REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ DES MATHEMATIQUES ET DE l'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité: Génie Logiciel

Par:

HADHBI ALI
KACEM MOHAMED

Sur le thème

Convolutional Neural Networks for Time Series Forecasting

Soutenu publiquement le 19 / 09 / 2022 à Tiaret devant le jury composé de :

Mr. DJAFRI Laouni	MCA	Président
Mr. BOUDAA Boudjemaa	MCA	Encadrant
Mme. LAAREDJ Zohra	MAA	Examinatrice

2021-2022

Abstract

Time series forecasting occurs when you make scientific predictions based on historical time stamped data. It involves building models through historical analysis and using them to make observations and drive future Strategic decision-making. Their main specificities compared to the most common areas of machine learning are their dependence over time and their seasonal behaviors that can appear in their evolution. In the literature, statistical models are widely used for time series forecasting. However, many complex models or approaches can be very useful in some cases. Generalized Autoregressive Conditional Heteroskedasticity (GARCH), Bayesian models and ARIMA vectors (VAR) are just a few examples. There are also even time series models borrowed from deep learning. Deep learning methods do offer a lot of promise for time series forecasting, specifically the automatic learning of temporal dependence and the automatic handling of temporal structures like trends and seasonality. Multilayer Perceptrons, Convolutional Neural Networks (CNN), or Recurrent Neural Network (RNN) with LSTM or GRU cells can be successfully employed for time series forecasting issues. Through this work, we aim to develop a CNN model for time series forecasting. The proposed model will be experimented and evaluated on real-world datasets with the known metrics in this field.
Key-words: Time Series Forecasting, CNN, GRU, Single-step, Multi-step, Univariate, Multivariate.

Résumé

Les prévisions de séries chronologiques se produisent lorsque vous faites des prédictions scientifiques basées sur des données historiques horodatées. Cela implique de construire des modèles grâce à une analyse historique et de les utiliser pour faire des observations et orienter la prise de décision stratégique future. Leurs principales spécificités par rapport aux domaines les plus courants du machine Learning sont leur dépendance dans le temps et leurs comportements saisonniers qui peuvent apparaître dans leur évolution. Dans la littérature, les modèles statistiques sont largement utilisés pour la prévision des séries temporelles. Cependant, de nombreux modèles ou approches complexes peuvent être très utiles dans certains cas. L'hétéroscédasticité conditionnelle autorégressive généralisée (GARCH), les modèles bayésiens et les vecteurs ARIMA (VAR) ne sont que quelques exemples. Il existe même des modèles de séries chronologiques empruntés à l'apprentissage en profondeur. Les méthodes d'apprentissage en profondeur offrent beaucoup de promesses pour la prévision des séries chronologiques, en particulier l'apprentissage automatique de la dépendance temporelle et la gestion automatique des structures temporelles telles que les tendances et la saisonnalité. Les perceptrons multicouches, les réseaux de neurones convolutifs (CNN) ou les réseaux de neurones récurrents (RNN) avec des cellules LSTM ou GRU peuvent être utilisés avec succès pour les problèmes de prévision de séries chronologiques. A travers ce travail, nous visons à développer un modèle CNN pour la prévision de séries temporelles. Le modèle proposé sera expérimenté et évalué sur des ensembles de données du monde réel avec les métriques connues dans ce domaine.
Mots-clés: Prévision de séries chronologiques, CNN, GRU, Single-step, Multi-step, Univariée, Multivariée.

Acknowledgment

First of all, praise be to God Almighty and thanked Him for His blessings throughout our research work, which allowed us to complete it successfully. We have tested your guidance day in and day out; you are the one who allows us to finish our studies. We will continue to trust you for our future.

We would like to express our gratitude and special thanks to our supervisor ***Mr. Boudaa Boudjemaa*** *who made this work possible and whose office door was always open whenever we encountered a problem or had any question about our research or writing, his guidance and advice carried us through all stages of writing our project. It was a great honor and privilege that we work under his supervision. We have been inspired by his dynamism, vision, honesty and motivation. Thank you sir for your patience with us.*

We would also like to thank our committee members for making our defense an enjoyable moment, and for your great comments and suggestions, thank you. Finally, we appreciate all the support we received from our family who have been the source of our inspiration.

Thank you all.

Dedications

This message is dedicated to the Face of God, my Creator, Master, Teacher, and Great Messenger, Muhammad (peace be upon him), who taught us the purpose of life. I dedicate my graduation to my late mother, God willing, she is the one who taught me that love has no age and that giving has no limits The female who shortens all women My dear mother, the candle that burned to light my life path For the soul of my departed mother From our world that remains in my eternal heart in my world to sacred love The pure angel, to whom God made heaven under her feet, to the one who showered me with the flood of her tenderness, to the one who starved to be sated, stayed up to sleep, tried to rest, cried to laugh, and watered me from the source of her tenderness and sincerity, to the one who raised me young and advised me big, the comfort of my eyes and my heart, my dear mother, may God have mercy on you.

I dedicate my graduation to my dear father, who worked hard and made every effort and supported me financially and morally and with all he could, may God prolong his life and grant him health and wellness.

I also dedicate this graduation to all my family, my sisters, my brothers, their wives, their daughters and their children who have been credited with removing many obstacles and difficulties from my path. I love you all.

I also do not forget to give a big thank you to all my friends with whom I have studied over the last five years. I should not forget my teachers, who had the greatest role in supporting me and providing me with valuable information, praying to God Almighty to prolong your life and bless you with bounties.

Hadhbi Ali

I dedicate this thesis to:

All my family members. Dad may Allah bless his soul, Mom may Allah grant her health and a long life.

My teachers, colleagues, friends and those who helped me to achieve this work.

Kacem Mohamed

Contents

List of Figures

List of Figures

List of Tables

List of Abbreviations

AI: Artificial Intelligence

AN: Artificial Neurons

ANN: Artificial Neural Networks

AR: auto-regression

ARIMA: Auto Regressive Integrated Moving Average

ARMA: Auto Regressive Moving Average

ARMAV: Auto Regressive Moving Average Vector

CNN: Convolutional Neural Networks

ConvLSTM: Convolutional Long Short-Term Memory

DBN: Deep Belief Networks

DL: Deep Learning

DNN: Deep Neural Networks

ESN: Echo State Setwork

ETS: Exponential Smoothing

FNN: Feed-forward Neural Network

GARCH: Generalized Auto Regressive Conditional Heteroskedastic

GRU: Gated Recurrent Unit

HWES: Holt Winter's Exponential Smoothing

KNN: K-Nearest Neighbors

LR: Linear regression

LSTM: Long Short Term Memory

MAE: Mean Absolute Error

MAPE: Mean Absolute Percentage Error

List of Abbreviations

MSE: Mean Squared Error

ML: Machine Learning

MLP: Multi-Layer Perceptron

NN: Neural Network

RC: Reservoir Computing

RNN: Recurrent Neural Networks

SES: Simple Exponential Smoothing

SMA: Simple Moving Average

SVM: Support Vector Machine

SVR: Support Vector Regression

VAR: Vector Auto Regression

General Introduction

1. Background

Time series data and its analysis are increasingly important due to the massive production of such data through, for example, the internet of things, the digitalization of healthcare, and the rise of smart cities. In the coming years we can expect the quantity, quality, and importance of time series data to grow rapidly. Indicators of production and efficiency in markets have long provided interesting data to study from a time series analysis. Most interesting and urgent has been the question of forecasting future economic states based on the past. Such forecasts aren't merely useful for making money they also help promote prosperity and avert social catastrophes.

In simple terms, time-series data refers to a consistent stream of data sets over the course of a period of time. Analyzing this type of data has become a recent area of focus in artificial intelligence, as accurate forecasting is becoming increasingly vital across all kinds of industries in order to make more informed decisions.

Machine Learning has been proven powerful in imaging, natural language, and speech because of huge annotated datasets available. On the other hand, time series problems usually do not have big annotated datasets. Also, the data from different domains exhibit considerable variations in important properties and features, temporal scales, and dimensionality. Further, time-series analysis requires the algorithm to learn time-dependent patterns within and across multiple modalities, unlike images or speech. Time series analysis mostly includes clustering, classification, anomaly detection, and forecasting each of which is uniquely useful to the business.

2. Problem statement

Traditionally, time series forecasting has been dominated by linear methods because they are well understood and effective on many simpler forecasting problems, however deep learning neural networks are able to automatically learn arbitrary complex mappings from inputs to outputs and support multiple inputs and outputs.

Accordingly, are deep learning models a viable option for time series forecasting? if so, how well it performs compared to traditional methods?

3. Approach

To answer the previous question, a CNN for time series forecasting is developed. The proposed model will be experimented and evaluated on real-world datasets with the known metrics in this field.

4. Outline

This thesis is divided into three chapters and a general introduction, which includes the following sections: background, problem statement, approach, and outline, and a general conclusion.

Chapters one and two are theoretical; chapter three will be the application section, in which the steps of implementation and the results will be presented; and finally, we will draw a broad conclusion based on the results and discussion from chapter three. Each chapter's structure is as follows:

- *Chapter one: Time series forecasting*
  This chapter introduces time series in general and time series forecasting in particular, with an overview of the various forecasting techniques and a discussion of real-world study cases that employed time series forecasting.

- *Chapter two : Convolutional Neural Networks (CNN)*
  This chapter covers the fundamentals of artificial intelligence, machine learning and its different types. And the fundamental concepts of deep learning, with focusing on Convolutional Neural Networks, their architecture and how it functions.

- *Chapter three: Convolutional Neural Networks for time series forecasting*
  In this chapter, we implement a CNN model for time series forecasting and measure its performance based on the metrics and baselines. To study whether CNNs are a viable option for time series forecasting.

CHAPTER 1:

Time Series Forecasting

CHAPTER 1: Time Series Forecasting

1.1 Introduction

A time series is a collection of observations in chronological order. These could be daily stock closing prices, weekly inventory figures, annual sales, or countless other things.

Forecasting the future values of an observed time series is an important problem in many areas, including economics, production planning, sales forecasting and stock control.

Forecasting is valuable to businesses because it gives the ability to make informed business decisions and develop data-driven strategies. Based on the state of the market today and forecasts for the future, financial and operational decisions are taken.

1.2 Time Series

1.2.1 Time Series Presentation

Time series forecasting has always been a very important area of research in many fields because many different types of data are stored as time series. For example, we can find many time series data in medicine, weather forecasting, biology, supply chain management and stock price forecasting. Due to the increasing availability of data and computing power in recent years, deep learning has become an essential part of the new generation of time series prediction models, with excellent results [1].

Unlike traditional machine learning models like auto-regression (AR) or exponential smoothing, which need feature engineering and some parameter optimization with domain expertise in mind; deep learning models learn features and dynamics entirely from the data. They can speed up the data preparation process and understand more complicated data patterns more thoroughly as a result of this. In recent years, a great number of innovative architectures have been built as various time series problems have been studied in many disciplines, making the development of new bespoke network components easier and faster. Therefore, we will proceed in this research by using one of the methods of deep learning in predicting time series [2].

Another difficulty or tradeoff is related to the number of data points in the time series, more data from the past increase the precision and the chance of detailed analysis.
However, while increasing the precision property of the data, we also increase the risk that the model cannot handle the data processing [3].

1.2.1.1 Definitions

Time series analysis is a method for studying a collection of data points over a period. Instead of capturing data points intermittently or arbitrarily, time series analyzers record data points at constant intervals over a predetermined length of time. This form of analysis, however, is more than just gathering data over time.

The ability to depict how variables change over time distinguishes time series data from other types of data. It provides an additional source of data as well as a predetermined order of data dependencies. To maintain consistency and dependability, time series analysis often requires a high number of data points. A large data set guarantees a representative sample size.

To achieve consistency and dependability, time series analysis often requires a large number of data points. A large data collection ensures that your sample size is representative and that your analysis can cut through noisy data. It also guarantees that any discovered trends or patterns are not outliers and that seasonal variation is taken into account. Time series data can also be used for forecasting or anticipating future data based on previous data. The ability to depict how variables change over time distinguishes time series data from other types of data. It provides an additional source of data as well as a predetermined order of data dependencies [3].

1.2.1.2 Examples

There is almost an endless supply of time series forecasting problems. Below are some examples from a range of industries to make the notions of time series analysis and forecasting more concrete.

- Forecasting the corn yield in tons by state each year.
- Forecasting whether an electroencephalography trace in seconds indicates a patient is having a seizure or not.
- Forecasting the closing price of a stock each day.
- Forecasting the birth rate at all hospitals in a city each year.
- Forecasting product sales in units sold each day for a store.
- Forecasting the number of passengers through a train station each day.
- Forecasting unemployment for a state each quarter.
- Forecasting utilization demand on a server each hour.
- Forecasting the size of the rabbit population in a state each breeding season [4].

1.2.2 Time Series Data and Components

Time series data is a set of observations obtained through repeated measurements over time. Time series metrics refer to a piece of data that is tracked with an increase in time. For example, a metric can indicate how much inventory was sold in a store from one day to the next.

Time series data is everywhere, because time is a component of everything that can be observed. As our world increasingly uses tools, sensors and systems are constantly releasing a relentless stream of time-series data. Operations and other business decisions often depend on accurate time-series forecasts. These time series usually consist of trend-cycle, seasonal, and irregular components. Existing methodologies attempt to first identify and then extrapolate these components to produce forecasts. This data has many Components, we will mention them [5].

1.2.2.1 Components

The various reasons or the forces, which affect the values of an observation in a time series, are the components of a time series. The four categories of the components of time series are:

- Trend
- Seasonality
- Cycle
- Irregular

1.2.2.1.1 Trend Component

A trend time series moves in a simple linear fashion. The trend shows the general tendency of the data to increase or decrease during a long period of time. And it is obtained by ignoring any short-term effects such as seasonal changes or noise. The increase or decrease does not need to be in the same direction throughout the given period of time [6]. For example, see the upward and downward trend in the plot of Japan’s population.

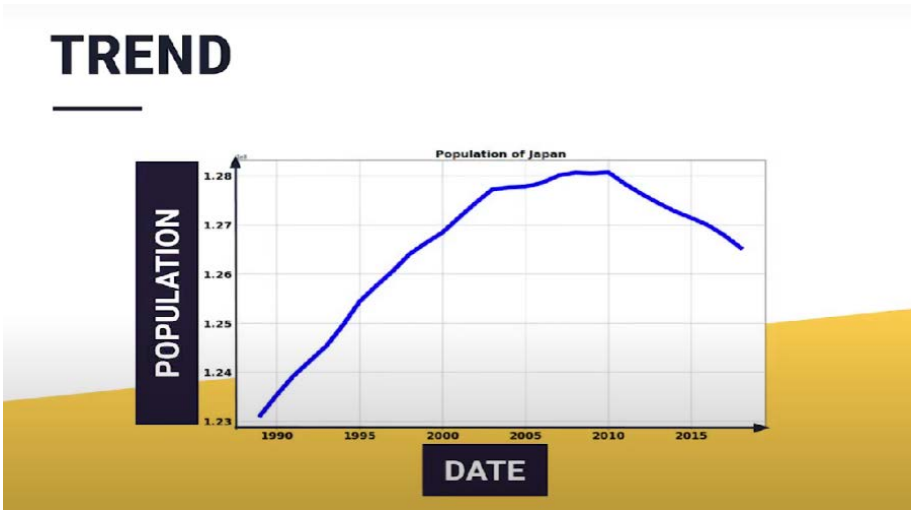

Figure 1.1 The upward and downward trend in the plot of Japan's population *[8]*.

1.2.2.1.2 Seasonality Component

A seasonal pattern exists when there are a series of nuances depending on seasonal factors (for example, quarter, month, or day of the week). Seasonality is always of a fixed, known period. Hence, seasonal time series are sometimes called periodic time series i.e. they repeat throughout the duration of the time series [7].

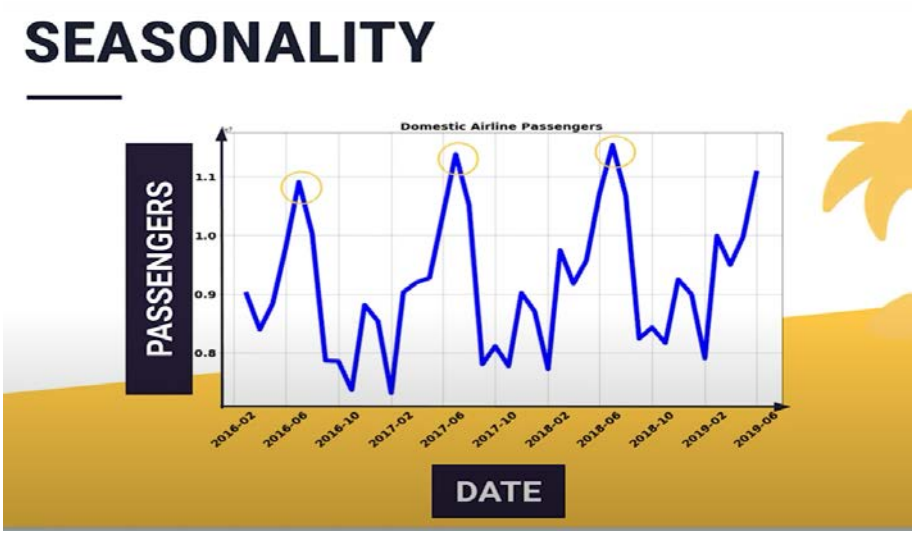

Figure 1.2 Predictions of Airline Passengers monthly *[8]*.

1.2.2.1.3 Cyclical Component

Cycles occur when a time series follows a non-seasonal, up-and-down pattern. Cycles are hard to predict because they do not occur in predictable time intervals. Atypical example is the business cycle, comprising the phases of recovery, prosperity, recession and depression [8].

Figure 1.3 Monthly housing sales *[8]*.

1.2.2.1.4 Irregular Component

Irregular variation in time series data occurs due to uncontrollable and unpredictable events, such as earthquakes, wars, floods, famines, and so on. This plot of the Nikkei 225 stock index shows a plunge in value, which coincided with the earthquake and tsunami in March 2011 [8].

Figure 1.4 Nikkei company stock price *[8]*.

1.2.2.2 Understanding time series

Typically, a time series is represented as a graph with each data point modelled on the x- and y-axes. The time series data is a trend time series if the plot shows the highest or lowest values over an extended period. Seasonal time series are those that repeat throughout time, measured in years, and cyclical time series are those that change often over time, as seen by the depiction. (Figure1.5) provides an illustration of the components of a raw time series as well as how a time series appears after seasonal and cyclical tendencies have been eliminated, or what residual (remainder) data is:

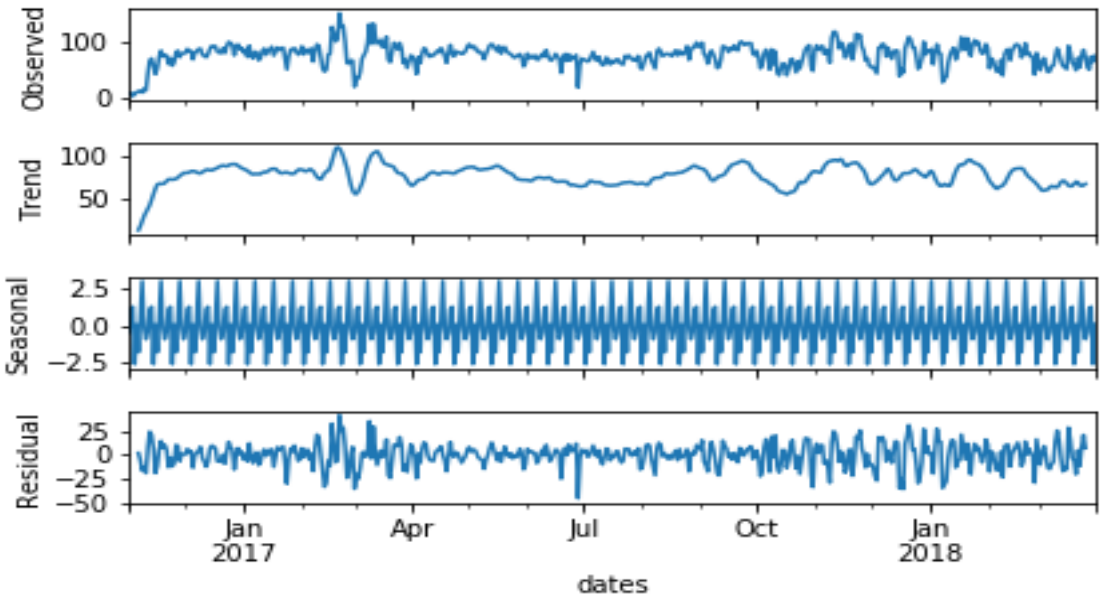

Figure 1.5 Components of time series *[75]*

1.2.3 Time Series Types

A single variable measured over time is referred to as a univariate time series. Univariate means one variate or one variable. Multiple variables measured over time is referred to as a multivariate time series: multiple variates or multiple variables:

- Univariate: One variable measured over time.
- Multivariate: Multiple variables measured over time [4].

When researching this field, any time series forecasting is divided into two groups: Univariate and multivariate. In the following section, we will discuss univariate time series and multivariate time series.

1.2.3.1 Univariate Time Series

A univariate time series refers to time series that has only one observation recorded sequentially over equal time increments [9].

Despite the fact that a univariate time series data set is typically presented as a single column of numbers, time is an implicit variable in the time series. The term variable, or index, does not need to be specified if the data are evenly spaced. When graphing a series, the time variable may be utilized explicitly. It is not, however, employed in the time series model, For example, if you are tracking hourly temperature values for a given region and want to forecast the future temperature using historical temperatures, this is univariate time series forecasting. (Figure1.6) is a simple example of temperature values with time measured by each hour [10].

Time	Temperature
5:00 am	59 °F
6:00 am	59 °F
7:00 am	58 °F
8:00 am	58 °F
9:00 am	60 °F
10:00 am	62 °F

Figure 1.6 Univariate time series (temperature values)

1.2.3.2 Multivariate Time Series

Multivariate time series has more than one time-dependent variable. Each variable depends not only on its past values but also has some dependency on other variables. This dependency is used for forecasting future values. along with the temperature values. In this case, there are multiple variables to be considered to optimally predict temperature. A series like this would fall under the category of multivariate time series. The following (Figure1.7) shows example for a multi-variate time series [11]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-09-17	465.864014	468.174011	452.421997	457.334015	457.334015	2.105680e+07
1	2014-09-18	456.859985	456.859985	413.104004	424.440002	424.440002	3.448320e+07
2	2014-09-19	424.102997	427.834991	384.532013	394.795990	394.795990	3.791970e+07
3	2014-09-20	394.673004	423.295990	389.882996	408.903992	408.903992	3.686360e+07
4	2014-09-21	408.084991	412.425995	393.181000	398.821014	398.821014	2.658010e+07
...	...	...	...	...	...	...	...
1927	2019-12-27	7238.141113	7363.529297	7189.934082	7290.088379	7290.088379	2.277736e+10
1928	2019-12-28	7289.031250	7399.041016	7286.905273	7317.990234	7317.990234	2.136567e+10
1929	2019-12-29	7317.647461	7513.948242	7279.865234	7422.652832	7422.652832	2.244526e+10
1930	2019-12-30	7420.272949	7454.824219	7276.308105	7292.995117	7292.995117	2.287413e+10
1931	2019-12-31	7294.438965	7335.290039	7169.777832	7193.599121	7193.599121	2.116795e+10

Figure 1.7 Multivariate time series (Bitcoin daily price index) *[76]*

1.2.4 Stationary Time Series

The major goal of this research is to learn about stationary time series. A time series is referred as stationary if its statistical properties do not change over time. In other words, it has a constant mean and variance and it is independent of time [12]. For example, in (Figure 1.8) the strong cycles in series might appear to make it non-stationary. But these cycles are aperiodic, they are caused when the lynx population becomes too large for the available feed, so that they stop breeding and the population falls to low numbers, then the regeneration of their food sources allows the population to grow again, and so on. In the long-term, the timing of these cycles is not predictable. Hence the series is stationary.

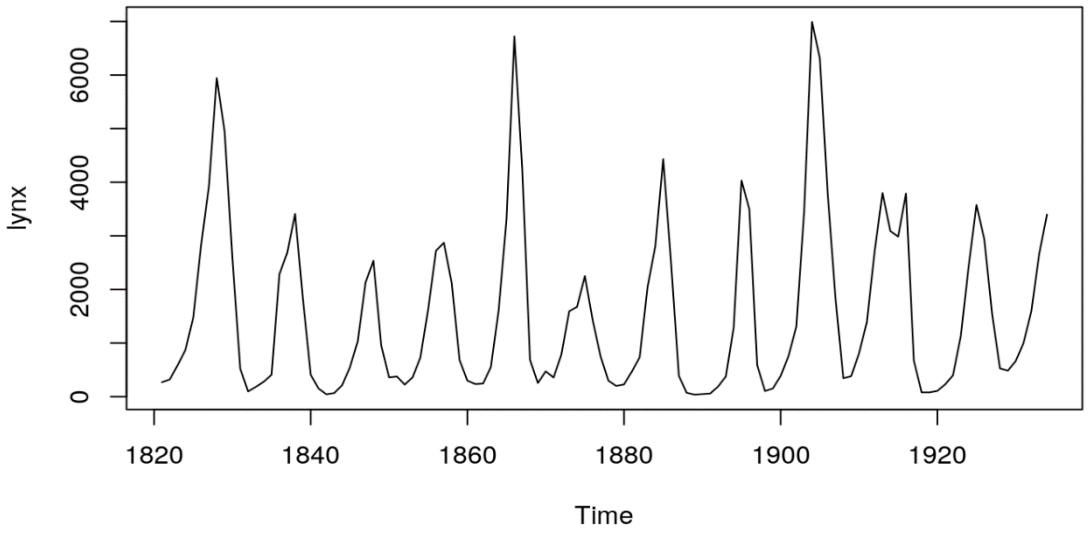

Figure 1.8 Annual total of lynx trapped in the McKenzie River district of north-west Canada *[77]*

One of the most important concepts in time series analysis is stationarity. Stationarity occurs when a shift in time doesn't change the shape of the distribution of your data. This is in contrast to non-stationary data, where data points have means, variances and covariance that change over time. This means that the data have trends, cycles, random walks or combinations of the three. Generally, in forecasting, non-stationary data are unpredictable and cannot be modeled [12]. Here is an example of static and non-constant data (Figure 1.9).

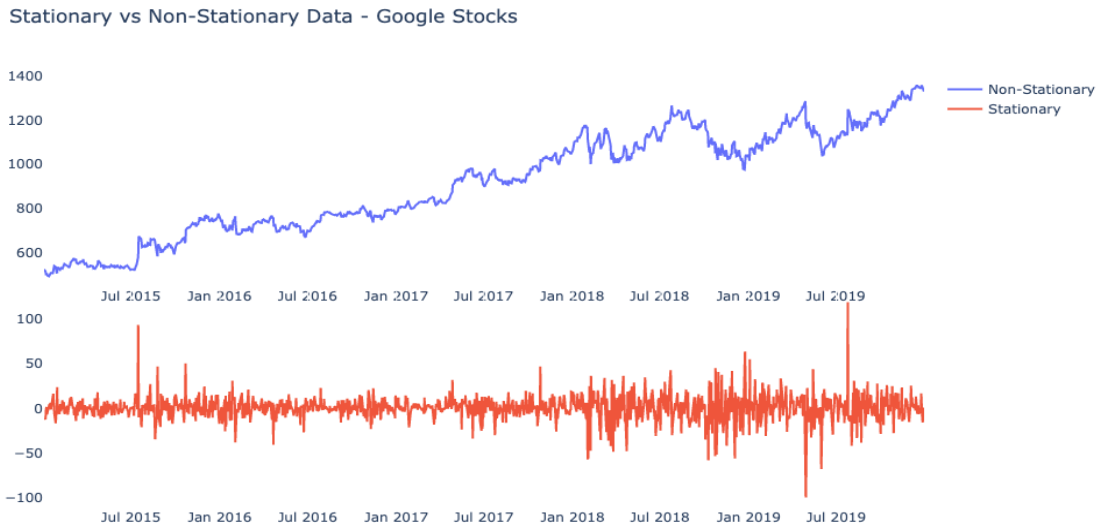

Figure 1.9 Stationary and non-stationary time series *[78]*

1.3 Time Series Forecasting

An often-heard motivation for time series analysis is the prediction of future observations in the series. This is an ambitious goal, because time series forecasting relies on extrapolation, and is generally based on the assumption that past and present characteristics of the series continue. It seems obvious that good forecasting results require a very good comprehension of a series' properties, be it in a more descriptive sense, or in the sense of a fitted model [13].

Time series forecasting is the process of predicting the data of future time steps. It can be thought of as a mechanism to process or analyze the historical data and use that learned information to make predictions about the future, time series forecasting can be classed into two types:

1.3.1 One-step ahead prediction

Always one step ahead Based on the corresponding forecast equation, every model supports one-step ahead forecasts. During the model estimate process, one-step ahead forecasts are required to compute model errors. For each data point, one-step ahead forecasts are computed sequentially using computed level and trend states for the current point, as well as seasonal states for the previous seasonal period. The forecast error is calculated by subtracting the forecast value (estimated at the preceding point) from the current observed value. The average value of absolute squared forecast errors is used to calculate overall model error, which is used to estimate the model. A greater model fit is associated with fewer mistakes. Statistical details display accuracy measures that provide numerous model summaries of one-step ahead forecast errors. This method is concerned only with predicting the next time step given the time series, for example predicting the value of a sensor in the 11th second given 10 seconds of past data [14].

1.3.2 Multi-step ahead prediction

Many scholars have focused on time series forecasting in recent years, but most of them focus on one-step-ahead predictions, which are not useful in everyday life. There are two primary ways for multi-step ahead prediction: iterate-based and direct-based methods. The output of time step t is one of the inputs of time step t+1 in the iterate-based approach. The biggest downside of this strategy is that after a few time steps, the mistake would have accumulated to a very large value. The direct-based method creates different models from different training instances; one model predicts the next time step, another model predicts the next two time steps, and so on. The biggest downside of this procedure is that it necessitates an excessive amount of time [15].

This, as the name suggests, is concerned with predicting a sequence of time steps given a time series, for example forecasting the weather for the next 10 days given the past data. Future studies on estimating the future values of several variables. It should be noted that the effective role of the Tab process stems from the accuracy of the prediction results, which are based mainly on the process of correct construction of the model generating these results. Its effectiveness is determined by achieving a set of statistical hypotheses and choosing a series of tests based on The extent of the relationship, the problem, the structure of the studied data [15].

1.4 Time Series Forecasting Models

Statistical time series models are a family of models that have been traditionally used a lot in many domains of forecasting. They are strongly based on temporal variation inside a time series and they work well with univariate time series [16]. Some advanced options exist to add external variables into the models as well. In the other hand, machine learning and deep learning models are able to automatically learn arbitrary complex mappings from inputs to outputs and support multiple inputs and outputs [4].

1.4.1 Statistical Models

In this section, we will talk about the use of statistical models in predicting time series. In particular, we divide them into two categories. Traditional statistical models such as:

ARIMA (Autoregressive Integrated Moving Average) and its family: AR, MA, ARMA, SARIMA and SARIMAX. Vector auto regression (VAR) and its derivatives VARMA and VARMAX, and Smoothing: Simple moving average (SMA), simple exponential smoothing (SES), double exponential smoothing (EES), Holt winter is exponential smoothing (HWES). GARCH (Generalized Autoregressive Conditional Heteroscedasticity).

1.4.1.1 ARIMA

The acronym ARIMA refers to the Auto Regressive Integrated Moving Average. Box and Jenkins popularized the ARIMA model (1976). It is made up of three different statistical models. For statistical data, it employs the Autoregressive, Integrated, and Moving Average (ARIMA) model. The ARIMA Model uses Auto Regressive Integrated Moving Average (ARIMA) and Auto Regressive Moving Average (ARMA) to assess and forecast evenly spaced univariate statistic information, transmission of function data, and intercession information (ARMA). A response Time Series forecasted by an ARIMA Model is a linear mixture of its own linked past values, previous Errors, and current and past values of alternative Time Series.

The ARIMA Model seeks to explain information autocorrelation and can be applied to both stationary and non-stationary statistics [17].

1.4.1.2 Exponential Smoothing

Exponential Smoothing was developed as a result of Robert G. Brown's work as an OR it is analyst for the United States Navy. In the early 1950s, Brown extended simple Exponential Smoothing to discrete data, and the method should improve for trend and seasonality.

Time Series established the universal exponential smoothing algorithm. In the residuals of Time Series projected using the Exponential Smoothing (ETS) Method, Taylor identified first order auto-correlation. One or more stochastic models correspond to each Exponential Smoothing (ETS) approach. It also follows the Robustness condition and is used as a model for extrapolating judgments [17].

1.4.1.3 Vector Auto regression (VAR)

The vector auto regression (VAR) model is a popular, flexible, and straightforward model for multivariate time series analysis. A natural extension of the univariate autoregressive model is the dynamic multivariate autoregressive model. The VAR model has been shown to be particularly useful for forecasting and describing the dynamic behavior of economic and financial time series.

Forecasts from univariate time series models and complex theory-based simultaneous equations models are frequently outperformed. VAR model forecasts are quite flexible because they can be made conditional on the potential future paths of specified variables in the model. The VAR model is used for structural inference and policy analysis, in addition to data description and forecasting. In structural analysis, certain assumptions about the causal structure of the data are imposed, and the resulting causal impacts of unexpected shocks or innovations to specified variables on the variables in the model are summarized. These causal effects are typically summarized using impulse response functions and forecast error variance decompositions [18].

1.4.2 Machine Learning Models

Prediction methods for machine learning methods in this study include all artificial intelligence-based prediction techniques. These methods are classified into several types based on the phenomenon being predicted. The following networks are introduced to provide a broad understanding of predictive models for machine learning: Linear regression (LR), Support vector regression (SVR), and K-Nearest Neighbors (KNN).

1.4.2.1 Linear Regression

Linear regression (LR) is the most basic approach for determining the relationship between the dependent and independent variables.

Simple linear regression is used when there is only one independent variable. Having numerous independent variables, on the other hand, changes the name of the model to multiple linear

regression [19]. The linear predictor functions, whose model parameters are estimated from the data, are used to determine the relationships in linear regression. These kinds of models are known as linear models [20]. Linear regression, like other regression analyses, examines the dependent variable's conditional probability distribution given the values of independent variables This study employed the multivariate linear regression (MLR) technique, which is one of several LR model variants. Several important asymptotic and finite sample results are presented and compared with time series regression statistical properties [21].

1.4.2.2 Support Vector Regression (SVR)

The Support Vector Machine (SVM) is an elegant machine learning algorithm proposed by Cortes and Vapnik in 1995 [22]. The algorithm's basic idea is to reduce structural risk. It is a concept that achieves generalization by balancing the model's complexity against its ability to fit the training data. The SVM has a regression version that is widely used in time-series prediction and imputation [23] [24].

The Support Regression Vector (SVR) is an SVM transformation for solving nonlinear regression problems. The data is transformed for nonlinear regression problems using a nonlinear kernel function that maps the input to a high-dimensional feature space. As a result, the overall performance of the SVM regression model is dependent on the proper selection of kernel parameters [25]. The basic idea behind SVM for function approximation is to map the data into a high-dimensional feature space, then perform a linear regression in the feature space.

1.4.2.3 K-Nearest Neighbors (KNN)

The k-nearest neighbors' algorithm (KNN) is a non-parametric method for classification and regression invented by Thomas Cover [26]. The input in both regression and classification is the k closest training instances in the feature space. Whether KNN is used for classification or regression determines the outcome: In classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor [27].

In KNN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors. KNN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, normalizing the training data can improve its accuracy dramatically [28].

1.5 Deep Learning Models

Deep Learning (DL) is a machine learning subfield inspired by brain anatomy and function. Despite their capabilities, no deep machine learning approaches are limited when it comes to analyzing raw data. This means that designing a feature extractor that can convert raw data into a format suited for no deep machine learning models would require careful engineering and extensive subject experience. DL techniques are approaches for learning representations using multiple levels of representation. This is accomplished by mixing non-linear modules that shift the representation from a raw input level to a higher, more abstract one. DL techniques can learn complex functions using this composition [29].

A lot of study is done in exploring multiple DL architectures due to their resilience and improved possibilities. Deep neural networks (DNN), deep belief networks (DBN), recurrent neural networks (RNN), convolutional neural networks (CNN), and other types of neural networks (ANN) [30]. The artificial neural network is at the heart of these structures. These designs have been used in a variety of domains, including computer vision, natural language processing, machine translation, and speech recognition, where they have produced results comparable to and in some cases better than human expert performance [31]. Based on ANN researchers have developed many architecture beginning with Multi-Layer Perceptron (MLP), then the more sophisticated methods, situated bellow:

1.5.1 Multi-Layer Perceptron (MLP)

A completely connected neural network is what is referred to as a multi-layer perceptron, which is a sort of artificial neural network where the architecture is such that all of the nodes, or neurons, in one layer are connected to the neurons in the following layer. One input layer, one or more hidden perceptron layers, and three basic building blocks make up the framework of MLP. Simple distribution of the input characteristics to the top hidden layer occurs in the input layer. The features dispersed by the input layer are fed into the first hidden layer as inputs. The output of each perceptron from the preceding layer is sent into the subsequent hidden layers. The output of each perceptron from the last hidden layer is sent into the perceptron's output layer. Even that Multi-Layer Perceptron is a very basic model it is widely chosen for studies [32].

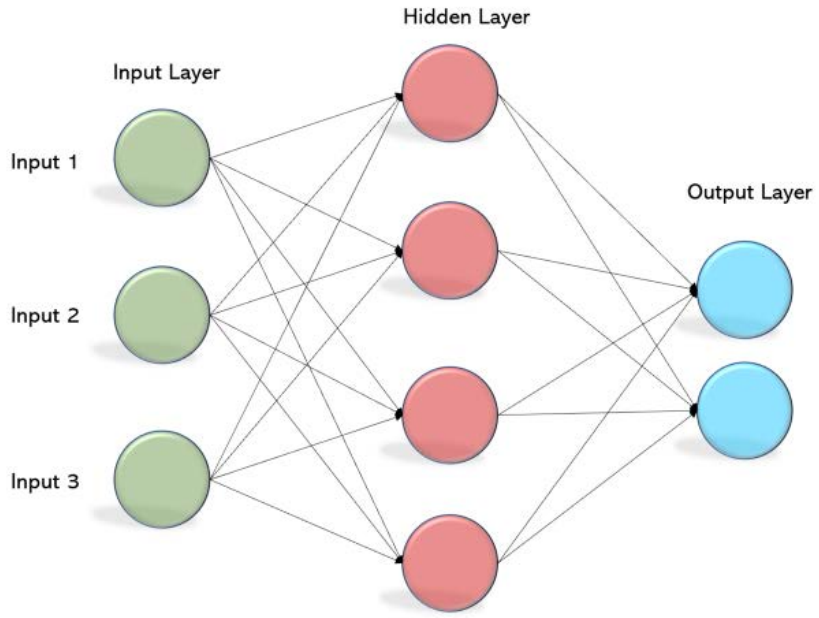

Figure 1.10 Multi-layer Perceptron (MLP) Model *[33]*

1.5.2 Recurrent Neural Network (RNN)

Recurrent neural networks (RNNs) are a type of ANN in which nodes are connected in a directed graph that follows a temporal sequence. Traditional time series models and RNNs are both capable of modeling time dependent relationships in data. In RNNs, each node in one layer is connected to every other node in the next layer via a directed, one-way connection. Every node has a real-valued activation function that changes over time, and each link (synapse) has a real-valued weight that may be changed. Input nodes receive data from outside the network, output nodes provide results, and hidden nodes modify data in route to output. RNN can be regarded not just as cyclic but also as a deep layer per time step with shared weights across time steps, as shown in the diagram. Backpropagation can be used to train the unfolded network over many time steps. The back prop algorithm is called backpropagation through time (BPTT) because we propagate through time in RNN

Despite the wonders of RNN, it suffers from couple of limitations. While training the network, when gradients are propagating back in time, all the way up to the initial layer, the gradients gothrough multiple simultaneous matrix multiplications and as a result of using Chain Rule, ifthey have values less than 1 (<1), they diminish exponentially until they become negligibleor ‘vanish’. This deems the network model impossible to learn not anything as weights will be updated. This is known as the ‘Vanishing Gradient

Problem.' Similarly, if the valuesof gradients propagating back in time are greater than 1 (>1), their values escalate and eventually destroy the model's capability to learn anything making it unstable. This is known as the 'Exploding Gradient Problem [34].

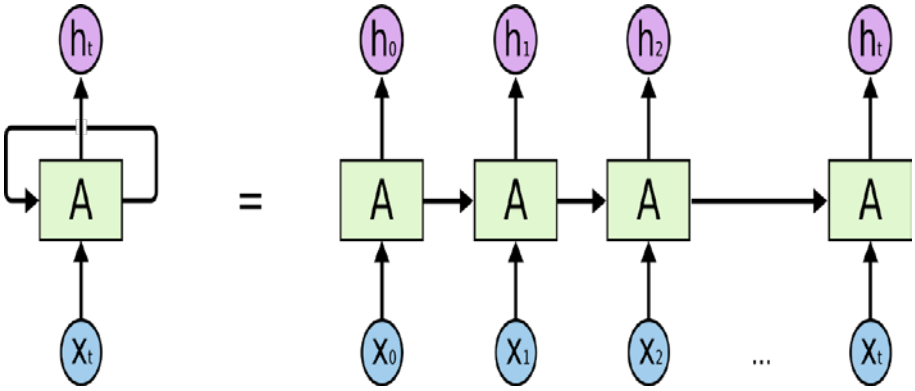

Figure 1.11 Recurrent Neural Network (RNN) Architecture *[81]*

1.5.3 Long Short-Term Memory (LSTM)

When RNNs are fed a long sequence of data, they have trouble transporting the knowledge from earlier time steps to later ones, causing them to miss vital information. RNNs are also plagued by the problem of disappearing gradients. LSTMs are being developed to address these concerns. They are dubbed LSTMs because they can store short-term memories for extended periods. The LSTMs are identical to RNNs, except that they contain memory blocks. The information is allowed to pass through gates in the memory blocks. To determine whether the gates are triggered or not, sigmoid activation units are used. The sigmoid function returns a value between 0 and 1, indicating how much information should be revealed. LSTMs have three gates: the forget gate, input gate, and output gate. The forget gate layer, which is a sigmoid layer, helps decide what information needs to be discarded [35]. LSTMs are illustrated in the Figure 1.12.

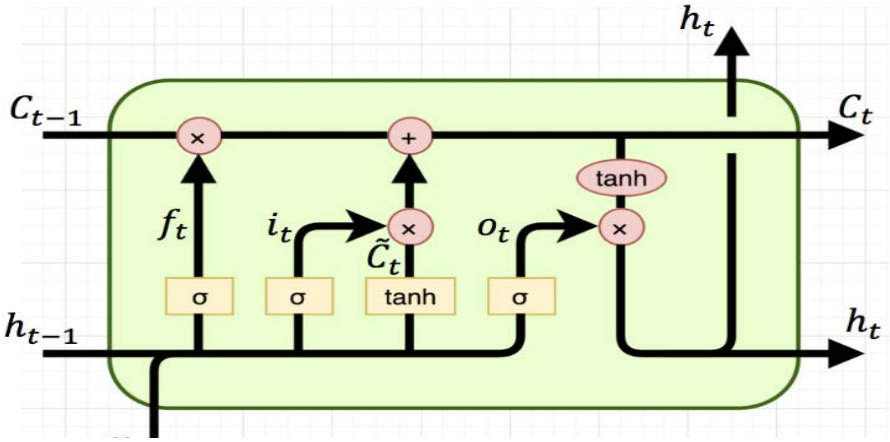

Figure 1.12 Illustration of the LSTM structure *[79]*

1.5.4 Gated Recurrent Unit (GRU)

GRU is a newer RNN that looks a lot like LSTMs GRUs transport information by employing hidden states rather than the cell state or memory [36]. A reset gate and an update gate are the two gates they have. The update gate functions in the same way as the LSTM's forget and input gates. On the other hand, the reset gate is used to determine how much information from the past must be erased. Because they have fewer tensor operations than LSTMs, GRUs are faster. RNNs have short-term memory difficulties, which LSTMs and GRUs are designed to solve [37]. The GRU structure is depicted in Figure 1.13.

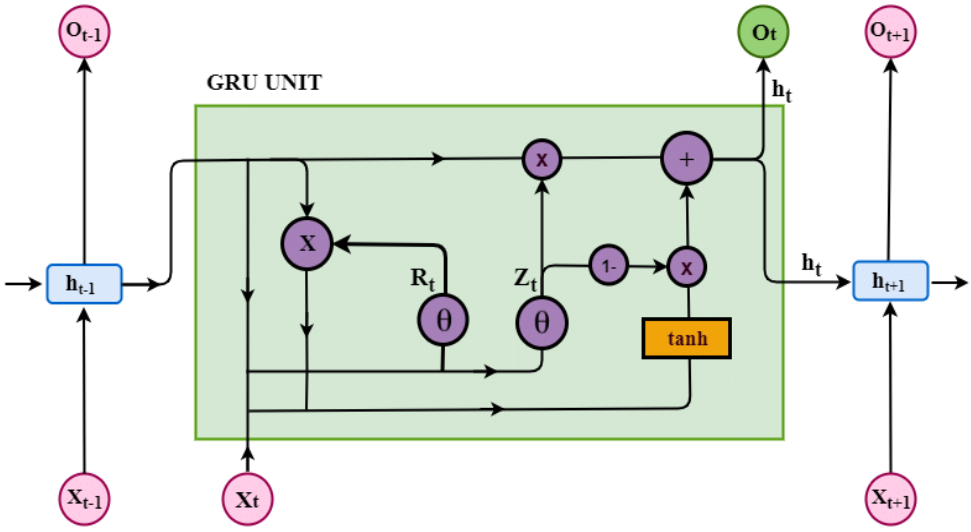

Figure 1.13 Illustration of the GRU structure *[80]*

1.5.5 Echo State Network (ESN)

Echo state network are a kind of recurrent neural network (RNN). ESN are based on Reservoir Computing (RC), which simplifies the training procedure of traditional RNNs. Reservoir Computing's input signal is connected to a non-trainable and random

dynamical system (the reservoir), thus creating a higher dimension representation (embedding). This embedding is then connected to the desired output via trainable units [38]. (Figure 1.14) is a modelization of the ESN architecture. Echo state network is a very powerful neural network for time series forecasting comparing to MLP and statistical methods when modeling chaotic time series data and many studies improve that [39].

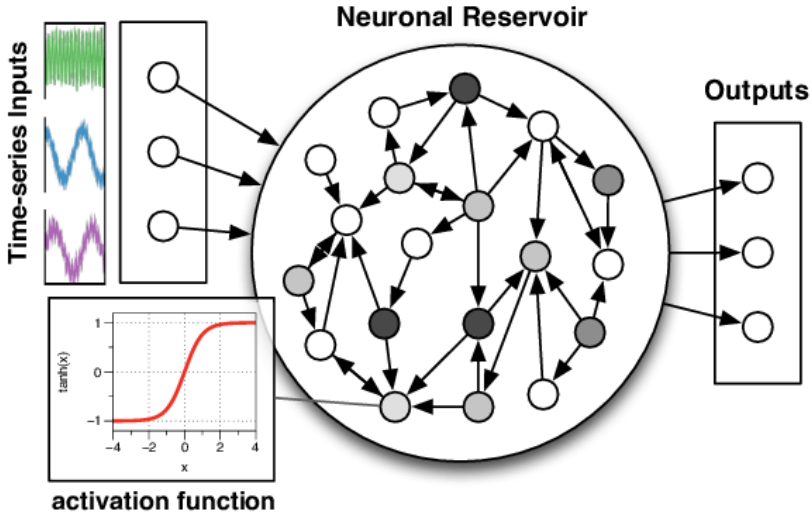

Figure 1.14 Architecture of ESN *[24]*

1.5.6 Convolutional LSTM (ConvLSTM)

LSTMs are extraordinary at recognizing temporal relationships yet they don't perform well in perceiving spatial connections. CNNs on the other hand are great in finding spatialconnections however not temporal relations. To handle this, Convolutional LSTMs (ConvLSTM) are built which caters spatiotemporal connections simultaneously i Although LSTMs excel at understanding temporal linkages, they struggle to perceive geographical connections. CNNs, on the other hand, excel in detecting spatial connections but not temporal ones. Convolutional LSTMs (ConvLSTM) are built to handle this, and they cater to spatiotemporal linkages in the data at the same time [24]. The pattern that exists based on the location of one data point relative to others is referred to as spatial connection in time series. The chronological sequence of the data points is the temporal relationship.

In time series, spatial relationship refers to the pattern that exists based on the location of one data point relative to others. Whereas temporal relationship is the sequential order of the data points. In comparison to conventional LSTM, ConvLSTM is able to cater the spatiotemporal structures by vectorizing the spatial information thereby overcoming the limitation of vector-variate representations in LSTM where spatial information is lost [40].

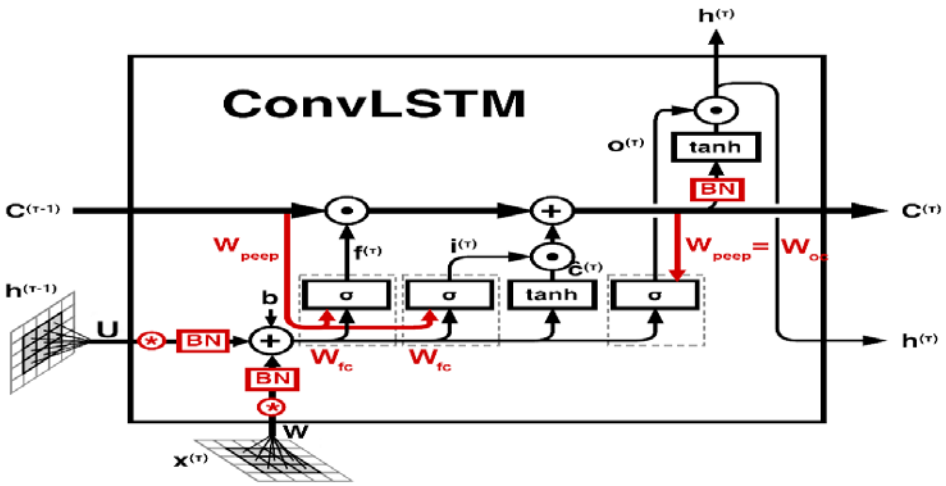

Figure 1.15 A ConvLSTM Cell *[41]*

1.6 Conclusion

In this chapter, we discussed the important aspects of time series data, analysis, and its components, as well as the various forecasting techniques in detail. Different types of models have been investigated, ranging from traditional statistical models to machine learning and deep learning models, and common evaluation metrics that are commonly used in time series have been illustrated. The next chapter will detail convolutional neural networks.

CHAPTER 2:

Convolutional Neural Networks

(CNN)

CHAPTER 2: Convolutional Neural Networks (CNN)

2.1 Introduction

In recent years, artificial intelligence, or AI, has been a hot topic in the media. Machine learning and deep learning, which are artificial intelligence subfields, appear in a slew of articles, many of which are not focused on technology. Robots known as AI agents will perform self-driving cars, catboats, virtual assistants, and a variety of other artificial intelligence (AI)-based industries in which human jobs will be rare and economic activity. In this chapter, we'll look at Deep Learning, which is a subfield of Machine Learning, which is a part of AI (Figure 2.1), as well as the relationship between AI, Machine Learning, and Deep Learning, as well as how and which problems may be solved utilizing these three techniques.

2.2 Historical context

2.2.1 Artificial Intelligence

"Artificial Intelligence, or AI, is the science of teaching computers to think and behave like humans in order to solve more complicated problems without the assistance of a programmer."

Artificial intelligence was first proposed in the 1950s by a group of pioneers in the young field of computer science, who wondered if machines might be programmed to behave intelligently and think like humans, an issue whose repercussions we are still debating today. For example, a chess programmer uses solely hardcoded rules and does not qualify as Machine Learning. Many of them believed that reaching the human level would need programmers to handcraft a large enough set of explicit rules for manipulating information. From the 1950s to the late 1980s, this method was known as symbolic AI. Although symbolic AI has made significant progress in tackling well-defined logical issues, clear rules for handling complicated problems such as image classification, language translation, and speech recognition remain elusive. This opens up a lot of room for a new method known as Machine Learning ML [42].

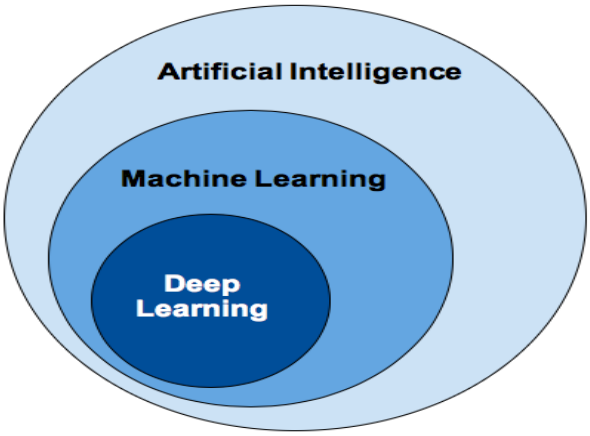

Figure 2.1: Deep Learning is a Subfield of Machine Learning which is a Subfield of AI *[82]*

2.2.2 Machine Learning

Machine Learning (ML) is the process of teaching computers to learn from their previous experiences and settings, which is referred to as the natural human learning process. We feed a dataset and a predicted result to the computer and let it learn and analyze the relationship between them in order to learn how that particular data could lead to this result" Machine Learning has piqued the interest of researchers in AI and computer sciences since 1983. The question has always been how a computer can learn from experiences (data) rather than programmers crafting data processing rules by hand, and how to learn how to perform tasks on its own. In traditional programming or symbolic AI, the programmer inputs rules or programmers (algorithms) and data to be processed based on these rules, and the programmer outputs answers. In machine learning, the programmer inputs data and the expected answers based on this data, and the programmer outputs the rule, which then matches the data and the answers (Figure 2.2). The rule can be applied to new data to get unique results [43].

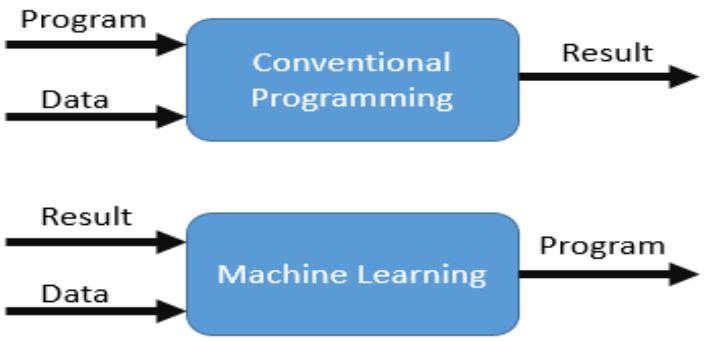

Figure 2.2: Traditional Programming vs. Machine Learning *[83]*

"It would be beneficial if computers could learn from their mistakes and thereby enhance the efficiency of their own programmers while they are being executed. Within the framework of appropriate programming, a simple but effective rote-learning facility can be given [44]."

Machine learning is capable of learning and improving as a result of its experiences. The raw data is used to extract relevant information that aids in learning and decision-making utilizing shallow or deep architecture (Figure 2.3) to grant that the machine learning process begins with raw data [45].

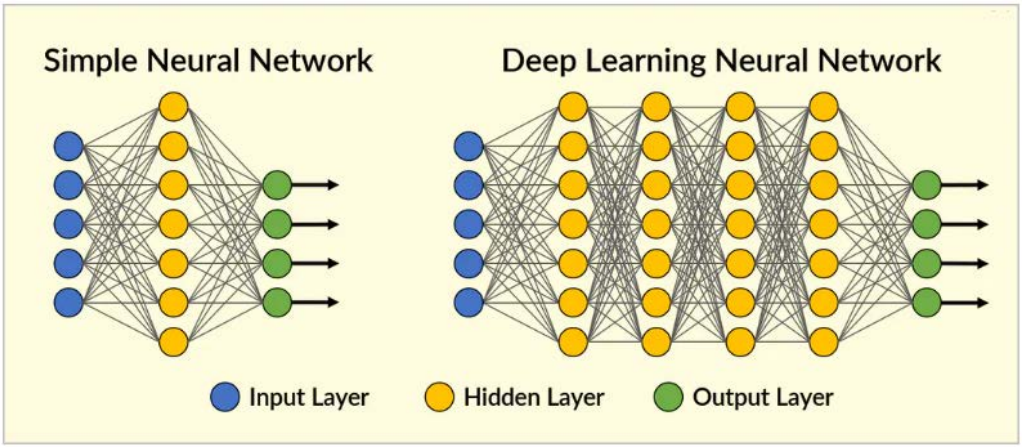

Figure 2.3: Difference between a simple Neural Network and a Deep Learning Neural Network *[66]*

2.2.3 Types of Machine Learning

There are so many different types of machine learning systems that it is useful to classify them in broad categories (Figure 2.4):

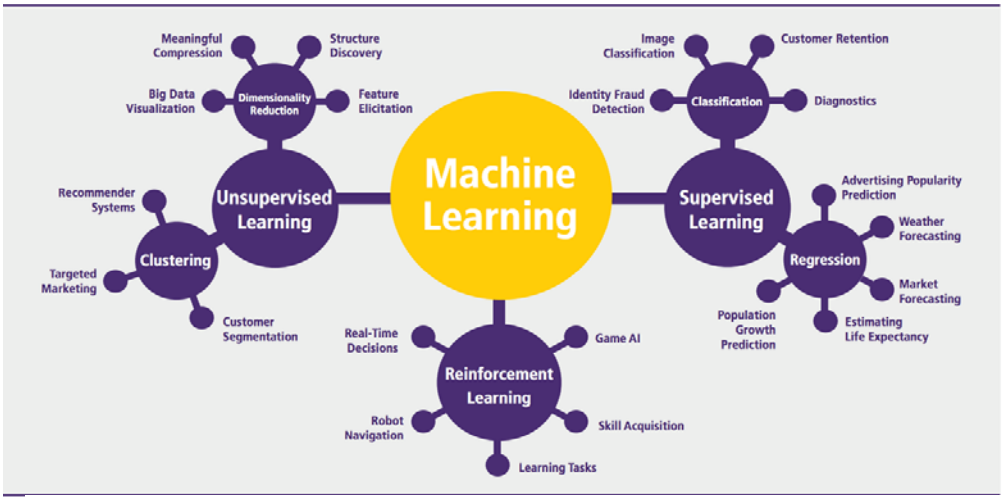

Figure 2.4: Machine-learning types with problems examples *[66]*

Machine Learning system is trained rather than be explicitly programmed, AI focus on teaching computers how to learn without being programmed for specific tasks Machine Learning can be carried out using following approaches.

2.2.3.1 Supervised learning

In machine learning and artificial intelligence, supervised learning, the most common type, is a group of algorithms that define a predictive model using data whose outcomes are known. The model relies on training on this data while the output is clear and tries to establish relationships between the data and its output in order to predict the new data (generalization of the model) through a suitable learning algorithm like a network of neurons, a random forest and a linear regression that works through an optimization routine to reduce the loss function. In general, almost all deep learning applications that are in the spotlight these days belong to this category, such as optical character recognition, speech recognition, image classification, and language translation. There are two main types of supervised machine learning problems, called classification and regression [45].

2.2.3.2 Unsupervised learning

Unsupervised learning In Machine Learning and Artificial Intelligent Unsupervised learning involves data that comprises input without any target output. Is a type of algorithm that learns patterns from untagged data. The hope is that the machine will be pushed to develop a compact internal representation of its surroundings through imitation, which is a key way of learning in humans, and then generate inventive material from it. The self-organization of unsupervised approaches captures patterns as probability densities or a mixture of neural feature preferences. Reinforcement learning, in which the computer is given merely a numerical performance score as guidance, and semi-supervised learning, in which only a small percentage of the data is tagged, are the other stages in the supervision spectrum. Neural Networks and Probabilistic Methodologies are two broad methods in Unsupervised Learning [46] .

2.2.3.3 Reinforcement learning

Reinforcement learning (RL) is a branch of machine learning that studies how intelligent agents should operate in a given environment to maximize the concept of cumulative reward. Reinforcement learning, along with supervised and unsupervised learning, is one of the three main machine-learning paradigms. Reinforcement learning differs from supervised learning in that it does not need the presentation of labelled input/output pairings or the explicit correction of sub-optimal behaviors. Reinforcement learning has proven to be effective in a variety of applications, including autonomous helicopter flight, robot legged movement, cell-phone network routing, marketing plan selection, factory control, and efficient webpage indexing [47].

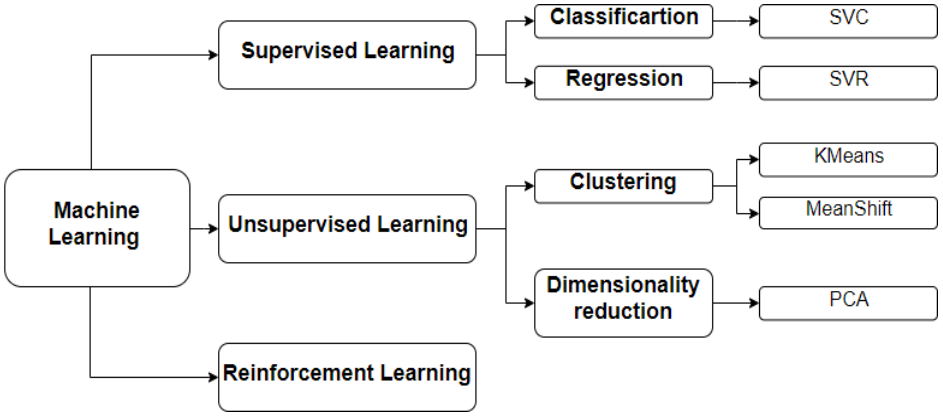

Figure 2.5: Machine Learning Approaches with Algorithm Example

2.3 Deep Learning and Neural Networks

Deep learning (also known as deep structured learning) is a type of machine learning technology that uses artificial neural networks to learn representations. Deep learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks, and convolutional neural networks have been used in fields such as computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection, and board game programmers, producing results that are comparable to, and in some cases superior to, traditional approaches. ANNs differ from biological brains in a number of ways. Artificial neural networks, in particular, are static and symbolic, whereas most living animals' biological brains are dynamic (plastic) and analogue [30].

Figure 2.6 shows an example of what the representations learned by a deep learning algorithm look like. As it is seen, the network transforms a digit image into representations that are increasingly different from the original image and increasingly informative about the final result to recognize what digit it is.

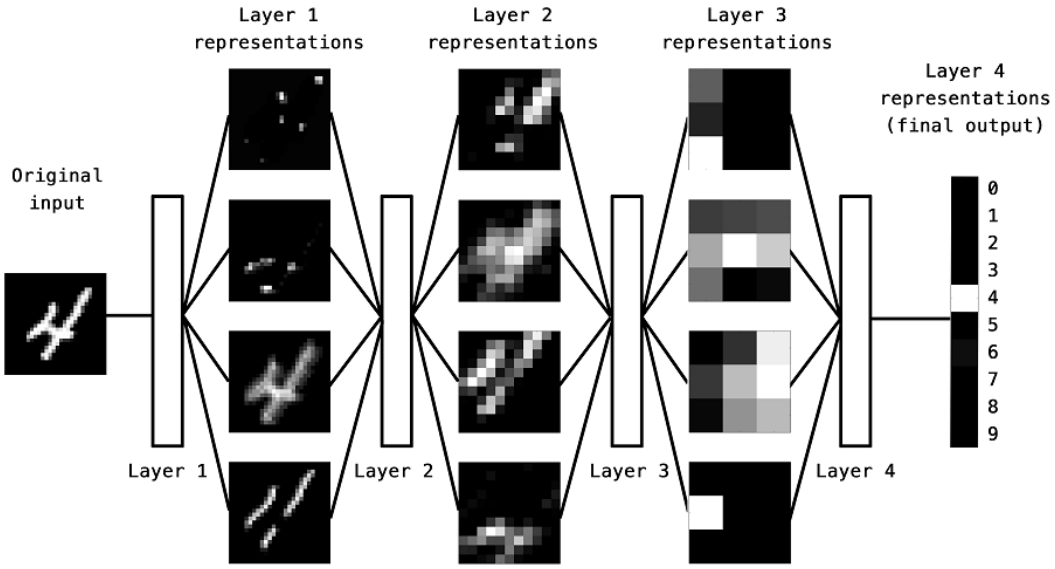

Figure 2.6: Deep representations learned by a digit classification model

Feature engineering is a key step in the model building process. It is a two-step process:

- Feature extraction
- Feature selection

2.3.1 Artificial Neurons

Artificial neurons are a mathematical function envisioned as a model for biological neurons. Artificial neurons are the basic building blocks of a neural network. One or more inputs are received by artificial neurons. Each element is usually weighted separately, and the sum is then processed through a nonlinear function called the activation function or the transfer function. Transfer functions are commonly sigmoidal, although they can also be nonlinear functions, multiple defined linear functions, or sigmoid functions. A move they are also frequently monotonous, continuous, differentiable, and finite.

Neurons in deep learning models are nodes through which data and computations pass, and this functional understanding of the neurons in our brain is translated into an artificial model that can be represented on a computer [48]. Neurons work like this (view figure 2.7):

- Neurons receive one or many input signals either from the raw dataset or from the previous neuron (the previous layer) of the network.
- Neurons do some calculations.
- Finally, Neuron sends output signals to neurons in the next hidden layer through a synapse.

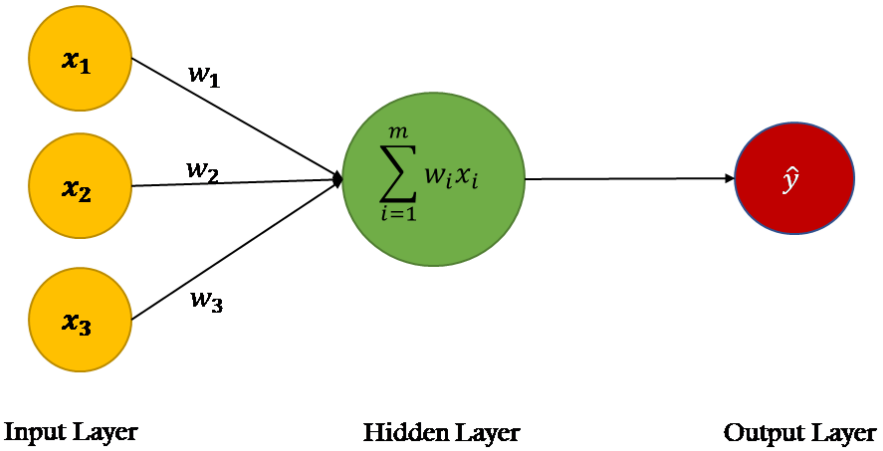

Figure 2.7: Diagram of the Functionality of an artificial neuron

Through synapses, neurons in Deep Learning models can link to more than one neuron in the preceding layer

A neuron receives its input from the previous neurons in the preceding layer of the model, then adds up signals multiplied by the corresponding weight then pass the result to an activation function; Figure 2.8 shows the complete process:

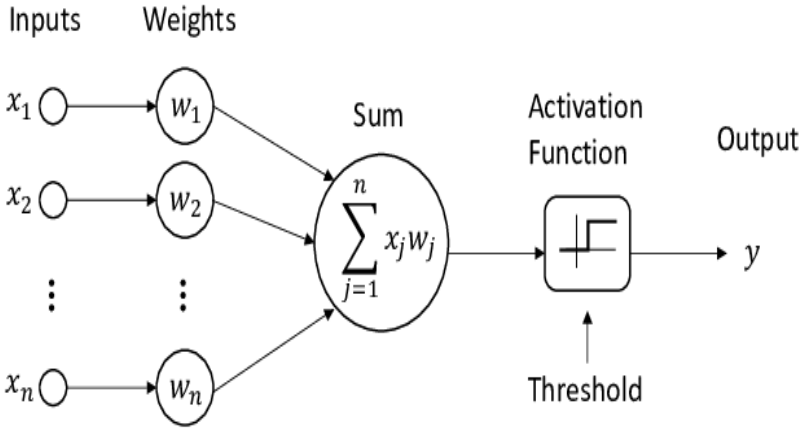

Figure 2.8: illustration of an artificial neuron

Mathematically, we have numbers of inputs $x_1, x_2, x_3, \ldots, _n$, each one of those inputs ismultiplied by specific weight $w_1, w_2, w_3 \ \ldots, w_n$.

The results of this multiplication are summed together to produce the logit of theNeuron:

$$\sum_{j=0}^{n} x_j w_j$$

In many cases, the logit also include bias, which is a constant:

$$\sum_{j=0}^{n} x_j w_j + b$$

This logit passed through a function *f* in order to produce our output $\boldsymbol{y} = (\boldsymbol{z})$.

We may also express this functionality in victor form, our input as a vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$, and the weights of the neuron as $\boldsymbol{w} = [w_1, w_2, \ldots, w_n]$, so our function become $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}.\boldsymbol{w} + \boldsymbol{b})$, where *b* is the bias term.

The role of the activation function is to calculate the output value of neurons, the value obtained passed through the next layer of our network using synapse [49].

2.3.2 Biological Neural Network

A neural network is a network or circuit of biological neurons, or, in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus, a neural network is either a biological neural network, made up of biological neurons, or an artificial neural network, used for solving artificial intelligence (AI) problems. The connections of the biological neuron are modeled in artificial neural networks as weights between nodes. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed [50]. This activity is referred to as a linear combination. Finally, an activation function controls the amplitude of the output.

Neurons in deep learning were inspired by neurons in the human brain Figure 2.9 shows the anatomy of a brain neuron:

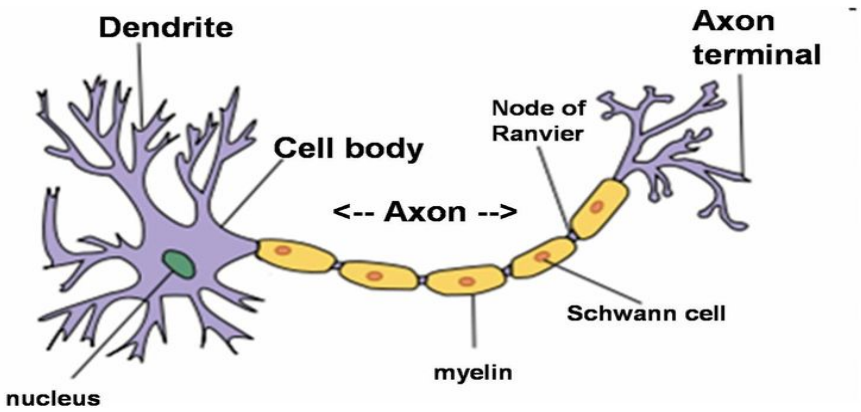

Figure 2.9: Neuron in Biology *[85]*

Neurons, as can be seen, have a unique structure. Neurons in the human brain work together in groups to execute functions that humans require in their daily lives. During his groundbreaking research in neural networks, Geoffrey Hinton wondered if we might develop a computer algorithm to imitate neurons in the human brain [49]. The hope is that by mimicking

brain anatomy, they will be able to capture some of its capabilities. Researchers and scientists researched the functioning of neurons in the human brain to accomplish this. An key finding is that the neuron is useless on its own. To generate meaningful activities, it instead requires networks of neurons (Neuron Network). The reason for this is that neurons send and receive messages from other neurons that are connected to them. The dendrites of neurons can receive signals from the preceding cell and transmit them through the axon. The neuron's dendrites are attached to the axon of another neuron. This link is referred to as a synapse. Deep learning has generalized the synapse notion [49].

2.3.3 Artificial neural networks

One sort of machine learning model is neural networks, which have been around for at least 50 years. Many key architectural improvements in neural networks were made in the mid-1980s and early 1990s. The amount of effort and data required to achieve effective outcomes, on the other hand, hindered adoption. In the early 2000s, processing power increased significantly, resulting in a "Cambrian explosion" of previously unattainable computational approaches, which rekindled interest in neural networks. The ANN is a feed-forward multilayer ANN. The standard ANN design (Figure 2.10) consists of an input layer, a collection of hidden layers, and an output layer. Artificial neurons are coupled via adaptive weights in each hidden and output layer [50].

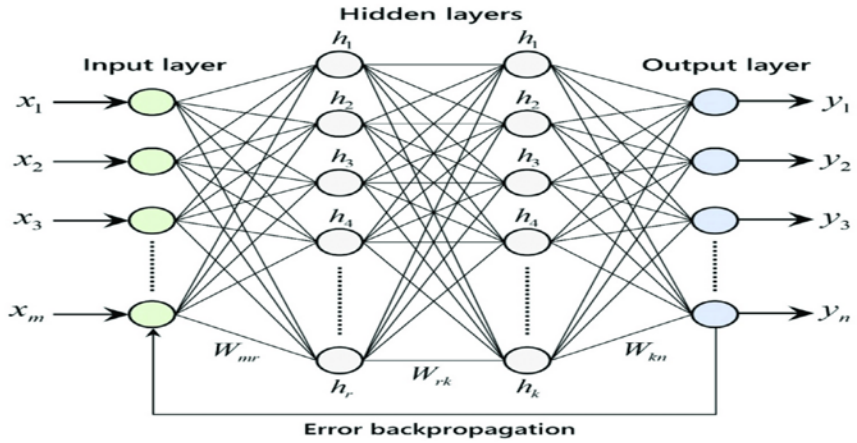

Figure 2.10: Artificial Neural Network Architecture *[86]*

The simplest type of Artificial Neural Networks ANNs was the feedforward Neural Network cause the information moves in one direction only, forward, from the input layer nodes through the nodes of the hidden layer and to the output layer nodes, Neural Network learn (update weight) by learning algorithm called Back-propagation [51].

- **Input layer:**

The input layer of a Neural Network contains a group of artificial neurons that hold the initial data for the neural network and bring it into the system for further processing by subsequent layers of the artificial neuron. The input layer is the very beginning of the workflow for the artificial neural network, input layers are followed by one or many hidden layers. On images processing input layer will hold the pixel intensity of the image for example an RBG image with width w=64 and height h=64, and depth d=3 will have an input dimension of 64×64×3.

- **Hidden layers:**

A neural network's hidden layers are a layer that sits between the input and output layers. Artificial neurons take in a collection of weighted inputs and produce outputs through an activation function in a hidden layer. The weights of hidden neural network layers are usually assigned at random, but they can also be fine-tuned (by using the weights of other models) and calibrated through the backpropagation process.

- **Output layer:**

The output layer is the final layer of an artificial neural network, and its neurons create the network's output value. The output layer is formed in a variety of ways depending on the neural network's architecture. In classification problems, the final output may be a set of probabilities, while in regression problems, the final output may be a real-valued output. The type of activation function employed on the output layer neurons controls the output**.**

2.3.4 Models of Artificial Neural Networks

Single artificial neuron cannot solve real life problems at all, however combining two or more artificial neurons are capable of solving complex real life problems. The neural network can be defined as an interconnection of neurons. Related neuron outputs and inputs are connected, through weights. Delay block can be placed between neurons if needed. Neurons of an artificial neural network are not randomly interconnected. There are standardized topologies of neural networks. These topologies are fixed and predefined so as to solve the problems in an efficient and easy way. These topologies can be examined in two basic classes which are feed-forward and recurrent topologies. Therefore feed-forward neural network (FNN) topology is also called acyclic graph. On the contrary, in simple recurrent neural network (RNN) topology the information does not flow only from input to the output direction, but also flows from output

to the input direction. Hence, these kinds of topologies can be referred to as semi-cyclic graphs [52].

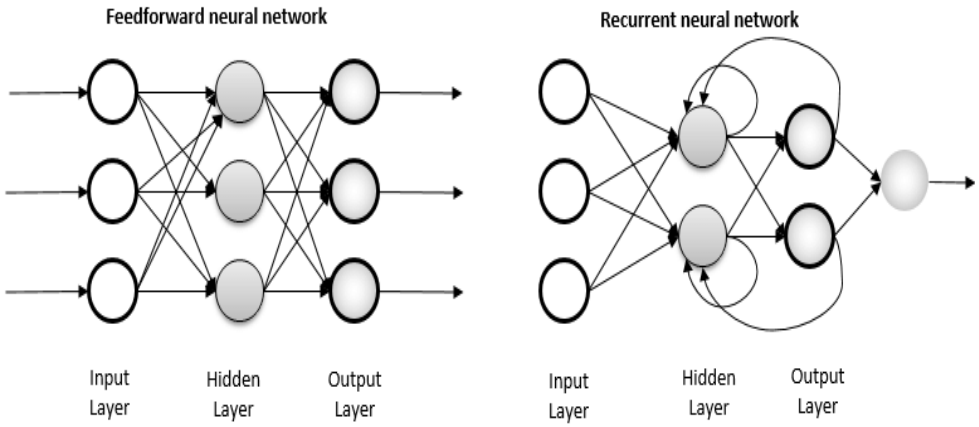

Figure 2.11: Feed-forward (FNN) and recurrent (RNN) topology

Neural networks can be described layer by layer which refers to a group of neurons in same level as shown on Figure 2.11. From input to the output direction, the first layer is called input layer and the last one is named output layer. All remaining layers placed between the input and output layer are labeled as hidden layers. Hidden layers are where neural networks stores abstract and internal representation of the training samples. A single hidden layer network which has a finite number of units can be trained to express any random function with an acceptable error ratio with respect to the universal approximation theorem. Although single hidden layer network is sufficient to learn any function, multi hidden layer networks can give better results.

2.3.5 Gradient Descent

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine learning, we use gradient descent to update the parameters of our model. Parameters refer to coefficients in Linear Regression and weights in neural networks.

Consider how we might reduce the squared error across all training examples by simplifying the problem. Let us pretend our linear neuron only has two inputs (and thus only two weights, w1 and w2). Then consider a three-dimensional space in which the horizontal dimensions correspond to the weights w1 and w2, and the vertical dimension corresponds to the error function E's value. Places on the horizontal plane in this space correspond to different weight settings, and the height at those points refers to the mistake suffered. We get a surface in this three-dimensional space, namely a quadratic bowl, if we evaluate the errors we make

over all possible weights, as illustrated in Figure 2.12 [49].

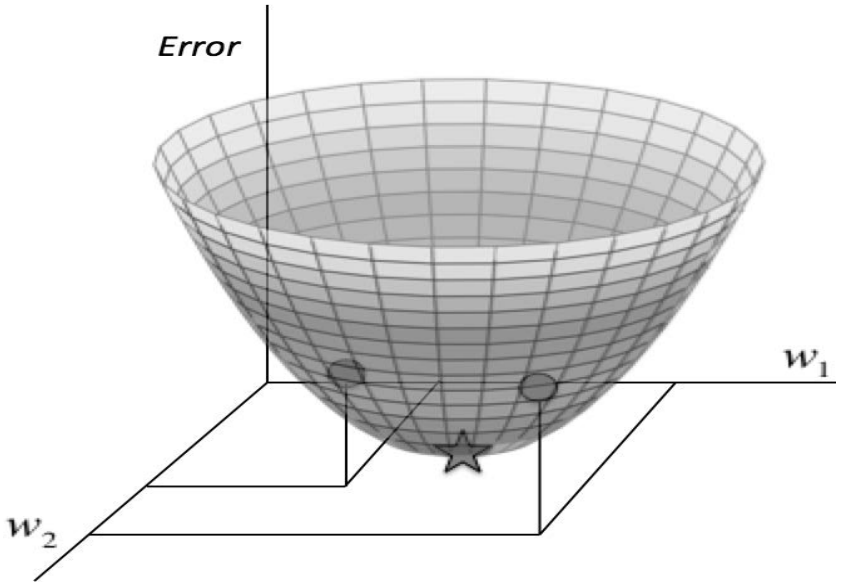

Figure 2.12: The Quadratic Error Surface for a Linear Neuron *[84]*

We can now devise a strategy for determining the weight values that minimize the error function; the weights for our network are randomly initialized, so we end up someplace on the horizontal plane. By calculating the gradient at our current location, we may determine the steepest descending path and then take a step in that direction, bringing us closer to the minimum than before. Following this technique, we can reconsider the direction of steepest descent by taking the gradient in this new direction and taking a step in this direction, as illustrated in Figure 2.13 [49].

This is known as the Gradient Descent Algorithm, and it was developed to address the problem of training individual neurons as well as the more general task of training entire networks.

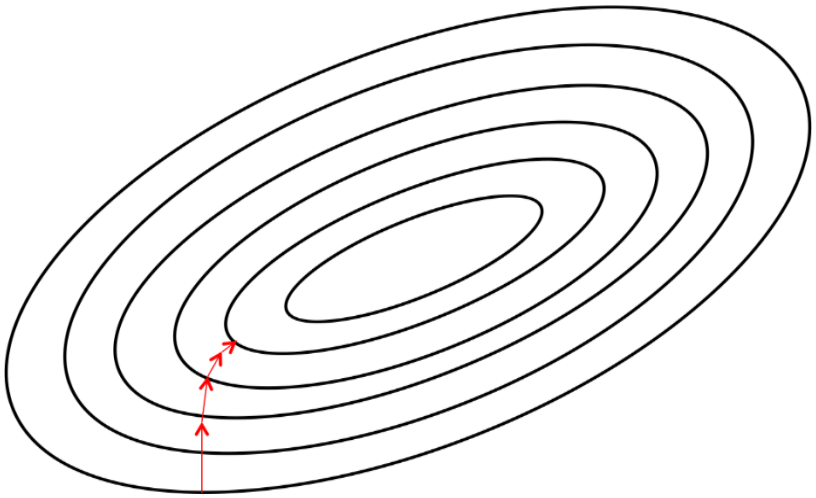

Figure 2.13: Visualizing the Error Surface as a set of Contours *[84]*

One of the most crucial hyperparameters in this process is the learning rate. Before recalculating our new heading, we need to figure out how far we want to go. Because the closer we get to the minimum, the shorter we want to step forward, and the closer we go to the minimum, the flatter our surface becomes, we may use the steepness as an indicator of how close we are to the minimum, but if our surface is mellow, training can take a long time. As a result, the gradient is frequently multiplied by a quantity, the learning rate. Choosing a learning rate is a difficult task (Figure 2.14).

We risk taking too long during the training process if we choose a modest learning rate, but if we choose a large value for the learning rate, we will most likely start drifting away from the minimum [49].

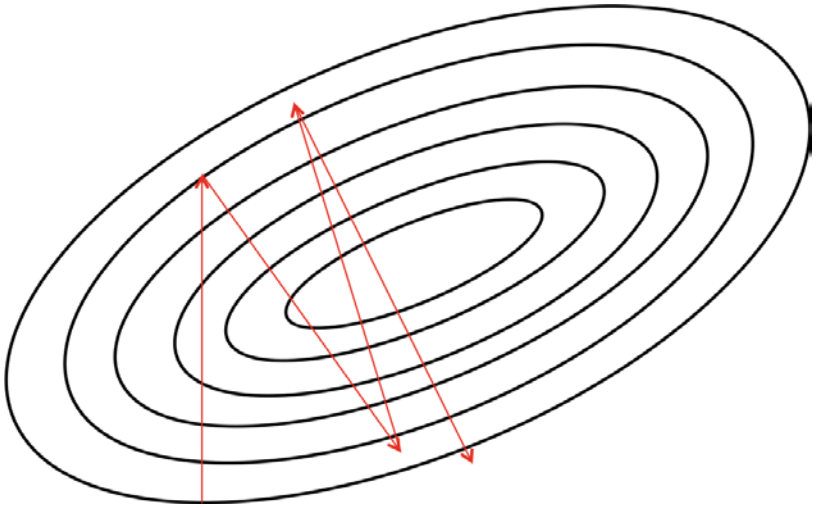

Figure 2.14: Convergence is Difficult when our Learning Rate is too Large *[84]*

2.4 Activation Functions

The activation function is a mathematical gate between the input, which is a value coming from the previous neuron and the output, and which is a value flowing to the next layer neurons (Figure 2.15). We can characterize it as a function that turns the neuron output on or off based on the applied rule [53].

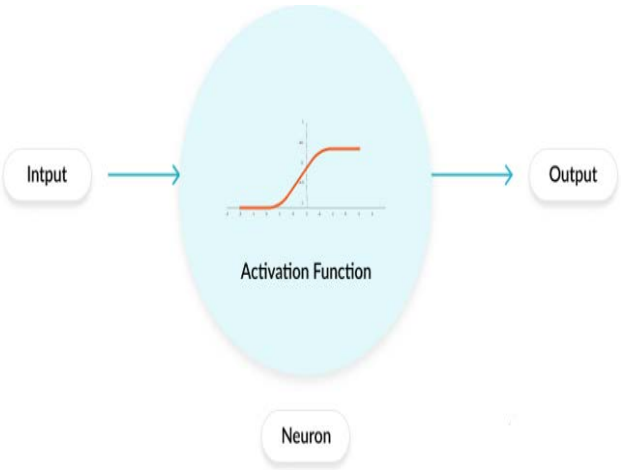

Figure 2.15 : Activation Function *[92]*

There are two types of activation functions that can be used on neural networks: linear activation functions and no-linear activation functions. The latter is the most commonly used because it can assist the network in learning complex data. Activation functions can also be used to filter out data. Here are some examples of common activation functions:

2.4.1 Linear activation functions

A linear activation function takes the form: $f(x) = cx$

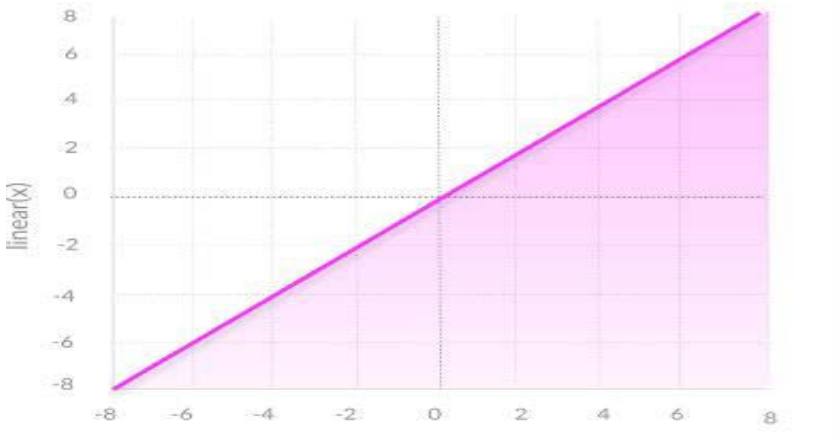

Figure 2.16: Linear Activation Function *[92]*

This produces a signal output that is identical to the input by multiplying the inputs by the weight assigned to each neuron. In some ways, a linear function is superior to a step function since it provides for several outputs rather than simply yes or no.

We can state that the neuron receives input x1, x2, x3,... xn, and that the linear neuron's output is supplied by:

$$y = w_1 x_1 + w_2 x_2 + w_3 x_3 \ \ldots + w_n x_n + b$$

Where $w_{1,2},w_3 \ldots w_n$ are the weight corresponding to $x_{1,2},x_3 \ldots x_n$ respectively and b is the bias [53].

A neural network with a linear activation function is simply a linear regression model. It has limited power and ability, to handle complexity-varying parameters of input data [54].

2.4.2 Activation Functions (Non-Linearity)

Instead of using linear activation, function modern models use non-linear activation function in order to create a complex mapping between the network's inputs and outputs, image processing and dataset that have high dimensionality.

Non-linear activation functions solve the problems of the linear-activation function

- Non-linear activation function allows backpropagation process because they have a derivative function; the derivative of a linear function is always.
- Non-linear activation function gives high accuracy comparing to the linear one those are the most used non-linear activation function.

2.4.2.1 Sigmoid

Which uses the function:

$$f(X) = \frac{1}{1 + e^{-x}}$$

The Sigmoid function has an S-shape (Figure 2.17), which means that if the input is little, the result is near to 0, but if the input is high, the output is closer to 1.

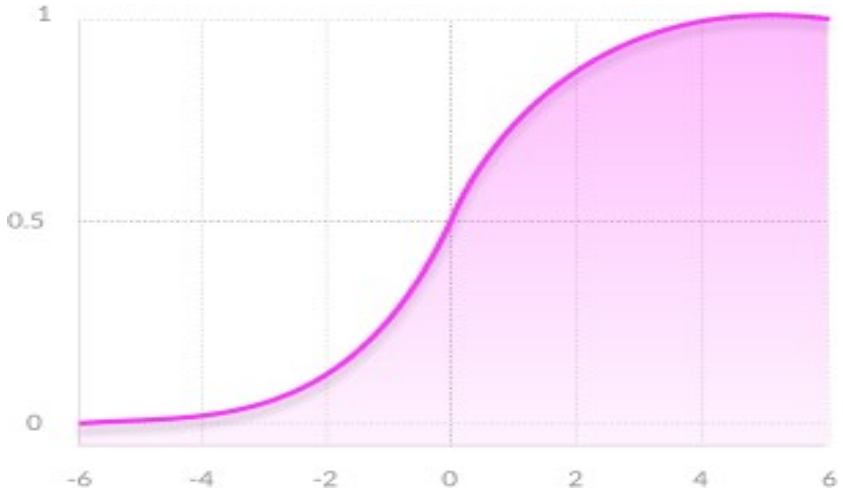

Figure 2.17: Sigmoid Function *[92]*

2.4.2.2 Tanh

Which is similar to sigmoid function but instead of ranging from 0 to 1, the output of tanh range from -1 to 1, use (x) = tanh(x) it's the ratio of the hyperbolic sine to the hyperbolic cosine:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)}$$

The graph of tanh function is similar to the sigmoid function (Figure 2.18).

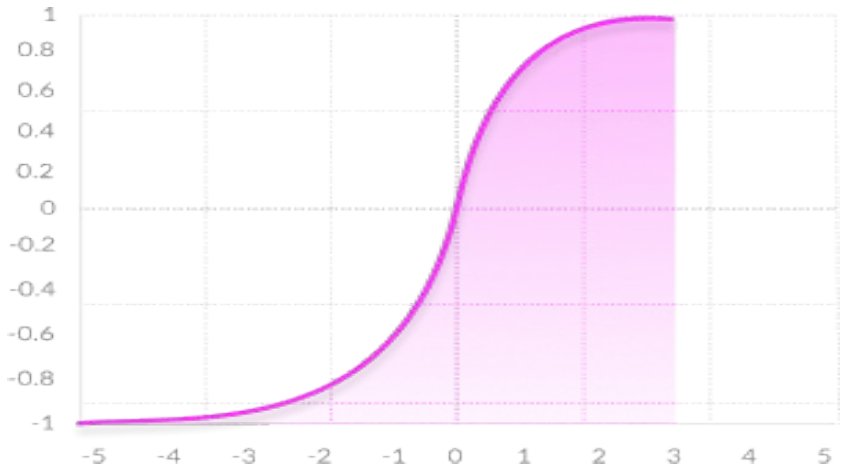

Figure 2.18: Tanh Hyperbolic *[92]*

The gradients get smaller and smaller during the backpropagation phase until they vanish; no gradients means no learning, which is known as the vanish gradient problem. Because of the sigmoid function, information is squeezed. The solution to this problem is to employ an activation function like RELU that does not squeeze information [55].

2.4.2.3 ReLU

Rectified linear is a more interesting transformation that activates a node only if the input is above a certain quantity. While the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable as demonstrated in Figure 2.19 [54].

$$f(x) = \max(0 , x)$$

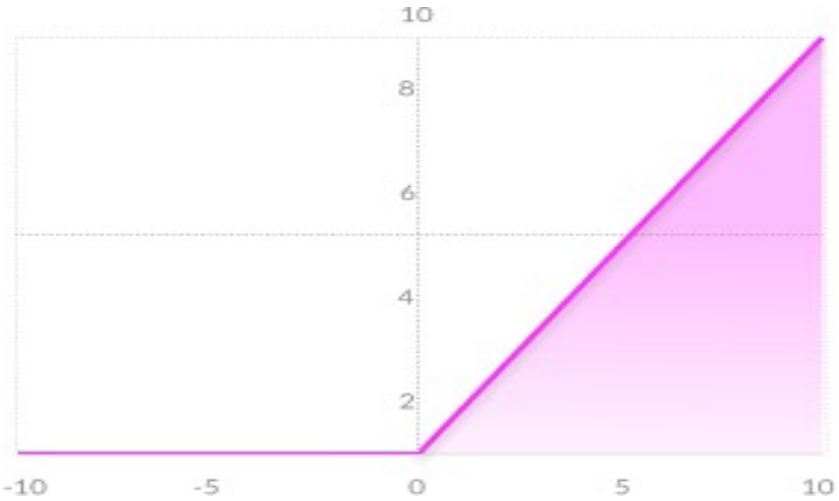

Figure 2.19: The ReLU function graph *[92]*

As we can see, (x) is zero when x is less than zero and (x) is equal to x when x is aboveor equal to zero. As we can see all the negative values become zero immediately, that may decrease the ability of the model to fit or train from the data properly because the ReLU block all the inputs less than zero that's called "dying ReLU problem" introducing some activation evenin the negative cases solve this problem. Leaky ReLU is an attempt to solve the dying ReLU problem [54].

2.4.2.4 SoftMax

SoftMax (Figure 2.20) handles the activation of the output neuron; we can use SoftMax to solve classification problems. the number of classes equal to the number of neurons in thelast layer the value obtained from the SoftMax represents the probability of belonging to a particular class.

A strong forecast indicates that one output is too close to 1, while the other output is clearly close to 0. Otherwise, our prediction is weak [54].

2.4.3 Loss Functions

The loss function measures how near a neural network is to the ideal it is being trained toward. Configuring the loss function is one of the most crucial tasks in a deep learning project

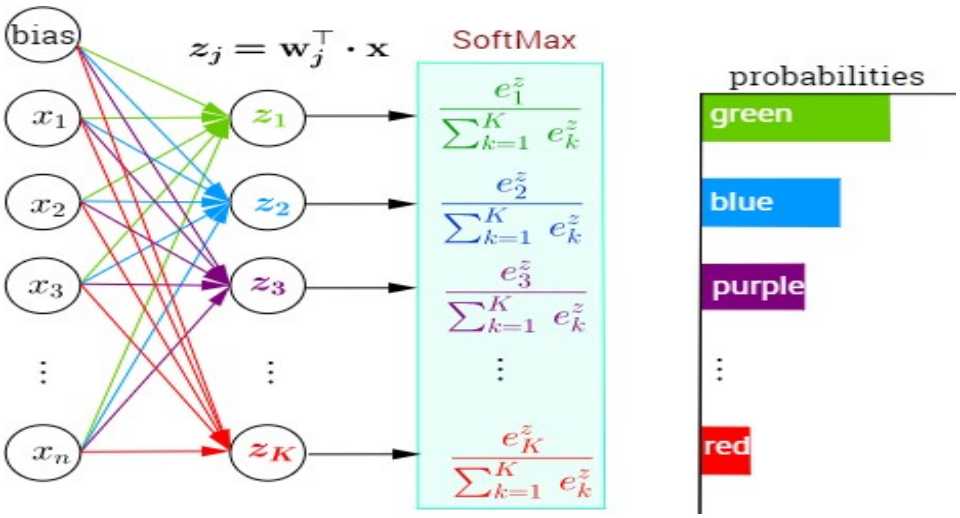

Figure 2.20: SoftMax *[11]*

to guarantee that the model works as intended. The loss function can give the neural network a lot of practical flexibility. Neural networks may perform a variety of tasks, ranging from forecasting continuous values to classifying discrete groups. Because the output format will vary, each task will require a distinct type of loss function. For particular tasks, we can define the loss anyway, we want. The loss function (Figure 2.21) is a function that has two parameters: Predicted Output and True Output [56].

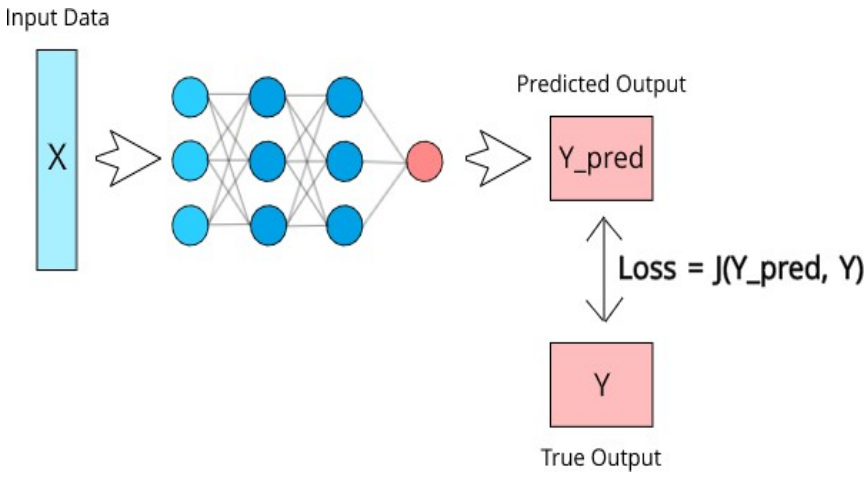

Figure 2.21: Neural Network Loss Visualization *[88]*

The function above calculates how poorly our model is performing by comparing the actual value that we should obtain as an output with what the model is predicting; if Y pred is

far from Y, the loss will be high; if the two values are similar, the loss will be low.

If the loss is significant, this enormous value will propagate across the network while it is training, and the weight will be adjusted, to say, a little more than usual (Figure 2.22). If the loss is minor, the weight will not change significantly because the network is already performing well [57].

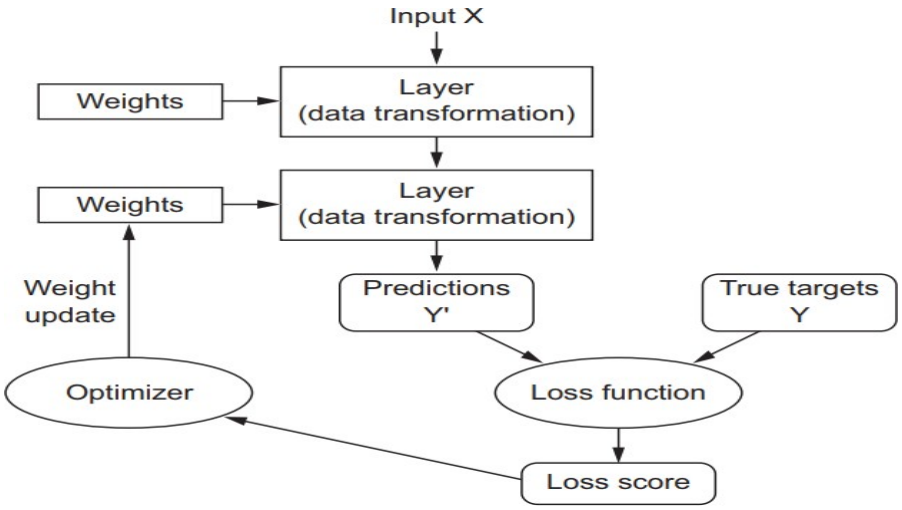

Figure 2.22: Neural Network Workflow *[87]*

The goal is to find the best weights for our network, those that reduce error, and the best approach to do so is to use an optimization process like gradient descent.

2.5 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of neural network CNN, sometimes known as ConvNet, is a family of models inspired by how the visual cortex of the human brain recognizes objects. In the 1990s, Yann LeCun and his colleagues presented a unique neural network design for categorizing handwritten digits from photographs, which led to the development of CNNs [58].

CNNs have attracted a lot of attention as a result of their excellent performance, particularly for image classification tasks, which has resulted in significant breakthroughs in Machine Learning and computer vision applications [59].

ConvNets have a deep feed-forward design that allows them to generalize far better than networks with completely connected layers [60], learn highly abstract characteristics, and efficiently identify objects. Because CNN can be trained easily and does not suffer from overfitting, it is far more difficult to create big networks using general models of Artificial Neural Network (ANN) than it is to implement in CNN. Due to their exceptional performance,

CNNs are widely employed in a variety of fields, including object detection, speech recognition, face detection, facial expression recognition, natural language processing, and many more. The basic idea behind CNNs is to extract local characteristics from input (often an image) at upper layers and combine them into more complex features at lower layers [61].

2.5.1 Convolution operation

Convolution is one of the most important operations in signal and image processing CNN's convolutions are popularly known to work on spatial or 2D data. What's less popular is that there are also convolutions for 1D data [62].

This thesis is concentrated on convolution in 1D spatial, which is mostly used in time series processing for feature extraction and it is the core block of Convolutional Neural Networks. This allows CNN to be used in more general data type including texts and other time series data. Instead of extracting spatial information, you use 1D convolutions to extract information along the time dimension, see Figure 2.23 [62].

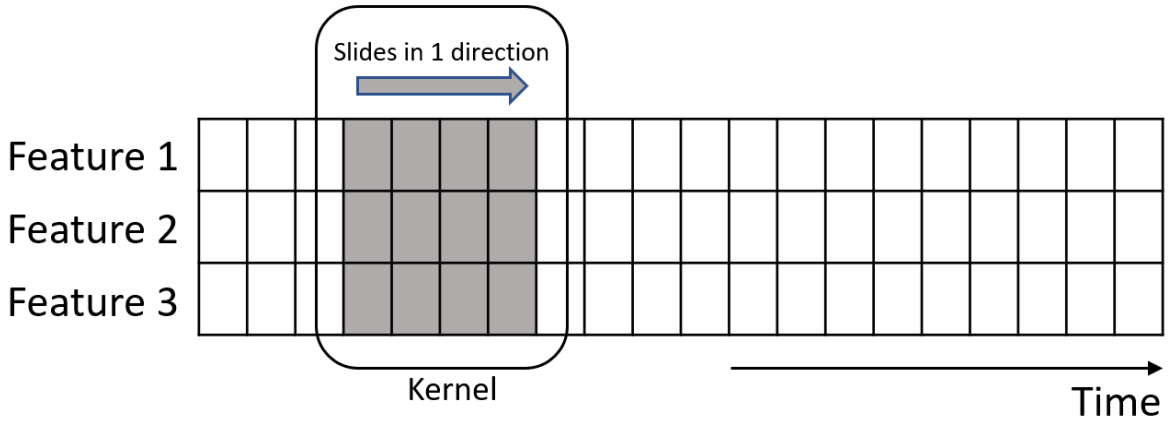

Figure 2.23: Conv1D: Convolving on time dimension *[62]*

2.5.2 Architecture of CNN

Each hidden layer in classic neural networks is made up of a number of neurons, each of which is fully linked to all neurons in the preceding layer. The main distinction between a typical Artificial Neural Network (ANN) and a Convolutional Neural Network CNN is that a CNN only has one completely connected layer, whereas an ANN has every neuron coupled to every other neuron (full connection), as seen in (Figure 2.24) [63].

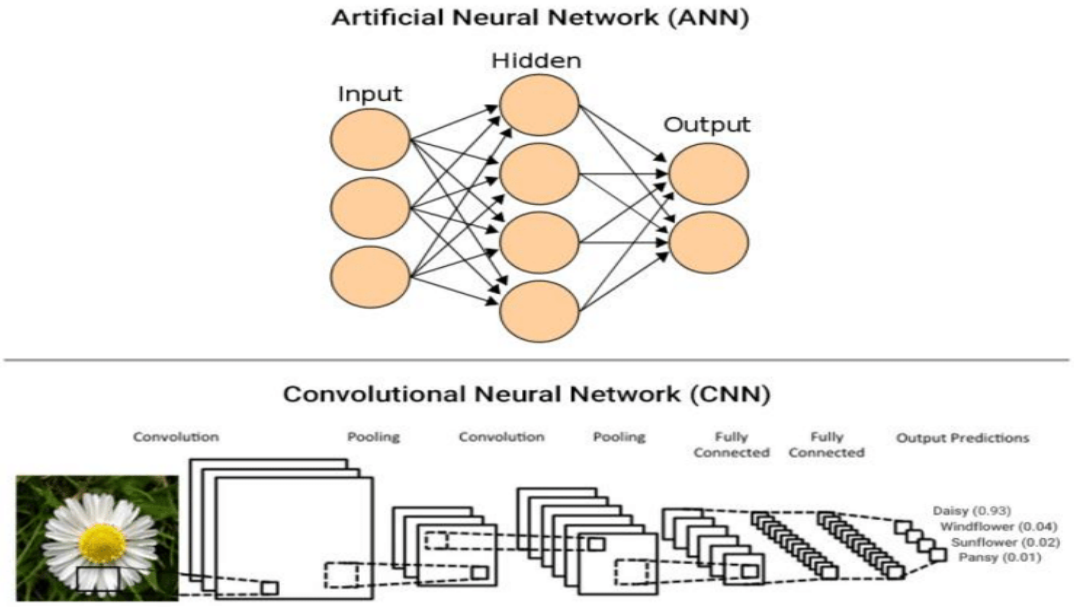

Figure 2.24: Artificial Neural Network and Convolutional Neural Network *[89]*

ANNs are not appropriate to images it leads to over-fitting easily due to image size. Consider an image of size [32x32x3]. If this image is passed through an ANN, it will be flattened into a vector of size 32x32x3, which means 3072 rows so; our ANN must have 3072 weights in its first layer to receive this input vector. For larger images, say [300x300x3], it results in a complex vector (270,000 weights), which requires a more powerful processor to process. All CNN fundamentals are based on three properties: local connectivity, parameter (weight) sharing, pooling and sampling of hidden units.

Convolutional Neural Network is based on a sequence of layers to achieve different tasks.The figure 2.25 shows the architecture of a typical ConvNet that contains the following layers divided on two-part Features Learning and Classification:

Convolutional layers,

Features Learning	Activation function layer (ReLU), Pooling layer, Fully connected layer,
Classification	Output layer with activation function (Softmax)

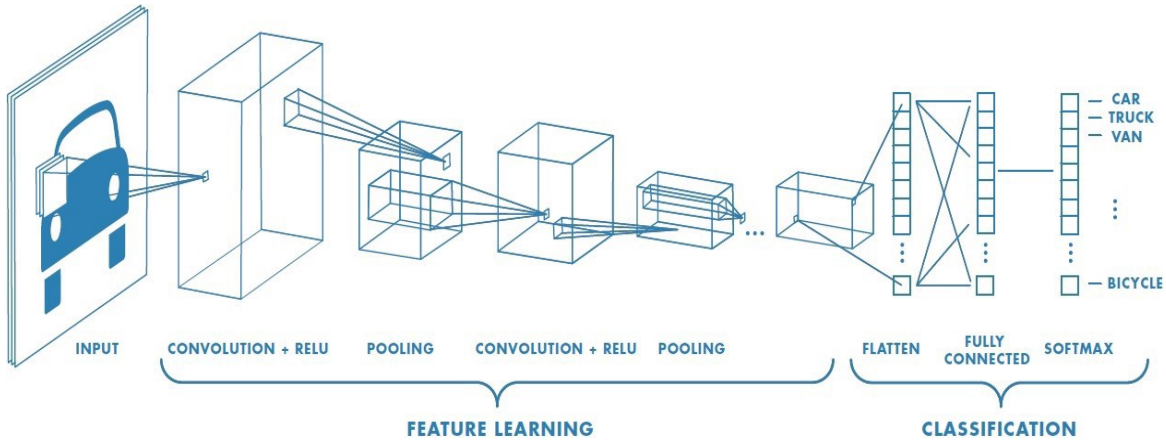

Figure 2.25: Convolutional Neural Network Architecture *[91]*

These layers are combined to form a full Convolutional Neural Network design, which includes a Convolutional and Activation layer, as well as a pooling layer if desired. When it comes to a classification problem like the one shown in the previous picture, the last layer of the CNN uses a SoftMax function (Sigmoid function may be used for binary classification) to calculate the likelihood of which class includes our input [53].

2.5.2.1 Convolution layer

Instead of ordinary matrix multiplication, which takes too long, a network that uses convolutional operation (*), also known as an element-wise product, is utilized. The Convolutional Layer is made up of a collection of filters (kernel or feature detectors), each of which is applied to all sections of the input data (Figure 2.26). A filter is defined by a collection of weights that can be learned. The provided number of filters equals the number of feature maps [45].

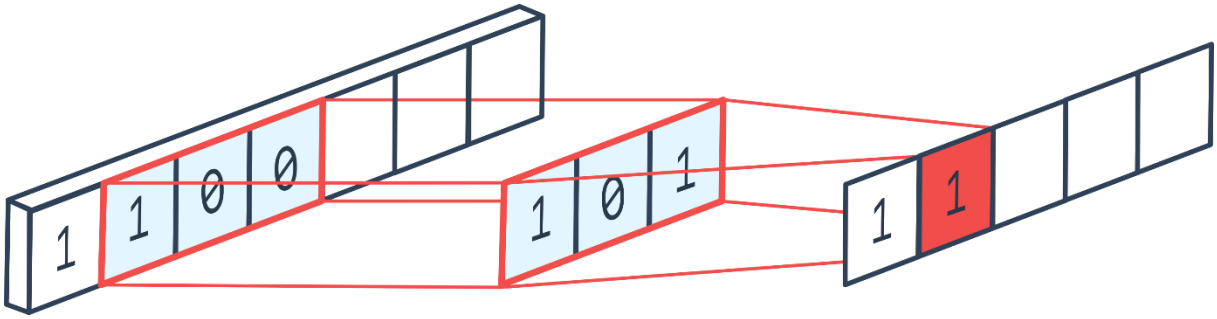

Figure 2.26 A 1D convolution with a kernel sized 3 and stride 1 *[62]*.

2.5.2.2 Filter/ Kernel

Each filter has some features such as corners and edges, and during the pass, the filter is slid over the width and height (depending to the stride parameter) of the input, forming a feature map for that filter. Each convolutional layer may have numerous kernels [45].

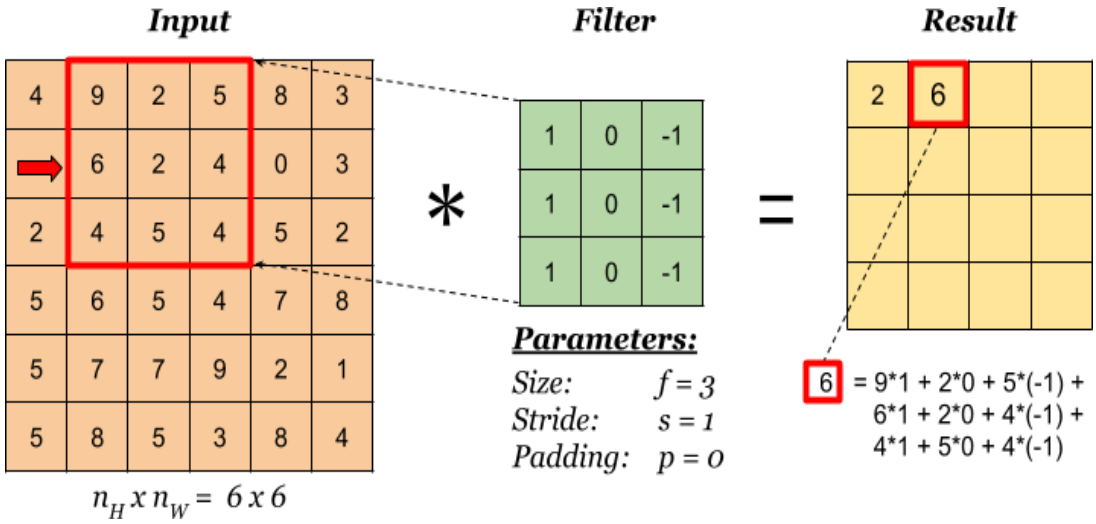

Figure 2.27: Example of Convolutional Operation *[90]*

A feature map is obtained after adding a bias term and then applying a nonlinear function to the output of the convolutional operation.

2.5.2.3 Hyperparameters

The convolutional and pooling layers have hyperparameters whose value must be defined beforehand, they are used to control the behavior of the model, here some important hyperparameters in the convolutional layer of the CNN:

a) Filter Size

Filer can take any size greater than 2 × 2; it should be less than the size of the input. The largest size used is 7 × 7 but only in the first convolutional layer, [49] a 2D convolutional filter will always have a third dimension in size. The third dimension is equal to the number of channels of the input image. For example, we apply a 3x3x1 convolution filter on gray-scale image that has 1 black and white channel like the previous example (Figure 2.27). We apply a 3x3x3 convolution filter on a colored image with 3 channels, Red, Green and Blue. In general, each image has dimensions W × H × D where W is the width in pixels, H is the height in pixels and D represent the dimension or the depth which is the number of channels.

b) Number of filters

There can be any reasonable number of filters, Google Net has 128 filters of 3 × 3 kernel size and 32 filter of 5 × 5 size, Alex Net used 96 filters of size 11 × 11 in the first convolution layer.

c) Stride

It specifies how many cells the filter must be moved in the input to calculate the next cell in the result, i.e., How many pixels must be moved at a time to create the filter's local receptive field (Figure 2.28), a little stride will result in an overlapping receptive field, whereas a large one will result in a smaller output dimension.

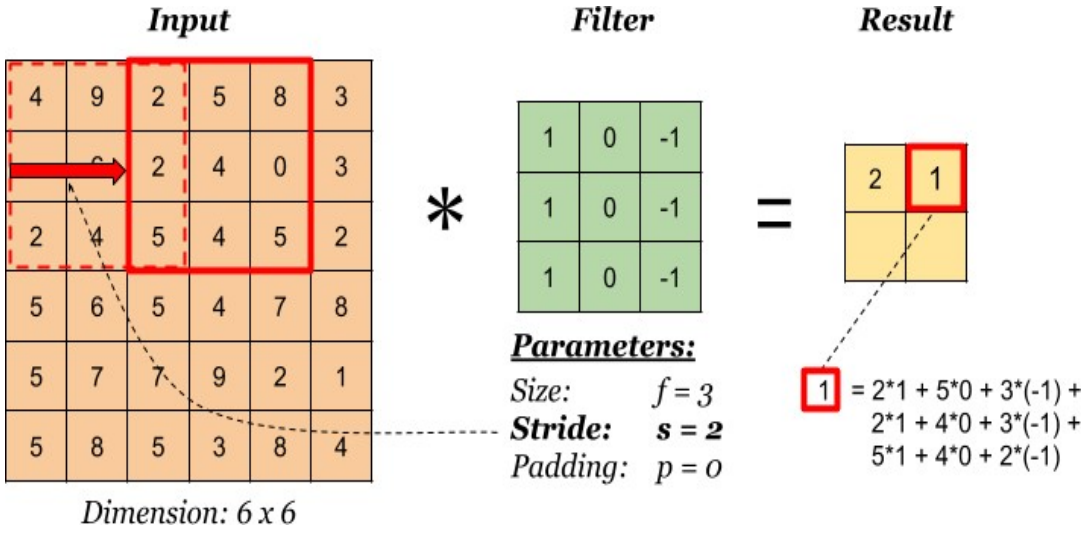

Figure 2.28: Filter with Stride (s) = 2 *[89]*

d) Zero padding

This hyperparameters describes the number of pixels to pad the input image (matrix), we add to the image a padding with p pixel (Figure 2.28). It helps to keep more of the information at the border of an image. Without padding, very few values at the next layer would be affected.

Notice that the dimension of the result has changed due to padding if we compare it withthe previous example (Figure 2.29).

Each filter in the convolution layer produces a feature map of size ([A − K + 2P]/S) +1, where: A the input volume size, K size of the filter, P the number of paddings applied and Sthe stride.

Suppose the input image has size 6 × 6 × 3, and 3 filters of size 3 × 3 are applied, where stride s = 1 and padding p = 0), we already say that the number of feature maps generated equal to the number of filters/kernels applied i.e., 5, the size of each feature map $\left(\frac{[6-3+0]}{1}\right)+1$ #3, therefore, the output volume will be 4 × 4 × 3. Convolution of 3D image will give a 2D output:

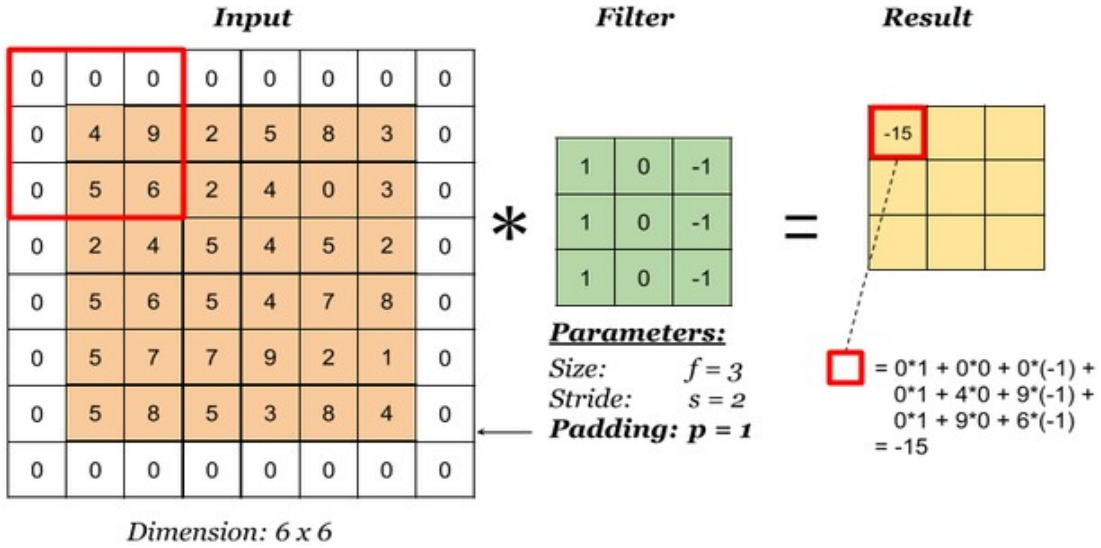

Figure 2.29: Zero Padding example with (p)=1 *[89]*

2.5.2.4 Pooling layer

Pooling layers, also known as subsampling layers, have no learnable parameters, such as weights or bias units. In CNNs, the sequence of convolution layers and activation function layers is followed by an optional pooling layer [64] to reduce the spatial size of the input and thus the number of parameters in the network. By down sampling each feature map output from

the convolutional layer and summarizing a section of neurons in the convolution layer, the pooling layer makes the model more resistant to alterations in the position of the features in the input image. Many pooling techniques are employed, the most common of which are maximum and average pooling. (See Figure 2.30.) It is a pooling operation that selects the maximum element (value) from the region of the feature map covered by the filter; the result of max-pooling layer is a feature map that contains the most relevant features of the preceding feature map while discarding less significant data. Instead of taking the maximum value from the input matrix, average pooling calculates the average [45].

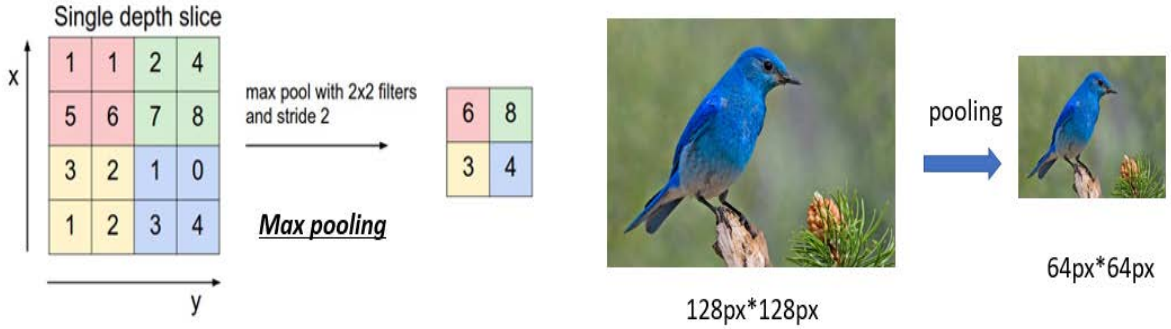

Figure 2.30: Max Pooling and Avg Pooling

2.5.3 Fully Connected Layer

A fully connected layer follows the previous sequence (Convolutional Layer, Pooling Layer) in ConvNets. Two stages make up a convolutional neural network: The feature extraction stage includes the stack of convolutional layers and the pooling layer, whereas the classification stage includes the fully connected layer (one or more) followed by a SoftMax function layer. The first part's major task is to extract enough features from the input photos. The last layer, which is most likely made up of Softmax functions, will calculate the probability that these features reflect each class, resulting in a class score. Every neuron in the preceding layer (convolution layer, pooling layer, or fully connected layer) is connected to every neuron in the following layer, and every value helps predict how strongly a value matches a specific class. (See Figure 2.31) Fully connected layers can learn more complex feature combinations. Softmax and Support Victor Machines (SVMs) are the two main classifiers used in CNNs. As previously stated, Softmax produces the probabilities of each class with a total probability of 1, whereas SVM produces the class scores, with the class with the greatest score being treated as the right one [45].

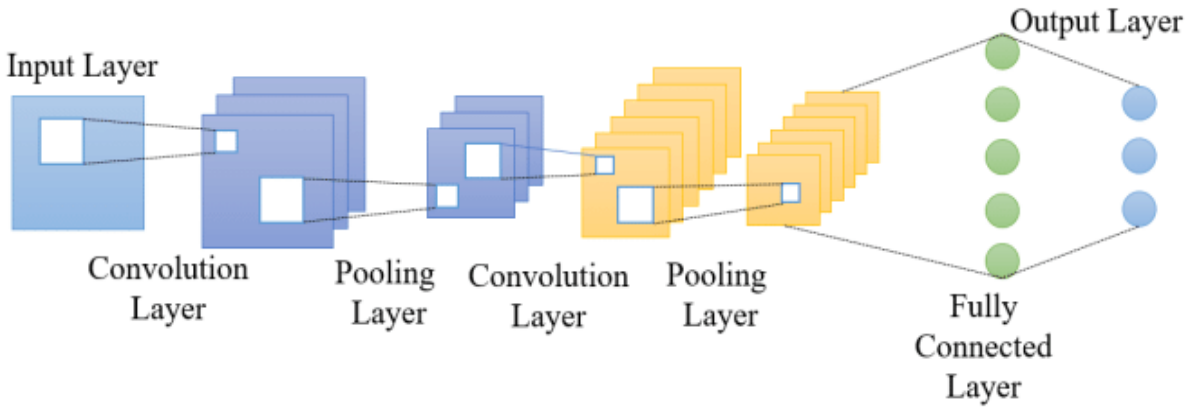

Figure 2.31: Connection between Convolutional Layer and Fully Connected Layer

2.5.4 Real world applications of Convolutional neural network

Convolutional neural networks (CNNs) are remarkably successful in many computer vision tasks. some real-world applications of CNNs including:

- Face detection: CNNs have been used to detect faces within images. The network takes an image as the input and produces a set of values that represent characteristics of faces or facial features at different parts of the image.
- Facial emotion recognition: CNNs have been used to help distinguish between different facial expressions such as anger, sadness, or happiness. CNNs can also be adapted to perform well with various lighting conditions and angles of faces within images.
- Object detection: CNN has been applied to object recognition across images by classifying objects based on shapes and patterns found within an image. CNN models have been created that can detect a wide range of objects from everyday items such as food, celebrities, or animals to more unusual ones including dollar bills and guns. Object detection is performed using techniques such as semantic or instance segmentation. CNNs have been used to localize and identify objects within images as well as create different views of those objects such as for use in drones or self-driving cars.
- Handwritten character recognition: CNNs can be used to recognize handwritten characters. CNNs take the image of a character as an input and break it down into smaller sections, identifying points that can connect or overlap with other points in order to determine the shape of the larger character. CNN models have been created that are able to identify different languages including Chinese, Arabic, and Russian even when they’re written differently.

- X-ray image analysis: CNNs have been used for medical imaging to identify tumors or other abnormalities in X-ray images. CNN models can take an image of a human body part, such as the knee, and determine where within that image there might be a tumor based on previous similar images processed by CNN networks. CNN models can also be used to determine abnormalities from X-ray images [65].

2.5.5 Advantages and disadvantages of ANNs and CNNs

	ANN	CNN
Advantages	• Storing information on the entire network. • Ability to work with incomplete knowledge. • Having fault tolerance. • Having a distributed memory.	• Very High accuracy in image recognition problems. • Automatically detects the important features without any human supervision. • Weight sharing.
Disadvantages	• Hardware dependence. • Unexplained behavior of the network. • Determination of proper network structure	• CNN do not encode the position and orientation of object. • Lack of ability to be spatially invariant to the input data. • Lots of training data is required.

Table 2.1: Advantages and disadvantages of ANNs and CNNs

2.6 Conclusion

In this chapter, we have discussed the important aspects of CNN, artificial intelligence and machine learning as well as different types of machine learning in detail. Deep learning and neural networks have been investigated, ranging from traditional artificial neurons, neural networks, artificial neural networks and artificial neural network models, and we have explained well the activation functions, the next chapter will deal with the time-series prediction method and some prediction results.

CHAPTER 3:

Convolutional Neural Networks For Time Series Forecasting

Chapter 3: Convolutional Neural Networks For Time Series Forecasting

3.1 Introduction

In this chapter we will build different time series forecasting models based on convolutional neural networks and train and test it on the datasets we chose, afterwards we measure the performance of our model using different baselines and metrics to find the appropriate model for our forecasting.

3.2 Proposed time series forecasting method

CNNs deal with time series problems effectively. Their ability to learn and automatically and extract features from raw input data can be applied to time series forecasting problems, recent studies which applied CNN to time-series forecasting tasks mainly involving financial data show promising results [4].

3.3 Implementation

This section is dedicated to an overview of programming language, libraries, tools and the hardware used overall for the implementation.

3.3.1 Python

Python is an interpreted object-oriented programming language invented around 1991 intended to emphasize code readability, it supports modules and packages, which encourages program modularity and code reuse.

Python is the convenient language to quickly implement an abstract idea, bring together different libraries to do something new, process and scrape data from the internet [66].

3.3.2 Matplotlib

Matplotlib is an open-source Python library used for visualizations of data. It provides a wide range of different plots [14].

3.3.3 Scikit-learn

Scikit-learn is an open-source software library that provides tools for data analysis. There are classes for pre-processing and overall machine learning problems, such as classification, regression, clustering or dimensionality reduction. It is built on NumPy and Matplotlib [67].

3.3.4 Pandas

Pandas is another open-source library for Python used for data analysis. It is good for data importing. Its class DataFrame is an excellent method for representing tabular data, assisting in data pre-processing, modification or slicing [68].

3.3.5 StatsModels

StatsModels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration.

3.3.6 Keras

Keras is a deep-learning framework built on top of python that provides high-level building blocks for developing almost any kind of deep-learning model in a much more convenient way than to build it all from scratch. Keras also allows the same code to be run on both the CPU and GPU [42].

3.3.7 Colaboratory

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs [69].

3.3.8 Hardware

For implementation, we used a windows machine with an Intel i5 CPU 8th generation, 8 GB of Ram and SSD 256 GB.

3.4 Experiments

Experiments were done with univariate and multivariate CNN models on two datasets, different metrics and baseline were used to evaluate the results of the forecast.

3.4.1 Datasets

For our experiments we used two time series datasets retrieved from Kaggle.

- The first dataset is on daily new Coronavirus (Covid-19) cases and deaths in the United States from January 2020 to Mai 2022 (Figures 3.1 and 3.2), The data is the product of dozens of journalists working across several time zones to monitor news conferences, analyze data releases and seek clarification from public officials on how they categorize cases [70].

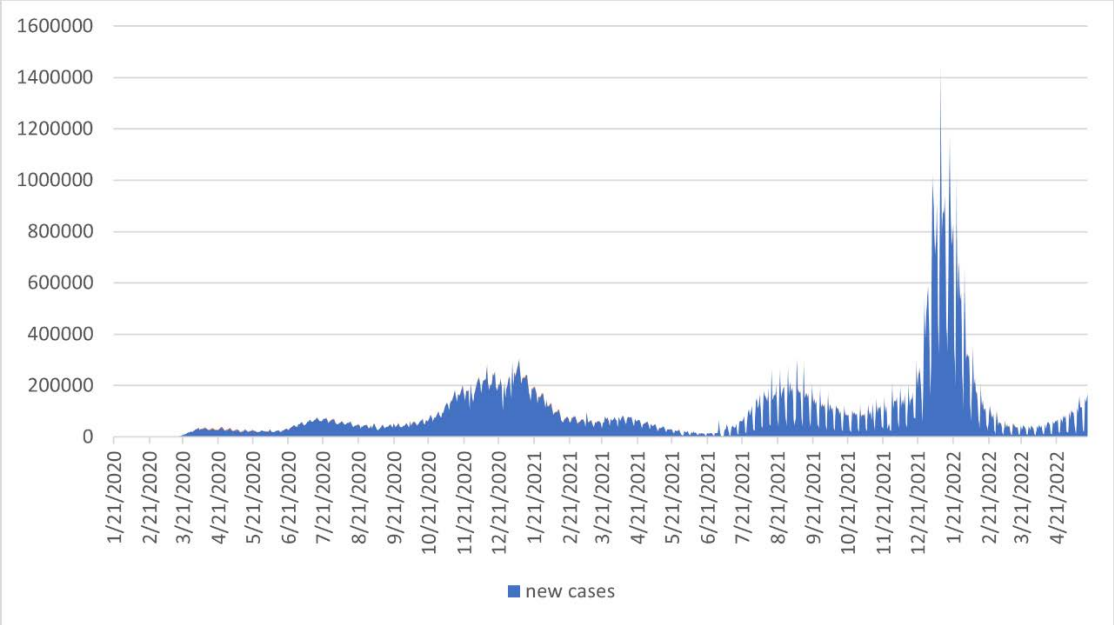

Figure 3.1 US new (Covid-19) confirmed cases *[70]*

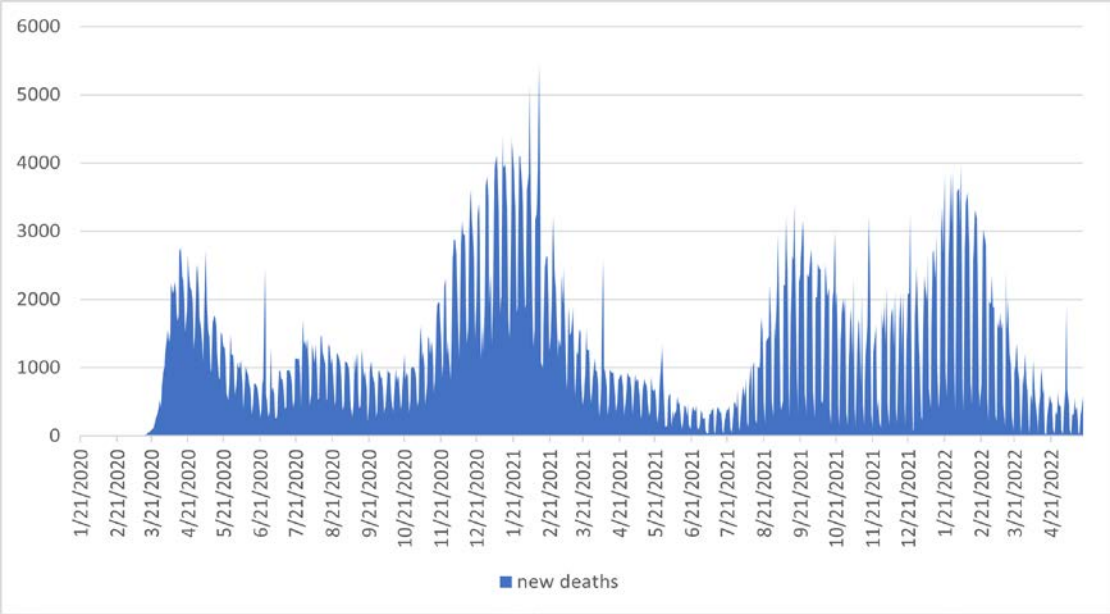

Figure 3.2 US new (Covid-19) confirmed deaths *[70]*

- The second dataset is a multivariate time series of accumulated property sales data for the 2007-2019 period, for one specific region. The data contains sales prices in us dollars for houses and units with 1,2,3,4,5 bedrooms. These are the cross-depended variables, Raw sales data was transformed to produce median price at quarterly intervals (Figure 3.3) [71].

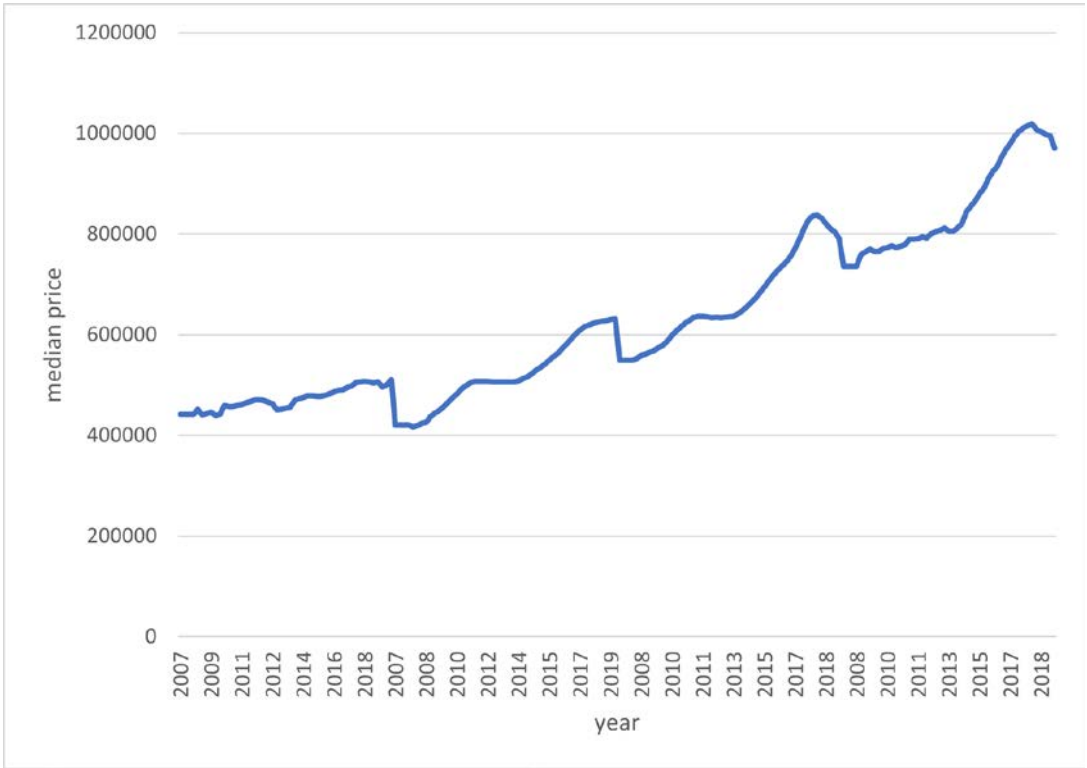

Figure 3.3 median price of property sales *[71]*

3.4.2 Implementation details

In the section we will describe the process of developing our CNN time series forecasting model implementation and data processing.

3.4.2.1 Data preparation

We first import the data from google drive cloud storage.

```
from google.colab import drive
drive.mount('/content/drive')
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/us.csv',index_col="Date")
```

Figure 3.4: Importing the data

After loading the data, we transform the time series to a supervised learning problem using the sliding window method. We can do this by using previous time steps as input variables and use the next time step as the output variable, this operation is done in the code below (Figure 3.5).

```
#data sequences
cases=data['new cases'].to_numpy()
T=10
X=[]
Y=[]
for t in range (len(cases)-T):
    x=cases[t:t+T]
    X.append(x)
    y=cases[t+T]
    Y.append(y)
X=np.array(X).reshape(-1,T)
Y=np.array(Y)
```

Figure 3.5: Code for data preparation

3.4.2.2 Train-Test Split

We split the loaded time series data into train and test sets 80% of the data for training set and 20% stays for testing.

This is done with the following lines of code.

```
#split 80/20 training/validation
Ntest=int(len(cases)*0.2)
Xtrain,Ytrain = X[:-Ntest],Y[:-Ntest]
Xtest, Ytest = X[-Ntest:],Y[-Ntest:]
```

Figure 3.6: Code for Train-Test Split

3.4.2.3 Building the CNN model

A one-dimensional CNN is a CNN model that has a convolutional hidden layer that operates over a 1D sequence. This is followed by a second convolutional layer, and then a pooling layer followed by a dense fully connected layer that interprets the features extracted by the convolutional part of the model. A fatten layer is used between the convolutional layers and the dense layer to reduce the feature maps to a single one-dimensional vector. We can define a 1D CNN Model for time series forecasting as follows.

```
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=2, activation='relu',
                 input_shape=(T, n_features), padding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2))
model.add(Flatten())
model.add(Dense(50, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mae')
```

Figure 3.7: code for building the CNN model

3.4.3 Evaluation Metrics

When we build a solution for any regression problem, we compare its performance with the existing work. But to compare the two works, there should be some standard metric, the evaluation metrics we used are MAE, MSE and MAPE.

3.4.3.1 MAE

Mean absolute error (MAE) is a fundamental and most used evaluation metric for regression problems. Here we try to calculate the difference between the actual and predicted values. This difference is termed an error, it can be positive or negative so we take the magnitude. Let's put *AV* as the actual value and *PV* as the predicted value,

so we have: $MAE = |AV - PV|$ [72].

3.4.3.2 MSE

Mean squared error (MSE) is a very popular evaluation metric for regression problems. It is similar to the mean absolute error, but the error is squared here, $MSE = (AV - PV)^2$ [72].

3.4.3.3 MAPE

The mean absolute percentage error (MAPE) is the mean or average of the absolute percentage errors of forecasts. Error is defined as actual or observed value minus the forecasted value. Percentage errors are summed without regard to sign to compute MAPE. This measure is easy to understand because it provides the error in terms of percentages. Also, because absolute percentage errors are used, the problem of positive and negative errors cancelling each other out is avoided. Consequently, MAPE has managerial appeal and is a measure commonly used in forecasting. The smaller the MAPE the better the forecast,

$MAPE = 100\% (|AV - PV| / AV)$ [72].

3.4.4 Baselines

A baseline model is essentially a simple model that acts as a reference in a forecast project. Its main function is to contextualize the results of trained models as it provides a point of comparison [73], the baselines we chose to benchmark our model are ARIMA, Zero Rule algorithm and the naïve forecasting algorithm.

3.4.4.1 ARIMA

ARIMA is the most popular baseline chosen in many projects, we described in in detail in chapter two.

3.4.4.2 Zero Rule algorithm

Zero Rule Algorithm for regression problems is to predict the central tendency. This could be the mean or the median, (also called the average) of the output value observed in the training data. This is likely to have a lower error than random prediction which will return any observed output value [73].

3.4.4.3 Naïve forecasting algorithm

Naïve forecasting is the technique in which the last period's sales are used for the next period's forecast without predictions or adjusting the factors. Forecasts produced using a naïve approach are equal to the final observed value [74].

3.5 Results and comparison

3.5.1 Results

To make the comparison and evaluation easier we classified the results into the univariate time series and the multivariate time series.

3.5.1.1 Forecast results using univariate time series

In Figure 3.8 we have the results from the univariate model, trained with the first dataset, the output is the predicted new Covid-19 confirmed cases plotted along with test set to show the outcome of the forecast.

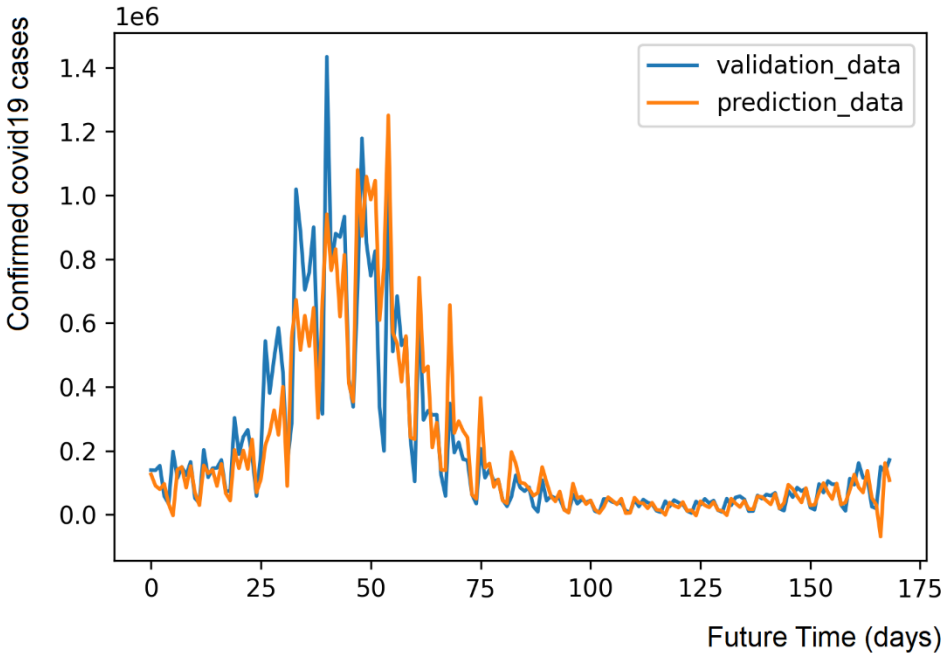

Figure 3.8: univariate model results for the COVID19 dataset prediction

The second univariate model used the property sales dataset, it has one feature which is the median sales price of the properties, results are shown in figure 3.9 along with the test set.

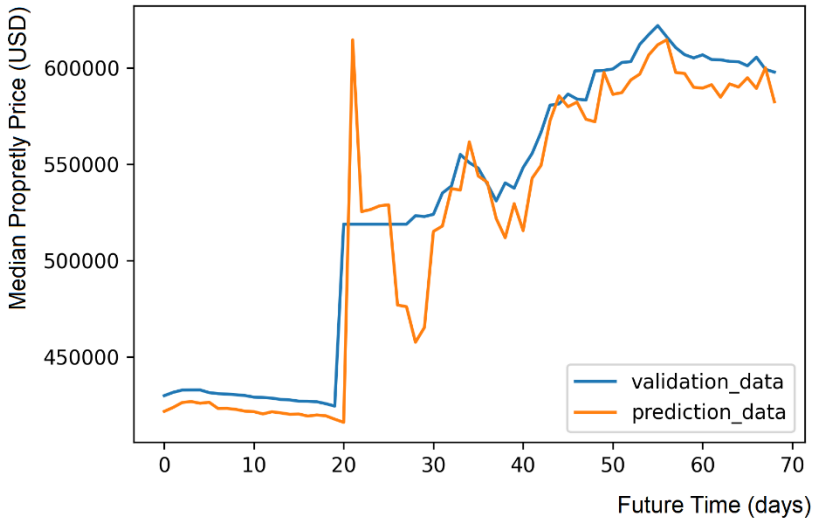

Figure 3.9: univariate model results for the sales dataset prediction

3.5.1.2 Forecast results using multivariate time series

We have the results from the multivariate model, this was trained with one more feature which is the new Covid-19 confirmed deaths, predictions of the new confirmed Covid-19 cases are paired with test set in figure 3.10.

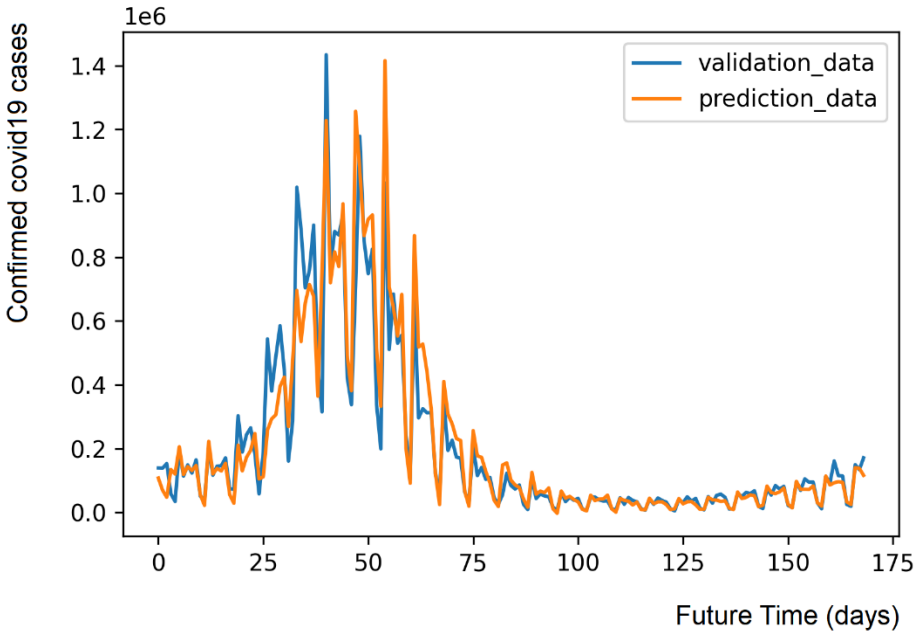

Figure 3.10: multivariate model results for the COVID19 dataset prediction

And these are the results for the property sales dataset shown in figure 3.11 for the multivariate model, which was trained with an additional feature which is the number of bedrooms of each property sold.

Figure 3.11: multivariate model results for the sales dataset prediction

We clearly notice from the graphs that the multivariate version of the model is outperforming the univariate model, now let's confirm that by moving on to the metrics comparison.

3.5.2 Performance Comparison

To measure how well our model performed we use the metrics and compare the results between the baselines and our CNN models.

Metrics and baselines for the Covid-19 dataset:

	MSE	MAE	MAPE
Univariate CNN model	13568179924.31	61740.32	57.39%
Multivariate CNN model	9701514846.04	50776.95	33.60%
ARIMA	90371021379.40	163118.01	224.29%
Naïve forecasting	35476955073.29	96146.33	67.35%
Zero rule forecasting	90468199050.99	163669.10	226.00%

Table 3.1: Baseline and CNN models metrics for the Covid-19 dataset

Metrics and baselines the property sales dataset:

	MSE	MAE	MAPE
Univariate CNN model	576005790.95	15052.95	2.98%
Multivariate CNN model	4657.88	4657.88	0.91%
ARIMA	2855425692.11	47692.08	9.37%
Naïve forecasting	157049577.17	4500.82	5.82%
Zero rule forecasting	5773427546.87	61892.79	11.11%

Table 3.2: Baseline and CNN models metrics for the property sales dataset

Looking at the metrics our most accurate predictions is the multivariate CNN model followed by the univariate CNN model, this outcome applies on both datasets.

3.5.3 Discussion

The outcome of the experiments done on the datasets insight that CNNs can handle time series data pretty well, the univariate CNN model for the Covid-19 dataset outperformed the ARIMA and Zero rule forecasting baselines with four times more in the MAPE value, but the results were close with the Naïve forecasting baseline with only 10% MAPE difference, and closer MSE, MAE values than the other baselines. It is worth noting that the smaller the value of MAPE the better the forecast. For the multivariate CNN model there is a noticeable increase in performance as the MAPE was two times better than the univariate model and the naïve forecasting model.

For the experiments on the second dataset the results were relatively similar to the first one. According to the MAPE values, the univariate CNN model is 8% better than the Zero rule forecasting algorithm, 6% better than the ARIMA model 3% better than the naïve forecasting. The results for the multivariate CNN model were really promising as we got the smallest MAPE value of 0.91% which is a significant increase in accuracy compared the univariate model.

We can see from these results that our CNN models performed pretty well compared to the other baselines especially in the sales dataset, and the more features we add to the model the more accurate the predictions are, this prove that deep learning forecasting models have an advantage over the traditional statistical models when there are more data and variables available.

Also, we think further testing is important, more datasets with different characteristics should be applied to better understand and further validate the results and characteristics for each model.

3.6 Conclusion

In this chapter, we implemented CNN univariate and multivariate models for time series forecasting and measured its performance based on the metrics and baselines. The study performed about CNN models applied to time series problems showed that they are a viable option for time series. It was also showed what are the desired characteristics that a model should have to be a viable option.

General Conclusion

Time series analysis were always a field extremely important across several areas, thus it was always under intense development. Supported by the recent advances in the deep learning area the objective of this work was to make a study on deep learning strategies for the time series domain.

Initially we discussed the aspects of time series data, analysis, and its components, as well as the various forecasting techniques in detail. Different types of models have been investigated, ranging from traditional statistical models to machine learning and deep learning models, and common evaluation metrics that are commonly used in time series have been illustrated.

We discussed the theory behind the artificial intelligence and machine learning as well as different types of machine learning in detail, ranging from traditional artificial neurons, neural networks and deep neural network models focusing on CNNs and their architecture.

We implemented on the basis of the metrics and baselines different CNN models including univariate and multivariate models for time series forecasting and evaluated its performance, we had some promising results especially with multivariate models.

The research done on CNN models used to solve time series problems revealed that they are a good choice for time series forecasting and they are capable to compete and outperform the traditional models, this offers a great opportunity to try complex deep learning models like the hybrid models on more datasets with different characteristics and larger data to explore their potential.

With the growing interest in exploring deep learning’s predictive capabilities, we expect it to dominate the forecasting field in the near future.

Bibliography

[1] M. D. Pra, *Time Series Forecasting with Deep Learning and Attention Mechanism | by Marco Del Pra | Towards Data Science*, 2020.

[2] Q. Yang and X. Wu, "10 challenging problems in data mining research," *International Journal of Information Technology & Decision Making*, vol. 5, p. 597–604, 2006.

[3] C. Stolte, *Time Series Forecasting: Definition & Examples | Tableau*, 2003.

[4] J. Brownlee, Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python, Machine Learning Mastery, 2018.

[5] J. V. Hansen and R. D. Nelson, "Forecasting and Recombining Time-Series Components by Using Neural Networks," *The Journal of the Operational Research Society*, vol. 54, p. 307–317, 2003.

[6] G. P. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," *European Journal of Operational Research*, vol. 160, pp. 501-514, 2005.

[7] E. Ghysels and D. R. Osborn, The Econometric Analysis of Seasonal Time Series, Cambridge University Press, 2001.

[8] Dataiku, *Concept: Time {Series} {Components} — {Dataiku} {Knowledge} {Base}*, 2022.

[9] J. Crespo Cuaresma, J. Hlouskova, S. Kossmeier and M. Obersteiner, "Forecasting electricity spot-prices using linear univariate time-series models," *Applied Energy*, vol. 77, pp. 87-106, January 2004.

[10] F. Ndiritu, *Univariate Time Series using Facebook Prophet | Engineering Education (EngEd) Program | Section*, 2022.

[11] A. Singh, *Multivariate Time Series | Vector Auto Regression (VAR)*, 2018.

[12] D. Kwiatkowski, P. C. B. Phillips, P. Schmidt and Y. Shin, "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?," *Journal of Econometrics*, vol. 54, pp. 159-178, 1992.

[13] J. Emery, *3 Advantages to Time Series Analysis and Forecasting | phData*, 2021.

[14] Matplotlib, *Matplotlib — Visualization with Python*, 2022.

[15] G. Chevillon, "DIRECT MULTI-STEP ESTIMATION AND FORECASTING," *Journal of Economic Surveys*, vol. 21, pp. 746-785, 2007.

Bibliography

[16] J. Korstanje, *How to Select a Model For Your Time Series Prediction Task [Guide] - neptune.ai*, 2021.

[17] G. Jain and B. Mallick, "A Study of Time Series Models ARIMA and ETS by Garima Jain, Bhawna Mallick :: SSRN," *A Study of Time Series Models ARIMA and ETS by Garima Jain, Bhawna Mallick :: SSRN*, June 2017.

[18] "Modeling Financial Time Series with S-PLUS®," 2006.

[19] D. A. Freedman, Statistical Models, 2012.

[20] H. L. Seal, The Historical Development of the Gauss Linear Model, Yale University, 1968.

[21] A. C. Rencher and W. F. Christensen, Methods of Multivariate Analysis, 2012.

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, p. 273–297, September 1995.

[23] U. Thissen, R. van Brakel, A. P. de Weijer, W. J. Melssen and L. M. C. Buydens, "Using support vector machines for time series prediction," *Chemometrics and Intelligent Laboratory Systems*, vol. 69, pp. 35-49, 2003.

[24] H. Soh and Y. Demiris, "Spatio-Temporal Learning With the Online Finite and Infinite Echo-State Gaussian Processes," *IEEE transactions on neural networks and learning systems*, vol. 26, June 2014.

[25] C. Deb, F. Zhang, J. Yang, S. E. Lee and K. W. Shah, "A review on time series forecasting techniques for building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 74, pp. 902-924, 2017.

[26] N. S. Altman, "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression," *The American Statistician*, vol. 46, pp. 175-185, 1992.

[27] J. Jagwani, M. Gupta, H. Sachdeva and A. Singhal, "Stock Price Forecasting Using Data from Yahoo Finance and Analysing Seasonal and Nonseasonal Trend," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2018.

[28] d. Hudson, Agricultural Markets and Prices, 2007.

[29] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, 2016.

[30] D. Ciresan, U. Meier and J. Schmidhuber, "Multi-column deep neural networks for image classification," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012.

[31] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*,

2012.

[32] T. Menzies, E. Kocaguneli, B. Turhan, L. Minku and F. Peters, Sharing Data and Models in Software Engineering, Morgan Kaufmann, 2014.

[33] A. Mohanty, *Multi layer Perceptron (MLP) Models on Real World Banking Data | by Awhan Mohanty | Becoming Human: Artificial Intelligence Magazine,* 2019.

[34] P. J. Werbos, "Backpropagation Through Time: What It Does and How to Do It," *Proc. IEEE,* vol. 78, pp. 1550-1560, 1990.

[35] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 06, p. 107–116, April 1998.

[36] R. Dey and F. M. Salem, *Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks,* 2017.

[37] R. Fu, Z. Zhang and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC),* November 2016.

[38] M. Ciortan, *Gentle introduction to Echo State Networks | by Madalina Ciortan | Towards Data Science,* 2019.

[39] P. L.-B. tez, M. Carranza-García and J. C. Riquelme, "An Experimental Review on Deep Learning Architectures for Time Series Forecasting," *International Journal of Neural Systems,* vol. 31, p. 2130001, February 2021.

[40] Y. Wang, M. Long, J. Wang, Z. Gao and P. S. Yu, "PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs," in *Advances in Neural Information Processing Systems,* 2017.

[41] A. Xavier, *An introduction to ConvLSTM. Nowadays it is quite common to find... | by Alexandre Xavier | Neuronio | Medium,* 2019.

[42] L. Cunha, "Deep learning with Python (2a ed) - François Chollet - Manning, outubro 2021, 504 pp.," *Interações: Sociedade e as novas modernidades,* p. 113–115, June 2022.

[43] H. Chen, "Machine learning for information retrieval: Neural networks, symbolic learning, and genetic algorithms," *Journal of the American Society for Information Science,* vol. 46, p. 194–216, April 1995.

[44] D. MICHIE, "``Memo" Functions and Machine Learning," *Nature,* vol. 218, p. 306–306, April 1968.

Bibliography

[45] M. A. Wani, F. A. Bhat, S. Afzal and A. I. Khan, "Advances in Deep Learning," *Studies in Big Data,* 2020.

[46] G. E. Hinton and T. J. Sejnowski, "Unsupervised learning : foundations of neural computation," 1999.

[47] B. J. Pandian and M. M. Noel, "Control of a bioreactor using a new partially supervised reinforcement learning algorithm," *Journal of Process Control,* vol. 69, p. 16–29, September 2018.

[48] R. A. Alzahrani and A. C. Parker, "Neuromorphic Circuits With Neural Modulation Enhancing the Information Content of Neural Signaling," *International Conference on Neuromorphic Systems 2020,* July 2020.

[49] N. Buduma and N. Locascio, Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms, 1st ed., O'Reilly Media, Inc., 2017.

[50] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities.," *Proceedings of the National Academy of Sciences,* vol. 79, p. 2554–2558, April 1982.

[51] R. Gour, *Artificial Neural Network for Machine Learning — Structure & Layers,* 2019.

[52] M. Garduño-Ramón, L. Morales-Hernández, J. Benitez-Rangel, R. Osornio-Rios and J. Sánchez-Gómez, "Methodology for automatic detection of trees and shrubs in aerial pictures from UAS," 2015.

[53] S. Pattanayak, Pro Deep Learning with TensorFlow, 2017.

[54] S. H. Han, K. W. Kim, S. Kim and Y. C. Youn, "Artificial Neural Network: Understanding the Basic Concepts without Mathematics," *Dementia and Neurocognitive Disorders,* vol. 17, p. 83, 2018.

[55] V. S. Bawa and V. Kumar, "Linearized sigmoidal activation: A novel activation function with tractable non-linear characteristics to boost representation capability," *Expert Systems with Applications,* vol. 120, p. 346–356, April 2019.

[56] J. Patterson and A. Gibson, Deep Learning: A Practitioner's Approach, Beijing: O'Reilly, 2017.

[57] W. Ballard, *Hands-On Deep Learning for Images with TensorFlow,* 2018.

[58] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network," in *Advances in Neural Information Processing Systems,* 1989.

Bibliography

[59] S. Raschka and V. Mirjalili, Python Machine Learning - Second Edition, 2017.

[60] A. N. Shoutko, *Immunity or morphogenesis in cancer development and treatment*, vol. 6, Open Access Text Pvt, Ltd., 2019.

[61] B.-S. Yang, T. Han and Z.-J. Yin, "Fault Diagnosis System of Induction Motors Using Feature Extraction, Feature Selection and Classification Algorithm," *JSME International Journal Series C*, vol. 49, pp. 734-741, January 2006.

[62] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar and P. A. Muller, *Deep learning for time series classification: a review - Data Mining and Knowledge Discovery*, 2019.

[63] I. Gogul and V. S. Kumar, "Flower species recognition system using convolution neural networks and transfer learning," in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, 2017.

[64] M. A. Nielsen, *Neural networks and deep learning*.

[65] A. Kumar, *Real-World Applications of Convolutional Neural Networks - Data Analytics*, 2021.

[66] C. Vudhya, *AI VS ML VS DL-Let's Understand The Difference - Analytics Vidhya*, 2021.

[67] *scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation*, 2022.

[68] Pandas, *pandas - Python Data Analysis Library*, 2022.

[69] C. Google, *Google Colaboratory*, 2022.

[70] J. Hanson, *Coronavirus (Covid-19) maps of United States*, 2022.

[71] H. Holdings, *House Property Sales Time Series*, 2019.

[72] R. Agrawal, *Evaluation Metrics for Your Regression Model - Analytics Vidhya*, 2021.

[73] J. Brownlee, *How to implement Baseline Machine Learning algorithms from scratch with python. Machine Learning Mastery*, 2020.

[74] J. Brownlee, *Evaluate naive models for forecasting household electricity consumption. Machine Learning Mastery*, 2020.

[75] s. significance, *statistical significance - Justifying the trend component in a time series? - Cross Validated*, 2019.

[76] A. L. Lima, *Bitcoin Price Prediction Using Recurrent Neural Networks and LSTM*, 2021.

[77] C. Lynx, *Annual Canadian Lynx trappings 1821–1934 — lynx*.

[78] B. Wise, *Time Series Analysis with ARIMA: Part 2 - Cisco Blogs*, 2020.

[79] L. S. T. M. Networks, *Understanding LSTM Networks – colah's blog*, 2015.

Bibliography

[80] W. Commons, *File:Gated Recurrent Unit, type 1.svg - Wikimedia Commons*, 2018.

[81] S. Vivek, *Basic architecture of RNN and LSTM*, 2017.

[82] I. N. T. E. L. L. I. G. E. N. C. E. ARTIFICIELLE, *INTELLIGENCE ARTIFICIELLE, MACHINE LEARNING, BIG DATA... QUELS IMPACTS POUR NOTRE FUTUR ? - BGS Associés*, 2019.

[83] R. Stefanus, *Conventional Programming VS Machine Learning | by Ruben Stefanus | Medium*, 2019.

[84] r. notes, *reading-notes.*

[85] W. Wikimedia, *File:Neuron.jpg - Wikimedia Commons.*

[86] M. H. Esfe, *A well-trained artificial neural network for predicting the rheological behavior of MWCNT–Al2O3 (30–70%)/oil SAE40 hybrid nanofluid - Scientific Reports*, 2021.

[87] D. Dan, *Neural Network Workflow - Concord Analytics, LLC*, 2019.

[88] H. Bommana, *Loss Functions Explained. Intuitive explanations of various Loss... | by Harsha Bommana | Deep Learning Demystified | Medium*, 2020.

[89] P. ProjectPro, *Introduction to Convolutional Neural Networks Architecture*, 2022.

[90] V. Kumar, *Convolutional Neural Networks. Basic fundamentals of CNN | by Vedant Kumar | Towards Data Science*, 2020.

[91] S. Saha, *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, 2018.

[92] I. IndianTechWarrior, *7 Types of Neural Network Activation Functions: How to Choose? – IndianTechWarrior*, 2021.

[93] F. convolution, *4 – Filtres de convolution – Mathinfo*, 2022.