UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

Forecasting exchange rate with deep learning algorithms: the US Dollar (USD) to Colombian Peso (COP) case

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

Autora:

Lucero Burbano Alejandra Valeria

Tutor:

PhD - Cuenca Pauta Erick Eduardo

Urcuquí, Abril 2024

Autoría

Autorización de publicación

Dedicatoria

Este proyecto de tesis está dedicado principalmente a Dios por darme vida, salud y permitirme llegar a este momento tan importante en mi formación profesional, a mis familiares y amigos. De manera especial:
A mis padres, Por su amor y apoyo incondicional. Gracias por guiarme durante estos años. Siempre estaré eternamente agradecida por inculcarme los mejores valores. Ustedes son la razón de mi crecimiento y mejora. Aunque papá ya no estés conmigo, te dedico este logro que, en algún momento, prometí cumplir. ¡Los quiero mucho!
A mi hermana, por siempre hacerme reír y ser esa fuente de alegría en mi diario vivir. Hemos vivido muchos momentos inolvidables y espero que en el futuro puedas contar con mi apoyo incondicional.
Para mis amigos, Sherald, Carlitos, Stalyn, Mateo y Leonel ustedes fueron parte fundamental de mi vida estudiantil. Gracias por apoyarme y animarme siempre a seguir adelante. Liss, Anabel y Geme, gracias por demostrarme que una buena amistad perdura aunque hayan kilometros de por medio
A mi pareja, Gracias por tu apoyo incondicional en cada momento de mi vida y por animarme cuando sentí que no podía más. Gracias por darme al amor de mi vida, nuestra hija, quien también ha sido mi apoyo y motivo para culminar esta etapa de mi vida. Muchas gracias a todos por su apoyo incondicional.

Alejandra Valeria Lucero Burbano

Agradecimiento

En primer lugar, quisiera agradecer a mi asesor y profesor, Erick Cuenca, por toda su paciencia y apoyo durante este proceso de tesis. Asimismo, quisiera expresar mi agradecimiento a todos los profesores que tuve durante mi carrera en Yachay Tech (YT), especialmente al profesor Rigoberto Foncesa, por ser quien supo apoyarme en los momentos difíciles de mi carrera y me motivó para continuar. Finalmente, quiero agradecer a cada una de las personas que hicieron más llevadera mi vida universitaria, especialmente durante mi último año. Gracias infinitas a todos.

Alejandra Valeria Lucero Burbano

Resumen

La predicción de series temporales puede proporcionar información vital para ayudarnos a tomar mejores decisiones. Cuanto mayor sea la profundidad y amplitud de nuestro estudio, mayor será la calidad de la información que adquirimos. Este proyecto de tesis compara tres modelos de aprendizaje profundo (ANN, LSTM, GRU) para la predicción del tipo de cambio USD/COP. Para determinar cuál de los tres modelos es el apropiado para predecir este tipo de cambio, se propone realizar 3 experimentos y utilizar MAE y R^2 como métricas. El primero tiene como objetivo elegir qué tamaño de conjunto de datos tiene el mejor rendimiento. El segundo tiene como objetivo elegir el tamaño del conjunto de datos, la relación entrenamiento-prueba, la configuración del tamaño del lote y el mejor modelo. Finalmente, tomamos las mejores configuraciones de los experimentos anteriores para probar nuestro modelo con un conjunto de datos desconocido. Como resultado de la realización de los 3 experimentos anteriores se obtuvo que el mejor modelo para predecir este tipo de cambio fue ANN en función de las métricas R^2 y MAE.

Palabras Clave:
Series de tiempo, Predicción, Aprendizaje Profundo, Redes Neuronales Artificiales, Memoria a largo y corto plazo, Unidad recurrente cerrada, Error absoluto medio, Coeficiente de determinación R-cuadrado.

Abstract

Time series prediction can provide vital information to help us make better decisions. The greater the depth and breadth of our study, the greater the quality of the information we acquire. This thesis project compares three deep learning models (ANN, LSTM, GRU) to predict the USD/COP exchange rate. To determine which of the three models is appropriate to predict this type of change, it is proposed to perform 3 experiments and use MAE and R^2 as metrics. The first aims to choose which data set size has the best performance. The second aims to choose the data set size, train-test ratio, batch size configuration, and the best model. Finally, we take the best configurations from the previous experiments to test our model on an unknown data set. As a result of the 3 previous experiments, it was obtained that the best model to predict this type of change was ANN based on the R^2 and MAE metrics.

Keywords:
Time Series, Forecasting, Deep Learning, Artificial Neural Networks, Long-Short Term Memory, Gated Recurrent Unit, Mean Absolute Error, coefficient of determination R-squared

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Exchange rates are essential indicators in global economics because they influence international commerce, investment choices, and the dynamics of financial markets. Forecasting exchange rates with time series analysis entails evaluating past data trends to predict future currency pair movements. To capture the patterns, seasonality, and volatility inherent in exchange rate data, this forecasting approach employs a variety of statistical models, machine learning algorithms, and hybrid techniques [9]. Additionally, time series forecasting methods such as Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing, and machine learning algorithms such as Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks examine historical exchange rate data to identify patterns and build predictive models. These models seek to represent the intrinsic complexity of currency fluctuations by considering elements such as economic data, interest rates, geopolitical events, and market emotion. Accurate exchange rate forecasting using time series analysis is critical for stakeholders in various industries. Investors use these estimates to make educated decisions about currency trading and portfolio management and by companies to avoid the risks associated with currency changes in international transactions. Governments and central banks use Exchange rate projections to establish monetary policies and estimate the economic impact of currency fluctuations on their respective countries. However, the market's complexity and volatility make forecasting exchange rates difficult. Currency changes are difficult to forecast due to unforeseen geopolitical developments, unexpected economic swings, and the interconnectedness of global markets. Despite these limitations, continual advances in analytical methodologies and data-driven approaches continue to improve the accuracy and reliability of exchange rate projections, assisting stakeholders in navigating the volatile global financial landscape.

1.2 Problem statement

The exchange rate is one of the most crucial aspects of the economy. Not only is it essential for macroeconomics as one of the critical indicators, but it is also essential at the micro level for the individuals who make up the economy, such as global traders or even small farmers, as it can impact the economy. Forecasting exchange rates and enabling accurate expectations among economic actors is essential. Central banks, which govern monetary policy, are the primary users of these forecasting models. While technology constantly evolves, searching for and testing alternative approaches and exploring new opportunities is excellent.

For the development of this work, we use the USD/COP as a currency exchange since our country, which has the dollar (USD) as its official currency, is limited to the north with Colombia. This country has the Colombian peso (COP) as its official currency, and exchanging their currencies could directly affect trade in this area.

1.3 Objectives

1.3.1 General Objective

This project aims to predict the future values of the USD/COP exchange rate using different deep-learning algorithms.

1.3.2 Specific Objectives

- To obtain all the data from USD/COP exchange rate historical values.
- To implement and compare the Artificial Neural Networks (ANNs), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) methods against each other.

- Select the most accurate model based on metrics such as R^2 and Mean Absolute Error (MAE).

1.4 Justification

Deep learning algorithms may prove effective in forecasting exchange rates because of their capacity to deal with complicated patterns, nonlinear relationships, and massive volumes of data. Several key justifications support the use of deep learning in exchange rate forecasting:

Pattern Recognition and Nonlinear Relationships: Deep learning algorithms such as recurrent neural networks (RNNs) and LSTM capture the complex patterns and nonlinear relationships found in the data. The nature of exchange rate behavior is generally highly complex and nonlinear, caused by economic, geopolitical, or market reasons. Deep learning models can capture these nuances more accurately than linear models, leading to more accurate predictions [10].

Feature Representation and Data Abstraction: The deep learning algorithms autonomously learn high-level data abstractions. Such machines can derive important characteristics from raw input data, allowing them to detect tiny links between economic indicators, market sentiment, or exchange rates. The capacity to be abstract and represent information makes forecasting more accurate [11].

Temporal Dynamics and Time Series Analysis: Time series analysis is fundamental in forecasting exchange rates, and deep learning models are appropriate. Due to their nature, RNNs and LSTM networks “remember” past information, allowing a system to comprehend temporal relationships in sequential data. This is especially beneficial in getting the material aspects of the exchange rate, involving the movement patterns and trends [12].

Handling Big Data and Information Complexity: Deep learning can process large amounts of financial and economic data. They can work with macroeconomic indicators and use news sentiment analysis and historical market data to integrate these inputs into a more holistic forecasting process [13].

Adaptability and Model Flexibility: The deep learning models have adaptive capacity and can adjust to evolving data sets. Such forecasts are not set in stone but can be continually updated and improved as new data surfaces, considering the evolving market environments and other forces that affect exchange rates [14].

Research and Development in Deep Learning: Deep learning research continually evolves, giving rise to new architectures, regularizers, and optimizers tailored for forecasting tasks like exchange rates [13].

Thus, deep learning algorithms are promising for exchange rate forecasting because they can manage complicated data connections, time variation, and substantial information sets. Nevertheless, the issues of interpretability, data scarcity, and model overfitting should be addressed for deep learning to significantly impact more accurate and reliable exchange rate forecasts.

Chapter 2

Theoretical Framework

This section explains the principles required to comprehend this work. It begins with a time series definition, followed by characteristics, classification, and forecasting applications. Second, specific artificial neural network models are examined and thoroughly explained.

2.1 Time Series

Time series prediction is a fundamental approach in data analysis that uses historical data arranged by time to produce forecasts or projections. A time series, in other words, is a succession of observations or data gathered at regular or irregular time intervals to predict future values based on patterns and trends detected in prior data. A time series' main components are trend, seasonality, cyclical, and random. This approach is used in various sectors, including finance, e.g., forecasting stock prices [15]; economics, e.g., GDP (gross domestic product) [16]; meteorology, e.g., predicting daily temperatures [17]; sales, e.g., predicting monthly sales of a retail store[18]; and many more, where future values must be expected [19]. In this situation, it is possible to construct an accurate forecast by accounting for several factors. However, because of how this study is defined, it exclusively concentrates on univariate time series. As a result, a univariate time series refers to a sequence of data points collected or observed sequentially over time involving a single variable or series of observations. In simpler terms, it represents a time-ordered data sequence consisting of only one type of measurement or observation at each time point [20], as shown in Eq. 2.1.

$$Y_t = f(Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}, \varepsilon_t) \tag{2.1}$$

Where:

- Y_t represents the value of the variable at a time
- $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}$ denote lagged values of the variable at time points before t, up to a certain lag order p. These lagged values are often called autoregressive terms in an autoregressive model (AR)
- f represents the functional relationship or model that describes how the variable at time t depends on its past values and potentially an error term (ε_t) that captures unexplained variability or randomness
- ε_t is the random error term at time t representing the difference between the observed value and the predicted value by the model

2.1.1 Time Series Components

The traditional approach to analyzing time series is predicated on the idea that the observation variable's values result from three factors whose combined action produces the measured values. These components are showing in Figure 2.1.

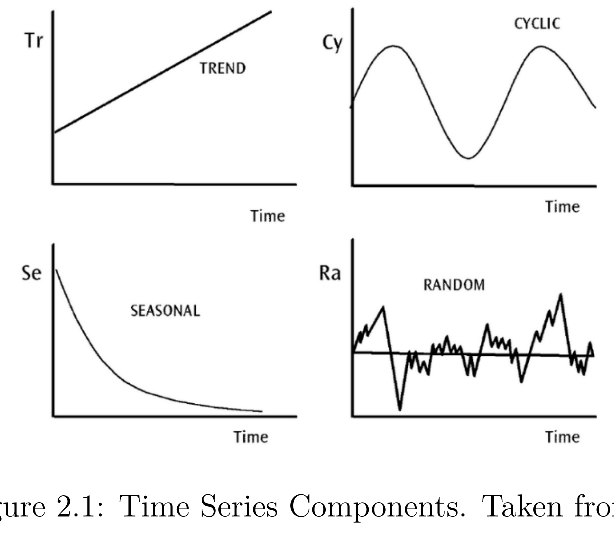


Figure 2.1: Time Series Components. Taken from [1]

Trend

A trend is the inclination of a time series to increase, diminish, or stagnate over a long period. In other words, a trend is a time series' long-term orientation. For example, time series data relating to population growth, the number of dwellings in a country, and so on show an increasing tendency, whereas series relating to death rates, the quantity of available natural resources, and so on show a falling trend [21].

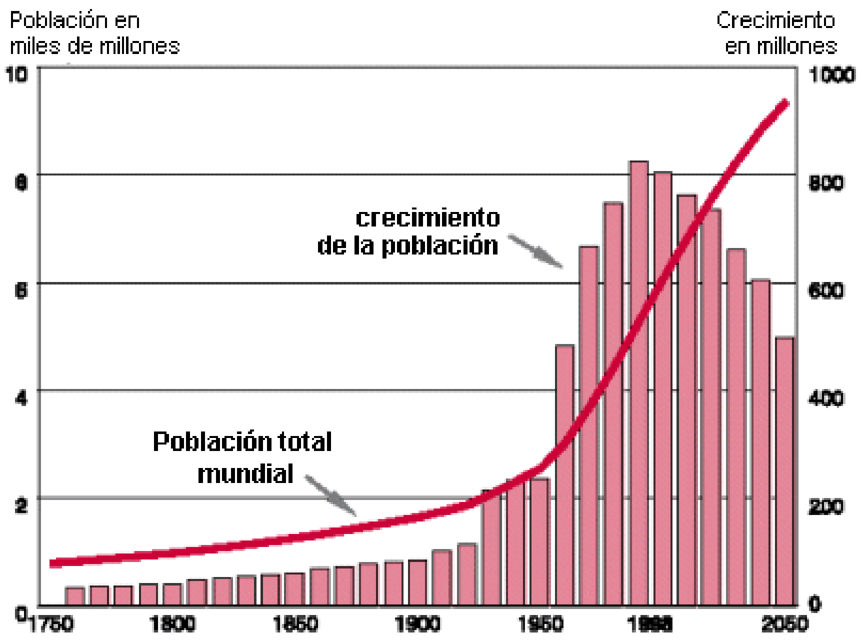


Figure 2.2: Trend example in population. Taken from [2]

Seasonal

Many time series have a periodicity or variation over a period (half-yearly, monthly, etc.). Weather and temperature conditions, customs, and other pertinent elements contribute to seasonal fluctuations [22]. For example, Retail sales in Ecuador surged throughout November and December due to the Christmas holidays. These impacts are simple to grasp and can be explicitly measured or even deleted from the data series, known as deseasonalizing the series [23].

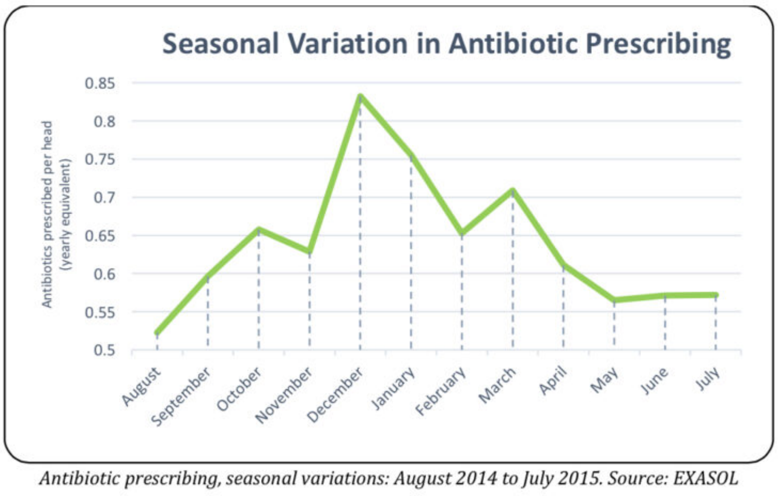


Figure 2.3: Seasonal variation in Antibiotic Prescribing. Taken from [3]

Cyclical

Cycles in time series analysis refer to any repeating pattern in the mean level of a series whose duration is not fixed or known and generally occurs over two or more years. The magnitude of cyclical effects is generally more variable than seasonal effects. Cycles may represent patterns of interest, but they are more challenging to identify and require more extended series to be adequately captured [24].

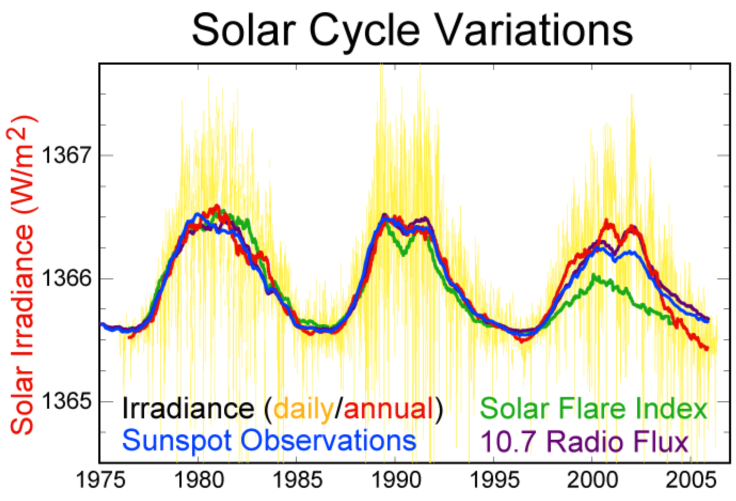


Figure 2.4: Ciclical variation in Solar Variations. Taken from [4]

Random

In the case of time series, another type of movement may be seen. It's just erratic and random movement. No hypothesis or trend, as the name implies, can be used to indicate irregular or random movements in a time series. These consequences are unpredictable, chaotic, and uncontrolled. Unexpected time series components include earthquakes, conflict, starvation, and floods [25].

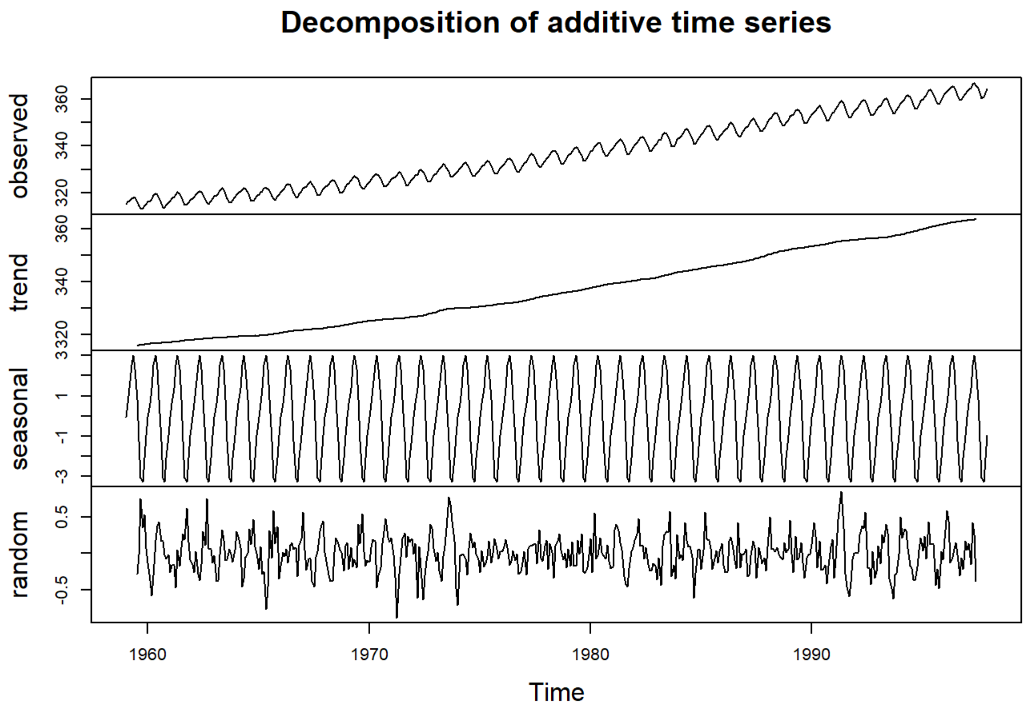


Figure 2.5: Random Variation Component. Taken from [5]

Of these components, the trend and seasonal are deterministic, while cyclical and random are random. For this reason, to account for the effects of these four components, two types of models are commonly used for time series: multiplicative and additive.

- **Multiplicative model.-** This model assumes that the seasonal pattern also increases as the data increases. Most series charts show this pattern. The trend and station components are multiplied in this model and added to the error component.

$$Y(t) = T(t) \times S(t) \times C(t) \times I(t)$$

- **Additive model.-** A data model in which the effects of individual factors are differentiated and aggregated together to model the data.

$$Y(t) = T(t) + S(t) + C(t) + I(t)$$

In both equations, Y(t) is the observation, and T(t), S(t), C(t), and I(t) are, respectively, the trend, seasonal, cyclical, and irregular or random variation at time [26].

2.2 Time Series Forecasting

Time series data may be used to estimate the exchange rate using a variety of methodologies and approaches. The examination of historical patterns is one of the most used methodologies. This strategy includes evaluating patterns and trends in previous exchange rate data to forecast future movements. Moving averages and exponential smoothing are two statistical procedures that may be used to find patterns in data [27]. In addition to historical trend analysis, seasonal and cyclical component analysis is critical in projecting exchange rate time series. Seasonal trends and economic cycles can impact the movement of exchange rates over time. Identifying and modeling these components can enhance prediction accuracy. The use of complex mathematical and statistical models, such as ARIMA models (AutoRegressive Integrated Moving Average) or GARCH models, is another critical strategy in predicting temporary series of exchange rates. (Autoregressive Conditional Heteroskedasticity with Generalized Autoregressive Heteroskedasticity). These models consider self-correlation, seasonality, and data volatility, which can help forecast future exchange rate movements. In addition to conventional methodologies, the use of modern technologies such as machine learning and artificial intelligence in predicting the pace of change has gained favor in recent years. Deep learning models, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), can scan vast datasets and find complicated patterns that other techniques may miss [28]. However, due to the volatile and frequently unpredictable character of financial markets, it is necessary to remember that projecting transitory series on the exchange rate is difficult. Macroeconomic, political, and social variables can unexpectedly influence exchange rates, making projections unpredictable. As a result, it is critical to employ numerous methodologies and information sources to make more accurate forecasts. In brief, forecasting exchange rate time series is a critical problem in the financial sector. It is based on the examination of historical data, the identification of trends, the consideration of seasonal and cyclical components, and the use of advanced mathematical and technical models. Although precise precision may be impossible due to market volatility, these strategies help make educated judgments in international commerce and financial risk management.

2.3 Artificial Intelligence Methods

Artificial intelligence (AI) is a science and engineering domain concerned with developing systems that exhibit intelligent behavior, such as perception, natural language processing, problem-solving, learning, and adaptation. It highlights the interdisciplinary nature of AI, which intersects with many domains, including mathematics, linguistics, psychology, neuroscience, mechanical engineering, statistics, economics, control theory, cybernetics, and philosophy. The introduction also discusses AI's scientific and engineering goals, such as developing intelligent agents, formalizing knowledge, and mechanizing reasoning in all areas of human endeavor [29].

2.3.1 Evolution of AI Methods

Developing AI techniques was very gradual; it had significant changes and discoveries. Initially, classical AI approaches aimed at mimicking human thought and problem-solving using rule-based systems and symbolic reasoning. However, the area underwent a change owing to the evolution of machine learning, where the system could learn and have proficiency over time using data.

Machine learning strategies such as supervised, unsupervised, and reinforcement learning are used in AI today. Such methods enable systems to see models, collect information, and anticipate through learning from big data, allowing system decision-making capacity.

Advances in a deeper level of artificial intelligence using deep learning, a subset of machine learning with multiple layered neuronal networks, have triggered some changes. Deep learning approaches have demonstrated outstanding performance in image recognition, natural language processing, and solving challenging tasks in various disciplines [30].

2.3.2 Artificial Neural Networks

The method of artificial neural networks (ANNs) has been proposed as an alternative strategy to time series forecasting and has garnered enormous popularity in recent years. The main goal of ANNs was to build a model that could mirror the intelligence of the human brain in a machine. ANNs, like human brains, attempt to discover regularities and patterns in input data, learn from experience, and then deliver generalized results based on their previously existing knowledge. Although ANNs were developed primarily for biological reasons, they have since been used in various applications, notably forecasting and classification [31]. The following sections will highlight the key characteristics of ANNs that make them popular for time series analysis and forecasting [32].

Neuron Layers

ANNs comprise multiple layers, most frequently three neuron layers. In Figure 2.6 we can see a simple Network with three layers. The first is the input layer, the second is the hidden layer, and the third is the output layer. Each layer is made up of a particular number of nodes that are linked to one another. The input layer receives the data to distribute to the hidden layer, which controls all computations and approximations. A single hidden layer is usually sufficient to conduct simple computations. On the other hand, the number of hidden layers is determined by the issues to be addressed. Finally, the network's output layer generates the final result [19].

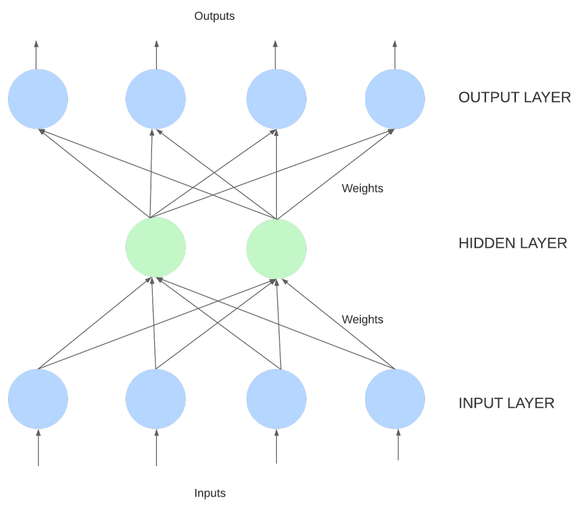


Figure 2.6: ANN architecture

Connections and weights

The connections between nodes, also called synapses, represent value and Weight. Each Weight plays a part in decision-making since their values are not fixed; in other words, they alter based on fresh data. In this way, ANN learns by varying the Weight of each link, resulting in a dynamic and intelligent model [19].

Activation function

Neurons in the hidden and output layers often use activation functions to introduce non-linearity into the network, enabling it to learn complex relationships in the data. Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, Tanh, etc.

However, the activation function should be chosen based on the demands of the situation [33].

The connections, weights, and activation functions work together, as seen in Figure 2.7.

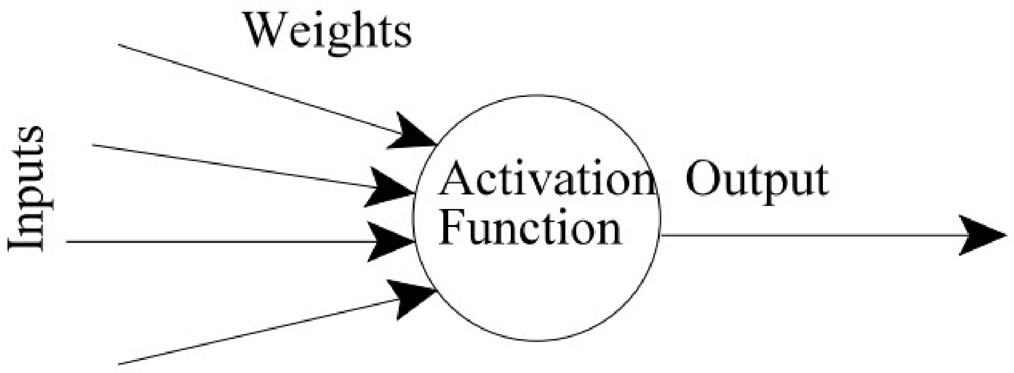


Figure 2.7: Weight of each element and input and output of the ANN system. Taken from [6]

Learning algorithm

The learning rate is a hyperparameter that reflects the steps required to produce a weight change during the training phase. That is, it determines how quickly the network will be learned. Hyperparameters often have values ranging from 0 to 1. The algorithm is often unstable if the learning rate is very high or near one, meaning the weights change very fast during training. If it is deficient, the algorithm will take a long time to converge, and learning will be sluggish [19].

Loss function:

A loss function is a mathematical function used in machine learning and optimization algorithms to quantify the difference between predicted and actual values. The primary goal of the loss function is to minimize its value during the training process. Various loss functions depend on the problem's nature, such as regression, classification, or generative tasks [34].

In a supervised learning context, where the model learns from labeled data, standard loss functions include Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss (also known as Log Loss) for classification tasks.

1. **Mean Absolute Error** MAE is a straightforward computation. It entails adding the errors' magnitudes (absolute values) to produce the "total error" and dividing it by n. The Eq. 2.2 describes this metric [35].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \tag{2.2}$$

Where:

n = number of data records

Y_i = true values

$\hat{Y}_i$ = predicted values

2. **Mean Squared Error (MSE):** The Mean Squared Error measures the average of the squared differences between predicted and actual values. It is commonly used in regression problems [36]. Defined by Eq. 2.3.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.3}$$

Where:

n = number of data records

Y_i = true values

$\hat{Y}_i$ = predicted values

3. **R-squared (R^2)** The coefficient of determination, often known as R square, shows the amount of the dependent variable's variation that the linear regression model can explain. It is a scale-free score, which means that regardless of how tiny or massive the numbers are, the R-squared value is less than one. The Eq. 2.4 describes this metric.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} \tag{2.4}$$

Where:

n = number of data records

Y_i = true values

$\hat{Y}_i$ = predicted value

$\bar{Y}$ = mean value of true variables.

If the value of R^2 is close to 1, we can deduce the accuracy of the model is the best, while if it tends to $-\infty$, we can say that the accuracy of the model is not good [37].

4. **Cross-Entropy Loss (Log Loss):** The cross-Entropy loss is used in classification tasks, especially in scenarios with multiple classes. It calculates the distance between the predicted probability distribution and the actual one-hot encoded labels [38]. Calculate Cross-Entropy Loss with the following Eq. 2.5.

$$\text{Cross-Entropy Loss} = -\frac{1}{n}\Sigma_{i=1}^{n}[Y_i Log(\hat{Y}_i) + (1 - Y_i) Log(1 - \hat{Y}_i)] \tag{2.5}$$

Where: n is the number of data points

Y_i is the truth value takin a value 0 or 1

$\hat{Y}_i$ is the Softmax probability for the i^{th} data point

2.3.3 Deep Learning

Deep learning is a subset of machine learning that utilizes artificial neural networks to model and solve complex problems. These neural networks are meant to imitate the structure and function of the human brain, allowing them to learn from massive volumes of data and make accurate predictions or judgments on previously unknown data [39]. Deep learning has grown in popularity and success in various domains, including computer vision, e.g., Object Detection [40], natural language processing, e.g., Conversational Agents and Chatbots [41], speech recognition, e.g., Voice assistants like Siri [42], and others.

Multilayer Perceptron (MPL)

An MPL is an artificial neural network, a fundamental concept in machine learning and deep learning. It is also known as a feedforward neural network because data flows through it in one direction, from the input to the output layer, without any cycles or feedback loops. The term "multilayer" refers to the fact that it consists of multiple layers of interconnected neurons (nodes) [43].

To provide a mathematical representation of an MPL, we'll use the following notation:

- Let X be the input data with n features, represented as a column vector:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n \times 1}$$

- Let $W^{(l)}$ represent the weight matrix of layer l, where l can be 1 for the first hidden layer, 2 for the second hidden layer, and so on. The dimensions of $W^{(l)}$ ar $m_l \times m_{l-1}$, where m_l is the number of neurons in layer l and m_{l-1} is the number of neurons in the previous layer.
- Let $b^{(l)}$ represent the bias vector of layer l. The dimensions of $b^{(l)}$ are $m_l \times 1$.
- Let $A^{(l)}$ represent the output (activation) of layer l, which is also the input to the next layer. The dimensions of $A^{(l)}$ are $m_l \times 1$.

- Let $Z^{(l)}$ represent the weighted sum of inputs to layer l before applying the activation function. The dimensions of $Z^{(l)}$ are $m_l \times 1$.
- Let $f^{(l)}(\cdot)$ be the activation function used in layer l.
- Let Y represent the output of the Multilayer Perceptron (MLP), which is the final prediction. The dimensions of Y depend on the specific task (e.g., binary classification, multi-class classification, regression).

With this notation, the forward propagation in the MLP can be expressed as follows:

1. Input Layer (Layer 0):
   $A^{(0)} = X$
2. Hidden Layers (Layer l, $l \geq 1$):
   $Z^{(l)} = W^{(l)}A^{(l-1)} + b^{(l)}$
   $A^{(l)} = f^{(l)}(Z^{(l)})$
3. Output layer (Layer L):
   $Z^{(L)} = W^{(L)}A^{(L-1)} + b^{(L)}$
   $Y = A^{(L)} = f^{(L)}(Z^{(L)})$

During training, the MLP is optimized using a loss function $L(Y, Y_{true})$ that measures the discrepancy between the predicted output Y and the actual target values Y_{true}. The most common loss functions include mean squared error (MSE) for regression and cross-entropy for classification tasks.

The backpropagation algorithm is then used to compute the gradients of the loss concerning the model parameters (weights and biases) and update them accordingly to minimize the loss function. This process is iterated over multiple epochs until the model converges to the desired performance [44].

2.3.4 Long Short Term Memory

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997. It was designed to overcome the vanishing gradient problem, which occurs when training traditional RNNs on long sequences, leading to difficulties in capturing long-range dependencies. LSTM has gained popularity due to its ability to handle long-term dependencies and maintain information over extended intervals. The main issue with standard RNNs is that they struggle to capture long-term dependencies in sequential data due to the vanishing or exploding gradient problem. When training RNNs on long sequences, gradients can become exponentially small, making it difficult for the network to learn from distant events in the past [45].

LSTM Architecture

LSTM overcomes the problem mentioned above by introducing a more sophisticated memory cell [45], which has three main components as seen in Figure 2.8:

1. **Cell State (c_t):** The cell state serves as the memory of the LSTM and allows information to flow through the network over time. It is designed to carry relevant information while filtering out unnecessary information. This ensures that the LSTM can remember relevant information for long periods.
2. **Input Gate (i_t):** The input gate determines how much fresh information is stored in the cell state. It determines what new data from the current input and the prior concealed state to let into the cell state.
3. **Forget Gate (f_t):** The forget Gate determines what information should be discarded from the cell state. It controls the flow of information from the previous cell state to the current cell state.
4. **Output Gate (o_t):** The output gate decides how much of the cell state should be exposed to the output. It selectively filters and outputs the relevant information to the next hidden state and the final output of the LSTM.

Main Idea

The cell state is the most essential component of LSTMs. A horizontal line runs across the top of the diagram, as seen in Figure 2.9. Furthermore, it moves quickly across the entire

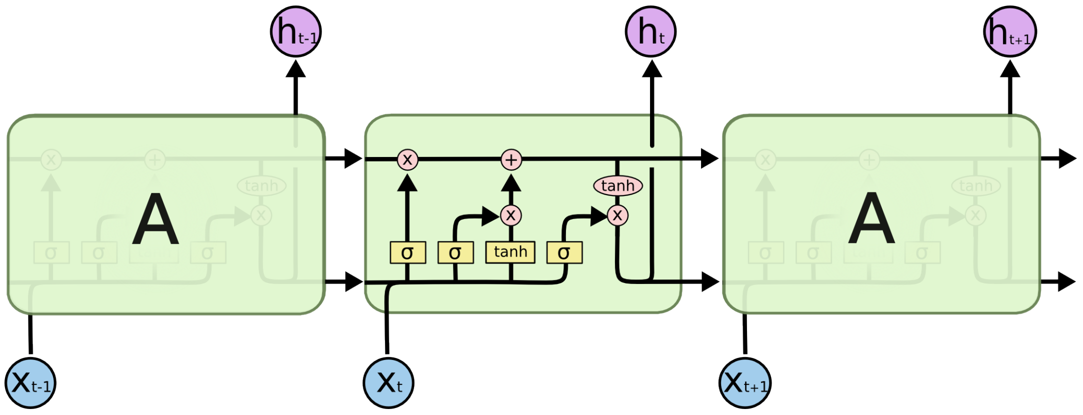


Figure 2.8: LSTM Architecture. Taken from [7]

chain with some interactions, yet it allows information to travel without alteration.

Using components known as gates, the LSTM carefully regulates the ability to delete or add information to the cell state. These gates comprise a sigmoid neural network layer and a pointwise multiplication operation.

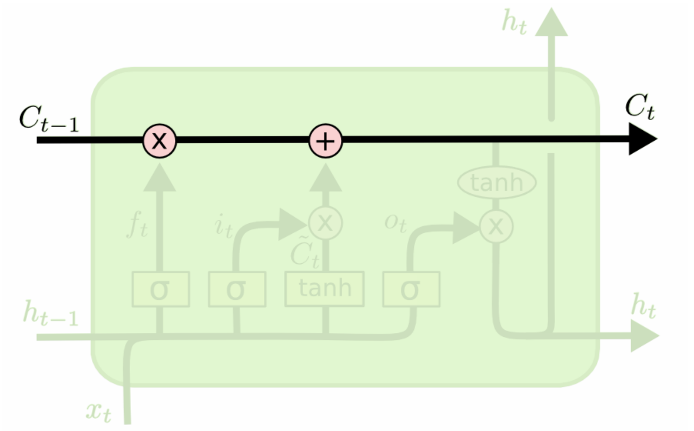


Figure 2.9: LSTM - Cell State. Taken from [7]

The output of the sigmoid layer is a range of integers between 0 and 1, indicating how much of each component should be allowed through. If the number is one, it allows everything through, while if the number is zero, nothing through.

The mathematical interpretation of this is in Eq. 2.6.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.6}$$

Where:

- $[h_{t-1}, x_t]$ concatenates the previous hidden state and the current input.

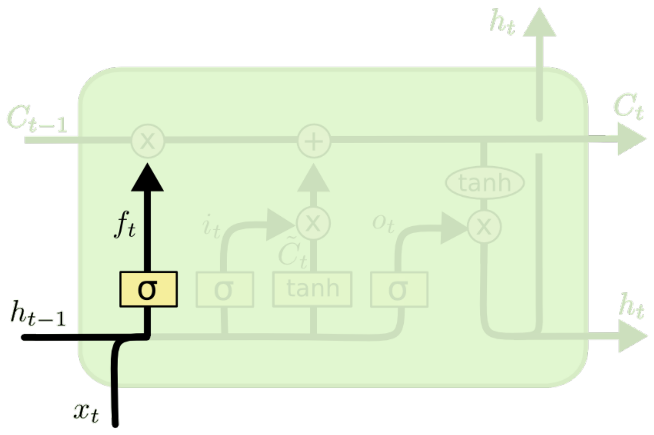


Figure 2.10: LSTM - Forget Gate. Taken from [7]

- W_f and b_f are learnable weight matrices and biases.

To determine what additional information we will store in the cell's state. First, we use a sigmoid layer called the "input gate layer," which decides what values we'll update. Also, we use a hyperbolic tangent layer to create a vector of new candidate values $\tilde{C}_t$ to add to the state, as we can see in Figure 2.11.

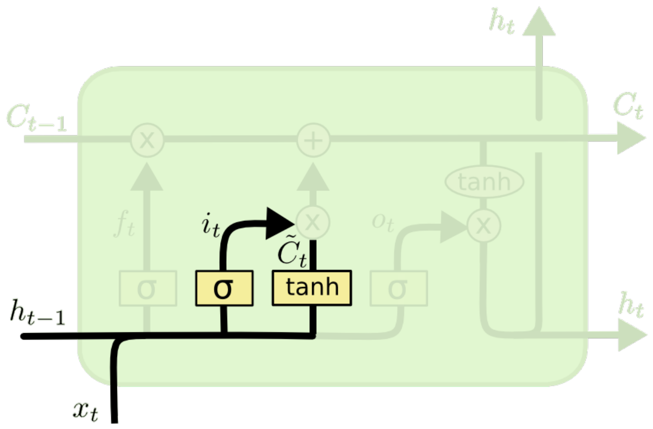


Figure 2.11: LSTM - Input Gate. Taken from [7]

The mathematical interpretation about this is in Eq. 2.7 and Eq. 2.8.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.7}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.8}$$

Where:

- $[h_{t-1}, x_t]$ concatenates the previous hidden state and the current input.
- W_i, W_C and b_i, b_C are learnable weight matrices and biases.

The next step is to update the old to the new cell state. We do it by multiplying the old state by f and adding the candidate values, scaled by how much we decided to update each state value. We can see the graphical and mathematical interpretations in Figure 2.12 and Eq. 2.9, respectively.

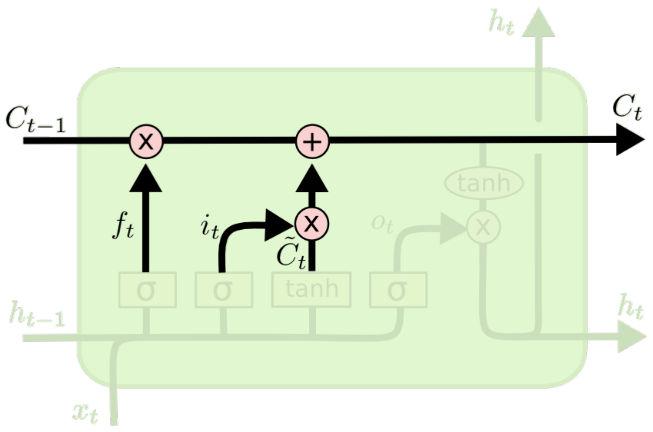


Figure 2.12: LSTM - Update cell state. Taken from [7]

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{2.9}$$

Finally, the output will be based on our cell state but in a filtered version. To do this, we run a sigmoid layer, which decides what outputs to generate. Simultaneously, we pass the cell state through the hyperbolic tangent to push the values between -1 and 1. After that, we multiply both parts to obtain the desired outputs.

We can see the graphical and mathematical interpretations in Figure 2.12, Eq. **??** and 2.11, respectively

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.10}$$

$$h_t = o_t \times \tanh(C_t) \tag{2.11}$$

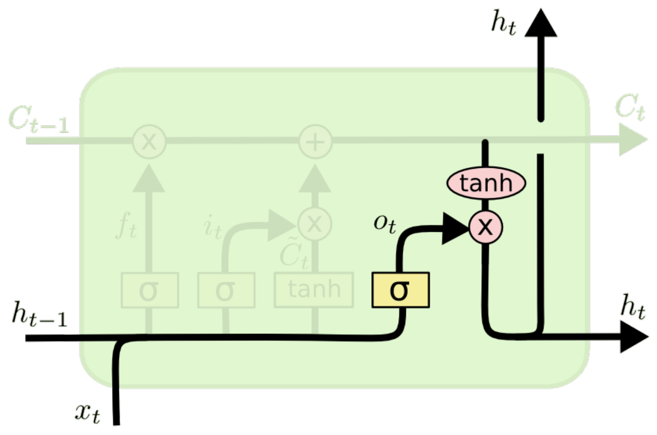


Figure 2.13: LSTM - output state. Taken from [7]

2.3.5 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) that was introduced by Cho et al. in 2014 [46]. It was designed to address issues with traditional RNNs, such as the vanishing gradient problem, making it difficult to train on long sequences.

GRU is a variation of the Long Short-Term Memory (LSTM) network, which also addresses the vanishing gradient problem but has a more complex architecture. GRU simplifies the LSTM architecture by combining the forget and input gates into a single ”update gate” and merging the cell and hidden states. This leads to a more compact and computationally efficient model while preserving the ability to capture long-term dependencies in sequences.

GRU architecture

The GRU architecture show in Figure 2.14.

1. **Update Gate (z_t):** It determines how much of the previous hidden state should be passed on to the current time step. It takes the current input (x_t) and the previous hidden state (h_{t-1}) as input and outputs a value between 0 and 1 for each element in the hidden state.

2. **Reset Gate (r_t):** It determines how much of the previous hidden state should be forgotten when computing the candidate hidden state. Similar to the update gate,

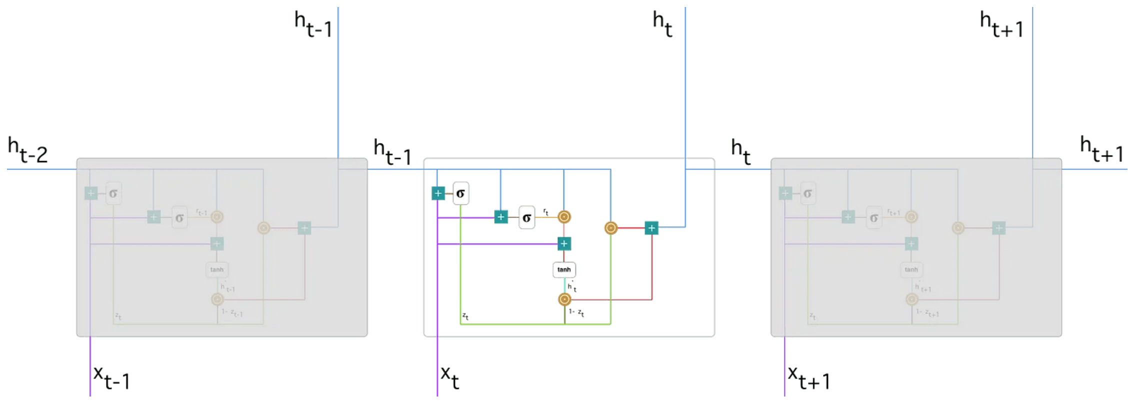


Figure 2.14: GRU architecture. Taken from [8]

it takes the current input (x_t) and the previous hidden state (h_{t-1}) as input and outputs a value between 0 and 1 for each element in the remote state. The reset gate is computed as follows:

3. **Candidate Hidden State ($\tilde{h}_t$):** It is the new candidate hidden state that will be computed based on the current input and the previous hidden state. It is obtained by applying the tanh activation function to the linear combination of the current input and the reset gate:

4. **Hidden State (h_t):** The final hidden state at time step t is a combination of the previous hidden state (h_{t-1}) and the candidate hidden state ($\tilde{h}_t$), controlled by the update gate (z_t):

Main Idea

Here's how GRU works based on [47].
We start with calculating the update gate z_t for time step t using the Eq. 2.12.

$$z_t = \sigma(W_z \cdot x_t + U_z \cdot h_{t-1}) \tag{2.12}$$

When x_t is plugged into the network unit, it is multiplied by its Weight W_z. The same goes for h_{t-1}, which holds the information for the previous $t-1$ units and is multiplied by its Weight U_z. Both results are added together, and a sigmoid activation function is applied to squash the result between 0 and 1. Following the above Figure 2.15, we have:

The update gate assists the model in determining how much past knowledge (from

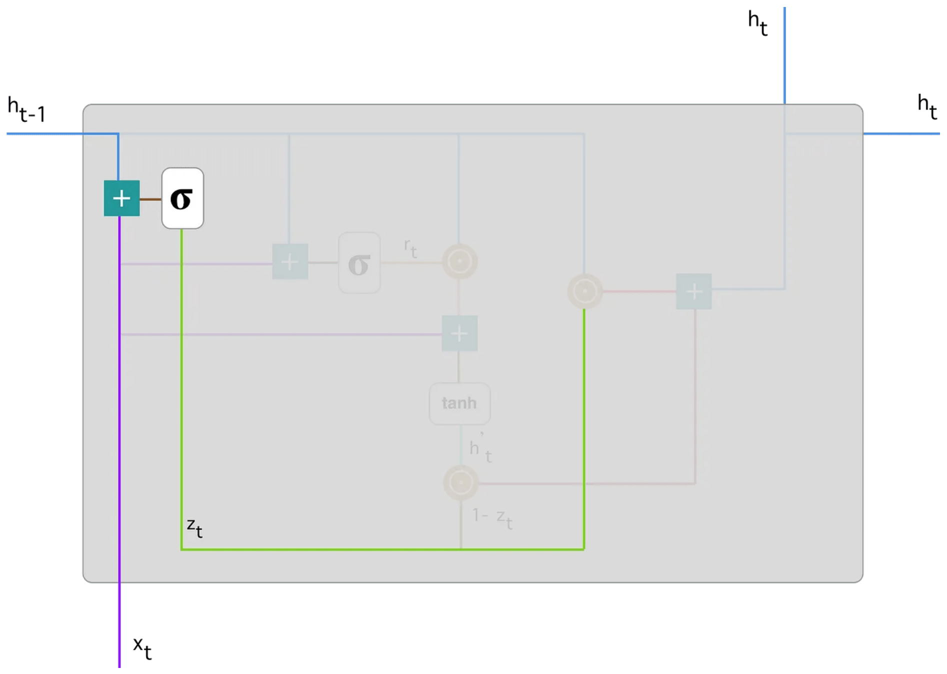


Figure 2.15: GRU - Update Gate. Taken from [8]

earlier time steps) should be passed on to the future. This is extremely powerful because the model may replicate all of the knowledge from the past, eliminating the risk of disappearing gradients. We'll see how the update gate is used later. For the time being, recall the z_t formula.

Second, we determine how much of the past we will forget. Eq. 2.13 is used to compute it. This formula is the same as the one for the update gate. The difference comes in the weights and the Gate's usage, which will be seen briefly. Figure 2.16 shows where the reset gate is.

$$r_t = \sigma(W_r \cdot x_t + U_r \cdot h_{t-1}) \tag{2.13}$$

The preceding operations are discussed here. We plug in h_{t-1} (blue line) and x_t (purple line), multiply them with their appropriate weights, add the results, and apply the sigmoid function as before.

The next step is to use Eq. 2.14 to update the candidate's hidden state. Figure 2.17 shows this operation.

$$\tilde{h}_t = \tanh(W_h \cdot x_t + r_t \odot (U_h \cdot h_{t-1})) \tag{2.14}$$

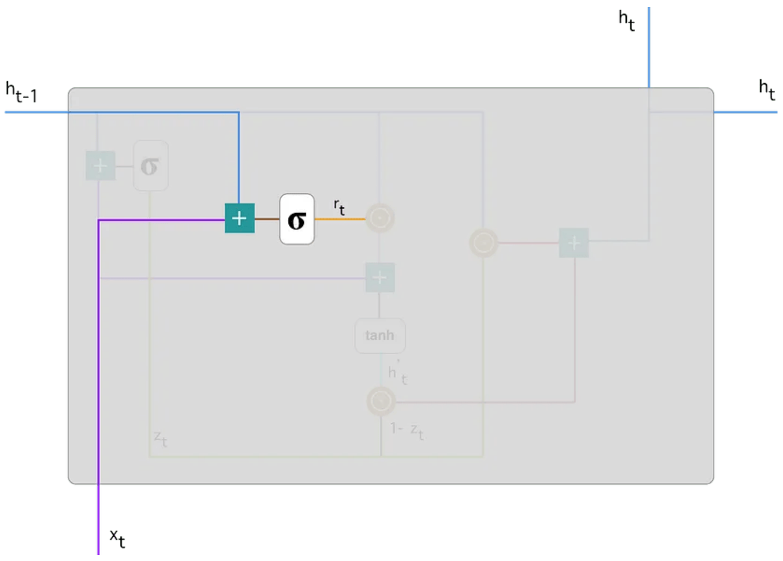


Figure 2.16: GRU - Reset gate. Taken from [8]

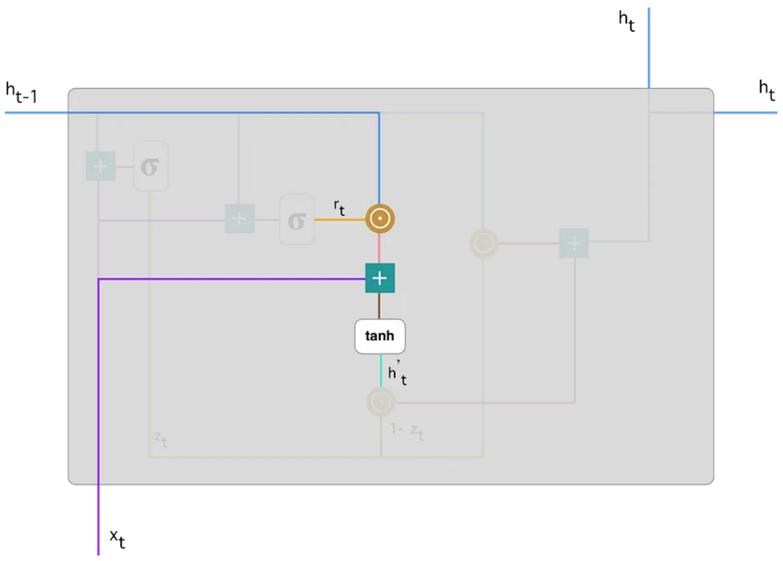


Figure 2.17: GRU - Candidate Hidden State. Taken from [8]

Firstly, multiply the input x_t with a weight W and h_{t-1} with a weight U. Secondly, calculate the Hadamard (element-wise) product between the reset gate r_t and $U \cdot h_{t-1}$. That will determine what to remove from the previous time steps. After that, add the results and apply the non-linear activation function *tanh*.

The last step in the network is to compute the hidden state (h_t) using Eq. 2.15, a vector that carries information for the current unit and transmits it to the network. The update gate is required to do this. It decides what to acquire from candidate hidden state ($\tilde{h}_t$) and what to collect from earlier stages (h_{t-1}).

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.15}$$

To accomplish this. First, multiply the update gates z_t and h_{t-1} using element-wise multiplication. Then, do element-wise multiplication on $(1-z_t)$ and $\tilde{h}_t$ and sum the results.

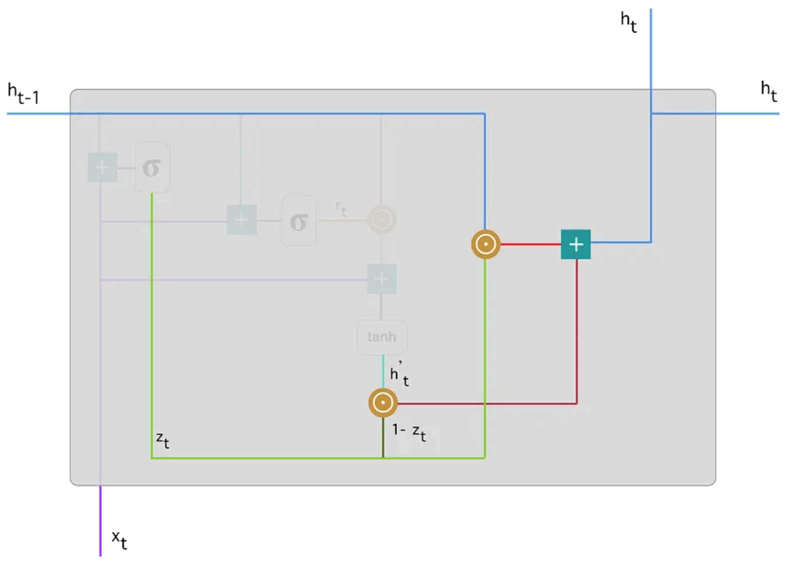


Figure 2.18: GRU - Hidden State. Taken from [8]

In Figure 2.18, we can see the operations described above, where z_t (green line) is used to calculate $1 - z_t$ (dark green line), which, applying Hadamard multiplication with $\tilde{h}_t$ (turquoise line), produces a result represented in the dark red line. We also perform another Hadamard multiplication between z_t and h_{t-1} (blue line), giving us the result represented by the bright red line. Finally, h_t (the blue line) results from the sum of the outputs corresponding to the bright red and dark red lines.

Chapter 3

State of the Art

This chapter examined time series forecasting approaches for finance, exchange rate, and price prediction. It provided a quick description of five research publications published between 2009 and 2021. These methods include recurrent networks, LSTM, GRU, and others. Finally, a brief discussion and table summary were held.

3.1 Tlegenova, D. (2015)

In this work, Daniya Tlegenova [48] uses the ARIMA model as a time series analysis model to simulate the annual exchange rate between USD/KZT, EUR/KZT, and SGD/KZT from 2006 to 2014. The exchange rate data used in this paper is from the National Bank of the Republic of Kazakhstan. For building an ARIMA model for exchange rate forecasting, she follows the three essential phases:

1. **Test pattern detection:** This step involves analyzing the time series data to identify existing patterns or tendencies. The paper uses graphical analysis to identify patterns such as seasonality, trend, and cyclical behavior. To sum up, this step is essential in constructing the ARIMA model as it helps to determine the appropriate order of differencing, autoregressive, and moving average terms.

2. **Model parameter estimates:** The paper explains that this step involves estimating the parameters of the ARIMA model using the maximum likelihood method. It is a crucial step in constructing the ARIMA model, as it helps to determine the appropriate values of autoregressive and moving average terms.

3. **Adequacy assessment:** The paper explains that this step involves comparing the forecasted value to determine the model's accuracy. Additionally, this paper uses three measures of forecasting accuracy, namely mean absolute error (MAE), mean fundamental percentage error (MAPE), and root mean square error (RMSE). Finally, this step is also essential, as it helps to determine the accuracy of the model and its usefulness in forecasting future values.

4. **Model prediction:** The paper presents the actual and forecasted values for all three currencies in a table and a graph. Moreover, this paper finds that the developed ARIMA model can predict exchange rate values more accurately than the previous forecast for the other two currencies.

The results of this paper show that the ARIMA model is an effective tool for forecasting exchange rates because there are notable trends and patterns in the exchange rate data for USD/KZT, EUR/KZT, and SGD/KZT over the period from 2006 to 2014. For example, Figure 2 from the PDF file shows a tendency to increase the USD/KZT exchange rate over eight years from 2006 to 2014, with a total increase of more than 44 percent. Also, Figure 3 illustrates that the EUR/KZT exchange rates rose significantly from 2008 to 2009 and from 2012 to 2013, with significant changes of more than 16 percent during the first increase and more than 5 percent during the second. Finally, the comparison of SGD/KZT exchange rates is shown in Figure 4, which shows some fluctuations and trends from 2006 to 2014. The paper concludes that the ARIMA model can identify patterns and trends in historical exchange rate data and be an effective tool for forecasting exchange rates.

3.2 Solís, E. 2021

In this work, [49] presented a deep learning model that combines a CNN layer with two LSTM layers and three regular densely connected NN layers for intraday stock price forecasting. The model was used to input the opening stock price for the last sixty days; its dataset was batch-built from a non-stationary time series and was based on a sliding window approach. Moreover, this model performs one-step and multi-step ahead forecasting with a low error rate. The testing findings demonstrate that the suggested design delivers the most outstanding performance in a short amount of time. The best result was obtained by training the model with a two-minute interval time series; in this case, the model's error rate is 6.7. The model produces an error rate of 9.94 when trained on a five-minute interval time series, which is greater than the model trained on two-minute intervals but also delivers good performance over a more extended period.

3.3 Vega, O. 2021

Vega, O. (2021) [21] compares sequential (MPL, LSTM, CNN) and non-sequential models (MLP&LSTM, LSTM&CNN, CNN&MPL) to forecast monthly banana exports. To reach these models, one neuron was used for the input layer, three hidden layers with seven neurons each, and one for the output layer. As optimizers, the activation functions employed were ReLU and ADAM. Furthermore, 700 seasons were used, with a micro-batch size of 32. The average square error (RMSE) was used to assess the model's performance. Two sets of data were utilized, each having 248 values reflecting the number of monthly banana exports, measured in metric tons and thousands of USD FOB, and acquired from the Central Bank of Ecuador's website between January 2000 and August 2020. The data is divided into 75% for training, 5% for validation, and 20% for testing. It should be noted that the sliding window approach was employed with input sizes of 6, 12, and 24 months and output sizes of 1 month. The results showed that increasing the sliding window size improved DNN performance by allowing better results in metrics like RMSE and Loss. The proposed non-sequential DNNs outperformed sequential ones due to their combined inference capacity. The computational complexity of these non-sequential DNNs was lesser than sequential ones due to their small number of trainable parameters. The best model, MLP&CNN, had 319 trainable parameters, outperforming CNN and LSTM networks.

3.4 Noboa, S. 2021

Noboa, S. (2021) [19] analyzes five ANN models (MLP, RNN, bi-directional LSTM, Conv-LSTM, and Conv-LSTM-MLPs) to forecast avocado prices in Ecuadorian marketplaces, notably the wholesale market of Ibarra. Two datasets were used: one with 472 avocado prices per week and another with 118 monthly. From 6/6/2011 to 8/2/2021, these statistics were acquired from Ecuador's Public Agricultural Information System (SIPA) website. 70% of the data was set aside for training and 30% for testing. Three tests were carried out to compare these models. The following hyperparameters were utilized for all models in the first experiment: SGDm as optimizer and 0.9 as momentum; MAE as loss function; 200 epochs for training; a batch size of 32; and a look-back of 4. On the other hand, the MLP model employed an input layer with ten neurons, two hidden layers with 20 and 10 neurons, respectively, and an output layer with one neuron. Furthermore, he employed ReLU as an activation function and a learning rate of 10^{-7}. The RNN model had an input layer of 40 neurons, two hidden layers of 40 neurons each, and an output layer of one neuron. Furthermore, he employed tanh as an activation function and a learning rate of 10^{-5}. An input layer with 32 neurons, two hidden layers with 32 neurons each, and an output layer with one neuron were employed for the bidirectional LSTM model. Furthermore, he used tanh and Sigmoid as activation functions and a learning rate of 10^{-5}. The Conv-LSTM model had an input layer with 32 neurons, three hidden layers, one convolutional layer, two LSTM layers with 32 neurons each, and a single neuron output layer. Furthermore, he employed a learning rate of 10^{-5} and ReLU, Tanh, and Sigmoid as activation functions. The Conv-LSTM-MPLs model had an input layer with 32 neurons, five hidden layers, one convolutional layer, two LSTM layers with 32 neurons, each two layers with 30 and 20 neurons, respectively, and a single neuron output layer. Furthermore, he employed a learning rate of 10^{-6} and ReLU, Tanh, and Sigmoid as activation functions. The best model, in this instance Conv-LSTM-MLPs, was chosen based on the MAE error for the second experiment, and three hyperparameters were varied: the training-test percentage between 60:40, 70:30, and 80:20; the look-back between 4, 8, and 16; and the optimizer between SGDm and Adam. It is worth noting that the SGDm configuration was already used in experiment one, and the Adam configuration is as follows: $\mu = 10^{-3}$, $\beta_1 = 0,9$, $\beta_2 = 0,999$, $\varepsilon = 10^{-7}$, and $amsgrad = false$. In the third experiment, the model with the best configurations from experiment 2 was utilized on two unseen data sets (avocado and red onion) that will prove the model's performance. Two alternative combinations were randomly picked because they performed well in experiment 2. For the most recent weekly avocado prices, a test MAE of 1.51 was produced. In other words, the model performs well with unknown data. Finally, the MAE for the weekly red onion is 2.30. Despite trimming the red onion data set to make things even more difficult for our model, our model can follow all patterns. Regarding the MAE measure, the results revealed that the algorithms performed better with weekly data than with monthly data. Furthermore, to demonstrate that all of the approaches investigated are still acceptable for time series prediction, even if some outperform others; for example, composite models such as Conv-LSTM and Conv-LSTM-MLP predicted weekly avocado prices the most accurately, even though these models required more time to train and, of course, had a more complicated network architecture. Finally, the significance of changing deep learning model hyperparameters during training was demonstrated.

3.5 Yadav et al., 2020

In this work, [50] compares the LSTM model in two experiments. The first experiment evaluated state-less and stateful LSTM, while the second compared the LSTM with one to seven hidden layer variations. The RMSE was employed as the loss function in both investigations, while ADAM was used as the optimization approach. The root mean square error (RMSE) was used to evaluate the performance of the various models. A batch size of 64 was utilized for training, whereas a batch size of 1 was used for testing. The number of epochs was adjusted to 30 to ensure consistency, and the trials were repeated 30 times. The model used input data from four businesses listed on the Indian stock exchange for 2560 days from 2008-12-29 to 2019-05-24, a span of more than ten years. For the training and test data, an 80:20 ratio was used, resulting in 512 test days. The first experiment's results indicate that a stateless LSTM model is preferable for time series prediction problems due to its higher stability. In contrast, the results of the second experiment suggest that n = 1 appears to be the best configuration in terms of mean RMSE. The one-way ANOVA test also confirmed this because of improved accuracy, quicker training, and a decreased danger of overfitting.

3.6 Ranjit et al., 2018

In this paper, Ranjit et al. (2018) [51] investigate the prediction of Nepalese currency against the American Dollar, Euro, and Pound Sterling using different ANN models. Then, they compare four distinct artificial neural network designs (MLP, SRNN, LSTM, and GRU). To produce this comparison, ten different configurations of each model were tested, with the number of neurons on the hidden layer ranging from 1 to 9. Moreover, the neurons on the input layer are 4, and 1 on the output layer. The hyperparameters utilized for the MPL model were sigmoid as the activation function and backpropagation as the training strategy. In contrast, the hyperparameters used for the SRNN, LSTM, and GRU models were tanh as the activation function and Root Mean Square Propagation (RMSProp) as the training algorithm. Furthermore, MAE was employed as the loss function in all models. The historical data utilized to forecast the USD/NPR, EUR/NPR, and GBP/NPR foreign currencies came from Investing.com, where the high, open, close, and low values were chosen as input since they were connected with the closing price, which is the exit price. Data for each foreign currency was acquired for 25 to 27 days. The results show that the LSTM model can forecast foreign exchange rates and outperform the other models. Various experiments were conducted by varying the number of hidden neurons until the best outcome was obtained. After multiple tests with different network designs, LSTM with structure 4-5-1 produced the most accurate MAE findings.

3.7 Panda and Narasimhan, (2007)

This paper [52] compares the forecasting performance of artificial neural network models with linear autoregressive and random walk models using six evaluation criteria. The authors use weekly exchange rate data for INR/USD from the FX database from January 6, 1994, to July 10, 2003, for 495 observations. Moreover, they used a rolling window approach to estimate the models and evaluate their forecasting performance using mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), mean total error percentage (MAPE), directional accuracy (DA) and Theil's U-statistic. The neural network used in this paper is a single-hidden-layer feedforward with a sigmoid transfer function in the hidden layer and a linear transfer function in the output layer. The weights are initialized to small values based on the technique of Nugyen and Widrow (1990) and mean squared error is taken as the cost function in their study. The rolling window approach used to estimate the models uses a window of 100 observations and then moves the window forward by one observation. This process is repeated until the end of the sample period is reached. The authors use this approach to evaluate models that predict performance over time and avoid over-fitting. In this work, a time series model known as a linear auto-regressive model uses lagged values of the dependent variable as predictors. The authors specifically regress the weekly exchange rate return, which is the dependent variable, on a set of independent factors that also includes the first 20 lagged values of the dependent variable. Subsequently, they identify a limited set of statistically notable variables and use them as explanatory factors to predict the weekly performance of exchange rates. The writers of this article employ a drift-free random walk model, which assumes that there is no trend or drift and that the exchange rate moves in a purely arbitrary walk manner. The best forecast value for tomorrow is derived from the current period value. In summary, the authors find that neural networks outperform linear autoregressive and random walk models in predicting exchange rates. However, linear autoregressive models perform better regarding correlation coefficient and the percentage of correct sign prediction. This paper's findings have important policy implications, as understanding exchange rate determination and forecasting can help authorities determine the best way to influence exchange rates and limit exchange rate volatility.

3.8 Discussion

Time series prediction using deep learning models is a trendy issue recently getting much attention. Several research have been described in this literature overview, and a brief comment is required. Some authors, such as Tlegenova, D. [48], show the procedures involved in building the ARIMA model, such as model identification, parameter estimates, and adequacy assessment, explained in the paper. The significance of model diagnostics in guaranteeing that the model is suitable for the data is also covered in the study. The study's constructed ARIMA model's forecast performance is assessed utilizing non-statistical techniques, including MAPE, RMSE, and MAE. However, in her paper, modeling annual exchange rates is considered one of its limitations because this might not be enough for some applications. The article offers a helpful overview of time series analysis and how it is used to estimate exchange rates. On the other hand, Panda, C., and Narasimhan, V.

[52] use a rolling window approach to estimate the models and evaluate their forecasting performance. They use a rolling window of 100 observations to assess the models, generate forecasts, and evaluate the forecasting performance of the models using various measures of forecast accuracy. The authors also discuss the policy implications of their findings, as understanding exchange rate determination and forecasting can help authorities determine the best way to influence exchange rates and limit exchange rate volatility. Moreover, Yadav et al. [50] and Ranjit et al. [51] employed basic deep learning architectures such as MLP, SRNN, LSTM, and GRU, finding that the LSTM model performs better in prediction for the two data types (Indian stock market and Nepalese currency). Furthermore, because of its concept of long-term dependencies, LSTM is the most extensively used RNN model in the world, making it superior to RNN for data prediction. However, because the settings for the two scenarios differed, we can conclude that selecting the correct hyperparameters, such as optimizers, cost functions, activation features, learning rate, weight reduction, seasons, and batch size, is critical to improving the performance. Similarly, Solis [49], Vega[21], and Noboa[19] employed LSTM in conjunction with various architectures such as CNN, Conv, MPL, and others, yielding superior results by maximizing the benefits of each model. They are making this approach a new trend to increase prediction performance. Furthermore, the quantity of data divided for training and testing was shared by all authors, with the majority using a 70:30 ratio. Finally, the cost functions for performance criteria, like the training and test radius, vary between numerous alternatives, with MAE and RMSE being the most commonly employed.

3.9 Summarize

Table 3.1 outlines all the research articles examined for the related works section in the next section. It should be noted that this is arranged from oldest to newest papers. Furthermore, context refers to which items were forecasted, temporality refers to the time employed, and techniques refer to the algorithms used.

Authors	Context	Temporality	Traditional statistical models	Artificial Neural Networks	Deep Learning	
			Simple	Simple	Simple	Mixture
			ARIMA, Linear Auto regressive, Random walk	SRNN, MLP	RNN, LSTM, GRU	MLP&LSTM, LSTM&CNN, CNN&MPL, bidirec LSTM, Conv LSTM, Conv-LSTM-MPL
Panda & Narismhan [52]	INR/USD exchange rate.	weekly	Yes	Yes	-	-
Tlegenova, D.[48]	Kazakhstan exchange rate	annual	Yes	-	-	-
Ranjit et al. [51]	Nepalese exchange rate	day	-	Yes	Yes	-
Yadav et al.[50]	Indian stock exchange	day	-	-	Yes	-
Solís, E.[49]	Amazon stock prices	intraday	-	-	-	Yes

Table 3.1 continued from previous page

Ref	Context	Tempo-rality	Traditional statistical models	Artificial Neural Networks	Deep Learning	
			Simple	Simple	Simple	Mixture
			ARIMA, Linear Auto regressive, Random walk	SRNN, MLP	RNN, LSTM, GRU	MLP&LSTM, LSTM&CNN, CNN&MPL, bidirec LSTM, Conv LSTM, Conv-LSTM-MPL
Vega, O.[21]	Ecuadorian banana exports price	month	-	Yes	Yes	Yes
Noboa, S.[19]	Ecuadorian avocado prices	weekly and monthly	-	-	-	Yes

Table 3.1: Summary table of all the related works.

Chapter 4

Methodology

4.1 Phases of Problem Solving

Figure 4.1 shows the methodology proposed for this work. It consists of five main stages: data collection, data preparation, model preparation, training models, and models evaluation. Each phase is explained in detail in the following sections.

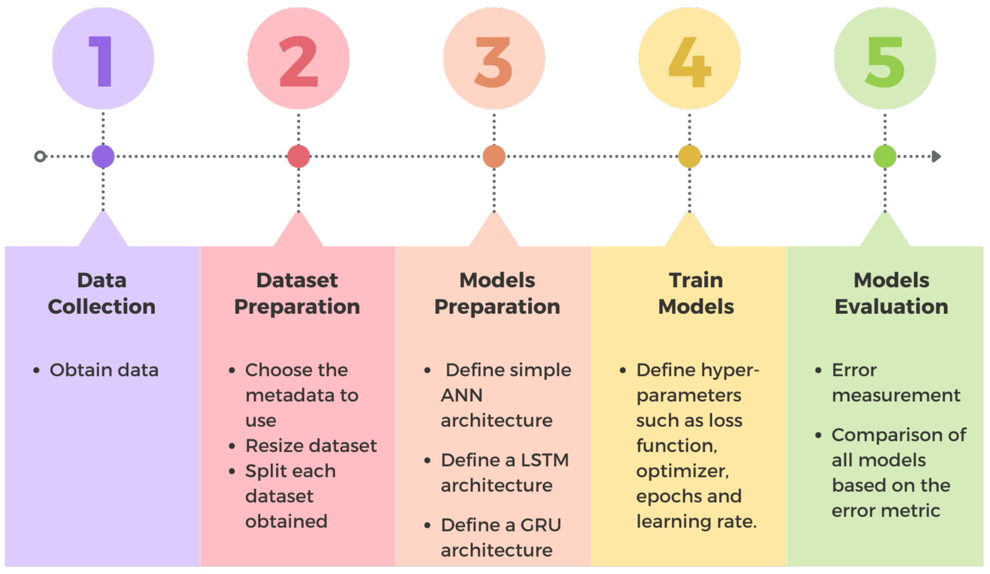
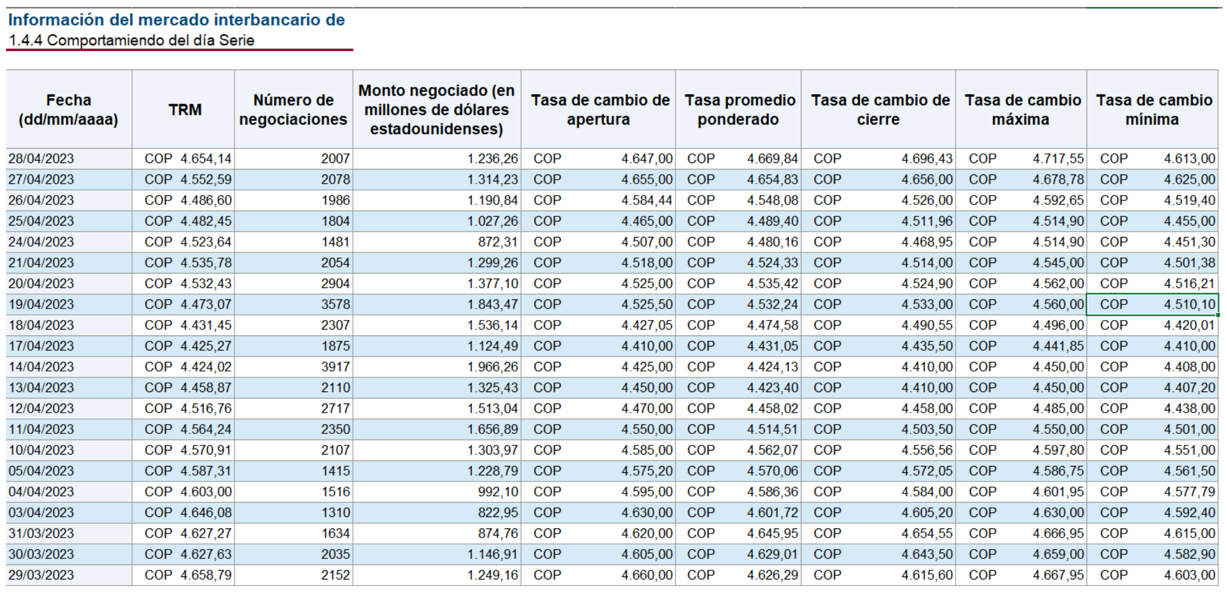

1
2
3
4
5
Data Collection
• Obtain data
Dataset Preparation
• Choose the metadata to use
• Resize dataset
• Split each dataset obtained
Models Preparation
• Define simple ANN architecture
• Define a LSTM architecture
• Define a GRU architecture
Train Models
• Define hyper-parameters such as loss function, optimizer, epochs and learning rate.
Models Evaluation
• Error measurement
• Comparison of all models based on the error metric

Figure 4.1: Flowchart of the proposed methodology

4.1.1 Data Collection

The raw data was acquired manually from the Banco de la Republica de Colombia (BANREP)[1] in .xlsx format. This platform provides COP/USD historical values dating back to November 27, 1991. Our data consists of COP/USD historical values from January 1, 1992, to April 28, 2023. This Data contains 11441 registers from around 31 years ago, as shown in Figure 4.2.


Información del mercado interbancario de
1.4.4 Comportamiendo del día Serie

Fecha (dd/mm/aaaa)	TRM	Número de negociaciones	Monto negociado (en millones de dólares estadounidenses)	Tasa de cambio de apertura	Tasa promedio ponderado	Tasa de cambio de cierre	Tasa de cambio máxima	Tasa de cambio mínima
28/04/2023	COP 4.654,14	2007	1.236,26	COP 4.647,00	COP 4.669,84	COP 4.696,43	COP 4.717,55	COP 4.613,00
27/04/2023	COP 4.552,59	2078	1.314,23	COP 4.655,00	COP 4.654,83	COP 4.656,00	COP 4.678,78	COP 4.625,00
26/04/2023	COP 4.486,60	1986	1.190,84	COP 4.584,44	COP 4.548,08	COP 4.526,00	COP 4.592,65	COP 4.519,40
25/04/2023	COP 4.482,45	1804	1.027,26	COP 4.465,00	COP 4.489,40	COP 4.511,96	COP 4.514,90	COP 4.455,00
24/04/2023	COP 4.523,64	1481	872,31	COP 4.507,00	COP 4.480,16	COP 4.468,95	COP 4.514,90	COP 4.451,30
21/04/2023	COP 4.535,78	2054	1.299,26	COP 4.518,00	COP 4.524,33	COP 4.514,00	COP 4.545,00	COP 4.501,38
20/04/2023	COP 4.532,43	2904	1.377,10	COP 4.525,00	COP 4.535,42	COP 4.524,90	COP 4.562,00	COP 4.516,21
19/04/2023	COP 4.473,07	3578	1.843,47	COP 4.525,50	COP 4.532,24	COP 4.533,00	COP 4.560,00	COP 4.510,10
18/04/2023	COP 4.431,45	2307	1.536,14	COP 4.427,05	COP 4.474,58	COP 4.490,55	COP 4.496,00	COP 4.420,01
17/04/2023	COP 4.425,27	1875	1.124,49	COP 4.410,00	COP 4.431,05	COP 4.435,50	COP 4.441,85	COP 4.410,00
14/04/2023	COP 4.424,02	3917	1.966,26	COP 4.425,00	COP 4.424,13	COP 4.410,00	COP 4.450,00	COP 4.408,00
13/04/2023	COP 4.458,87	2110	1.325,43	COP 4.450,00	COP 4.423,40	COP 4.410,00	COP 4.450,00	COP 4.407,20
12/04/2023	COP 4.516,76	2717	1.513,04	COP 4.470,00	COP 4.458,02	COP 4.458,00	COP 4.485,00	COP 4.438,00
11/04/2023	COP 4.564,24	2350	1.656,89	COP 4.550,00	COP 4.514,51	COP 4.503,50	COP 4.550,00	COP 4.501,00
10/04/2023	COP 4.570,91	2107	1.303,97	COP 4.585,00	COP 4.562,07	COP 4.556,56	COP 4.597,80	COP 4.551,00
05/04/2023	COP 4.587,31	1415	1.228,79	COP 4.575,20	COP 4.570,06	COP 4.572,05	COP 4.586,75	COP 4.561,50
04/04/2023	COP 4.603,00	1516	992,10	COP 4.595,00	COP 4.586,36	COP 4.584,00	COP 4.601,95	COP 4.577,79
03/04/2023	COP 4.646,08	1310	822,95	COP 4.630,00	COP 4.601,72	COP 4.605,20	COP 4.630,00	COP 4.592,40
31/03/2023	COP 4.627,27	1634	874,76	COP 4.620,00	COP 4.645,95	COP 4.654,55	COP 4.666,95	COP 4.615,00
30/03/2023	COP 4.627,63	2035	1.146,91	COP 4.605,00	COP 4.629,01	COP 4.643,50	COP 4.659,00	COP 4.582,90
29/03/2023	COP 4.658,79	2152	1.249,16	COP 4.660,00	COP 4.626,29	COP 4.615,60	COP 4.667,95	COP 4.603,00

Figure 4.2: Data obtained from platform

Figure 4.2 shows the metadata obtained from the downloaded file. Which consists of the date with daily frequency, Market Representative Rate (TRM) in COP (Colombian Peso), number and amount of negotiations in millions of US dollars, and opening, weighted, closing, maximum, and minimum exchange rates.

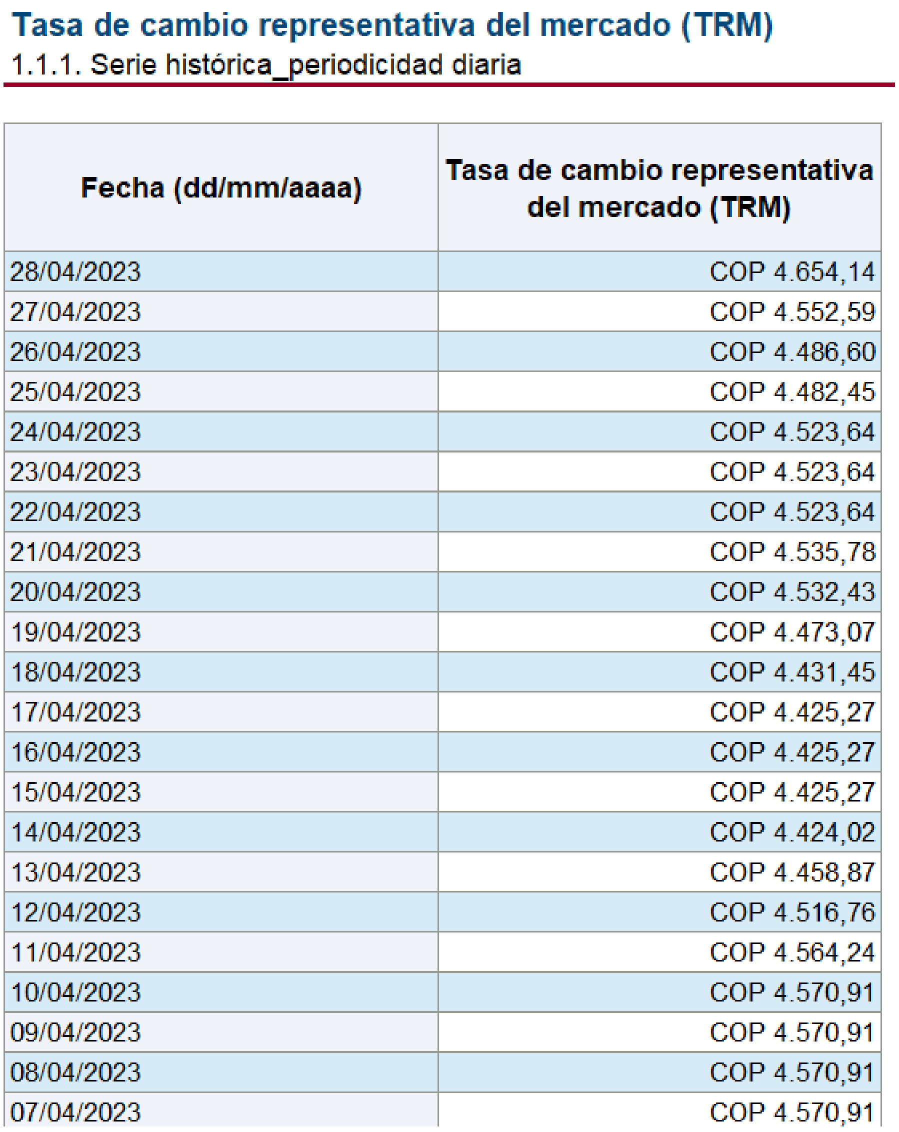
4.1.2 Data Preparation

Firstly, the TRM values obtained from the download were taken to train and test the proposed networks. As a result, the data shown in Figure 4.3.

Secondly, the dataset was divided into ten sub-sets with a size of 11441, 11083, 10725, 10010, 8580, 5720, 2860, 1430, 715, and 358 registers, respectively.

These subsets are shown in Figure 4.4.

[1]`https://www.banrep.gov.co/es/estadisticas/trm`

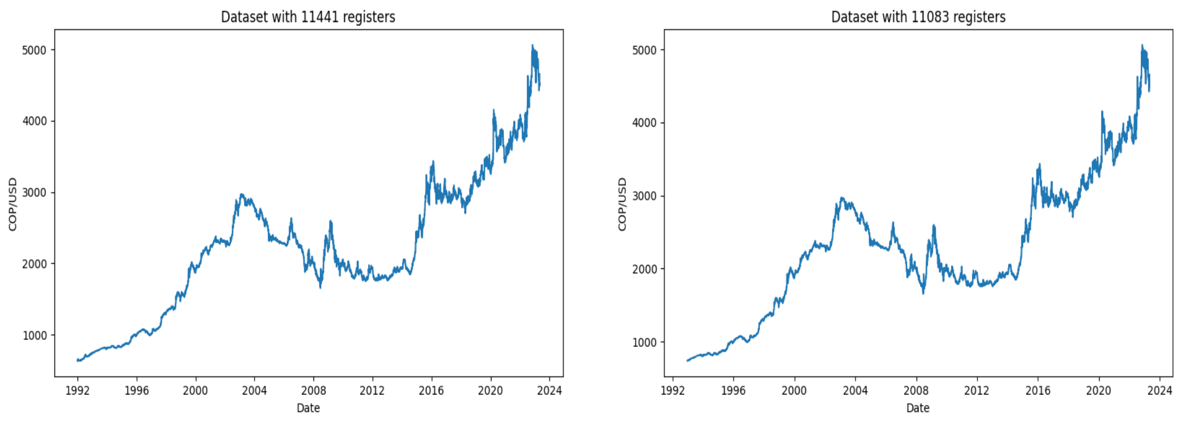


Tasa de cambio representativa del mercado (TRM)

1.1.1. Serie histórica_periodicidad diaria

Fecha (dd/mm/aaaa)	Tasa de cambio representativa del mercado (TRM)
28/04/2023	COP 4.654,14
27/04/2023	COP 4.552,59
26/04/2023	COP 4.486,60
25/04/2023	COP 4.482,45
24/04/2023	COP 4.523,64
23/04/2023	COP 4.523,64
22/04/2023	COP 4.523,64
21/04/2023	COP 4.535,78
20/04/2023	COP 4.532,43
19/04/2023	COP 4.473,07
18/04/2023	COP 4.431,45
17/04/2023	COP 4.425,27
16/04/2023	COP 4.425,27
15/04/2023	COP 4.425,27
14/04/2023	COP 4.424,02
13/04/2023	COP 4.458,87
12/04/2023	COP 4.516,76
11/04/2023	COP 4.564,24
10/04/2023	COP 4.570,91
09/04/2023	COP 4.570,91
08/04/2023	COP 4.570,91
07/04/2023	COP 4.570,91

Figure 4.3: Data used to train and test the proposed networks



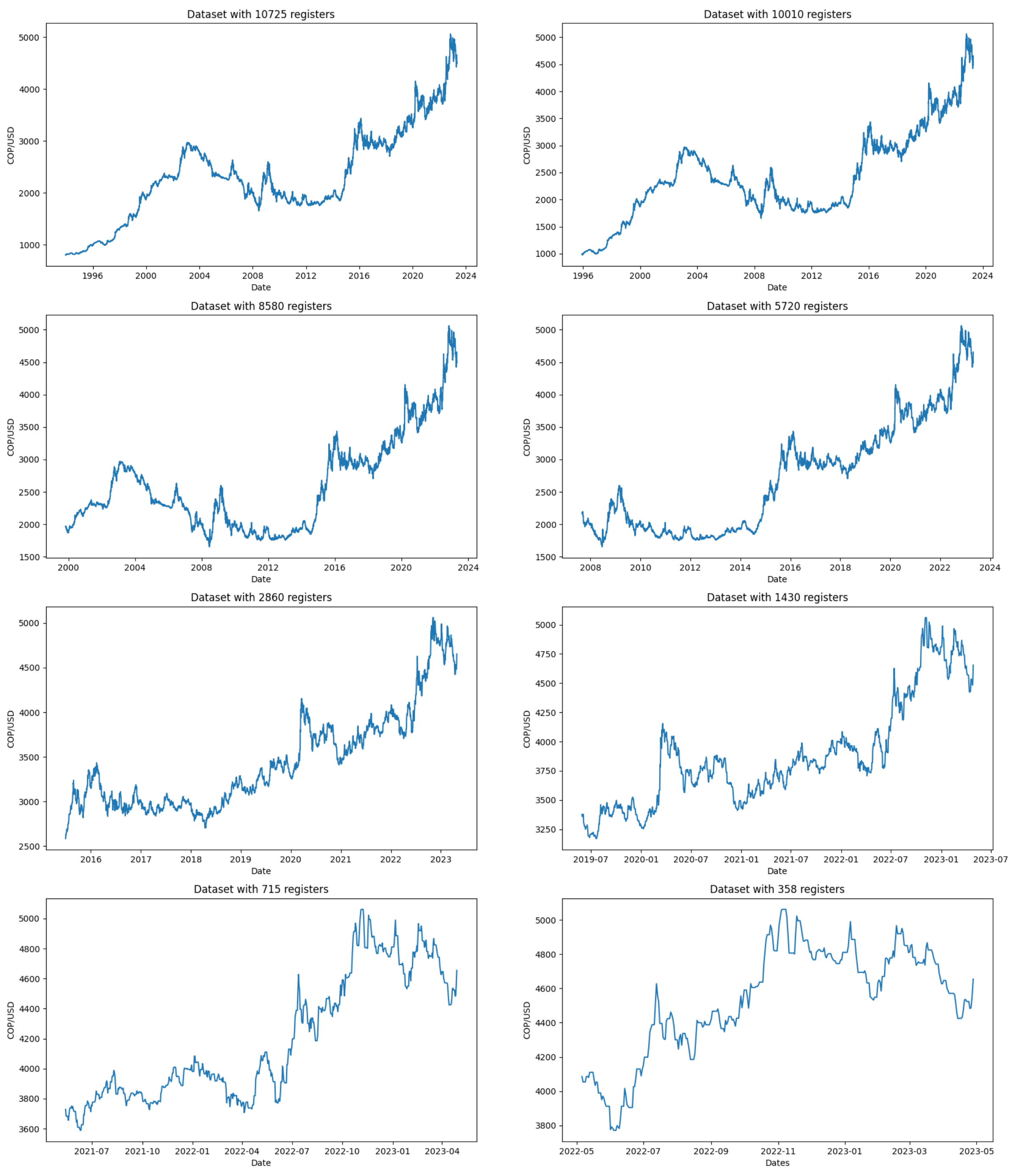


Figure 4.4: 10 subsets obtained

This process has been done to obtain the data set with the least minor influence of volatility on the records. Since this is daily data, this volatility can influence our training and testing of the proposed networks.

Moreover, resizing the dataset instead of training with the complete set offers practical advantages for various reasons. Firstly, it can improve generalization by mitigating overfitting, as the model is less likely to memorize noise or outliers in the entire dataset. Secondly, resizing helps address sampling variability issues, particularly in unbalanced or skewed datasets, by allowing random sampling to create subsets that better represent the underlying data distribution. Additionally, resizing facilitates exploratory data analysis and model prototyping, as working with a smaller subset enables faster visualization and insight generation to identify potential patterns or relationships in the data. Overall, resizing the dataset provides more manageable and efficient model training processes while still capturing essential characteristics of the data.

4.1.3 Models Preparation

This section introduces the three architectural models presented in this study. All models share some hyperparameters, such as the optimizer, training method, loss function, number of epochs, and techniques to avoid overfitting. The optimizer used is ADAM, the loss function is a mean absolute error (MAE), the number of epochs is set to one hundred, and the validation-based utilized is EarlyStopping.

Artificial Neural Network (ANN)

This model comprises three layers of a feedforward neural network (Figure 4.5a). The first layer is an input layer with one input node, while the second layer is a hidden layer with twelve neurons. Finally, the predictions are handled by a single node in the output layer. The activation function in this design is RELU, and the learning rate is set at 0.001 by default.

Choosing twelve hidden neurons in the ANN architecture could be justified based on the complexity of the forecasting task and the size of the input feature space. With a moderate number of hidden neurons, the ANN can capture essential patterns in the data without excessive model complexity.

Too few hidden neurons may result in underfitting, while too many can lead to overfitting. Therefore, selecting twelve hidden neurons balances the capture of complex patterns and prevents overfitting in the ANN architecture.

Long-Short Term Memory (LSTM)

The Long Short-Term Memory network, or LSTM network, is a recurrent neural network that uses backpropagation over time to avoid the vanishing gradient problem. LSTM networks employ memory blocks that are connected in layers rather than neurons. This model comprises three layers (Figure 4.5b). The first layer is an input layer with one input node, while the second layer is a hidden layer with fifty LSTM cells and a relu activation function. It's worth noting that the recommended model at the output layer only has one prediction node [53].

Choosing fifty hidden neurons in the LSTM architecture allows a more sophisticated memory mechanism to capture long-term dependencies in the time series data. LSTM units have multiple internal gates that regulate the flow of information, requiring a more significant number of hidden neurons to learn complex patterns effectively.

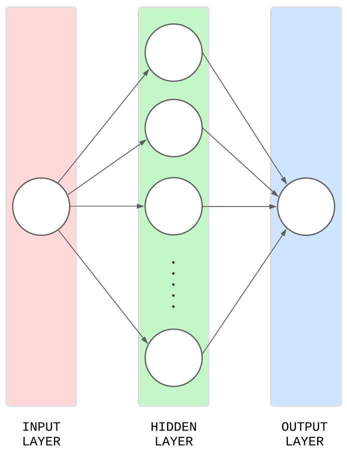
The increased number of hidden neurons in LSTM networks compared to traditional ANNs reflects their ability to model intricate temporal dynamics and capture dependencies across various time steps in the sequence.

Gated-Recurrent Unit (GRU)

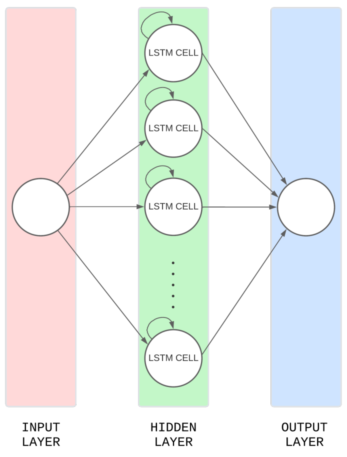
Unlike the LSTM unit, the GRU unit does not need to employ a memory unit to manage the flow of information. It has complete access to all secret states and can be used without restriction. This model comprises three levels (see Figure 4.5c). The first layer consists of one input node, whereas the second layer consists of seven GRU cells and a linear activation function. The third layer is an output layer made up of one prediction node.

Selecting seven hidden neurons in the GRU architecture balances computational efficiency with modeling capacity. GRU units have fewer parameters than LSTM units, allowing for fewer hidden neurons while still capturing relevant patterns in the data.

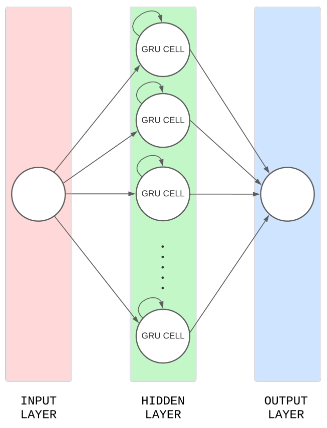
By choosing seven hidden neurons, we ensure that the GRU architecture remains lightweight and computationally efficient while still capable of capturing essential temporal dependencies in the time series data.



(a) ANN model architecture



(b) LSTM model architecture



(c) GRU model architecture

Figure 4.5: Three models

4.1.4 Train Models

The training phase for the three suggested models is the same. It comprises fine-tuning hyper-parameters such as the optimizer, epoch count, and activation function. Fine-tuning these hyperparameters aims to reduce training time, avoid over-fitting, and discover the values that best suit the proposed models.

Optimizer

Optimizer algorithms are optimization methods that help increase the performance of a deep learning model. These optimization methods or optimizers significantly impact the accuracy and speed of deep learning model training. Modify the weights of each epoch and minimize the loss function while training the deep learning optimization model. Deep learning optimizers include Gradient Descent, Stochastic Gradient Descent, Stochastic Gradient Descent with Boosting, Mini-Batch Gradient Descent, Adagrad, RMSProp, AdaDelta, and Adam. Choosing the right optimizer to forecast exchange rates is crucial to achieving accurate and efficient results. However, in this project, Adam (Adaptive Moment Estimation) was used among the various optimization algorithms available as it is famous for its adaptive learning rate and boosting capabilities [54].

Epochs

Each iteration in training to modify weight is referred to as an epoch. Thus, the number of epochs is critical since the correct number ensures proper learning. We picked one hundred epochs to train these three models in this case. However, there are different stopping rules for deciding when training ends. One of them is to prevent the training process when the error measure has seen no improvement over a certain number of epochs (early stopping). That is the rule we used [55]

4.1.5 Models Evaluation

The R-squared (R^2) and mean absolute error (MAE) are frequently used in model evaluation studies, particularly in regression tasks. Both, however, have advantages and downsides. For example, RMSE penalizes variation by giving greater weight to mistakes with larger absolute values than errors with smaller fundamental values. The MAE, on the other hand, provides equal weight to all faults. As a result, the RMSE is never smaller than the MAE [56]. Consequently, we employ MAE as a model assessment metric to improve forecast accuracy.

Chapter 5

Results and Discussion

This section summarizes the findings of two experiments designed for this project. The goal of this study is to anticipate correct results so that all studies and models assessed aim for reduced error.

5.1 Materials Description

To create this work, we utilized Tensorflow 2.13.0[1] and Keras API 2.13.1[2], both Python3-based deep learning frameworks. These were implemented in Google Colaboratory (Colab), an environment combining all these tools in one location without needing pre-configuration. Finally, the data for our models consists of ten data sets derived from the unprocessed data set examined in Section 4.1.2

5.2 Experiments Description and Setup

This section describes the two experiments created for this project (see Section 4.1.2). In short, the best-performing data size is sought in the first trial. The second experiment aims to reduce the metric error measurement.

[1]https://www.tensorflow.org/install/pip?hl=es-419
[2]https://keras.io/

5.2.1 Experiment 1

This experiment compares the three models suggested by our methodology. As a result, ten data sets with the USD/COP exchange rate are employed. In this experiment, the proportion for training and testing was 80:20 for all the models and experiments in this work. As a result, 80% of the Dataset's registers will be used for training, while the remaining 20% will be used for testing. Splitting the Dataset into training and testing sets is crucial for robust model evaluation, preventing overfitting, tuning hyperparameters, selecting the best-performing model, assessing generalization, and controlling bias in evaluation. It ensures that the model's performance metrics are reliable and accurately reflect its ability to generalize to new, unseen data. This experiment aims to discover the lowest MAE and R^2 values possible. The model and data set that yield values closest to the actual values will be used in the following two tests.

Setup

The setup for experiment 1 is presented in this subsection. It is worth noting that several hyper-parameters are shared between models, as indicated in Table 5.1. On the other hand, Tables 5.2, 5.3 and 5.4 provide a brief synopsis of the hyper-parameters employed in the models suggested in Section 4.1.3

Hyper-parameter	**Information**
Optimizer	Adam
Loss function	MAE
Epochs	100
Batch size	32
Validation-based	Early Stop

Table 5.1: Common hyper-parameters in all models.

Hyper-parameter	**Information**
Input Layer (1)	1 neuron
Hidden Layer (1)	12 neurons
Output Layer (1)	1 neuron
Activation Function	ReLU

Table 5.2: ANN hyper-parameters in all models.

Hyper-parameter	Information
Input Layer (1)	1 neuron
Hidden Layer (1)	50 neurons
Output Layer (1)	1 neuron
Activation Function	ReLU

Table 5.3: LSTM hyper-parameters in all models.

Hyper-parameter	Information
Input Layer (1)	1 neuron
Hidden Layer (1)	7 neurons
Output Layer (1)	1 neuron
Activation Function	Linear

Table 5.4: GRU hyper-parameters in all models.

5.2.2 Experiment 2

In this experiment, we will alter two hyperparameters at the same time. The train-test ratio is between 60:40, 70:30, and 80:20, and the batch size is between 1, 32, and 64. As a result, this experiment aims to see if adjusting these hyperparameters affects the MAE error and R^2. It should be noted that the model chosen for this experiment depends on the outcomes of Experiment 1.

Setup

Table 5.5 shows the values to be examined. First, we select a fixed train-test ratio, and then that combination is tested with a single batch size. As a result, the initial combination is to choose a train-test ratio of 60:40, followed by a batch size of 1. The following experiments will have the same train-test ratio as the previous ones but with 32 and 64 batch sizes, respectively. In all, 27 tests were conducted.

Hyper-parameter	Information
Train-Test ratio	60:40, 70:30, 80:20
Batch size	1, 32, 64

Table 5.5: Hyper-parameters tested.

5.2.3 Experiment 3

This section will test experiment two's optimal performance model and setups with an unseen data set. The significance of this measurement is that we want to illustrate how well our model performs in various settings. The initial data set comprises the last four months' worth of Colombian peso exchange rates (a continuation of our data); however, our model will not use this data in training or testing processes. In other words, once the training and testing processes are done, this portion of the data will be used. Remember that the original data sets were collected for training and testing from January 1, 1992, to April 28, 2023. In this sense, with this experiment, we will test our model predictability from April 29, 2023, to September 11, 2023—a dataset with 136 observations.

Setup

The setup of this experiment is determined by the results of experiments one and two. However, we know which data set will be examined, as seen in Figure 5.1.

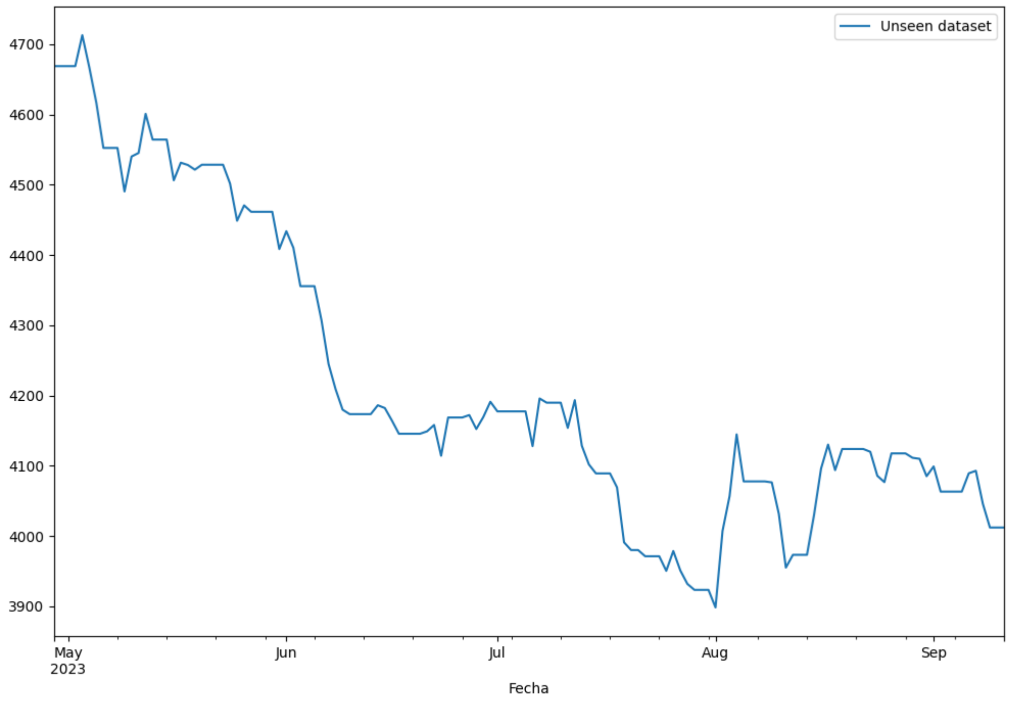


Figure 5.1: Unseen Dataset

5.3 Results

5.3.1 Results Experiment 1

This experiment's results are given in two ways: a summary table and a collection of graphs. Table 5.6 models. However, LSTM in 5720 observations stands out compared to other models, while GRU in 10725 observations stands out compared to other models. The performance of ANN is seen in Figure 5.4. Figures 5.8 and 5.12 show the performance of LSTM and GRU, respectively. Figures 5.5, 5.9, and 5.13 visually compare ANN, LSTM, and GRU performance in test data.

Observation	ANN		LSTM		GRU	
Numbers	R^2	MAE	R^2	MAE	R^2	MAE
11441	0.974	0.095	0.969	0.118	0.932	0.146
11083	0.961	0.130	0.921	0.181	0.898	0.177
10725	0.943	0.153	0.967	0.115	0.994	0.057
10010	0.995	0.052	0.986	0.089	0.969	0.123
8580	0.813	0.365	0.91 1	0.157	0.993	0.039
5720	0.994	0.039	0.993	0.041	0.990	0.055
2860	0.965	0.177	0.886	0.293	0.983	0.114
1430	0.965	0.113	0.964	0.110	0.933	0.180
715	0.869	0.107	0.928	0.065	0.929	0.068
358	0.883	0.130	0.940	0.082	0.89	0.13

Table 5.6: Summary of Experiment 1 Results

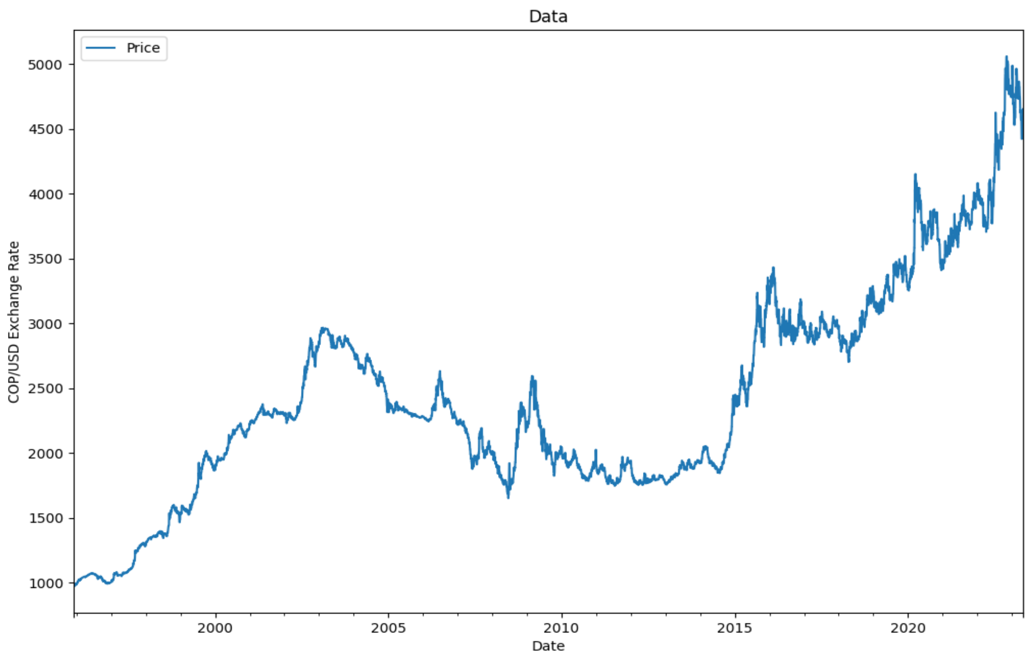


Figure 5.2: Dataset with 10010 observations

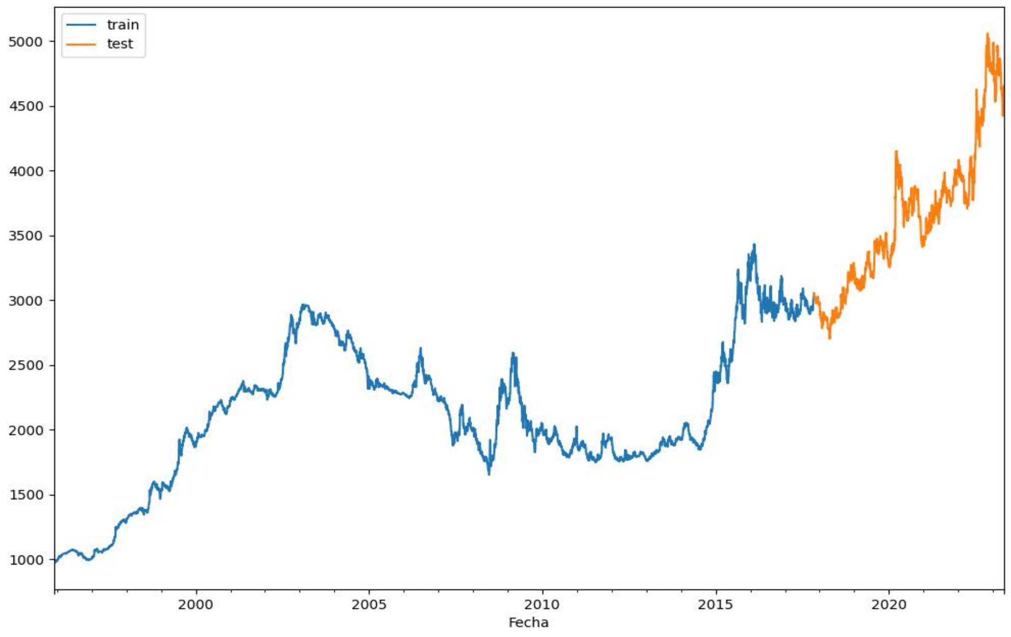


Figure 5.3: Train Test Split - 10010 observations

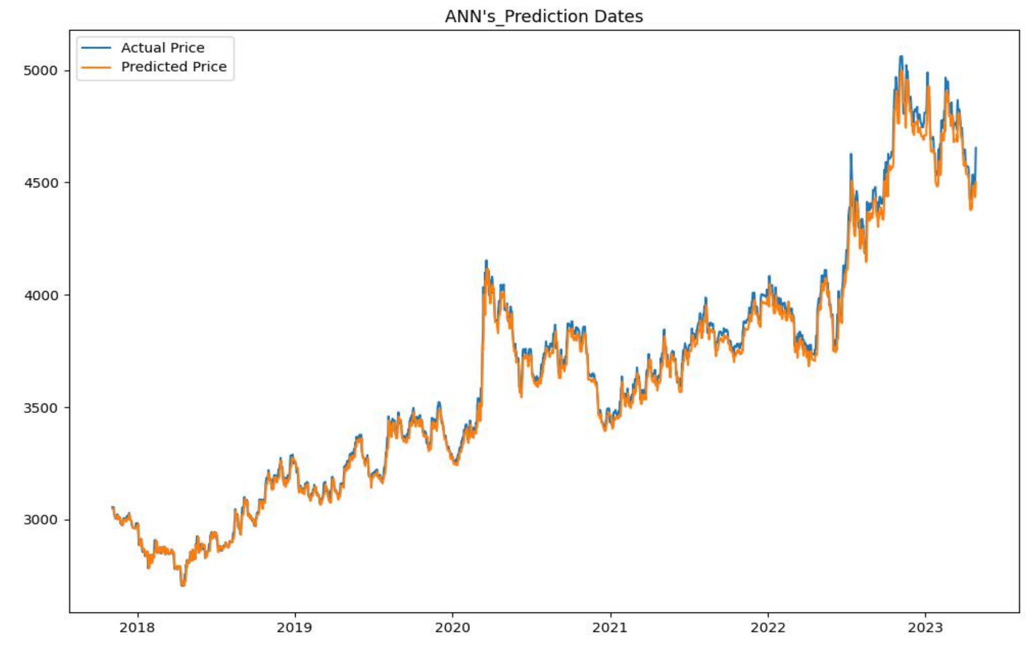


Figure 5.4: ANN - Forecasting Test data - 10010 observations

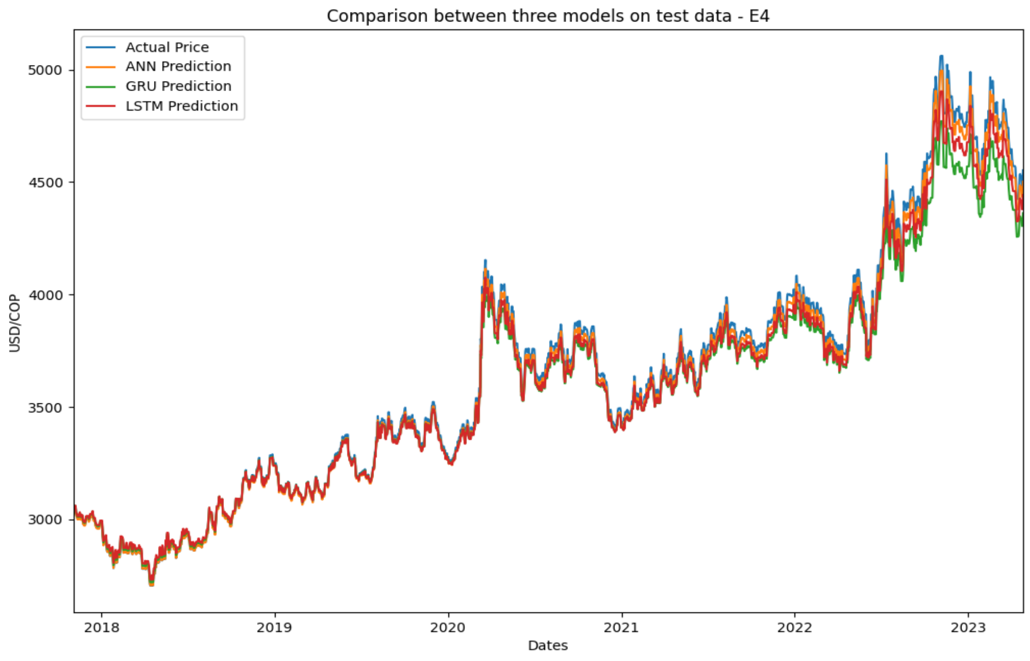


Figure 5.5: Forecasting Test data in all models - 10010 observations

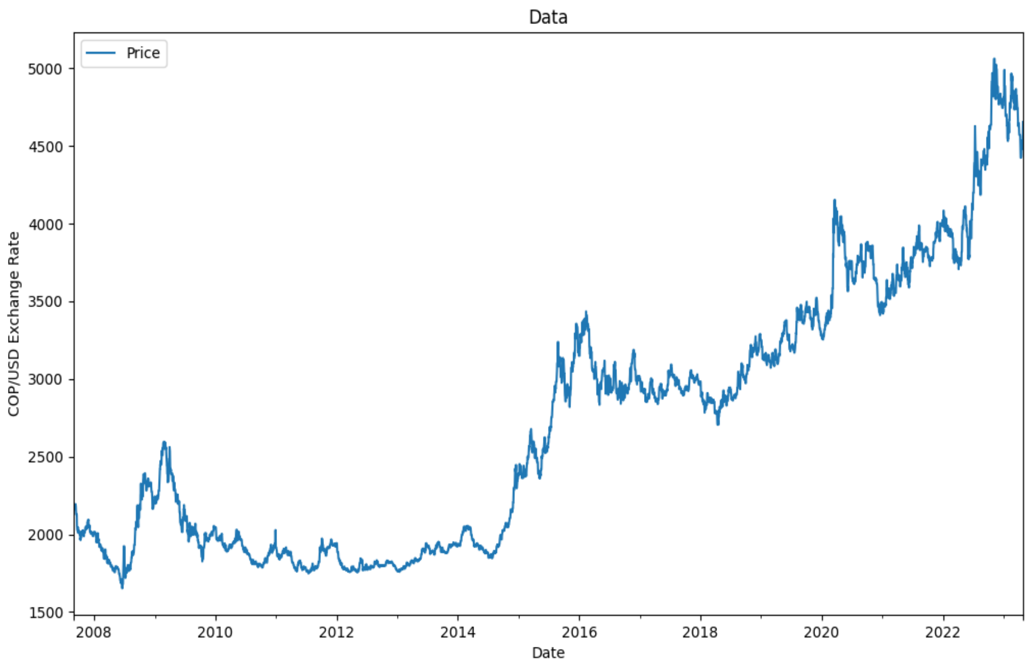


Figure 5.6: Dataset with 5720 observations

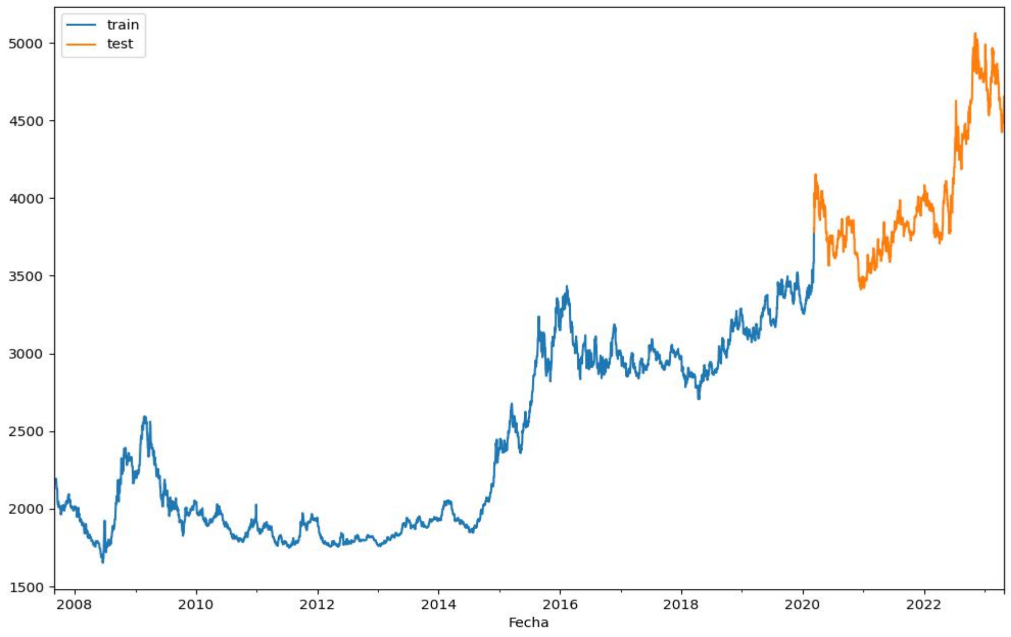


Figure 5.7: Train Test Split - 5720 observations

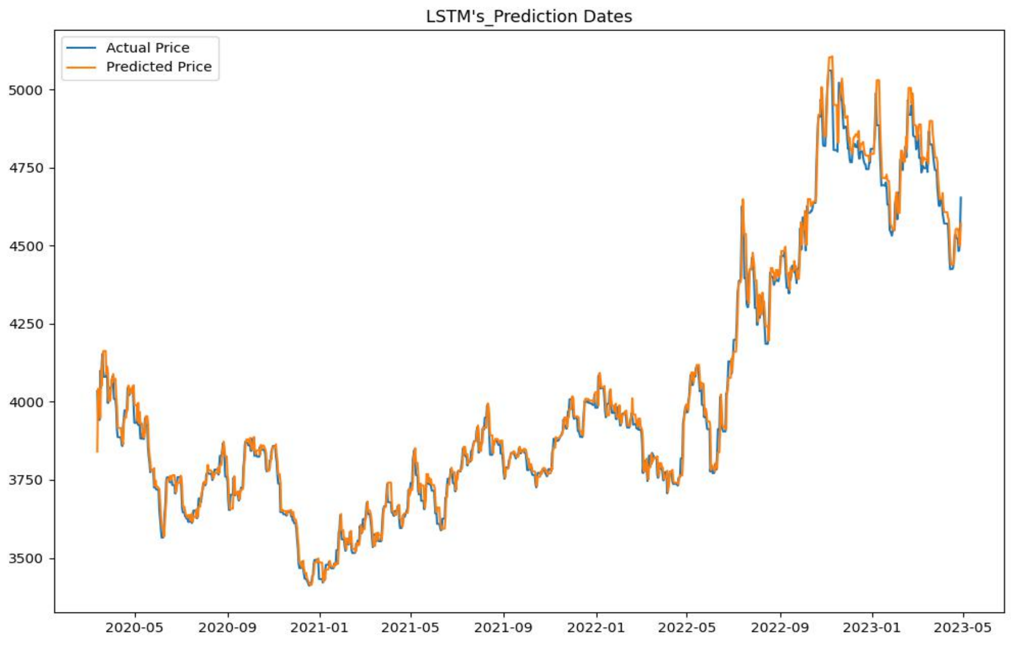


Figure 5.8: LSTM - Forecasting Test data - 5720 observations

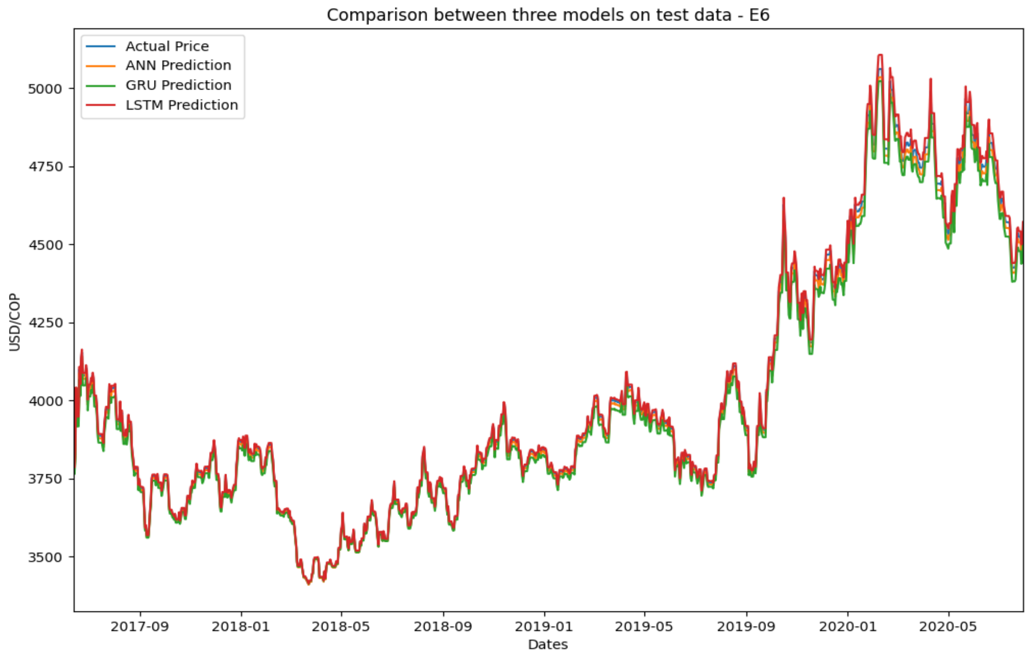


Figure 5.9: Forecasting Test data in all models - 5720 observations

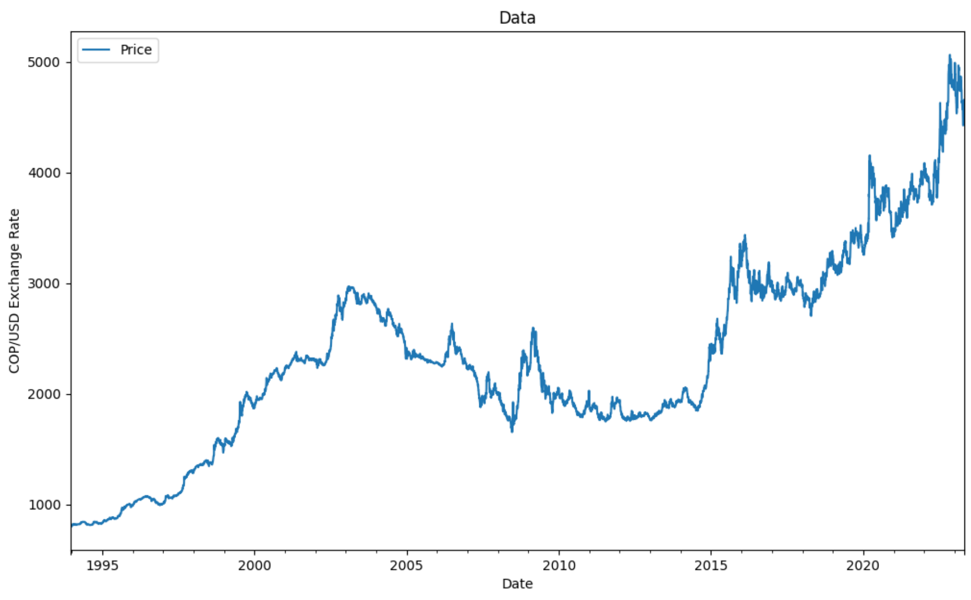


Figure 5.10: Dataset with 10725 observations

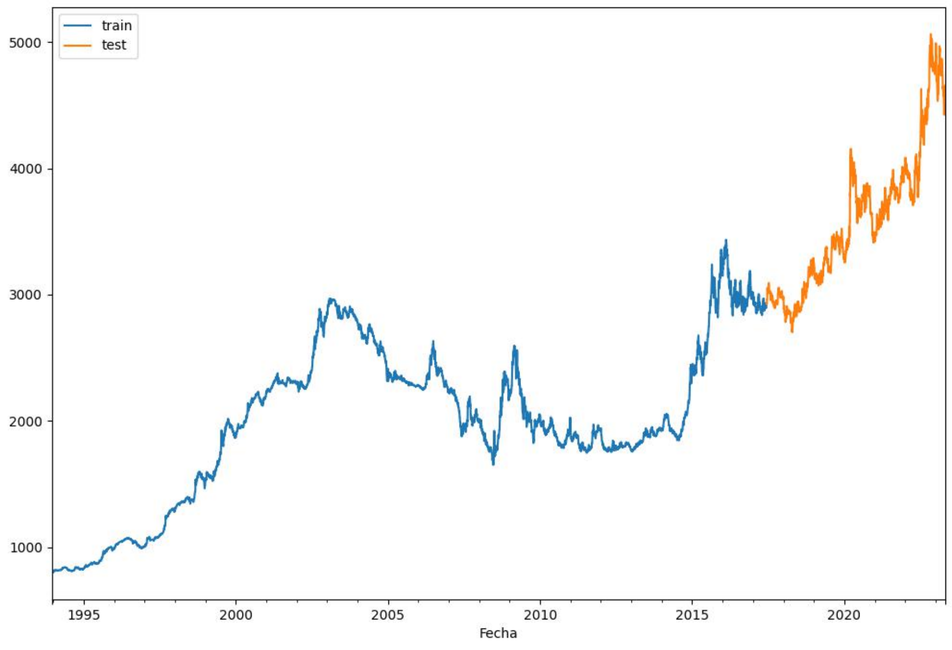


Figure 5.11: Train Test Split - 10725 observations

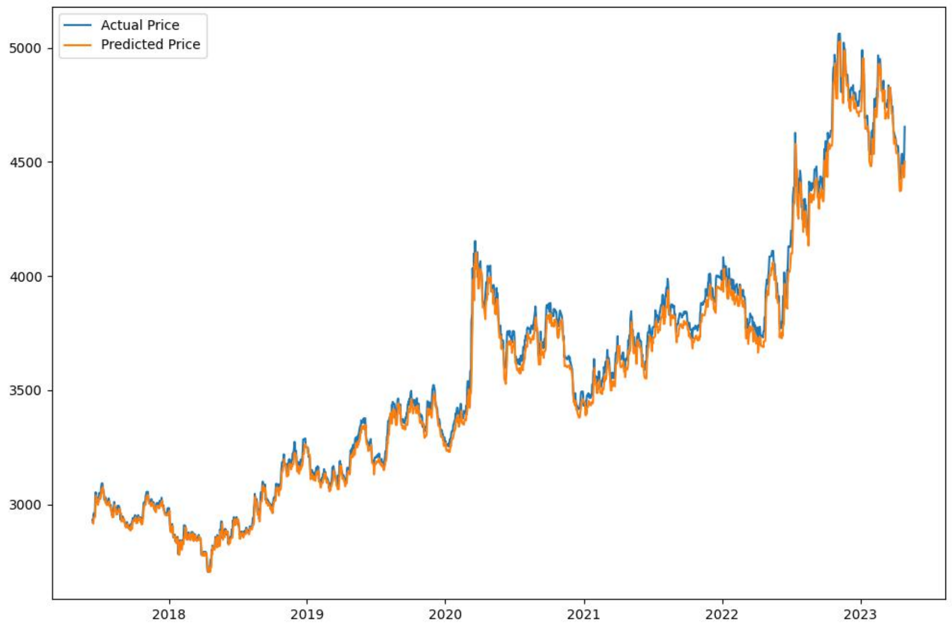


Figure 5.12: GRU - Forecasting Test data - 10725 observations

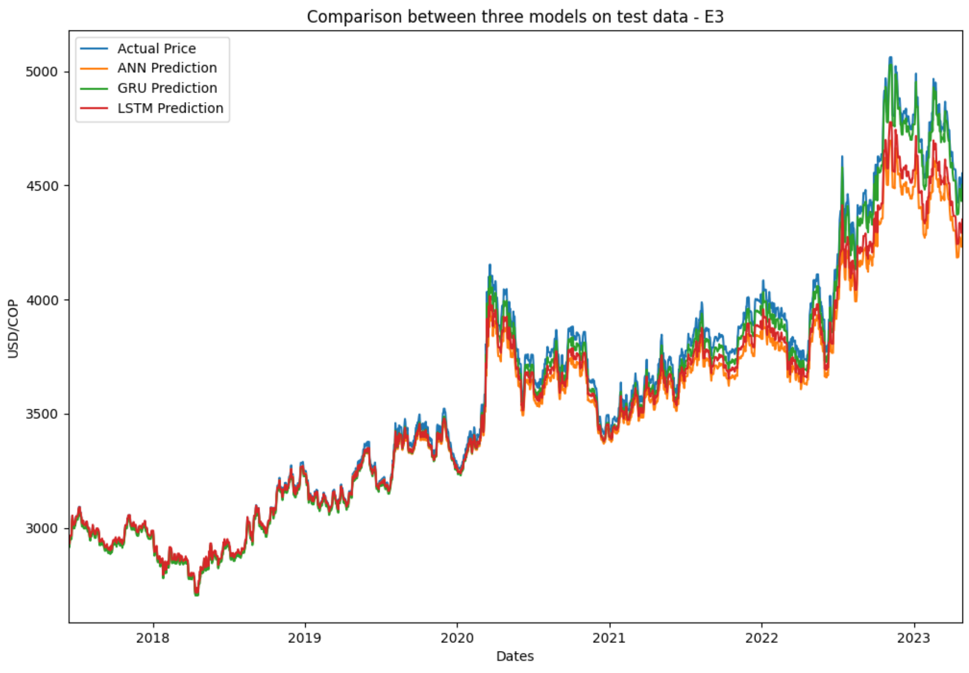


Figure 5.13: Forecasting Test data in all models - 10725 observations

5.3.2 Results Experiment 2: train-test ratio and batch size

According to the results obtained from experiment 1. We compare the best datasets for each model; in this case, we use the datasets 10725, 10010, and 5720 for experiment 2. Table 5.7 shows the tests' results on the three datasets. This way, we can deduce that the batch size affects training and testing, so the MAE and R^2 metrics fluctuate depending on this number. Obtaining that using the ANN model and the data set with 10725 observations, the training-test ratio of 70:30, together with the batch size of 64, gives us the best values of R^2 (0.999) and MAE (0.027). These results are shown in Figures 5.15, 5.14 and 5.16.

Observations Number	Train-Test ratio	Batch size	ANN		LSTM		GRU	
			R^2	MAE	R^2	MAE	R^2	MAE
10725	60:40	1	0.757	0.545	0.377	0.810	0.602	0.681
		32	0.904	0.285	0.987	0.093	0.999	0.035
		64	0.998	0.038	0.972	0.125	0.998	0.050
	70:30	1	0.755	0.515	-0.256	1.087	-0.240	1.123
		32	0.930	0.224	0.432	0.551	0.816	0.352
		64	0.999	0.027	0.865	0.276	0.995	0.065
	80:20	1	0.932	0.184	0.392	0.547	0.799	0.317
		32	0.969	0.113	0.997	0.032	0.943	0.145
		64	0.978	0.093	0.833	0.233	0.998	0.026
10010	60:40	1	0.882	0.361	0.749	0.516	-0.979	1.237
		32	0.999	0.042	0.998	0.047	0.985	0.137
		64	0.997	0.064	0.975	0.153	0.967	0.170
	70:30	1	0.882	0.361	0.749	0.516	0.749	0.516
		32	0.989	0.089	0.981	0.106	0.946	0.243
		64	0.996	0.060	0.928	0.216	0.827	0.303
	80:20	1	0.846	0.312	0.806	0.361	0.848	0.320
		32	0.993	0.061	0.991	0.068	0.615	0.431
		64	0.995	0.046	0.937	0.168	0.957	0.142
	60:40	1	0.882	0.335	0.725	0.485	0.798	0.443
		32	0.876	0.287	0.981	0.128	0.969	0.150

Table 5.7 continued from previous page

Observations Number	Train-Test ratio	Batch size	ANN		LSTM		GRU	
			R^2	MAE	R^2	MAE	R^2	MAE
5720		64	0.862	0.301	0.927	0.218	0.996	0.057
	70:30	1	0.820	0.368	-0.108	0.881	0.505	0.584
		32	0.954	0.166	0.970	0.107	0.969	0.120
		64	0.805	0.326	0.991	0.060	0.932	0.172
	80:20	1	0.957	0.105	0.963	0.102	0.941	0.130
		32	0.981	0.077	0.994	0.034	0.882	0.177
		64	0.899	0.168	0.873	0.179	0.994	0.043

Table 5.7: Summary Results of Experiment 2

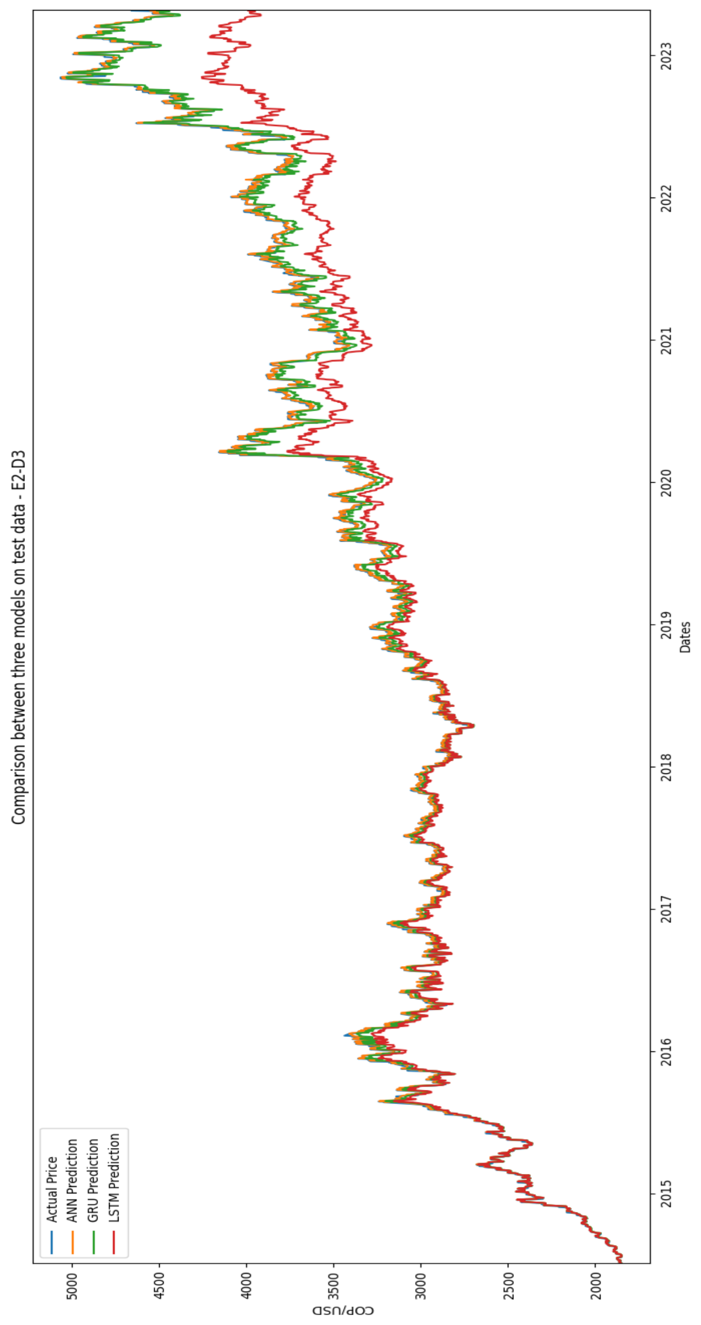


Figure 5.14: Comparison of the three models for the 70:30 train-test ratio and batch size of 64

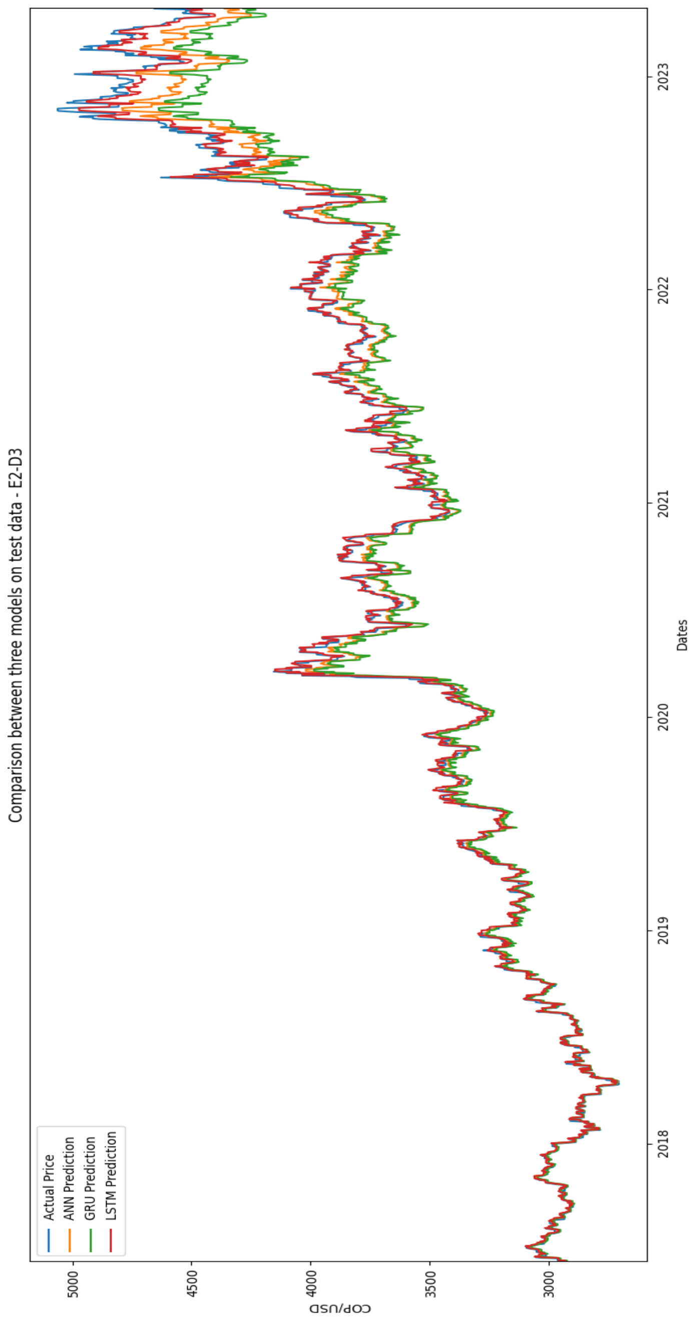


Figure 5.15: Comparison of the three models for the 80:20 train-test ratio and batch size of 32

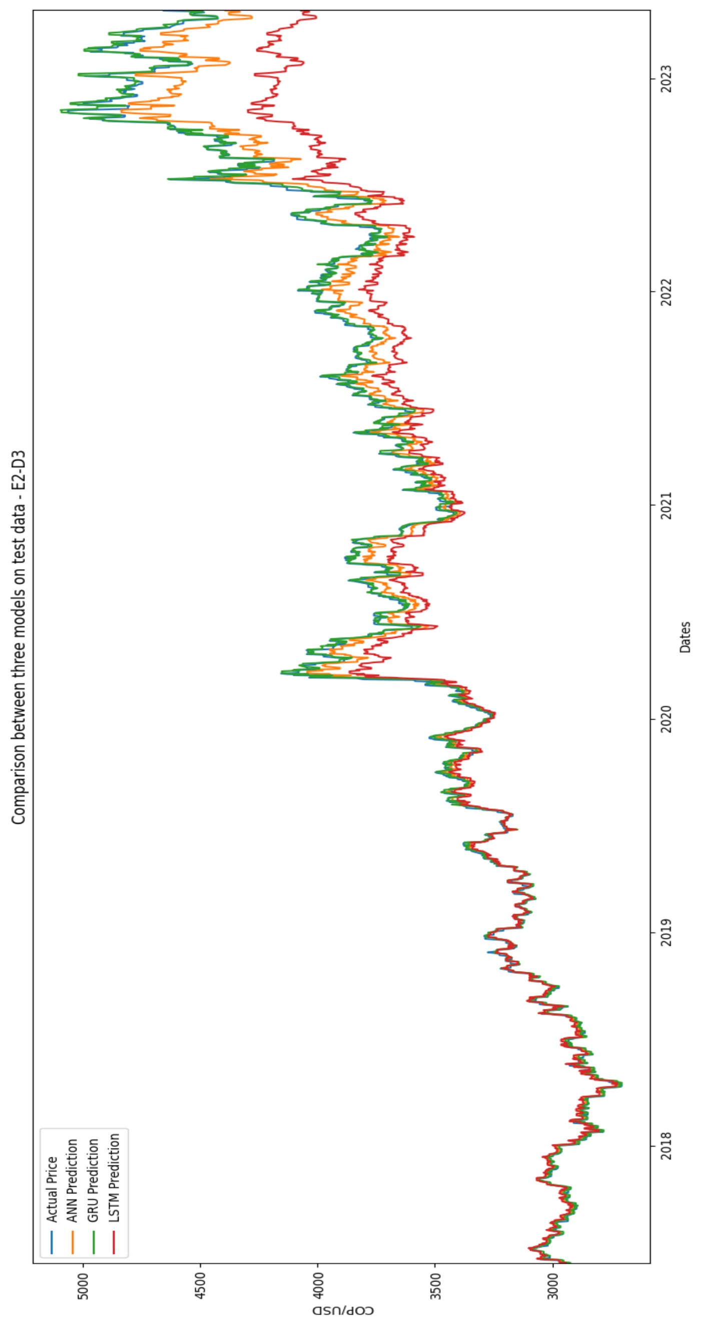


Figure 5.16: Comparison of the three models for the 80:20 train-test ratio and batch size of 64

5.4 Discussion

We will support this discussion using the results from Tables 5.6 and 5.7. First, in Table 5.6, it was observed that changing the data size directly influences the calculation of the R^2 and MAE metrics. Furthermore, it can be seen that different models can achieve higher performance if the data size is chosen correctly. Regarding the three models tested, let's start with the ANNs. Surprisingly, it can be said that this model works well with all data sets, especially in the Dataset with 10010 observations. Therefore, it should always be considered as a reference model. Regarding LSTM, its forecasting behavior was not as expected as suggested by the literature since ANNs were better. However, LSTM gave us better results in the Dataset with 5720 observations. Moreover, the GRU in the Dataset with 10725 observations returns the best R^2 and MAE metrics results. Thus, these models are the perfect choice for gaining accuracy in our experiments.

From the results of experiment one, as shown in Table 5.7, we can say that adjusting all possible hyperparameters for each model using the 3 data sets with the best performance is critical to obtaining an error reduction. In this sense, models work best with the proper test training and batch size combination. For example, for data size equal to 10725, combined with a batch size similar to 64 and test-training of 70:30, an R^2 of 0.999 and an MAE of 0.027 were obtained for the ANN model. Therefore, choosing the right hyperparameter combination can guarantee good R^2 and MAE metrics results. From this experiment, we can also say that a ratio of 80:20 and batch size of 64 showed promising results for the GRU model, obtained for R^2 equal to 0.998 and MAE of 0.026. Therefore, when working with deep learning models, the researcher must tune the most critical hyperparameters to obtain a promising result.

A detailed comparison between the artificial neural network (ANN) and the other architectures specifically designed to address the problem has been crucial to understanding why the ANN produced better results. Although differences in the R^2 metric may seem minimal, each decimal can represent a significant variation in the model's ability to explain variability in the output data. In this case, the ANN outperforms the LSTM by 0.002 and the GRU by 0.001 regarding R^2.

It is a valid observation and crucial to fully understand the process of tuning and training machine learning models. Hyperparameter optimization, which includes tuning the learning rate, number of neurons, hidden layers, activation functions, and others, is an essential part of the modeling process.

However, in this specific case, I focused on tuning parameters related to the data set due to certain considerations. While hyperparameter optimization can significantly improve model performance, it can also increase the complexity of the training and cross-validation process. Additionally, it is important to note that hyperparameter tuning is an iterative process that requires multiple experiments and validations. Tuning parameters related to the data set first can provide a solid foundation on which to build later and tune model hyperparameters.

Chapter 6

Conclusions

In conclusion, this study embarked on a comprehensive journey encompassing the acquisition, preparation, and forecasting of the Colombian peso's exchange rate against the US dollar.

The experimental phase revealed that optimal performance was achieved with specific dataset sizes, with notable results observed for datasets comprising 10725, 10010, and 5720 instances.

The ANN model excelled in predicting time series, while the LSTM and GRU models fell short of anticipated performance based on existing literature.

An essential takeaway from the study is the significance of hyperparameter tuning, particularly in deep learning models, to enhance predictive accuracy. By adjusting parameters such as batch size and test-train ratio alongside data size, considerable improvements in model accuracy were demonstrated, where the ANN model exhibited the most significant error reduction.

Chapter 7

Future Works

The prediction of time series applied to exchange rate prices is a field that must be studied in great detail. Unfortunately, a single project is not enough due to the extension of this field. Therefore, some ideas or projects that can be explored in the future are as follows:

- Improvement of the code to extract, forecast, and show the data obtained.
- Significant focus could be placed on optimizing hyperparameters within deep learning models. Hyperparameters are crucial in determining a model's performance and generalization ability, yet finding the optimal configuration can be challenging and time-consuming. One approach to address this challenge is to utilize automated hyperparameter optimization techniques. Maybe using a Keras Tuner.
- To implement multivariate time series to understand better all the variables, such as weather, oil prices, etc., that participate and affect the exchange rate.
- To experiment with new deep learning architectures, such as transformers and the attention mechanism, that have shown better results.

Bibliography

[1] J. Jose, "Introduction to time series analysis and its applications," 08 2022.

[2] V. Arcila, "¿hacia donde debería apuntar la nueva formación del médico veterinario zootecnista?" *REDVET*, vol. IX, 12 2008.

[3] Y. Lu, "Artificial intelligence: a survey on evolution, models, applications and future trends," *Journal of Management Analytics*, vol. 6, no. 1, pp. 1–29, 2019. [Online]. Available: https://doi.org/10.1080/23270012.2019.1570365

[4] S. V. Berdyugina and I. G. Usoskin, "Active longitudes in sunspot activity: Century scale persistence," *Astronomy Astrophysics*, vol. 405, no. 3, pp. 1121–1128, jun 30 2003.

[5] F. Parra, "8 Series Temporales | Estadística y Machine Learning con R," https://bookdown.org/content/2274/series-temporales.htmldescomposicion-temporal.

[6] R. Dastres and M. Soori, "Artificial neural network systems," *International Journal of Imaging and Robotics*, vol. 21, pp. 13–25, 03 2021.

[7] C. Olah, "Understanding lstm networks," Retrieved from https://colah.github.io/posts/2015-08-Understanding-LSTMs/#fnref1, 2015, web page.

[8] S. Kostadinov, "Understanding GRU Networks - Towards Data Science," https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be, nov 10 2019.

[9] M. Islam and E. Hossain, "Foreign exchange currency rate prediction using a GRU-LSTM hybrid network," *Soft Computing Letters*, vol. 3, p. 100009, Dec. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666222120300083

[10] A. Mahmoud and A. Mohammed, "A Survey on Deep Learning for Time-Series Forecasting," *Studies in Big Data*, pp. 365–392, dec 15 2020.

[11] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[12] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications: With R Examples.* Springer.

[13] B. Jan, H. Farman, M. Khan, M. Imran, I. U. Islam, A. Ahmad, S. Ali, and G. Jeon, "Deep learning in big data analytics: A comparative study," *Computers Electrical Engineering*, vol. 75, pp. 275–287, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790617315835

[14] A. Casolaro, V. Capone, G. Iannuzzo, and F. Camastra, "Deep learning for time series forecasting: Advances and open problems," *Information*, vol. 14, no. 11, 2023. [Online]. Available: https://www.mdpi.com/2078-2489/14/11/598

[15] A. Harel and G. Harpaz, "Forecasting stock prices," *International Review of Economics & Finance*, vol. 73, pp. 249–256, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1059056020303191

[16] "FORECASTING AND MANAGEMENT OF GROSS DOMESTIC PRODUCT," *Journal of International Studies*, vol. 12, no. 4, pp. 214–228, 2019, publisher: Fundacja Centrum Badań Socjologicznych. [Online]. Available: https://www.ceeol.com/search/article-detail?id=982181

[17] M. Murat, I. Malinowska, M. Gos, and J. Krzyszczak, "Forecasting daily meteorological time series using ARIMA and regression models," *International Agrophysics*, vol. 32, no. 2, pp. 253–264, Apr. 2018. [Online]. Available: http://archive.sciendo.com/INTAG/intag.2018.32.issue-2/intag-2017-0007/intag-2017-0007.pdf

[18] Y. Ensafi, S. H. Amin, G. Zhang, and B. Shah, “Time-series forecasting of seasonal items sales using machine learning – A comparative analysis,” *International Journal of Information Management Data Insights*, vol. 2, no. 1, p. 100058, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667096822000027

[19] S. D. Noboa Chavez, “Deep Learning for Agricultural Products Price Forecasting: The Case of Ecuador,” Graduation Project, UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY, Urcuquí, Dec. 2021. [Online]. Available: https://repositorio.yachaytech.edu.ec/bitstream/123456789/456/1/ECMC0088.pdf

[20] P. Newbold and C. W. J. Granger, “Experience with forecasting univariate time series and the combination of forecasts,” *Journal of the Royal Statistical Society: Series A (General)*, vol. 137, no. 2, pp. 131–146, 1974. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.2307/2344546

[21] O. Vega, “Forecasting time series by using deep neural networks,” Ph.D. dissertation, UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY, Urcuquí, Dec. 2021. [Online]. Available: https://repositorio.yachaytech.edu.ec/bitstream/123456789/430/1/ECMC0076.pdf

[22] R. Adhikari and R. K. Agrawal, “An introductory study on time series modeling and forecasting,” *CoRR*, vol. abs/1302.6613, 2013. [Online]. Available: http://arxiv.org/abs/1302.6613

[23] J. Villavicencio, “Introducción a Series de Tiempo,” Oct. 2011. [Online]. Available: http://www.estadisticas.gobierno.pr/iepr/LinkClick.aspx?fileticket=4_BxecUaZmg%3D

[24] A. Jebb, L. Tay, W. Wang, and Q. Huang, “Time series analysis for psychological research: Examining and forecasting change,” *Frontiers in Psychology*, vol. 6, 06 2015.

[25] J. Jose, “Introduction to time series analysis and its applications,” 08 2022.

[26] “Modelos aditivos y modelos multiplicativos - Minitab,” https://support.minitab.com/es-mx/minitab/20/help-and-how-to/statistical-

modeling/time-series/supporting-topics/time-series-models/additive-and-multiplicative-models/.

[27] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice.* OTexts, may 8 2018.

[28] R. Shumway and D. Stoffer, "Time series analysis and its applications : with r examples," 2017.

[29] G. Tecuci, "Artificial intelligence," *WIREs Computational Statistics*, vol. 4, no. 2, pp. 168–180, Mar. 2012. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/10.1002/wics.200

[30] Y. Lu, "Artificial intelligence: a survey on evolution, models, applications and future trends," *Journal of Management Analytics*, vol. 6, no. 1, pp. 1–29, 2019. [Online]. Available: https://doi.org/10.1080/23270012.2019.1570365

[31] D. K. Bebarta, A. K. Rout, B. Biswal, and P. K. Dash, "Forecasting and classification of indian stocks using different polynomial functional link artificial neural networks," in *2012 Annual IEEE India Conference (INDICON)*, 2012, pp. 178–182.

[32] R. Adhikari and R. K. Agrawal, "An Introductory Study on Time Series Modeling and Forecasting," *CoRR*, vol. abs/1302.6613, 2013, arXiv: 1302.6613. [Online]. Available: http://arxiv.org/abs/1302.6613

[33] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, 2010. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, chapter 5: Machine Learning Basics.

[35] Cort J. Willmott and Kenji Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model

performance," *Climate Research*, vol. 30, no. 1, pp. 79–82, 2005. [Online]. Available: https://www.int-res.com/abstracts/cr/v30/n1/p79-82

[36] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2009, chapter 2: Supervised Learning.

[37] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, p. e623, jul 5 2021.

[38] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[39] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[40] J. Chai, H. Zeng, A. Li, and E. W. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666827021000670

[41] A. S. Pillai and R. Tedesco, *Machine Learning and Deep Learning in Natural Language Processing.* CRC Press, oct 18 2023.

[42] T. , "Speech Recognition with Deep Learning - CoderHack.com - Medium," https://medium.com/coderhack-com/speech-recognition-with-deep-learning-c3633348e756, sep 15 2023.

[43] M.-C. Popescu, V. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, 07 2009.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[46] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014. [Online]. Available: https://arxiv.org/abs/1409.1259

[47] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[48] D. Tlegenova, "Forecasting exchange rates using time series analysis: The sample of the currency of kazakhstan," 2015.

[49] E. D. Solís Garcés, "Financial time series forecasting applying deep learning algorithms," Ph.D. dissertation, UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY, Urcuquí, Aug. 2021. [Online]. Available: https://repositorio.yachaytech.edu.ec/bitstream/123456789/397/3/ECMC0069.pdf

[50] A. Yadav, C. K. Jha, and A. Sharan, "Optimizing lstm for time series prediction in indian stock market," *Procedia Computer Science*, vol. 167, pp. 2091–2100, 2020, international Conference on Computational Intelligence and Data Science. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920307237

[51] S. Ranjit, S. Shrestha, S. Subedi, and S. Shakya, "Comparison of algorithms in foreign exchange rate prediction," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 2018, pp. 9–13.

[52] C. Panda and V. Narasimhan, "Forecasting exchange rate better with artificial neural network," *Journal of Policy Modeling*, vol. 29, no. 2, pp. 227–236, 3 2007.

[53] O. Surakhi, M. A. Zaidan, P. L. Fung, N. Hossein Motlagh, S. Serhan, M. .Alkhanafseh, R. Ghoniem, and T. Hussein, "Time-lag selection for time-series forecasting using neural network and heuristic algorithm," *Electronics*, vol. 10, 10 2021.

[54] A. Gupta, "A Comprehensive Guide on Optimizers in Deep Learning," https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/, jul 12 2023.

[55] H. Bhavsar and A. Ganatra, “A comparative study of training algorithms for supervised machine learning,” *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 4, pp. 2231–2307, 2012.

[56] T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature,” *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247–1250, 2014. [Online]. Available: https://gmd.copernicus.org/articles/7/1247/2014/