Hadoop framework to process our dataset across a cluster of commodity hardware. The configurations of the cluster used in this work is given in table 3.1. After configuring our cluster, the Avro files were sent to it using the Command Line Interface (CLI) and Hadoop takes care of storing and duplicating data into the different nodes.

In order to find seismic events and the stations that simultaneously trigger them, we used two MapReduce functions. The first one for detecting seismic events at a given station by implementing the STA/LTA algorithm, while the second function will search for the events detected at the same time in more than one station.

Algorithm 1: The first Map algorithm

```

Function map1 (key, value):
    data = getDataFromBlock();
    initializ(Trigger_threshold, Detrigger_threshold, STA_window,
    LTA_window, STA_average, LTA_average,
    ratio=STA_average/LTA_average, channel, station)
    for  $x$  in data do
        update( STA_window, LTA_window, STA_average, LTA_average,
        ratio)
        if  $ratio \geq Trigger\_threshold$  then
            set(eventStart)
        end
        if  $ratio \leq Detrigger\_threshold$  then
            set(eventEnd)
            context.write(key=station,
            value=eventStart+eventEnd+channel);
        end
    end
end

```

In the first map function, we defined two windows, one for calculating the Short Time Average (STA window) and the other for the Long Time Average (LTA window). The two windows are slid by one sample and the average of each window is calculated, then the above averages are used to calculate the STA/LTA ratio. We

consider an assumed seismic event in a single component when the ratio exceeds a defined trigger threshold. We save the time at which the event occurred and we continue sliding the windows and calculating the ratio until it falls below the dettrigger threshold. At this moment we declare the end of the assumed seismic event (algo. 1). Thus, the first map function returns the station name as a key and the value is a concatenation of the start time, end time and the channel name.

In the first Reduce function the list of value for each key(station) is sorted by the start time, then the whole list is treated so that if the event is detected in only one channel it is considered as a noise. Otherwise, if it is detected in more than one channel in the same station this event is considered as an assumed earthquake and the reduce function returns one as a key and a concatenation of the start time, end time, the channel and the station name as a value (algo. 2). Those <key, value> pairs will be treated by the second MapReduce function.

The second Map function just read then returns the data returned by first reduce function (algo. 3).

The second Reduce function work on the output of the first reduce function, it returns the events detected in more than one station. The values returned by the first MapReduce function are sorted by detection time and each event is compared with the events that follow. When the interval of time between two events is lower than a defined value, the function check if the stations where the events occurred are neighbors. For that, a table of nearest neighbors is defined. That table contains the nearest neighbors for each station (algo. 4).

Algorithm 2: The first Reduce algorithm

```

Function reduce1 (key, valueList):
    sort(valueList)
    x=0
    while  $x < \text{valueList.size}() - 1$  do
        t1 = valueList(x).geteventStart()
        chan1 = valueList(x).getchannel()
        value = valueList(x)
        count = 1
        y = x+1
        while  $y < \text{valueList.size}()$  do
            t2 = valueList(y).geteventStart()
            chan2 = valueList(y).getchannel()
            if  $t2 - t1 < \text{delay}$  and  $\text{chan1} \neq \text{chan2}$  then
                value += valueList(y)
                count++
            else
                x=y-1
                break
            end
            y++
        end
        if  $\text{count} > 1$  then
            context.write(key=1, value=(value, key))
        end
        x++
    end
end

```

Algorithm 3: The second map algorithm

```

Function map2 (key, value):
    context.write(key, value)
end

```

Algorithm 4: The second Reduce algorithm

```

Function reduce2 (key, valueList) :
    sort(valueList)
    x=0
    while  $x < \text{valueList.size}() - 1$  do
        t1 = valueList(x).geteventStart()
        value = valueList(x)
        count = 1
        y = x+1
        while  $y < \text{valueList.size}()$  do
            t2 = valueList(y).geteventStart()
            if  $t2 - t1 < \text{delay}$  then
                station1 = valueList(x).getstation()
                station2 = valueList(y).getstation()
                if areNeighbours(station1, station2) then
                    value += valueList(y)
                    count++
                end
            else
                x=y-1
                break
            end
        end
        if count > 1 then
            context.write(key=1, value=value+key)
        end
    end
end

```

3.3.3 Results and Discussion

To test our implementation we used a dataset consisting of data from 14 stations. To find out the improvement realized by MapReduce we considered a traditional implementation as a reference and compared its results with that of MapReduce.

The time needed for processing data by the reference implementation was almost 13 hours and half, while MapReduce needed nearly 9 hours to accomplish the tests. The MapReduce implementation decreased the processing time by 34%. As the

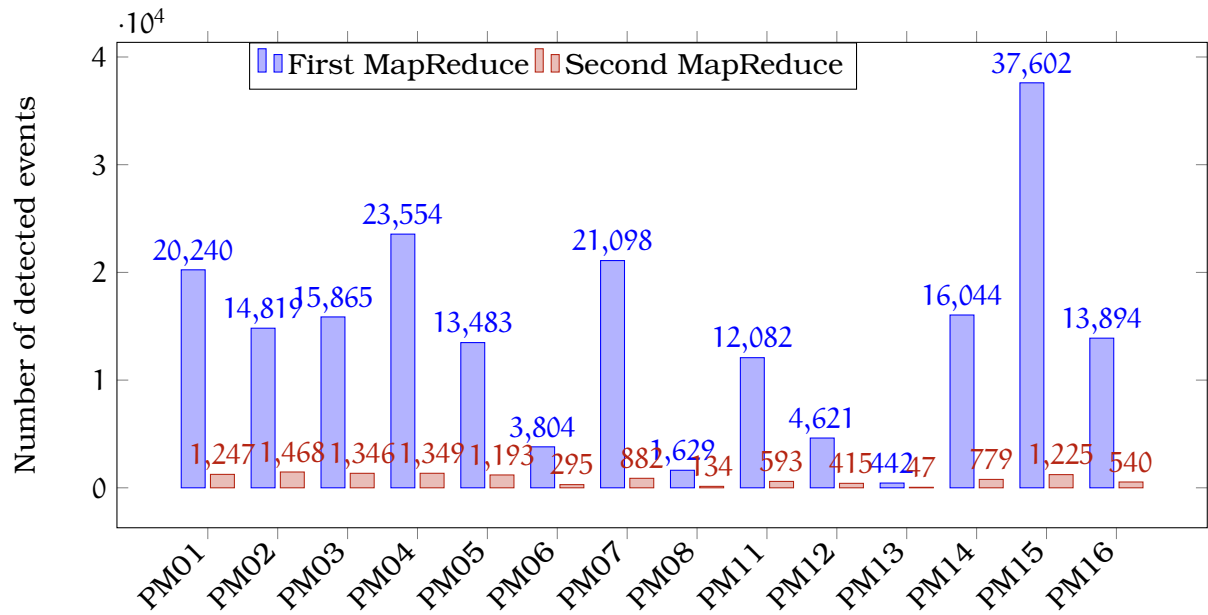


Figure 26 – Number of seismic events detected per Station

dataset become larger the factor will increase too. Parallel processing allows multiple processors to work on these divided tasks, so that they run entire programs in less time.

By applying the STA/LTA trigger algorithm to the seismic data corresponding to the XB seismological network in morocco, we were able to detect 199177 events in the first MapReduce function. By applying the second MapReduce, the number of events detected in more than one station decreased to 11513 events. Figure 26 shows the number of events in each seismic station.

3.4 Conclusion

Seismology is a branch of science that investigates seismic activity from a variety of origins tectonic, volcanic, etc., including the transmission of seismic waves across the earth's interior and their environmental impacts, such as ground shaking and

tsunamis. This field arose from a need to learn more about earthquakes and attempt to forecast their occurrence. This is done by examining a region's seismic history in order to determine the characteristics and frequency of potential earthquakes. The most important principles in seismology were briefly covered in this chapter.

We have shown also in this chapter the usability of Hadoop MapReduce for seismic detection, particularly using Short Term Average to Long Term Average (STA/LTA) algorithm. We compared MapReduce implementation performance with that of a traditional implementation. The results show that time needed for processing goes from almost 13 hours and half using the traditional implementation to nearly 9 hours by using MapReduce. So MapReduce decreases the processing time needed for processing large amount of seismic data.

We implemented STA/LTA trigger algorithm, which is widely used in seismic detection, using Hadoop MapReduce. This implementation allows to find out how effective it is in this type of tasks as well as to accelerate the detection process by reducing the processing time. We tested our implementation on a seismological dataset of 14 broadband seismic stations and compare it with the traditional one. The results show that MapReduce decreased the processing time by 34% compared to the traditional implementation.

Deep Learning for Earthquake Detection

4.1 Introduction

Automatic earthquake detection is widely studied to replace manual detection, however, most of the existing methods are sensitive to seismic noise. Hence, the need for Machine and Deep Learning has become more and more significant. Regardless of successful applications of the Fully Convolutional Networks (FCN) in many different fields, to the best of our knowledge, they are not yet applied in earthquake detection. In a first part of this chapter, we propose an automatic earthquake detection model based on FCN classifier. We used a balanced subset of STanford EArthquake Dataset (STEAD) to train and validate our classifier. Each sample from the subset is re-sampled from 100Hz to 50Hz then normalized. We investigated different, widely used, feature normalization methods, which consist of normalizing all features in the same range, and we showed that feature normalization is not suitable for our data. On the contrary, sample normalization, which consists of normalizing each sample of our dataset individually, improved the accuracy of our classifier by

~16% compared to using raw data. Our classifier exceeded 99% on training data, compared to ~83% when using raw data. To test the efficiency of our classifier, we applied it to real continuous seismic data from XB Network from Morocco and compared the results to our catalog containing 77 earthquakes. Our results show that we could detect 75 out of 77 earthquakes contained in the catalog.

In order to get better results using deep learning, more data is needed, thereby come data augmentation to the scene, which helps to generate new data. In a second part of this chapter, we are using data augmentation with Generative Adversarial Networks to enhance a time series classification accuracy. In order to test our method, we used the Earthquakes dataset from the UCR archive. The experimental results show that the classification accuracy increases when balancing the dataset.

4.2 Earthquake detection using Convolution Neural Networks

Earthquake detection requires discriminating real earthquakes from noise signals, which makes it a classification problem. Earthquake detection is a very crucial and challenging phase in seismic processing, especially for single station-based detection, because every station records a very wide range of non-earthquake waveforms. Manual detection is a time consuming work due to the huge amount of seismic data, therefore, automatic earthquake detection is essential and widely studied.

A large number of automatic earthquake detection methods exist [33], some of them are time domain methods, such as the short term average to long term average (STA/LTA), which is the most used in seismic stations. Other time domain methods are used, such as the maximum likelihood detector [34], envelope-based detector [1], and modified data envelope detector [35]. On the other hand, some frequency domain methods are based on the Power Spectral Density (PSD) [36] and the Walsh

transform [37]. However, most of the existing methods are sensitive to noise and suffer from false and missed detections [38].

In recent years, methods based on Machine and Deep Learning have shown great potentials, especially Artificial Neural Networks, which are widely used in seismic detection [39; 40; 41]. The Convolutional Neural Networks (CNNs) known as very successful in the computer vision area become more and more popular in seismic area [42; 43; 44; 45]. Recurrent Neural Networks (RNNs), another architecture of Neural Networks known to be suitable for many time-series applications such as text to speech and voice recognition [46], are also used in seismology [47; 48; 49].

Unsupervised clustering methods are also used in seismology. They can cluster seismic samples into different clusters without prior knowledge of labels. Different clustering methods are used in many seismic studies, such as k-means [50; 51], Deep Convolutional Autoencoders [52], and Self-Organizing Maps [53; 54; 55].

Fully Convolutional Networks (FCN) [56] are a Neural Network architectures that have been successfully applied in many different fields, such as image segmentation[17; 57], medical image analysis [58; 59], character recognition [60], time-series classification [61; 62; 15] and also in seismology; for earthquake localization, by taking a window of three-component waveform data from multiple stations and predicting the earthquake location with a 3D image [63], and for fault detection, where the FCN model extracts fault features from synthetic seismic data and recognize the locations of faults with an accuracy of ~97% [64]. Despite their higher achievements, to the best of our knowledge, FCN had not yet been applied to seismic detection.

In this study, we describe the application of FCN for earthquake detection using seismic waveforms from a single seismic station. The basic of the earthquake detection problem is turned into a classification problem by using a subset of STanford EArthquake Dataset (STEAD) to train our classifier. Our approach does not require

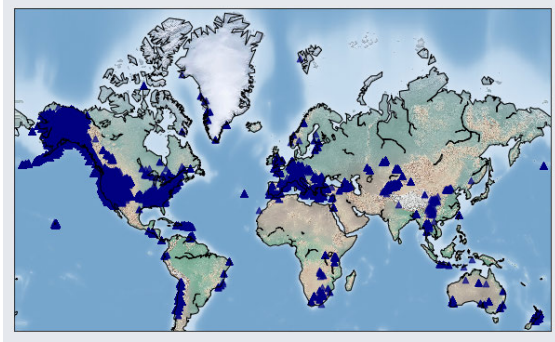
a feature extraction technique, which makes it independent of the choice of sensitive features. We tested the effectiveness of our classifier by applying it to real continuous data From the XB seismic network implemented in morocco between 2009 and 2013 [31].

In the following, we first describe the dataset used to train our classifier and the real continuous dataset used for testing. Then, we present the method and the steps applied in the training process. Finally, we describe the results and discuss the performance of our classifier Using real continuous seismic data.

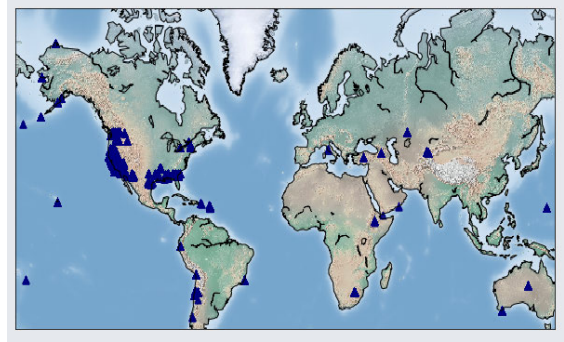
4.2.1 Data description

The STanford EArthquake Dataset (STEAD) [65] is a large-scale and global dataset that contains two waveform classes; seismic noise and local earthquake waveforms, which are recorded at local distances (within 350 km of earthquakes). STEAD comprises about 1.2 million waveforms, recorded by worldwide located seismometers, resampled at 100Hz, and have 60 seconds duration (6000 features). Local-earthquakes class contains about 1 050 000 three-component seismograms associated with ~450 000 earthquakes that occurred between January 1984 and August 2018 (Figure 27(a)). The seismic noise class contains about 100 000 waveforms that have been recorded since 2000 in the United States and Europe (Figure 27(b)).

The earthquake waveforms are requested from continuous time-series archived at the Incorporated Research Institutions for Seismology Data Management Center (IRIS DMC). Three types of arrival statuses exist, "Manual" picks; picked manually by human analysts, "automatic" picks; measured by automatic algorithms and "autopicker" that are determined using an AI-based model. STEAD is provided as



(a) Locations of seismic stations recording earthquakes



(b) Locations of seismic stations recording seismic noise

Figure 27 – Distribution of seismic stations used to record earthquakes (a) and noise (b).

individual arrays containing three waveforms that correspond to three-component seismograms, each waveform has 6000 features.

The XB network used to test our model was deployed in both Morocco and Spain, in the frame of the Project to Investigate Convective Alboran Sea System Overturn (PICASSO), from 2009 to 2013. The XB network contained 93 seismic stations labeled as PICASSO Morocco (PM) and PICASSO Spain (PS). Figure 28 shows the stations of the XB network, where 44 stations were installed in Morocco. We used data of January 2011 from 19 stations, measured by High gain Broadband (BH) seismic instruments and sampled at 50Hz, to test our model.

4.2.2 Training with the Fully Convolutional Networks

To train our model, we choose a subset of STEAD measured by BH seismic instruments, since we have only BH waveforms from XB network. We found 7874 unique noise waveforms of BH type in STEAD. In order to create a balanced dataset, we extracted the same quantity of waveforms from the earthquake class. because classification is affected by imbalanced datasets and resulting a reduction in accu-

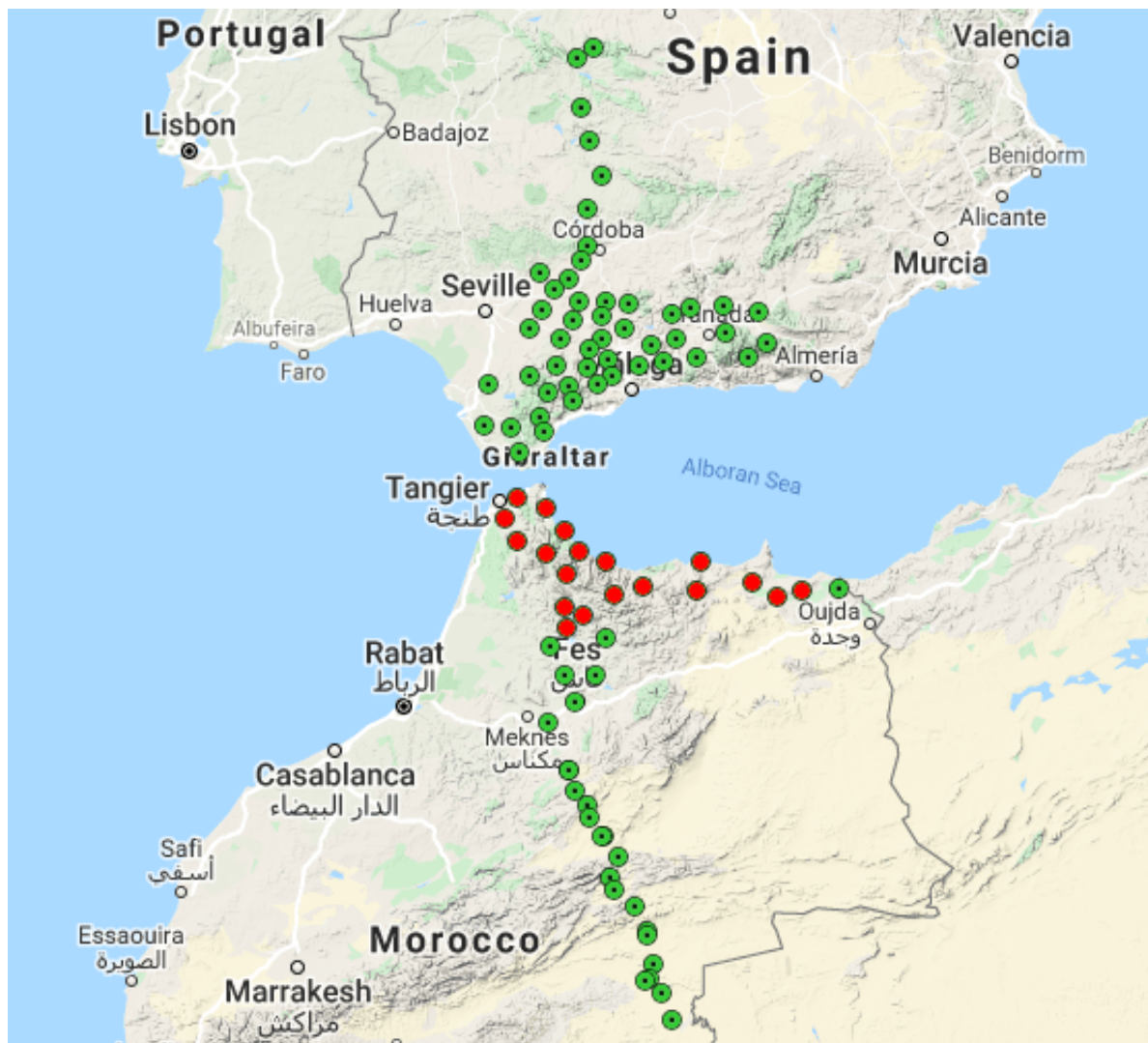


Figure 28 – XB seismic network installed in both Morocco and Spain, it contains 93 seismic stations, from which 19 stations are used in this study and are colored in red.

Table 4.1 – Train/Validation/Test subsets distribution used in our study.

Training-set	Validation-set	Test-set
12000	3000	748
(6000 from earthquake class, 6000 from noise class)	(1500 from earthquake class, 1500 from noise class)	(374 from earthquake class, 374 from noise class)

racy as shown by [66]. The selected waveforms are associated with a wide range of earthquake sizes from magnitude 0 to magnitude 6.3. Earthquakes were recorded within 330 km of the earthquakes, are mainly shallower than 210 km and have Signal to Noise Ratio (SNR) between -5 and 100 decibels.

Our dataset is comprised of 15 748 samples and divided into train/validation/test subsets as shown in Table 4.1. The portion of the test-set is small because we will test our model on real continuous data from the XB seismic network. The

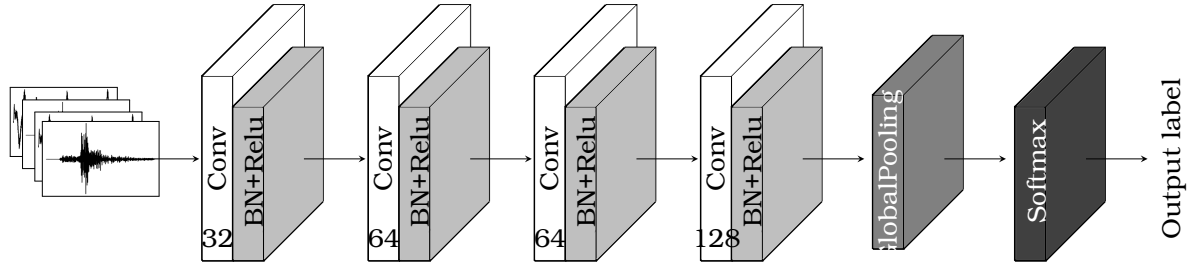


Figure 29 – The FCN architecture consists of 4 convolutional layers followed by Batch-Normalization (BN) then a Rectified Linear Units (ReLU) activation function, finally global-pooling and softmax layers.

samples used in STEAD are 60 seconds waveforms sampled at 100Hz. Since we are applying our model to data from XB network that is sampled at 50Hz, we resampled our dataset to 50Hz, so that every sample have 3000 features instead of 6000.

The Fully Convolutional Network classifier used in this study is comprised of 4 convolutional layers with different filter numbers and sizes (Figure 37), followed by

batch normalization that normalizes the output of the convolution layer and a ReLU activation function, which enables better training of deeper networks, compared to other activation functions [67], then a Global Pooling layer that reduces the amount of parameters in the network to an output prediction for the model. Finally, since the output is One Hot Encoded, a softmax function is placed in the output layer that normalizes the output into 2 probabilities corresponding to belonging to the two classes earthquake and noise. The adaptive moment estimation algorithm (Adam) is used as optimizer for our classifier.

The classifier is trained to distinguish between earthquake and noise signals using the STEAD subset described above. The training/validation subsets were randomly split using a 5-fold cross-validation. The training was performed on 100 epochs, where each epoch is a complete pass through the entire training dataset, with early stopping enabled, which stop the training when the loss (error) does not decrease during training. We used a learning rate decay, where the learning rate is reduced by a factor of 10 once learning stagnates for a number of epochs. The predictions were compared to the real classes then the loss and accuracy are calculated.

Normalization is one of the most used data preparation techniques in deep learning, because features often have different ranges of values, which make the training process takes a long time to converge. Feature normalization and standardization are the most used methods. To select the best normalization method, we compared different methods against raw data (without applying any normalization method) and selected the one that gave the best accuracy on the training/validation data. The methods applied to our input data are the following:

- MinMax: Transform features by scaling each feature individually between zero and one.

- MaxAbs: Scale each feature by its maximum absolute value such that the maximal absolute value of each feature in the training set will be 1.0.
- Standard: Standardize features by removing the mean and scaling to unit variance.
- RobustScaler: Removes the median and scales the data according to the quantile range independently on each feature.

4.2.3 Results and discussions

In this section, we will present and discuss the effect of using different normalization methods and batch sizes on the classifier accuracy. We investigated the effect of normalizing features on the model accuracy and compared it against using raw data. We conducted many experiments using the same training process for the normalization methods described above. Table 4.2 shows the mean accuracy of 5-fold cross-validation. We can see that MinMax, MaxAbs and Robust normalizations decrease the model accuracy compared to raw data, while standardization improves slightly the accuracy. Overall, we see that normalizing the input features did not bring a big improvement to our model, so we suspected the feature normalization to be not suitable for our data. Therefore, we tried to normalize our data per sample instead of normalizing per feature. By normalizing per sample, we mean that each sample of our dataset is normalized individually. We reported the results in Table 4.3.

By using sample-normalized data, we can clearly see an improvement of the accuracy compared to feature-normalized and raw data. All the methods improved the accuracy without exception, compared to feature-normalized, especially the MinMax method, which is improved by ~35%. The sample-standardization method made the best accuracy over the other methods, it reaches 99% on the training data, with

Table 4.2 – The accuracy for different feature normalization methods.

	Raw data	Standard	MinMax	MaxAbs	Robust
Training	0.838	0.858	0.639	0.822	0.778
Validation	0.785	0.885	0.619	0.683	0.718
Test	0.611	0.894	0.502	0.513	0.657

an improvement of $\sim 16\%$ and $\sim 14\%$ compared to raw data feature-standardization respectively.

As seen in Figure 30(b), when standardizing per feature, the range of earthquake classes is very large compared to that of noise classes, which makes no difference with raw data (Figure 30(a)), except for the scale of the signals. It can be observed from Figure 30(c) that both earthquake and noise samples have close ranges when standardized per sample. Hence, the classifier is forced to classify samples based on their shape instead of their amplitude. In the rest of our tests, only the sample-standardization method will be presented, since it outperformed the other methods.

Different batch sizes are investigated, where each batch is a subset of signals given to the network at once. Figure 31 shows an example of the evolution of the loss function during the training process and it is clear that our classifiers converge as the training progresses. We can see that for larger batch sizes, the training loss is bigger and the validation loss is smoother, because large batch sizes are less sensitive to outliers, and converge slower than small batch sizes as stated by other studies [68].

Figure 32(a) shows the accuracy during the training process, we can clearly see that larger batch sizes have lower accuracies compared to smaller batch sizes. While for the validation dataset (Figure 32(b)), larger batch sizes tend to be slower and more stable.

Table 4.3 – The accuracies for different per-sample normalization methods.

	Raw data	Standard	MinMax	MaxAbs	Robust
Training	0.838	0.990	0.986	0.985	0.982
Validation	0.785	0.987	0.985	0.986	0.916
Test	0.611	0.998	0.995	0.993	0.973

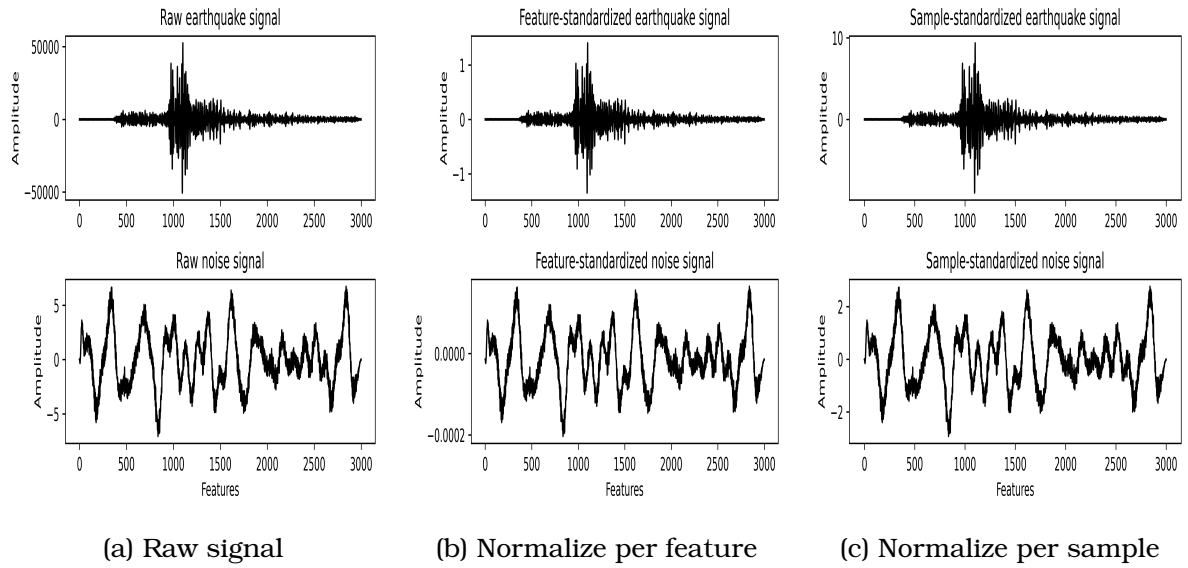


Figure 30 – Standardization effect for the same two signals, in top an earthquake signal while in bottom a noise one. Figure (a) shows raw signals, in (b) both signals are standardized per feature, (c) shows the sample-standardization version for both signals.

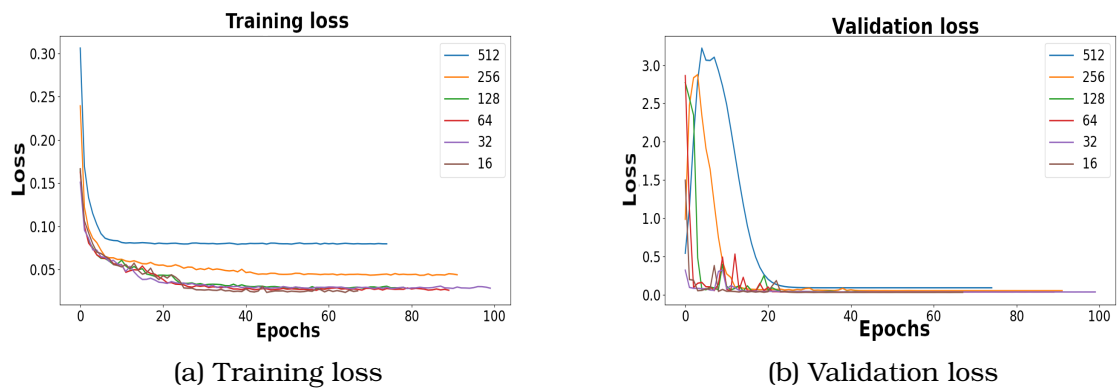


Figure 31 – Model loss per batch size.

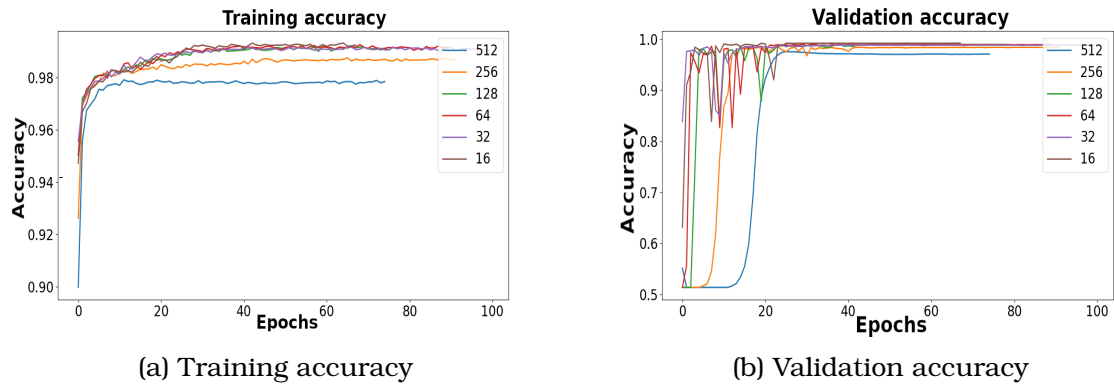


Figure 32 – Training and validation subsets accuracies per batch size. Large batches tend to have lower accuracies for both subsets.

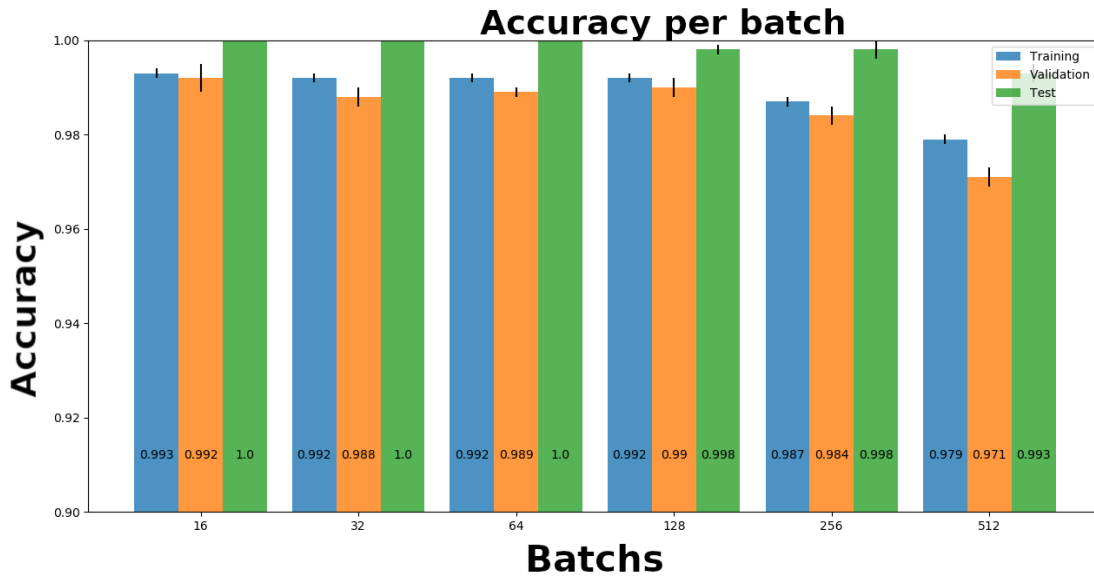


Figure 33 – The mean accuracy per batch size. The best accuracies in train/validation/test subsets are obtained by using a 16 batch size.

Figure 33 shows the mean accuracy of 5-fold cross-validation. The best accuracies in training, validation and test are 99.3%, 99.2% and 100% respectively, obtained by using a 16 batch size. For smaller batches, the accuracy in training reached 100%, but in validation it has fallen to 70%, which means that the model over-fit and can not generalize for new data. The high accuracy in test-set is due

to the small amount of data, because we are interested in testing our classifier on continuous data from XB network.

To check the effectiveness of our best classifier, we tested it on real three-component seismic data from the XB network. The test was applied to data from the first month of 2011, from 19 seismic stations, presented in red in Figure 28. The frequency of the seismic data is about 50Hz, and the input feature, which will be fed to the classifier, is a sliding window of 60 seconds length (3000 features), and the window is moved by 15 sec after each test.

To verify our results, we compared the earthquakes detected by our classifier to a seismic catalog that we have. Our catalog contains 77 earthquakes of magnitude > 2 , located in the region of XB network. By comparing our results with the catalog, we found that our classifier detected 75 out of the 77 earthquakes contained in the catalog. Our analysis shows that our classifier is able to reliably detect local earthquake signals in continuous real data.

4.3 Seismic data Augmentation using Deep Learning

Time Series Classification (TSC) is one of the most challenging problems in deep learning and is encountered in many real-world applications in various domains such as climate [37], medicine [69], and transportation [70]. In recent years many researchers have been studying TSC. [33] provided a simple and strong baseline for TSC with deep neural networks. Their results showed that Fully Convolutional Network (FCN) achieves premium performance compared to other states of the art approaches. [16] proposed the augmentation of FCN with Long Short Term Memory (LSTM) for TSC. The proposed model enhanced the performance of FCN with a nominal increase in model size. [53] presented Earliness-Aware Deep Convolutional Networks (EA-ConvNets), a deep learning framework, for early time series classification (eTSC) which is the problem of classifying a time series after as few measurements

as possible with the highest possible accuracy. They performed a set of experiments on several benchmark data sets, which demonstrated that their method yields significantly better predictions than various state of the art methods designed for early TSC. [54] proposed a novel end-to-end Multi-scale Convolutional Neural Network (MCNN), which unifies feature extraction and classification. They conducted an empirical evaluation with various methods on a large number of benchmark datasets, and show that MCNN achieved superior accuracy performance than other states of the art methods.

[50] proposed a Bi-directional LSTM (BiLSTM) model based on multivariate time series data, where they divided input data into fields to learn the features of those fields. [51] proposed a tree-ensemble method referred to as Time Series Forest (TSF) for TSC, which employs a combination of entropy gain and a distance measure, referred to as the Entrance gain. Their experimental studies show that the Entrance gain improves the accuracy of TSF. [71] proposed a new model based on the deconvolutional networks and Symbolic Aggregate Approximation (SAX) discretization to learn the representation for multivariate time series. They designed a network structure to capture the cross-channel correlation with deconvolution and applied SAX discretization on the feature vectors to further extract the bag of features. [56] used Support Vector Regression (SVR) with linear and sigmoid kernel for computing the Lyapunov exponents of chaotic short time series and validated their technique using generated time series obtained from well-established chaotic maps and with real-world time series such as EEG signals. Their results show that SVR with sigmoid kernel can be efficiently used to model the chaotic data and is more efficient than the linear kernel.

Other approaches used the Collective of Transformation-Based Ensembles (COTE) [60; 17], Reservoir Computing (RC) [57; 61], bag-of-features representation [62] and Word ExtrAction for time Series cLassification (WEASEL), which trans-

forms time series into feature vectors, using a sliding-window approach, which are then analyzed through a machine learning classifier [58; 59].

Data augmentation is the process of generating new data based on existing ones. This idea has been shown to improve the performance in many deep learning problems including TSC problems. [72] studied the effect of Data Augmentation using VAE and GAN on the prediction accuracy of small data sets. Their results show that Data Augmentation can lead to an increase in prediction accuracy of approximately 3%. [73] investigated the application of data augmentation techniques based on the Dynamic Time Warping distance (DTW). Their experiments reveal that data augmentation can significantly improves the deep model's accuracy. [74] proposed new time series data augmentation techniques, where the space in which they are embedded is induced by Dynamic Time Warping (DTW). They demonstrated that their method is particularly useful when the number of available examples is limited.

[75] proposed a data augmentation techniques based on window slicing and warping to improve the performance of classification with Convolutional Neural Networks (CNN) when dealing with small time series datasets and showed that their approach improved the overall classification performance. [76] proposed a novel data augmentation strategy based on duplication, concatenation, and resampling. The results show that the proposed method was superior to window slicing and permutation methods. [77] proposed various data augmentation methods for wearable sensor data such as jittering, scaling, and rotating. Those methods and CNNs are applied to the classification of the motor state of Parkinson's Disease patients and increased the classification performance with 9%.

This section presents a Time Series Classification (TSC) with data augmentation. Data augmentation is done by Generative Adversarial Networks (GANs), while the TSC is performed using Fully Convolutional Network (FCN) [78]. Our model is tested on the imbalanced earthquake dataset from the UCR Time Series Classifica-

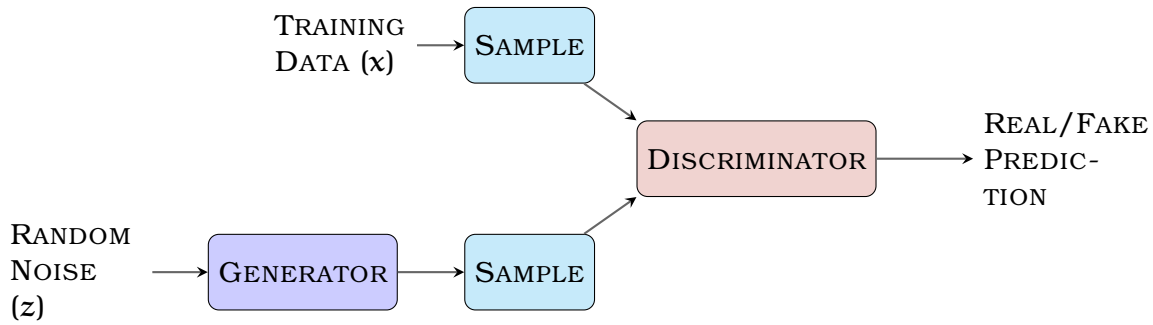


Figure 34 – representation of GANs architecture

tion Archive [15]. In the rest of this chapter, we first provide background materials concerning GANs and the earthquake dataset respectively. Then, we present the proposed method and finally we describe and discuss our results.

4.3.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a type of Neural Networks that were introduced by [64]. It becomes widely used in last years in data generation, especially in the computer vision area. A GAN is distinguished by its two blocks, one is the generator G and the other is the discriminator D , where the two blocks are competing against each other (figure 34).

The role of the generator is to generate new data similar to the real data, by trying to learn the input distribution p_{data} of the real data. The generator takes a simple random variables z and returns the generated data $G(z)$. The discriminator is trying to predict whether the generated data is real or fake. Given an input x , the discriminator returns the probability $D(x)$ that x came from the real distribution.

The discriminator is trained to maximize the probability of assigning the correct label to both real and generated data ($\log D(x)$), while the generator is trained to

minimize $\log(1 - D(G(z)))$. So D and G play the following two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = E_{x \sim p_{\text{data}}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

In other words, the weights of the Generator are updated in order to minimize the probability of the Discriminator to assigning the correct label to both real and generated data, while the weights of the Discriminator are updated so that to maximize that probability, hence the name Adversarial Networks comes.

4.3.2 Earthquake dataset

The earthquake Dataset used in this work is obtained from the UCR Time Series Classification Archive. The earthquake classification problem involves predicting whether a major event is about to occur. The data is taken from the Northern California Earthquake Data Center and each sample is an averaged reading for one hour. The dataset contains two classes, figure 35 shows samples from the training set. A positive case is defined where a major event of over 5 on the Richter scale is not preceded by another major event for at least 512 hours. A negative case is considered where there is an event below 4 that is preceded by at least 20 readings in the previous 512 hours that are non zero.

The dataset contains 368 negative cases and 93 positives. It is divided into a training set and a test set. The training set contains 322 samples where 264 are negative and 58 are positive cases while the test set contains 104 negative and 35 positive cases.

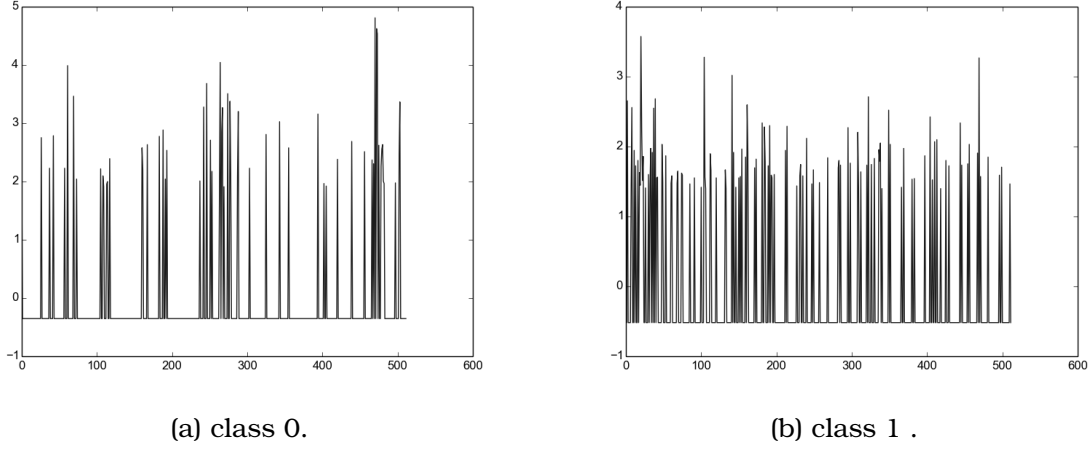


Figure 35 – class samples from the training set

4.3.3 Model Training

The training set is imbalanced and the amount of data from the negative class is nearly 4 times the amount of positive class. We will balance the training set by augmenting the positive class, and when the data is balanced we will augment the two classes simultaneously and see its impact on the classification result. The data augmentation is applied to the training set while the testing set remains untouched.

To perform the data augmentation on our training set, we used a Generator with two fully connected layers followed by two convolutional layers, while the Discriminator is composed of Three convolutional layers followed by a fully connected layer (figure 36).

We trained our GANs on 100 epochs and saved the resulted model that will be used to generate the data. We created an augmented training set by concatenating the generated data and the original training set.

The data classification is performed using the FCN architecture shown in figure 37. The FCN architecture is comprised of three stacked convolutional layers, each

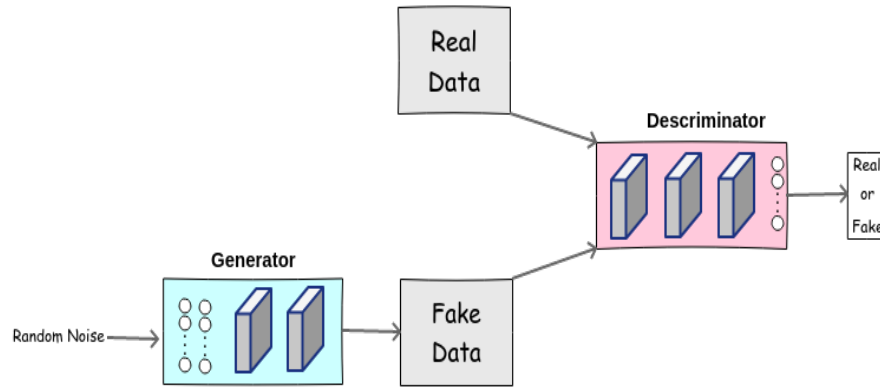


Figure 36 – GAN architecture

layer is followed by a batch normalization and a Relu activation function, then a global average pooling is following the final convolution layer to reduce the number of parameters of the output. A softmax activation function is placed in the output layer to normalizes the output into 2 probabilities corresponding to belonging to the two classes.

The FCN classifier is trained using a training/validation subsets from the augmented training set. The training/validation subsets were randomly split using a k-fold cross validation, with k=5. The training was performed on 100 epochs, where each epoch is a complete pass through the entire training set. We used a learning rate decay, where the learning rate is reduced by a factor of 10 once learning stagnates for a number of epochs.

4.3.4 Results and discussion

In this section, we will present and discuss the effect of augmenting the earthquake dataset on the classifier accuracy. By training our GANs architecture, we generated synthetic data for both positive and negative classes that are similar to the real data from the training set. Figure 38 (a) shows generated data sample and

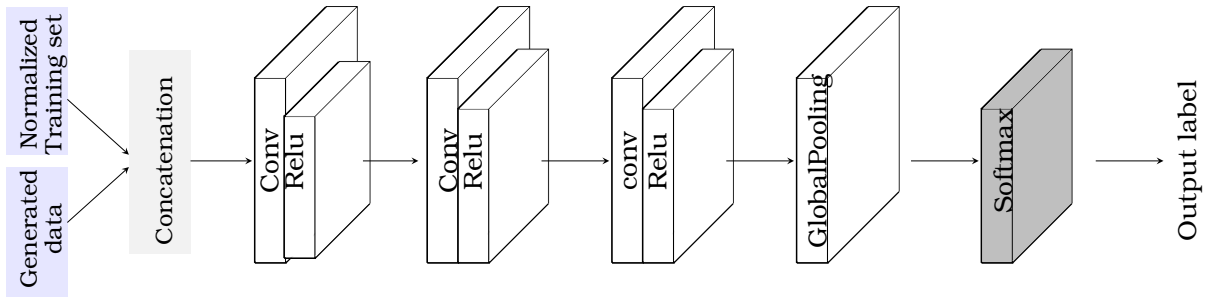


Figure 37 – FCN architecture

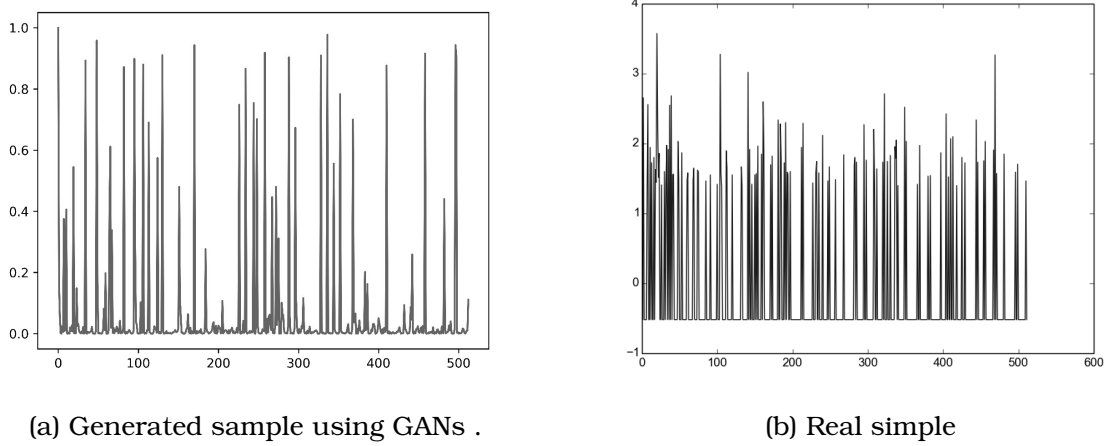


Figure 38 – GANs generated sample compared to real sample

38 (b) shows real sample from training set from the positive class. Generated samples have also peaks similar to real data, with some noise near zero. The difference in scale between the generated and real data is due to the sigmoid function in the end of the GANs architecture. This difference will be handled by normalizing the real data as a preprocessing step.

By concatenating the normalized real training set with the generated data we created our new training set, which will be used to train our FCN. In the aim of picking the best model, we conducted multiple experiments on multiple batch sizes and data distribution values. Figure 39 shows the averaged accuracy using 5-fold

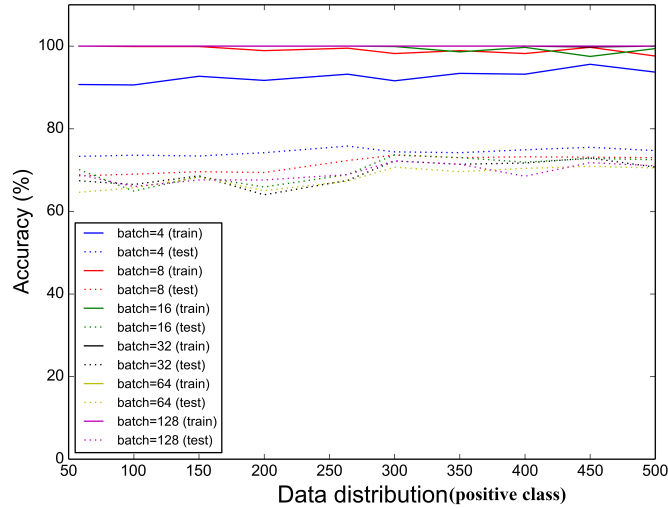


Figure 39 – Experiments on batch size and data distribution

cross validation. We started with the original training set without adding generated data (264 negatives and 53 positives), while in the next four experiments we added data to only the positive class until balancing the two classes (100, 150, 200 and 264 for positive class). As for the last five experiments, we added data to both positive and negative classes (300, 350, 400, 450 and 500 for both classes).

Figure 39 shows that as the batch gets bigger the training accuracy increases while the test accuracy decreases, in other words, it leads to overfitting. We get the best test accuracy (77.69%) when we balanced the training set by adding only data from the positive class with a batch size equal to 4. For the other batch sizes, we see a slight improvement as the training set increases, which means that by balancing the two classes, the network got more samples to learn with and started to recognize positive samples in the test set.

4.4 Conclusion

In this chapter, we have presented a seismic detection model, based on a Fully Convolutional Networks classifier which is trained on STanford EArthquake Dataset (STEAD) and tested on real continuous seismic data. By making a separate standardization for each sample of our dataset, instead of normalizing per feature, the performance of our classifier is increased significantly by $\sim 16\%$ compared to raw data. Our experiments show that the use of small batch sizes is more adequate for our dataset, however, very small batch sizes (8 and lower) make the model over-fit and can not generalize for new data. By applying our classifier to real continuous data from XB network in Morocco, we were able to detect local earthquakes already existing in our catalog. Our method does not require hand-engineered features and is able to discriminate between earthquakes and seismic noise with high accuracy. Our results demonstrated that FCN classifier holds vast promise for making seismic detection more accurate.

In a second part, we used data augmentation to enhance the performance of the earthquake dataset classification problem. We used Generative Adversarial Networks (GANs) to augment the training set, which included less positive classes than negative, then we concatenated the training set with the generated data from GANs to create our new training set that will be used for classification, while the test set remains untouched. We used a Fully Convolutional Networks architecture as a classifier and conducted multiple experiments on multiple parameters such as batch size and data distribution. The results show that the best accuracy on the test set is achieved when the training set is balanced. Since our best accuracy reached 77.69%, we get the best accuracy for the earthquakes dataset compared to other works, where the same dataset has been used [16; 17].

Conclusion and perspectives

In this thesis, we addressed the earthquake detection Problem using Hadoop Framework as well as Fully convolutional Networks (FCN). We first introduced deep learning, presented the major concepts of this field, and we defined some terminologies generally used in deep learning. Then we introduced some big data concepts and technologies, such as Apache Hadoop and Spark.

We have shown the usability of Hadoop MapReduce for seismic detection, particularly using Short Term Average to Long Term Average (STA/LTA) algorithm. We compared MapReduce implementation performance with that of a traditional implementation. The results show that time needed for processing goes from 13 hours and half using the traditional implementation to nearly 9 hours by using MapReduce. Therefore MapReduce decreases the processing time needed for processing large amount of seismic data.

Later, we have presented a seismic detection model, based on a Fully Convolutional Networks classifier which is trained on STanford EArthquake Dataset (STEAD) and tested on real continuous seismic data. By making a separate standardization for each sample of our dataset, instead of normalizing per feature, the performance

of our classifier is increased significantly by $\sim 16\%$ compared to raw data. Our experiments show that the use of small batch sizes is more adequate for our dataset, however, very small batch sizes (8 and lower) make the model over-fit and can not generalize for new data. By applying our classifier to real continuous data from Morocco's XB network, we were able to detect local earthquakes that already exist in our catalog of Morocco region, obtained from the National Geographic Institute (IGN) of Spain. Our method does not require hand-engineered features and is able to discriminate between earthquakes and seismic noise with high accuracy. Our results demonstrated that FCN classifier holds vast promise for making seismic detection more accurate.

The last contribution of this thesis is Seismic Time Series augmentation and classification. Data augmentation is done by Generative Adversarial Networks (GANs) to augment the training set, which included less positive classes than negative, then we concatenated the training set with the generated data from GANs to create our new training set which is used for classification.

In this thesis, we have seen how Fully Convolutional Networks are involved to perform earthquake detection. Some concepts discussed in this thesis can also be applied to time series problems in other fields including speech recognition, sentiment analysis, etc,. In our future work, we will attempt to take advantage of FCN, to perform seismic phase detection for P and S phases.

By developing an early warning systems, we aim to alert people and devices when seismic waves generated by an earthquake are likely to arrive at their location. Looking forward, our goal is to apply seismic detection on the entire XB network. This requires further optimizing since Short Term Average to Long Term Average lacks accuracy, and we should combine it with The FCN approach to improve its detection accuracy. Also, we aim to propose a platform to help seismologist to detect earthquakes on a given seismic time series.



Bibliography

- [1] R. Allen, "Automatic earthquake recognition and timing from single trace," *Bulletin of the Seismological Society of America*, vol. 68, pp. 1521–1532, 10 1978. Cited pages [18](#) and [84](#).
- [2] R. Allen, "Automatic phase pickers: Their present use and future prospects," *Bulletin of the Seismological Society of America*, vol. 72, pp. S225–S242, 12 1982. Cited page [18](#).
- [3] S. Magana-zook, J. Gaylord, D. Knapp, D. Dodge, and S. Ruppert, "Large-scale seismic waveform quality metric calculation using hadoop," *Computers & Geosciences*, vol. 94, 05 2016. Cited pages [18](#), [72](#), and [76](#).
- [4] Y. Choubik, A. Mahmoudi, M. Himmi, and L. El Moudnib, "Sta/lta trigger algorithm implementation on a seismological dataset using hadoop mapreduce," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, p. 269, 06 2020. Cited page [18](#).
- [5] A. Trnkoczy, "Understanding and parameter setting of sta/lta trigger algo-

- rithm,” *IASPEI New Manual of Seismological Observatory Practice*, vol. 2, pp. 1–19, 01 2002. Cited pages 18 and 73.
- [6] F. Magrini, D. Jozinović, F. Cammarano, A. Michelini, and L. Boschi, “Local earthquakes detection: A benchmark dataset of 3-component seismograms built on a global scale,” *Artificial Intelligence in Geosciences*, vol. 1, pp. 1–10, 2020. Cited page 18.
- [7] T. Perol, M. Gharbi, and M. Denolle, “Convolutional neural network for earthquake detection and location,” *Science Advances*, vol. 4, 02 2017. Cited page 19.
- [8] S. Mousavi, W. Ellsworth, Z. Weiqiang, L. Chuang, and G. Beroza, “Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking,” *Nature Communications*, vol. 11, p. 3952, 08 2020. Cited page 19.
- [9] O. M. Saad, G. Huang, Y. Chen, A. Savvaidis, S. Fomel, N. Pham, and Y. Chen, “Scalodeep: A highly generalized deep learning framework for real-time earthquake detection,” *Journal of Geophysical Research: Solid Earth*, vol. 126, no. 4, p. e2020JB021473, 2021. Cited page 19.
- [10] A. Shaheen, U. Bin Waheed, M. Fehler, L. Sokol, and S. Hanafy, “Deep learning for low-magnitude earthquake detection on a multi-level sensor network,” 03 2021. Cited page 19.
- [11] Z. Ross, M.-A. Meier, and E. Hauksson, “P wave arrival picking and first-motion polarity determination with deep learning,” *Journal of Geophysical Research: Solid Earth*, vol. 123, 04 2018. Cited page 19.

-
- [12] Z. Ross, M.-A. Meier, E. Hauksson, and T. Heaton, "Generalized seismic phase detection with deep learning," *Bulletin of the Seismological Society of America*, vol. 108, 08 2018. Cited page [19](#).
- [13] Y. Choubik, A. Mahmoudi, and M. Majid, "Fully convolutional networks for local earthquake detection," *International Journal of Advanced Computer Science and Applications*, vol. 12, 01 2021. Cited page [19](#).
- [14] Y. Choubik, A. Mahmoudi, and M. Majid, "Submitted to ceser international journal of artificial intelligence.," Cited page [19](#).
- [15] K. Paneri, V. Tv, P. Malhotra, L. Vig, and G. Shroff, "Regularizing fully convolutional networks for time series classification by decorrelating filters," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 10003–10004, 07 2019. Cited pages [19](#), [85](#), and [98](#).
- [16] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *IEEE Access*, vol. PP, 09 2017. Cited pages [20](#), [95](#), and [104](#).
- [17] Y. Huang, F. Zhou, and J. Gilles, "Empirical curvelet based fully convolutional network for supervised texture image segmentation," *Neurocomputing*, vol. 349, 04 2019. Cited pages [20](#), [85](#), [96](#), and [104](#).
- [18] L. Yann and Y. Bengio, "Convolutional networks for images, speech, and time-series," 11 1997. Cited page [33](#).
- [19] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," vol. 37, pp. 29–43, 12 2003. Cited page [41](#).
- [20] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," vol. 51, pp. 137–150, 01 2004. Cited page [42](#).

-
- [21] B. Sharma, A. Kumar, and V. Murthy, "Evaluation of seismic events detection algorithms," *Journal of the Geological Society of India*, vol. 75, pp. 533–538, 03 2010. Cited page [55](#).
- [22] D. Houlston, G. Waugh, and J. Laughlin, "Automatic real-time event detection for seismic networks," *Computers & Geosciences*, vol. 10, pp. 431–436, 12 1984. Cited page [55](#).
- [23] S. Marasco and G. Cimellaro, *Introduction to Dynamics of Structures and Earthquake Engineering*. 03 2018. Cited page [57](#).
- [24] D. B. Slemmons, "State-of-the-art for assessing earthquake hazards in the united states. report 6. faults and earthquake magnitude," 1977. Cited page [60](#).
- [25] R. Musson, G. Grünthal, and M. Stucchi, "The comparison of macroseismic intensity scales," *Journal of Seismology*, vol. 14, pp. 413–428, 2010. Cited page [68](#).
- [26] T. Addair, D. Dodge, W. Walter, and S. Ruppert, "Large-scale seismic signal analysis with hadoop," *Computers & Geosciences*, vol. 66, 05 2014. Cited page [72](#).
- [27] L. Huang, X. Dong, and T. Clee, "A scalable deep learning platform for identifying geologic features from seismic attributes," *The Leading Edge*, vol. 36, pp. 249–256, 03 2017. Cited page [72](#).
- [28] Y. Yan, L. Huang, and L. Yi, "Is apache spark scalable to seismic data analytics and computations?," pp. 2036–2045, 10 2015. Cited page [72](#).
- [29] Z. Chen and T. Zhang, "Processing and analysis of seismic data in hadoop platform," pp. 1–5, 02 2017. Cited page [72](#).

-
- [30] G. Cortés, A. Morales-Esteban, X. Shang, and F. Martínez-Álvarez, “Earthquake prediction in california using regression algorithms and cloud-based big data infrastructure,” *Computers & Geosciences*, vol. In press, 11 2017. Cited page 73.
- [31] L. Alan and H. Gene, “Program to investigate convective alboran sea system overturn,” 2009. Cited pages 74 and 86.
- [32] IRIS, “Seed reference manual: Standard for the exchange of earthquake data,” 10 2012. Cited page 75.
- [33] 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*, vol. 88, pp. 95–106, 02 1998. Cited pages 84 and 95.
- [34] W. Freiburger, “An approximate method in signal detection,” *Quarterly of Applied Mathematics*, vol. 20, 01 1963. Cited page 84.
- [35] S. Stewart, “Real time detection and location of local seismic events in central california,” *Bulletin of the Seismological Society of America*, vol. 67, pp. 433–452, 04 1977. Cited page 84.
- [36] Shensa, “The deflection detector: its theory and evaluation on short-period seismic data. technical report,” 01 1977. Cited page 84.
- [37] T. Goforth, “An automatic signal detection algorithm based on the walsh transform,” *Bulletin of the Seismological Society of America*, vol. 71, p. 1351, 01 1981. Cited pages 85 and 95.
- [38] Y. Vaezi and M. Baan, “Comparison of the sta/lta and power spectral density methods for microseismic event detection,” *Geophysical Journal International*, vol. 203, pp. 1896–1908, 12 2015. Cited page 85.

-
- [39] L. Fortuna, S. Graziani, M. Presti, and G. Nunnari, "A neural network for seismic events classification," in *[Proceedings] IGARSS'91 Remote Sensing: Global Monitoring for Earth Management*, pp. 1663 – 1666, 07 1991. Cited page [85](#).
- [40] F. Giudicepietro, A. Esposito, and P. Ricciolino, "Fast discrimination of local earthquakes using a neural approach," *Seismological Research Letters*, vol. 88, pp. 1089–1096, 07 2017. Cited page [85](#).
- [41] J. Akram, O. Ovcharenko, and D. Peter, "A robust neural network-based approach for microseismic event detection," in *SEG Technical Program Expanded Abstracts 2017*, pp. 2929–2933, 08 2017. Cited page [85](#).
- [42] T. Perol, M. Gharbi, and M. Denolle, "Convolutional neural network for earthquake detection and location," *Science Advances*, vol. 4, 02 2017. Cited page [85](#).
- [43] B. Zhao, S. Xiao, H. Lu, and J. Liu, "Waveforms classification based on convolutional neural networks," in *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pp. 162–165, 03 2017. Cited page [85](#).
- [44] S. Yuan, J. Liu, S. Wang, T. Wang, and P. Shi, "Seismic waveform classification and first-break picking using convolution neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. PP, pp. 1–5, 01 2018. Cited page [85](#).
- [45] A. Wilkins, A. Strange, Y. Duan, and X. Luo, "Identifying microseismic events in a mining scenario using a convolutional neural network," *Computers & Geosciences*, vol. 137, p. 104418, 04 2020. Cited page [85](#).
- [46] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural Computation*, vol. 31, pp. 1–36, 05 2019. Cited page [85](#).

- [47] J. Zheng, J. Lu, S. Peng, and T. Jiang, "An automatic microseismic or acoustic emission arrival identification scheme with deep recurrent neural networks," *Geophysical Journal International*, vol. 212, pp. 1389–1397, 02 2018. Cited page 85.
- [48] M.-A. Meier, Z. Ross, A. Ramachandran, A. Balakrishna, S. Nair, P. Kundzicz, Z. Li, J. Andrews, E. Hauksson, and Y. Yue, "Reliable real-time seismic signal/noise discrimination with machine learning," *Journal of Geophysical Research: Solid Earth*, 12 2018. Cited page 85.
- [49] J. Doubravová, J. Wiszniowski, and J. Horalek, "Single layer recurrent neural network for detection of swarm-like earthquakes in w-bohemia/vogtland - the method," *Computers & Geosciences*, vol. 93, 05 2016. Cited page 85.
- [50] Y. Chen, G. Zhang, M. Bai, S. Zu, Z. Guan, and M. Zhang, "Automatic waveform classification and arrival picking based on convolutional neural network," *Earth and Space Science*, vol. 6, 04 2019. Cited pages 85 and 96.
- [51] C. Song, Z. Liu, Y. Wang, F. Xu, X. Li, and G. Hu, "Adaptive phase k-means algorithm for waveform classification," *Exploration Geophysics*, vol. 49, 01 2017. Cited pages 85 and 96.
- [52] S. Mousavi, Z. Weiqiang, W. Ellsworth, and G. Beroza, "Unsupervised clustering of seismic signals using deep convolutional autoencoders," *IEEE Geoscience and Remote Sensing Letters*, vol. PP, pp. 1–5, 05 2019. Cited page 85.
- [53] A. Köhler, M. Ohrnberger, and F. Scherbaum, "Unsupervised pattern recognition in continuous seismic wavefield records using self-organizing maps," *GEO-PHYSICAL JOURNAL INTERNATIONAL*, vol. 182, pp. 1619–1630, 09 2010. Cited pages 85 and 95.

- [54] A. Köhler, M. Ohrnberger, C. Riggelsen, and F. Scherbaum, "Unsupervised feature selection for pattern search in seismic time series," in *Proceedings of the 2008 International Conference on New Challenges for Feature Selection in Data Mining and Knowledge Discovery - Volume 4*, FSDM'08, p. 106–120, JMLR.org, 2008. Cited pages 85 and 96.
- [55] C. Song, Z. Liu, Y. Wang, X. Li, and G. Hu, "Multi-waveform classification for seismic facies analysis," *Computers & Geosciences*, vol. 101, pp. 1–9, 04 2017. Cited page 85.
- [56] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1–1, 05 2016. Cited pages 85 and 96.
- [57] L. Bi, J. Kim, E. Ahn, A. Kumar, D. D. F. Feng, and M. Fulham, "Step-wise integration of deep class-specific learning for dermoscopic image segmentation," *Pattern Recognition*, vol. 85, pp. 78–89, 01 2019. Cited pages 85 and 96.
- [58] J. Fan, X. Cao, and P.-T. Yap, "Birnet: Brain image registration using dual-supervised fully convolutional networks," *Medical Image Analysis*, vol. 54, 02 2018. Cited pages 85 and 97.
- [59] C. Li, X. Wang, W. Liu, L. J. Latecki, B. Wang, and J. Huang, "Weakly supervised mitosis detection in breast histopathology images using concentric loss," *Medical Image Analysis*, vol. 53, 02 2019. Cited pages 85 and 97.
- [60] F. Such, S. Pillai, F. Brockler, V. Singh, P. Hutkowski, and R. Ptucha, "Intelligent character recognition using fully convolutional neural networks," *Pattern Recognition*, vol. 88, 12 2018. Cited pages 85 and 96.
- [61] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International Joint Confer-*

- ence on Neural Networks (IJCNN), pp. 1578–1585, 05 2017. Cited pages 85 and 96.
- [62] F. Karim, S. Majumdar, H. Darabi, and S. Chen, “Lstm fully convolutional networks for time series classification,” *IEEE Access*, vol. PP, 09 2017. Cited pages 85 and 96.
- [63] X. Zhang, J. Zhang, C. Yuan, S. Liu, Z. Chen, and W. Li, “Locating induced earthquakes with a network of seismic stations in oklahoma via a deep learning method,” *Scientific Reports*, vol. 10, 12 2020. Cited page 85.
- [64] Y. Shi, L. Huang, X. Dong, T. Liu, and J. Ning, “Application of Fully Convolutional Neural Network on Fault Detection,” in *AGU Fall Meeting Abstracts*, vol. 2018, pp. S13B–04, Dec. 2018. Cited pages 85 and 98.
- [65] S. Mousavi, Y. Sheng, Z. Weiqiang, and G. Beroza, “Stanford earthquake dataset (stead): A global data set of seismic signals for ai,” *IEEE Access*, vol. PP, pp. 1–1, 10 2019. Cited page 86.
- [66] M. Üstüner, F. Balik Sanli, and S. Abdikan, “Balanced vs imbalanced training data: Classifying rapideye data with support vector machines,” vol. XLI-B7, 07 2016. Cited page 89.
- [67] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS) 2011*, vol. 15, pp. 315–323, 01 2011. Cited page 90.
- [68] N. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” 09 2016. Cited page 92.

-
- [69] A. Maletzke, C. Ferrero, C. Tibes, E. Cherman, and W. Zalewski, "Medical time series classification using global and local feature extraction strategies," *Journal of Health Informatics*, vol. 9, p. 73, 08 2017. Cited page [95](#).
- [70] R. Blandford, "An automatic event detector at the tonto forest seismic observatory," *Geophysics*, vol. 39, 10 1974. Cited page [95](#).
- [71] Z. Wang, W. Song, L. Liu, F. Zhang, J. Xue, Y. Ye, M. Fan, and M. Xu, "Representation learning with deconvolution for multivariate time series classification and visualization," 10 2016. Cited page [96](#).
- [72] F. J. Moreno-Barea, J. M. Jerez, and L. Franco, "Improving classification accuracy using data augmentation on small data sets," *Expert Systems with Applications*, p. 113696, 07 2020. Cited page [97](#).
- [73] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Data augmentation using synthetic data for time series classification with deep residual networks," 08 2018. Cited page [97](#).
- [74] G. Forestier, F. Petitjean, A. Dau, G. Webb, and E. Keogh, "Generating synthetic time series to augment sparse datasets," pp. 865–870, 11 2017. Cited page [97](#).
- [75] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data Augmentation for Time Series Classification using Convolutional Neural Networks," in *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, (Riva Del Garda, Italy), Sept. 2016. Cited page [97](#).
- [76] P. Cao, X. Li, K. Mao, F. Lu, G. Ning, L. Fang, and Q. Pan, "A novel data augmentation method to enhance deep neural networks for detection of atrial fibrillation," *Biomedical Signal Processing and Control*, vol. 56, p. 101675, 02 2020. Cited page [97](#).

-
- [77] T. Um, F. Pfister, D. C. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulic, “Data augmentation of wearable sensor data for parkinson’s disease monitoring using convolutional neural networks,” pp. 216–220, 11 2017. Cited page 97.
- [78] J. Kortström, M. Uski, and T. Tiira, “Automatic classification of seismic events within a regional seismograph network,” *Computers & Geosciences*, vol. 87, 11 2015. Cited page 97.

Résumé:

STA/LTA est l'algorithme le plus couramment utilisé pour la détection des tremblements de terre. Étant donné que les algorithmes de détection automatique impliquent fréquemment des erreurs cruciales. Dans cette thèse, nous avons implémenté l'algorithme de déclenchement STA/LTA en utilisant Hadoop MapReduce. Cette implémentation nous a permis de connaître son efficacité dans ce type de tâches ainsi que d'accélérer le processus de détection en réduisant le temps de traitement.

D'autre part, la détection automatique des séismes est largement étudiée pour remplacer la détection manuelle. Par conséquent, le besoin d'apprentissage profond est devenu de plus en plus important pour le traitement des séries chronologiques. Nous avons proposé un modèle de détection automatique des tremblements de terre basé sur le classifieur FCN. Pour tester l'efficacité de notre classificateur, nous l'avons appliqué à des données sismiques continues réelles du réseau XB du Maroc et comparé les résultats à notre catalogue.

Mots clé: Apprentissage Profond, Donnée Massive, Séries Chronologiques, Détection de Séisme

Abstract:

STA / LTA is the most commonly used algorithm for earthquake detection, and it involves signal processing using filters in different time frames to identify amplitude outliers and hence seismic phases. Because automated identifications frequently involve crucial errors, this method is generally corrected further by experts in seismic signal analysis. In this thesis, we implemented STA/LTA trigger algorithm using Hadoop MapReduce. This implementation allowed us to find out how effective it is in this type of tasks as well as to accelerate the detection process by reducing the processing time. We tested our implementation on a seismological dataset of broadband seismic stations implemented in Morocco and compare it with the traditional one.

On the other hand, automatic earthquake detection is widely studied to replace manual detection, however, most of the existing methods are sensitive to seismic noise. Hence, the need for Deep Learning has become more and more significant to process seismic time series. We proposed an automatic earthquake detection model based on FCN classifier. To test the efficiency of our classifier, we applied it to real continuous seismic data from XB Network from Morocco and compared the results to our catalog.

Keywords: Deep Learning, Big Data, Time Series, Earthquake Detection.

Année Universitaire: 2020/2021