Modeling Egyptian currency exchange rates using LSTM networks

Omar A. Morshdy **(211001749)**

Ahmed K. Ahmed **(211000202)**

Nadeen M. Farid **(2020002561)**

Mohamed G. ElGemeie **(202000206)**

Mohamed H. Abdrabou **(211001466)**

Amira A. ElSharaby **(202000674)**

Omar M. Shehata (**222000109)**

MATH-203i

Dr. Amira Mofreh

March 3, 2023

Abstract

Unstable and quick fluctuations in the Egyptian pound exchange rates have been at an all-time high in the past year, and modeling techniques are needed to predict future fluctuations. Standard time series forecasting methods like ARIMA, and seasonal methods have been deemed unserviceable due to the sudden changes of the inflation, and papers reviewing Egyptian pound exchange rates using ANN's, but no recent papers used LSTM networks. In this paper we trained an LSTM network with multiple exchange rates retrieved from Google and fine-tuning the hyperparameters, to predict the next day's USD-EGP exchange rate given the past rates and achieved a score on mean absolute error (MAE) of 0.01012 and 0.01591 on root mean squared error (RMSE) on the testing data. Further research needs to be implemented on selecting presentational training data from a long time-series dataset to improve LSTM model's anticipation of volatile changes.

Key words: Time-series, LSTM model, RNN, Egyptian currency.

Table of Contents

Abstract *2*

Introduction **6**

Literature review *7*

1.1 Egyptian currency modeling **7**

1.2 Maximizing LSTM Model **9**

1.3 Disadvantages of the LSTM Model **10**

1.4 LSTM Model as partial differential equations **11**

1.5 Conclusion **12**

Methodology *13*

2.1 Data Acquisition **13**

2.1.1 Date Cleaning and Remarks 13

2.2 LSTM Network Training **14**

2.2.1 Network architecture 15

2.2.1 Backpropagation 16

2.2.1.1 Terminology 16

2.2.1.2 Minimizing the cost function. 17

2.2.5 LSTM Cell equations 19

2.3 Testing and Validation **21**

Results *22*

Conclusion *23*

References *24*

Table of Figures

FIG. 1 THREE CURRENCIES PLOTTED OVER TIME [1] 13

FIG. 2 TRAINING, TESTING SPLIT [2] 14

FIG. 3 LSTM MODEL ARCHITECTURE [3] 15

FIG. 4 THE MODEL'S CODE WRITTEN IN PYTHON [4] 15

FIG. 5 LSTM CELL UNIT [5] 19

FIG. 6. USD-EGP EXCHANGE RATE COMPARED TO MODEL'S PREDICTED VALUES. [6] 22

Table of Equations

$x_i^{(l)}, i \in 1 \dots n^{(l)}, l \in 1 \dots 4$ 16

$a_i^{(l)}, i \in 1 \dots n^{(l)}, l \in 1 \dots 4$ 16

$a_i^{(l)} = x_i^{(l)}, l = 1$ 16

$z_i^{(l)} = \left(\sum_{g=1}^{n^{(l-1)}} w_{ig}^{(l)} a_g^{(l-1)}\right) + b^{(l)}, i \in 1 \dots n^{(l)}, g \in 1 \dots n^{(l-1)}, l > 1$ 16

$a_i^{(l)} = f(\ \ {}_i^{(l)})$ 17

$a^{(4)} = \sum_{i=1}^{n} a_i^{(3)} w_i^{(4)}$ 17

$c = |a^{(4)} - y|$ 17

$\frac{\partial c}{\partial w^{(l)}} = \frac{\partial z^{(l)}}{\partial w^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial c}{\partial a^{(l)}}$ 17

$\frac{\partial c}{\partial a^{(l)}} = \frac{a^{(l)} - y}{|a^{(l)} - y|}$ 18

$\frac{\partial a^{(l)}}{\partial z^{(l)}} = \acute{f}(z^{(l)})$ 18

$\frac{\partial z^{(l)}}{\partial w^{(l)}} = a^{(l-1)}$ 18

$\frac{\partial c}{\partial w^{(l)}} = a^{(l-1)} \acute{f}(z^{(l)}) \frac{a^{(l)} - y}{|a^{(l)} - y|}$ 18

$w\_new^{(l)} = w^{(l)} - \alpha \frac{\partial c}{\partial w^{(l)}}$ 18

$\mathrm{F_T} = \Sigma(\mathrm{X_T U^F} + \mathrm{H_{T-1} W^F} + \mathrm{B^F})$ 19

$\mathrm{I_T} = \Sigma(\mathrm{X_T U^I} + \mathrm{H_{T-1} W^I} + \mathrm{B^I})$ (15) 20

$\tilde{C}_t = tanh(x_t U^c + h_{t-1} W^c + b^c)$ 20

$\mathrm{C_T} = \mathrm{F_T} * \mathrm{C_{T-1}} + \mathrm{I_T} * \tilde{\mathrm{C}}_\mathrm{T}$ (17) 20

$\mathrm{O_T} = \Sigma(\mathrm{X_T U^O} + \mathrm{H_{T-1} W^O} + \mathrm{B^O})$ 20

$\mathrm{H_T} = tanh(C_t) * \mathrm{O_T}$ 21

Introduction

Financial plans and business predictions are crucial for the investment field. Nevertheless, because of several tangible and intangible causes, very erratic movements in stock markets are seen, making prediction a difficult task. Many stacks are at risk as a result of these unstable economic and noneconomic forces, which also include inflation, erratic exchange rates, and shifting political conditions. This necessitates effective stock price prediction tools so that decision-makers may build the foundation for successful and informed investing decisions. Because traditional statistical methods frequently fail to produce accurate predictions when involving multiple nonlinear characteristics, stock price prediction has recently been a significant research area where researchers have exploited the alluring benefits of machine learning methods. Moreover, these techniques are restricted to time-series data of the stock market values. Potentially unpredictable swings in stock prices exhibit unpredictably and are difficult to forecast. Support vector machines (SVM), artificial neural networks (ANN), and other techniques have been used to construct stock price prediction models in the machine learning paradigm. Moreover, deep learning, the most recent development in machine learning, has changed predictive modeling by making it possible for trading algorithms to anticipate stock values with greater accuracy. This is because deep learning techniques automatically infer high-level characteristics in business applications rather than substantially relying on extensive historical data pertaining to economic and behavioral assumptions. The purpose of this study is to investigate the Egyptian currency to US dollar exchange rate mathematical model using the long-short term memory (LSTM) model. This would enable us to comprehend how such massive amounts of currency exchange rates work.

Literature review

1.1 Egyptian currency modeling

As shown in [9] which presents a novel approach to predicting exchange rates using artificial neural networks (ANNs). The authors explain that exchange rate prediction is a complex challenge that has long been a topic of interest in international finance, as it is influenced by a wide range of factors and forces. The authors introduce ANNs as a non-linear modeling technique that can overcome the limitations of linear models. ANNs are a type of machine learning algorithm that is inspired by the structure and function of the human brain. They consist of layers of interconnected nodes that perform mathematical operations on input data to produce output predictions. In their study, the authors trained an ANN to predict the exchange rate between the Egyptian pound and the US dollar using fundamental economic variables such as inflation, interest rates, and trade balances. The authors explain the process of training the ANN by providing input data and corresponding output data and adjusting the weights and biases of the nodes in the network to minimize the error between the predicted and actual exchange rates. Results show that the ANN model can predict the exchange rate with a mean absolute percentage error (MAPE) of 2.8%. This suggests that the model is accurate by 97.2% in forecasting exchange rates based on macroeconomic variables, furthermore, discusses the implications of their study for international finance and the potential applications of ANNs in other areas of finance.

Overall, the paper provides a valuable contribution to the field of international finance by demonstrating the effectiveness of ANNs in predicting exchange rates. The study highlights the potential of non-linear modeling techniques in financial forecasting and suggests that ANNs may be a useful tool for financial analysts and policymakers. The authors note that further research is needed to explore the potential of ANNs in other areas of finance and to develop more

sophisticated models that can capture the complex dynamics of financial markets, which is the reason behind this paper in making predictions using the more sophisticated LSTM model.

Similarly, a comprehensive analysis of the exchange rate dynamics in Egypt provided by [2] focuses on estimating and forecasting the volatility of the exchange rate using various modeling techniques such as ARCH models and State Space models. The authors begin by highlighting the importance of exchange rate volatility in emerging markets, such as Egypt, and the challenges associated with modeling and predicting it. They explain that exchange rate volatility can have significant economic and social implications, including affecting trade, investment, and inflation. The paper presents an empirical analysis of daily exchange rate data spanning from January 2003 to June 2013, and uses various modeling techniques, including ARCH models and State Space models such as Stochastic Volatility models and Time-Varying Parameter models, to estimate and forecast the volatility of the exchange rate. They find that the exchange rate returns in Egypt suffer from the volatility clustering phenomenon and that there exists a time-varying variance in the exchange rate series that must be appropriately dealt with while modeling nominal exchange rates. They also examined the relationship between the stock market and exchange rate market volatilities in Egypt and found that there is a risk mismatch between the two markets and recommend further research to identify other exogenous variables that can explain the volatility in the exchange rate returns in Egypt.

Overall, the paper provides valuable insights into the exchange rate dynamics in Egypt and highlights the importance of considering observed and unobserved volatility in modeling exchange rates. The study demonstrates the effectiveness of various modeling techniques in predicting exchange rate volatility and provides a basis for future research in this area.

1.2 Maximizing LSTM Model

Long short-term memory (LSTM) is a type of recurrent neural network (RNN) architecture that has gained widespread popularity in recent years for its ability to effectively model sequential data. Since its inception in 1995, several variants of the LSTM architecture have been proposed, each with its own set of computational components and hyperparameters. Despite the popularity of LSTM, there has been a lack of large-scale studies comparing the performance of different LSTM variants on a variety of tasks. And so, in [3] a team of researchers conducted a large-scale analysis of eight LSTM variants on three representative tasks: speech recognition, handwriting recognition, and polyphonic music modeling. The hyperparameters of all LSTM variants for each task were optimized separately using random search, and their importance was assessed using the functional Analysis of Variance (ANOVA) framework.

The study is the largest of its kind on LSTM networks, summarizing the results of 5400 experimental runs that took approximately 15 years of CPU time. The study aimed to identify the most critical components of LSTM and provide guidelines for efficient adjustment of hyperparameters. The results of the study show that none of the LSTM variants can significantly improve upon the standard LSTM architecture, and that the forget gate and the output activation function are the most critical components. The forget gate is a unique component of LSTM that allows the network to selectively forget information from previous time steps. The output activation function is the function that converts the output of the LSTM cell to a usable form for the next layer in the neural network. The study found that these two components are critical for the performance of LSTM on all three tasks. The study also observed that the hyperparameters of LSTM are virtually independent, meaning that the performance of the network is not significantly impacted by the interaction between different hyperparameters. This finding is important because

it allows for more efficient optimization of hyperparameters and reduces the computational resources required to train LSTM networks.

Overall, the study provides valuable insights into the performance of different LSTM variants on a variety of tasks and highlights the critical components of LSTM. The findings have practical implications for researchers and practitioners using LSTM in machine learning applications, as they provide guidelines for efficient adjustment of hyperparameters and highlight the importance of the forget gate and output activation function in LSTM architecture.

1.3 Disadvantages of the LSTM Model

While LSTM has become a popular and effective architecture for modeling sequential data, there are still some limitations and challenges associated with its use. Some potential limitations of using LSTM for sequential data modeling include:

1) Computationally expensive: LSTM models can be computationally expensive to train, especially when dealing with large datasets or complex architectures. This can require significant computing resources and time, which can be a barrier for some applications.
2) Difficulty in interpreting model behavior: The complex architecture of LSTM models can make it difficult to interpret how the model is making predictions or to understand the underlying mechanisms driving its performance. This can make it challenging to identify and address issues or to optimize the model for specific tasks.
3) Sensitivity to hyperparameters: LSTM models have many hyperparameters that need to be carefully tuned to achieve optimal performance. This can be time-consuming and requires significant expertise in machine learning and LSTM architecture.

4) Limited ability to handle long-term dependencies: While LSTM is designed to handle long-term dependencies in sequential data, there may still be cases where the model struggles to capture complex relationships between distant time points in a sequence. This can lead to errors or suboptimal performance in some cases.
5) Overfitting: Like any machine learning model, LSTM can be prone to overfitting when the training data does not adequately represent the underlying patterns in the data. This can result in poor generalization performance on new data.
6) Limited ability to handle non-sequential data: While LSTM is designed for sequential data modeling, it may not be well-suited for other types of data. This can limit its utility in some applications where non-sequential data is also important.

Overall, while LSTM has many advantages for modeling sequential data, it is important to carefully consider its limitations and potential challenges when applying it to specific tasks. Researchers and practitioners should be aware of these limitations and work to address them to ensure optimal performance and generalization in their applications.

1.4 LSTM Model as partial differential equations

Differential equations play an important role in LSTM because they are used to model the dynamics of the system over time. The dynamics of the system are determined by the equations that govern the behavior of the system, and in LSTM, these equations describe how the network processes and updates the input at each time step according to [7]. By modeling the dynamics of the system using differential equations, LSTM can capture the underlying patterns and trends in the data, even when the data is complex and difficult to model. However, the use of differential equations in LSTM can also pose challenges. One of the main challenges is that solving differential equations can be computationally expensive, especially when dealing with large datasets or

complex models. To overcome this challenge, researchers have developed various techniques for approximating the solutions to the differential equations, including using truncated power series expansions, numerical integration methods, and other optimization techniques.

Overall, the use of differential equations in LSTM is a key factor in the success of the model. By modeling the dynamics of the system over time, LSTM is able to capture long-term dependencies in sequential data, which is essential for many real-world applications. However, the use of differential equations in LSTM also poses challenges, and researchers are continuing to explore new techniques for approximating the solutions to these equations to improve the performance and scalability of the model.

1.5 Conclusion

In conclusion, this review explains how past studies forecasted the Egyptian pound using Machine learning and statistical methods, moreover, explained guidelines for using the LSTM model, while stating it connection with differential equations. And although time forecasting the Egyptian pound with ANN claimed great results, no past studies modeled the Egyptian pound using LSTM networks which offers better results for sequence data. In response to this gap, this paper time forecasts the Egyptian pound using LSTM model while integrating best practices for better accuracy.

Methodology

2.1 Data Acquisition

Egyptian pound exchange rate time series data was collected from a website focusing on business news and financial information hosted by Google named Google Finance. The data contained three exchange rates of the Egyptian pound including US dollar (USD), Euro (EUR), and Great British pound (GBP) to increase the dimensionality of the data points. Ranging from 2003 1st of December to 2023 31st of March, originally holding 6241 rows for the USD-EGP, EUR-EGP, GBP-EGP exchange rates plotted in Fig. 1.

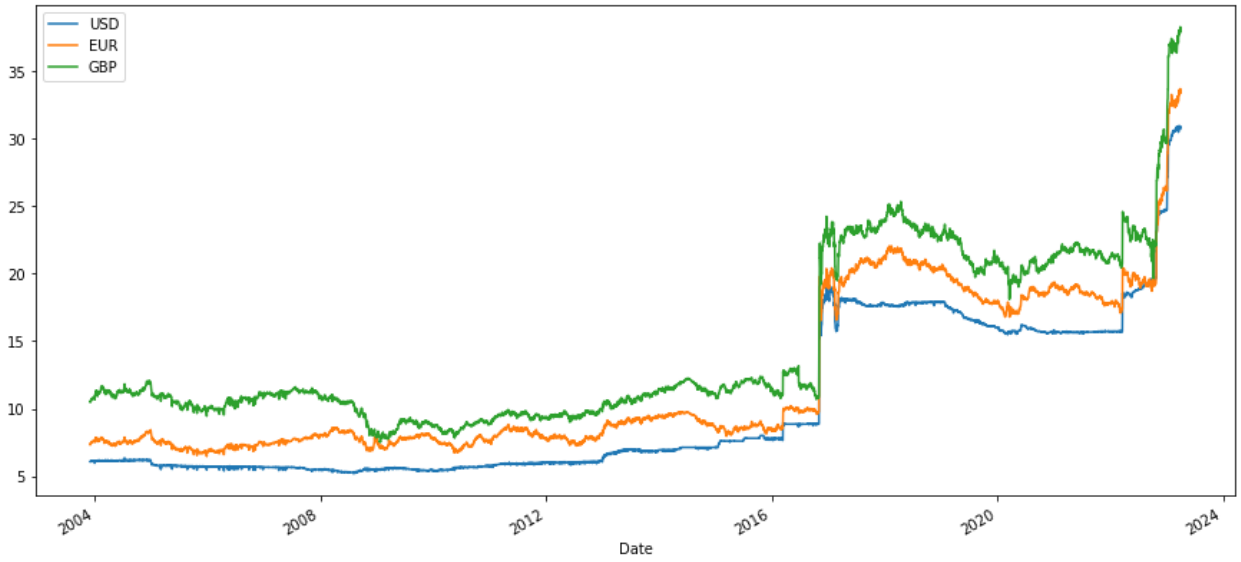

Fig. 1 Three currencies plotted over time.[1]

2.1.1 Date Cleaning and Remarks

The original data didn't cover all the dates from 2003 1st of January to 2023 31st of March for all currencies, and so, only dates that are common between the three exchange rates were kept. Leaving the time series data with 6239 rows in total, missing values were found only in the EUR and GBP currencies which were replaced with a 3-cell moving average window. That replaces the

missing cell's value with the average of the cell before it and the cell after it. Data after 2016 1st of January till 2020 1st of January was used for training the model as it contained unpredictable changes in the data and stable fluctuations, as data before 2016 1st of January was too stable and data after 2020 1st of January was to unpredictable due to the inflation. This range offered a good presentation of the current exchange rate. The three currencies were normalized using Min-Max normalization between 0 and 1. The training and testing data were plotted with respect of time in Fig 2. All plots present in this paper were produced using matplotlib in Python.

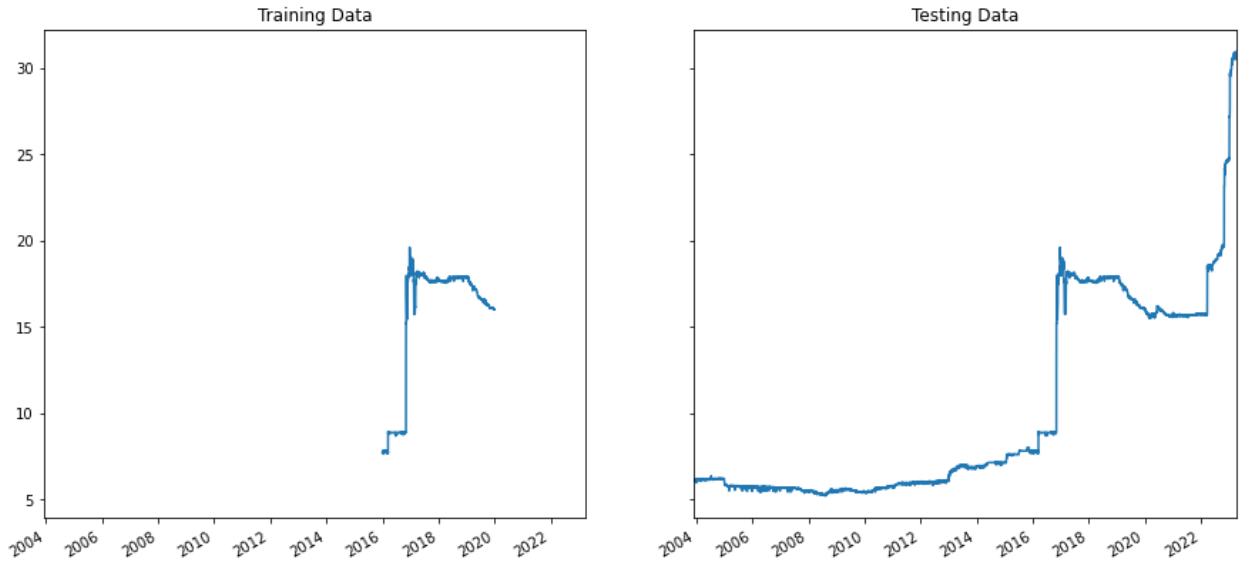

Fig. 2 Training, Testing split [2]

2.2 LSTM Network Training

The network was trained on 23% of the data ranging from 2016 1st of January to 2020 1st of January. The data was passed at a rate of 10 data points per batch with a batch size of 10 for 250 epochs, given 9 data points for all three currencies the model must predict the last 10th data point for the USD exchange, using mean absolute error as the loss function with the "Adagrad" optimizer which according to [6] outperforms other standard optimizers while trained using image datasets of different characteristics. Conventional methods of splitting the dataset to 80% training

data and 20% testing data resulted in terrible scores, as 80% of the data was very stable and not presentational of the current fluctuations.

2.2.1 Network architecture

The network consists of 2 LSTM layers, the first layer returns its hidden states for all three currencies and the second layer only returns the output of the USD currency. After which, 2 Dense layers, first one with a Leaky ReLU activation function (alpha = 0.3) , and second one with linear activation as shown in Fig. 3. This architecture was inspired by [4] which also used this architecture to predict stock market indices. This model was created with Keras and Tensorflow in python.

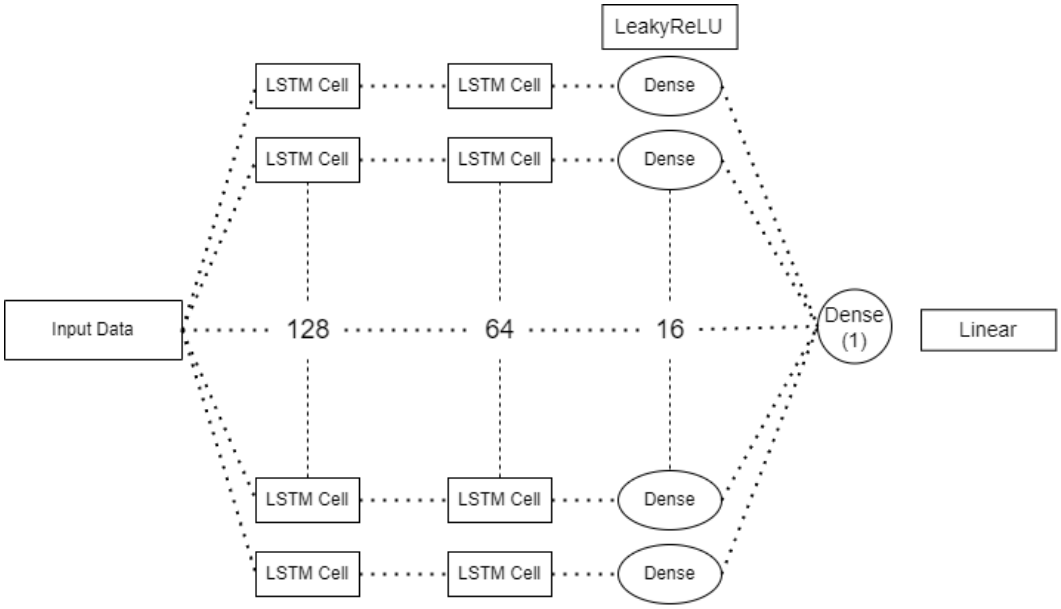

Fig. 3 LSTM Model Architecture [3]

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128 , input_shape = (10,3),
return_sequences =True))
model.add(tf.keras.layers.LSTM(64 , return_sequences = False))
model.add(tf.keras.layers.Dense(16,activation='LeakyReLU'))
model.add(tf.keras.layers.Dense(1,activation='linear'))
```

Fig. 4 The model's code written in python [4]

2.2.1 Backpropagation

Backpropagation is an algorithm used to update a neural network's weights to minimize a loss function using the chain rule. The backpropagation algorithm can be utilized for neural networks, economic systems, or to any system composed of plain calculations as mentioned by [1]. The following equations will explain the predefined settings of our network in symbols followed by the required derivatives.

2.2.1.1 Terminology

Given an input for the first layer which has an activation output for each neuron.

$$x_i^{(l)}, i \in 1 \dots n^{(l)}, l \in 1 \dots 4 \quad (1)$$

$$a_i^{(l)}, i \in 1 \dots n^{(l)}, l \in 1 \dots 4 \quad (2)$$

$$a_i^{(l)} = x_i^{(l)}, l = 1 \quad (3)$$

- $x_i^{(l)}$: Input value for the input layer.
- $a_i^{(l)}$: Activation output from each neuron.
- n^l: Number of neurons in that layer, in our case 128, 64, 16, and 1 for each of the four values of l respectively.
- l: Index of each layer starting from 1 till 4.

When l is equal to 1 (hence the first layer), the input is passed as it is to the activation output without passing through an activation function. In contrast, each layer's activation except the first is derived from the activation of the previous neurons multiplied by weight and added with a bias and then passed through an activation function.

$$z_i^{(l)} = \left(\sum_{g=1}^{n^{(l-1)}} w_{ig}^{(l)} a_g^{(l-1)}\right) + b^{(l)}, i \in 1 \dots n^{(l)}, g \in 1 \dots n^{(l-1)}, l > 1 \quad (4)$$

$$a_i^{(l)} = f(z_i^{(l)}) \qquad (5)$$

- $z_i^{(l)}$: The neuron's value before the activation function.
- $w_{ig}^{(l)}$: The weight of a current layer's neuron and each associated neuron from the previous layer.
- $b^{(l)}$: The bias associated with each layer, indexed from 1 to the number of layers.
- $a_g^{(l-1)}$: The activations in the previous layer.
- $f(z_i^{(l)})$: The activation function that produces the activation output for each neuron.

The output layer and the cost function are calculated from one neuron and is derived as the following:

$$a^{(4)} = \sum_{i=1}^{n} a_i^{(3)} w_i^{(4)} \qquad (6)$$

$$c = |a^{(4)} - y| \qquad (7)$$

- $a^{(4)}$: The last neuron's activation.
- y: The actual value to be predicted.
- c: The mean absolute error (cost function).

2.2.1.2 Minimizing the cost function.

The algorithm utilizes the derivative of the cost function c with respect to all the weights $w_{ig}^{(l)}$. In hence, measuring the effects of the weights with the cost function, which chooses how much $w_{ig}^{(l)}$ should change to minimize the cost function. This derivative is composed of many partial derivatives due to the chain rule as stated below:

$$\frac{\partial c}{\partial w^{(l)}} = \frac{\partial z^{(l)}}{\partial w^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial c}{\partial a^{(l)}} \qquad (8)$$

The derivatives of each term in the left-hand-side:

$$\frac{\partial c}{\partial a^{(l)}} = \frac{a^{(l)} - y}{|a^{(l)} - y|} \qquad (9)$$

$$\frac{\partial a^{(l)}}{\partial z^{(l)}} = \acute{f}(z^{(l)}) \qquad (10)$$

$$\frac{\partial z^{(l)}}{\partial w^{(l)}} = a^{(l-1)} \qquad (11)$$

The final equation to calculate the derivative:

$$\frac{\partial c}{\partial w^{(l)}} = a^{(l-1)} \acute{f}(z^{(l)}) \frac{a^{(l)} - y}{|a^{(l)} - y|} \qquad (12)$$

- $\frac{\partial c}{\partial w^{(l)}}$: The effect of the weights at layer l with the cost function.
- $\frac{\partial c}{\partial a^{(l)}}$: The effect of the activation function on the cost function.
- $\frac{\partial a^{(l)}}{\partial z^{(l)}}$: The effect of the neuron's function on the activation function.
- $\frac{\partial z^{(l)}}{\partial w^{(l)}}$: The effect of the weights on the neuron's function.

The new weight is calculated by subtracting the derivative multiplied by a learning rate, which changes for each layer due to the "Adagrad" optimizer, and the same step is down in a backward fashion for all previous layers.

$$w\_new^{(l)} = w^{(l)} - \alpha \frac{\partial c}{\partial w^{(l)}} \qquad (13)$$

- $w\_new^{(l)}$: The new weight to be replaced.
- α: The learning rate, usually equal to 0.01.

2.2.5 LSTM Cell equations

The LSTM consists of four main gates output gate, input gate, input candidate gate, forget gate, and the cell state according to [10]. The gates are used to control the amount of data transferred to the next layer or LSTM unit.

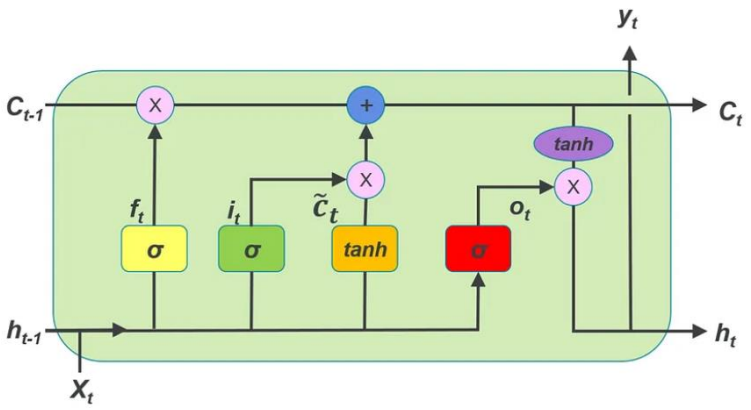

Fig. 5 LSTM Cell Unit [5]

Firstly, some information needs to be removed from the previous cell state, hence, the forget gate is used to take the current input x_t , past cell's hidden state h_{t-1} to combine them and pass them through a sigmoid function with their respective weight and a single bias. The weights and bias could be matrices depending on the dimensionality of the input data. The f_t value decides how much of the prior states should be forgotten. If f_t = 0 , then forget everything. If f_t = 1, then forget nothing.

$$f_t = \sigma(x_t U^f + h_{t-1} W^f + b^f) \qquad (14)$$

- x_t: Input to the current timestamp.
- U^f: Weight matrix associated with the forget gate.
- h_{t-1}: Hidden state of the previous timestamp.
- W^f: Weight matrix associated with the forget gate.

- b^f : Bias matrix associated with the forget gate.

Secondly, the input gate is responsible for giving a scale for the importance of the current timestamp input deciding whether to add it or not. We will be applying the sigmoid function over it. As a result, the value of i at timestamp t will be between 0 and 1.

$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i + b^i\right) \qquad (15)$$

- U^i: Weight matrix associated with the input gate.
- W^i: Weight matrix associated with the input gate.
- b^i : Bias matrix associated with the input gate.

Thirdly a candidate for the cell state is used to calculate the cell state at the current timestamp.

$$\tilde{C}_t = tanh(x_t U^c + h_{t-1} W^c + b^c) \quad (16)$$

- U^c: Weight matrix associated with the candidate gate.
- W^c: Weight matrix associated with the candidate gate.
- b^c : Bias matrix associated with the candidate gate.

Then the new cell state is derived from the equation below, this holds remnants of data from all past training examples, hence forming a propagating memory.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \qquad (17)$$

Fourthly, the output gate is responsible for outputting the predicted value at the current timestamp via the equation below.

$$O_t = \sigma(x_t U^o + h_{t-1} W^o + b^o) \qquad (18)$$

- U^o: Weight matrix associated with the output gate.

- W^o: Weight matrix associated with the output gate.
- b^o : Bias matrix associated with the output gate.

Also, the hidden state which could be optionally passed to the next layer is calculated via equation below.

$$h_t = tanh(C_t) * O_t \qquad (19)$$

In conclusion, LSTM can extend feed-forward neural networks by adding more weights and biases, increasing the flexibility of the input data. Furthermore, creating a propagating memory feature and making it better for complex time-series data.

2.3 Testing and Validation

Testing data included the whole dataset and was validated using mean absolute error (MAE) root mean squared error (RMSE) and mean squared error (MSE). Validation was done after inverse normalizing the normalized predicted values from the model. Moreover, the days passed in single batch was specified as 10, then the model would intake 9 days and predict the 10^{th} day which was done via generator.

Results

The model achieved a remarkable accuracy compared to the small given data, and attained the following scores:

Type	Training Data	Test Data
RMSE	0.03748	0.01591
MAE	0.01016	0.01012
MSE	0.0014	0.00025

Table 1 Different results using multiple metrics.

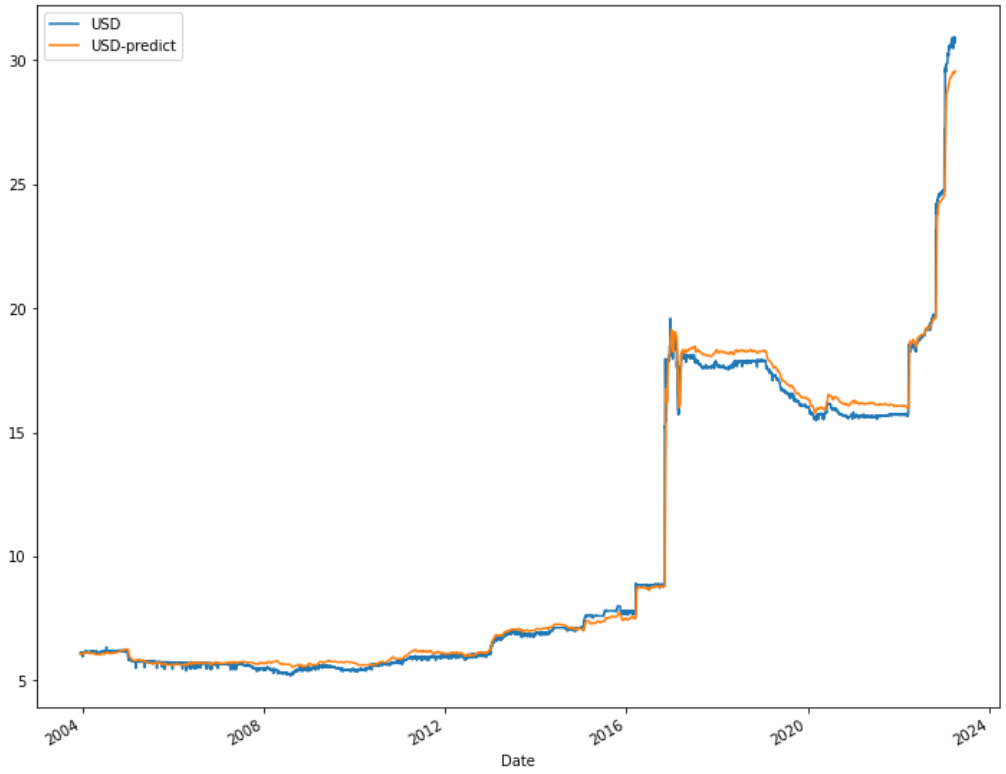

Fig. 6. USD-EGP exchange rate compared to model's predicted values. [6]

As shown in Fig. 6., our model was validated on the testing data and produced predictions colored as the orange line, and compared the actual values of the testing data, colored as blue line. Surprisingly, it was able to predict the start of the data with precise accuracy and was able to

account for sudden changes in the data with decent accuracy. This is due to the careful selection of the training data to account for the high volatility of the testing data. Utilizing the small batch size helped in keeping the model broad through a large count of epochs, making it fit for sudden changes.

Conclusion

LSTM networks prove excellence in time-series forecasting of the Egyptian currency, as the volatility of the data increase, LSTM handles these sudden changes with near perfect results. If the training data was a bad presentation of the current exchange rates, results would have differed, and in fact, utilizing the conventional 80% to 20% split for the training and testing data respectively yielded terrible results. And so, further research needs to be done on choosing the most presentational part of the time-series data to better predict current values, as a result, improving the model to be used for a volatile prediction.

References

[1] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE. Institute of Electrical and Electronics Engineers*, *78*(10), 1550–1560. https://doi.org/10.1109/5.58337

[2] D. Rofael and R. Hosni, "Modeling exchange rate dynamics in Egypt: Observed and unobserved volatility," *Mod. Econ.*, vol. 06, no. 01, pp. 65–80, 2015.

[3] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, 2017.

[4] *Researchgate.net*. [Online]. Available: https://www.researchgate.net/profile/Murtaza-Roondiwala/publication/327967988_Predicting_Stock_Prices_Using_LSTM/links/5bafbe6692851ca9ed30ceb9/Predicting-Stock-Prices-Using-LSTM.pdf.

[5] 3Blue1Brown, "Backpropagation calculus | Chapter 4, Deep learning," 03-Nov-2017. [Online]. Available: https://www.youtube.com/watch?v=tIeHLnjs5U8.

[6] A. Agnes Lydia and F. Sagayaraj Francis, "Adagrad - an optimizer for stochastic gradient descent," *Ijics.com*. [Online]. Available: https://ijics.com/gallery/92-may-1260.pdf.

[7] B. Stevens and T. Colonius, "FiniteNet: A fully convolutional LSTM network architecture for time-dependent partial differential equations," *arXiv [cs.LG]*, 2020.

[8] M. Rastogi, "Tutorial on LSTMs: A computational perspective," *Towards Data Science*, 06-Apr-2020. [Online]. Available: https://towardsdatascience.com/tutorial-on-lstm-a-computational-perspective-f3417442c2cd.

[9] A. El-Mahdy, O. Saker, and A. S. El-Shafei, “Predicting the exchange rate of the Egyptian pound using neural networks,” *Ekb.eg*. [Online]. Available: https://sjrbs.journals.ekb.eg/article_274124_63114f9083c261036ec56c003f4e74e3.pdf.

[10] H. Shah, V. Bhatt, and J. Shah, “A neoteric technique using Arima-LSTM for time series analysis on Stock Market Forecasting: Semantic scholar,” Advances in Intelligent Systems and Computing, 01-Jan-1970. [Online]. Available: https://www.semanticscholar.org/paper/A-Neoteric-Technique-Using-ARIMA-LSTM-for-Time-on-Shah-Bhatt/40599c57487aa34d7b635a8f23f35676269a8081