# UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

## Escuela de Ciencias Matemáticas y Computacionales

## TÍTULO: Sales Forecast by using Convolutional Neural Networks

Trabajo de integración curricular presentado como requisito para la obtención del título de
Ingeniero en Tecnologías de la Información

**Autor:**

Velasteguí Sandoval Ronny Xavier

**Tutor:**

Ph.D Chang Tortolero Oscar Guillermo

Urcuquí, febrero 2020

UNIVERSIDAD
YACHAY
TECH

Urcuquí, 11 de febrero de 2020

## SECRETARÍA GENERAL
## (Vicerrectorado Académico/Cancillería)
## ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
## CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
## ACTA DE DEFENSA No. UITEY-ITE-2020-00001-AD

En la ciudad de San Miguel de Urcuquí, Provincia de Imbabura, a los 11 días del mes de febrero de 2020, a las 12:00 horas, en el Aula CHA-01 de la Universidad de Investigación de Tecnología Experimental Yachay y ante el Tribunal Calificador, integrado por los docentes:

| | |
|---|---|
| **Presidente Tribunal de Defensa** | Mgs. FONSECA DELGADO, RIGOBERTO SALOMON |
| **Miembro No Tutor** | Dr. PELUFFO ORDONEZ, DIEGO HERMAN , Ph.D. |
| **Tutor** | Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D. |

Se presenta el(la) señor(ita) estudiante **VELASTEGUI SANDOVAL, RONNY XAVIER**, con cédula de identidad No. **0951683291**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, con el objeto de rendir la sustentación de su trabajo de titulación denominado: **SALES FORECAST BY USING CONVOLUTIONAL NEURAL NETWORKS**, previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN.**

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

| | |
|---|---|
| **Tutor** | Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D. |

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

| Tipo | Docente | Calificación |
|---|---|---|
| Miembro Tribunal De Defensa | Dr. PELUFFO ORDONEZ, DIEGO HERMAN , Ph.D. | 10,0 |
| Tutor | Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D. | 10,0 |
| Presidente Tribunal De Defensa | Mgs. FONSECA DELGADO, RIGOBERTO SALOMON | 10,0 |

Lo que da un promedio de: **10 (Diez punto Cero)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

VELASTEGUI SANDOVAL, RONNY XAVIER
**Estudiante**

Mgs. FONSECA DELGADO, RIGOBERTO SALOMON
**Presidente Tribunal de Defensa**

Oscar Chang

Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
**Tutor**

Dr. PELUFFO ORDONEZ, DIEGO HERMAN , Ph.D.
**Miembro No Tutor**

MEDINA BRITO, DAYSY MARGARITA
**Secretario Ad-hoc**

# Autoría

Yo, **RONNY XAVIER VELASTEGUI SANDOVAL**, con cédula de identidad 0951683291, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así como, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor(a) del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, December 2019.

Ronny Xavier Velastegui Sandoval
CI: 0951683291

# Autorización de publicación

Yo, **RONNY XAVIER VELASTEGUI SANDOVAL**, con cédula de identidad 0951683291, 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, December 2019.

Ronny Xavier Velastegui Sandoval
CI: 0951683291

# Acknowledgements

I want to express my deepest thanks to my advisor PhD Oscar Chang, who let me learn all his knowledge as my professor and tutor in several subjects. So, I had the great opportunity to take his Artificial Intelligence course. During this course, I became very interested in neural networks, specifically convolutional neural networks. So, his experience and knowledge helped me to successfully complete my final degree project this year. Also, I want to thank all my professors and tutors who, during these last five years of studies at Yachay Tech University, have transmitted me important knowledge to understand all aspects of my professional career. Thanks to my classmates with whom I enjoyed during these five years of my university career. Finally, it is time to thank the most important people in all my life, my family. They supported me during my time at Yachay Tech.

# Resumen

Todas las empresas necesitan un método efectivo para predecir las ventas futuras y existen varios métodos clásicos, puramente estadísticos, que se utilizan ampliamente en la industria. Sin embargo, estos métodos de predicción no siempre son efectivos cuando se enfrentan a predicciones sin una tendencia clara en datos pasados. Recientemente, han aparecido técnicas más sofisticadas con capacidades de predicción mejoradas, basadas en aprendizaje automático y redes neuronales. Por ejemplo, las redes neuronales Feedforward y las redes neuronales recurrentes han demostrado una buena precisión al trabajar en la predicción de series temporales. Pero, ambos métodos fallan cuando se utilizan muchas capas ocultas, esto se debe al conocido problema de desaparición de gradiente.

Este trabajo propone un nuevo método de predicción de ventas basado en redes neuronales convolucionales. Este tipo de red neuronal se usa generalmente para tareas de procesamiento de imágenes. Pero en este trabajo, exploramos nuevas aplicaciones y desarrollamos modelos que producen buenos resultados en la predicción de ventas para datos reales de productos farmacéuticos. Los datos utilizados pertenecen a una base de datos de una franquicia de farmacias ecuatoriana. Esta base de datos contiene ventas semanales de productos durante un período de 4 años. Con este conjunto de datos, se implementaron y probaron varios métodos de predicción clásicos y basados en inteligencia artificial. Luego, nuestro método propuesto basado en redes neuronales convolucionales fue diseñado y programado en Matlab. Después de esto, todos estos métodos de predicción se compararon utilizando tres métricas: precisión de predicción, número de pesos y número de iteraciones. Finalmente, procedimos a determinar cual método de predicción es mejor tanto en precisión y eficiencia como en sus ventajas y desventajas

**Keywords**: Predicción de ventas, Redes neuronales artificiales, Redes neuronales convolucionales, Aprendizaje profundo.

# Abstract

All companies need an effective method to predict future sales and several classic, purely statistical methods exist and are heavily utilized in the industry. However, these prediction methods are not always effective when faced with predictions with no clear trend in past data. Recently more sophisticated techniques with improved prediction capacities, based on Machine Learning and Neural Networks, have appeared. For instance, Feedforward Neural Networks and Recurrent Neural Networks have demonstrated good precision when working on time series prediction. But, both methods fail when many hidden layers are utilized, this is due to the well-known gradient vanishing problem.

This work proposes a novel sales prediction method based on Convolutional Neural Networks. This type of neural network is generally used for image processing tasks. But in this work, we explore new applications and develop models that produce good results in sales prediction for real pharmaceutical product data. The used data belongs to an Ecuadorian pharmacy franchise database. It contains weekly sales of products over a period of 4 years. With this data set, several classical prediction methods and artificial intelligence prediction methods were implemented and tested. Then, our proposed method based on convolutional neural networks was designed and programmed in Matlab. After this, all these prediction methods were compared using three metrics: prediction accuracy, number of weights and number of iterations. Finally, we proceeded to determine which prediction method is better both in accuracy and efficiency as well as their advantages and disadvantages.

**Keywords**: Sales Forecast, Artificial Neural Networks, Convolutional Neural Networks, Deep Learning.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Sales forecasts are a fundamental part of any company, since, these companies can plan and make important decisions that determine the future success or failure of the company [1]. This is because a good prediction will allow us to correctly decide important actions such as purchase of raw material, increase of personnel, preparation of advertising, acquisition of equipment, among others. While there are several methods to predict sales, used by companies today, it is not always possible to obtain a completely accurate result, however, a good approximation can make a difference in the future of the company. The methods of sales prediction are classified mainly into two large groups, qualitative methods, and quantitative methods [2]. The difference between both methods lies in the presence of past data. On the one hand, qualitative methods are used to predict sales when there is no previous data. On the other hand, quantitative methods are used when a certain amount of past data is available and we assume that there is a possible trend in them. Depending on the situation, each method may work better than the other.

Quantitative methods are subdivided into classical statistical methods and methods based on artificial intelligence. Some examples of classical statistical methods are the Naive approach, the Average approach, and the Moving average approach. The Naive approach is the simplest prediction method of all, but not necessarily the most accurate. The other two mentioned methods use averages to do the predictions, but, like the naive method, both accuracies obtained are very low. The advantage of these methods is that they are very easy to understand and implement, that is why they are still used in some companies [3].

While it is true that classical statistical methods such as the Naive approach, Average approach, Moving average approach, among others, have been widely used by companies for years. Currently, companies are opting to use more modern methods based on artificial intelligence [4]. One of these methods is artificial neural networks. Thus, the Feedforward Neural Networks and Recurrent Neural Networks have demonstrated good precision when facing time series prediction problems. But, it is important to mention that, when trying to make very complex predictions, where the data does not show a very clear trend, these methods are not very effective. This is because to predict complex time series, you usually need to use a large number of hidden layers and neurons, this confronts the well-known Vanishing Gradient problem, which affects deep networks [5].

It is for this reason that, in this work, it is proposed to develop a novel sales prediction method based on Convolutional Neural Networks. While this type of neural network is generally used for image processing tasks [6], in this work we develop new applications in sales prediction. For this, we will first choose a sales database corresponding to an Ecuadorian Pharmacy Franchise. This database contains weekly sales of products for 4 years. Once the data set is defined, several classical prediction methods and also prediction methods based on artificial intelligence were implemented. Then, our novel method based on convolutional neural networks was proposed and implemented. After this, all these prediction methods were compared using three metrics: prediction accuracy, number of weights and number of iterations. Thus, we proceeded to determine which prediction method is better both in terms of accuracy and efficiency, also, the advantages and disadvantages of each prediction method were determined.

# Chapter 2

# Preliminaries

## 2.1 Problem statement

All companies need an effective method to predict future sales. While there are several classic methods to predict sales in general, most of them are purely statistical methods. These methods are not always effective when faced with predictions where there is no clear trend in past data.

Also, there are currently more sophisticated techniques that can be used to make these types of predictions, these techniques are based on Machine Learning and Neural Networks. So, Feedforward and Recurrent Neural Networks have demonstrated good precision doing time series prediction. But, the disadvantage of both methods is that they suffer the gradient vanishing problem by using too many hidden layers due to the large number of weights.

For this reason, in this work, we develop a novel sales prediction method based on Convolutional Neural Networks (CNN). While this type of neural network is generally used for image processing tasks, in this work we will explore its possible effective application in sales prediction.

## 2.2 Document organization

The structure of this work is organized in 8 main Chapters. These chapters are Introduction, Preliminaries, Theoretical Framework, Methodology, Result, Discussion, Conclusions and Future works.

In Chapter 1, a brief introduction to this work is presented.

In Chapter 2, the problem statement and document organization are presented.

In Chapter 3, the general objectives and specific objectives are presented.

In Chapter 4, a brief background about time series, time series prediction, and artificial neural networks are presented.

In Chapter 6, the prediction models and comparison metrics used in this work are described.

In Chapter 7, the prediction results of all proposed models are presented using tables and graphs.

In Chapter 8, the results obtained in the previous chapter are analyzed and the advantages and disadvantages of each prediction model are discussed.

In Chapter 9, the final conclusions of this work are presented. Also, this chapter analyzes whether the general and specific objectives were successfully met.

In Chapter 10, the possible modifications and improvements to the proposed models are defined.

# Chapter 3

# Objectives

## 3.1 General Objective

The general objective of this project is to propose a novel sales prediction method based on artificial intelligence, through the use of convolutional neural networks.

## 3.2 Specific Objectives

The next specific objectives will be followed to complete the general objective.

- To obtain and pre-process sales data to do the predictions
- To implement classical prediction models actually used.
- To propose and implement the novel CNN based prediction model.
- To compare the proposed model with the other implemented models.

# Chapter 4

# Theoretical Framework

## 4.1 Time Series

Time series is a sequence of data collected through time [7]. Time series are very important in many fields of science [8]. Some examples of time series are daily temperature of a city, the daily number of dead in a specific country, monthly number of birth in a city, the daily measure of stock market prices, number of daily crimes in Ecuador during last year, monthly concentration of CO2 in the atmosphere during last years, etc.

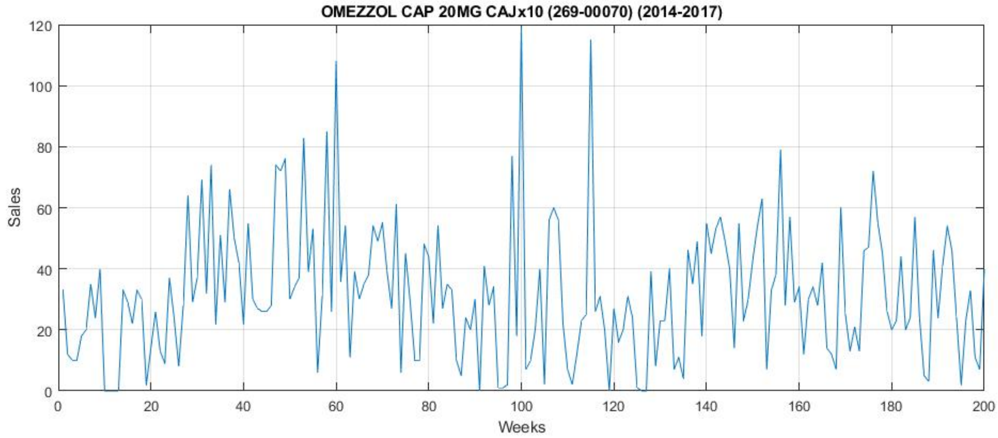


Figure 4.1: Time series representing weekly sales of a product during four years

## 4.2 Classification of Time Series

Following, we will see a simple classification of time series mainly based on mean and variance [2].

### 4.2.1 Stationary Time Series

A time series is called stationary if its mean and variance remain constant over time [1].

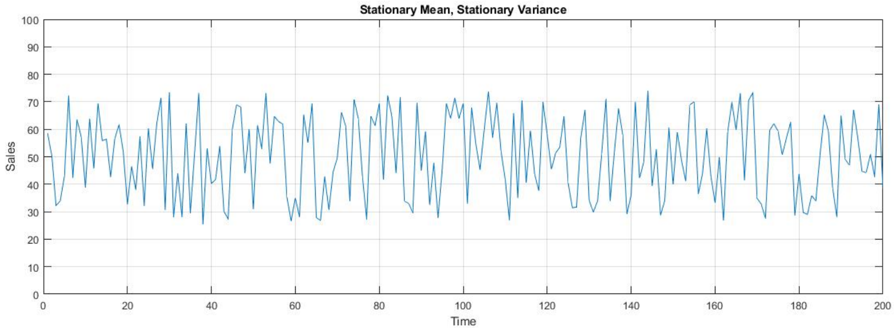


Figure 4.2: Stationary Time Series

### 4.2.2 Non-stationary Time Series

A time series is called nonstationary if its variance and/or mean change through time [9].

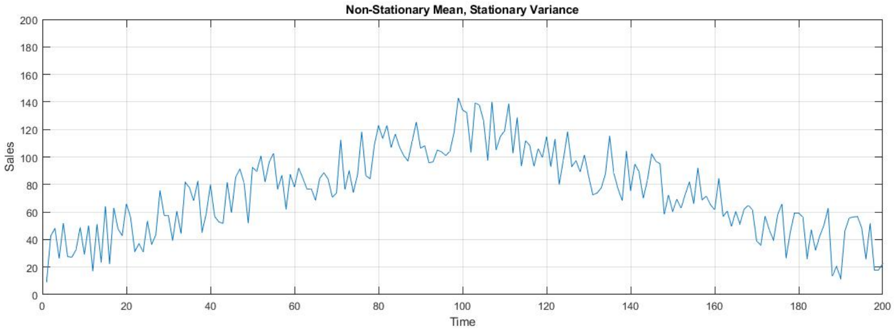


Figure 4.3: Time Series with Non-stationary mean and Stationary variance

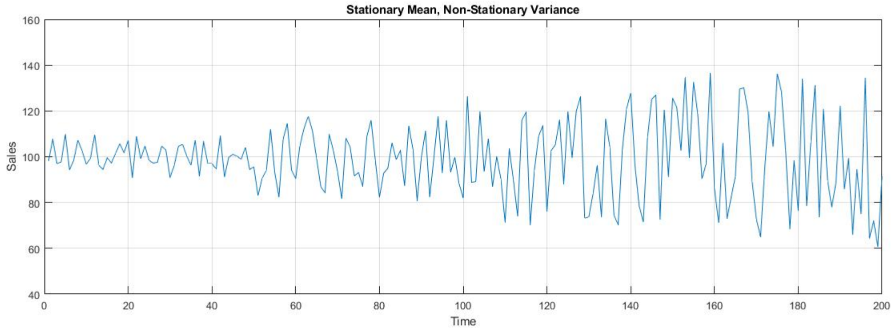


Figure 4.4: Time Series with Stationary mean and Non-stationary variance

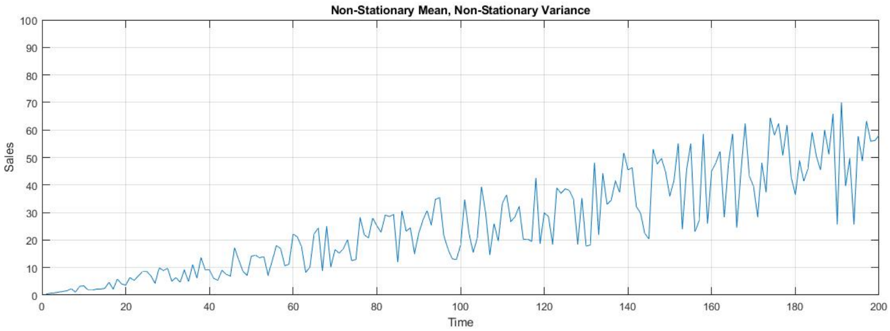


Figure 4.5: Non-stationary Time Series

## 4.3 Time Series Prediction

The understanding of the past is the key to predict future [9]. So, time series prediction or time series forecasting is the process of analyzing data of a time series, and propose a model to predict the future of that time series [10].

### 4.3.1 Applications of Time Series Prediction

Time series forecasting has many important applications. Some of them are the following:

In economy, a very important use of time series forecasting is to predict the stock market. This is because the stock market changes very often, and usually, these changes involve millions of dollars. For this reason, investors want methods to predict the trend of the stock market and in this way try to obtain the greatest profit, and also avoid possible losses [3].

In demography, time series forecasting can be used to predict migratory trends of the population. So, this information can be very useful to governments to take actions in advance, because migration has a direct impact on the general development of a country [11].

In the study of the environment, an important use of time series forecasting is to predict the concentration of CO2 in the atmosphere. This type of prediction is very important today because are directly related to climate changes, specifically with global warming. [12]

### 4.3.2 A specific case study: Pharmacy Sales Prediction

Farmaenlace is an Ecuadorian pharmacy franchise composed of three important pharmacies: Farmacias Economicas, Farmacias Natural, and Farmacias Medicity. It is one of the most successful franchises in Ecuador and its success is partly due to its business strategy. Thus, one of the business strategies of any company is the use of different methods of predicting time series. So, in the case of Farmaenlace, a crucial factor could be the prediction of sales/inventory of different medicines and products. This type of prediction is very important because allows us to prevent products from expiring, and also it is an indicator to guide advertising strategies. So, for this project, we will use a database of real sales of Farmaenlace products and medicines.

## 4.4 Time Series Prediction Methods

There are many methods to forecast time series. And generally, they can be classified into two large groups: Quantitative Methods and Qualitative Methods [4].

### 4.4.1 Quantitative Methods

Quantitative Methods are based on statistical and mathematical analysis. These methods work fine when we have enough and ordered data in the time series to analyze. Next, we will detail some of these methods.

#### 4.4.1.1 Naïve Approach

The Naïve Approach is the simplest method to forecast time series. This method consists on predict the next future value of the time series using only the last value observed [8]. So, this method uses the following equation:

$$\hat{y}_{t+1} = y_t \tag{4.1}$$

$y_t$: Time series value at time t
$\hat{y}_{x+1}$: Predicted time series value at time t+1

As we can see in the equation, this method estimates that the value to be predicted is equal to the last value observed. As we know, time series rarely follow that simple pattern [3], for this reason, this method is not so accurate when making predictions.

#### 4.4.1.2 Simple Average Approach

This method uses all past data of the time series to do the prediction. So, this method uses the following equation:

$$\hat{y}_{x+1} = \frac{1}{x}\sum_{i=1}^{x} y_i \tag{4.2}$$

$y_i$: Time series value at time i
$x$: Number of total time steps
$\hat{y}_{x+1}$: Predicted time series value at time x+1

As we can see in the equation, this method estimates the future value using the total average of all past data of time series [3].
Due that this method uses more information to make the prediction, it may seem that this method is always better than the previous one, but this is not always true. This method works best when the time series has a constant average, otherwise, its accuracy may be worse than the Naive Method.

#### 4.4.1.3 Moving Average Approach

This method is an improvement to the Simple Average Approach. So this method does not necessarily use all the observations of the time series, instead, it only uses a subset of it [1]. This method uses the following equation:

$$\hat{y}_t = \frac{1}{p}(y_{t-1} + y_{t-2} + y_{t-3}... + y_{t-p}) \tag{4.3}$$

$\hat{y}_t$: Predicted time series value at time t
$y_{t-1}$: Time series value at time t-1
$p$: Window size of time steps

As we can see in the equation, this method estimates the future value using the average of the last p observations of time series. Note that p is a fixed parameter that we must adjust.

#### 4.4.1.4 Simple Exponential Smoothing

As we saw in the previous methods, when we make time series predictions, the use of the latest time series data leads to better prediction accuracy. Instead, the use of very old time series data does not contribute significantly to the improvement of accuracy [4].
Based on this idea, the Simple Exponential Smoothing method uses all observations of the time series to make a forecast, but the contribution of each observation decreases exponentially as we move into the past.

$$\hat{y}_{t+1} = \alpha * y_t + (1 - \alpha) * \hat{y}_t \tag{4.4}$$

$y_t$: Time series value at time t
$\hat{y}_{x+1}$: Predicted time series value at time t+1
$\alpha$: Smoothing factor, $0 < \alpha < 1$

### 4.4.2 Qualitative Methods

Qualitative Methods are not based on statistical nor mathematical analysis. These methods are based only on personal experience, intuitions, opinions, etc. These methods are usually used when we do not have available previous data [12]. In short, they are subjective methods and therefore its accuracy is too low to be useful today in companies. So. due to the total subjectivity of these methods, we will not delve into them in this work.

## 4.5 Artificial Neural Networks

Artificial Neural Networks (ANN) are a powerful and parallel computational model created in (1943) by Warren McCulloch and Walter Pitts [13]. This model was inspired by the functioning of the human brain. The advantage of this model over other computational models is that it can learn to solve problems by itself [6]. That is, in most cases, these models do not need to be programmed with specific-rules to solve problems. Instead, these models can find the solution by themselves after training [14].

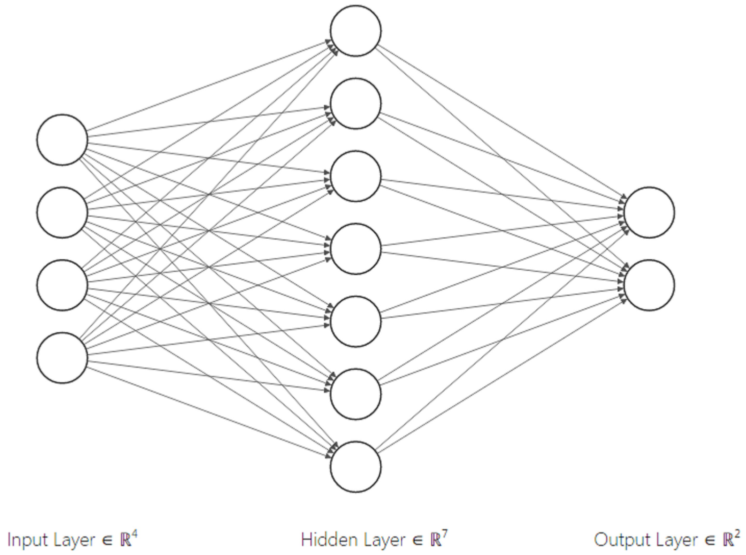

Input Layer ∈ $\mathbb{R}^4$
Hidden Layer ∈ $\mathbb{R}^7$
Output Layer ∈ $\mathbb{R}^2$

Figure 4.6: Artificial Neural Network

### 4.5.1 Basic components of ANN

#### 4.5.1.1 Neuron

In ANN, a neuron is the minimal unit of processing. This is responsible for receiving a set of input values, processing them through a set of arithmetic operations, applying an activation function, and finally generating a resulting value [5].

#### 4.5.1.2 Neurons Layer

A neural network is formed by layers of neurons. A neuron layer is a set of neurons at the same process level. Generally speaking, there are 3 basic types of neuron layers: Input Layer, Hidden Layer, Output Layer [15]. Input Layer is the first layer of ANN and is responsible for entering the input data to the ANN. Hidden Layers are the heart of the entire network. They are responsible for taking the data of the input layer and process it. There may be one or more hidden layers, and the information is processed through each layer until finally reaching the output layer. Output Layer is the last layer of ANN and is responsible for giving the final result [16].

#### 4.5.1.3 Connections and Weights

In ANN, the connections between neurons are represented by values called weights. These weights represent the strength of the synapse between two neurons [17]. Also, the values of weights are not fixed, they constantly change during the learning process. After a network performs the learning process, the set of weight values of the network represents all the knowledge learned [18].

#### 4.5.1.4 Synaptic Potential

Each neuron has a synaptic potential associated with it. This synaptic potential is a value obtained by doing arithmetic operation between the inputs and weights associated with the neuron [19]. After, this value will be passed to a special function called Activation Function.

#### 4.5.1.5 Activation Function.

All neurons have a non-linear function at the output. This function is called Activation Function. The objective of this function is to transform the synaptic potential to a narrow range normally between [-1, 1] or [0, 1] [20]. The most used activation functions are Sigmoid, Hyperbolic Tangent, RELU, and L-RELU [13].

#### 4.5.1.6 Learning Algorithm

As we already know, the ANN power lies in the capacity to learn to solve problems itself. This is reached by the use of a learning algorithm. There are many types of learning algorithms, but the common objective of all of them is to adjust the weights of ANN. This adjustment is made to try to progressively reduce the error observed during training [14].

#### 4.5.1.7 Learning Rate

The Learning Rate is a hyper-parameter associated with Learning Algorithm. It is usually a number between $[0 - 1]$ and determines the "speed" of the ANN learning process. So, during the learning process, if this hyper-parameter is very high (close to 1), the weights will change very fast in each learning epoch. However, if this hyper-parameter is very low (close to 0), the weights will change very slowly [15].

## 4.6 Classification of Artificial Neural Networks

### 4.6.1 Classification of ANN According Topology

This classification is based according to the organization of neurons and connections in the ANN structure.

#### 4.6.1.1 Mono-Layer ANN

Mono-Layer ANN has only one layer of neurons, where neurons are connected with each other. One of the best known ANN of this type is the Hopfield Network [21].

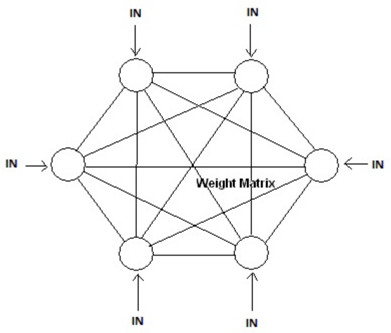


Figure 4.7: Hopfield Neural Network

#### 4.6.1.2 Multi-Layer ANN

In this category, we have ANN architectures with more than 1 layer. In this type of network, usually, the information flows from the input layer to the output layer. Also, depending on the direction of the connections, this category is divided into two subcategories [6].

#### 4.6.1.2.1 Networks only with forward connections

In this type of ANN, the information only can flow from the input layer to the output layer without any recurrence or cycle of connections. One of the best known ANN of this type is the Multi-Layer Perceptron [5].

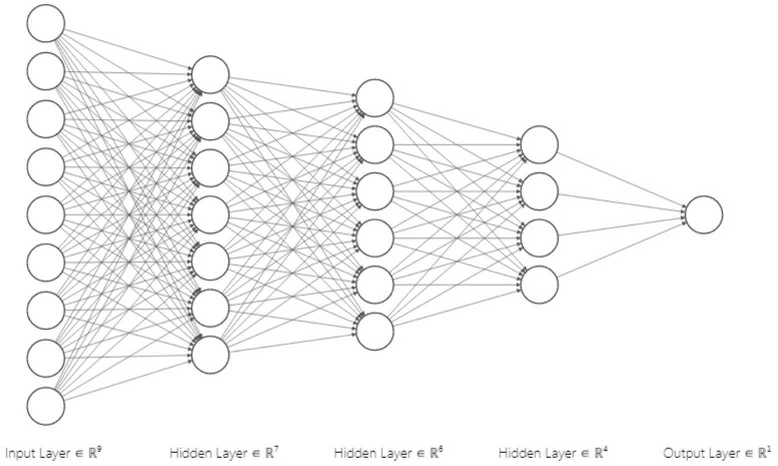

Input Layer ∈ $\mathbb{R}^9$
Hidden Layer ∈ $\mathbb{R}^7$
Hidden Layer ∈ $\mathbb{R}^6$
Hidden Layer ∈ $\mathbb{R}^4$
Output Layer ∈ $\mathbb{R}^1$

Figure 4.8: Multi-Layer Perceptron

#### 4.6.1.2.2 Networks with backward and forward connections

In this type of ANN, the information can flow both forward and backward directions. The backward connections can help to improve the performance because they allow to feedback the internal structure of the network during the learning process [17]. Two of the best known ANN of this type are the Elman Neural Network and Jordan Neural Network.

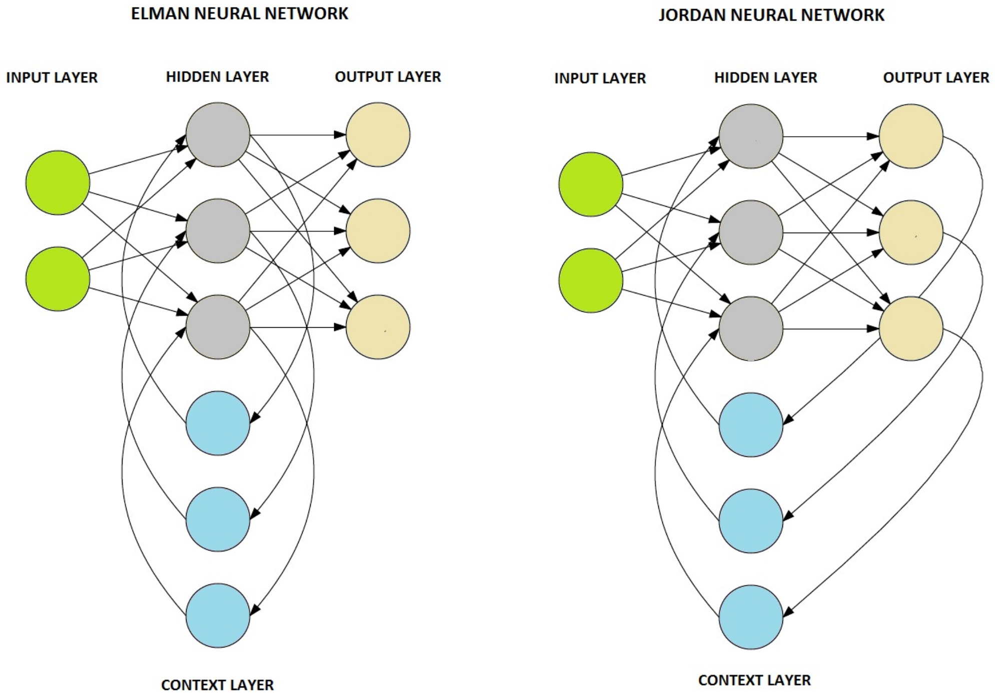


Figure 4.9: Elman Neural Network and Jordan Neural Network

### 4.6.2 Classification of ANN According Learning Paradigm

As we already know, all ANN use some learning algorithm to adjust their weights. So, this algorithm can be classified into 2 principal groups.

#### 4.6.2.1 ANN with Supervised Learning

In this category the training data that ANN use is formed by labeled patterns. Labeled patterns are observations in which we have the target value. Thus, during training, the network processes these patterns and tries to approximate the original target value [19]. In this way, the learning algorithm will adjust weight to get the minimal error between original target values and their approximation.

#### 4.6.2.2 ANN with Non-Supervised Learning

In this case, the ANN does not need labeled patterns. The ANN only analyzes input patterns and tries to group them according to their similarities. It is a very powerful technique for clustering information. One of the best known ANN of this type is the Self-organized maps, also called Kohonen Maps [14].

## 4.7 Gradient Descent

Gradient Descent is one of the most important optimization algorithm used to train artificial neural networks [22]. The idea of this algorithm is to reduce an objective function $J(\Theta)$, also known as cost function. In the context of ANN, the cost function usually represents the error between target values and estimate values. This cost function depends on a set of parameters $\Theta \in R^d$, in the context of ANN, this set of parameters consists of the weights and biases [23].

So, the operation of this algorithm, as the name itself says, is to update the parameters in the opposite direction of the cost function gradient $\nabla_\Theta J(\Theta)$. In this way, the parameters $\Theta$ will be updated iteratively until reaching a local minimum of the cost function. Also, the size of the step parameter changes in each iteration is given by a hyperparameter known as the learning rate $\eta$ [24].

Gradient Descent method is divided in three main variants: Batch Gradient Descent, Mini-batch Gradient Descent, and Stochastic Gradient Descent. The difference between them is the amount of input data used to calculate the gradient of the cost function in each iteration. So, depending on the amount of input data we can find a balance between the accuracy of the parameters update and the computational cost used to perform said update [25].

### 4.7.1 Batch Gradient Descent

Batch Gradient Descent uses the complete input data set to calculate the cost function. So, due that we need calculate the gradient using all input data in each iteration, the computational cost will be very high to perform only one parameter update [26]. For this reason, in this method the accuracy of each parameter update is very high, but also the computational cost used to perform said update is also high. Also, Batch Gradient Descent guarantees us to reach a local minimum of the cost function.

$$\Theta = \Theta - \eta * \nabla_{\Theta} J(\Theta) \tag{4.5}$$

### 4.7.2 Stochastic Gradient Descent

Stochastic Gradient Descent uses only one element of the input data set to calculate the cost function. So, due that we need to calculate the gradient using only one element of input data set in each iteration, the computational cost will be very low to perform the parameter update [27]. But, the disadvantage of this method is the low accuracy of each parameter update, which means that this method will take a lot of iterations to reach the local minimum. Due to the fluctuation of this method during the update of parameters, in some cases is possible to reach a good local minimum or also a global minimum of the cost function. But, a disadvantage of these fluctuations is the difficulty to reach the exact minimum [28].

$$\Theta = \Theta - \eta * \nabla_{\Theta} J(\Theta; x^{(i)}; y^{(i)}) \tag{4.6}$$

Where $x^{(i)}y^{(i)}$ are the i training sample and label respectively

### 4.7.3 Mini-batch Gradient Descent

Mini-batch Gradient Descent combines the advantages of the two previous methods. So, this method uses an input data sub-set of size n to calculate the gradient of the cost function [29]. In this way, by changing the hyper-parameter n, we could get the equilibrium between the accuracy of parameter update and the computational cost to do said update.

$$\Theta = \Theta - \eta * \nabla_{\Theta} J(\Theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{4.7}$$

Where $x^{(i:i+n)}y^{(i:i+n)}$ are a subset of training samples and labels respectively, so, the size of this subset is n+1.

### 4.7.4 Momentum

Momentum is an optimization for Stochastic gradient descent and Mini-batch gradient descent. Thus, momentum allows to reduce the fluctuations and oscillations that affect the two mentioned methods. In this way, the convergence to minimum is reached in less iterations [30]. So, the idea is basically to add a parameter $\gamma$ called momentum, which can have a value between 0 and 1. A common value for momentum $\gamma$ is 0.9 [31].

$$v_t = \gamma * v_{t-1} - \eta * \nabla_{\Theta} J(\Theta) \tag{4.8}$$

$$\Theta = \Theta - v_t$$

## 4.8 Backpropagation Algorithm

Backpropagation (BP) is one of the most popular algorithms used in the training of neural networks [32]. BP was first implemented in 1970 by Seppo Linnainmaa and allowed to greatly improve the performance of neural networks [33]. So, Backpropagation is an iterative and recursive method to update weights of ANN. BP usually works together Gradient Descent Algorithm to adjust weights with the objective of minimize the loss function. Also, it is important to mention that the BP algorithm requires that all activation functions be derivable [34].

In terms of operation, this algorithm is composed of two principal phases: Forward Phase and Backward Phase. In the first one, the outputs of each of the layers are calculated sequentially until they reach the output layer. At the output layer, the error between the target value and the estimated value is calculated. Then, the Backward Phase starts. In this phase, the partial derivatives with respect to the error of each neuron will be calculated [35]. Finally, partial derivatives and the learning rate will be used to update the weights recursively from the output layer to the input layer.

$$LOSS = \frac{1}{2}\sqrt{\frac{\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}{m}} \tag{4.9}$$

$y_i$: Observed value i
m: MiniBatch size
$\hat{y}_i$: Predicted value i

## 4.9 Deep Neural Networks

Deep Neural Networks (DNN) are Artificial Neural Networks with many hidden layers [36]. Unlike Shallow Networks, DNN allows us to process complex data more effectively. This is because each layer decomposes and extracts features of the input data as it travels the network structure [37]. In this way, while the input data goes deeper and deeper into the hidden layers, the network extracts increasingly complex features. This allows us to solve very complex problems for Shallow Networks, for example, Image Recognition [38].

### 4.9.1 CNN Artificial Neural Network

Convolutional Neural Network (CNN) is another famous type of Deep Neural Network. These networks are very used to solve image recognition problems [39]. CNN is very similar to Multi-Layer Perceptron, but the difference is the connections. While in the Multilayer Perceptron each layer is fully connected to the previous layer, in CNN this not happen. So, CNN neurons are based in receptive fields of human vision. This means that each layer and neuron is specialized in extract specific features of images [40]. Thus, by sequentially grouping several hidden convolutional layers, it will be possible to extract increasingly complex characteristics. Also, another advantage of CNN is that these convolutional layers are not fully connected with the previous layer, this represents a great diminution of weights to adjust, and therefore, an improvement in efficiency [41].

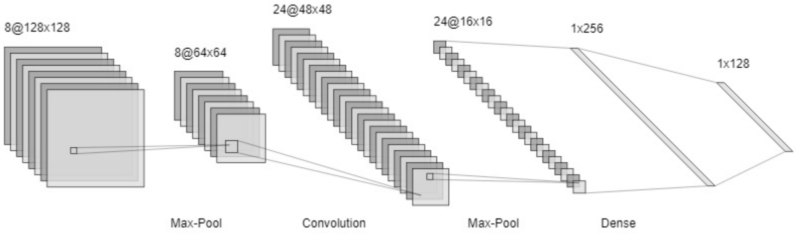
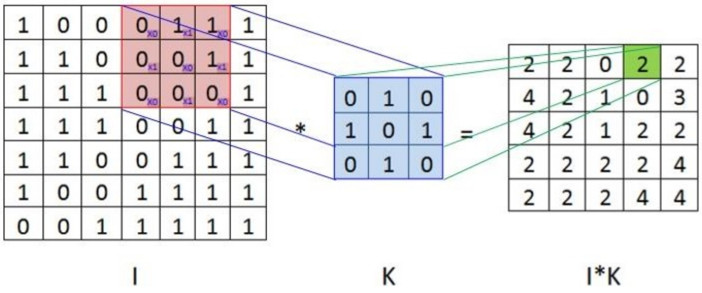


Figure 4.10: Convolutional Neural Network

#### 4.9.1.1 Convolutional Layers

Convolutional Layers are responsible to extract features of input data, such data are usually 2D images [42]. CNN layers are composed of convolutional neurons. These convolutional neurons represent filters usually in rectangular windows form. Such filters are matrices which contain weights, and these matrices are convolved with the input data matrix. So, CNN can have many Convolutional Layer and the input of each layer corresponds to the output of the previous layer, thus forming a deep neural network [43].



$$\begin{array}{|c|c|c|c|c|c|c|}\hline 1&0&0&0&1&1&1\\\hline 1&1&0&0&0&1&1\\\hline 1&1&1&0&0&0&1\\\hline 1&1&1&0&0&1&1\\\hline 1&1&0&0&1&1&1\\\hline 1&0&0&1&1&1&1\\\hline 0&0&1&1&1&1&1\\\hline\end{array} * \begin{array}{|c|c|c|}\hline 0&1&0\\\hline 1&0&1\\\hline 0&1&0\\\hline\end{array} = \begin{array}{|c|c|c|c|c|}\hline 2&2&0&2&2\\\hline 4&2&1&0&3\\\hline 4&2&1&2&2\\\hline 2&2&2&2&4\\\hline 2&2&2&4&4\\\hline\end{array}$$

Figure 4.11: Convolutional Product, K is a convolutional filter

#### 4.9.1.2 Convolutional Filter Size

Convolutional filter size is a hyperparameter associated with a convolutional neuron that we can choose according to the problem to solve. This size determines how many connections will have the convolutional neuron [44]. If the input data is an image, this parameter determines who many pixels will enter to the convolutional neuron at the same time.

For example, if the input is one channel 2D image of 100x100 size, and we choose a 3x3 square filter. This means that the convolutional neuron will do a convolutional product with only a small section of the image of size 3x3 at every step. This is a great advantage respect to a fully connected neuron. This is because that convolutional neuron only

would have 3x3=9 different weight to adjust, but a fully connected neuron would have 100x100=10000 weight to adjust [45].
Now, if the neuron filter size is 3x3, how will the neuron analyze the entire 100x100 input image? This is achieved since the 3x3 filter travels through the entire image sequentially. The filter will shift to the right and down of the image, making a convolution product at each step. Also, the size of the shift at each step is a hyperparameter called stride.
To analyze one channel image, common filter sizes are 3x3, 5x5, 7x7, 9x9. In the case of RGB images, common filter sizes are: 3x3x3, 5x5x5, 7x7x7, 9x9x9, etc [46].

#### 4.9.1.3 Pooling Layers

Pooling layers are additional layers that are placed after each convolutional layer. The objective of these layers is to reduce the dimension of data. There are two principal types of pooling layers: Max Pooling and Average Pooling. Max Pooling Layer takes the output of the previous convolutional layer (represented by matrices) and applies a Max Pooling Filter (following the same process as convolution filters), which calculate the maximum between the values captured by the filter. Average Pooling Layer is the same, but instead of calculating the maximum value, it calculates the average of the values captured by the filter [47].

#### 4.9.1.4 Fully Connected Layer

In a CNN architecture, after all convolutional and pooling layers, a fully connected layer is located. This final layer takes all the abstractions generated by previous layers and used it to generate the classification or regression final result [43].

# Chapter 5

# Related Works

In this chapter, we will review some important related works about sales forecasting methods. So, seven research published from 1997 to the present year 2019 will be reviewed. The methods described range from the simplest Shallow-MLP, to more advanced and current methods such as LSTM or Autoencoders.

## 5.1 Thesing and Vornberger (1997)

In that year, Thesing and Vornberger propose a method to predict time series using artificial neural networks [48]. So, they start mentioning that statistical prediction methods of time series have two principal problems. The first problem that they mention is that each time series to predict, need a specific statistic prediction model depending on its trend, seasonality and other characteristics. The second problem is that statistical prediction methods are effective in univariate time series, but in multivariate time series are not effective. So, they implement an ANN prediction method and show that is more accurate than the other two statistical methods.

They use a database of a German Supermarket Chain. This database contains information about: weekly sales, advertising campaigns, and price reductions. So, they use a feed-forward multilayer perceptron (MLP) to predict future weekly sale, based on information of n last weeks. In the experimental section, they realize many tests varying some ANN hyperparameter: n size, number of hidden neurons, learning rate, momentum and size of validation data set. After that, they choose the ANN configuration that reaches the best accuracy and compare it with the Naïve Prediction Method and Moving Average Method, which are the two prediction methods currently used by this Supermarket Chain.

After applying these three prediction methods to forecast sales 20 different products, they calculate Mean Square Error (MSE). Thus, the Naive Prediction Method reached an MSE of 1.16, Moving Average Prediction Method reach an MSE of 1.01, and ANN Method reaches an MSE of 0.84. In this way, they conclude that ANN Prediction Method obtains the best accuracy making multivariate sales predictions.

## 5.2 Zhang, Patuwo and Hu (1998)

Zhang, Patwo, Hu show a very detailed summary of researches on time series predictions using neural networks made until 1998 [18]. First, they analyze the advantages that ANN forecasting methods have over statistical forecasting methods. So, they mention three principal advantages: adaptability, non-linearity, and universal approximator. The adaptability advantage means that unlike statistical prediction models, the ANN prediction model does not need many assumptions about the time series to predict. This is because ANN has the ability to find itself implicit relationships hidden in the time series to use. The non-linearity advantage means that unlike linear prediction models, ANN prediction model does not assume that the analyzed time series is generated from linear processes. This is an advantage because most of the real-world time series are generated from non-linear processes, and linear prediction models cannot accurately predict these types of time series. Finally, the universal approximator advantage refers that ANN actually can approximate any continuous function. And, because any time series can be represented as a continuous function, this means that an ANN with enough hidden neurons could approximate it correctly.

So, after defining the advantages of ANN prediction models, they review several of these models and analyze each architecture. The analysis includes the number of input nodes, number of hidden nodes, number of output nodes, size of data set, transfer function, training algorithm, data normalization, and performance measure.

Finally, the authors also mention some limitations that ANN prediction methods can present. The first limitation is that while ANN can find the relationship between inputs and outputs by itself, there is no way to explicitly obtain this relationship function, this is due to the Black-Box nature of ANN. The second limitation is that ANN methods can suffer of overfitting problems, producing a very low generalization capacity. The third limitation is the no-existence of some effective mechanism to find the best configuration of ANN that produces the best accuracy. And the last limitation that they mention is that ANN models need of great amount of data and computing time for the training phase.

## 5.3 Khashei and Bijari (2010)

In that year, Khashei and Bijari propose a novel hybrid method to predict time series [49]. This hybrid method combines a statistical predicted method called Auto-regressive Integrated Moving Average (ARIMA), with an Artificial Neural Network (ANN) prediction method. The principal difference between these two models is that the first one is a linear prediction method, but the second one is a nonlinear prediction method. So, they start to mention the difficult to know the implicit characteristic of time series in real problem, specifically the linear/non-linear behavior. For example, some time series can exhibit both linear and nonlinear behavior under some conditions. Based on that, Khashei and Bijari decided to take advantage of two completely different prediction models: ARIMA and ANN, and in this way, improve accuracy when making prediction

with time series that show both linear and non-linear behavior.
The architecture that they implement is divided into two principal stages. In the first stage, they use the ARIMA method to preprocess input time series data. For this stage, ARIMA is implemented in combination with Box and Jenkins Methodology. In the second stage, they use ANN that takes previous preprocessed data and calculates the final prediction. So, after some experiments, they chose an ANN architecture with: 8 Inputs Neuron, 3 Hidden Neurons, and 1 Output Neuron. Also, they chose the Sigmoid Activation Function. The algorithms used for training was Backpropagation and Stochastic Gradient Descent with Momentum (SGDM)
To measure the accuracy of the model, Khashei and Bijari use three data sets: The Wolf's Sunspot data set, The Canadian Lynx data set, and the British pound/US dollar exchange data set. So, they compare the prediction accuracy of three different models: the ANN prediction model, ARIMA prediction model, and the Hybrid ARIMA/ANN prediction model that they propose. Finally, in the results, they find that the hybrid model exceeds the prediction accuracy of the independent ARIMA and ANN models.

## 5.4 Jie Wang and Jun Wang (2016)

Jie Wang and Jun Wang propose a time series prediction method that uses a Recurrent Neural Network (RNN) [50]. They start to mention that ANN has the ability to deal with non-linear and non-stationary time series. Also, they mention that RNN, which is a specific type of ANN, has also the ability to handle temporary or spatial dependencies. So, they propose to use an Elman Recurrent Neural Network (ERNN), which is a very simple recurrent neural network architecture, to predict time series.
The ERNN architecture that they use is very similar to a 3-Layer Multilayer Perceptron. Both have one input layer, one output layer, and one hidden layer. But the difference is that ERNN architecture also has one recurrent layer. This recurrent layer takes the output of the hidden layer and sent it back to itself in the next iteration step. This extra layer allows the network to handle temporary dependencies between the inputs. In other words, the ERNN will be able to remember recent events and use them to predict the future.
Finally, the authors used a data set of crude oil spot prices to measure the accuracy and compare the results of the proposed ERNN prediction method versus a Feedforward ANN prediction method. This was done by calculating the MSE of both models. So, the authors model obtains significantly better accuracy, especially when making short-term predictions, also known as one-step-ahead prediction.

## 5.5 Chang, Naranjo and Guerron (2017)

In that year, Chang, et al. propose a deep learning prediction model to forecast pharmacy sales [51]. They started describing an interesting ANN architecture called autoencoder. The autoencoder is an ANN that takes input data and reproduces the same data at the output layer, usually compressing data in the hidden layers. So, they propose an

architecture composed of two shallow fully connected networks and one autoencoder.
So, the complete training process of this architecture works as follows. The first phase of training starts with an autoencoder to compress the input data. In the second phase, a shallow fully connected network takes the abstractions generated by the previous autoencoder and does the predictions in very old data of time series. In the last phase, a final shallow fully connected network takes the abstractions generated by the autoencoder and does the predictions in recent data of time series, this is done because the most recent data is more important for the final prediction. After training this proposed architecture, using pharmaceutical sales data, they make sales predictions to test the accuracy of the model. Thus, they proved that the model obtains an accuracy of 91% in the best case of prediction, and 55% in the worst case of prediction.

## 5.6 Tsantekidis, Passalis, Tefas, Kanniainen, Gabbouj and Iosifidis (2017)

In that year, Tsantekidis, et al. propose a deep learning prediction model to forecast Stock Prices [52]. So, they use a Convolutional Neural Network (CNN) to predict if the price movement of stocks will up, down or maintain. The authors propose this deep architecture because they say that CNN has the ability to work with input large scale of data. So, for the experimental part, they use a very large data set of about 4 million limit order events. Finally, they compare the proposed method against the Multilayer Perceptron Method (MLP) and Support Vector Machine Method (SVM).
It is important to note that, instead of using a regression network as in the aforementioned architectures, they use a classification network that classifies time series into 3 labels. Thus, although this architecture cannot predict the exact value of the stock price in the future, it can predict whether the stock price will go up, down or stay.
The architecture of their CNN model is composed by: one 2D Convolutional Layer with 16 filters of size (4, 40), one 1D Convolutional Layers with 16 filters of size (4,1), one 1D Convolutional Layers with 32 filters of size (3,1), one 1D Convolutional Layers with 32 filters of size (3,1) two Max Pooling Layers with size (2,1), one Fully Connected Layers with 32 neurons, and one Fully Connected Layers with 3 neurons. All layers, with except the output layer, use Leaky Rectified Linear Units (LRELU) as activation function. The output layer uses the Softmax activation function. They use Adaptive Moment Estimation Algorithm (ADAM) which is learning algorithm widely used for Deep Neural Network Training.
Once the proposed model is trained, they compare it with MLP and SVM traditional prediction models. The chosen MLP architecture is composed of: one input layer, one hidden layer of 128 neurons with LRELU activation function, and one output layer of 3 neurons with Softmax activation function. Finally, after calculating the prediction accuracy of these three models, they found that the CNN prediction model obtained slightly better accuracy compared to the other two models.

## 5.7 Bandara, Shi, Bergmeir, Hewamalage, Tran and Seaman (2019)

Bandara, et al. propose a novel sale forecasting prediction method based on Long-Short Term Memory (LSTM) neural network architecture [53]. They start to mention that most of the prediction methods actually used, both statistical and ANN-based methods, are univariate. They say that multivariate prediction methods are more powerful because, in many forecasting problems, different time series could present important relationships between them. Also, they mention that LSTM architectures can forecast long-term predictions with high accuracy, compared with other simple Recurrent Neural Networks such as Elman RNN. This is because LSTM supports more effectively the vanishing gradient problem which affects most of the deep neural network architectures. They applied this proposed method to forecast sales demand in E-Commerce, specifically they use a sales database of Walmart online store. In this way, they compare this prediction method vs other methods currently used by this platform, these are Naïve method, ARIMA method, and Exponential Smoothing method.

The proposed architecture is composed of three principal components: pre-processing layer, LSTM training layer, and post-processing layer. The first part is assigned to prepare the input data by normalization and re-scale. The second part includes the LSTM structure that will be trained. The last part is assigned to de-normalize and rescaling according to the original value scale.

The data that they used for training is a set of time series of sales of 1742 items sold online by Walmart.com. To calculate the prediction accuracy, they used the mean absolute percentage error (mMAPE). So after training, they propose 12 different architectures of the LSTM prediction method, and compare these proposed methods with the Naïve method, ARIMA method, and Exponential Smoothing method. They found that all the LSTM architectures proposed reach high accuracy compared with the other traditional methods mentioned above.

## 5.8 Summary Table

In this section, we will summarize the most important aspects of all the related works previously described. This information will be organized in the table 5.1.

| **Name of work** | **Author** | **Year** | **Proposed Prediction Architecture** | **Approach** |
|---|---|---|---|---|
| ”Sales forecasting using neural networks” | Thesing and Vornberger | 1997 | Feed-forward multilayer perceptron (Shallow-MLP) | Regression |
| ”Forecasting with artificial neural networks: The state of the art” | Zhang, Patuwo and Hu | 1998 | Feed-forward multilayer perceptron (Shallow-MLP) | Classification, Regression |
| ”An artificial neural network (p, d, q) model for timeseries forecasting” | Khashei and Bijari | 2010 | ARIMA + ANN | Regression |
| ”Forecasting energy market indices with recurrent neural networks: Case study of crude oil price fluctuations” | Jie Wang and Jun Wang | 2016 | Elman Recurrent Neural Network (ERNN) | Regression |
| ”A deep learning algorithm to forecast sales of pharmaceutical products” | Chang, Naranjo and Guerron | 2017 | Deep MLP + Autoencoder | Regression |
| “Forecasting stock prices from the limit order book using convolutional neural networks” | Tsantekidis, Passalis, Tefas, Kanniainen, Gabbouj and Iosifidis | 2017 | Convolutional Neural Network (CNN) | Classification |
| ”Sales demand forecast in e-commerce using a long short-term memory neural network methodology” | Bandara, Shi, Bergmeir, Hewamalage, Tran and Seaman | 2019 | Long-Short Term Memory (LSTM) | Regression |

Table 5.1: Summary table of related works

# Chapter 6

# Methodology

In this section, all the procedures followed to reach the final results will be shown. First, we will show a brief explanation of the software used to program the implementation of methods. Second, the database, data normalization and data input format used will be detailed. Third, we will explain architectures configurations of the four prediction methods to use: Shallow MLP, Deep MLP, CNN. This section also includes a detailed explanation of all the setup of the network's parameters. Fourth, metrics and formulas used to perform the comparison of the prediction methods implemented are described in detail.

## 6.1 MATLAB

MATLAB is a numerical computing environment, and also a programming language developed by MathWorks. The power of this language resides in the intuitive manipulation of matrices, for this reason, it is very used in the research area to implement mathematical models. So, due to the matrix nature of all operations involved in the creation and training of Neural Networks, MATLAB language is a very good option to implement the ANN architectures of our project.
For our project, MATLAB R2018a version will be used. This version includes many advantages to other previous versions. The most important difference is that the 2018a MATLAB Version allows more advanced modifications in deep neural networks, specifically in the use of LSTM and CNN. For example, older versions do not include the ADAM Optimization Algorithm, which is a very used algorithm in the training of Deep Neural Networks in the last years. Also, all the MATLAB methods and functions to create and train deep neural networks are implemented intuitively, but always following the rigorous mathematical basis behind them.

## 6.2 Dataset

The data set used in this project for training and testing of the ANN predictions models was obtained from an Ecuadorian Pharmacy Industry. So, for this project, we use sales databases of the pharmacy chain Farmaenlace. So, this data set is composed of weekly sales of 100 different products over a period of 5 years. Each time series of each product information are stored in one .txt file and includes the name of the product, code of product, and a total of 200 weekly sales.

So, based on the work [51], the distribution of the data set that we use is the following: 75% for training and 25% for test. That means that we will predict last year of sales based on the previous four years approximately.

## 6.3 Data Normalization

Once the data set have been defined, in this section, we proceed to normalize input data. The normalization of data is fundamental because some weekly sales values of the time series can be very high or very low compared with the rest of the values of time series. So, these big differences could cause poor accuracy during training and testing of the ANN model. Thus, through normalization we can narrow the time series sales values into a small and pre-defined range, in this case, it is [0 - 1].

$$n_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{6.1}$$

$x_i$: Original time series value at time i
$x_{max}$: Maximum value of time series
$x_{min}$: Minimum value of time series
$n_i$: Normalized time series value at time i

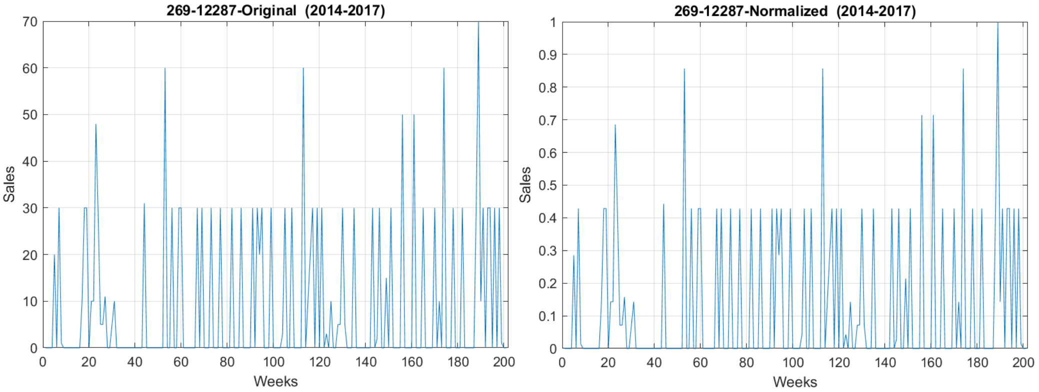


Figure 6.1: Original and Normalized time series

## 6.4 ANN Prediction Models

In this section, the three prediction time series architecture used in this project will be defined. First, we will explain 2 common neural network architectures used actually in sales forecasting, these are Shallow MLP and Deep MLP. After that, we will see in detail the proposed CNN architecture for sale forecasting. So, for each of these three architectures, we propose some different models.

Also, is important to mention that all the prediction models used in this work are Multi-step ahead prediction methods. That means that, during the test phase, each model predicts future sales using the previously predicted values.

### 6.4.1 Shallow Multi-Layer Perceptron Architecture

The first forecasting architecture that we will use is the Shallow Multi-Layer Perceptron (Shallow - MLP). We started with this architecture because it is one of the simplest and easiest to understand ANNs. Due that MLP is a universal function approximator, we can consider the sales time series like functions where time correspond to X-Axis and sales correspond to Y-Axis. In this way, we can address this problem as a regression problem. So, the objective is to forecast the weekly future sales, based on n lasts weeks' sales.

The principal structure of this shallow architecture is based on [48]. From this previous investigation, we extracted the most important parameters of the neural network such as the number of neurons, learning coefficient, loss function, etc. While a few other parameters were calculated experimentally for our purposes.

#### 6.4.1.1 Shallow-MLP Model 1

So, based on previous work on the use of this type of network to make sales predictions [48], we choose the following parameters setup:

| | |
|---|---|
| **Input layer size (n)** | 16 |
| **Hidden layer size** | 10 |
| **Output layer size** | 1 |
| **Time series size** | 200 |
| **Percentage of training data** | 75% |
| **Percentage of test data** | 25% |
| **Initial learning rate** | 0.005 |
| **Learning rate drop factor** | 0.1 |
| **Learning rate drop period** | 20 |
| **Transfer function** | RELU |
| **Optimization algorithm** | SGDM, Backpropagation |
| **Mini-batch size** | 32 |
| **Momentum** | 0.9 |
| **Loss function** | Half-mean-squared-error |
| **Max. Epochs number** | 125 |
| **Iterations per epoch** | 4 |
| **Max. Iteration** | 500 |
| **Regularization technique** | Dropout 0.5 |

Table 6.1: Shallow MLP Model 1 parameters

The training process of this architecture functions as follow. Once the weekly sales data set has been normalized and formatted, taking in this case n = 16, the data enter in the input layer. Then, the input layer passes this information to the next layer called the hidden layer, which is composed of 10 neurons. The neurons of hidden layers do a weighted sum between their respective weights and the data coming from the input layer. After that, hidden neurons apply an activation function to the weighted sum obtained. In this case, the activation function is RELU. Finally, the output layer, which in this case has only one neuron, takes all 10 output of hidden layer, and performs a weighted sum and applies the RELU activation function, obtaining the final output. This output corresponds to the sale prediction one week ahead after the input data.

After calculating the output of all patterns for one complete batch, in this case the batch size is 32, the ANN proceed to calculate the Half-MSE loss function. With this loss function, we proceed to correct the ANN weights using the SGDM optimization algorithm in combination with Backpropagation. This complete process is carried out in each iteration of each epoch, until reaching the maximum number of epochs established, in this case 125.

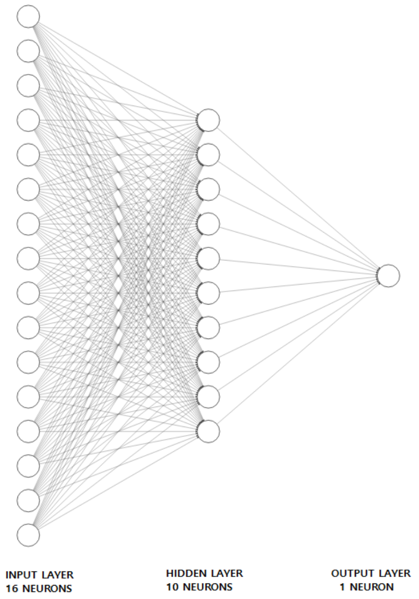


Figure 6.2: Shallow-MLP Model 1 architecture

#### 6.4.1.2 Shallow-MLP Model 2

This model is based on the previous model, but the only difference is that it has 100 neurons in the hidden layer.

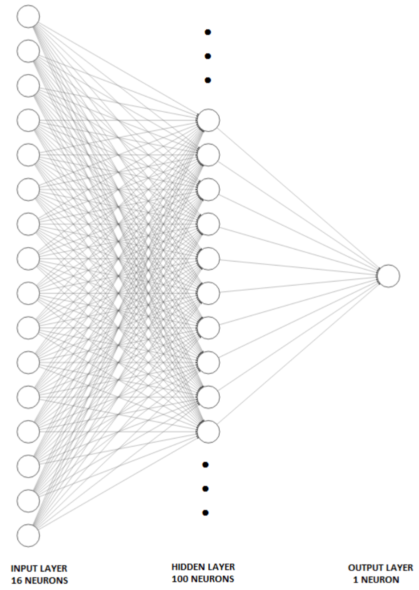


Figure 6.3: Shallow-MLP Model 2 architecture

#### 6.4.1.3 Shallow-MLP Model 3

This model is based on the previous model, but the only difference is that it has 1000 neurons in the hidden layer.

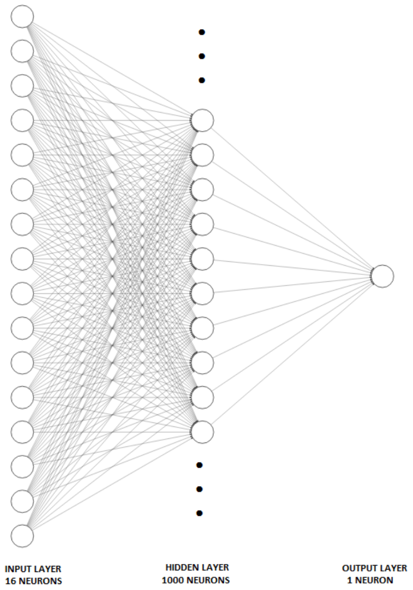


Figure 6.4: Shallow-MLP Model 3 architecture

### 6.4.2 Deep Multi-Layer Perceptron Architecture

The second approach of forecasting architecture that we will use is the Deep Multi-Layer Perceptron (Deep MLP). This architecture works very similar to the previous method; the only difference is the number of hidden layers. Shallow MLP only has one hidden layer, but if the number of hidden layers is more than 1, then the architecture is considered a Deep MLP. So, when we add hidden layers, the ANN acquires more computational power and abstraction ability. Though, the adding of many hidden layers can be counterproductive and reduce the accuracy of the prediction model. This is because when we add many hidden layers in a MLP architecture, it is possible to occur vanishing gradient problem.

The principal structure of this deep architecture is based on [51]. From this previous investigation, we extracted the most important parameters of the neural network such as the number of layers, number of neurons of each layer, learning coefficient, loss function, etc. While a few other parameters were calculated experimentally for our purposes.

#### 6.4.2.1 Deep-MLP Model 1

So, based on previous work on the use of this type of network to make sales predictions [51], we choose following parameters setup:

| | |
|---|---|
| **Input layer size (n)** | 16 |
| **First hidden layer size** | 10 |
| **Second hidden layer size** | 10 |
| **Output layer size** | 1 |
| **Time series size** | 200 |
| **Percentage of training data** | 75% |
| **Percentage of test data** | 25% |
| **Initial learning rate** | 0.005 |
| **Learning rate drop factor** | 0.1 |
| **Learning rate drop period** | 20 |
| **Transfer function** | RELU |
| **Optimization algorithm** | SGDM, Backpropagation |
| **Mini-batch size** | 32 |
| **Momentum** | 0.9 |
| **Loss function** | Half-mean-squared-error |
| **Max. Epochs number** | 125 |
| **Iterations per epoch** | 4 |
| **Max. Iteration** | 500 |
| **Regularization technique** | Dropout 0.5 |

Table 6.2: Deep MLP Model 1 parameters

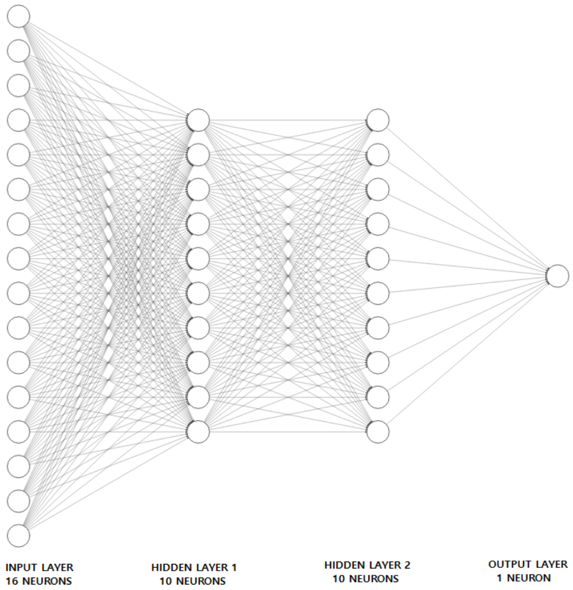


Figure 6.5: Deep-MLP Model 1 architecture

#### 6.4.2.2 Deep-MLP Model 2

This model is based on the previous model; the only difference is that it has three hidden layers of 10 neurons each one.

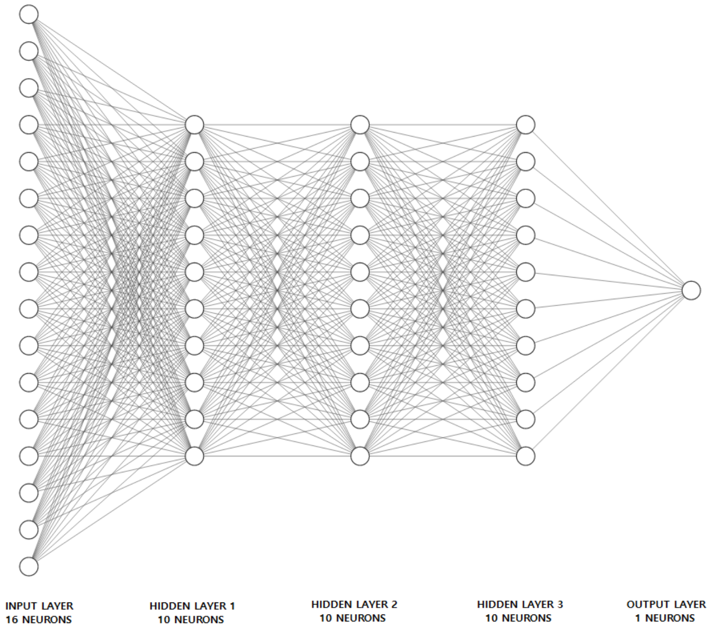


Figure 6.6: Deep-MLP Model 2 architecture

#### 6.4.2.3 Deep-MLP Model 3

This model is based on the previous model; the only difference is that it has four hidden layers of 10 neurons each one.

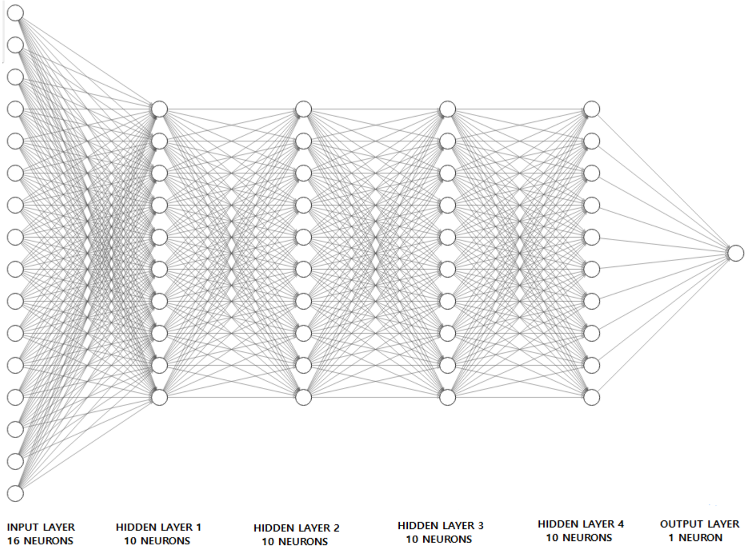


Figure 6.7: Deep-MLP Model 3 architecture

#### 6.4.2.4 Deep-MLP Model 4

This model is based on the previous model; the only difference is that it has five hidden layers of 10 neurons each one.

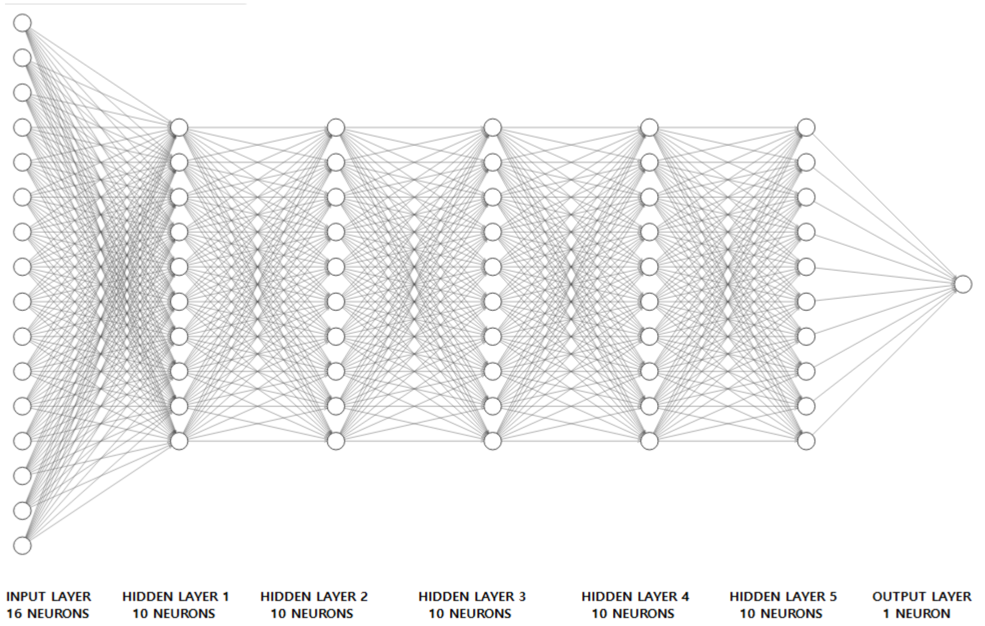


Figure 6.8: Deep-MLP Model 4 architecture

## 6.5 Convolutional Neural Network Architecture

Finally, this section details the sales prediction architecture based on CNN. This architecture will work similar to the Deep-MLP; the only difference is the addition of convolutional layers. These convolutional layers replace the fully connected layers, and therefore reducing the number of weights in the architecture [52].
We will propose different configurations of this architecture, in each configuration we will change: number of convolutional layers, number of filters, number of neurons and number of pooling layers. Thus, after performing the training and the test of each configuration, we will choose the best one in terms of precision and efficiency. Finally, these CNN architecture models will then be compared against the other models based on Shallow MLP and Deep MLP.

The principal structure of this CNN architecture is based on [52]. From this previous investigation, we extracted the most important parameters of the neural network such as the filter size, filter number, number of layers, number of neurons of each layer, learning coefficient, loss function, etc. While a few other parameters were calculated experimentally for our purposes.

### 6.5.1 CNN - Model 1

| | |
|---|---|
| **Input layer size (n)** | 16 |
| **1D Convolutional Layer 1** | Number of filters: 4,<br>Stride = 1,<br>Filter size: [7 x 1],<br>Padding: Same |
| **1D Max Pooling Layer 1** | Pooling filter size: [2 x 1], Stride = 2 |
| **1D Convolutional Layer 2** | Number of filters: 8,<br>Stride = 1,<br>Filter size: [5 x 1],<br>Padding: Same |
| **1D Max Pooling Layer 2** | Pooling filter size: [2 x 1], Stride = 2 |
| **1D Convolutional Layer 3** | Number of filters: 16,<br>Stride = 1,<br>Filter size: [3 x 1],<br>Padding: Same |
| **1D Max Pooling Layer 3** | Pooling filter size: [2 x 1], Stride = 2 |
| **1D Convolutional Layer 4** | Number of filters: 32,<br>Stride = 1,<br>Filter size: [1 x 1],<br>Padding: Same |
| **1D Max Pooling Layer 4** | Pooling filter size: [2 x 1], Stride = 2 |
| **Dropout** | Dropout factor = 0.2 |
| **Fully Connected Layer** | 10 |
| **Output layer size** | 1 |
| **Time series size** | 200 |
| **Percentage of training data** | 75% |
| **Percentage of test data** | 25% |
| **Initial learning rate** | 0.005 |
| **Learning rate drop factor** | 0.1 |
| **Learning rate drop period** | 20 |
| **Transfer function** | RELU |

Table 6.3: CNN Model 1 parameters

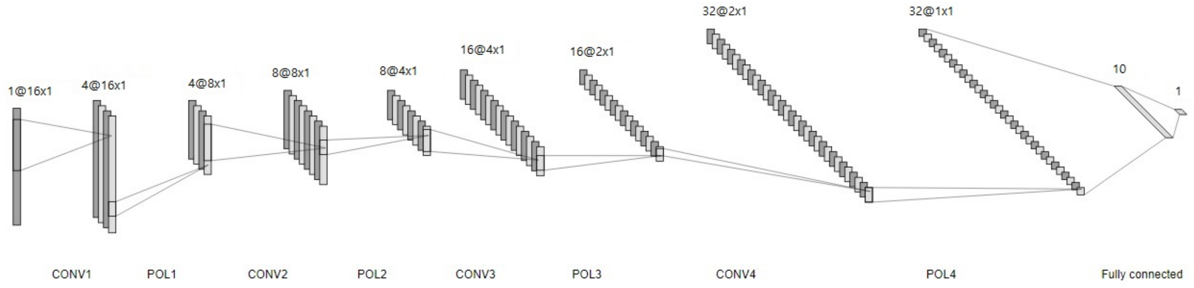


Figure 6.9: CNN - Model 1

### 6.5.2 CNN - Model 2

| | |
|---|---|
| **Input layer size (n)** | 16 |
| **1D Convolutional Layer 1** | Number of filters: 4, Stride = 1, Filter size: [7 x 1], Padding: Same |
| **1D Max Pooling Layer 1** | Pooling filter size: [2 x 1], Stride = 2 |
| **1D Convolutional Layer 2** | Number of filters: 8, Stride = 1, Filter size: [5 x 1], Padding: Same |
| **1D Max Pooling Layer 2** | Pooling filter size: [2 x 1], Stride = 2 |
| **1D Convolutional Layer 3** | Number of filters: 16, Stride = 1, Filter size: [3 x 1], Padding: Same |
| **1D Max Pooling Layer 3** | Pooling filter size: [2 x 1], Stride = 2 |
| **1D Convolutional Layer 4** | Number of filters: 32, Stride = 1, Filter size: [1 x 1], Padding: Same |
| **1D Max Pooling Layer 4** | Pooling filter size: [2 x 1], Stride = 2 |
| **Dropout** | Dropout factor = 0.2 |
| **Fully Connected Layer** | 5 |
| **Output layer size** | 1 |
| **Time series size** | 200 |
| **Percentage of training data** | 75% |
| **Percentage of test data** | 25% |
| **Initial learning rate** | 0.005 |
| **Learning rate drop factor** | 0.1 |
| **Learning rate drop period** | 20 |
| **Transfer function** | RELU |

Table 6.4: CNN Model 2 parameters

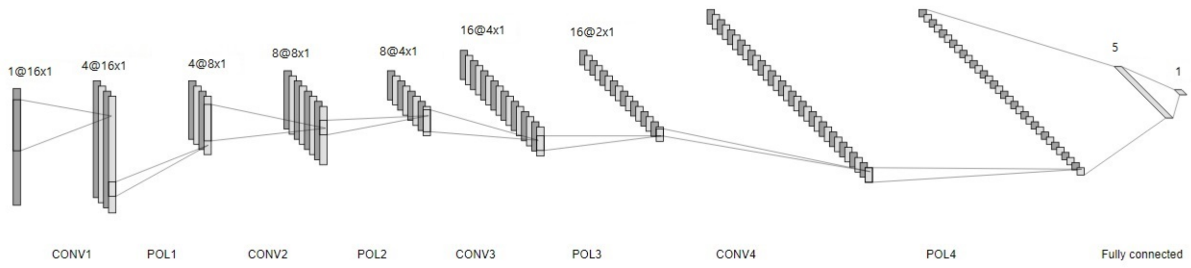


Figure 6.10: CNN - Model 2

## 6.6 Comparison metrics

In this section, we will discuss the methods and metrics used to measure both the accuracy and efficiency of sale forecast models used in this project. For this, three metrics will be used: accuracy of predictions, number of iterations and number of weights.

### 6.6.1 Accuracy

The accuracy corresponds to measure how much difference exists between the forecast weekly sales and real weekly sales. This value is measured through the root mean square error (RMSE). Thus, the lower the RMSE value, the more accurate the model prediction will be.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{6.2}$$

$y_i$: Observed value i
n: Number of total samples
$\hat{y}_i$: Predicted value i

### 6.6.2 Number of iterations

In addition to measuring the accuracy of the prediction model, it is important to measure the computational cost necessary to reach that level of precision. Since, although a high precision prediction model may exist, it is not very feasible if the number of iterations and processes involved is unreasonably high. Thus, one of the metrics used to measure the computational cost necessary during the training process is the number of iterations carried out.

### 6.6.3 Number of weights

To measure the computational cost, we use not only the number of iterations but also the number of weights in the network architecture. This is because some architectures reach high accuracy in very few iterations, but they could have a lot of weights. This means that on each training iteration the network must update a lot of weights, and this involves a lot of floating-point operations. Thus, in this project, we use both the number of iterations and the number of weights to be able to calculate the efficiency of the network.

# Chapter 7

# Results

To explain the result section in a clear way, three time series examples will be used. The first time series is an example in which the three prediction architectures used in this work produce a general high prediction accuracy. The second time series is an example in which the three prediction architectures produce a general medium prediction accuracy. Finally, the third time series is an example in which the three prediction architectures used in this work produce a general low prediction accuracy. In other words, these three time series examples represent the best case, the average case and the worst case in forecasting accuracy respectively.

Thus, predictions of these three time series were made using several models based on the 3 main architectures (Shallow-MLP, Deep-MLP, and CNN). After this, from each architecture, the model that obtained the lowest RMSE Test was chosen, that is, the model with the most prediction accuracy. Finally, these three models were compared using the 3 metrics described in the methodology section. In addition to the prediction methods based on ANN, we will also use three classical quantitative prediction methods (Naïve, Average, and Moving Average) which will serve as a reference for comparisons of the prediction accuracy.

Also, as we already mentioned in the previous chapter, all the prediction models used in this work are Multi-step ahead prediction methods. That means that, during the test phase, each model predicts future sales using the previously predicted values.

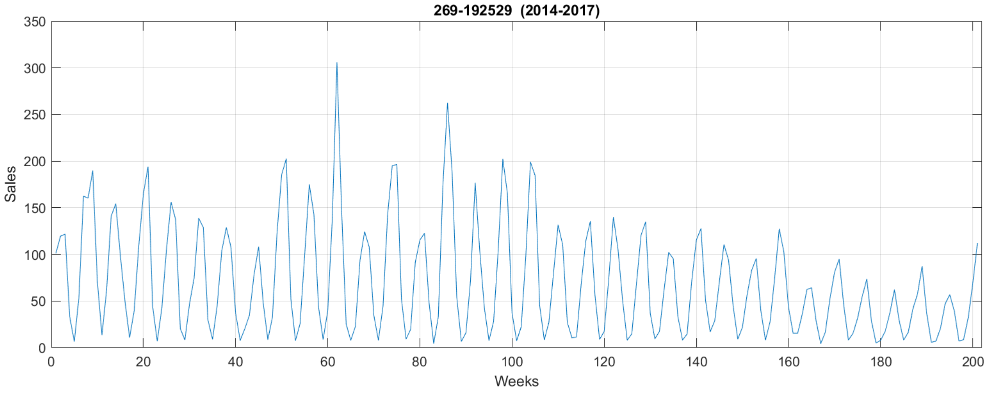


Figure 7.1: Time series 1

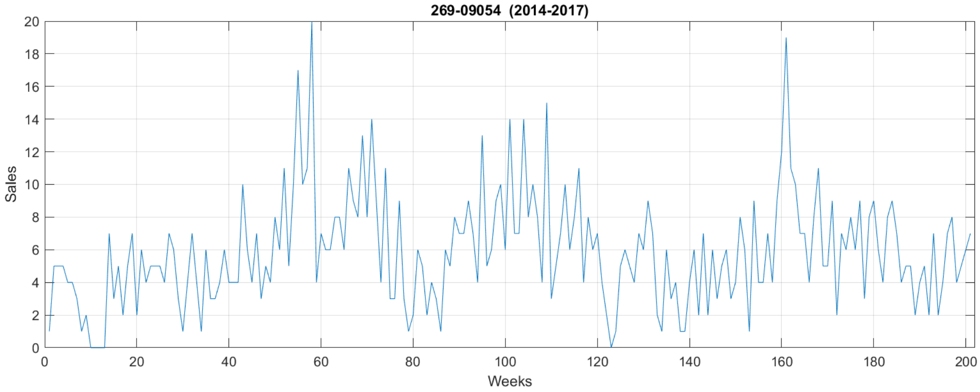


Figure 7.2: Time series 2

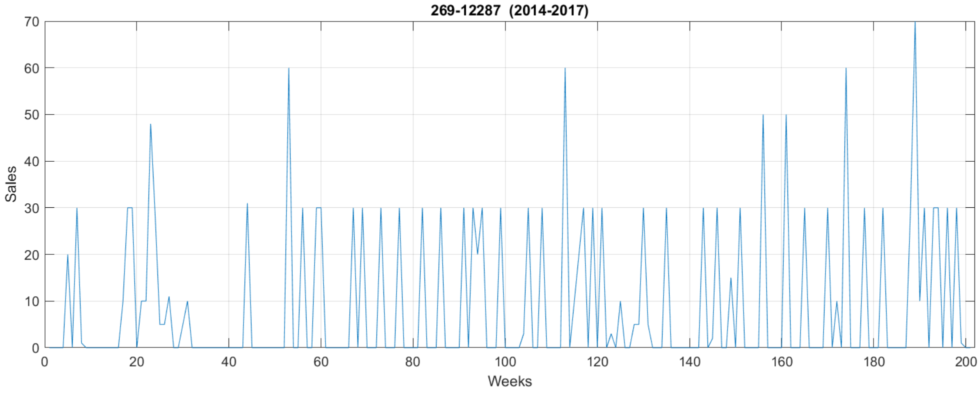


Figure 7.3: Time series 3

## 7.1 Time series 1

The first time series predicted is one of the examples in which the three prediction architectures reach very good performance. While this is not a time series as trivial as the stationary time series, the three prediction architectures used in this project achieve great prediction accuracy, as we will see below in the following results tables.

### 7.1.1 Classic quantitative methods

#### 7.1.1.1 Naïve approach

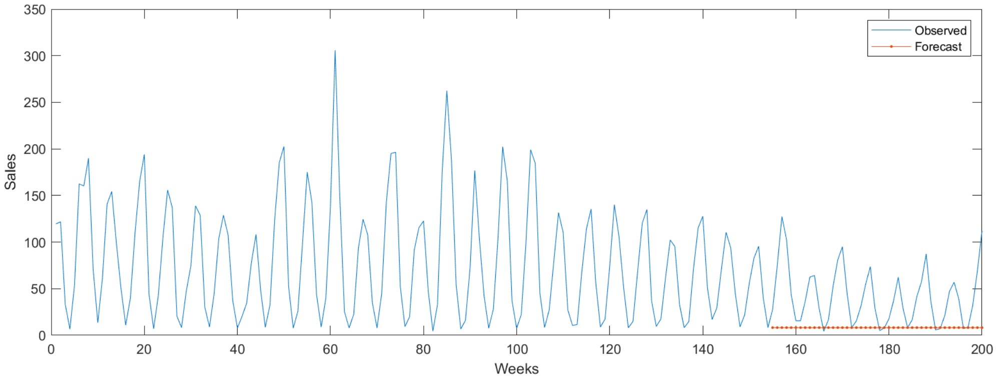


Figure 7.4: Time Series 1, Naive Prediction

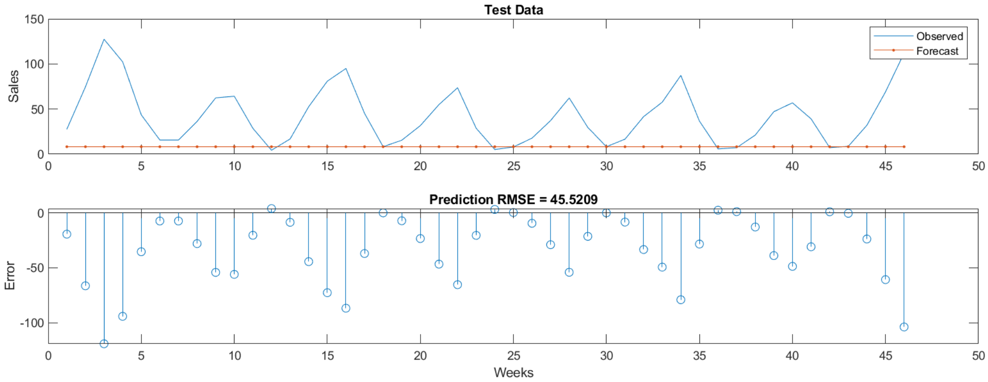


Figure 7.5: Time Series 1, Naive Prediction Error

#### 7.1.1.2 Average approach

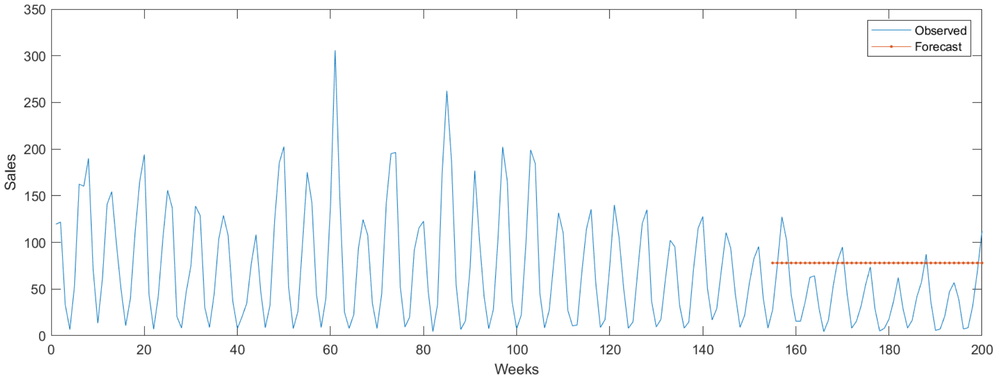


Figure 7.6: Time Series 1, Average Prediction

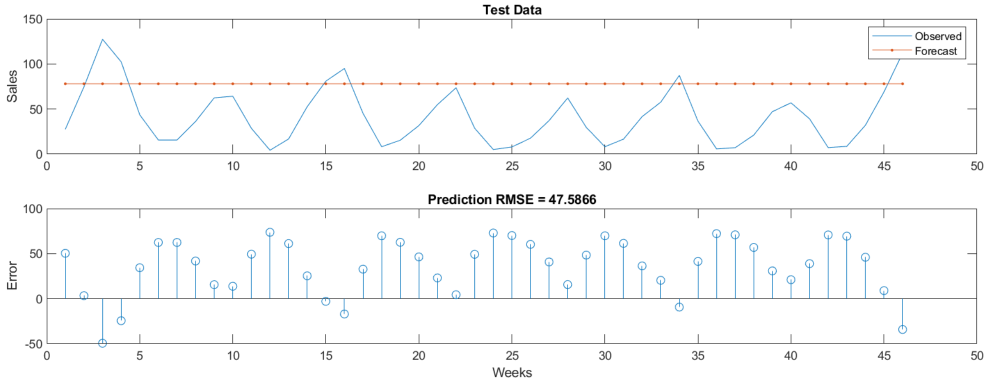


Figure 7.7: Time Series 1, Average Prediction Error

#### 7.1.1.3 Moving average approach

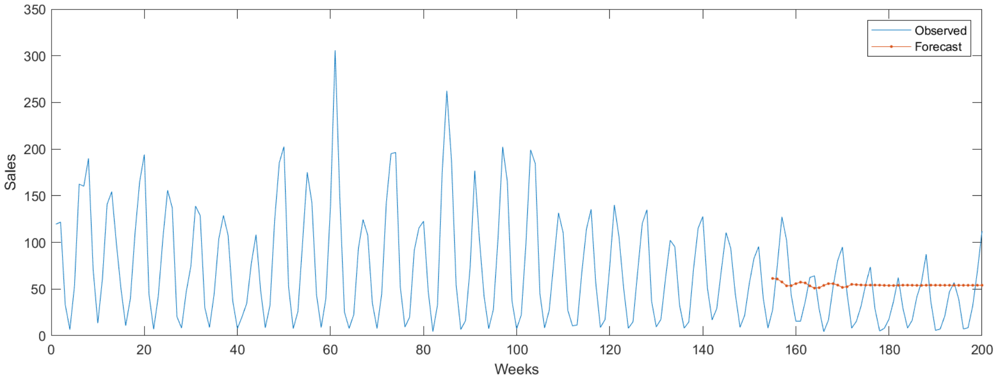


Figure 7.8: Time Series 1, Moving Average Prediction

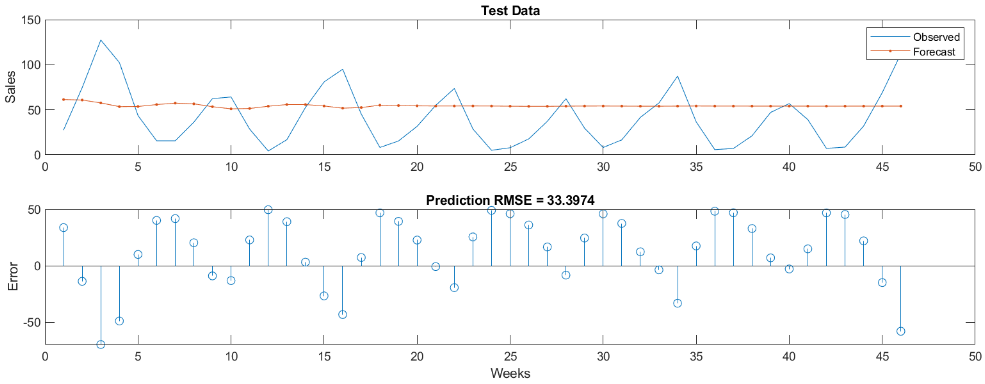


Figure 7.9: Time Series 1, Moving Average Prediction Error

### 7.1.2 Neural network based methods

#### 7.1.2.1 Shallow Multi-Layer Perceptron Architecture

| Time Series 1 | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Model 1** | 170 | 20.76 | 15.98 | 100 |
| **Model 2** | 1700 | 17.35 | 12.25 | 150 |
| **Model 3** | 17000 | 14.48 | 11.42 | 200 |

Table 7.1: Time Series 1, Shallow MLP prediction results

In table 7.1 we can see that the prediction accuracy increase along with the number of weights. So, in this case, the model with the best prediction accuracy is the model 3.

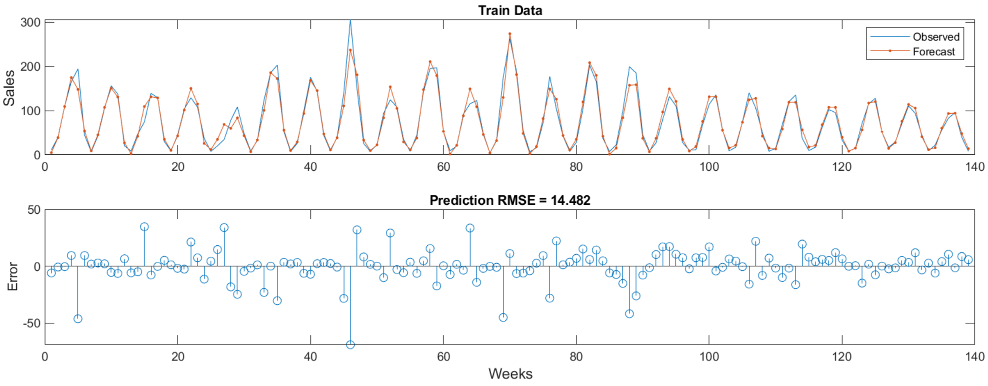


Figure 7.10: Time Series 1, Shallow MLP Model 3, Train Accuracy

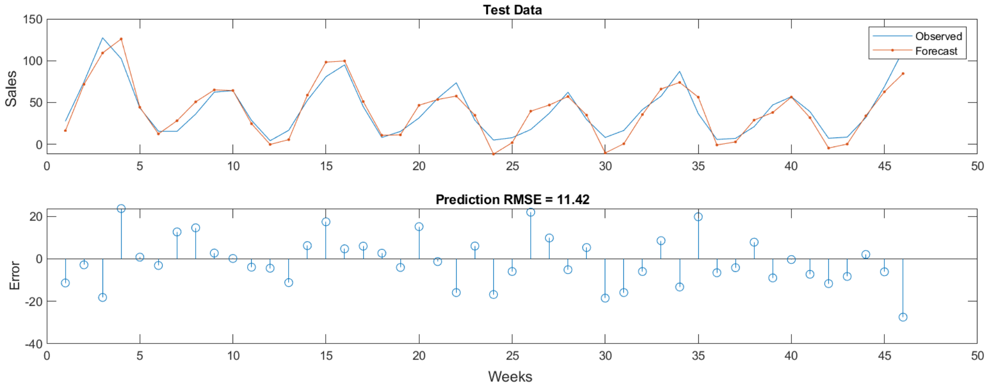


Figure 7.11: Time Series 1, Shallow MLP Model 3, Test Accuracy

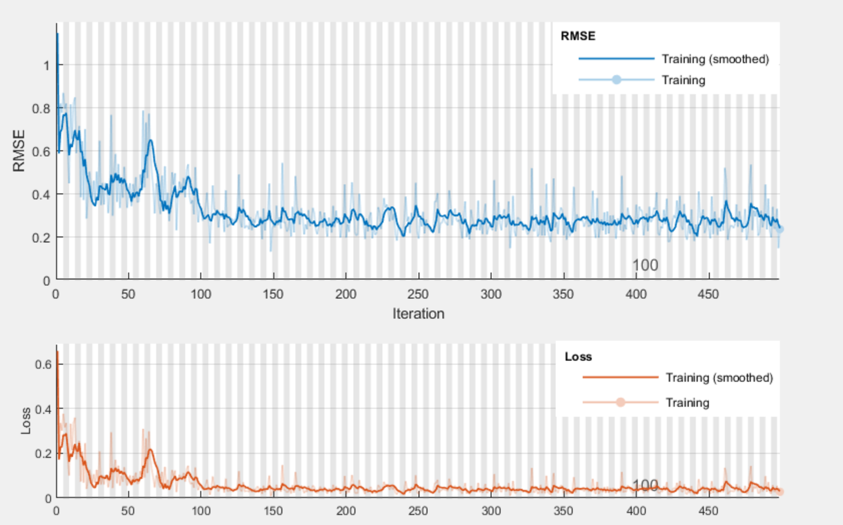


Figure 7.12: Time Series 1, Shallow MLP Model 3, Mini-batch RMSE and Mini-batch Loss

#### 7.1.2.2 Deep Multi-Layer Perceptron Architecture

| Time Series 1 | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Model 1** | 270 | 20.18 | 18.62 | 100 |
| **Model 2** | 370 | 20.09 | 16.71 | 150 |
| **Model 3** | 470 | 19.40 | 16.65 | 200 |
| **Model 4** | 570 | 19.37 | 15.41 | 250 |

Table 7.2: Time Series 1, Deep MLP prediction results

In table 7.2 we can see that the prediction accuracy increase along with the number of weights, which also means that increase along with the number of hidden layers. So, the model with the best prediction accuracy is the model 4.

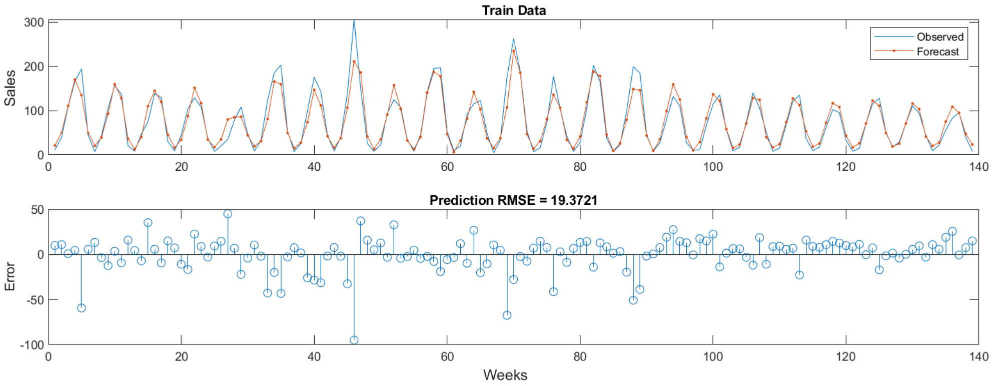


Figure 7.13: Time Series 1, Deep MLP Model 3, Train Accuracy

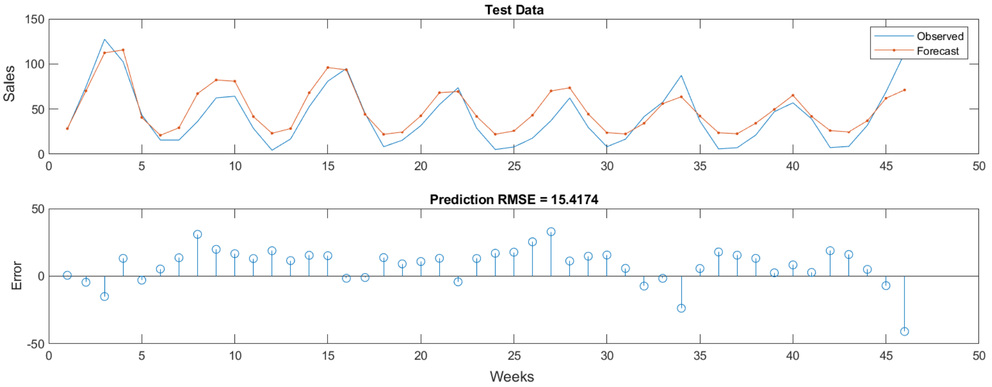


Figure 7.14: Time Series 1, Deep MLP Model 3, Test Accuracy

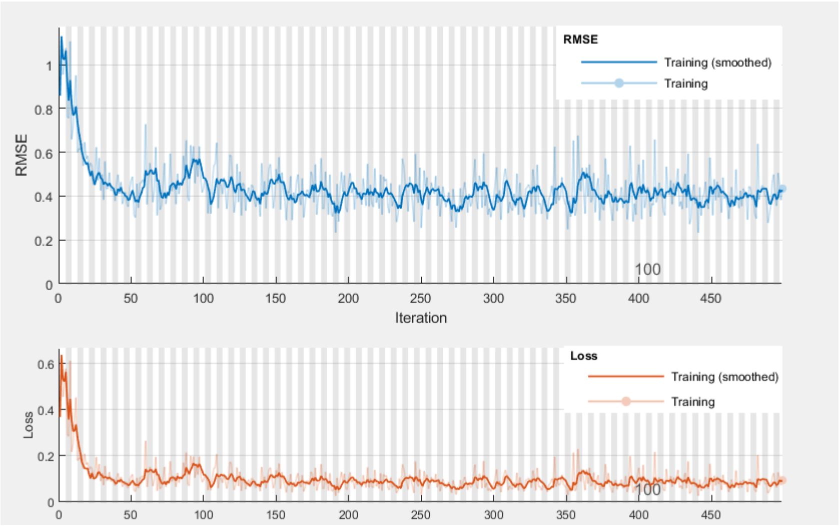


Figure 7.15: Time Series 1, Deep MLP Model 3, Mini-batch RMSE and Mini-batch Loss

#### 7.1.2.3 CNN prediction architecture

| Time Series 1 | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Model 1** | 478 | 16.58 | 14.11 | 200 |
| **Model 2** | 313 | 18.23 | 15.36 | 200 |

Table 7.3: Time Series 1, CNN prediction results

In this architecture, we will analyze both models because both have advantages and disadvantages. On the one hand, the first model has the best prediction accuracy. On the other hand, the second model is not so accurate than the first model, but it uses less number of weights.

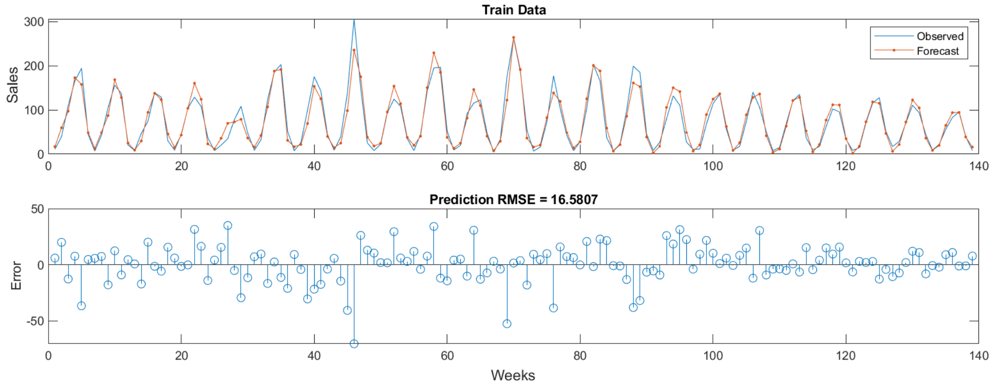


Figure 7.16: Time Series 1, CNN Model 1, Train Accuracy

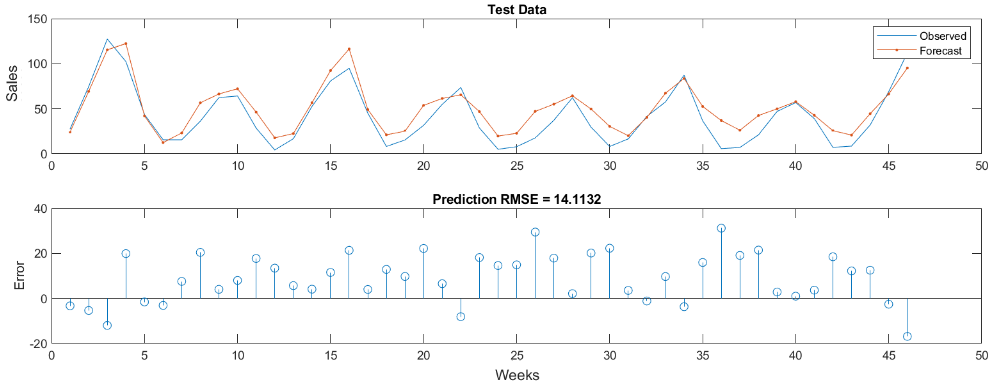


Figure 7.17: Time Series 1, CNN Model 1, Test Accuracy

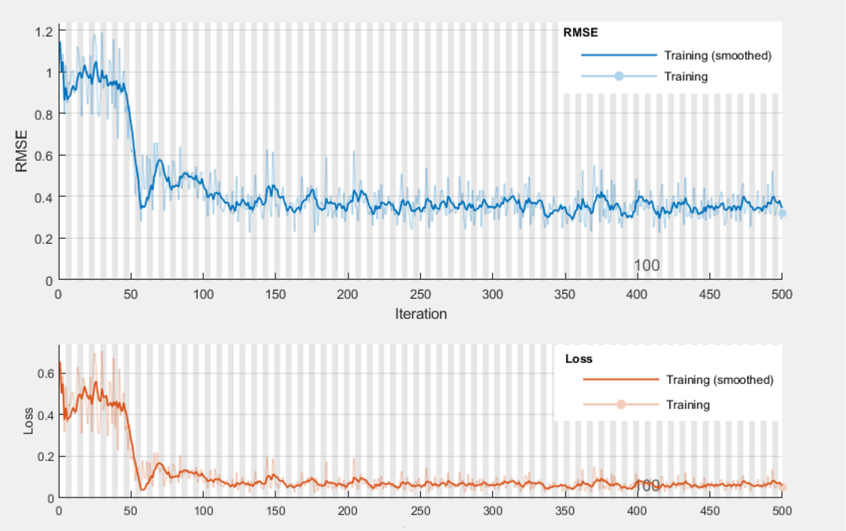


Figure 7.18: Time Series 1, CNN Model 1, Mini-batch RMSE and Mini-batch Loss

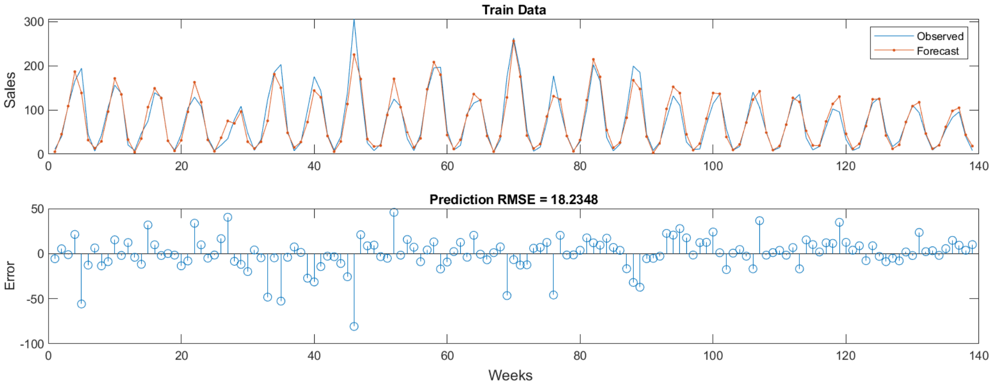


Figure 7.19: Time Series 1, CNN Model 2, Train Accuracy

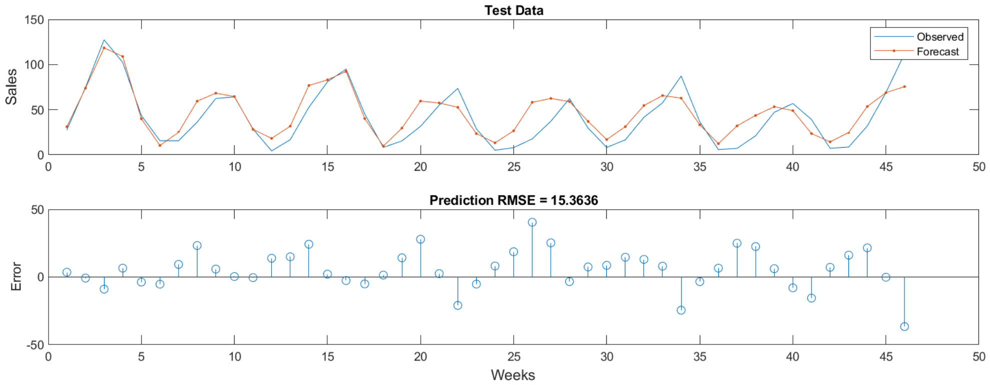


Figure 7.20: Time Series 1, CNN Model 2, Test Accuracy

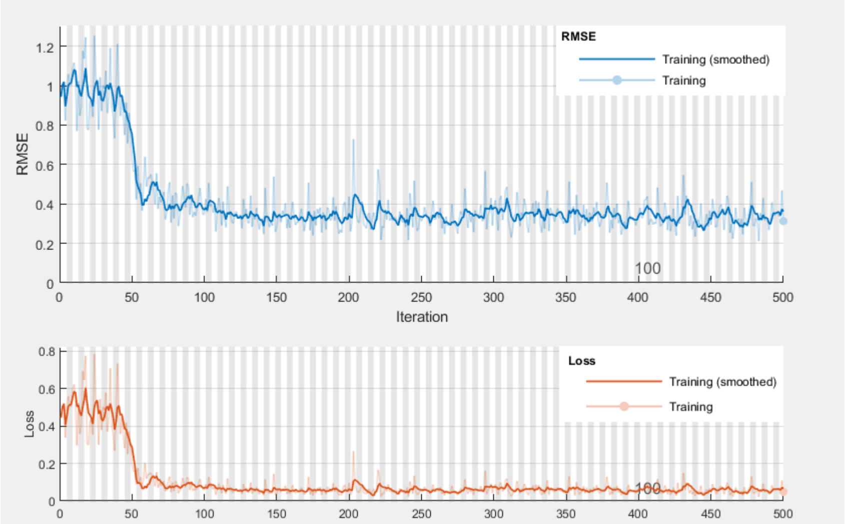


Figure 7.21: Time Series 1, CNN Model 2, Mini-batch RMSE and Mini-batch Loss

## 7.2 Time series 2

The second time series predicted is one of the examples in which the three prediction architectures reach medium performance. While this is not a time series as trivial as the first time series, the three prediction architectures used in this project achieve medium prediction accuracy, as we will see below in the following results tables.

### 7.2.1 Classic quantitative methods

#### 7.2.1.1 Naïve approach

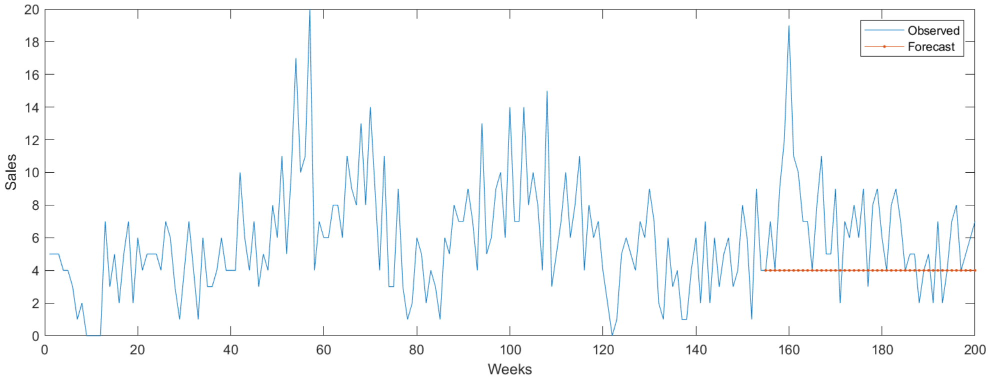


Figure 7.22: Time Series 2, Naive Prediction

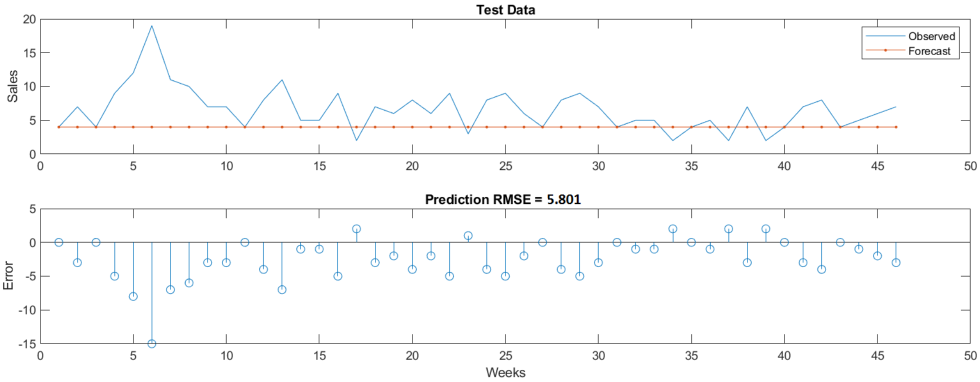


Figure 7.23: Time Series 2, Naive Prediction Error

#### 7.2.1.2 Average approach

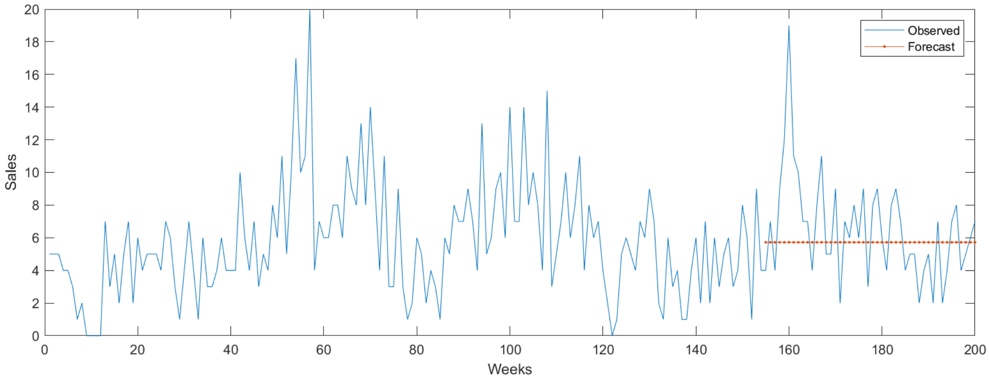


Figure 7.24: Time Series 2, Average Prediction

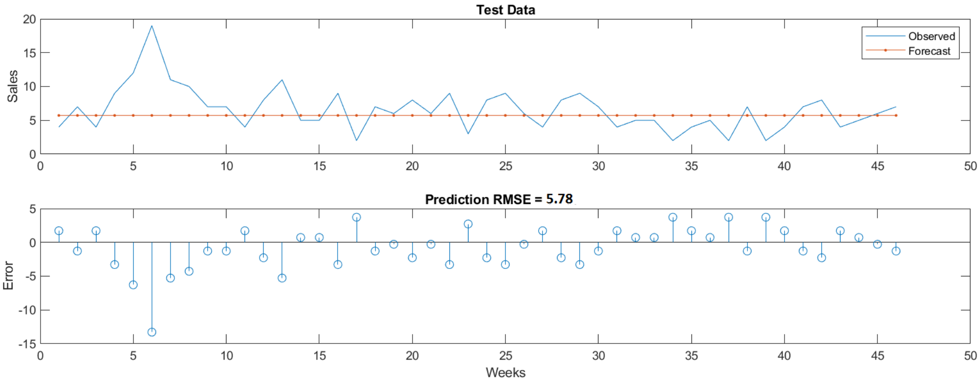


Figure 7.25: Time Series 2, Average Prediction Error

#### 7.2.1.3 Moving average approach

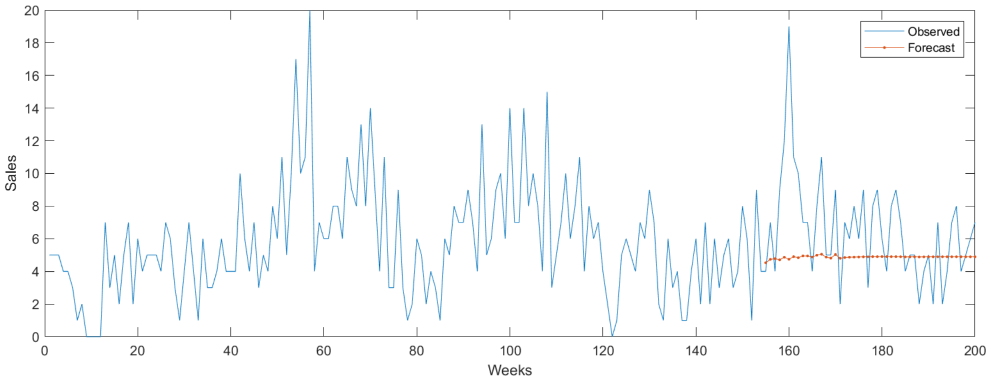


Figure 7.26: Time Series 2, Moving Average Prediction

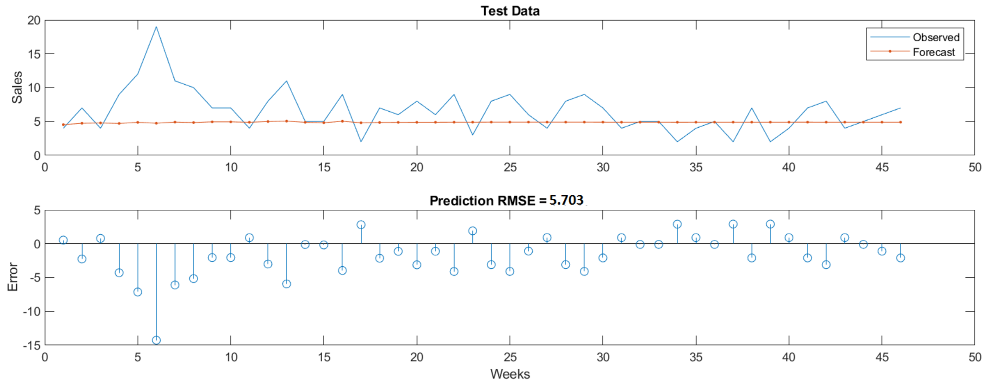


Figure 7.27: Time Series 2, Moving Average Prediction Error

### 7.2.2 Neural network based methods

#### 7.2.2.1 Shallow Multi-Layer Perceptron Architecture

| Time Series 2 | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Model 1** | 170 | 2.191 | 5.41 | 100 |
| **Model 2** | 1700 | 1.184 | 5.03 | 150 |
| **Model 3** | 17000 | 0.546 | 4.29 | 200 |

Table 7.4: Time Series 2, Shallow MLP prediction results

In table 7.4 we can see that the prediction accuracy increase along with the number of weights. So, in this case, the model with the best prediction accuracy is the model 3.

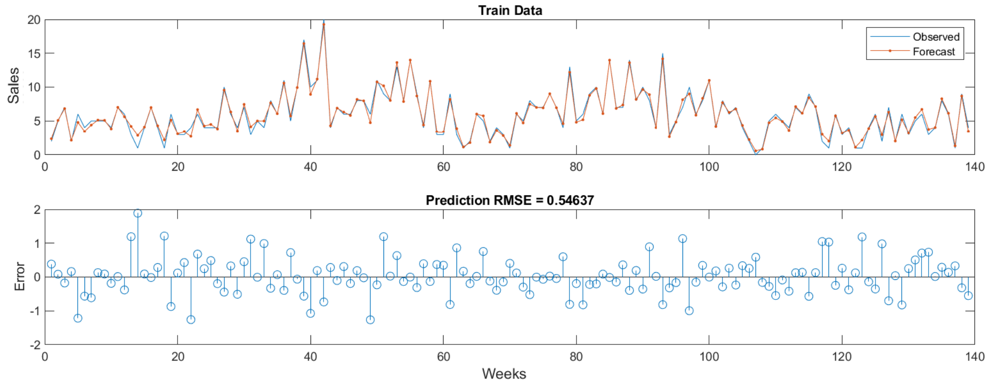


Figure 7.28: Time Series 2, Shallow MLP Model 3, Train Accuracy

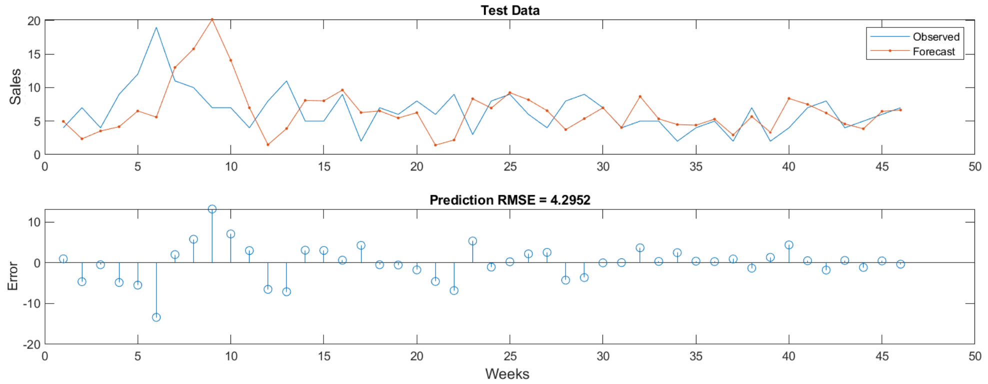


Figure 7.29: Time Series 2, Shallow MLP Model 3, Test Accuracy

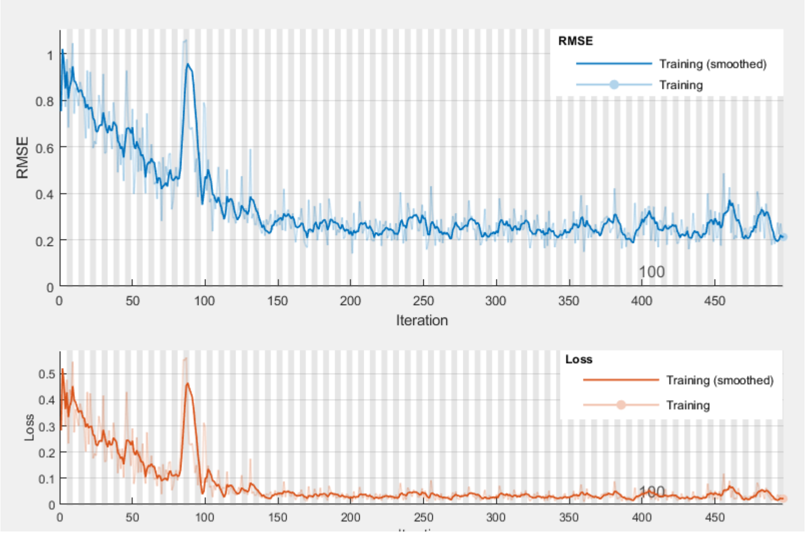


Figure 7.30: Time Series 2, Shallow MLP Model 3, Mini-batch RMSE and Mini-batch Loss

#### 7.2.2.2 Deep Multi-Layer Perceptron Architecture

| Time Series 2 | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Model 1** | 270 | 1.61 | 4,70 | 100 |
| **Model 2** | 370 | 1.59 | 4.69 | 150 |
| **Model 3** | 470 | 1.55 | 4.64 | 200 |
| **Model 4** | 570 | 1.46 | 4.13 | 250 |

Table 7.5: Time Series 2, Deep MLP prediction results

In table 7.5 we can see that the prediction accuracy increase along with the number of weights, which also means that increase along with the number of hidden layers. So, the model with the best prediction accuracy is the model 4.

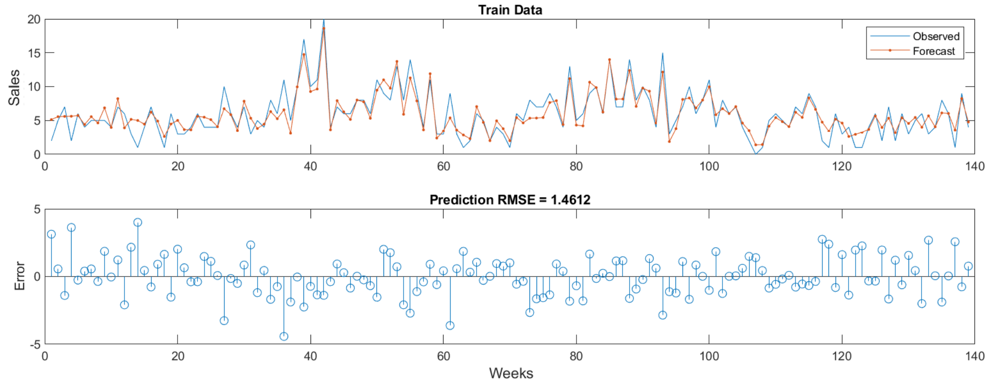


Figure 7.31: Time Series 2, Deep MLP Model 3, Train Accuracy

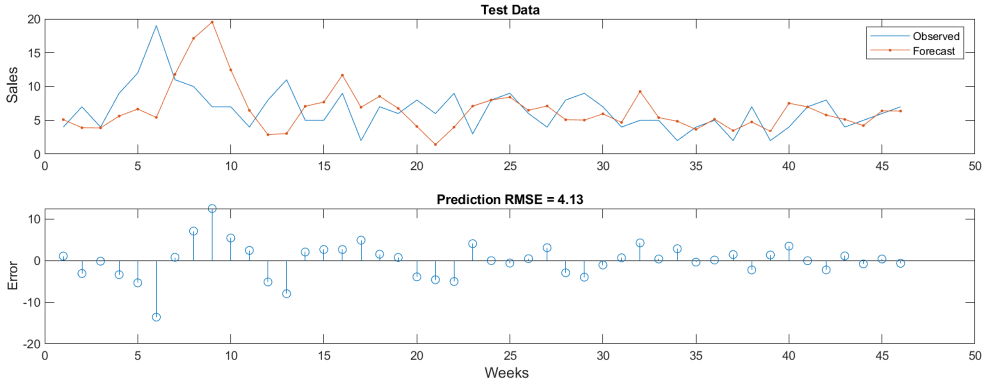


Figure 7.32: Time Series 2, Deep MLP Model 3, Test Accuracy

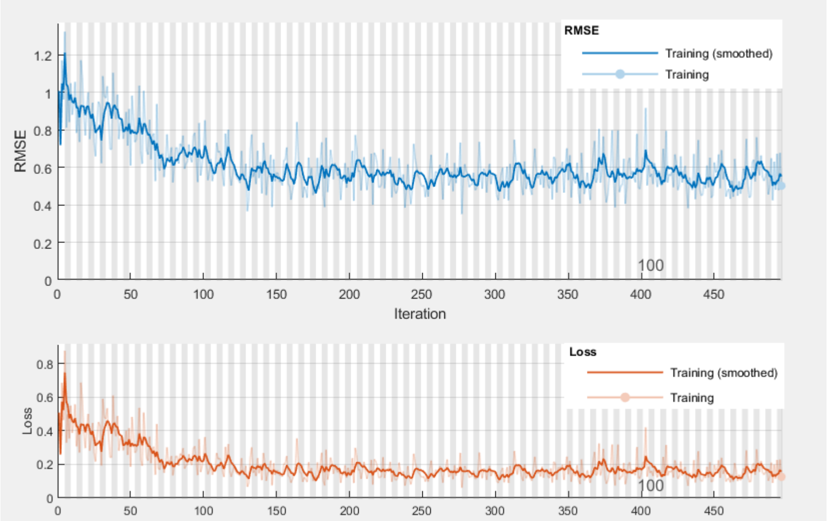


Figure 7.33: Time Series 2, Deep MLP Model 3, Mini-batch RMSE and Mini-batch Loss

#### 7.2.2.3 CNN prediction architecture

| Time Series 2 | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Model 1** | 478 | 1.25 | 3.64 | 200 |
| **Model 2** | 313 | 1.28 | 3.84 | 200 |

Table 7.6: Time Series 2, CNN prediction results

In this architecture, we will analyze both models because both have advantages and disadvantages. On the one hand, the first model has the best prediction accuracy. On the other hand, the second model is not so accurate than the first model, but it uses less number of weights.

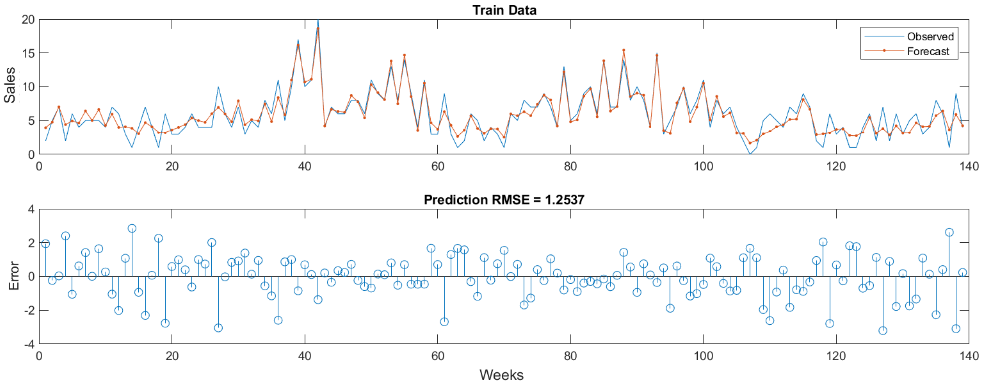


Figure 7.34: Time Series 2, CNN Model 1, Train Accuracy

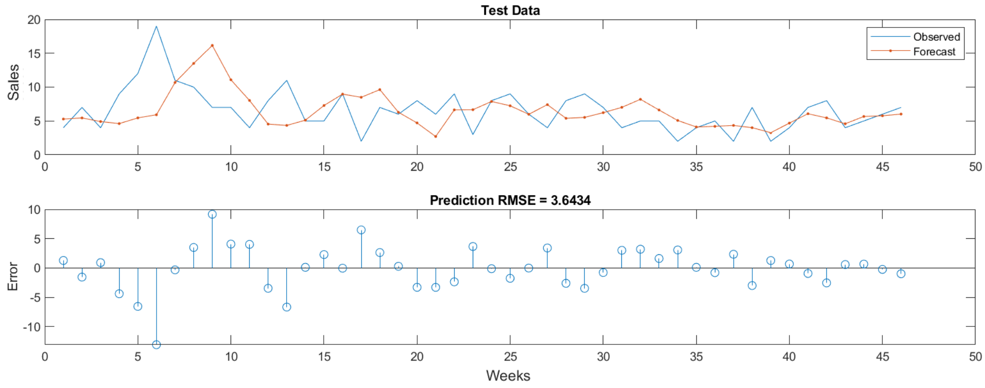


Figure 7.35: Time Series 2, CNN Model 1, Test Accuracy

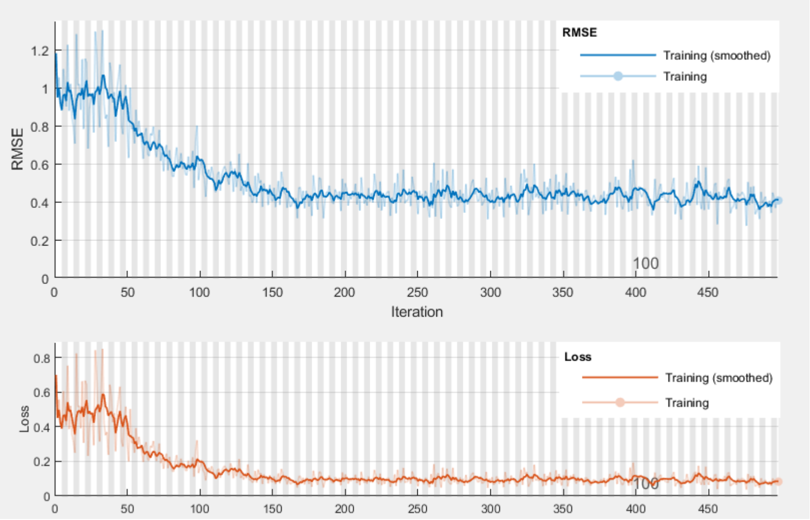


Figure 7.36: Time Series 2, CNN Model 1, Mini-batch RMSE and Mini-batch Loss

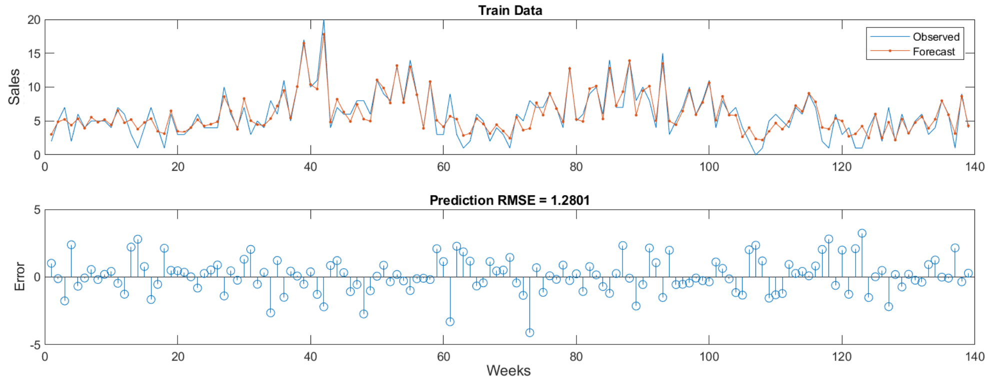


Figure 7.37: Time Series 2, CNN Model 2, Train Accuracy

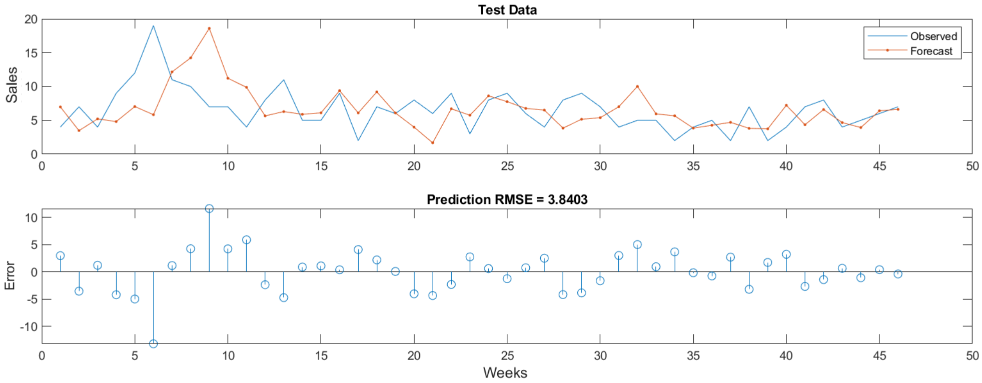


Figure 7.38: Time Series 2, CNN Model 2, Test Accuracy

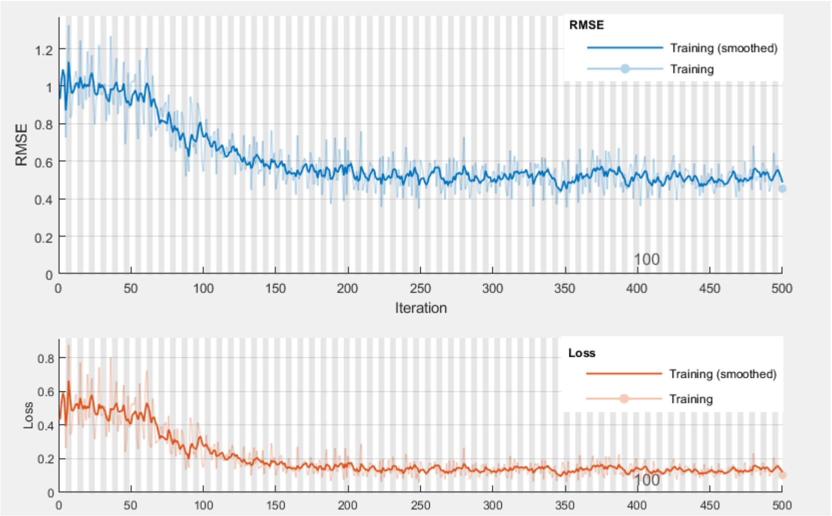


Figure 7.39: Time Series 2, CNN Model 2, Mini-batch RMSE and Mini-batch Loss

## 7.3 Time series 3

Unlike the previous time series predicted, this third time series does not present very clear predictable trends, instead show very random behavior. This causes it to be very difficult to predict future sales, resulting in a very high Test RMSE. Despite this, it is still important to analyze this type of time series as they represent the worst case when using these prediction methods.

### 7.3.1 Classic quantitative methods

#### 7.3.1.1 Naïve approach

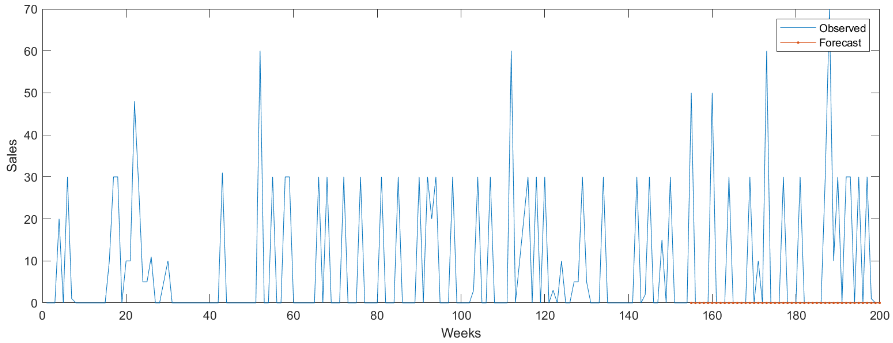


Figure 7.40: Time Series 3, Naive Prediction

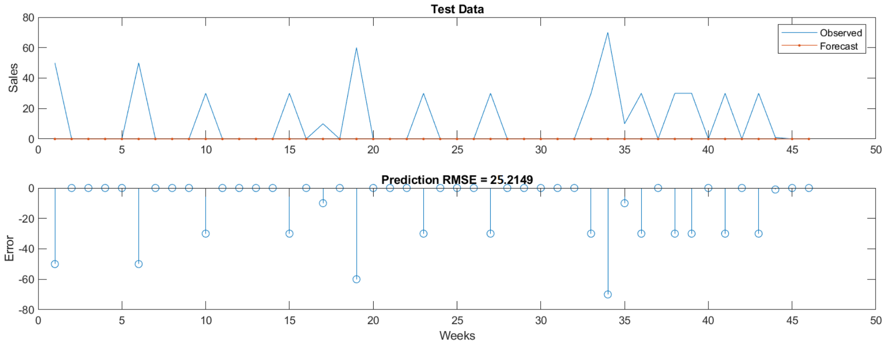


Figure 7.41: Time Series 3, Naive Prediction Error

#### 7.3.1.2 Average approach

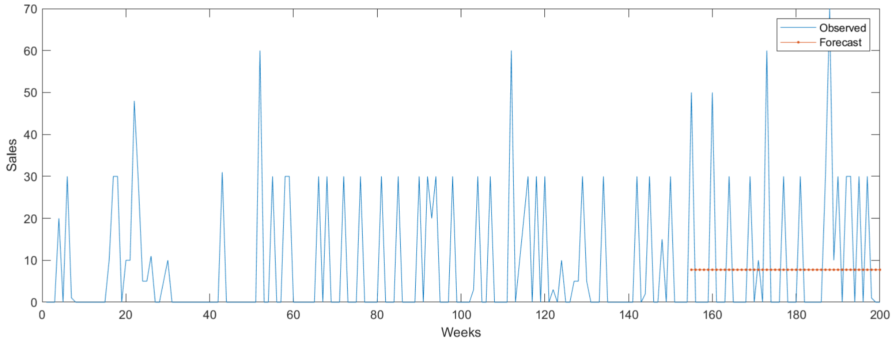


Figure 7.42: Time Series 3, Average Prediction

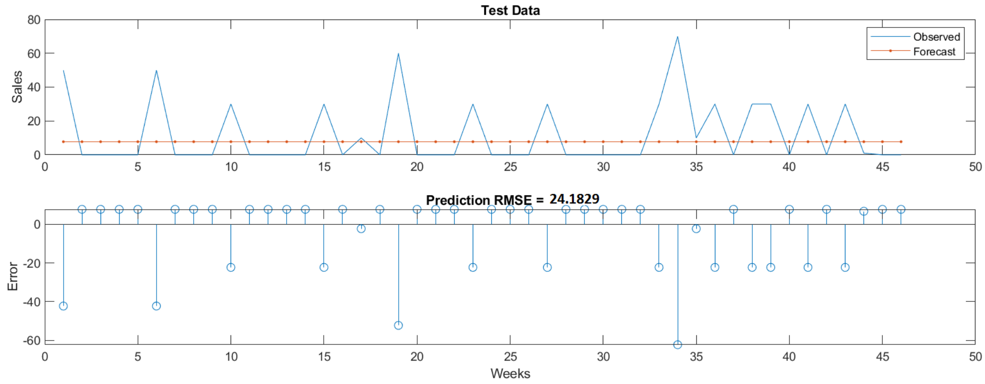


Figure 7.43: Time Series 3, Average Prediction Error

#### 7.3.1.3 Moving average approach

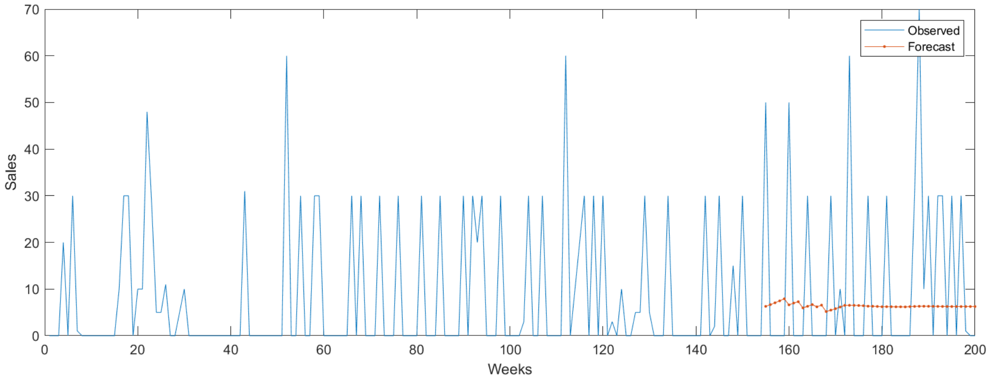


Figure 7.44: Time Series 3, Moving Average Prediction

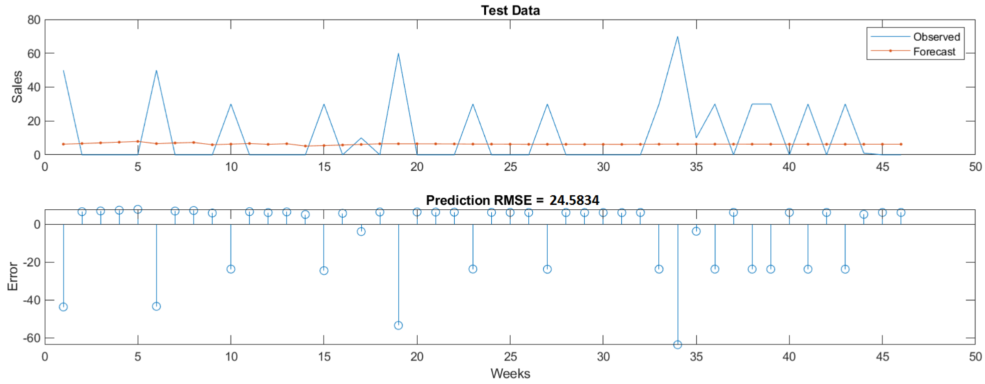


Figure 7.45: Time Series 3, Moving Average Prediction Error

### 7.3.2 Neural network based methods

#### 7.3.2.1 Shallow Multi-Layer Perceptron Architecture

| Time Series 3 | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Model 1** | 170 | 9.42 | 23.92 | 100 |
| **Model 2** | 1700 | 4.93 | 23.50 | 200 |
| **Model 3** | 17000 | 2.29 | 23.48 | 200 |

Table 7.7: Time Series 3, Shallow MLP prediction results

In table 7.7 we can see that the prediction accuracy increase along with the number of weights. So, in this case, the model with the best prediction accuracy is the model 3.

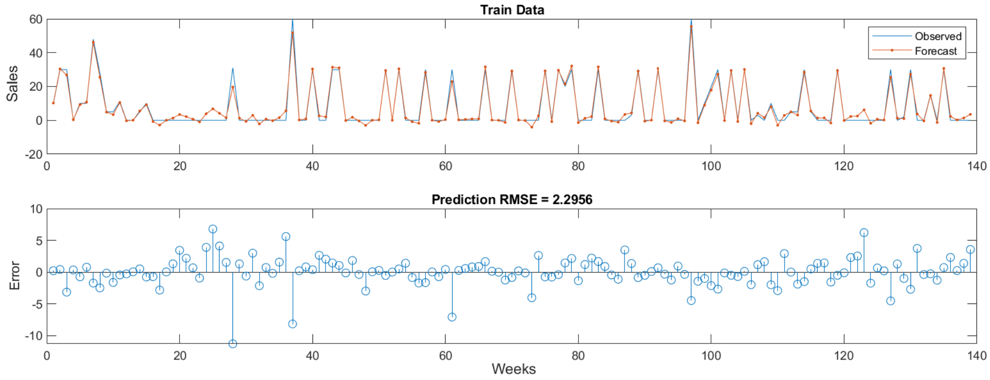


Figure 7.46: Time Series 3, Shallow MLP Model 3, Train Accuracy

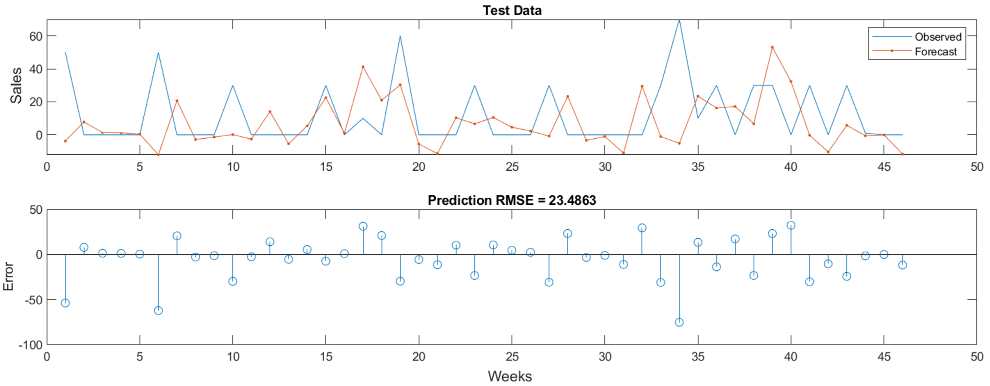


Figure 7.47: Time Series 3, Shallow MLP Model 3, Test Accuracy

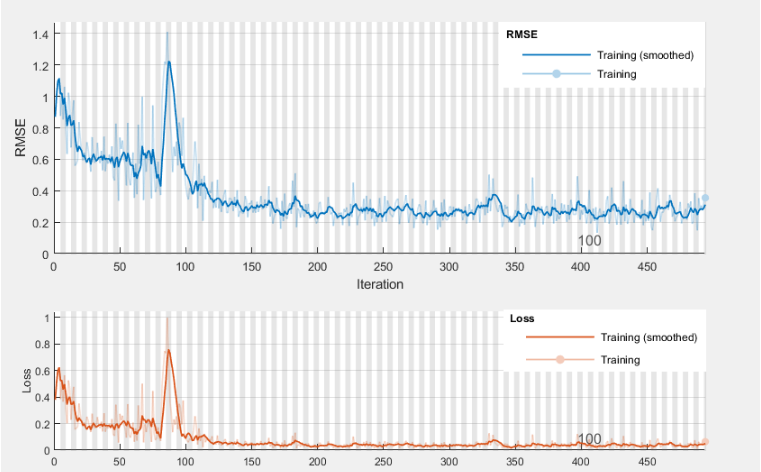


Figure 7.48: Time Series 3, Shallow MLP Model 3, Mini-batch RMSE and Mini-batch Loss

#### 7.3.2.2 Deep Multi-Layer Perceptron Architecture

| **Time Series 3** | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Model 1** | 270 | 6.48 | 23.02 | 200 |
| **Model 2** | 370 | 6.23 | 22.56 | 250 |
| **Model 3** | 470 | 5.98 | 21.22 | 250 |
| **Model 4** | 570 | 5.94 | 21.18 | 250 |

Table 7.8: Time Series 3, Deep MLP prediction results

In table 7.8 we can see that the prediction accuracy increase along with the number of weights, which also means that increase along with the number of hidden layers. So, the model with the best prediction accuracy is the model 4.

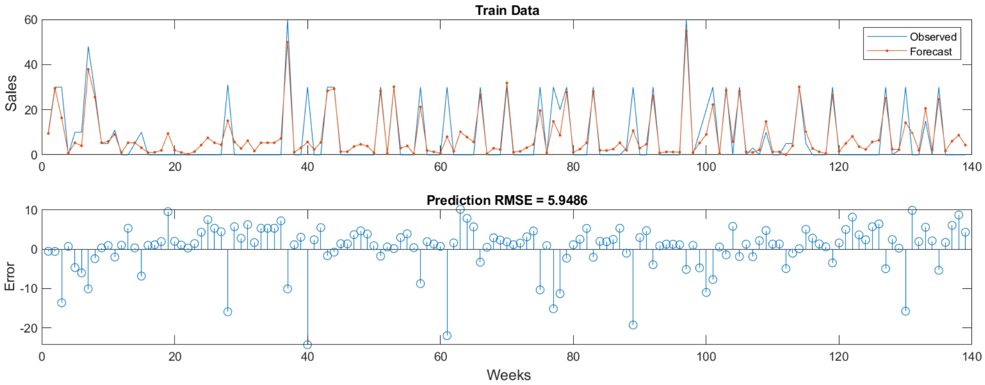


Figure 7.49: Time Series 3, Deep MLP Model 3, Train Accuracy

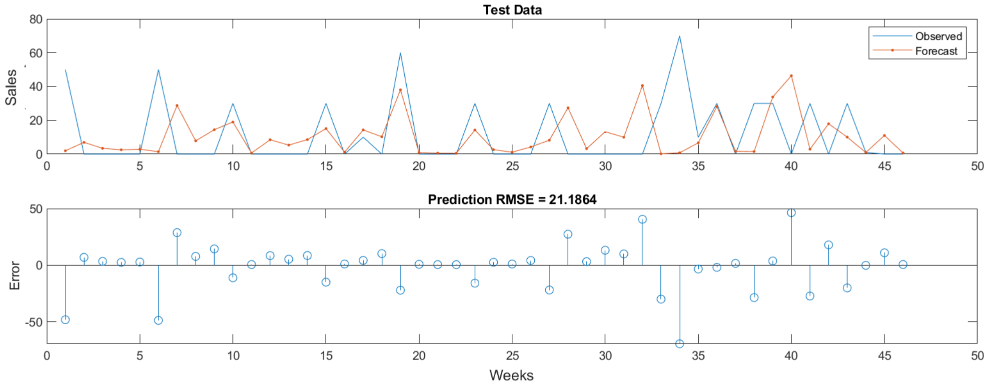


Figure 7.50: Time Series 3, Deep MLP Model 3, Test Accuracy

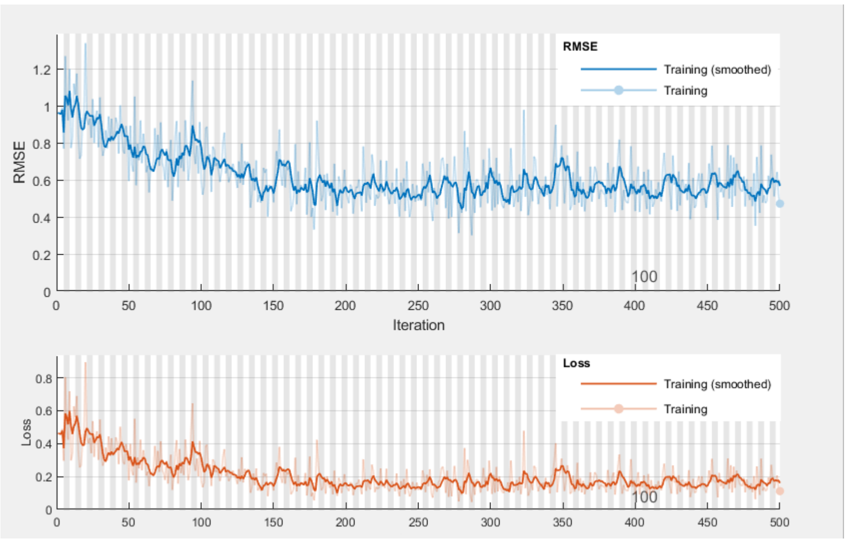


Figure 7.51: Time Series 3, Deep MLP Model 3, Mini-batch RMSE and Mini-batch Loss

#### 7.3.2.3 CNN prediction architecture

| Time Series 3 | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Model 1** | 478 | 4.36 | 19.97 | 200 |
| **Model 2** | 313 | 4.77 | 20.22 | 250 |

Table 7.9: Time Series 3, CNN prediction results

In this architecture, we will analyze both models because both have advantages and disadvantages. On the one hand, the first model has the best prediction accuracy. On the other hand, the second model is not so accurate than the first model, but it uses less number of weights.

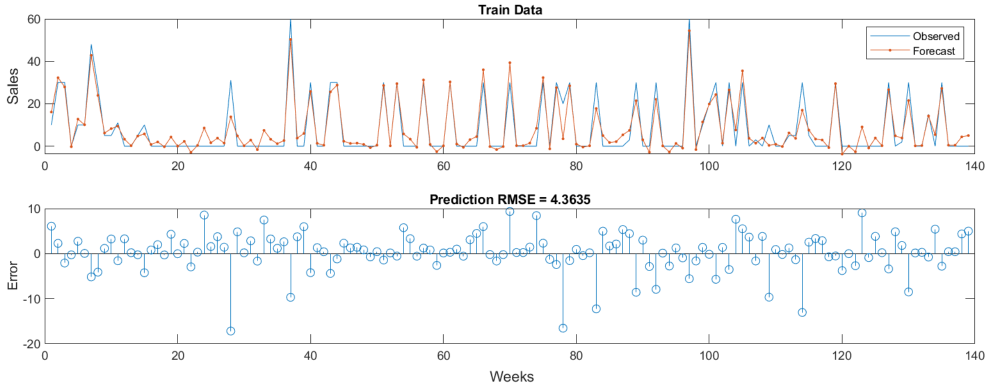


Figure 7.52: Time Series 3, CNN Model 1, Train Accuracy

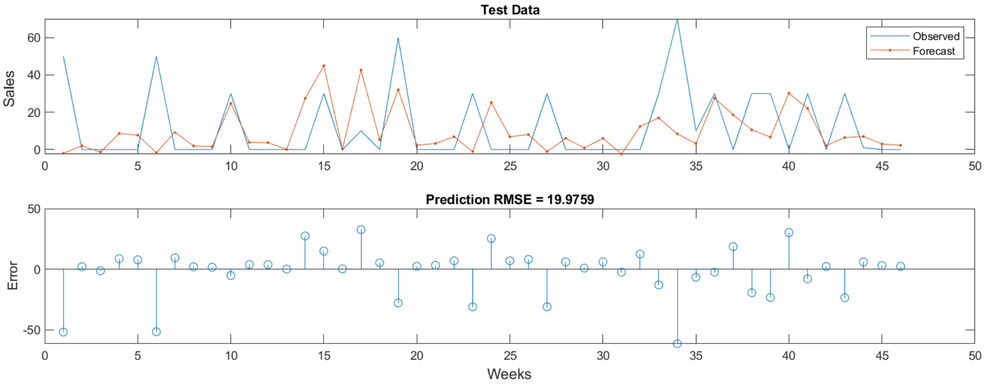


Figure 7.53: Time Series 3, CNN Model 1, Test Accuracy

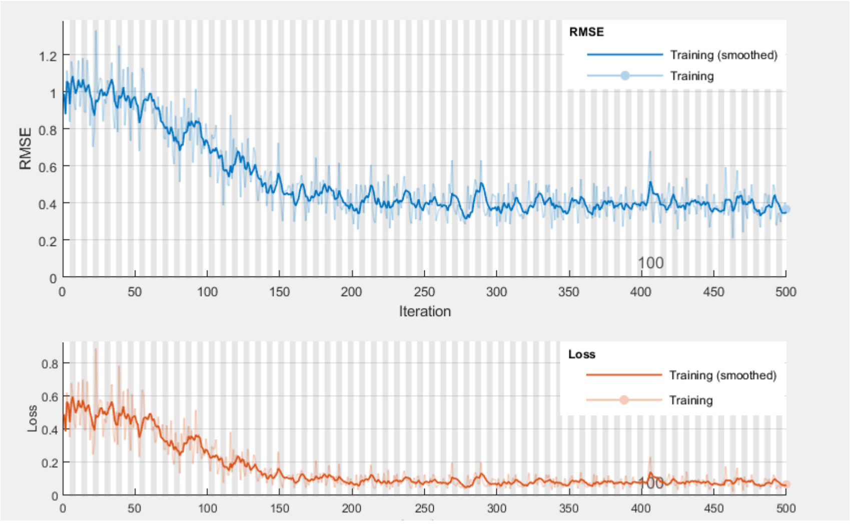


Figure 7.54: Time Series 3, CNN Model 1, Mini-batch RMSE and Mini-batch Loss

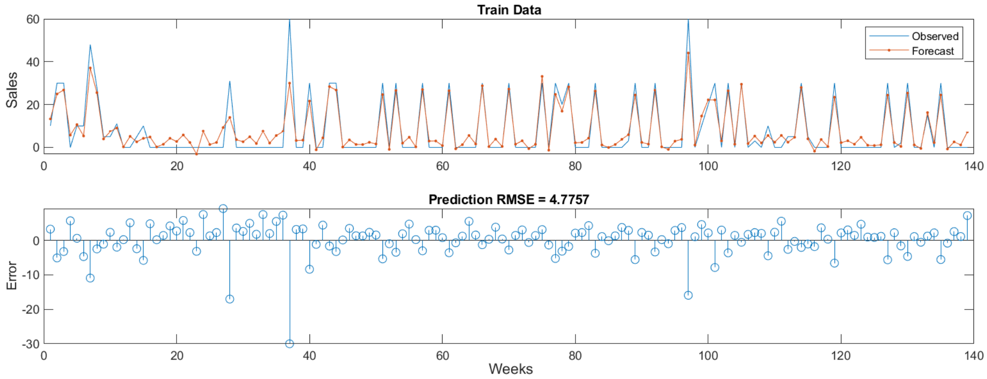


Figure 7.55: Time Series 3, CNN Model 2, Train Accuracy

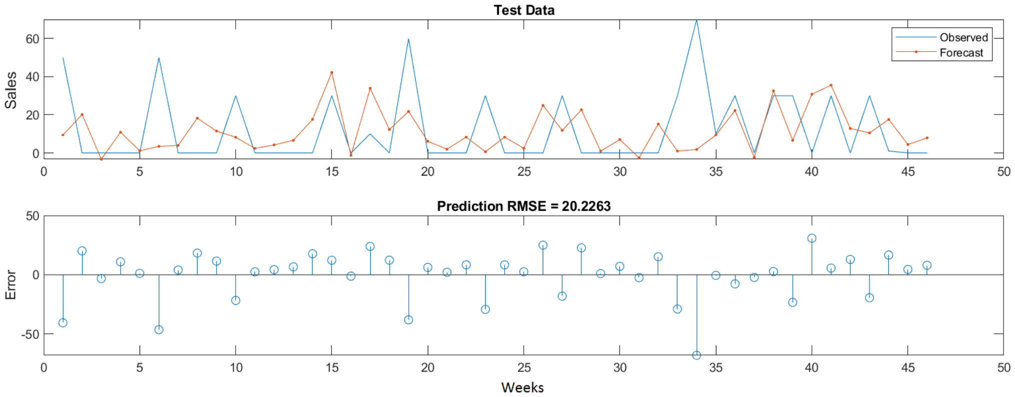


Figure 7.56: Time Series 3, CNN Model 2, Test Accuracy

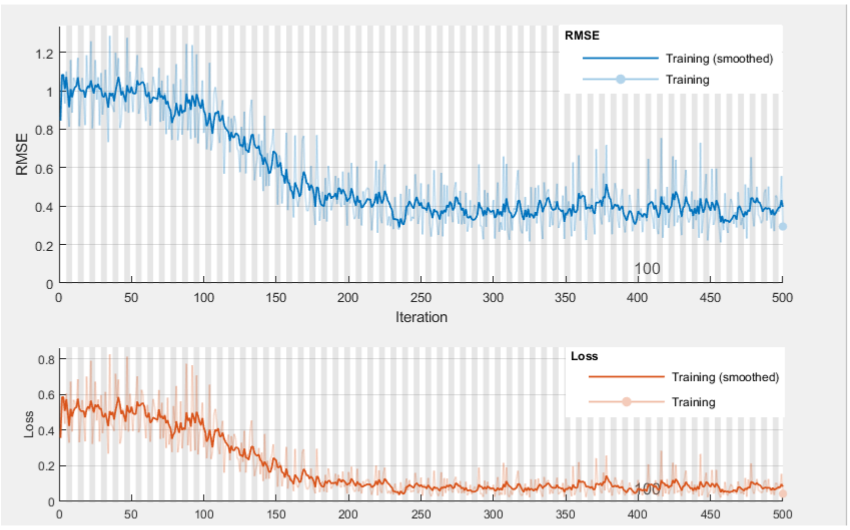


Figure 7.57: Time Series 3, CNN Model 2, Mini-batch RMSE and Mini-batch Loss

# Chapter 8

# Discussion

After seeing the results through tables and graphs, it is time to discuss and interpret the results obtained. The objective of this section is to describe the advantages and disadvantages of each prediction model. For this, we will analyze and compare the models based on the metrics: accuracy, number of weights and iterations. In this way, we can choose the model that achieves the highest prediction accuracy using the lowest number of weights in its structure.

## 8.1 Comparison of the models

Once we have chosen the best models of the three main architectures, then we will proceed to compare them. For this, we will use three important metrics: prediction accuracy, number of weights and iterations. Also, we will compare these prediction methods based on neural networks, with the classical quantitative prediction methods: Naïve, Average, and Moving Average.

| **Time Series 1** | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Naïve method** | ————- | ————- | 45.52 | ————- |
| **Average method** | ————- | ————- | 47.58 | ————- |
| **Moving average method** | ————- | ————- | 33.39 | ————- |
| **Shallow-MLP Model 3** | 17000 | 14.48 | 11.42 | 200 |
| **Deep-MLP Model 4** | 570 | 19.37 | 15.41 | 250 |
| **CNN Model 1** | 478 | 16.58 | 14.11 | 200 |
| **CNN Model 2** | 313 | 18.23 | 15.36 | 200 |

Table 8.1: Comparison of prediction models using Time Series 1

In table 8.1, the prediction results of the time series 1 are shown. First, we can see that the three classical quantitative prediction methods reach the lowest prediction accuracy. So, the Naïve prediction method reaches a prediction accuracy Test RMSE of

45.52. The Average method reaches an accuracy of 47.58. Finally, the Moving average method reaches an accuracy of 33.39.

Regarding prediction models based on ANN, we can see that the model with the best prediction accuracy is the Shallow-MLP Model 3. This model reaches a Test RMSE of 11.42. But, the problem with this model is that to reach that high accuracy, the model needs to adjust 17000 weights. That means a high computational cost during training compared to the other models.

Now, the prediction model based on ANN with the lowest prediction precision is the Deep-MLP Model 4. This model reaches a Test RMSE of 15.41. But, the main difference of this model is that it uses less weights compared to Shallow-MLP Model 3.

The CNN Model 1 is more accurate than Deep-MLP Model 4, but less accurate than Shallow-MLP Model 3. However, the biggest advantage of this model is that it uses a smaller amount of weights than the other two models, so it needs to adjust 478 weights during training.

Finally, CNN Model 2 is the model with the smallest amount of weights of all other models. While this model is not as accurate as the CNN Model 1, this model only uses 313 weights. This means a great reduction of computational cost at the moment of adjusting weights during training.

| **Time Series 2** | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Naïve method** | ———- | ———- | 5.801 | ———- |
| **Average method** | ———- | ———- | 5.780 | ———- |
| **Moving average method** | ———- | ———- | 5.703 | ———- |
| **Shallow-MLP Model 3** | 17000 | 0.546 | 4.295 | 200 |
| **Deep-MLP Model 4** | 570 | 1.461 | 4.709 | 250 |
| **CNN Model 1** | 478 | 1.250 | 3.640 | 200 |
| **CNN Model 2** | 313 | 1.280 | 3.840 | 200 |

Table 8.2: Comparison of prediction models using Time Series 2

In table 8.2, the prediction results of the time series 2 are shown. First, we can see that the three classical quantitative prediction methods reach the lowest prediction accuracy. So, the Naïve prediction method reaches a prediction accuracy Test RMSE of 5.801. The Average method reaches an accuracy of 5.780. Finally, the Moving average method reaches an accuracy of 5.703.

Unlike the results obtained in time series 1, in this case, Shallow-MLP Model 4 is not the most accurate model. So, despite that it has a huge number of weights and neurons to process, this model only reaches a Test RMSE of 4.209. Also, the prediction model based on ANN with the lowest prediction precision is the Deep-MLP Model 4. This model reaches a Test RMSE of 4.709. But, the main difference of this model is that it uses fewer weights compared to Shallow-MLP Model 3.

The CNN Model 1 is the most accurate prediction model for time series 2. This model reaches a Test RMSE of 3.640. Also, the biggest advantage of this model is that it uses a smaller amount of weights than the other two models, so it needs to adjust 478 weights during training. Finally, CNN Model 2 is the model with the smallest amount of weights of all other models. While this model is not as accurate as the CNN Model 1, this model only uses 313 weights. This means a great reduction of computational cost at the moment of adjusting weights during training.

| **Time Series 3** | | | | |
|---|---|---|---|---|
| | **Number of weights** | **Train RMSE** | **Test RMSE** | **Iterations** |
| **Naïve method** | ————- | ————- | 25.21 | ————- |
| **Average method** | ————- | ————- | 24.18 | ————- |
| **Moving average method** | ————- | ————- | 24.58 | ————- |
| **Shallow-MLP Model 3** | 17000 | 2.29 | 23.48 | 200 |
| **Deep-MLP Model 4** | 570 | 5.94 | 21.18 | 250 |
| **CNN Model 1** | 478 | 4.36 | 19.97 | 200 |
| **CNN Model 2** | 313 | 4.77 | 20.22 | 250 |

Table 8.3: Comparison of prediction models using Time Series 3

In table 8.3, the prediction results of the time series 3 are shown. As we already know, due to the irregular and random behavior of this time series, the prediction models used in this project reach a very low precision. Regardless of this, we will analyze the performance of these models anyway using the comparison metrics already mentioned.

First, we can see that the three classical quantitative prediction methods reach the lowest prediction accuracy. So, the Naïve prediction method reaches a prediction accuracy Test RMSE of 25.21. The Average method reaches an accuracy of 24.18. Finally, the Moving average method reaches an accuracy of 24.58.
Regarding prediction models based on ANN, the model with the best prediction accuracy is the CNN Model 1. This model reaches a Test RMSE of 19.97, and to reach that accuracy, the model needs to adjust 478 weights. The model with the second best prediction accuracy is CNN Model 2. This model is not so accurate like the previous model but uses less weights because the model needs to adjust 313 weights during training. Finally, the models with the worst prediction precisions are Deep-MLP and Shallow MLP based models, these models reach a Test RMSE of 21.18 and 23.48 respectively.
As we can see, in this time series example, the models that reach high accuracy are CNN Based Models. So, despite Shallow-MLP and Deep MLP based models have many weights, they do not exceed the prediction accuracy of CNN Based models.

# Chapter 9

# Conclusions

In this project, a CNN architecture, usually dedicated to image processing, is used to do a novel task: time series prediction. Our specific objective was to predict weekly sales of an Ecuadorian pharmacy franchise by using CNN.
To analyze the effectiveness of this CNN Architecture, two different prediction architectures were also implemented: Shallow-MLP and Deep-MLP. The Shallow-MLP prediction method is formed by an artificial neural network of only one hidden layer. Deep-MLP prediction method is formed by an artificial neural network of more than one hidden layer. Also, we implement three classical quantitative prediction methods: Naïve, Average, and Moving Average, to compare them with the proposed model.
For experimental purposes, three time series examples were used to do predictions. So, the first time series example represents a case in which the prediction methods reach high accuracy. The second time series example represents a case in which the prediction methods reach average accuracy. Finally, the third time series example represents a case in which the prediction methods reach low accuracy.

Once the prediction models and time series examples used have been clearly defined, the next step was generating the prediction of these time series, using each one of these prediction models. After that, we select the most accurate model of each one of the three architectures. Finally, we compared the prediction results between previously selected models. For this, three comparison metrics were used: prediction accuracy, number of weights, and number of iterations.
We showed that three classical quantitative prediction methods reach low accuracy compared with the ANN-based prediction methods. So, we showed that Shallow MLP and Deep MLP prediction architectures reach high prediction accuracy, but they use a lot of weights in their structures. For example, the most accurate Shallow-MLP prediction model adjusts during training a total amount of 17000 weights. In the case of Deep-MLP prediction method, the most accurate model adjusts during training a total amount of 570 weights.
Then, we proved that the CNN Based Prediction Models reach a high accuracy using less weights amount in their structure. So, using less than 500 weights, the CNN prediction models reach higher precision than Deep-MLP prediction models, and in most cases also

reach higher precision than Shallow-MLP prediction models. This means that the training phase of CNN Prediction methods has a very low computational cost, compared to the other two prediction methods. This fact is important when quick predictions for thousands of products are required, as in a typical pharmacy franchise.
In conclusion, in this work, a novel and effective approach for time series prediction was presented, producing successfully prediction sales for real pharmaceutical products.

# Chapter 10

# Future works

- It is possible to extend this project to different data sets, and not just limit to pharmaceutical data sales. In this way, this CNN prediction method can be used in more fields. So, this method can be used to predict: daily temperature of a city, the daily number of dead in a specific country, the monthly number of birth in a city, daily measure of stock market prices, etc. So, we could study the behavior of this prediction method in other different time series.

- Another recommendation for future work is to combine this prediction architecture CNN based, with other different prediction methods. For example, a good idea could be to combine this method with the LSTM prediction method, which is one of the most important prediction methods nowadays. Also, another example could be to combine this method with other deep learning techniques like Auto-encoders. In this way, we could be able to reduce more effectively the vanishing gradient problem, which affects very deep architectures.

- Another recommendation for future work of this project is to change the initial predefined hyper-parameters configuration of the architectures and evaluate the error descent during the training phase. So, changing the predefined hyper-parameters could be possible to find a better configuration of the architectures. In this way, it will be possible to improve both the accuracy and efficiency of the prediction architecture.

- Although we only use 1D Convolutional Filters in this project, another recommendation is to implement 2D Convolutional Filters. So, we could combine a subset of time series forming a 2D matrix like an input image. In this way, we could apply a 2D Convolutional Filter to this new input and also find hidden patterns between many time series. This technique would be very useful when two or more time series are very related to each one. For example, there are many pharmaceutical products in the data set used in this work which usually are bought together. This implies an important relationship that can be analyzed through 2D Convolutional Filters.

- Due to the high parallelization of Convolutional Neural Networks in general, a good idea for future work is to implement this method using parallel computing. So, we could apply each convolutional filter of each layer, in a parallel form. In this way, the training phase of the prediction architecture could be very fast.

# References

[1] R. McCleary, R. A. Hay, E. E. Meidinger, and D. McDowall, *Applied time series analysis for the social sciences.* Sage Publications Beverly Hills, CA, 1980.

[2] W. W. Wei, "Time series analysis," in *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*, 2006.

[3] J. D. Cryer and N. Kellet, *Time series analysis.* Springer, 1991.

[4] C. Chatfield, *The analysis of time series: an introduction.* Chapman and Hall/CRC, 2003.

[5] M. H. Hassoun *et al.*, *Fundamentals of artificial neural networks.* MIT press, 1995.

[6] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[7] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting.* springer, 2016.

[8] J. D. Hamilton, *Time series analysis.* Princeton university press Princeton, NJ, 1994, vol. 2.

[9] R. S. Tsay, "Financial time series," *Wiley StatsRef: Statistics Reference Online*, pp. 1–23, 2014.

[10] J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," *Physical review letters*, vol. 59, no. 8, p. 845, 1987.

[11] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications: with R examples.* Springer, 2017.

[12] H. Madsen, *Time series analysis.* Chapman and Hall/CRC, 2007.

[13] J. M. Zurada, *Introduction to artificial neural systems.* West publishing company St. Paul, 1992, vol. 8.

[14] H. White, *Artificial neural networks: approximation and learning theory.* Blackwell Publishers, Inc., 1992.

[15] B. Yegnanarayana, *Artificial neural networks.* PHI Learning Pvt. Ltd., 2009.

[16] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of microbiological methods*, vol. 43, no. 1, pp. 3–31, 2000.

[17] W. S. Sarle, "Neural networks and statistical models," 1994.

[18] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[19] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artificial intelligence*, vol. 70, no. 1-2, pp. 119–165, 1994.

[20] G. Daniel, *Principles of artificial neural networks.* World Scientific, 2013, vol. 7.

[21] J. K. Paik and A. K. Katsaggelos, "Image restoration using a modified hopfield network," *IEEE Transactions on image processing*, vol. 1, no. 1, pp. 49–63, 1992.

[22] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd International Conference on Machine learning (ICML-05)*, 2005, pp. 89–96.

[23] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade.* Springer, 2012, pp. 421–436.

[24] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean, "Boosting algorithms as gradient descent," in *Advances in neural information processing systems*, 2000, pp. 512–518.

[25] Y. Bengio, P. Simard, P. Frasconi *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[26] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[27] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht, "Gradient descent only converges to minimizers," in *Conference on learning theory*, 2016, pp. 1246–1257.

[28] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in neural information processing systems*, 2013, pp. 315–323.

[29] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[30] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the twenty-first international conference on Machine learning.* ACM, 2004, p. 116.

[31] L. C. Baird III and A. W. Moore, "Gradient descent for general reinforcement learning," in *Advances in neural information processing systems*, 1999, pp. 968–974.

[32] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception.* Elsevier, 1992, pp. 65–93.

[33] B. J. Wythoff, "Backpropagation neural networks: a tutorial," *Chemometrics and Intelligent Laboratory Systems*, vol. 18, no. 2, pp. 115–155, 1993.

[34] F. S. Wong, "Time series forecasting using backpropagation neural networks," *Neurocomputing*, vol. 2, no. 4, pp. 147–159, 1991.

[35] A. T. Goh, "Back-propagation neural networks for modeling complex systems," *Artificial Intelligence in Engineering*, vol. 9, no. 3, pp. 143–151, 1995.

[36] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2015, pp. 4580–4584.

[37] A. G. Howard, "Some improvements on deep convolutional neural network based image classification," *arXiv preprint arXiv:1312.5402*, 2013.

[38] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *2013 IEEE international conference on acoustics, speech and signal processing.* IEEE, 2013, pp. 7398–7402.

[39] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[41] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[42] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[43] P. Y. Simard, D. Steinkraus, J. C. Platt *et al.*, "Best practices for convolutional neural networks applied to visual document analysis." in *Icdar*, vol. 3, no. 2003, 2003.

[44] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *2013 IEEE international conference on acoustics, speech and signal processing.* IEEE, 2013, pp. 8614–8618.

[45] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[46] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," in *Advances in neural information processing systems*, 2014, pp. 2042–2050.

[47] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012).* IEEE, 2012, pp. 3304–3308.

[48] F. M. Thiesing and O. Vornberger, "Sales forecasting using neural networks," in *Proceedings of International Conference on Neural Networks (ICNN'97)*, vol. 4. IEEE, 1997, pp. 2125–2128.

[49] M. Khashei and M. Bijari, "An artificial neural network (p, d, q) model for timeseries forecasting," *Expert Systems with applications*, vol. 37, no. 1, pp. 479–489, 2010.

[50] J. Wang and J. Wang, "Forecasting energy market indices with recurrent neural networks: Case study of crude oil price fluctuations," *Energy*, vol. 102, pp. 365–374, 2016.

[51] O. Chang, I. Naranjo, and C. Guerron, "A deep learning algorithm to forecast sales of pharmaceutical products," 10 2017.

[52] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, vol. 1. IEEE, 2017, pp. 7–12.

[53] K. Bandara, P. Shi, C. Bergmeir, H. Hewamalage, Q. Tran, and B. Seaman, "Sales demand forecast in e-commerce using a long short-term memory neural network methodology," *arXiv preprint arXiv:1901.04028*, 2019.