Democratic and Popular Republic of Algeria
Ministry of Higher Education and Scientific Research

University of Mohamed Boudiaf – M'sila

Faculty of Technology

Department of Electronics

ACADEMIC MASTER'S End of Study Thesis by:

ABUOUN Abdelrahman

Forex Market Prediction Using Recurrent Neural Network

Members of the jury:

LADJAL Mohamed	President
OUALI Mohammed Assam	Examiner
DJERIOUI Mohamed	Supervisor

Acknowledgement

Dedication

Abstract

This thesis looks at the use of Recurrent Neural Networks (RNNs) to predict Forex market movements, utilizing their ability to detect temporal dependencies in financial data. The study handles traditional forecasting difficulties through the use of LSTM and GRU models. Comprehensive data preprocessing and robust model training show that RNNs have the potential to improve the accuracy of currency exchange rate forecasts.

Keywords: Forex Market, Machine learning, Recurrent Neural Network

Contents

List of Figures

List of Tables

General Introduction

The financial market plays a crucial role in the global economy, influencing investment decision-making and financial risk management. The ability to predict stock market trends has long been a subject of intense research due to its profound implications for both individual investors and institutional stakeholders [13, 35]. With the advent of artificial intelligence (AI), the landscape of financial market analysis has been significantly transformed. This thesis aims to explore the effectiveness of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models in forecasting stock market trends, providing a comprehensive overview of the research objectives, methodology, and structure of the dissertation.

The financial market, characterized by its dynamic and often unpredictable nature, has traditionally been analyzed using fundamental and technical analysis. Fundamental analysis involves evaluating a company's financial statements and economic indicators to determine its intrinsic value, while technical analysis focuses on historical price and volume data to forecast future market movements [33, 53]. With the increasing availability of financial data and advances in computational power, AI has emerged as a powerful tool for enhancing these traditional approaches [9, 20].

Artificial intelligence, particularly machine learning, has revolutionized various domains, including finance. Machine learning models, such as neural networks, have demonstrated remarkable capabilities in identifying complex patterns and making predictions based on large datasets. Recurrent Neural Networks (RNNs), and their advanced variants LSTM and GRU, have shown significant promise in time series forecasting due to their ability to capture temporal dependencies and long-term patterns in sequential data. This thesis investigates how these models can be leveraged to enhance stock market prediction[17].

The literature review delves into the existing body of knowledge on stock market prediction, highlighting the pivotal role of datasets and pre-processing techniques in model development. It also underscores the necessity of integrating both fundamental and technical indicators to construct robust pre-

dictive models. Subsequently, the methodology section offers an in-depth examination of machine learning principles, focusing on neural networks and RNNs, thereby setting the stage for comprehending LSTM and GRU architectures. The experimental design is meticulously detailed, encompassing dataset descriptions and the evaluation metrics employed to assess model performance[22].

Following the methodology, the results and discussion section presents the empirical findings of the study. It elucidates the performance of LSTM and GRU models in predicting stock market trends, providing a comparative analysis of their respective strengths and weaknesses. This section also explores the broader implications of the results, situating them within the context of existing stock market prediction research. Moreover, the comparison with state-of-the-art approaches highlights the novelty and significance of the contributions made by this research[26].

In conclusion, this thesis aims to advance the understanding of machine learning techniques in the realm of stock market prediction. By evaluating the performance of LSTM and GRU models and benchmarking them against current methodologies, the study seeks to offer valuable insights that can inform investment strategies and enhance financial decision-making processes. The findings contribute to the ongoing discourse on the application of AI in finance, demonstrating how modern computational methods can improve the accuracy and reliability of stock market forecasts[15, 16].

Chapter 1

Review on financial markets

1.1 Introduction

Financial market prediction is one of the most challenging problems that scientists have faced for decades, especially because there is a team of scientists who believe in a random walk hypothesis (RMH) and see that financial market prices evolve according to a random walk. Hence, price changes are random, and thus it is impossible to predict. On the other side, another team of scientists believes in the efficient market hypothesis (EMH) and sees that the price in the financial markets is predictable. They then developed various theories and techniques to forecast the prices in the financial market[24, 18].

Artificial intelligence, in recent years, has entered almost all fields of life. In the beginning, scientists faced some challenges in applying artificial intelligence techniques in financial markets, and they did not achieve satisfactory results; the accuracy was low. However, with the rapid development of artificial intelligence branches and techniques, the results improved and became better.

1.2 Type of analysis

In financial market prediction, there are two well-known analytical approaches that analysts and investors use in predicting future prices: fundamental analysis and technical analysis.

1.2.1 Fundamental analysis

Fundamental analysis is a method of measuring a stock's intrinsic value. Analysts who follow this method try to find undervalued or overvalued stocks. It measures a security's intrinsic value by examining related economic and financial factors. Intrinsic value is the value of an investment based on the issuing company's financial situation and current market. Fundamental analysts study anything that can affect the security's value, from macroeconomic factors such as the state of the economy and industry conditions to microeconomic factors like the company's management. The end goal is to determine a number that an investor can compare with a security's current price to see whether the security is undervalued or overvalued. Fundamental analysis is a valuation tool used by stock analysts to determine whether a stock is overvalued or undervalued by the market. It considers economic, market, industry, and company conditions to estimate the intrinsic value of a firm and find opportunities to buy at a discount or sell at a premium. Fundamental analysis is used most often for stocks, but it is useful for evaluating any security, from a bond to a derivative[41, 27].

1.2.2 Technical analysis

Technical analysis predicts the appropriate time to buy or sell a stock used by those believing in the castle-in-the-air view of stock pricing. It is the technique of applying the tenets of the firm-foundation theory to the selection of individual stocks. Technical analysis is essentially the making and interpreting of stock charts. Its practitioners, called chartists, study the past movements of common stock prices and the volume of trading for a clue to the direction of future change. Most chartists believe that the market is only 10 percent logical and 90 percent psychological. They generally subscribe to the castle-in-the-air school and view the investment game as one of anticipating how the other players will behave. Technical analysis uses price trends and price action to create indicators. Some of the indicators create patterns that have names resembling their shapes, such as the head and shoulders pattern. Others use trend, support, and resistance lines to demonstrate how traders view investments and indicate what will happen. Technical analysts favor studying the historical price trends of the stock to predict short-term future trends[39, 12].

1.3 Datasets

There are a variety of datasets that can used in financial prediction, these datasets vary depending on the analysis method used. For fundamental analysis, the dataset is textual, and for the technical dataset, the data is numerical. In this work, we propose to classify the datasets into two main categories according to the mentioned analysis methods, and each category has sub-categories, as shown in Figure 1.1.

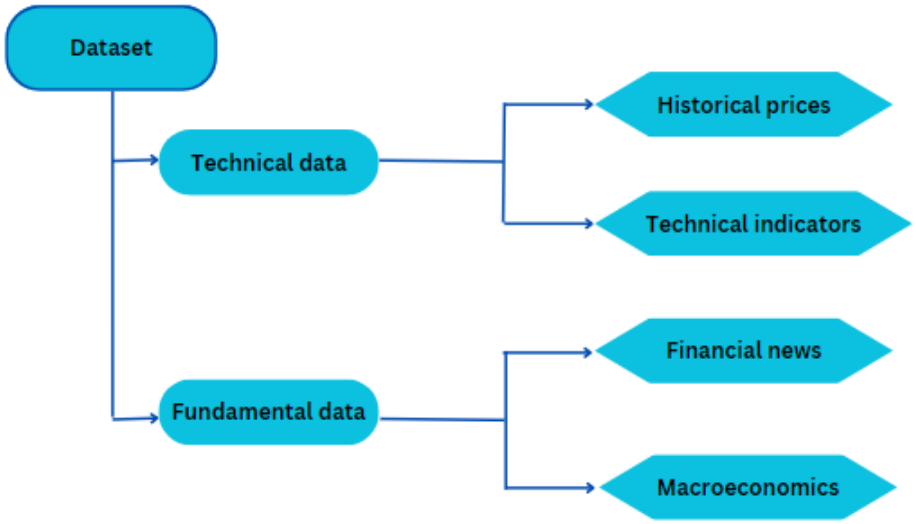

Figure 1.1: Datasets in Financial market.

1.3.1 Fundamental datasets

Fundamental datasets constitute an essential component of financial analysis, offering insights into the underlying drivers of market movements. Financial news serves as a real-time source of information, delivering updates on corporate actions, industry developments, and geopolitical events that impact asset prices. This data enables investors to stay abreast of current market conditions and make informed decisions based on the latest trends and developments. Additionally, fundamental datasets encompass macroeconomic indicators, providing a comprehensive view of the broader economic landscape. Metrics such as GDP growth, inflation rates, and employment figures offer

valuable context for understanding the fundamental forces shaping market dynamics, and guiding investors in assessing the overall health of economies and sectors. Figure 1.2 shows an example of a Fundamental dataset.

Date	Label	Top1	Top2	Top3
2008-08-08 2016-07-01	0 1	1989 unique values	1989 unique values	1988 unique values
2008-08-08	0	b"Georgia 'downs two Russian warplanes' as countries move to brink of war"	b'BREAKING: Musharraf to be impeached.'	b'Russia Today: Columns of troops roll into South Ossetia; footage from fighting (YouTube)'
2008-08-11	1	b'Why wont America and Nato help us? If they wont help us now, why did we help them in Iraq?'	b'Bush puts foot down on Georgian conflict'	b"Jewish Georgian minister: Thanks to Israeli training, we're fending off Russia "
2008-08-12	0	b'Remember that adorable 9-year-old who sang at the opening ceremonies? That was fake, too.'	b"Russia 'ends Georgia operation'"	b'"If we had no sexual harassment we would have no children..."'
2008-08-13	0	b' U.S. refuses Israel weapons to attack Iran: report'	b"When the president ordered to attack Tskhinvali [the capital of South Ossetia], we knew	b' Israel clears troops who killed Reuters cameraman'

Figure 1.2: Fundamental dataset.

1.3.2 Technical datasets

technical datasets focus on quantitative metrics derived from market activity, providing valuable insights into price movements and market sentiment. Historical prices offer a historical record of asset valuations over time, enabling investors to identify trends, patterns, and key support and resistance levels. By analyzing historical price data, traders can gain a deeper understanding of market behavior and anticipate potential future price movements. Moreover, technical indicators employ mathematical calculations to assess market momentum, volatility, and trend strength. These indicators, ranging from simple moving averages to complex oscillators, offer valuable signals for timing trades and identifying potential entry and exit points. Overall, technical datasets play a crucial role in guiding trading strategies and risk management, complementing fundamental analysis to provide a holistic view of market opportunities and risks. Figure shows 1.3 an example of a technical

dataset.

Date	# Open	# Close	# EMA50	# RSI
1999-01-06 2024-02-01	0.83 1.6	0.83 1.6	1.16 1.19	12.2 91.5
1999.01.06 20:00	1.16	1.1623	1.173195294	21.88449848
1999.01.07 00:00	1.1623	1.1634	1.173238431	24.59941323
1999.01.07 04:00	1.1635	1.1641	1.173265882	26.35353964
1999.01.07 08:00	1.1657	1.1657	1.173328627	30.34250826
1999.01.07 12:00	1.1656	1.1694	1.173473725	38.62173496
1999.01.07 16:00	1.169	1.1713	1.173548235	42.40726098
1999.01.07 20:00	1.1712	1.1712	1.173544314	42.25953143
1999.01.08 00:00	1.1713	1.1713	1.173548235	42.47533862
1999.01.08 04:00	1.1711	1.1685	1.173438431	38.17317777
1999.01.08 08:00	1.1686	1.1666	1.173363922	35.54244154

Figure 1.3: Technical dataset.

1.4 Pre-processing

Preprocessing is the process of handling problems in datasets, such as noises, missing values, or other faults, that can dramatically impair the performance of machine learning models. These problems can be handled by preprocessing algorithms. During this process, the data will be cleaned up and converted appropriately. In order to get the data ready for a machine learning model, pre-processing is an essential step that must be taken. Specifically, it entails converting the input into a format that is suitable with the model and maximizing convergence in order to enhance the quality of the outputs.

1.4.1 Pre-processing of technical dataset

In this type of data, in which the actual value depends on the previous value directly, we should not have to eliminate any row of the dataset, so firstly we should have to ensure that the collected data is proper, and there are no missing values, and does not contain defects as much as possible. Some of the articles applied normalization as an essential step before feeding data to the model to eliminate the destructive effect of outliers and to make convergence

better, and other articles do not apply any pre-processing[36, 29, 47].

Based on the literature that we have consulted for this study, we can conclude that if the dataset is in the same range and does not have missing values or other defects we should not have to apply the preprocessing step.

1.4.2 Preprocessing of fundamental dataset

Fundamental data like news articles, social media tweets, etc. is textual data, and to process this type of data we use a subfield of Natural Language Processing(NLP) called sentiment analysis. Sentiment analysis also known as opinion mining is a subfield of NLP that analyzes textual data to determine whether a text conveys a positive, negative, or natural. To analyze and identify these textual data, firstly we have to remove useless information from the extracted articles, and then we need a sentiment dictionary(sentiment lexicon) which contains words and sentiment types. We can create a sentiment lexicon or use an existing one, and there are a variety of sentiment lexicons that we can use in this step like NTUSD, HowNet, SentiWord Net[5, 31, 4].

1.5 Conclusion

In conclusion, the chapter on stock market prediction using machine learning examines financial market complexities and forecasting challenges. It discusses the Random Walk and Efficient Market Hypotheses, noting machine learning's potential to uncover patterns despite these theories. The chapter covers fundamental and technical analysis, emphasizing the importance of diverse, high-quality datasets and preprocessing steps. It highlights the promise of machine learning in improving forecast accuracy and decision-making in financial markets through interdisciplinary approaches and robust data handling.

Chapter 2

Deep learning

2.1 Introduction

In this chapter, we will carefully present the core and conceptual foundations of machine learning. We will begin with a mathematical overview of machine learning to establish a solid base for understanding. Following this, we will introduce neural networks and the corresponding mathematical principles that underpin their functionality. Finally, we will delve into recurrent neural networks, exploring their unique characteristics and applications in depth. This comprehensive approach will provide a robust framework for grasping the fundamental concepts and advanced techniques in machine learning.

2.2 A mathematical overview of machine learning

2.2.1 Learning process

In machine learning, a model is a mathematical function that represents the relationship between input variables, known as features, and an output variable, known as the target[38]. This function can take various forms, including linear, polynomial, or other non-linear functions. The general form of a linear model is given by:

$$\hat{y} = \sum_{i=0}^{n} \omega_i * x_i \tag{2.1}$$

In Equation 2.1, $\hat{y}$ is the predicted target, representing the model's output. The term x_i denotes the features, which are the variables of this equation.

The parameter n represents the number of features in the dataset, and ω_i are referred to as weights or parameters. These weights are crucial as they determine the influence of each feature on the predicted output. We will explain the mathematical significance and role of these weights in the learning process in the following example[7].

In this example, we will use a virtual dataset with one feature to illustrate the mathematical concept of the learning process. Before selecting our model, we examine the dataset to determine if any preprocessing steps are necessary. After preprocessing, we analyze how the feature's samples are distributed in relation to the output y. This step is essential to understand the relationship between the feature and the target variable, which will inform our model selection and training approach.

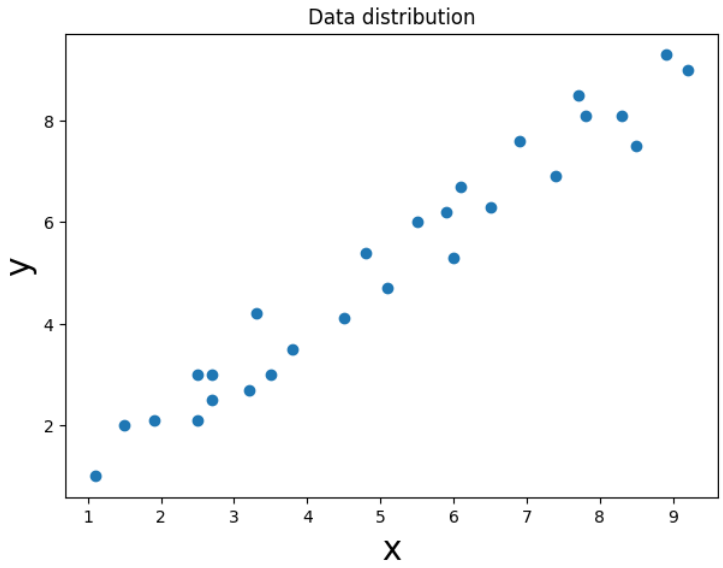

Figure 2.1: Dataset distribution.

In Figure 2.1, we observe that the distribution of points is approximately linear. Therefore, we employ a linear model as described in Equation 2.2. In this model, ω_0 represents the bias, which shifts the curve vertically. On the other hand, ω_1 represents the slope, which determines the incline of the curve. This linear model helps us capture the relationship between the feature and the target variable effectively.

$$\hat{y} = \omega_0 + \omega_1 * x \tag{2.2}$$

The learning process is geared towards determining optimal values for these weights such that the curve represented by $\hat{y}$ closely approximates the data distribution. This entails ensuring that for each sample in the dataset, the predicted output is highly consistent with the actual output. In essence, the objective is to minimize the discrepancy between the predicted values and the ground truth, thereby enhancing the model's predictive accuracy and fidelity to the underlying data distribution[19].

2.2.2 Loss function

Loss functions serve as mathematical measures to quantify the disparity between the predicted output generated by the model and the actual output observed in the dataset. While a plethora of loss functions exists, the choice of a specific one is contingent upon the nature of the task at hand[17]. In the case of regression tasks, the mean squared error (MSE) is a commonly employed metric. MSE calculates the average squared difference between the predicted output and the actual output. Mathematically, it is represented as:

$$J(\omega_i) = \frac{1}{m}\sum_{i=0}^{m}(\hat{y^i} - y^i)^2 \tag{2.3}$$

This function is characterized by its continuity and differentiability, properties that are crucial in calculus for determining minimum and maximum values of a function. Therefore, in the learning process, the objective is to iteratively update the weights of the model in order to minimize the loss function. The ultimate aim is to find the values of the weights that either achieve the minimum value of the loss function or converge to a value close to it. This iterative optimization process involves adjusting the weights in a manner that systematically reduces the prediction error, thereby enhancing the model's accuracy and effectiveness in capturing the underlying patterns in the data.

2.2.3 Gradient Descent

It is an optimization algorithm used to find the optimal values for the weights that achieve the minimum loss function[38]. It works by iteratively updating the values of the weights, this process is called training the model. But how does this algorithm update the values to new values closer to the optimal values rather than farther away? In mathematics, the first derivative of a continuous function in a point represents the slope at this point, if it is

equal to a positive value, that means the curve at this point is increasing, and if it is equal to a negative value, that means the curve at this point is decreasing[51]. so we update ω_0 and ω_1 by deriving J(ω) in 2.3 partial derivation to know whether we should increase or decrease their values if the derivative is positive that means the curve is increasing at these points and the minimum value is lower than the actual value, so we should decrease the actual values and vice versa. the gradient descent equation that achieves this is given by:

$$\omega_i = \omega_i - \ell * \frac{\partial}{\partial \omega_i} J(\omega_i) \tag{2.4}$$

In 2.4 ℓ is the learning rate and it is a small value that controls the step size at which the optimization algorithm updates the model's weights. In our example, the gradient descent equations are as follows:

$$\omega_0 = \omega_0 - \ell * \frac{2}{m} \sum_{i=0}^{m} (\hat{y} - y) \tag{2.5}$$

$$\omega_1 = \omega_1 - \ell * \frac{2}{m} \sum_{i=0}^{m} ((\hat{y} - y) * x) \tag{2.6}$$

after applying the gradient descent to the model in 2.2, we get the result in Figure 2.2 where $\omega_0 = 0.11447$ and $\omega_0 = 0.98483$. The final model is $\hat{y} = 0.98483x + 0.11447$.

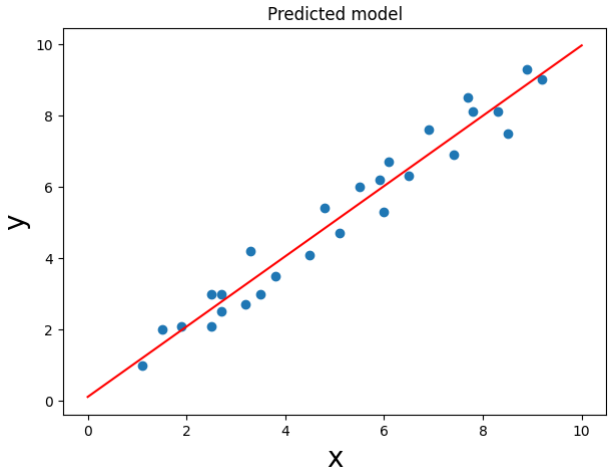

Figure 2.2: Linear regression model.

2.3 Neural network

A neural network (NN) is a computational model inspired by the structure and functionality of the human brain, as illustrated in Figure 2.3. It consists of interconnected nodes, or neurons, organized into layers. These networks are capable of learning complex patterns and relationships from data. From a mathematical perspective, it is important to understand what lies within these networks and what exactly a neuron or node signifies. A neuron in a neural network represents a mathematical function that takes inputs, applies weights, adds a bias, and then passes the result through an activation function. This process allows the network to learn and model complex relationships within the data[15].

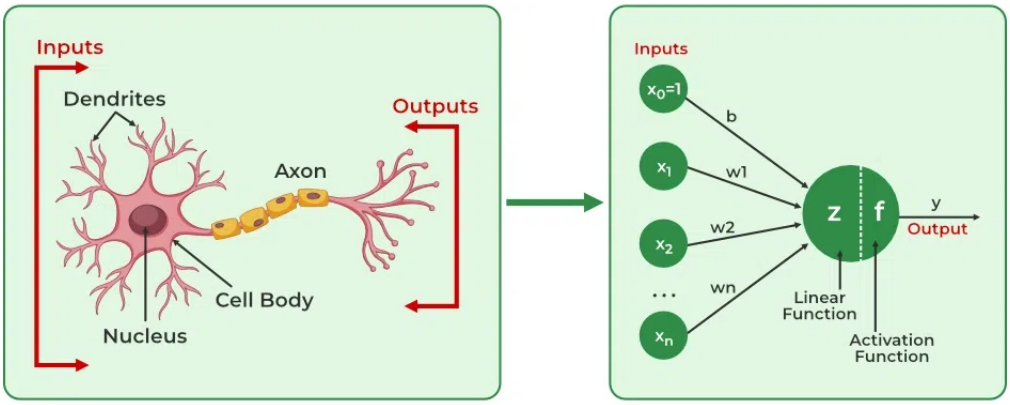

Figure 2.3: Neural network and human brain.

In the example from the previous section, the data distribution was linear, so we used and learned a linear model. If the data were polynomial, we would use a polynomial model. However, in real applications, data distributions are typically more complex. Mathematically, we can combine multiple models to create a more complex one. For instance, in Figure 2.4, three linear models are summed to form a single, more complex model. In neural networks (NNs), each neuron represents a simple model, and by summing these models across layers, we construct a highly complex model in the output layer. This layered approach allows NNs to capture and learn intricate patterns in the data.

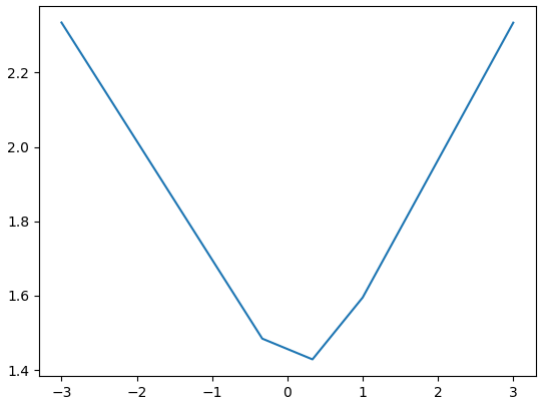

Figure 2.4: Neural network model.

2.3.1 Activation functions

In neural networks, if all the layers use only linear transformations, the output of the network is still a linear combination of the inputs, in this case, the model will not be able to recognize complex patterns and then will not be able to handle complex problems, so we need to break the linearity in the layers to capture complex patterns instead of linear ones. Activation functions break this linearity by applying a nonlinear transformation to the output of each layer before passing it to the next layer, this allows the neural network to build complex patterns[25]. The main condition when applying any activation function is to be a differentiable function because in the gradient descent algorithm, we use the derivative of this function in updating the weights[7]. Many activation functions can be expressed mathematically in a neural network's node. Here are some of them:

Sigmoid function

It is a mathematical function that converts the input value to a value between 0 and 1, it is particularly used in the output layer in NN where the output is a probability value between 0 and 1 so we use it with binary classification problems, and according to a defined threshold we say the output belongs to class 1 or class 0[50]. Figure 2.5 shows sigmoid curve and it is defined as:

$$a(z) = \frac{1}{1 + e^{-z}} \tag{2.7}$$

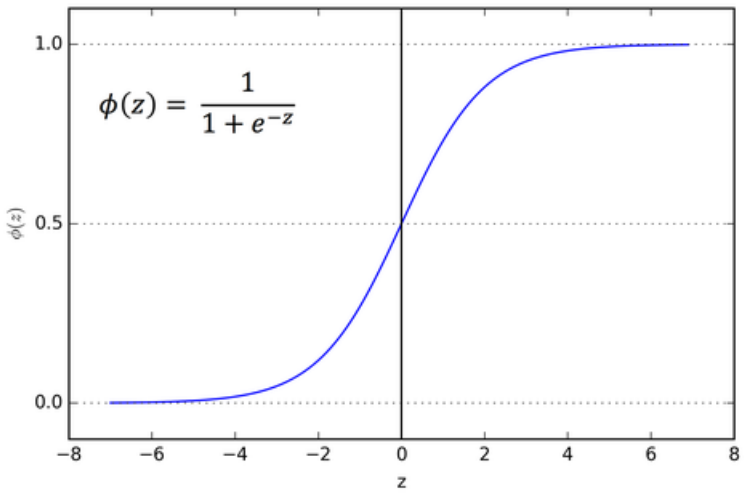

Figure 2.5: Sigmoid function.

Softmax activation function

It is a mathematical function that converts the input vectors into a probability distribution vector between 0 and 1, where the probabilities sum to 1, and the input vector belongs to the highest probability[49], as illustrated in Figure 2.6, it is defined as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{2.8}$$

where z_i is the ith component of the input vector, and n is the total number of components. This activation function used with multi-class classification problems.

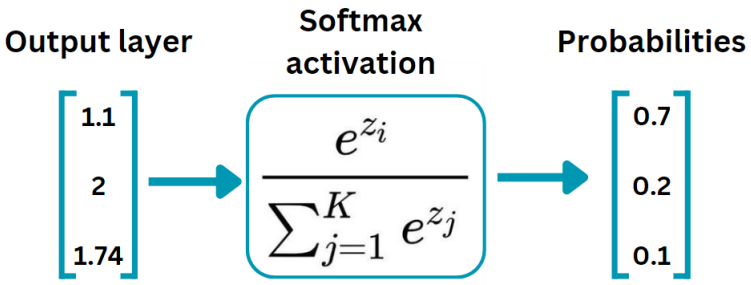

Figure 2.6: Softmax activation functin.

Hyperbolic Tangent Activation Function

It is a mathematical function that converts the input value to a value between -1 and 1, as shown in Figure 2.7, providing symmetry about the origin. This characteristic aids in maintaining data normalization within the network, facilitating gradient propagation, and potentially contributing to faster convergence during training. it is given by:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.9}$$

It is commonly employed in hidden layers and recurrent neural network (RNN) architectures, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), where the symmetric range is particularly useful for sequential data processing.

One limitation of the tanh function is that it can be affected by the vanishing gradient problem, where the gradients become very small during backpropagation, making it difficult to train the network effectively. To mitigate this issue, other activation functions like ReLU have been developed[25].

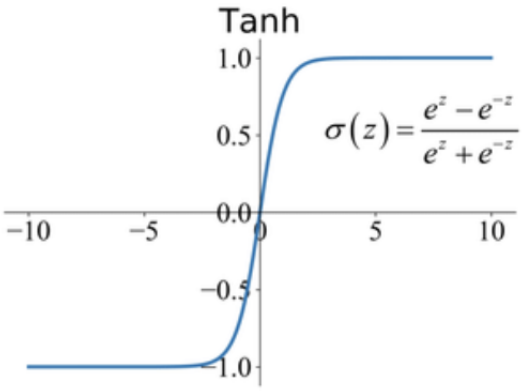

Figure 2.7: Tanh activation function.

Rectified Linear Unit

The Rectified Linear Unit (ReLU) activation function is a cornerstone in neural networks, renowned for its computational simplicity and ability to address the vanishing gradient problem commonly seen with other activation functions like sigmoid and tanh. Defined mathematically as: ReLU(x)=max(0,x), it introduces nonlinearity while retaining computational efficiency, allowing

for faster training in deep architectures. ReLU's sparse representations, created by outputting zero for non-positive inputs, lead to more efficient computations and reduced memory usage. Despite these benefits, ReLU has limitations, such as the "dying ReLU" problem, where neurons can become inactive due to persistently negative inputs. Variants like Leaky ReLU, Parametric ReLU (PReLU), and Exponential Linear Unit (ELU) aim to address these issues by incorporating a small slope or smooth transitions for negative inputs. ReLU's widespread adoption in deep neural networks, especially in convolutional layers and deep feedforward architectures, underscores its significant role in advancing deep learning methodologies[1].

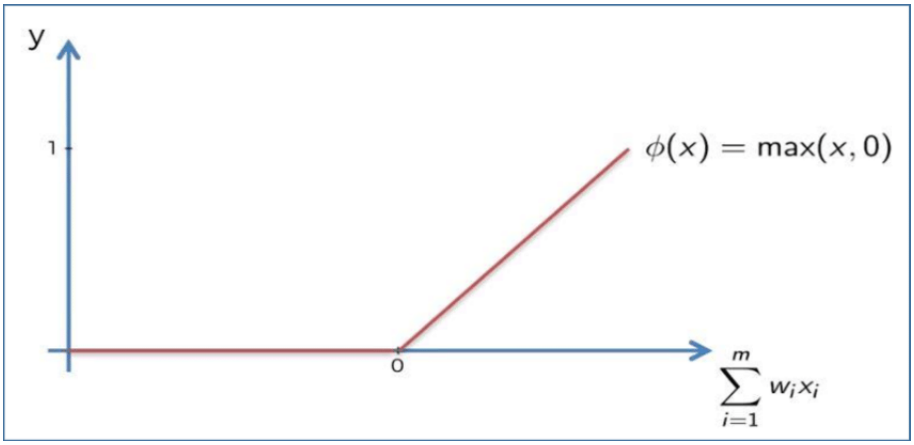

Figure 2.8: Relu activation function.

2.3.2 Forward propagation

Forward propagation is the process by which data is passed through a neural network from the input layer to the output layer. Each layer of the network performs calculations on the input data and passes the results to the next layer. This process continues until the data reaches the output layer, where the final predictions are made. To illustrate the forward propagation process, consider the network shown in Figure 2.9. In this network, the input data is first processed by the initial layer, which applies weights and biases and passes the result through an activation function. The output of this layer becomes the input for the next layer, and this process repeats through all subsequent layers until the final output is produced. This step-by-step transformation of data through the network layers is what enables neural networks to learn and model complex relationships within the data[38].

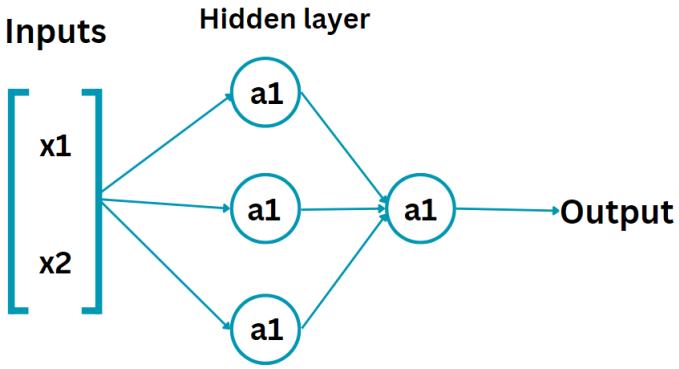

Figure 2.9: Network with one hidden layer.

We begin this process by feeding the input vector into the first hidden layer and computing the output from each perceptron. Subsequently, we apply the activation function to the outputs to introduce non-linearity. The mathematical equation defines the operation of the first layer is expressed as follows:

$$a_1 = b_1 + \omega_1 1 * x_1 + \omega_1 2 * x_2 \tag{2.10}$$
$$a_2 = b_2 + \omega_2 1 * x_1 + \omega_2 2 * x_2 \tag{2.11}$$
$$a_3 = b_3 + \omega_3 1 * x_1 + \omega_3 2 * x_2 \tag{2.12}$$

After computing the output of this layer and applying the activation function, we utilize it as inputs to the next layer. This process is iterated for each perceptron in all layers until the final output is obtained. To simplify these calculations, we leverage linear algebra and vectorize these equations. In linear algebra notation, the previous equations can be represented as follows:

$$[a_1] = [b_1] + [x_1 \quad x_2] \begin{bmatrix} w_{11} & w_{11} \\ w_{11} & w_{11} \end{bmatrix} \tag{2.13}$$
$$[a_2] = [b_2] + [x_1 \quad x_2] \begin{bmatrix} w_{21} & w_{21} \\ w_{21} & w_{21} \end{bmatrix} \tag{2.14}$$
$$[a_3] = [b_3] + [x_1 \quad x_2] \begin{bmatrix} w_{31} & w_{31} \\ w_{31} & w_{31} \end{bmatrix} \tag{2.15}$$

Finally, we vectorize all equations in this layer as follows:

$$l1 = [b_1 \quad b_2 \quad b_3] + [x_1 \quad x_2] \begin{bmatrix} w_{11} & w_{11} & w_{11} \\ w_{11} & w_{11} & w_{11} \end{bmatrix} \tag{2.16}$$

2.3.3 Backward propagation

Backward propagation, often referred to as backpropagation, is a method used in NN for training neural networks. It is an algorithm that adjusts the weights and biases of a neural network in response to errors in its output. It does this by calculating the gradient of the error function with respect to each weight and bias in the network, and then adjusting those weights and biases to minimize the error. The chain rule plays a central role in this process, as it allows the decomposition of complex gradient calculations into simpler parts by considering the derivative of each layer's activation function with respect to its input. Through partial derivatives, backpropagation calculates how changes in the network's parameters (weights and biases) affect the loss function, providing the necessary information to adjust these parameters using an optimization algorithm, such as stochastic gradient descent or Adam. This iterative process, repeated across multiple epochs, is fundamental to the supervised learning paradigm in neural networks, driving the model's convergence towards a lower loss and improved predictive performance[38].

For network with tow hidden layer the backward propagation equations is as follows:

$$E = \frac{1}{2}(\hat{y} - y)^2 \tag{2.17}$$

where y is the true output. During the backward pass, this error is propagated backward to update the weights.

The partial derivative of the error with respect to each weight is computed using the chain rule. For instance, the error gradient with respect to a weight w_{ij} in the second hidden layer is:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_{ij}} \tag{2.18}$$

where a_2 is the activation function output and z_2 is the weighted sum of inputs for the second hidden layer. This gradient is then used to update the weights using a learning rate η:

$$w_{ij}^{\text{new}} = w_{ij} - \eta \cdot \frac{\partial E}{\partial w_{ij}}$$

The same process is applied to the first hidden layer, where the gradient for a weight w_{jk} is:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{jk}}$$

where a_1 and z_1 correspond to the activation and weighted sum in the first hidden layer. By iteratively applying backpropagation through each layer, the network gradually learns to reduce the error, improving its performance in tasks such as classification and regression.

2.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) represent a type of artificial neural network (ANN) characterized by a memory-based architecture specifically designed to process sequential data, often referred to as time series data. RNNs maintain a memory of past inputs through hidden states, enabling subsequent inputs to be influenced by outputs from other nodes within the network. The primary distinction between ANN and RNN architectures lies in their data flow patterns. In ANN architecture, data travels unidirectionally, from the input layer to the output layer, constituting what is known as feed-forward architecture. In contrast, RNN architectures feature recurrent connections that facilitate bidirectional data flow, as depicted in Figure 2.10, a characteristic commonly referred to as feedback[17].

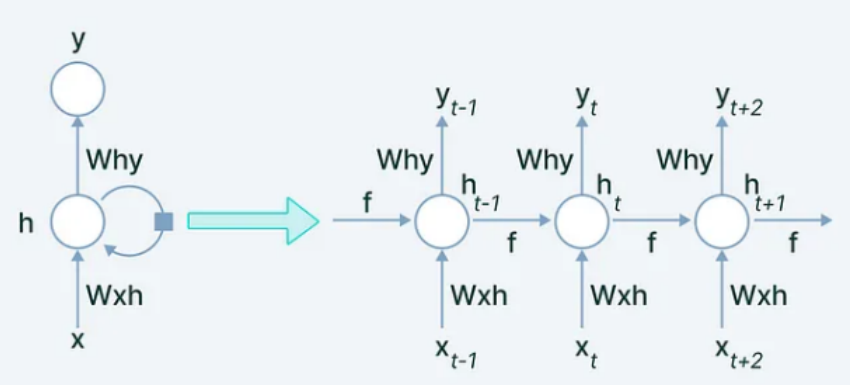

Figure 2.10: Recurrent neural network.

2.4.1 Application domains

Recurrent Neural Networks (RNNs) are widely employed across various domains due to their effectiveness in processing sequential data. In Natural Language Processing (NLP), RNNs play a pivotal role in tasks like language translation, sentiment analysis, and text generation. Their ability to comprehend and generate human-like text contributes to the development of conversational agents and virtual assistants. Moreover, RNNs are instrumental

in speech recognition systems, accurately transcribing spoken language into textual format, and powering voice-activated devices and applications[26].

In the financial sector, RNNs are utilized for time series prediction, aiding in stock market forecasting, asset price prediction, and risk assessment. Additionally, in computer vision, RNNs are applied in image captioning, generating descriptive textual annotations for visual content, thereby enhancing image accessibility and interoperability[26].

Beyond these domains, RNNs find applications in diverse fields such as music generation, DNA sequence analysis, and healthcare for patient monitoring and disease prediction. The versatility of RNNs serves as a catalyst for innovation and advancement across various industries and academic disciplines[26].

2.4.2 Types of architecture

According to inputs and outputs sequences, RNNs have many types of architecture as follows[15, 2]:

- **One-to-many**: In this architecture, a vector serves as the input at the initial time step, generating a sequence of vectors across all subsequent time steps. Image captioning exemplifies this type of architecture, where an image serves as the input, and the output comprises a sequence of words describing the image[52]. This approach enables the network to process complex inputs, such as images, and generate meaningful textual descriptions, capturing the essence of the visual content.

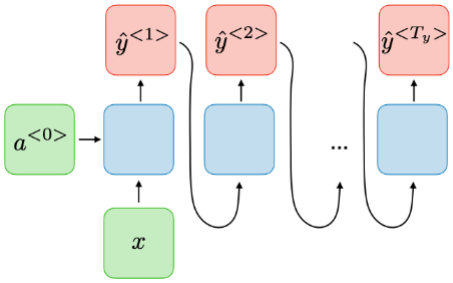

Figure 2.11: One-to-many architecture.

- **Many-to-one**: In this architecture, a sequence of vectors is utilized as the input at each time step, culminating in the generation of a single vector at the final time step. Sentiment analysis exemplifies this

architecture, where the input comprises a sentence represented as a sequence of words, and the output is an expression of sentiment, typically ranging from positive to negative, or rated on a scale from 0 to 5[44]. This framework enables the network to analyze and interpret sequential data, such as text, and produce meaningful insights or classifications based on the overall context of the input sequence.

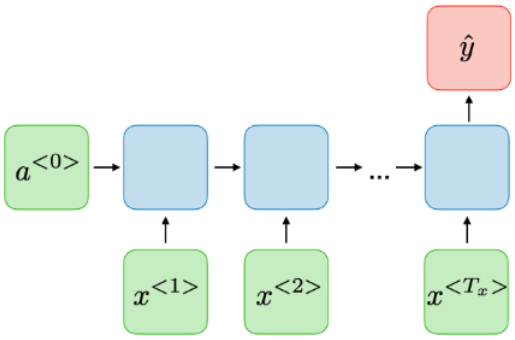

Figure 2.12: Many-to-one architecture.

- **Many-to-many**: In this architecture, there are two types of networks, in the first type for each time step there is a vector as an input and a vector as an output as shown in Figure 2.13 so the input sequence is equal to the output sequence Tx = Ty that means the input and the output have the same length. One example of this type is Part-of-speech tagging, the task is to tag each word in a sentence as verb, noun, adjective, etc.[21].

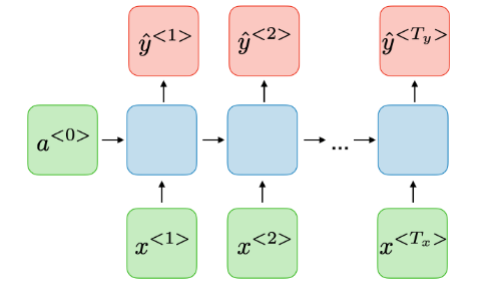

Figure 2.13: Many-to-many architecture.

In the second type, the input is a sequence and the output is a sequence same as the first type, but in this network, the first time steps are fed

by a sequence as input and the next time steps generate a sequence as output as shown in Figure 2.14, in other words, we feed input and generate output form different time steps, in this case, the input length can equal to output length or not. One example of this type is machine translation, firstly we write the complete text in a language and then generate the text in another language[34].

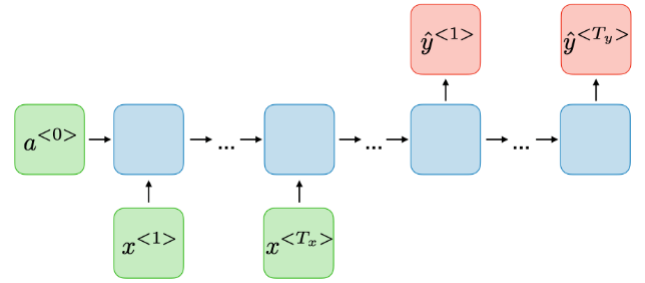

Figure 2.14: Many-to-many architecture.

2.5 Long short-term memory

Long Short-Term Memory (LSTM) networks have a significant advantage over traditional Recurrent Neural Networks (RNNs) due to their gating mechanism. This mechanism enables LSTMs to retain long-term memory, which is particularly beneficial for sequential tasks and natural language processing (NLP). For instance, consider a task where the network is required to generate the next word in a sequence of text. Traditional RNNs often struggle with this task because they typically rely on short-term memory, retaining information from only the most recent few sentences. This limitation can lead to inaccuracies, as the RNN may fail to utilize relevant information from earlier in the text.

In contrast, LSTMs can maintain information over longer periods, which allows them to generate more accurate and contextually appropriate predictions. For example, in network-generated text, when provided with specific information such as a name or an object, an LSTM can recall and correctly incorporate this information even after several intervening sentences. This capability enhances the LSTM's performance in generating text that is coherent and contextually relevant, demonstrating a clear advantage in tasks that require understanding and leveraging long-term dependencies.[22, 14, 16].

LSTM Component can be described deeply as follows:

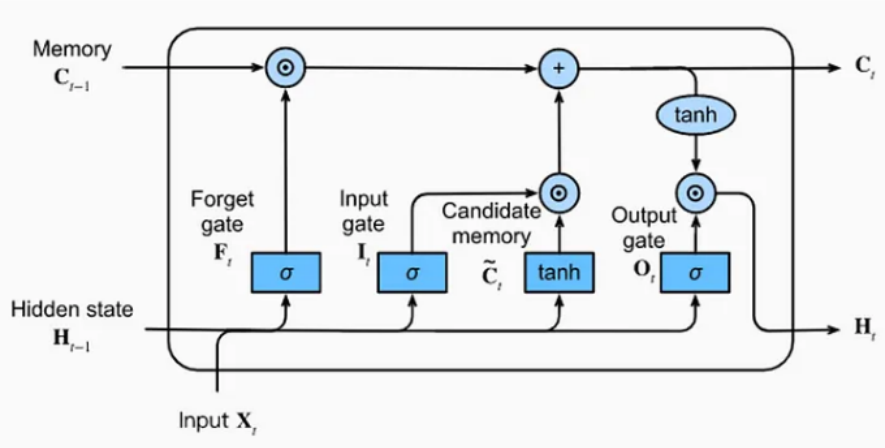

Figure 2.15: LSTM architecture.

- **Forget Gate:** The forget f_t gate in an LSTM network is responsible for determining which information from the previous cell state should be discarded. This gate uses a sigmoid activation function to produce a value between 0 and 1 for each number in the cell state C_{t-1}. A value of 1 indicates "keep this entirely," while a value of 0 indicates "forget this completely." The gate's decision is based on the current input x_t and the previous hidden state h_{t-1}, combined through a weighted sum and a bias term. This allows the LSTM to selectively forget irrelevant information, which is crucial for handling long-term dependencies in the data. The mathematical equation in the forget gate is given by:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.19}$$

- **Input Gate:** The input gate controls the amount of new information added to the cell state C_t. It also uses a sigmoid function to create an input gate activation vector i_t, which decides which values will be updated. Simultaneously, the LSTM computes a vector of candidate values $\tilde{C}_t$ using the hyperbolic tangent (tanh) function. The actual update to the cell state is a combination of these two vectors, allowing the network to selectively add new information based on the current input x_t and the previous hidden state h_{t-1}.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.20}$$

- **Candidate Cell State:** The candidate cell state $\tilde{C}_t$ represents the new information that could be added to the cell state. It is calculated using

the tanh function applied to the weighted sum of the current input x_t and the previous hidden state h_{t-1}. This candidate state is then modulated by the input gate to ensure only the most relevant information is added to the cell state, helping the network to maintain pertinent information while avoiding overloading the memory with unnecessary data.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.21}$$

- **Cell State:** The cell state C_t is the core of the LSTM's memory. It is updated by combining the old cell state C_{t-1}, modulated by the forget gate f_t, with the candidate cell state $\tilde{C}_t$, modulated by the input gate i_t. This dual control mechanism allows the LSTM to preserve important information over long periods while incorporating relevant new information. The cell state acts as a conveyor belt, carrying information across many time steps without significant modifications, thus effectively managing long-term dependencies.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.22}$$

- **Output Gate:** he output gate determines the value of the next hidden state h_t, which is also used as the output of the LSTM unit. The gate uses a sigmoid function to create an output gate activation vectoro_t, which decides which parts of the cell state should be output. The cell state C_t is then passed through a tanh function to push the values to be between -1 and 1, and is multiplied by the output gate's activation vector. This filtered version of the cell state forms the hidden state h_t, which influences both the current output and the subsequent states.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.23}$$

- **Hidden State:** The hidden state h_t is the output of the LSTM cell for the current time step and serves as the input to the next LSTM cell in the sequence. It is computed by taking the tanh of the current cell state C_t and modulating it with the output gate o_t. This state encapsulates both the long-term memory contained in C_t and the short-term dynamics captured by the current input x_t and the previous hidden state h_{t-1}. The hidden state is crucial as it carries forward the relevant information needed for predicting the next steps in the sequence.

$$h_t = o_t * \tanh(C_t) \tag{2.24}$$

2.6 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is a type of recurrent neural network designed to manage and utilize long-term dependencies more efficiently than traditional RNNs. It employs two key mechanisms: the update gate and the reset gate. The update gate controls how much of the previous hidden state is carried forward to the current hidden state, enabling the network to maintain and use information over extended time periods. The reset gate, on the other hand, determines how much of the past information to forget when incorporating new input, allowing the GRU to reset its memory when necessary. By combining these gates, the GRU can adaptively preserve and update its memory, ensuring that relevant information is retained while irrelevant information is discarded. This makes GRUs particularly effective for tasks involving sequential data, such as language modeling and time-series prediction, where capturing long-term dependencies is crucial. Additionally, GRUs are computationally efficient compared to LSTMs, as they use fewer parameters, making them a popular choice in many practical applications[11, 26].

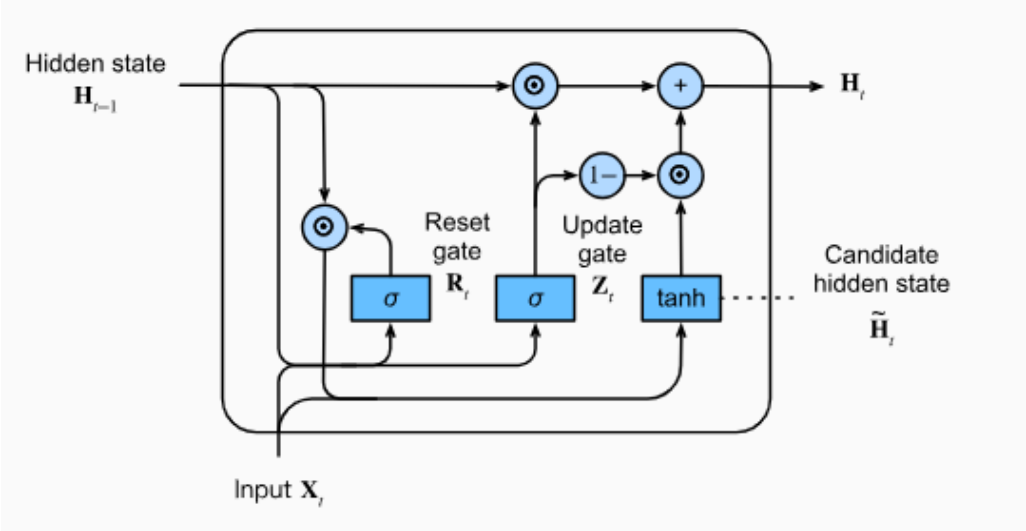

Figure 2.16: GRU architecture.

GRU Components can be described deeply as follows:

- **Update Gate:** The update gate in a GRU controls how much of the previous hidden state h_{t-1} needs to be passed along to the current hidden state h_t. This gate uses a sigmoid function to produce an update gate vector z_t that decides the weight given to the old hidden state versus the candidate hidden state. By combining the information from

the current input x_t and the previous hidden state h_{t-1}, the update gate helps the GRU maintain information over multiple time steps, balancing between preserving past knowledge and incorporating new information.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{2.25}$$

- **Reset Gate:** The reset gate in a GRU determines how much of the past information to forget. It uses a sigmoid activation function to compute the reset gate vector r_t, which decides the degree to which the previous hidden state h_{t-1} influences the candidate hidden state $\tilde{h}_t$. When the reset gate is close to zero, the GRU effectively forgets the past hidden state, focusing only on the current input x_t. This mechanism allows the GRU to reset its memory when processing sequences with long dependencies.

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{2.26}$$

- **Candidate Hidden State:** The candidate hidden state $\tilde{h}_t$ in a GRU represents the new content to be added to the current hidden state. It is calculated using the tanh function applied to a weighted sum of the current input x_t and the element-wise product of the reset gate r_t and the previous hidden state h_{t-1}. This interaction ensures that the influence of the past state is modulated by the reset gate, allowing the network to adaptively decide how much past information is relevant for the current time step.

$$\tilde{h}_t = \tanh(W_h \cdot [r_t * h_{t-1}, x_t] + b_h) \tag{2.27}$$

- **Hidden State:** The hidden state h_t in a GRU is a blend of the previous hidden state h_{t-1} and the candidate hidden state $\tilde{h}_t$. This is controlled by the update gate z_t, which decides the proportion of the old state to retain versus the new candidate state to incorporate. The hidden state is computed by Equation 2.28 This formulation allows the GRU to preserve long-term dependencies by maintaining relevant information over many time steps, while also integrating new information efficiently.

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{2.28}$$

2.7 Differences between LSTM and GRU

Table 2.7 represents the main differences between LSTM and GRU.

	LSTM	**GRU**
Gates	Has three gates (input, forget, output)	Has two gates (update, reset)
Memory Cell	Maintains a separate cell state	Combines the cell state and hidden state into a single hidden state
Complexity	More complex due to the additional gate and separate cell state	Simpler and often faster to compute due to fewer gates and combined state
Performance	Better for tasks requiring learning long-term dependencies.	Comparable performance in many tasks, with fewer parameters and faster training

Table 2.1: Differences between LSTM and GRU.

There are few more differences between LSTM and GRU. First, LSTM can control on the exposure of memory, while GRU exposes the memory, which is not controllable. Then, GRU does not have the output gate like in LSTM. Also, the input gate and forget gate in LSTM are substituted by the update gate, and thus the reset gate is directly applied to the previous hidden state. Since GRU has fewer parameters than LSTM, we could make a preliminary guess that GRU will be faster than LSTM and need less data. However, if the data scale is large, LSTM may lead to a better result.

2.7.1 Advantage and limitation

Advantages

LSTM and GRU networks provide significant advantages in handling sequential data compared to traditional recurrent neural networks (RNNs). Both architectures introduce gating mechanisms that enable the networks to selectively retain or forget information over multiple time steps, facilitating more effective learning and prediction. One notable advantage of LSTM networks is their explicit memory cell and gating structures, which offer greater flexibility in modeling complex temporal relationships and handling longer sequences of data. LSTMs are particularly effective in tasks that require modeling long-range dependencies, such as natural language processing and

speech recognition. On the other hand, GRU networks offer a simpler architecture with fewer parameters compared to LSTMs, resulting in faster training and lower computational overhead. GRUs also tend to perform well in scenarios with limited training data or when computational resources are constrained[17, 14].

limitations

Despite their advantages, LSTM and GRU architectures have certain limitations that should be considered. One common limitation is their susceptibility to overfitting, especially when dealing with small datasets or when the model complexity is high. Additionally, interpreting the inner workings of these gated recurrent units can be challenging, making it difficult to gain insights into the learned representations. This lack of interpretability can hinder the debugging and optimization processes, particularly in complex models. Furthermore, while LSTMs and GRUs offer significant advantages in handling sequential data, they may not always outperform simpler models or alternative architectures in certain tasks. Careful consideration of these limitations is essential when choosing the appropriate architecture for a given task, ensuring that the selected model effectively balances performance and complexity[17, 11].

2.8 Conclusion

This chapter provides a comprehensive analysis of the mathematical and conceptual foundations of machine learning and neural networks. It covers loss functions, gradient descent, and the training and optimization of models. Detailed explanations of neural networks, activation functions, forward and backward propagation, and advanced models like RNNs, LSTMs, and GRUs are included. The chapter establishes a robust groundwork for understanding and exploring advanced AI techniques in subsequent chapters.

Chapter 3

Experiments and Results

3.1 Introduction

In this chapter, we propose a system designed to predict forex prices using LSTM and GRU. We will provide a detailed explanation of our system, including the architecture and methodologies employed. Following this, we will present the results obtained from our predictive models and conduct a thorough comparison with state-of-the-art approaches to assess the efficacy and accuracy of our system.

3.2 Proposed System

The proposed system, shown in Figure 3.1, aims to accurately predict financial time series data using LSTM and GRU network. The process initiates with the collection of a comprehensive dataset from the MetaTrader database, which offers historical financial data essential for robust model training. This dataset is divided into three subsets: 70% for training, 15% for validation, and 15% for testing, ensuring a balanced framework for model evaluation and fine-tuning. To effectively capture temporal dependencies, the dataset is segmented into overlapping time steps, creating sliding windows of sequential data optimized for LSTM and GRU processing. These sequential data windows are subsequently fed into the model, which is specifically designed to learn and predict complex patterns over time. This approach leverages the LSTM and GRU's ability to model long-term dependencies, thereby enhancing predictive performance and reliability[22]. This methodological framework ensures that the model is trained on a diverse array of data points, validated to mitigate overfitting, and tested to assess its generalization capabilities on new, unseen data. The comprehensive preprocessing and

structured data handling establish a solid foundation for the LSTM model, resulting in accurate and dependable financial time series predictions[14].

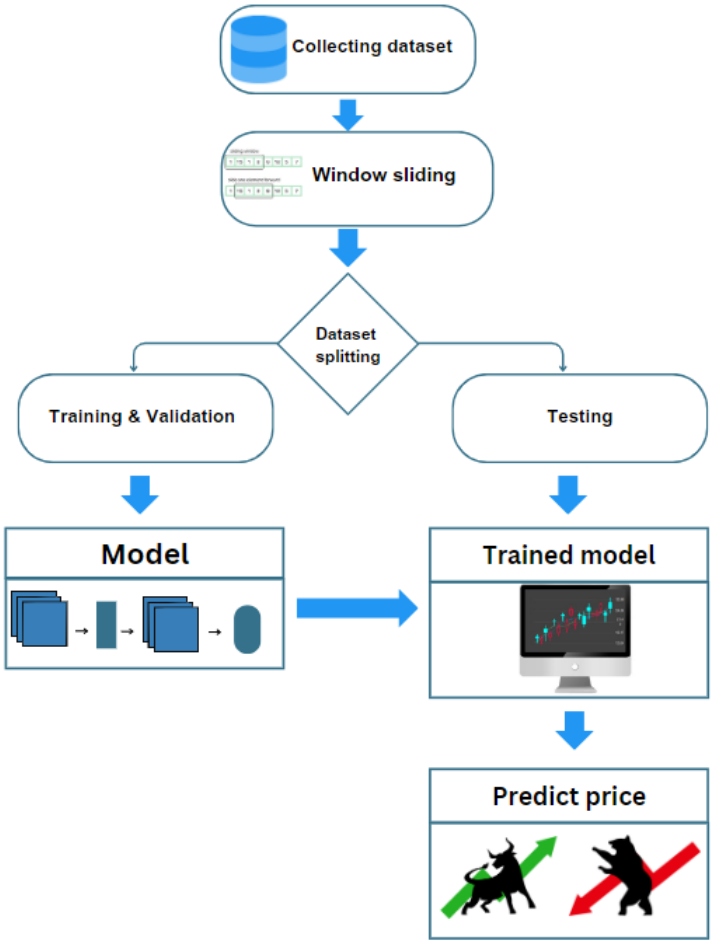

Figure 3.1: Proposed System.

The proposed architecture employs a deep learning architecture built using the Sequential API of TensorFlow's Keras. The model comprises three RNN (LSTM or GRU) layers, designed to capture and learn from the temporal dependencies in sequential data. The first RNN layer consists of 180 units, the second RNN layer with 90 units, and the third RNN layer with 20 units, providing a robust foundation for extracting complex patterns from the input sequences. To mitigate overfitting and enhance generalization, a Dropout layer with a dropout rate of 0.2 is introduced between the layers[28]. Finally, a Dense layer with a single unit is used to produce the output, mak-

ing the model suitable for regression tasks. This architecture balances depth and complexity, ensuring the model is both powerful and efficient for various time-series prediction problems.

3.3 Dataset Description

The dataset used in this study comprises detailed financial metrics and technical indicators essential for predictive modeling in financial time series analysis. It includes the fundamental price data: open, high, low, and close prices, which provide comprehensive insights into daily market fluctuations. Additionally, the dataset incorporates two critical technical indicators: the 30-day Exponential Moving Average (EMA) and the Relative Strength Index (RSI). The EMA is a widely used tool that gives more weight to recent prices, making it particularly useful for identifying short-term trends and momentum. The RSI, on the other hand, is a momentum oscillator that measures the speed and change of price movements, offering valuable information about overbought or oversold market conditions[37]. By integrating these technical indicators with the core price data, the dataset provides a robust foundation for building predictive models that can capture both trend-following and mean-reverting behaviors in financial markets. This comprehensive dataset enables the application of advanced machine learning techniques, facilitating the development of models with enhanced predictive accuracy and reliability[46].

To calculate the 30-day Exponential Moving Average (EMA) and the Relative Strength Index (RSI), specific formulas and steps are employed to ensure accurate representation of market trends and momentum. The EMA is calculated by applying more weight to recent prices, using the formula in Equation 3.1, where P_t is the current price, α is the smoothing factor given by $\frac{2}{n+1}$ and n is the number of days, and EMA_{t-1} is the EMA of the previous day. This method ensures the EMA responds more swiftly to recent price changes compared to the Simple Moving Average.

$$EMA_t = (P_t \times \alpha) + (EMA_{t-1} \times (1 - \alpha)) \tag{3.1}$$

The RSI is calculated to measure the magnitude of recent price changes to evaluate overbought or oversold conditions. It involves two main steps: calculating the average gains and losses over a specified period (usually 14 days), and then applying these averages in the RSI formula in Equation 3.2, where RS is (Relative Strength) is the ratio of average gain to average loss. The average gain is calculated as the sum of all gains over the period divided

Date	Open	High	Low	Close	EMA50	RSI
1999.01.06 20:00	1.16	1.1636	1.1589	1.1623	1.173195	21.8845
1999.01.07 00:00	1.1623	1.1641	1.1612	1.1634	1.173238	24.59941
1999.01.07 04:00	1.1635	1.1648	1.1635	1.1641	1.173266	26.35354
1999.01.07 08:00	1.1657	1.1679	1.1651	1.1657	1.173329	30.34251
1999.01.07 12:00	1.1656	1.1698	1.1627	1.1694	1.173474	38.62173
1999.01.07 16:00	1.169	1.1735	1.1656	1.1713	1.173548	42.40726
1999.01.07 20:00	1.1712	1.1723	1.1701	1.1712	1.173544	42.25953
1999.01.08 00:00	1.1713	1.172	1.169	1.1713	1.173548	42.47534
1999.01.08 04:00	1.1711	1.1715	1.1682	1.1685	1.173438	38.17318
1999.01.08 08:00	1.1686	1.1693	1.1651	1.1666	1.173364	35.54244
1999.01.08 12:00	1.1663	1.1673	1.1575	1.1587	1.173054	27.16094
1999.01.08 16:00	1.1586	1.1619	1.1533	1.1572	1.172995	25.9115
1999.01.08 20:00	1.1595	1.16	1.1574	1.1585	1.173046	28.96151
1999.01.11 00:00	1.1575	1.1582	1.1561	1.1579	1.173023	28.38079
1999.01.11 04:00	1.1571	1.159	1.1569	1.1586	1.17305	30.14075
1999.01.11 08:00	1.1585	1.161	1.1569	1.1588	1.173058	30.665
1999.01.11 12:00	1.159	1.1592	1.1513	1.1517	1.17278	23.82858
1999.01.11 16:00	1.1514	1.1557	1.1495	1.1502	1.172721	22.67827
1999.01.11 20:00	1.1492	1.1508	1.1487	1.1503	1.172725	22.94533
1999.01.12 00:00	1.1504	1.1518	1.1447	1.1479	1.172631	21.06488

Figure 3.2: Dataset used in this study.

by the number of days, and similarly, the average loss is the sum of all losses divided by the number of days. These calculations yield an oscillator that fluctuates between 0 and 100, providing insights into the momentum and potential reversal points in the market.

$$RSI = 100 - \frac{100}{1 + RS} \tag{3.2}$$

Figure 3.3 shows these indicators on the EURUSD chart.

3.4 Evaluation Metrics

Evaluation metrics in machine learning are quantitative measures used to assess the performance and effectiveness of a predictive model. These metrics provide insights into how well a model's predictions match the actual outcomes, guiding improvements in model development and selection. Different metrics capture various aspects of performance, such as accuracy, error rates, and the ability to generalize to new data, making them essential tools for validating and comparing models in diverse applications[7].

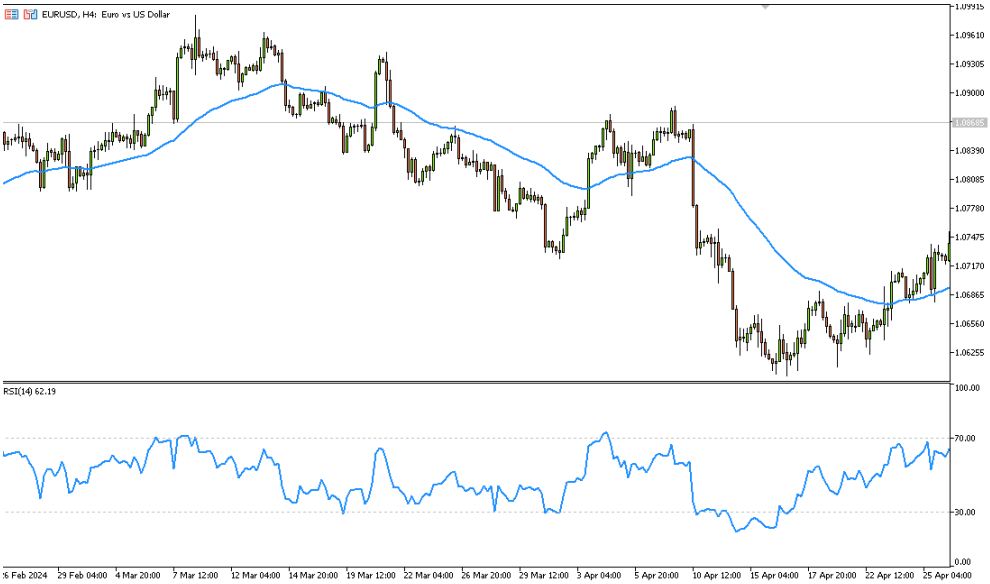

Figure 3.3: EURUSD chart including EMA and RSI indicators.

3.4.1 Mean Squared Error

Mean Squared Error (MSE) is a common evaluation metric used in regression tasks within machine learning to measure the average squared difference between the observed actual values and the predicted values by a model. The MSE provides a quantifiable measure of the accuracy of predictions, where lower values indicate better model performance[8]. Mathematically, MSE is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.3}$$

3.4.2 Mean Absolute Error

Mean Absolute Error (MAE) is another widely used evaluation metric for regression models that assesses the average magnitude of the errors in a set of predictions, without considering their direction. MAE measures the average absolute difference between the actual values and the predicted values, providing a straightforward interpretation of prediction accuracy[8]. Mathematically, MAE is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.4}$$

3.4.3 Root Mean Square Error

The Root Mean Square Error (RMSE) is a widely used metric for evaluating the accuracy of a predictive model, particularly in regression tasks. It measures the average magnitude of the errors between predicted values and actual values, providing a sense of how well the model performs. The RMSE is calculated by taking the square root of the average of the squared differences between predicted and actual values[8]. Mathematically, RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.5}$$

3.4.4 Mean Absolute Percentage Error

The Mean Absolute Percentage Error (MAPE) is a common metric for evaluating the accuracy of predictive models, especially in time series forecasting. It measures the average absolute percentage difference between the predicted and actual values, providing a normalized measure of prediction accuracy. MAPE is particularly useful because it expresses error as a percentage, making it easier to interpret and compare across different datasets[8]. Mathematically, MAPE is defined as:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{3.6}$$

3.5 Results

This section aims to present the obtained results in the form of price charts. By visualizing the predicted versus actual prices for each of the three currency pairs—EURUSD, GBPUSD, and USDCHF—we can effectively illustrate the performance of the LSTM and GRU models. These charts will provide a clear, graphical representation of how closely the predicted values follow the actual market trends, allowing for an intuitive comparison of model accuracy. Additionally, the price charts will help in identifying any discrepancies or patterns in the predictions, facilitating a deeper understanding of the models' predictive capabilities and areas for potential improvement.

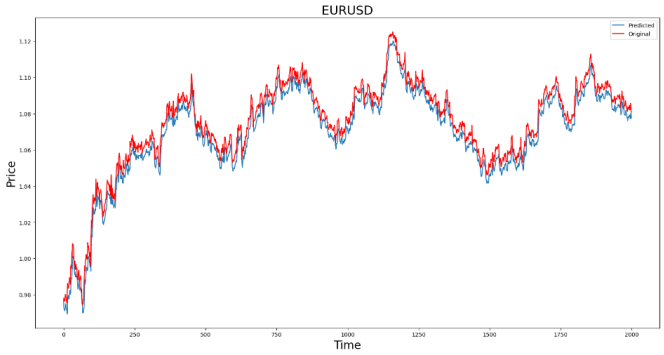

Figure 3.4: Predicted and actual price chart for EURUSD using LSTM model.

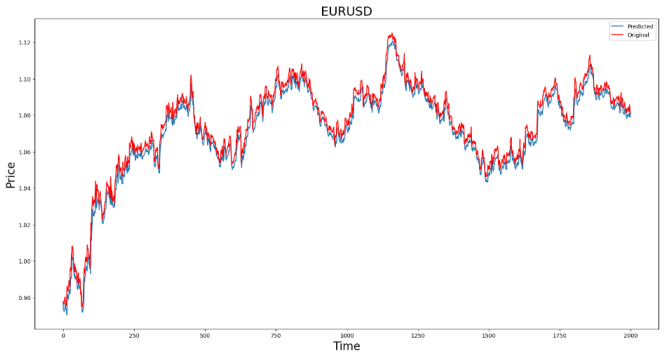

Figure 3.5: Predicted and actual price chart for EURUSD using GRU model.

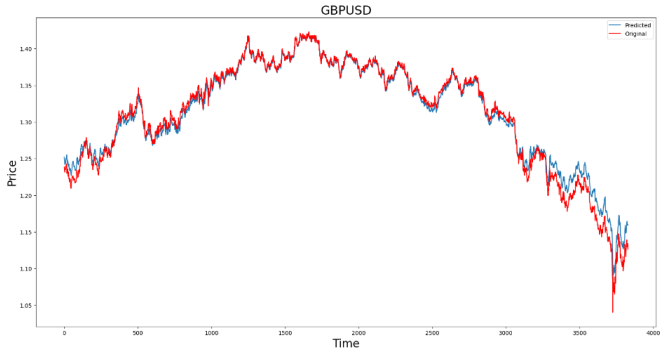

Figure 3.6: Predicted and actual price chart for GBPUSD using LSTM model.

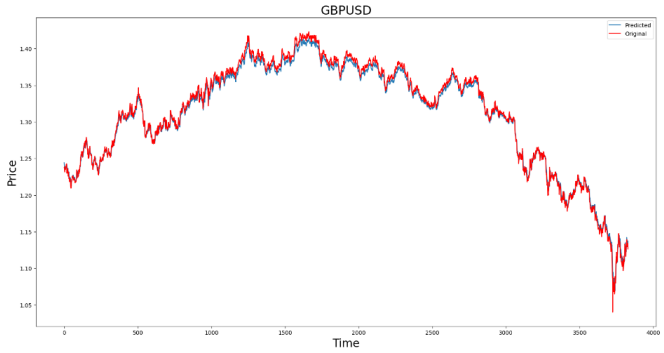

Figure 3.7: Predicted and actual price chart for GBPUSD using GRU model.

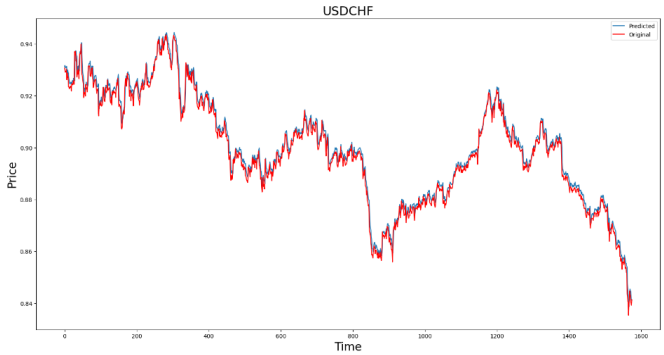

Figure 3.8: Predicted and actual price chart for USDCHF using LSTM model.

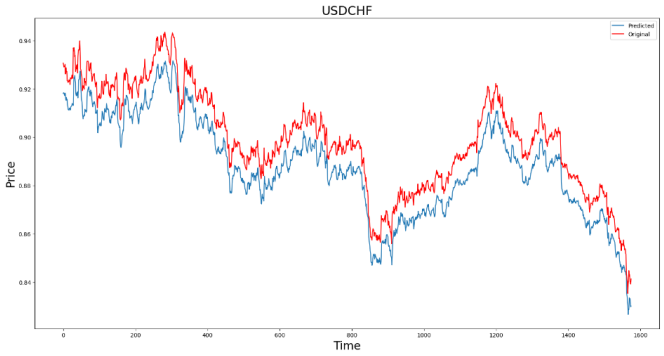

Figure 3.9: Predicted and actual price chart for USDCHF using GRU model.

3.6 Performance comparison

In evaluating the efficacy of LSTM and GRU models for financial market prediction, it is essential to consider the robustness of the methodology and

the reliability of the findings results. Two fundamental evaluation metrics, MAPE and RMSE, were appropriately employed to quantify the accuracy of the forecasts, aligning with established practices in financial forecasting research. The results, carefully reported in Tables 3.6 and 3.6 for LSTM and GRU, respectively, provide an in-depth overview of the model's predictive ability across a wide range of experimental settings and time periods. MAPE and RMSE are highly recognized in financial research for their usefulness in evaluating predictive models.

Target	**MAPE**	**RMSE**
EURUSD	0.365	0.006
GBPUSD	0.2981	0.0065
USDCHF	0.323	0.0068

Table 3.1: The results using LSTM model.

Target	**MAPE**	**RMSE**
EURUSD	0.4058	0.0062
GBPUSD	0.344	0.0074
USDCHF	0.4621	0.0085

Table 3.2: The results using GRU model.

3.7 Comparison with state of the art

This section aims to compare our study with current state-of-the-art methodologies and results in the field. By performing an accurate comparison, we can evaluate our study's novelty, efficacy, and contributions to the larger landscape of existing research. We begin by reviewing the most recent and relevant literature in the field, focusing on studies that address similar research questions or employ comparable methodologies. This allows us to establish a baseline understanding of the prevailing approaches and findings in our domain of interest. Next, we systematically evaluate the key differences and similarities between our study and the state-of-the-art. This includes a comparative analysis of the research objectives, methodologies, datasets used, and experimental results. We also consider any advancements or innovations introduced in our study that distinguish it from prior works. Furthermore, we critically examine the strengths and limitations of both our study and the state-of-the-art methodologies. This includes an assessment of

the robustness, scalability, and generalizability of the results, as well as any potential biases or constraints inherent in the methodologies employed. By synthesizing these comparisons, we aim to highlight the unique contributions and advancements offered by our study, as well as identify areas for further research and improvement. Ultimately, this comparative analysis provides valuable insights into the current state of the field and the positioning of our study within it. The results of reviewed research, are carefully reported in Table 3.7.

Study	**Target**	**MSE**	**RMSE**	**MAE**	**MAPE**
[3]	Tesla	0.0033	0.0581	0.0413	/
[43]	TCS	0.0025	0.0497	0.0397	0.1715
[45]	COL	2.25	1.5	/	1.96
[42]	Carbon	3.1695	/	1.0504	/
[30]	Gold	/	18.0720	12.0819	0.7879
[32]	600000.SH.	/	0.7019	/	/
[6]	S&P 500	/	49.8362	/	1.0269
[48]	CSI 300	0.0091	/	0.0693	1.1216
[10]	Bitcoin	/	330.26	/	0.0357
[23]	STC	0.72	17.07	0.28	0.721
[40]	CSI300	291.37	17.07	11.96	/
Proposed model	**Forex**	/	**0.0065**	/	**0.298**

Table 3.3: The results of the state of the art.

3.8 Discussion

In this study, the performance of two deep learning models, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), was evaluated for predicting financial time series data for three major currency pairs: EURUSD, GBPUSD, and USDCHF. The results demonstrated that the LSTM model consistently outperformed the GRU model in both Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). Specifically, the LSTM model achieved RMSE values of 0.365 for EURUSD, 0.2981 for GBPUSD, and 0.323 for USDCHF, while the GRU model showed higher RMSE values of 0.4058, 0.344, and 0.4621, respectively. This suggests that the LSTM model is more effective at capturing long-term dependencies in financial data, a critical factor for accurate predictions.

When comparing these results with state-of-the-art studies, such as an RMSE of 0.0581 for Tesla stock prices [3] and 0.0497 for TCS stock prices

[43], our models exhibit competitive performance, especially given the complexities inherent in forex market predictions. Additionally, the low MAPE values for both models, particularly the LSTM's 0.006 for EURUSD, indicate strong predictive reliability. These findings are consistent with existing literature, which highlights the effectiveness of LSTM networks in time series forecasting due to their ability to model temporal dependencies effectively

However, it is crucial to note that direct comparisons with state-of-the-art studies are challenging due to differences in datasets, target variables, and evaluation metrics. Despite these nuances, our study makes a significant contribution by demonstrating the practical applicability of LSTM and GRU models in financial forecasting, offering a methodological framework that can be further optimized and validated across different datasets. Future research should focus on enhancing these models' performance through advanced optimization techniques and testing their robustness across a broader range of financial instruments and market conditions.

Overall, this research provides valuable insights into the application of deep learning techniques in financial time series analysis, confirming the LSTM model's superior capability in predicting complex financial data and paving the way for further advancements in this field.

3.9 Conclusion

In conclusion, our study demonstrates that Long Short-Term Memory (LSTM) networks exhibit superior accuracy compared to Gated Recurrent Units (GRUs) in the context of predicting financial time series data. The LSTM model consistently outperforms the GRU model across various metrics, indicating its enhanced capability to capture and leverage long-term dependencies in sequential data. Moreover, when benchmarked against the state-of-the-art, our results are competitive and within acceptable ranges, validating the effectiveness of our approach. These findings reinforce the utility of LSTM networks in applications requiring precise sequential data modeling and highlight the robustness of our proposed methodology.

General conclusion

In conclusion, this dissertation has undertaken a comprehensive investigation into the efficacy of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models in forecasting stock market trends. The research endeavor has been underpinned by a rigorous examination of existing literature in stock market prediction, with meticulous attention to fundamental and technical analysis approaches. Moreover, the significance of datasets and pre-processing techniques has been duly acknowledged, drawing upon reputable sources in the field.

The methodological approach adopted in this dissertation adheres to high academic standards, providing a thorough exposition of machine learning concepts and neural network architectures. The experimental design upholds principles of credibility and validity, ensuring robustness in dataset descriptions and evaluation metrics.

The results and discussion section rigorously analyzes empirical findings, offering insights into the performance of LSTM and GRU models in stock market prediction. The comparative analysis conducted herein underscores the scholarly integrity of the research, providing a nuanced interpretation of the implications of the results within the broader landscape of stock market prediction research. Additionally, a comparison with state-of-the-art methodologies serves to reinforce the credibility and significance of the dissertation's contributions to the field.

In summation, this dissertation represents a notable contribution to the understanding of machine learning techniques in stock market prediction, upholding scholarly rigor and integrity throughout. By evaluating LSTM and GRU models and comparing them with established methodologies, this research endeavor provides valuable insights that can inform decision-making processes in finance and investment.

Bibliography

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Shervine Amidi. Cs230: Cheatsheet - recurrent neural networks. Stanford University.

[3] Yasmeen Ansari, Sadaf Yasmin, Sheneela Naz, Hira Zaffar, Zeeshan Ali, Jihoon Moon, and Seungmin Rho. A deep reinforcement learning-based decision support system for automated stock market trading. *IEEE Access*, 10:127469–127501, 2022.

[4] Matin N Ashtiani and Bijan Raahemi. News-based intelligent prediction of financial markets using text mining and machine learning: A systematic literature review. *Expert Systems with Applications*, 2023.

[5] C Sarai R Avila. Tweet influence on market trends: Analyzing the impact of social media sentiment on biotech stocks. *arXiv preprint arXiv:2402.03353*, 2024.

[6] Hum Nath Bhandari, Binod Rimal, Nawa Raj Pokhrel, Ramchandra Rimal, Keshab R Dahal, and Rajendra KC Khatri. Predicting stock market index using lstm. *Machine Learning with Applications*, 9:100320, 2022.

[7] Christopher M Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:1122–1128, 2006.

[8] Alexei Botchkarev. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*, 2018.

[9] William Brock, Josef Lakonishok, and Blake LeBaron. Simple technical trading rules and the stochastic properties of stock returns. *The Journal of finance*, 47(5):1731–1764, 1992.

[10] Junwei Chen. Analysis of bitcoin price prediction using machine learning. *Journal of Risk and Financial Management*, 16(1):51, 2023.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[12] Robert D. Edwards, John Magee, and W. H. C. Bassetti. *Technical Analysis of Stock Trends*. CRC Press, 11 edition, 2018.

[13] Eugene F Fama. Efficient capital markets. *Journal of finance*, 25(2):383–417, 1970.

[14] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research*, 270(2):654–669, 2018.

[15] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[16] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[17] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016.

[18] S. Harder. *The Efficient Market Hypothesis and Its Application to Stock Markets*. GRIN Verlag, 2010.

[19] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[20] Bruno Miranda Henrique, Vinicius Amorim Sobreiro, and Herbert Kimura. Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, 124:226–251, 2019.

[21] Sintayehu Hirpassa and G.S. Lehal. Improving part-of-speech tagging in amharic language using deep neural network. *Heliyon*, 9(7), 2023.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] Mutasem Jarrah and Morched Derbali. Predicting saudi stock market index by using multivariate time series based on deep learning. *Applied Sciences*, 13(14):8356, 2023.

[24] D.J. Jüttner. *The Random Walk Hypothesis and Stock Market Efficiency*. Research paper. School of Economic and Financial Studies, Macquarie University, 1974.

[25] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[26] S. Kostadinov. *Recurrent Neural Networks with Python Quick Start Guide: Sequential learning and language modeling with TensorFlow*. Packt Publishing, 2018.

[27] M. Krantz. *Fundamental Analysis For Dummies*. Wiley, 2023.

[28] Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. In *International conference on machine learning*, pages 2052–2061. PMLR, 2017.

[29] Yanhui Liang, Yu Lin, and Qin Lu. Forecasting gold price using a novel hybrid model with iceemdan and lstm-cnn-cbam. *Expert Systems with Applications*, 206, 2022.

[30] Yanhui Liang, Yu Lin, and Qin Lu. Forecasting gold price using a novel hybrid model with iceemdan and lstm-cnn-cbam. *Expert Systems with Applications*, 206:117847, 2022.

[31] Ying-Lei Lin, Chi-Ju Lai, and Ping-Feng Pai. Using deep learning techniques in forecasting stock markets by hybrid data with multilingual sentiment analysis. *Electronics*, 2022.

[32] Keyan Liu, Jianan Zhou, and Dayong Dong. Improving stock price prediction using the long short-term memory model combined with online social networks. *Journal of Behavioral and Experimental Finance*, 30:100507, 2021.

[33] Andrew Lo. *Adaptive markets: Financial evolution at the speed of thought*. Princeton University Press, 2017.

[34] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[35] Burton G Malkiel. The efficient market hypothesis and its critics. *Journal of economic perspectives*, 17(1):59–82, 2003.

[36] Mahdi Massahi and Masoud Mahootchi. A deep q-learning based algorithmic trading system for commodity futures markets. *Expert Systems with Applications*, 237, 2024.

[37] John J Murphy. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.

[38] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[39] Cheol-Ho Park and Scott Irwin. The profitability of technical analysis: A review. Research Report 2004-04, AgMAS Project, 2004.

[40] Chenyang Qi, Jiaying Ren, and Jin Su. Gru neural network based on ceemdan–wavelet for stock price prediction. *Applied Sciences*, 13(12):7104, 2023.

[41] Scott Richardson, İrem Tuna, and Peter Wysocki. Accounting anomalies and fundamental analysis: A review of recent research advances. *Journal of Accounting and Economics*, 50(2), 2010.

[42] Hanxiao Shi, Anlei Wei, Xiaozhen Xu, Yaqi Zhu, Hao Hu, and Songjun Tang. A cnn-lstm based deep learning model with high accuracy and robustness for carbon price forecasting: A case of shenzhen's carbon market in china. *Journal of Environmental Management*, 352:120131, 2024.

[43] Pushpendra Singh Sisodia, Anish Gupta, Yogesh Kumar, and Gaurav Kumar Ameta. Stock market analysis and prediction for nifty50 using lstm deep learning approach. In *2022 2nd international conference on innovative practices in technology and management (ICIPTM)*, volume 2, pages 156–161. IEEE, 2022.

[44] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015.

[45] Yassine Touzani and Khadija Douzi. An lstm and gru based trading strategy adapted to the moroccan market. *Journal of big Data*, 8(1):126, 2021.

[46] Ruey S Tsay. *Analysis of financial time series*. John wiley & sons, 2005.

[47] Chaojie Wang, Yuanyuan Chen, Shuqi Zhang, and Qiuhui Zhang. Stock market index prediction using deep transformer model. *Expert Systems with Applications*, 208, 2022.

[48] Chaojie Wang, Yuanyuan Chen, Shuqi Zhang, and Qiuhui Zhang. Stock market index prediction using deep transformer model. *Expert Systems with Applications*, 208:118128, 2022.

[49] Meiqi Wang, Siyuan Lu, Danyang Zhu, Jun Lin, and Zhongfeng Wang. A high-speed and low-complexity architecture for softmax function in deep learning. In *2018 IEEE asia pacific conference on circuits and systems (APCCAS)*, pages 223–226. IEEE, 2018.

[50] Anjar Wanto, Agus Perdana Windarto, Dedy Hartama, and Iin Parlina. Use of binary sigmoid function and linear identity in artificial neural networks for forecasting population density. *IJISTECH (International Journal of Information System and Technology)*, 1(1):43–54, 2017.

[51] Stephen J Wright. Numerical optimization, 2006.

[52] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[53] Yudong Zhang and Lenan Wu. Stock market prediction of s&p 500 via combination of improved bco approach and bp neural network. *Expert systems with applications*, 36(5):8849–8854, 2009.