UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Físicas y Nanotecnología

TÍTULO: Application of Neural Networks to Chaotic Systems in Economics.

Trabajo de integración curricular presentado como requisito para la obtención del título de Físico

Autor:

Erick Santiago Guagua Torres

Tutor:

Ph.D. Duncan J. Mowbray

Urcuquí, Febrero 2023

AUTORÍA

Yo, **Erick Santiago Guagua Torres**, con cédula de identidad 0952352474, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autora (a) del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Febrero 2023.

Erick Santiago Guagua Torres

CI: 0952352474

AUTORIZACIÓN DE PUBLICACIÓN

Yo, **Erick Santiago Guagua Torres**, con cédula de identidad 0952352474, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad. Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior

Urcuquí, Febrero 2023.

Firmado electrónicamente por:
ERICK SANTIAGO
GUAGUA TORRES

Erick Santiago Guagua Torres

CI: 0952352474

Dedicatoria

Papá,

Esta tesis va dedicada a ti, en memoria de tu amor incondicional y tu guía constante en mi vida. Aunque ya no estás físicamente aquí conmigo, tu espíritu y tus enseñanzas siempre estarán presentes en mi corazón. Gracias por enseñarme a ser fuerte y a no rendirme ante los desafíos. Aunque no estés aquí para ver este logro, sé que estás orgulloso de mí.

Descansa en paz, papá. Te quiero y te extraño siempre.

Mamá,

Esta tesis es para ti, en reconocimiento a todo lo que has hecho por mí. Eres mi ángel y mi confidente. Gracias por siempre apoyarme y creer en mí, incluso cuando yo mismo dudaba de mis capacidades. Gracias por enseñarme el valor de la dedicación, la determinación y el amor incondicional. Eres mi roca, mi fuente de inspiración y mi guía en la vida. Gracias por estar ahí para mí en cada momento, por consolarme cuando estoy triste y por celebrar conmigo mis logros. Eres la luz de mi vida y no puedo agradecerte lo suficiente por todo lo que haces por mí.

Te quiero mucho, mamá. Este logro es tuyo tanto como mío.

A mi familia,

Esta tesis es para todos ustedes, en reconocimiento a su amor y apoyo incondicional. Gracias por ser mi hogar, mi fortaleza y mi luz en los momentos más oscuros. Gracias por creer en mí y alentarme a perseguir mis sueños. Gracias por estar ahí para consolarme cuando estaba triste y celebrar conmigo mis logros. Cada uno de ustedes ha dejado una huella indeleble en mi vida y estoy eternamente agradecido por su amor y apoyo. Gracias por ser mi familia. Los quiero mucho.

A mis amigos,

Esta tesis es para ustedes, en reconocimiento a su amistad y apoyo incondicional a lo largo de los años. Gracias por hacerme reír en los momentos más difíciles, por ser mi hombro en los momentos de tristeza y por celebrar conmigo mis logros. Gracias por escuchar mis ideas, alentarme a perseguir mis sueños y estar ahí para mí en los buenos y malos momentos. Cada uno de ustedes ha dejado una huella indeleble en mi vida y estoy eternamente agradecido por su amistad y apoyo. Gracias por ser parte de mi vida y por ser mis amigos.

Santiago

Agradecimiento

Agradezco a mi madre, padre y familia que siempre me han apoyado, a mis amigos que estuvieron en el camino para este logro, y a todas esas personas que me ayudaron a llegar hasta aquí, aunque algunos ya no están este trabajo es una dedicatoria hacia ellos.

I want to acknowledge my thesis advisor Duncan J. Mowbray for his mentoring, patience and help through the entire project.

This work employed the Imbabura cluster of Yachay Tech University, which was purchased under contract No. 2017-024 (SIE-UITEY-007-2017).

Resumen

El entendimiento de las fuerzas del mercado que impulsan la economía, como se ejemplifica en índices como el S&P 500, es de vital importancia para el modelado, predicción y dirección tanto de economías desarrolladas como emergentes. El conocimiento o la visión de las futuras actuaciones del mercado podrían reducir el riesgo y aumentar las ganancias para los inversores. Sin embargo, los anteriores intentos de encontrar un modelo predictivo para describir el desempeño del mercado de valores han encontrado éxitos limitados debido a la naturaleza estocástica, no linealidad y alto grado de complejidad de tales sistemas. Como primer paso hacia ese entendimiento del mercado, aquí analizamos la relación entre dos características fundamentales del mercado: el cambio diario en el precio y el volumen del S&P 500. Primero, usamos un análisis estadístico simple para demostrar el comportamiento no lineal del índice a corto plazo. Para cuantificar la dependencia del cambio diario en el precio de factores que no sean el cambio en el volumen, también hemos analizado la variación en el precio cuando el volumen es más o menos constante y encontramos que tales otros factores podrían describirse como ruido blanco débil. Finalmente, entrenamos un modelo de red neuronal de una sola capa con todas las neuronas densamente interconectadas y mostramos que, aunque puede reproducir los datos de entrenamiento, de acuerdo con el teorema de simulación universal, es incapaz de predecir los datos de prueba, incluso a un nivel semicalitativo. Estos resultados demuestran la necesidad de "deep learning" a través de redes neuronales de múltiples capas para describir el comportamiento de los índices del mercado de valores.

Palabras clave: inteligencia artificial, aprendizaje automático, índice S&P 500, modelado predictivo, relación precio y volumen.

Abstract

The understanding of the market forces that drive the economy, as exemplified in indices such as the S&P 500, is of vital importance for the modeling, prediction, and direction of both developed and emerging economies. Knowledge of or insight into the market's future performance could reduce risk and increase profits for investors. However, previous attempts to find a predictive model to describe stock-market performance have met with limited success, due to the stochastic nature, non-linearity, and high degree of complexity of such systems. As a first step towards such an understanding of the market, here we analyze the relation between two fundamentals market features: the daily change in price and volume of the S&P 500. First, we use a simple statistical analysis to demonstrate the non-linear behavior of the index on short timescales. To quantify the dependence of the daily change in price on factors other than the change in volume, we have also analyzed the variation in price when the volume is more or less constant and find such other factors could be described as weak white noise. Finally, we train a single-layer neural network model with all neurons densely interconnected and show that although it can reproduce the training data, in agreement with the universal simulation theorem, it is unable to predict the test data, even at a semi-qualitative level. These results demonstrate the need for "deep learning" via multi-layer neural networks to describe the behavior of stock-market indices.

Keywords: artificial intelligence, machine learning, S&P 500 index, predictive modeling, price and volume relationship.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, technological advance is necessary for all areas of our life. One of these primary areas of technological development and application is agriculture. Adapting new agriculture technologies has made this field into the so-called modern agriculture or agroindustry. One of the many implementations of new technology in this field is using neural networks to recognize a field from satellite or aerial photos and evaluate its humidity, maturity, growth status, etc. This new technology is of considerable benefit, as it allows the use of images for an algorithm to learn about the different states of a field or farm. When put into practice, such methods are very useful, and the need for personnel to survey large areas is reduced. In the case of Ecuador, with a highly agricultural economy, the application of such types of technology is scarce. Although much research about this topic exists, it is hard to find concrete examples. For example, in one study of spatial images of sugarcane crops in Ecuador, the authors concluded that using a convolutional neural network, a type of neural network for working with images, they obtained an effectiveness of more than 94% for recognizing the status of sugarcane crops[1].

Another application of neural networks affecting our day-to-day life, in the case of Ecuador, is in public transport. In one example, a neural network was used to count people using public transport[2]. The number of people is important in guaranteeing safe road trips, as transport vehicles must not exceed an established number of passengers. The authors found a 97% effectiveness

when counting people. The 3% error demonstrates the complexity of identifying people on a bus. Applications such as these exist in many different areas but generally have yet to be applied significantly in Ecuador. This lack of applications is partially due to the need for more professionals with experience in neural networks and artificial intelligence (AI).

So far, we know what a neural network can do, but we also need to know what exactly a neural network is. To answer this, a formal definition of a neural network is necessary. First, the overall idea of a neural network is to mimic the human brain. This idea is because, in theory, although the human brain performs many fewer computations per second than a state-of-the-art computer[3], the brain's ability is still currently beyond that of computers.

The idea of learning is fascinating and gives us an advantage over computers. For this reason, neuroscience is still actively studying this important and intriguing organ. However, it is known that the brain can take new information and process it through learning, performing tasks based on previous experience and knowledge obtained through this action. Thus, the purpose of neural networks is to mimic the advantages of the human brain within a computational framework. Additionally, throughout the literature, one can find different names for neural networks, including artificial neural networks (ANNs) and simulated neural networks (SNNs). These names are due to their similarity to the human brain, with ANNs comprising node layers containing an input layer, one or more hidden layers, and an output layer, as depicted in Figure 1.1[4]. It is important to note that neural networks are a subset of machine learning and are often the heart of deep learning algorithms.

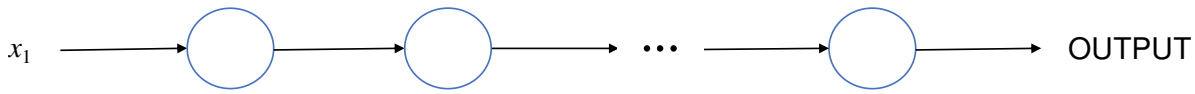

Figure 1.1: A simple neural network with one input, n neurons, and one output

As discussed above, how a neural network works is similar to the synapses of a brain. Each node or artificial neuron can transmit a signal to other neurons with which it is connected. Furthermore, there is an associated weight and threshold for transmitting this signal for artificial

networks. To understand this procedure best, when a neuron receives a signal, it processes it and transmits it to neurons connected to it. The signal in this context is a real number. Additionally, the output of each neuron is obtained through some nonlinear function (activation function) of the sum of its inputs. Here each connection between nodes has a specific name, called edges. Neurons and edges, in most cases, have a weight that adjusts as learning proceeds. The strength of the signal's connection depends on the weights, thus increasing or decreasing based on them. As mentioned before, an important property of neurons is that they have a threshold such that a signal is sent if it is above a specified threshold value. Under these conditions, it is said that a neuron is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network[4]. Generally, neurons are distributed in layers, and different layers can derive different input variations. In this way, the signal travels from the input layer (the first layer) to the output layer (the last layer).

We will now consider the historical background of how neural networks grew from simple circuits to the powerful tools that we know today, and have been very helpful to modern society.

In the 1940s, the field of neural networks began with the proposal by McCulloch and Pitts of their eponymous MP model. This includes their invention of a simple neural network model with electrical circuits in 1943[5]. In that year, they modeled a neuron as a switch that receives input from other neurons and, depending on the total weighted input, is either activated or remains inactive. This weight is the value for which an input from another cell is multiplied, and, by analogy with the human brain, it corresponds to the strength of a synapse. Furthermore, this signal can be either positive (excitatory) or negative (inhibitory).

In 1949, an important law was presented by the psychologist Hebb[6]. In this work he hypothesized that there exist variations in the intensity of synaptic connections. This hypothesis suggests that the learning process occurs at the synaptic interface between neurons. This means that before and after the synapse in neurons the intensity of the synaptic connection varies. This hypothesis was the basis of the well-known Hebb rule in neural networks. This law states that the strength of synaptic connections between neurons is variable, and that it is this variability that is the basis for learning and memory[7].

In 1957, Rosenblatt proposed the so-called "Perceptron" model[8] based on the MP model[9]. This model has its structure based on neurophysiology. For this reason, the principle of modern neural networks starts with the Perceptron model. This model is an MP neural network model with continuously adjustable weights. Once the network is trained, it can achieve the classification and recognition of certain inputs. It is the first genuine neural network, despite its simplicity. Rosenblatt demonstrated the ability of two-layer sensors to classify inputs and presented a promising research direction for three-layer sensors with hidden layer processing features. Rosenblatt's neural network model incorporates some of the most fundamental ideas of modern neural computing, resulting in a significant advancement in neural network methodologies and technology.

In 1959, Widrow, Hoff, *et al.* proposed a neural network training method based on adaptive linear elements (ADALINE). The Widrow–Hoff learning rules, also known as the least mean square deviation algorithm or *d* rule, were applied to a real-world project. This makes ADALINE the first artificial neural network to solve practical problems and promote neural network research[10]. The ADALINE network model for adaptive systems is a continuous-valued adaptive linear neuron network model. Two decade after the MP model was presented, their network had properties similar to the brain. This is clearly evident because it can perform sophisticated pattern recognition, and even function after some neurons have been damaged[11].

Interest in the field of neural networks grew after various demonstration that simple "perceptron" networks could learn from examples. As a precursor of artificial intelligence, Minsky and Papert conducted a mathematical analysis of the functions and limitations of perceptron-based network systems[12]. In 1969, Minsky and Papert argued that simple linear perception has limitations, and is unable to tackle the challenge of classifying two types of linear inseparable samples[13].

In 1982, Hopfield proposed a discrete neural network, known as the discrete Hopfield network, which effectively promoted neural network research[14]. For the first time, this type of network introduces the Lyapunov function. Later researchers have also referred to the Lyapunov function as the energy function. This network's stability is demonstrated by Hopfield's model not only performing nonlinear mathematical summarization on the artificial neural network's information storage and retrieval function, but also providing dynamic equations and learning equations.

Hopfield's model also provides important formulas and parameters for the network algorithm, stimulating the construction and learning of artificial neural network theory by a large and diverse group of scholars' enthusiasm for studying neural networks and actively participate in this academic field. Nowadays, people are paying ever more attention to neural network research because of the great potential of the Hopfield neural network in many areas. Ever increasing numbers of people are studying neural networks, which is greatly promoting neural network development.

Rumelhart *et al.* proposed the back propagation (BP) algorithm, or error back propagation, in 1986 This model is based on the multi-layer neural network model, and solves the problem of multi-layer neural network weight correction[15]. The forward neural network learning problem demonstrates that the multi-layer neural network has strong learning abilities, as it can complete many learning tasks and solve many practical problems.

Chua and Yang proposed a cellular neural network (CNN) model in 1988[16]. A CNN model is a large-scale nonlinear computer simulation of cellular automata. The main difference in CNNs is that their interconnections are localized to only be present between neighbouring units.

More general delayed cellular neural networks (DCNNs), Hopfield neural networks (HNNs), and bidirectional associative memory networks (BAMs) are obtained by broadening the activation function classes of neural networks. Hundreds of neural network models have been proposed after years of development and the development in this field continues. Neural networks currently show even more potential than ever before, even in the performance of human tasks, such as driving.

Knowing where neural networks come from and how they have been evolving until today, it is also interesting to know what the stock market is and why it is important for an economy and what will be the advantage of applying this type of network to this economic system.

The connection between economic development and economic growth is of great importance. Therefore the development of a stock market is vital in a developing economy since, in developed countries, there is already a well-structured stock market. It is one of many reasons their

economies remain relatively stable (small fluctuation) compared to other countries (developing countries). The development of stock markets improves economic activity by providing liquidity to companies[17]. Based on the market, they decide whether to invest in new projects that benefit the host country through the generation of economic activities. Considering that stock markets in developed countries have an important role in the economy, Bolaños[17] considers that under this statement, it might be possible that financial development encourages economic growth or vice versa. Although several studies show that a well-developed stock market can stimulate economic growth in the long term[18–20], there is not enough evidence demonstrating that the stock market is a key principal for economic growth.

Experimental investigations that seek a link between financial development with the economy have been made[21–23]. Nevertheless, these kinds of studies are very few for developing economies, and the results are mixed and do not show a clear trend. Among all these developing economies, Ecuador is also included because there are not many studies about the Ecuadorian stock market.

Consequently, the Ecuadorian stock market is considered one of the less developed markets in Latin America. To understand why it is necessary, we should first briefly explain the market. For an international market, the idea is often to unify every stock in one so called index. However, in a small market, there are no such indices. Furthermore, the need for development in operation and technological parts reduces the efficiency of an already inefficient stock market. To be successful, a market must follow international standards, which are given to be used by national and foreign investors to be confident and have confidence in the market. These parameters are useful since knowing the liquidity and how to trade in a specific market is necessary.

Due to the need to respond to the evolution and nature of economic and commercial processes in Ecuador that have been developing over time, establishing the Ecuadorian stock market was necessary. The main reason was to provide people with a suitable and modern mechanism to promote domestic savings, distribute wealth, and boost productive activities.

According to Arauz[24], in 1847, Guayaquil was registered as the first attempt to create a stock exchange. This attempt tried to follow the example of the London stock market, which was the most important at that time. However, unfortunately, the initiative lasted only a few months.

Then, in 1873, the mercantile exchange in Guayaquil was created to take advantage of the cacao boom. This market supported 20 companies that were traded. However, again, the initiative was stopped due to the country's instability due to the Liberal Revolution. A decade later, in 1884, the commercial exchange was created in Guayaquil. After that, the Ecuadorian stock market, in general, changed over time due to the different laws to regularize the economy and the entities representing the country's biggest capitals. Consequently, in 1930, Guayaquil was selected to see the Ecuadorian stock and commodities Exchange creation.

In 2009, debates and negotiations to create a new law started, and in 2014 these gave results in the regularization in the stock market to finally set that law in 2017. That law allows the creation of an appropriate environment along with other factors such as macroeconomic policy, public policy, public-private partnerships, and private development to arrange for an effective market to emerge.

A stock market, equity market, or share market refers to the aggregation of buyers and sellers of stocks (also called shares), which represent fractional ownership claims on businesses. These may include securities listed on a public stock exchange or stock only traded privately, such as shares of private companies sold to investors through equity crowdfunding platforms. An efficiently functioning stock market is considered critical to economic development, as it allows companies to access capital from the public quickly.

However, how does it work? The stock market is one of the most important ways for companies to raise capital since it allows businesses to be publicly traded and raise additional financial capital for expansion by selling shares of ownership of the company in a public market. The liquidity an exchange affords the investors enables their holders to quickly and easily sell securities. This advantage is an attractive feature of investing in stocks, compared to other less liquid investments such as property and other immovable assets.

As a result, the stock market works for two purposes. The first is to provide capital to companies to fund and expand their businesses. For instance, if a company has a million shares of stock that are initially sold for $10 a share, thus the company now has $10 million of capital that the company can use to grow its business. In this way, by offering stock shares instead

of borrowing the capital needed for expansion, the company avoids incurring debt and paying interest charges on that debt. The second purpose the stock market serves is to allow buyers to share in the profits of publicly-traded companies. Buyers or investors can profit from stock buying in one of two ways. Some stocks pay regular dividends, i.e., a given amount of money per share of stock paid to the investor. The other way investors can profit from buying stocks is by selling their stock for a profit when the stock price increases above the purchase price. For example, if an investor buys shares of a company's stock at $10 a share and the price subsequently rises to $15 a share, the investor can then realize a 50% profit on their investment by selling their shares.

1.1 Problem Statement

The stock market represents a way companies may obtain capital through selling share, providing the company an opportunity to increase its funds and develop its business. The idea of this thesis is to predict the difference daily change in close price from the difference in volume to test whether we can predict when to buy and sell, in other words, the behavior of the market. This idea could have a very important impact by reducing the risk to an investor and increasing the profits of a company or vice versa. Furthermore, studies exist to try to predict the behavior of the stock market[25–28], which also is very important and leads to the same benefits as the motivation of this thesis. In general, studying the stock market is important because it directly rewards companies and investors and indirectly affects the country's economy. Moreover, the stock market is one reason that developed countries can maintain stable economies in contrast with developing countries. This thesis will help with a better understanding of the stock market's behavior and can be implemented to study the Ecuadorian stock market. Although several prediction methods exist, a neural network was selected in this case because it tries to predict prices that do not follow a linear pattern. A neural network has been demonstrated to be a powerful tool for studying those pattern[29–31].

1.2 General and Specific Objectives

This thesis aims to train a neural network to predict the daily change in closing price based on the daily change in volume, using the S&P 500 index data as a test case, which has shown a dynamical system behavior. The specific objective is to create a predictive model to find the daily change in closing price from the daily change in volume and apply it to the Ecuadorian stock market. This application aims to generate a more stable economy since a stable stock market means stable economic growth and, consequently, an improvement in the socioeconomic standards of the Ecuadorian population.

1.3 Overview

This thesis has five chapters. In Chapter 1 we have provided an introduction to the concepts of neural networks, the stock market, the importance of the stock market in an economy, and why we want to apply a neural network to try to make a predictive model. Chapter 2 provides the necessary theoretical background for this thesis, including a complete description of neural networks, their origin, definition, history, types, and mathematical definition, as well as concepts behind deterministic chaotic systems, white noise, and the Standard and Poor's 500 (S&P 500) stock index. Chapter 3 describes the methodology we employ, introduces the data we use, its structure, and the computational details, which consist of two analyses, the simple analysis, and the analysis through a neural network. Chapter 4 describes our results and provides a discussion thereof, and is separated into two well-defined sections. Section 3.2.1 covers the first data analysis, which consists of using specific tools to analyze the data and get information about its behavior, whereas Section 4.2 covers the second analysis of the data, i.e., the application of a neural network to the data of interest. Finally, in Chapter 5, we describe our conclusions, summarizing the principal results, and provide an outlook for future research.

Chapter 2

Theoretical Background

This chapter will explain in great technical detail the theory needed to understand this thesis.

2.1 Neural Networks

Neural networks come from a huge field of study, machine learning, which is a sub-field of artificial intelligence, and depending on the neural network's structure, this can be inside the smaller field of deep learning. for this reason, it is necessary to define clearly what these fields are, how they are related, and why neural networks belong to them. Figure 2.1 gives us an idea of the relationship between these three fields.

2.1.1 Artificial Intelligence

The field of artificial intelligence (AI) began in the 1950s. At that time, several computer scientists asked whether computers could "think". This question generated many more questions

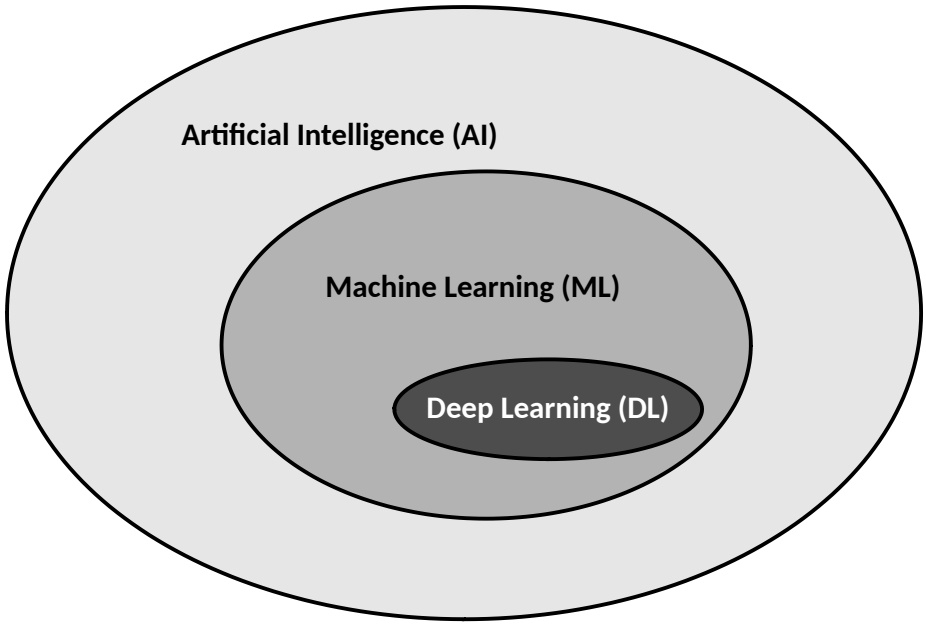

Figure 2.1: Relationship between Artificial Intelligence (AI) (light grey region), Machine Learning (ML) (grey region), and Deep Learning (DL) (dark grey region) disciplines.

that nowadays are still being explored. However, many of these questions have been answered. Generally, AI is described as the effort to automate intellectual tasks normally performed by humans[32]. Consequently, AI is a field that encompasses both machine learning and deep learning, and other approaches that do not need to involve learning. Curiously, most AI references until the 1980s did not specifically mention learning in this context. This was because experts believed that having a sufficiently large set of explicit rules for manipulating knowledge stored in explicit databases by programmers could achieve human-level artificial intelligence. This approach is known as symbolic AI[33].

Although this approach can successfully solve logical and well-defined problems, such as chess, its performance is poorer when solving more complex and fuzzy problems, such as natural language translation. To deal with this complexity, a new approach appeared: machine learning.

2.1.2 Machine Learning

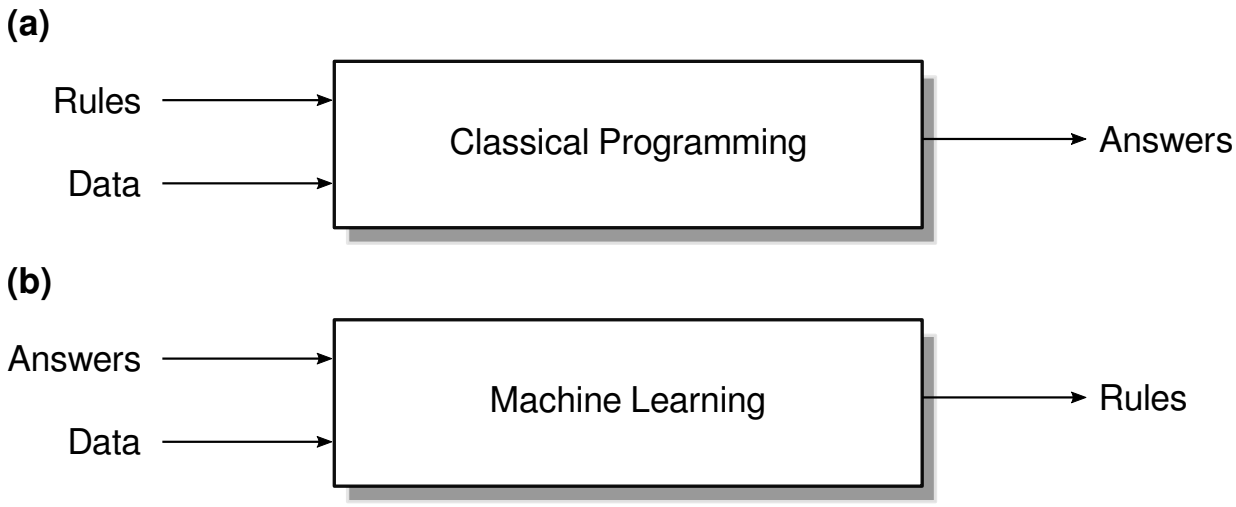

Figure 2.2: Comparison of (a) Classical Programming and (b) Machine Learning paradigms.

Generally, a computer does useful work when a human programmer writes the rules to accomplish the work. In other words, a computer program transforms input data into appropriate answers. However, machine learning changes the steps. Instead of classical programming, the machine gets the input data and the answers and determines the rules that it needs to follow. These two paradigms are shown schematically in Figure 2.2.

Although machine learning systems must be programmed, the heart of these systems is training, for which the computer program is made. This works by feeding the system with the input data and answers, and generating the rules as output. As a result, the rules to followed are not explicitly programmed step by step. The way to make this possible is once the input data is fed, the system seeks a statistical structure in the data that eventually allows the system to find the rules for automating the task.

For example, if you wish to be alerted when your crop is ready to harvest, you feed you machine learning system with many pictures already tagged by humans showing when a crop is or is not ready. The system will then learn/find statistical rules or patterns to associate specific

features in the pictures to the state of the crop.

Although machine learning is associated with mathematical statistics, it differs in many important ways from statistics. As an analogy, the differences between them are similar to how medicine is related to chemistry but cannot be reduced to chemistry since medicine deals with its own rules. So, in contrast to statistics, machine learning usually works with large, complex datasets. Such datasets may consist of millions of images, and depending on their quality, each consisting of millions of pixels. In such cases a classical statistical analysis becomes impractical. Consequently, machine learning, particularly deep learning, consists of comparitavely fewer mathematical theories and may be considered to be closer to an engineering discipline. This is because machine learning is a very practical field based on many empirical findings and is strongly dependent on advanced software and hardware.

Before explaining deep learning and understanding the difference between deep learning and machine learning approaches, it is first necessary to know what exactly machine learning algorithms do. We can see in Figure 2.2 that machine learning systems need three things:[34]

- Input data points or data
- Sample expected output or answers
- Performance assessment measure or machine learning mechanism

The machine learning mechanism used depends on the type of task the algorithm is performing. However, the overall goal is the same: to determine the differences between the actual and predicted values.

A machine learning model ”learns” how to transform input data into its expected output by being exposed to known examples of inputs and outputs. Hence, the main challenge in machine learning and deep learning is effectively changing data from inputs to outputs. Or, to put it another way, the goal is the development of representations of the input data at hand, representations that bring us closer to the intended output. In this way, by using direction from a feedback signal,

machine learning essentially consists of looking for meaningful representations and rules over some input data within a preset space of possibilities. This straightforward concept enables the resolution of an astonishingly wide range of intellectual challenges. These range from speech recognition to autonomous driving. Now that we know what machine learning entails let us look at what makes deep learning unique.

2.1.3 Deep Learning

The word "deep" in "deep learning" implies a representation of successivly deeper layers rather than some deep understanding. Consequently, the number of layers a model has is referred to as the depth of the model[35]. Furthermore, other terms exist in the literature for deep learning, such as "layered representations learning"[36] or "hierarchical representation learning"[37]. On the one hand, successive layers commonly called deep learning include from tens to hundreds of layers. On the other hand, machine learning methods using just one or two layers to represent the data are often called shallow learning.

Neural networks are the model used in deep learning. To learn, these models are structured so that the layers go on top of each other. Here, "neural networks" refers to the field of medicine and neurobiology, as the human brain inspires these models. However, they are not complete models for a human brain.

Technically, deep learning is a multi-step method for learning data representations. It can note how a network with several layers, e.g., four, can transform an image of a handwritten digit to recognize what digit it is. It is a simple idea, but at the end of the day, even very simple mechanisms that scale well can look like magic. To provide a better understanding of this process, in Figure 2.3 we provide a schematic representation of such a deep learning algorithm.

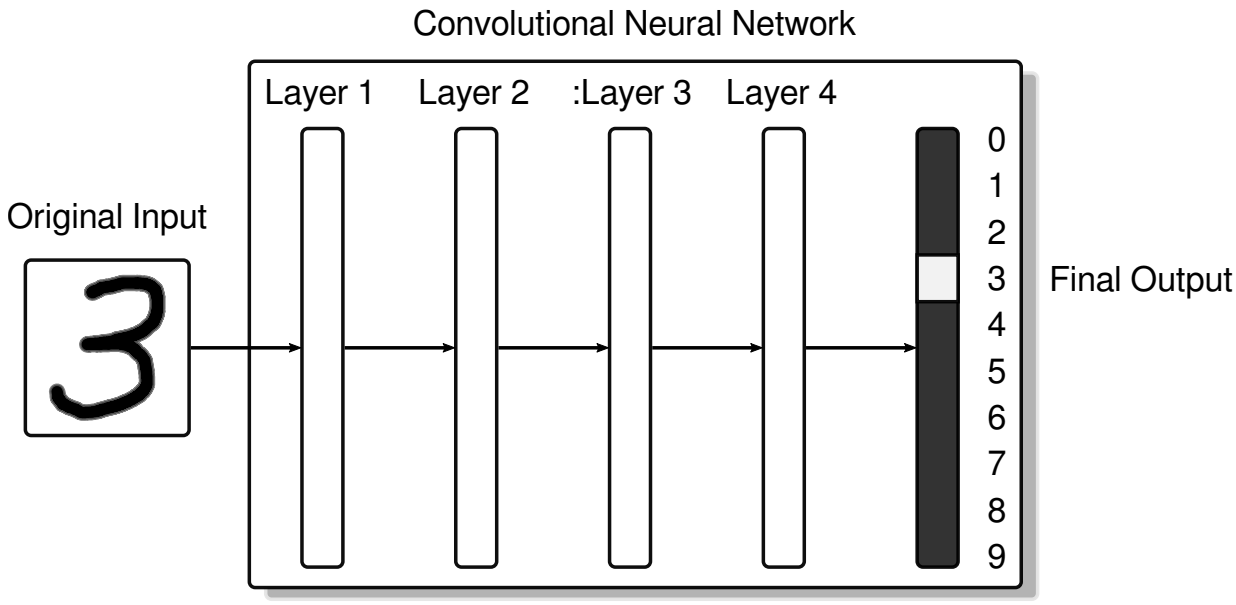

Figure 2.3: Schematic representation of a convolutional neural network deep learning algorithm with four layers for interpreting a handwritten digit.

2.1.4 Definition

Having explained from which fields neural networks arose, it is now necessary to define them so as to explain what neural networks actually are. Neural networks are a way of abstracting the human brain's neural network that uses information processing theory its theoretical base[38]. It is then implemented as a computational model. The model, among other things, consists of many interconnected nodes or neurons[39]. Each node consists of a specific output function. These are the so-called activation functions. Moreover, the nodes' connections contain a weight which is applied to any signal passing through the connection. This is equivalennt to the "memory" of the neural network[40]. The output of the network varies depending on how the network is connected, the weight values, and the activation functions[7]. Nonetheless, the network itself can generally represent an approximation of a function, an algorithm, or a logical expression[41].

In artificial neural network a neuron is a processing unit capable of representing different

objects, such as features, letters, concepts, or abstract patterns. The types of processing units in the network are classified into three categories: the input, output, and hidden units[7], as depicted schematically in Figure 1.1. In the input unit, also known as the input layer, the neural network is fed with the data set of interest[42]. The output unit, also known as the output layer, generates the output of the system processing result. The hidden unit, also called the hidden layer, is located between the input and output units and cannot be observed outside the system[43]. The strength of the connections between cells is given by the weight of the connection between neurons. The relationship between neurons represents how strong the processing of information between these is.

Neural networks are used in several fields, such as neuroscience, artificial intelligence, computer science, and other disciplines. According to Wu[44] in the last decade great progress has been made in research on artificial neural networks. Achievements include pattern recognition, intelligent robots, automatic control, prediction and estimation. Moreover, artificial neural networks have solved practical problems that modern classical programming techniques cannot solve.

As explained before, a neural network works similarly to the synapses of a brain, with each node or artificial neuron transmitting a signal to other neurons via connections with an associated weight or strengthand threshold. To understand this procedure in a better way, when a neuron receives a signal it then processes the signal and transmits this signal to neurons connected to it. The signal in this context is a real number, additionally to the output of each neuron it obtains through some linear or non-linear function, i.e., an activation function, of the sum of its inputs. Here each connection between nodes or neurons has a specific name and is so-called edges. Neurons and edges in most cases have a weight that is adjusted as learning proceeds. The strength of the signal transmitted via a connection depends on the weight. For this reason, the strength of a connection between nodes or neurons increases or decreases according to its weight. Also it is necessary to remember that an important fact of neurons is that they have a threshold such that a signal is sent only if it has received a linear combination of signals above a specified threshold value. It is then said that a neuron is activated, and it proceeds to send data to the next layer of the network. Otherwise, the neuron is deactivated, and no data is passed along to the next layer of the network[4]. Generally, neurons are distributed in layers. These different layers can deferentiate between different values on their inputs. Moreover, signals travel from the input layer, i.e., the

first layer, through the network of layers to the output layer, i.e., the last layer.

After many years of historical development we can conclude that neural networks currently have the following four intrinsic characteristic.

1. Non-linear: This is a universal feature of nature, and, as such, the intelligence of our brain is a non-linear phenomenon.
2. Unlimited: Multiple interconnected neurons constitute a neural network, and its behavior depends on the interconnections and interactions between the units rather than individual neurons.
3. Non-qualitative: Due to its self-adaptive, self-organizing, and self-learning ability, a neural network not only processes information that may present a high variability, but also deals with its internal nonlinear dynamical behavior, generating a constant change in the network.
4. Non-convexity: Under certain conditions, the evolutionary direction of a system depends on a particular state function, so that the function associated with a neural network may have multiple equilibrium states, generating diversity of system evolution.

2.1.5 Classification

There is no official list of types of neural networks. Generally in the literature networks are classified based on their purpose or uses. Following this logic, neural networks may be classified into the following most common types:[11]

- Perceptron Network: The simplest and oldest form of a neural networkconsisting of a single neuron.[8]
- Feedforward Sigmoid Network or Multi-Layer Perceptrons (MLPs): Consists of an input layer, a hidden layer or layers, and an output layer.

- Convolutional Neural Networks (CNNs): Similar to feedforward networks but focused on image recognition, pattern recognition, and/or computer vision. These networks harness principles from linear algebra, particularly matrix multiplication, to identify patterns within an image.
- Recurrent Neural Networks (RNNs): Identified by their feedback loops, these learning algorithms are primarily leveraged for time-series data to make predictions about future outcomes, such as stock market predictions or sales forecasting.

Neural networks can also be classified by their topology in accordance of what we have mentioned before[45].

2.1.6 Mathematical Definition

The neural networks employed in this thesis belong to the subcategory of supervised learning. In this case, a set of N pieces of training data $T = \{x_n, t_n\| : 1 \leq n \leq N$, along with a target function g with target values $t_n = g(x_n)$, are the inputs to the problem[3]. Our aim is to determine, for all input values in the domain of g, this target function through approximation, where this function is typically unknown. Using the mapping obtained from a set of training data, many classification and regression problems may be formulated and solved using artificial neural networks.

These neural networks are made up of a collection of linked processing units, each of which receives input from an external source or from other units, and computes an output that may subsequently be passed to other processing units. In this was the network recreates the connections in the human brain[46]. Processing units are made up of propagation rules that map all incoming input onto a single input value. Activation functions are applied to these input values to determine the unit's output or activation. This process is shown schematically in Figure 2.4. Here we depict a neural network with a single hidden layer, i.e., a layer of neurons that only receives and forwards information between other layers of neurons. The dimensions given by the number of input and output units will be matched by the output function.

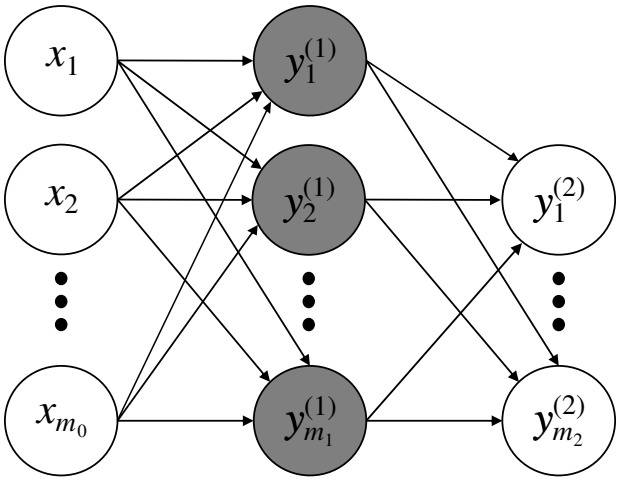

Figure 2.4: A fully connected feed-forward neural network with a single hidden layer. $x_1, x_2, ..., x_{m_0}$ represent input to the network, with x_0 used as a bias introduced into the activation function, $y_1^{(1)}, y_2^{(1)}, ..., y_{m_1}^{(1)}$ are neurons each receiving input from the initial layer with y_0 acting as the bias. $y_1^{(2)}, y_2^{(2)}, ..., y_{m_2}^{(2)}$ represent the final output of the network. Adapted from Ref. 3.

The neural network depicted in Fig. 2.4 is feed-forward, meaning that closed cycles in the network graph are forbidden and each layer only propagates to the following layer. This implies that the link between input and final output can be defined as an explicit function. The fundamental equation for the input to neuron y_i from m neurons x_k is defined as[47]

$$y_i = f(z_i) = f\left(\sum_{k=0}^{m} w_{ik} x_k\right) \tag{2.1}$$

and, for the case of a multilayered system is given by

$$y_i^{(l)} = f\left(z_i^{(l)}\right) = f\left(\sum_{k=0}^{m_{(l-1)}} w_{ik}^{(l)} y_k^{(l-1)}\right) \tag{2.2}$$

where $m_{(l)}$ represent the number of neurons in the l^{th} layer, and $w_{ik}^{(l)}$ are the weights for neurons of index k fed into neuron i located in layer l.

The activation function f in (2.2) determines the output of the unit[3]. As we will see, these functions play a crucial role in error backpropagation. This makes it necessary and important

to note their differences. Activation functions that are monotonic are desireable because they prevent different inputs from mapping to the same output. For this reason, it generates an additional extrema in the error surface. The three most commom choices for activation functions are[46,48]

$$\text{sigmoid}(z) = \frac{1}{1+\exp(-z)} \quad \Rightarrow \quad \text{sigmoid}'(z) = \text{sigmoid}(z)(1-\text{sigmoid}(z)), \tag{2.3}$$

$$\text{softmax}(z,i) = \frac{\exp(z_i)}{\sum_k \exp(z_k)} \quad \Rightarrow \quad \text{softmax}'(z) = \text{softmax}(z,i)\Big(\delta_{i,j} - \text{softmax}(z,j)\Big), \tag{2.4}$$

$$\text{relu}(z) = \max\{0,z\} \quad \Rightarrow \quad \text{relu}'(z) = \Theta(z) = \begin{cases} 0, & z \leq 0 \\ 1, & z > 0 \end{cases}. \tag{2.5}$$

The softmax(z) and logistic sigmoid(z) functions are monotonic and smooth, and their output permits a probabilistic interpretation with all output restricted between 0 and 1. This is in contrast to the rectified linear unit (ReLU) relu(z) function, which features the Heaviside unit step function $\Theta(z)$, as its derivative.

In this thesis we will mostly use the relu(z) function, which must be properly calibrated, as we will see in Chapter 4. With the activation functions playing their role, the missing step in neural network training amounts to the determination of the weights w_{ik} in (2.2). Starting with the set of training data selected, and assuming the network has sufficient neurons, there exists a set of weights that will generate a best approximation to the target function. However, it is also necessary to know the performance of our neural network as the amount of training data increases and the weights adjust to the new data. The way to find the difference between the target function and its approximation is through the choice of error term.

There are two primary choices for the evaluation of the error[49]. The first is the sum of squared errors, which is given by

$$E = \sum_n E_n = \frac{1}{2}\sum_{nk} \left(y_k(x_n) - t_{nk}\right)^2, \tag{2.6}$$

where t_{nk} is the k^{th} entry of the n^{th} target value. The second is the cross-entropy error function

$$E = \sum_n E_n = -\sum_{nk} t_{nk} \log\left(y_k(x_n)\right). \tag{2.7}$$

In both cases, using $y_i(x_n) = f(z_i)$, they satisfy

$$\frac{\partial E_n}{\partial z_i^{L+1}} = \frac{\partial E_n}{\partial y_i^{L+1}} \frac{\partial y_i^{L+1}}{\partial z_i^{L+1}} = (y_i(x_n) - t_{ni}) f'\left(z_i^{L+1}\right) \tag{2.8}$$

where E is the error resulting from batch training, and $\{x_1, \ldots, x_n\}$ is the set of n values inputted to the neural network. In order to update the weights we use the total error. This is counter to the case of tochastic training, where based on the individual errors E_n the weight are updated. As many training sets tend to have redundancies, batch training tends to be the preferable approach[3]. In general, E may be considered to be a surface through the space of all weights w_{ik} for which the desired goal is find a global minimum under the criterion $\nabla E_n = 0$. In the next section we will discuss one of the most commonly applied methods for achieving a weight optimization for any type of feed-forward neural network.

2.1.7 Gradient-based optimization

Each layer in a neural network, depending on its activation function, transforms its input data in the following manner. In this case the activation function we choose is relu:

$$y_i = \text{relu}\left(\sum_{k=0}^{m} w_{ik} y_k + b_k\right) = \max\left\{0, \sum_{k=0}^{m} w_{ik} y_k + b_k\right\}, \tag{2.9}$$

where w_{ik} and b_k are tensors that represent the data in a neural network. In this way w_{ik} and b_k, along with relu, are attributes of the network. These variables are the weights or trainable parameters, also known as the kernel and bias attributes, respectively[33]. Their purpose is to be a "storage container" for the information learned by the neural network from the training data.

These tensors are filled with small random values at the beginning of the training. Once the network starts to learn, these tensors are gradually adjusted. This process is known as training, and is repeated within a loop, i.e., the training loop, until a sufficiently low loss value is reached and the loop stops. This can be summarized in four steps:

1. Select a set of test targets, denoted y_{actual}, and a set of training samples, denoted x.

2. Execute the forward pass of the network on x to provide predictions, y_{pred}.
3. Calculate the network's loss function on the batch, i.e., the discrepancy between y_{pred} and y_{actual}.
4. If the loss on this batch is above a specified cutoff, update all network weights and go to Step 2. Otherwise, terminate as the network has completed its training successfully.

Once the loss value is below a specified criteria, the network is said to have "learned" to map its inputs, x, correctly to the targets, y_{actual}. Although the first three steps are relatively easy to perform, step four has a degree of complexity because it is rarely just one weight to be updated, but rather hundreds, thousand or even millions. This makes it imperative to automate this step, e.g., using the gradient descent approach.

Modern neural networks are powered by the optimization method known as gradient descent. The main idea is as follows. Considering the relation $z = x + y$, for example, a small change in y only results in a small change in z. If the direction of the change in y is known, one can infer the direction of the change in z.

All of the operations used in our models, such as dot product and matrix add, transform their input in a smooth and continuous way, and are differentiable. Such functions can be combined to create a larger function that is still differentiable. A minor change in the network's coefficients causes a modest and predictable change in the loss value. This is especially true for the function that maps the network's coefficients to the loss of the network on a batch of data. This makes it possible to describe how the loss changes when one alters the network's coefficients in various directions, using a mathematical operator known as the gradient, i.e., "∇". By calculating this gradient, one may use it to update all of the coefficients at once rather than one at a time, minimizing the loss.

Theoretically, having a differentiable function means one can find its minimum analytically. However, there may exist more than one minimum and the main point is find the lowest or "global" minimum. Applying this to a neural network means finding analytically the combination of each weight value such that one obtains the smallest loss function. One way to do this is solving the

equation $\nabla_w f(w) = 0$ for w. This is a polynomial equation of N variables, where N is the number of coefficients in the model.

Although this is straightforward in theory, in practice it is challenging to calculate the gradient of such a complex expression. For this reason we introduce the backpropagation algorithm.

Basically, the algorithm uses the known derivatives of simple operators, such as "+", relu, and "·". UsingBased on these atomic procedures, it is simpler to calculate the gradient of arbitrarily complicated linear combinations. Most importantly, each of the several chained tensor operations that make up a neural network has a straightforward, well-known derivative. For example, the pseudocode 2.1 can be represented in a functional expression with the variables w_1, b_1, w_2, b_2, which belong to the first and second dense layers, respectively. This may be accomplished by using the ·, tanh, softmax, and + operators, as well as the loss function, yielding:

$$\texttt{loss\_value} = \texttt{loss}\left(y_{actual}, \max\left\{0, \tanh(x \cdot w_1 + b_1) \cdot w_2 + b_2\right\}\right) \tag{2.10}$$

To solve (2.10) we need to apply the chain rule to two function f and g's composed function $f(g(x))$, where $x_1 = g(x)$ and $y = f(x_1)$. The gradient is then $\nabla_x y(x) = \nabla_{x_1} y \nabla_x x_1(x)$. This means that if we know the derivative of each function, f and g, we can calculate the derivative of the function $f(g(x))$.

Listing 2.1: Example for backpropagation

```
from tensorflow import keras
from tensorflow.keras import layers
model = keras.Sequential([
    layers.Dense(64, activation="tanh"),
    layers.Dense(1, activation="relu")])
```

However, this is not always easy. This is because of the the network's complexity, which do not necessarily have analytical solutions. For minimizing the error the best approach is often an iterative one. In each interaction the algorithm chooses how to update a weight with Δw_t where

$$w_{t+1} = w_t + \Delta w_t \tag{2.11}$$

and t represent the current time step, or epoch. By epoch we mean an update of the weight w after a batch error has been computed. In other words, an epoch is a single iteration of the self-consistent cycle. Gradient descent is a basic first-order optimization algorithm, and provides a nice balance between accuracy and computational efficiency, especially compared to second-order algorithms such as the Hessian[50]. The gradient ∇E, in gradient descent, is used to weight update Δw_t. This is used in the form

$$\Delta w_t = -\gamma \frac{\partial E}{\partial w_t} + \lambda \Delta w_{t-1}. \tag{2.12}$$

This means that at position Δw_t, the algorithm takes a step into the negative direction of the gradient on the error surface, and where γ is the learning rate and λ is the momentum[51]. The speed of the weight update in each iteration is given by these hyper-parameters and what fraction of the weight from previous iterations is included in the updated weight. The value of γ is of great importance, because it needs to be large enough for the input data to be updated in a meaningful way through the network, but not so high as to the weights to undergo large oscillation and hence fail to converge. To reduce the such oscillations, employing the momentum is often a good strategy. However, it is common to start with a relatively aggressive/fast learning rate and gradually decrease it after a certain error threshold is crossed or a plateau has been reached[52].

For computing the weight updates ∇E_n is a useful error propagation. The respective derivative of an error function E_n at input value x_n with respect to the weight w_{ij} for a feed-forward network with L hidden layers, $L + 1$ being the output layer, is give by

$$\frac{\partial E_n}{\partial w_{ij}^{L+1}} = \frac{\partial E_n}{\partial z_i^{L+1}} \frac{\partial z_i^{L+1}}{\partial w_{ij}^{L+1}} = \delta_i^{L+1} y_j^L \tag{2.13}$$

where δ_i^{L+1} is the contribution of the i^{th} output unit to the error E_n. Following a similar procedure in the l^{th} hidden layer, where $l \in (1, L)$, we obtain

$$\frac{\partial E_n}{\partial w_{ij}^{l}} = \frac{\partial E_n}{\partial z_i^{l}} \frac{\partial z_i^{l}}{\partial w_{ij}^{l}} = \delta_i^l y_j^{l-1} = \left(\sum_k \frac{\partial E_n}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_i^l} \right) y_j^{l-1}. \tag{2.14}$$

Here the definition of error at a given layer is given by the formula for δ_i^l

$$\delta_i^l = f'\left(z_i^l\right) \sum_k w_{ik}^{l+1} \delta_k^{l+1}. \tag{2.15}$$

This equation describes a recursive technique for generating the gradient ∇E_n at the output layer by propagating the error at layer l back through the network to layer $l-1$ and assessing the derivatives $\partial E_n / \partial w_{ij}^{l}$ that update the weights via gradient descent. This process is presented schematically in Fig. 2.5, and described by (2.16).

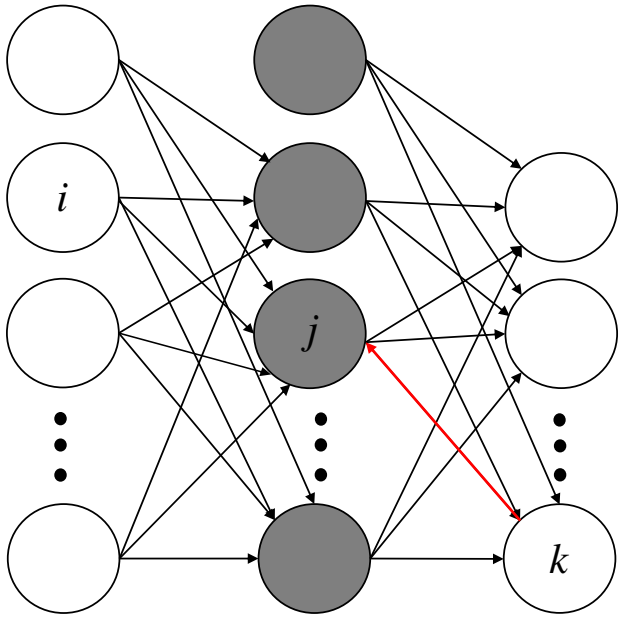

Figure 2.5: Visualization of error propagation through a single hidden layer neural network. Once the error term is computed and used to adjust the weights at the layer of interest, the error terms from the previous layer can be used to determine error terms downstream. Adapted from Ref. 3.

$$\delta_j^{(1)} = \left[\sum_k w_{jk}^{(2)} \delta_k^{(2)} \right] f'\left(z_j^{(1)}\right) \quad \delta_k^{(2)} = (y_k - t_k)\, f'\left(z_k^{(2)}\right) \tag{2.16}$$

The computational cost of using error back-propagation is rather high, as it scales with the total number of neurons and weights w_{ij}[52]. This indicates that shallower networks scale in a better way than deeper ones because of the iterative nature of back-propagation. Furthermore, in larger multilayer networks where every neuron is connected with every other neuron, i.e., each neuron in each layer is connected to every neuron in the next layer, this means that there is an advantage in the propagating cost since the input value is dominated by the number of weights

used. As a result, attempting to limit the amount of input flowing through the network is equally critical to comuptational performance as attempting to keep the total number of neurons as low as possible.

This input flow can be reduced using methods such as convolution[47] and pooling[50]. These methods either compact input data before it enters the network or group it in such a way that connections between layers do not have to be from every neuron in the next and preceding layers to every other neuron in the next and preceding layers.

A neural network may theoretically mimic any continuous target function to a reasonable degree of approximation by using a sufficiently large number of hidden neurons[53] or with just one layer (input layer) according to the universal approximation theorem.

After training and minimizing the error function with respect to the network weights, it is critical that the network be able to generalize beyond the training data set. The simplest way to accomplish this is to test the network's performance on test data that was not used for training, referred to as the test set. A network is considered to generalize if a reduction in error over training data leads to an overall improvement in the network's ability to predict the output of test data.

Regularization is one strategy for avoiding over-fitting and providing better agreement with data other than the training data. Because over-fitting is a symptom of having too many hidden layer neurons for the problem, adding a penalty to the error function, as in the case of L2-regularization[49],

$$E(\omega)^* = E(\omega) + \eta w^T w, \tag{2.17}$$

helps to ensure that the network's weights w are inclined in an exponentially way to zero. This effectively "kills" any neurons that do not contribute substantially to the overall error. Choosing the number of hidden layers, the number of neurons to use in each layer, the activation and error functions, and hyperparameters such as γ, λ, and η, are all important considerations that often necessitate a significant amount of trial and error before obtaining a computationally efficient network that approximates well the desired target function.

Adam[54], which refers to adaptive moment estimation, is one of the most commonly used weight update methods, known as optimizers, in addition to the standard gradient descent method. At each time step t, Adam's learning rate γ in Eq. 2.11 takes into account estimations of the first and second moments of the gradient of the error $g_t = \partial E/\partial w[t]$, at each time step t. Within the Adam optimizer, the first and second moments of the gradient, m_t and v_t, respectively, are derived using exponentially moving averages and the gradient evaluated in the current batch.

$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\, g_{t+1} \quad v_{t+1} = \beta_2 v_t + (1-\beta_2)\, g_{t+1}^2, \tag{2.18}$$

where the values for the hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are commonly employed in implementations of neural networks[48], with the initial moments set to zero. Due to this initialization, the estimators are biased towards zero. For this reason, a bias correction must be implemented so that the expected values of the moments match those of powers of the gradient.

This is accomplished through

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}. \tag{2.19}$$

Finally, (2.12) in the framework of the Adam optimizer is updated to

$$\Delta w_t = -\gamma \hat{m}_t / \sqrt{\hat{v}_t + \epsilon}, \tag{2.20}$$

where in order to prevent division by zero ϵ is introduced into (2.20). Note that employing the gradient's moving average rather than the gradient itself usurps the role filled by (2.11). While Adam has some drawbacks, such as not generalizing well to all issues that the conventional gradient descent approach is equipped to solve[55], its reduced time makes it a viable choice for many neural network implementations, including the ones utilized in this thesis.

2.2 White Noise

Statistics frequently define white noise, which is used in signal processing, as a random signal with equal intensities at at various frequencies, a consistent power spectral density, and samples

that are a series of unconnected random variables with no mean and little variance[56]. In some circumstances, it might be necessary for the samples to be independent and to have the same probability. In some cases works by specifying parameters that make that the input is random, unrelated, and has zero mean. Even when using binary variables, white noise can be created. A series of 0s and 1s, for instance, would be white if the series was statistically uncorrelated. A normal distribution, as well as any continuous distribution of variables, can also be white. Numerous scientific and technological domains, including physics, acoustical engineering, telecommunications, and statistical forecasting, use the phrase with these or similar meanings. Also white noise theory can be incorporated into neural network models. For instance, a data scientist might purposefully introduce noise to their neural network in order to enhance both the model's overall accuracy and training performance. Instead than referring to a specific signal, white noise describes a statistical model for signals and signal sources. White light gave rise to the term "white noise" , despite the fact that most white light does not have a flat power spectral density over the visible range[57]. There are a few ways to include the fundamentals of noise in a model. Dropout regularization, a method that is frequently used to make use of noise[58], is one such method. This technique forces the network to train and learn under constrained bandwidth by randomly zeroing out some of the network's units. In figure 4.5 is show how looks like the white noise in the data set used in this thesis with an histogram.

2.3 Standard and Poor's 500

The Standard and Poor's 500, sometimes known as the S&P 500, is an index of 500 of the largest publicly traded firms in the United States that is weighted by market capitalization[59]. Because the index takes other factors into account, it is not a precise list of the top 500 U.S. corporations by market cap. It is recognized as one of the best indicators of the performance of well-known American stocks and, by extension, the performance of the stock market as a whole because it is one of the most widely used equity indices[60]. To better understand, let's define a capitalization-weighted index, sometimes referred to as a market value-weighted index. It is a kind of stock market index where each index component is included in proportion to its entire

market capitalization (commonly called market cap). The index will be weighted more heavily using the capitalization-weighted technique for index components whose market caps are higher. Smaller companies' success will proportionally have less of an impact on the performance of the index as a whole. The price-weighted, fundamental-weighted, and equal-weighted index construction methods are further approaches for determining the value of stock market indices. A hypothetical portfolio of investment holdings known as a market index is used to represent a certain area of the financial market. The prices of the underlying holdings are used to calculate the index value[61]. Some indices are valued according to market capitalization, revenue, float, and fundamental weighting. The unique influence of each item in an index can be modified through the use of weighting. To track market trends, investors monitor various market indices. The Dow Jones Industrial Average (DJIA), the S&P 500 Index, and the Nasdaq Composite Index are the three most widely used stock indices for monitoring the performance of the American market. The Bloomberg U.S. Aggregate Bond Index is one of the most widely used substitutes for U.S. bonds in the bond market, where Bloomberg is a key provider of market indexes. These portfolios are frequently used as benchmarks or for creating index funds because investors cannot invest directly in an index[62].

Chapter 3

Methodology

3.1 Data

The data we use in this thesis is the stock market data of the S&P 500 index obtained from Ref. 63. After initially preprocessing the data set, we obtain 18234 data entries spanning the period from January 3rd, 1950, to June 17th, 2022, containing the date, open, high, low, close, adjusted close price in $, and total volume traded. To get an idea of how the data looks, the data values for the third week of June, 2022, are shown in the table 3.1. Within economics the data provided in Table 3.1 is defined as:

- Open: the opening price of a share on a given date.
- Low: the lowest price that a share reaches on a given date.
- High: the highest price that a share reaches on a given date.
- Close: the closing price of a share on a given date.

Table 3.1: S&P 500 index date, price open, high, low, and close in $ and volume in shares for the third week of June, 2022.

Date				Price ($)				Volume
Weekday	Day	Month	Year	Open	High	Low	Close	(shares)
Friday	17th	of June	2022	3665.90	3707.71	3636.87	3674.84	6954110000
Thursday	16th	of June	2022	3728.18	3728.18	3639.77	3666.77	4511200000
Wednesday	15th	of June	2022	3764.05	3837.56	3722.30	3789.99	44746100000
Tuesday	14th	of June	2022	3763.52	3778.18	3705.68	3735.48	4126400000
Monday	13th	of June	2022	3838.15	3838.15	3734.3	3749.63	457282000

- Adjusted Close: the closing price after adjustments for all applicable splits and dividend distributions of a share on a given date.
- Volume: the total number of shares traded on a given date.

Here a share is a financial market unit that is used in real estate investment trusts, limited partnerships, and mutual funds.

3.2 Computational details

The computational part of this thesis is divided in two part. The first is a simple analysis, once the data was arranged properly, the next step was get information through pictures representing the relation of the different column in the data the original as well as the artificial made, all this first part was just done in a jupyter notebook with several libraries like pandas. Although it got negative result it was what it must expect when it comes to an index as volatile as the S&P 500, after this result one way to make a complex analysis and try to predict the daily close price difference from the daily volume difference was a neural network. The first analysis It consists of obtaining a graphic form of the relationship of the different columns, as explained

above, graphs were made such as date vs. volume, as well as the daily difference in the close price vs. the daily difference in volume. But in addition to these graphs, functions such as .corr() offered by the panda library were used, in this function the Pearson's method was used to find the correlation coefficient between the columns, and the important columns gave an extremely small value indicating a nullity in the linear correlation. The test statistic that assesses the statistical association, or relationship, between two continuous variables is called Pearson's correlation coefficient. Because it is based on the method of covariance, it is regarded as the best method for determining the relationship between variables of interest. It provides details on the size of the association or correlation as well as the relationship's slant. Since the method of Pearson and others offered by pandas only yield the linear correlation coefficients, for this reason it choose the neural network not only to be able to predict but also to know what is the correlation coefficient that exists between these columns but due to the white noise presented in the data set it is not possible to predict just using the daily difference in volume.

3.2.1 First Analysis: Simple Analysis

In this section, different functions given by multiple libraries in Python are used to find the relationship that exists between the columns of the data obtained, as well as order and edit them in a way that it can works with the data in a simple form. For these purposes it was used Pandas library, which is a Python library that is used for data science/data analysis and machine learning. Additionally, Pandas it is based on the NumPy library, another Python library, which is a library used for working with arrays, also has functions for working in domain of linear algebra, matrices, and so on, besides this library was used directly for generate arrays or transform from pandas data frame to array. Furthermore, for have a representation of how is the behavior of one column vs other columns, for instance, date vs volume, it was used the Matplotlib library, which is used for creating static, animated and interactive visualization in Python.

The main idea of the simple analysis is obtaining pictures to get an interpretation of the relation between two columns, and besides to get the correlation coefficient between columns. The method that was used to get the correlation coefficient was the Pearson's method, which is a measure

of linear correlation between two sets of data, because it wanted to know the linear correlation between all the columns especially between the absolute value of daily volume difference and the absolute value of daily close price difference, which were created with the .diff() method of the pandas library, this method work making the difference between i^{th}– (i^{th} + 1) columns, and give a nan value in the first row (in new columns created to save this difference), for that reason the first column was dropped, with the .drop() method also owned by pandas, but if with change the argument axis in .diff() to $axis = -1$ (by default is $axis = 1$), the last row gives the nan value

3.2.2 Second Analysis: Neural Network

Due to the extremely low value of the correlation coefficient, demonstrating complete non-linearity, between the columns of data of interest (that is the daily close difference vs daily volume difference). It took the decision to use the artificial neural network that from his creation until now has shown be a powerful tool to make, between other human-brain tasks, regressions with data sets that has shown non-linear behavior, in this case (this thesis) between two different columns.

To start, the input data was pre-processed to avoid overfitting, since the data has a region which concentrate the 92% of the set, between 1×10^4 to 5×10^9, which is a problem when it left the neural network choose randomly (truly is pseudo randomly because the chose is based on a function that generate those random selection) points for training the network, thus it separate the data in a properly way using the binning method. Binning or discretization is used for the transformation of a continuous or numerical variable into a categorical feature (interval in our case). Also it was done three time the binning each base in different columns that are explained below.

The first method consist in binning the data symmetrically, using the .qcut() function, which is a pandas library function used for data grouping, such that each interval has the same (or almost the same) amount of data point, it was selected $q = 10$, where q is the number of bins, and how are these are binning is show in Figure 3.1, where class is a way to label the interval, these labels

were chosen for simplicity, the order in the table is the same as it was printed in the program by the function `value_count()`, which is a function that pandas has to count the amount of point that each bin has. However, this method did not work, since it is the same as not performing a binning, since the probability of choosing data that is in the aforementioned region is very high, since this represents a large part of the set (almost all of it) and the function .randint(start, end), which is a function to choose randomly (pseudo-randomly) integers among those given in its argument, it is a function that is based on statistical methods, therefore the selection of the points are mostly from just one region.

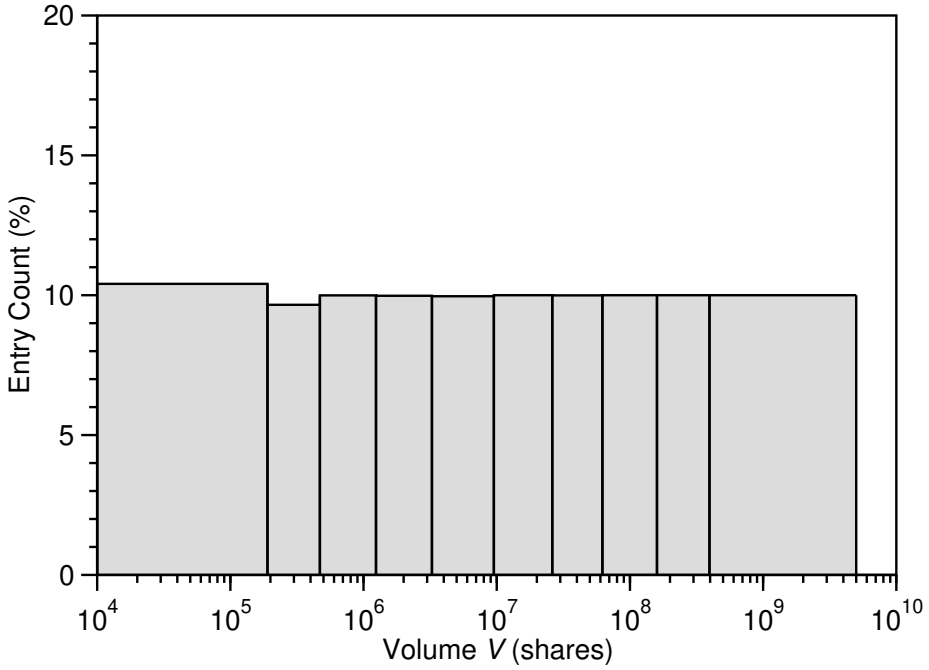

Figure 3.1: S&P 500 entry count in % versus volume V in shares for symmetric binning.

The second method consist in binning the data without symmetry, using a for loop, that generate an array that contains the start and end of each intervals, using the maximum and minimum value of the guide column, in this case the absolute value of daily volume difference (VD just for simplicity, see algorithm appendix). Then it was created the intervals through other for loop, and finally it gets, the interval show in Figure 3.2, again here it is was used the *.qcut*() function, that allow insert as an argument a personalized interval. Here, the data points also were

selected randomly, nevertheless due to the first interval that accumulates practically the entire set, a minimum quantity limit was created so that the points of certain intervals are considered viable to be selected randomly, if the minimum number of necessary points is not met, then 75% of the points of that interval are chosen, since it is necessary have enough information of some region to obtain a better interpretation of that specific region, in this case the variable was set as $limit = 150$.

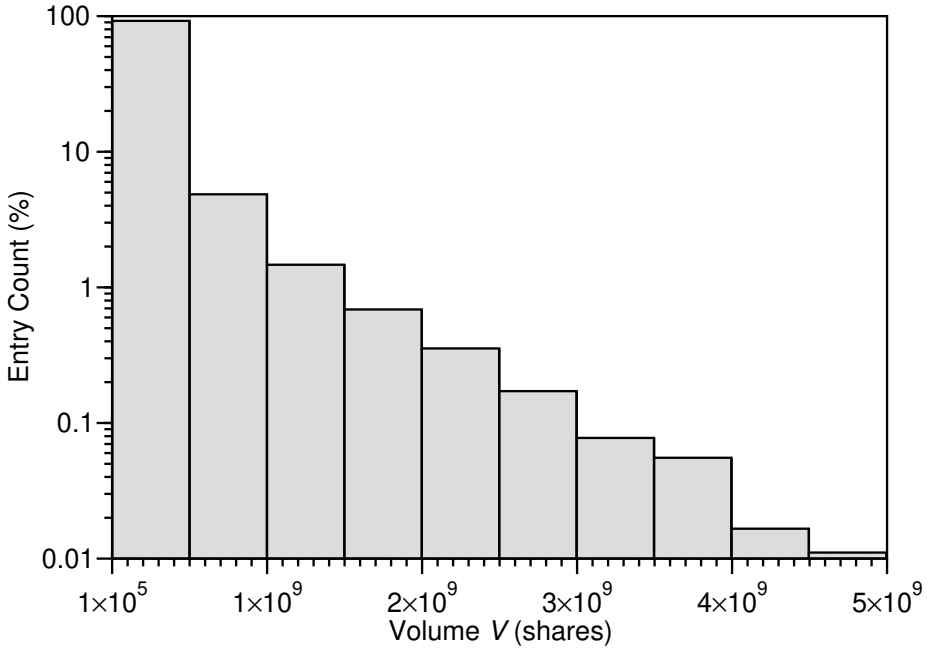

Figure 3.2: S&P 500 entry count in % versus volume in V shares for even binning.

The third method is basically the same as the second but following the recommendation of Keenan L.[3], which is binning but in a logarithmic scale, so it took the logarithm in base 10 of the columns of interest and create a new one with the result of this procedure. Then the following step is the same as the second method, it generates the personalized interval, then it use *.qcut*() to finally get the intervals show in Figure 3.2. Keenan use this methodology to avoid the overfitting since the data set show, under this transformation, a more uniform distribution, and when the data is chose randomly it has more probability to choose point that belong to regions that not represent a big part of the set, thereby avoiding the overfitting.

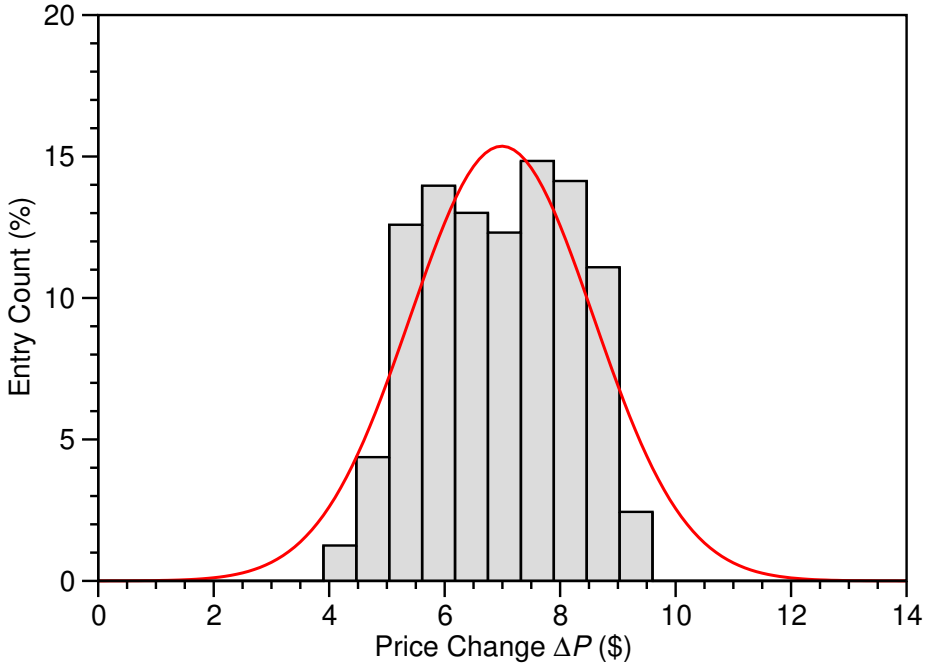

Figure 3.3: S&P 500 entry count in % versus price change ΔP in \$ for even binning and Gaussian fit (red solid line) with average μ = \$6.99 and standard deviation σ = \$1.59.

Once it has chosen the points, the next step is the creation of the the training data set and the test data set. The training data are the points selected randomly, while the test data are the remaining points, the training data is divided in *train_data* and *train_target*, where the former are the data while the latter are the answer, following the concept of figure 2.2. Whereas the test data is divided in `test_data` and `test_target`, where the former are the data while the latter are the answer, following the concept of figure 2.2. After defining each set, both sets are normalized, following the recommendation of reference 33, such that there is not a great difference between the values, since the learning of the network that uses the weights that are given by these values would be affected, for which normalization is necessary to avoid a bad learning and therefore a high errors. Once done this the procedure is feed the neural network with these data point and get the result.

It wanted prove the universal approximation theorem, which said basically that every neural

network can approximate any function with just one layer and at least the same number of neurons as data points. The purpose of proving the theorem is show that the neural network is well building and work properly. Then with the result of that model test using the test data to see how good is the performance. This process is made when it is selected 1 data point per bin, when the bin$\bar{8}$, and when bin $\bar{1}6$, it was selected 1 and 2 points per bin. As the theorem said is just one layer with neurons 4 times the amount of train data ($N_{neuron} = 4N_{data}$), with relu as the activation function and just one neurons in the output layer.

Chapter 4

Results & Discussion

The Results & Discussion chapter is split into two sections, which show the two different ways of how was analyse the data in this thesis: the first section shows a simple analysis of the set of data through many libraries in python, the second section corresponds to a complex analysis using neural networks through TensorFlow library using the API Keras in python.

4.1 First Analysis: A simple analysis

In this section it will see how it was analysed the data through simple line of code and how those result are interpreted in order to choose a neural network to perform a more exhaustive analysis. So, all the plots were done with grace in Ubuntu due to the tool present in this program to do a good plot with simple steps, all the correlation values were done using the function *.corr*() which is part of the pandas library in python.

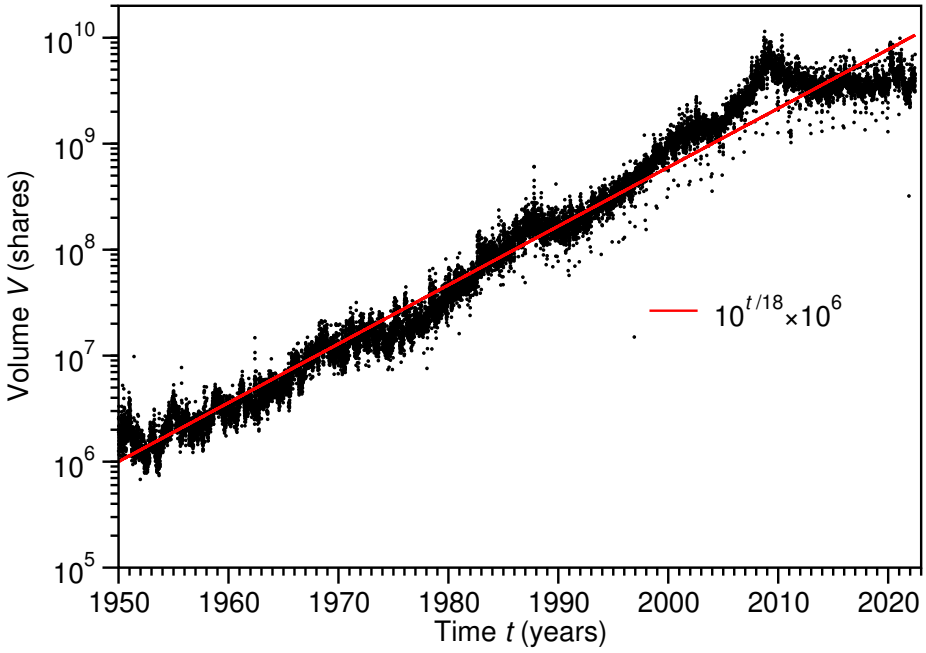

Figure 4.1: Historical S&P 500 volume V in shares versus time t in years (black dots) from January 3rd 1950 to June 17th 2022 from Ref. 63, and fit of the form $10^{t/18} \times 10^6$ (red line).

4.1.1 Volume vs Date

Figure 4.1 shows the volume vs date, and at first look it can note that through the time (1950-2022) the amount of shares (volume) has increased exponentially, so the overall behavior is exponential through the time, although the last decade the behavior has been constant (flat region),which indicates a decrease in the US economy in the last decade, although it grew during the coronavirus pandemic (2019-2021) as can be seen in its last peak shown in the graph, besides from 1950 to 1990 the picture show a turbulent behavior because of its many peaks and valley which indicate the several historical event that affect the global economic, for instance the missile crisis in 1962 which is show in the map with a valley but before that it was a peak, and consequently the biggest economy in the world, the US economy. Also, in a general way the picture is like a ladder since it pattern repeated as step, the first one is in the early of 1970s', the next in the early of 1990s', then in the early of 2000s', after the trend it climbed pretty fast in five years to stop and decreased,

which is the step if it is noted that in each step decrease and after increase. An important fact that we see is that the volume, on average, grows by an order of magnitude every 18 years.

4.1.2 Open vs Volume and Close vs Volume

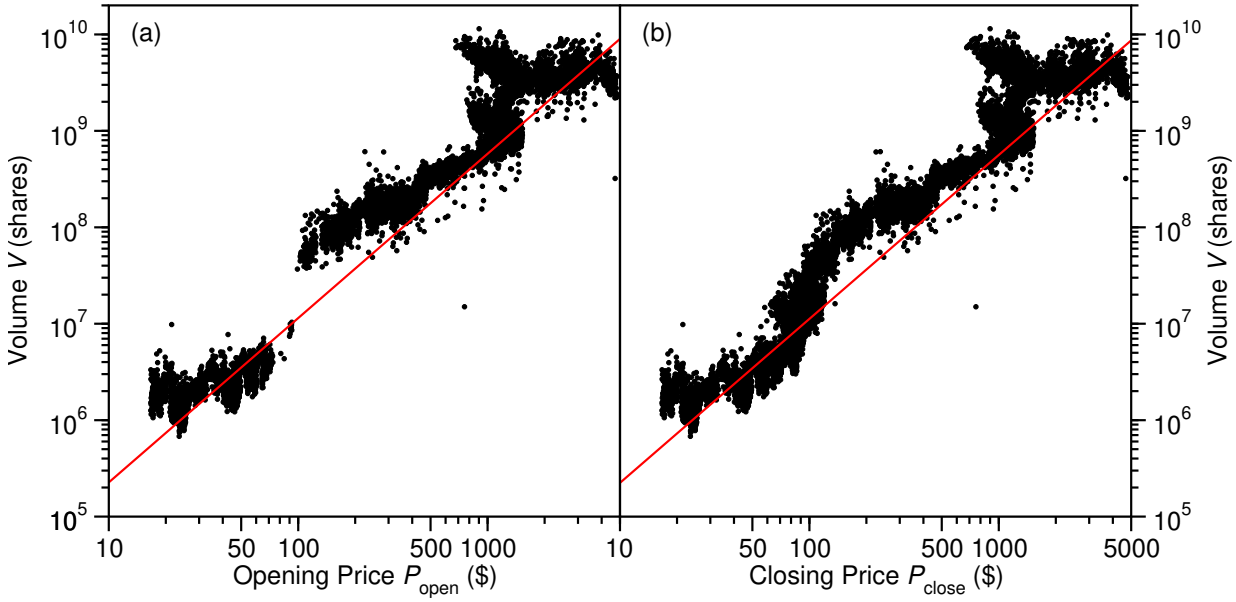

Figure 4.2: S&P 500 volume V in shares versus (a) opening price P_{open} and (b) closing price P_{close} in \$ (black dots) and fits of the form $4467P^{1.7}$ (red lines).

Figure 4.2(a) shows how the open price has been increasing as increase the volume, since the volume is in logarithmic scale, the figure indicate that the overall behavior is exponential, although when volume reach the region between 10^9 and 10^{10} the exponential behavior start to decrease and increase , in the last part a jump in the price indicate that in very short period of time, and if it is compared to the figure 4.1, that incredible increase and decrease is due to the "the stock market crash of 2008". An important aspect of the open price in the set of data got it is that between 1×10^7 and 4×10^7 there is a gap of information that is because in that period of time exist (1960 to 1981) a lack of information of the open price was caused by the not registration of

the open price.

The variation of the open price and close price indicate a power law of the form $V^{0.59}$, or $V \sim 4667\ P^{1.7}$. The figure 4.2(b), it is practically the same as 4.2(a) but the main different is that the gap of information does not exist since the close price in the period of time mentioned before is correct documented with information that show the exponential increase in the overall behavior, other different, although quite obvious, is the price, because those different are 1 to 5 dollars or in some case cents therefore the difference between these two plots are no notorious.

4.1.3 Correlations

Table 4.1: Percent correlation between components of the S&P 500 index.

	Open	High	Low	Close	Volume
Open	100.0	99.9	99.9	99.9	78.8
High	99.9	100.0	100.0	100.0	78.7
Low	99.9	100.0	100.0	100.0	78.4
Close	99.9	100.0	100.0	100.0	78.6
Volume	78.8	78.7	78.4	78.6	100.0

In order to know in a numeric way the relation between the variables of the data it was used the .corr() function, in table 4.1 is summarize all the correlation values between numerical variables. Furthermore, because the date format is not a numeric one it was used the function *.to_numeric*() in the date column which simple works converting the column into a numerical value then it was used again the function *.corr*() to find the correlation between just the following three numerical values: Volume, Open and Close, these values are shown in table 4.2, since this are a more general variables than the others and it can conclude in a more general fashion how is the behavior of the variable between them.

In a general view, in both table, all correlation coefficients are positive, so both increase,

Table 4.2: Percent correlation with time for S&P 500 volume in shares and open and close prices in $.

	Volume	Open	Close
Date	77.2	83.3	83.2

thus this is an evidence of the increase of the stock market and the US economy, and this can be confirm by the table 4.2 in this table also all the correlation coefficients are positive meaning that through the time the S&P 500 index has increased and confirming the picture 4.1 and all the correlation coefficient of the table 4.1.

4.1.4 Daily Close Difference vs Volume

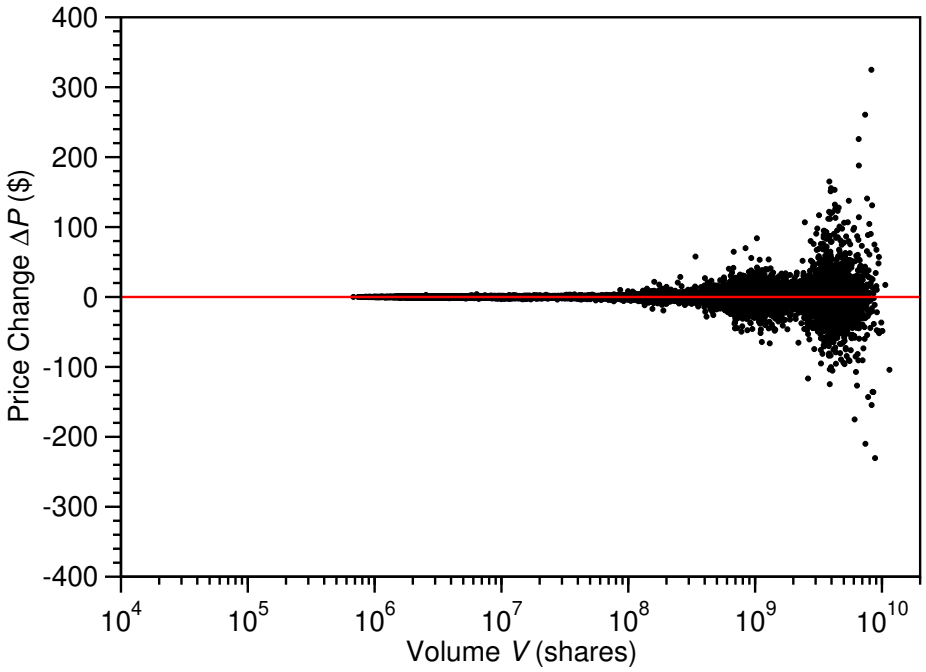

Figure 4.3: S&P 500 price change ΔP in $ versus volume V in shares (black dots).

The figure 4.3 shows how the daily close price difference increase as the volume increase, the

volume is in logarithmic scale and the overall behavior follow a very low exponential increase and decrease indicating a exponential behavior. the daily difference is between today and yesterday, where today is choose as the first row and yesterday the second row and so on, the sequence that is used it is following the date column. Most of the difference in the price are near zero when the volume belong to the region between 6×10^5 and 1×10^8, then the different start to increase reaching like a trumpet shape or view from another way the difference increased until forms a kind of oval with the major axis as the predominant axis that occurs because in the region start the increasing/decreasing of the first "v" inverted mention in the previous section, then start again increase the difference until reach a small circle or a oval but with the minor axis bigger that the major axis, this is because the vertical line mentioned before, which increase to a very high price in a very short time. Furthermore, in this region exist point in which the difference reach more that $300 which indicate the big fall in the prices and confirms the abrupt drop mentioned in the previous section.

The daily close price difference increase as the volume increase, the volume is logarithmic scale and the overall behavior follow a very low exponential increase or decrease indicating a exponential behavior, besides most of the difference in the price are near zero when the volume belong to the region between 6×10^5 and 1×10^8 from this point start to increase to reach two different oval shape separated for a bridge all of the shapes are direct related to the previous section when the two "v" inverted appear and the vertical line also.

4.1.5 Daily Close Difference vs Daily Volume Difference

The figure 4.4 shows how the daily close difference increase as the daily volume difference increase, the volume is in logarithmic scale and the overall behavior follow a very low exponential increase and decrease indicating a exponential behavior. the daily difference is between today and yesterday, where today is choose as the first row and yesterday the second row and so on, the sequence that is used it is following the date column. In contrast to the figure 4.3, this do not present the oval, what it show is just a continuous increase without any drop in the shape indicating the continuous exponential increase in the difference of the close price as the volume

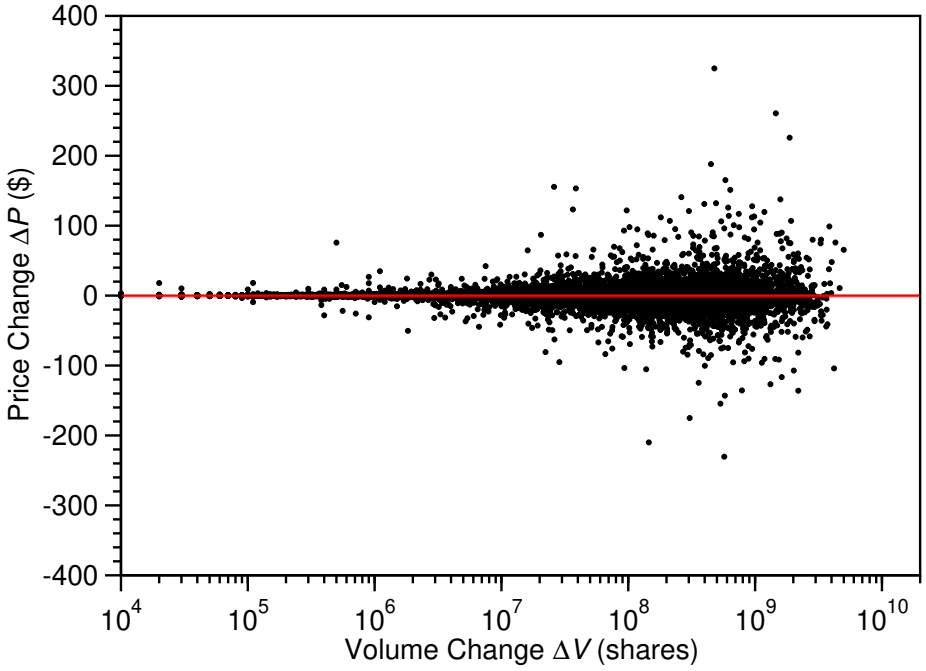

Figure 4.4: S&P 500 price change ΔP in \$ versus volume change ΔV in shares (black dots).

grows and therefore as the figure 4.1 indicate a exponential increase in the volume through the time, so the difference in price will be still increasing through the time.

4.1.6 Negligible volume

The figure 4.5 show how it is the distribution of the daily difference in close price when the daily volume difference is between -10000 and 10000, when the difference belong to this interval the difference in volume is consider negligible. The distribution follow a Gaussian distribution, with average μ = \$0.06 and standard deviation σ = \$0.19 for small volume changes, i.e., less than ten thousand shares as it can note with the Gaussian bell over the histogram, thus when the difference in volume it is extremely low the difference in close price has a big probability to be between -0.4 and 0.4 which are just 40 cent meaning practically no change in the price, therefore when the variation in volume in volume is negligible the variation in the close price also is negligible.

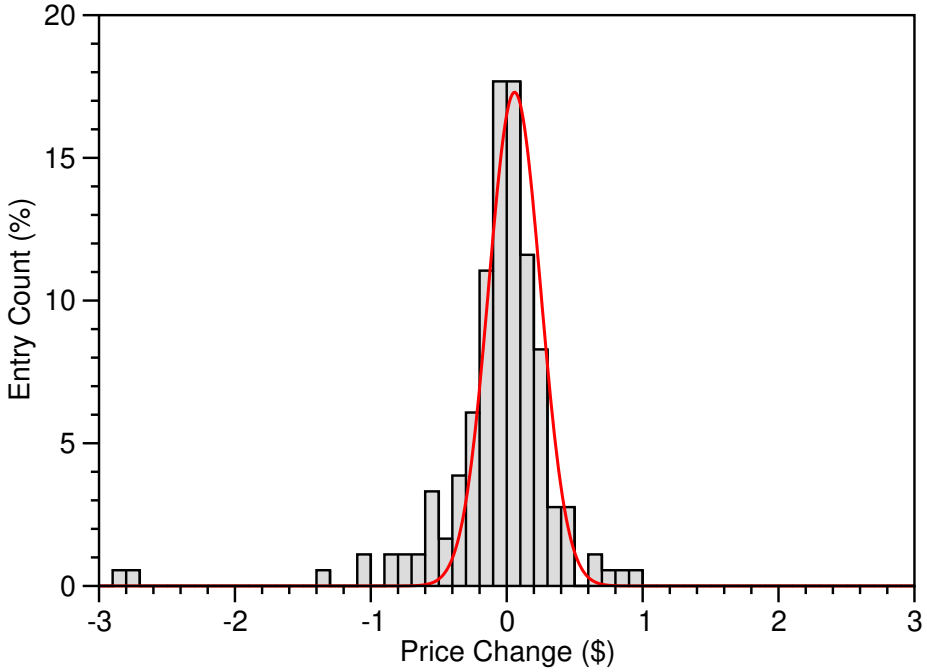

Figure 4.5: Entry count in % of S&P 500 versus price change ΔP in \$ and Gaussian fit with average μ = \$0.06 and standard deviation σ = \$0.19 for small volume changes, i.e., less than ten thousand shares ($|\Delta V| \leq 10000$),.

Also, the cases when the difference in volume is very low is 189 out of 18234 data point, and do not represent enough weight to affect the overall behavior of the daily difference in volume vs the daily difference in close price, thereby this just represent a very low white noise.

4.2 Analysis Using Neural Networks

All the result presented in this section where done in the HPC at Yachay Tech University, in the table 4.3 we can find the time that each one took as well as the number of neurons in each network, the amount of sample per bins and data points. Furthermore, in table 4.3 we can find the epoch at the code stop and the mean square error. This table indicate, that the convergence is almost the

Table 4.3: Number of samples, bins, samples/bin, neurons, neurons/sample, epochs, training time and mean square training errors for neural networks trained with S&P 500 index normalized change in volume as input and normalized absolute change in price as expected output.

# samples	# bins	$\frac{\text{\# samples}}{\text{\# bins}}$	# neurons	$\frac{\text{\# neurons}}{\text{\# samples}}$	# epochs	time (hh:mm:ss)	error (%)
8	8	1	32	4	5×10^6	39:43:31	0.0615
16	16	1	768	48	5×10^6	34:00:59	0.0101
32	16	2	3840	120	5×10^6	46:50:16	0.0756

same in the three cases, with a set number of epochs, but if we increase the amount of neurons the time increase by hours, also it is possible to reduce the mean square error (mae) with more epochs. Although, it is not necessary to still reducing the mae since the test date it will give the same amount of error.

An interesting result it is that with fixed number of epochs or training steps it could obtained rather consistent mean squared errors. Nevertheless, the middle network, one sample per bin and 48 neurons per sample, present the lowest time and error, this suggest both binning and having a sufficient number of neurons per data sample is important to be able to effectively describe the training data.

Figure 4.6 show actual versus predicted normalized absolute change in price of the S&P 500 index using a 32, 768 and 3840 node neural network, respectively, with training (blue dots) and test (black dots) data compared to $y = x$ (red line). The result were obtained in the epoch 5000000, with an error of 0.0615%, 0.0101% and 0.0756%, respectively. Also it can note that add more nodes and more training samples improve the description for the test data, even though the description is still pretty poor. This suggests that with more nodes and more training data, it can improve the quality of the neural network for reproducing the test data. Also, although the results show that, with enough time & computational resources, it can reproduce the training data, in order to get a description of the test data, it most probably need a deep learning / multi-layer algorithms, to describe the clearly the stochastic nature, non-linearity, and high degree of

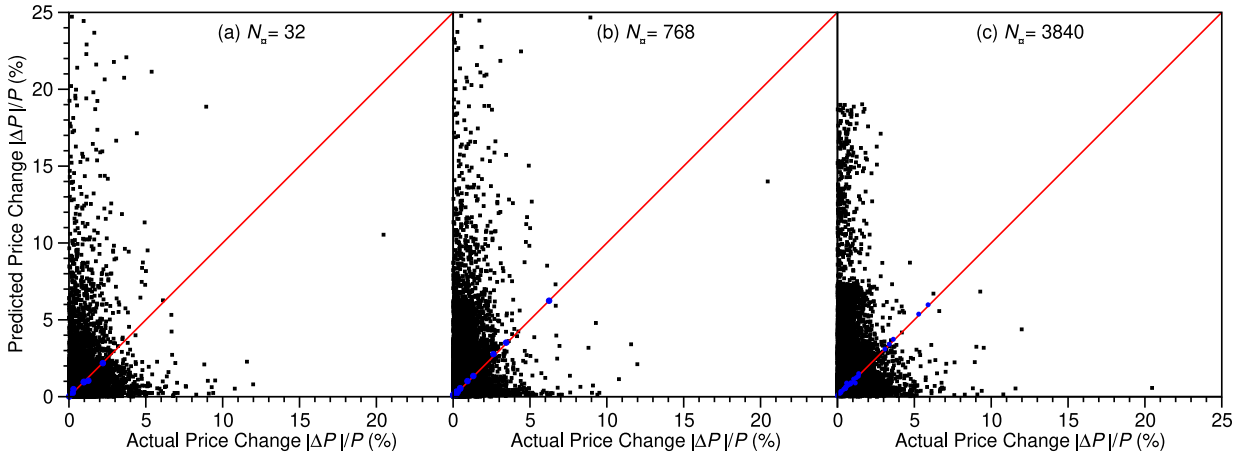

Figure 4.6: Comparison of predicted and actual normalized absolute price change $|\Delta P|/P$ of the S&P 500 index using neural networks with (a) $N_¤ = 32$, (b) $N_¤ = 768$, and (d) $N_¤ = 3840$ neurons and (a) 8, (b) 16, and (c) 32 training samples (blue circles) and test data (black squares).

complexity of S&P500 index.

Although, the number of data items chosen are small, representing 0.2%, 0.1% and 0.04%, respectively, and an insight is increase the number of data points. The increase of the number of sample present some problems. This is due to the fact that the data are so close to each other, that choosing a larger number of data would generate a rather high error and the learning of the network would be affected. In addition, increasing the number of data would imply increasing the number of neurons and with it the time for the calculations. It should also be noted that the multiplicative factor of the number of neurons per sample is different in each case, because if they are kept fixed, the error when training the network is very high, for which reason it was necessary to change these parameters in each training raising them gradually until finding the numbers indicated as shown in the table 4.3. However, these were not the only parameters that were adjusted, the learning rate of the Adam activation function was changed to 0.001, both higher and lower values generate a large error in the network when it is trained.

To summarize, all these results indicate that with few sample it is possible to predict success-

fully the train data, as the universal theorem said, indicating that it is possible to create a predictive neural network model with the sufficient amount of neurons for each data point. However, with just a single-layer neural network the computational power and time for the amount of data that the set has is incredible high, and it can not be possible with the resources used in this thesis. Also, this indicate that to predict the test data it is necessary more samples to train the network, but this option has to deal with problem like overfitting. Furthermore, other conclusion is that a deep neural network can works, since multi-layer network (deep learning) can describe better the behavior of the index than a single-layer network due to the stochastic nature, non-linearity, and high degree of complexity of S&P500 index.

The source code to recreate these results as well as the pickle file that contains the associate weights of each case to recreate the loss and validation loss function are in Ref. 64.

Chapter 5

Conclusions & Outlook

In this thesis, we have searched for a predictive model of the daily variation in closing price based on the daily change in the volume of the S&P 500 index via the training of a neural network with the data obtained from Ref. 63. We have also performed two different sets of analysis to see what information that the data hasmodel needs in order to understand the market's behavior.

First, a simple analysis of the S&P 500 data obtained was carried out to observe its behavior. We have shown the dependence of volume time, which exhibits exponential growth, as is somewhat expected. Specifically, we see that the volume, on average, grows by an order of magnitude every 18 years. We also find a very strong correlatiion between the opening and closing price, with their relationship with the volume taking the form $V^{0.59}$. Although both show large fluctuations, their behavior generally exhibits a power-law behaviour.

The relationship between the different columns in the data set was also addressed analytically. Specifically, the correlation coefficients were obtained through the Pearson method. All of these coefficients are summarized in Tables 4.1 and 4.2. Overall, all prices (open, high, low, close, adjusted close) on a given day are very highly correlated ($\sim 100\%$), whereas their correlatioons with the volume are much weaker ($\sim 79\%$).

Additionally, the relationship between the daily change in closing price with volume and daily volume change was analyzed. Both present behavior, in general, of a weak exponential, since their growth has not been explosive, as was the case for the opening and closing prices' dependence on the volume. Overall, we find that the daily price change, volume change, and volume, all increasing exponentially with time but more smoothly for the volume change relative to the change in closing price. Additionally, the Pearson correlation method was again applied to find the correlation coefficient between the daily closing price change and variation in volume. This gave a rather low value, motivating us to analyze these quantities using a neural network. Finally, we studied the behavior of the daily price change when the volume was more or less constant. We found that the resulting distribution in the closing price change follows the general behavior presented by a low white noise.

We then performed an analysis with a single-layer neural network in an attempt to train and obtain a predictive model for the S&P 500's behaviour. However, due to the stochastic nature, non-linearity, and high degree of complexity of the S&P 500 index, a network that could reproduce the plethora of test data with the 0.2% fraction of training data we could use was not possible in the time allotted with the computational resources available. To try to improve the performance of the network, the data set was divided by the binning method in order to avoid overfitting and ensure that the network was trained with data from all relevant regions within the data set. This becomes necessary when data is highly concentrated in one region, as is the case here. By using a logarithm method it was possible to have a well-separated and consequently a more even distribution of points. Such training data allowed us to verify the universal simulation theorem using our neural network, as applied to the S&P 500 index. This theorem suggests that it may be possible to make a predictive model and a network that can predict the testing data. However, due to the small quantity of samples that we could effectively employ for training, to successfully predict the testing data we would need to increase the amount of samples to train the network. Consequently there exists the need to create a multi-layer network to make the predictive model. This is because a deep network can describe better non-linear behavior such as that of the index. Also, the resource used in this thesis, both computational and temporal, did not permit us to test with a larger training set, since the computational requirements were too high for the time allotted. However, parallelization of the neural network and the use of high-end graphical cards with thousands of processors could dramatically reduce the time needed to train

the network. It is also important to note that, although the computational requireds to train a neural network can be quite demanding, once trained with the weights and connections known, its application to other data sets is much more straight-forward.

On the one hand, although the neural networks employed in this thesis were single-layer neural networks, for future projects a multi-layer model should be built to analyze the data from other perspectives, and contrast the computational cost and time with the single-layer model proposed and employed in this thesis. Also, finding the equation of the system through the neural network, one would be able to make a predictive model capable of modeling the future market behavior. Moreover, one could find the non-linear correlation coefficients that the studied data present to obtain a better description of the index. On the other hand, other relevant indices or even large companies individually should be considered for study using our approach. Such an attempt could provide predictive models to help shareholders and investors better choose where to invest their portfolios. Finally, it could be a good idea to carry out a more detailed analysis of the market, perhaps including other relevant market data. Such data may not only be numerical but also categorical, so as to have both a better understanding of the market and concomittantly improve the network's training.

Bibliography

[1] Cevallos, J. C.; Villagomez, J. A.; Andryshchenko, I. Convolutional neural network in the recognition of spatial images of sugarcane crops in the Troncal region of the coast of Ecuador. *Procedia Computer Science* **2019**, *150*, 757–763.

[2] Chato, P.; Chipantasi, D. J. M.; Velasco, N.; Rea, S.; Hallo, V.; Constante, P. Image processing and artificial neural network for counting people inside public transport. 2018 IEEE Third Ecuador Technical Chapters Meeting (ETCM). 2018; pp 1–5.

[3] Lyon, K. From Fundamentals to Spectroscopic Applications of Density Functional Theory. **2020**,

[4] Education, I. C. Neural Networks. `https://www.ibm.com/cloud/learn/neural-networks`.

[5] McCulloch, W. S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* **1943**, *5*, 115–133.

[6] Hebb, D. O. The first stage of perception: growth of the assembly. *The Organization of Behavior* **1949**, *4*, 60–78.

[7] Wu, Y.-c.; Feng, J.-w. Development and application of artificial neural network. *Wireless Personal Communications* **2018**, *102*, 1645–1656.

[8] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* **1958**, *65*, 386–408.

[9] Kullback, S.; Rosenblatt, H. M. On the analysis of multiple regression in k categories. *Biometrika* **1957**, *44*, 67–83.

[10] Hui, S.; Zak, S. H. The widrow-hoff algorithm for mcculloch-pitts type neurons. *IEEE transactions on neural networks* **1994**, *5*, 924–929.

[11] Krogh, A. What are artificial neural networks? *Nature biotechnology* **2008**, *26*, 195–197.

[12] Minsky, M. L.; Papert, S. A. Perceptrons: expanded edition. 1988.

[13] Minsky, M.; Papert, S. *Perceptrons: an introduction to computational geometry*; 1969.

[14] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* **1982**, *79*, 2554–2558.

[15] Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning representations by back-propagating errors. *nature* **1986**, *323*, 533–536.

[16] Chua, L. O.; Yang, L. Cellular neural networks: Theory. *IEEE Transactions on circuits and systems* **1988**, *35*, 1257–1272.

[17] Bolaños, J. A. Stock market and economic growth in Ecuador. 2018.

[18] Choong, C.-K.; Baharumshah, A. Z.; Yusop, Z.; Habibullah, M. S. Private capital flows, stock market and economic growth in developed and developing countries: A comparative analysis. *Japan and the World Economy* **2010**, *22*, 107–117.

[19] Mohtadi, H.; Agarwal, S. Stock market development and economic growth: Evidence from developing countries. *On line] Available at: http//www. uwm. edu/mohadi/PA-4-01. pdf* **2001**,

[20] Singh, A. The stock market and economic development: should developing countries encourage stock markets? **1991**,

[21] Kassimatis, K. Financial liberalization and stock market volatility in selected developing countries. *Applied Financial Economics* **2002**, *12*, 389–394.

[22] Prasad, E.; Rogoff, K.; Wei, S.-J.; Kose, M. A. *India's and China's recent experience with reform and growth*; Springer, 2005; pp 201–228.

[23] Grabel, I. Assessing the impact of financial liberalisation on stock market volatility in selected developing countries. *The Journal of Development Studies* **1995**, *31*, 903–917.

[24] Arauz Galarza, A. D. Microestructura del mercado de valores ecuatoriano. M.Sc. thesis, Quito: FLACSO sede Ecuador, 2009.

[25] Boyacioglu, M. A.; Avci, D. An adaptive network-based fuzzy inference system (ANFIS) for the prediction of stock market return: the case of the Istanbul stock exchange. *Expert Systems with Applications* **2010**, *37*, 7908–7912.

[26] Dutta, A.; Bandopadhyay, G.; Sengupta, S. Prediction of stock performance in the Indian stock market using logistic regression. *International Journal of Business and Information* **2012**, *7*, 105.

[27] Ou, P.; Wang, H. Prediction of stock market index movement by ten data mining techniques. *Modern Applied Science* **2009**, *3*, 28–42.

[28] Nabipour, M.; Nayyeri, P.; Jabani, H.; Mosavi, A.; Salwana, E.; *et al.*, Deep learning for stock market prediction. *Entropy* **2020**, *22*, 840.

[29] Guresen, E.; Kayakutlu, G.; Daim, T. U. Using artificial neural network models in stock market index prediction. *Expert Systems with Applications* **2011**, *38*, 10389–10397.

[30] Dase, R.; Pawar, D. Application of Artificial Neural Network for stock market predictions: A review of literature. *International Journal of Machine Intelligence* **2010**, *2*, 14–17.

[31] Vaisla, K. S.; Bhatt, A. K. An analysis of the performance of artificial neural network technique for stock market forecasting. *International Journal on Computer Science and Engineering* **2010**, *2*, 2104–2109.

[32] McCarthy, J. What is artificial intelligence. *URL: http://www-formal. stanford. edu/jmc/whatisai. html* **2004**,

[33] Chollet, F. *Deep learning with Python*; Simon and Schuster, 2021.

[34] Shinde, P. P.; Shah, S. A review of machine learning and deep learning applications. 2018 Fourth international conference on computing communication control and automation (ICCUBEA). 2018; pp 1–6.

[35] LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *nature* **2015**, *521*, 436–444.

[36] Oliver, N.; Horvitz, E.; Garg, A. Layered representations for human activity recognition. Proceedings. Fourth IEEE International Conference on Multimodal Interfaces. 2002; pp 3–8.

[37] Lee, O.-J.; Jung, J. J.; Kim, J.-T. Learning hierarchical representations of stories by using multi-layered structures in narrative multimedia. *Sensors* **2020**, *20*, 1978.

[38] Dong, J.; Hu, S. The progress and prospects of neural network research. *Information and Control* **1997**, *26*, 360–368.

[39] Jenkins, B.; Tanguay, A. Handbook of neural computing and neural networks. 1995.

[40] Bulsari, A. Some analytical solutions to the general approximation problem for feedforward neural networks. *Neural networks* **1993**, *6*, 991–996.

[41] Luo, Z.; Xie, Y.; Zhu, C. A study of the convergence of the CMAC learning process. *Acta Automatica Sinica* **1997**, *23*, 455–461.

[42] Balcázar, J. L.; Gavalda, R.; Siegelmann, H. T. Computational power of neural networks: A characterization in terms of Kolmogorov complexity. *IEEE Transactions on Information Theory* **1997**, *43*, 1175–1183.

[43] Setiono, R.; Leow, W. K. FERNN: An algorithm for fast extraction of rules from neural networks. *Applied Intelligence* **2000**, *12*, 15–25.

[44] Wu, Y.; Wang, S. A new algorithm to improve the learning performance of neural network through result-feedback. *Journal of Computer Research and Development (in Chinese)* **2004**, *41*, 488–492.

[45] Fiesler, E. Neural network classification and formalization. *Computer Standards & Interfaces* **1994**, *16*, 231–239.

[46] Haykin, S.; Network, N. A comprehensive foundation. *Neural networks* **2004**, *2*, 41.

[47] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **1998**, *86*, 2278–2324.

[48] Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; *et al.*, Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* **2016**,

[49] Bishop, C. M.; *et al.*, *Neural networks for pattern recognition*; Oxford university press, 1995.

[50] Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press, 2016; `http://www.deeplearningbook.org`.

[51] Qian, N. On the momentum term in gradient descent learning algorithms. *Neural networks* **1999**, *12*, 145–151.

[52] Bishop, C. M.; Nasrabadi, N. M. *Pattern recognition and machine learning*; Springer, 2006; Vol. 4.

[53] Brockherde, F.; Vogt, L.; Li, L.; Tuckerman, M. E.; Burke, K.; Müller, K.-R. Bypassing the Kohn-Sham equations with machine learning. *Nature communications* **2017**, *8*, 1–10.

[54] Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**,

[55] Wilson, A. C.; Roelofs, R.; Stern, M.; Srebro, N.; Recht, B. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems* **2017**, *30*.

[56] Mancini, R.; Carter, B. Op amps for everyone. texas instruments. **2009**,

[57] Stein, M. L. *Interpolation of spatial data: some theory for kriging*; Springer Science & Business Media, 1999.

[58] Jindal, I.; Nokleby, M.; Chen, X. Learning deep networks from noisy labels with dropout regularization. 2016 IEEE 16th International Conference on Data Mining (ICDM). 2016; pp 967–972.

[59] Wang, J.-C. Investigating market value and intellectual capital for S&P 500. *Journal of intellectual capital* **2008**,

[60] Clements, A.; *et al.*, S&P 500 implied volatility and monetary policy announcements. *Finance Research Letters* **2007**, *4*, 227–232.

[61] Zhou, W.-X.; Sornette, D. Is there a real-estate bubble in the US? *Physica A: Statistical Mechanics and its Applications* **2006**, *361*, 297–308.

[62] Anderson, R. C.; Reeb, D. M. Board composition: Balancing family influence in S&P 500 firms. *Administrative science quarterly* **2004**, *49*, 209–237.

[63] Yahoo Finance. `https://finance.yahoo.com`, 2022.

[64] ESGT Gitlab. `https://gitlab.com/erick.guagua/thesis.git`, 2022.