

Deep Learning Assignment 3- Stock Price Prediction using RNN

Rohan Padaya

A1872217

The University of Adelaide

rohanvasant.padaya@student.adelaide.edu.au

Abstract

This paper investigates the efficiency of various Recurrent Neural Network architectures for stock price prediction using historical data of competing brands like Apple and Samsung. Recurrent Neural Networks (RNNs), known for their effectiveness in processing sequential data, are explored in various configurations to evaluate their capabilities in predicting financial time series. We examined and compared several models, including a Vanilla RNN, an Enhanced RNN with additional layers like Dropout and Batch Normalization, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) networks. The Vanilla RNN model achieved a validation Mean Squared Error (MSE) of 0.00047384 for Apple and 0.00036177 for Samsung, demonstrating remarkable predictive accuracy. Interestingly, the Enhanced RNN, designed to be more complex and robust, did not significantly outperform the simpler Vanilla model. In contrast, LSTM and GRU models, while more sophisticated in capturing long-term dependencies, showed higher MSEs in our experiments. The baseline Vanilla RNN model not only provided a strong benchmark but also highlighted the importance of model selection based on data characteristics and task complexity in stock price prediction.

1. Introduction

In recent times, the success of deep learning technology in various fields has led to significant advancements in its application, particularly in addressing challenges in predicting stock price trends [1]. RNNs, or Recurrent Neural Networks, are a type of deep neural network that specializes in handling sequential data, making them ideal for applications like time series analysis and natural language processing [2]. Their unique complex architectures like Long Short-Term Memory (LSTM) are capable of capturing temporal dependencies, making them ideally suited for analyzing and predicting stock market trends [3].

Predicting stock prices is a complex task due to the plethora of factors and market noise involved. Current methods for forecasting stock prices often rely on a limited selection of influential features and factors [4]. In the contemporary machine learning field, there are numerous research papers that discuss using RNNs for stock market forecasting. The paper [4] uses 4 hidden LSTM layers for stock market price prediction. [1] explores different architectures of Recurrent Neural Networks such as AT-RNN-M, AT-LSTM-M, and AT-GRU-M to find the best model for stock price prediction.

1.1. Problem Description

In this paper, we aim to explore different architectures on real-world data of the stock market for popular brands like Apple and Samsung. Our research begins by establishing a baseline Vanilla RNN model, serving as a reference model for subsequent comparisons. We then delve into more sophisticated versions, adding layers like Dropout and Batch Normalization to create what we call Enhanced RNNs. We also dive into more complex structures such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) models. Each of these models is tested with different hyper-parameter settings, allowing us to thoroughly evaluate their abilities in predicting stock prices.

Throughout this study, the models are subjected to different hyper-parameter settings to assess and optimize their performance. The primary metric for evaluating the effectiveness of each model is the Mean Squared Error (MSE) on validation and test datasets. This metric will help us determine how accurately each model predicts stock prices and which architecture is most effective for this purpose in the context of financial time-series analysis.

1.2. Dataset Description

For our study on stock price prediction using Recurrent Neural Networks (RNNs), we utilized datasets involving the historical stock prices of two major companies, Apple and Samsung. These datasets were obtained from the Ya-

hoo Finance website, a reliable and commonly used platform for financial data. The Apple dataset consists of historical stock prices, including variables like opening price, closing price, adjusted closing price, highest and lowest price of the day, and trading volume. Similarly, the Samsung dataset includes similar variables, providing a comprehensive view of the stock's performance over time. Both datasets cover significant time periods of historical data which gives our model effective and vast data for training. The Apple dataset includes data points from 20 November 2018 to 17 November 2023, while the Samsung dataset spans from 20 November 2018 to 20 November 2023, ensuring a broad and relevant range for analysis. Each dataset comprises thousands of entries, reflecting the daily fluctuations in the stock market and providing a robust foundation for our RNN models. The data can be accessed from Yahoo Finance at the following URL: <https://finance.yahoo.com/>.

1.3. Method Overview

As outlined in the Introduction, our study utilizes various Recurrent Neural Network architectures for predicting stock prices using historical data of brands like Apple and Samsung. The data is initially visualized and pre-processed, making them compatible for training the models. The process involves building a baseline Vanilla RNN model with 2651 trainable parameters. This baseline model is further enhanced by adding dropout and batch normalization layers. Additionally, we experimented with more complex architectures like LSTM and GRU.

2. Method Description

This study uses TensorFlow and Keras libraries to implement and train different Recurrent Neural Network (RNN) models. All the models were run on a specialized external GPU with limited resources.

2.1. Data Visualization

After loading both Apple and Samsung datasets, we plotted their time series to check for any trends, seasonality, and volatility in the stock prices. Figure 1 and Figure 2 shows the time series plot of both Apple and Samsung brands. These visualizations are instrumental in identifying trends, seasonal patterns, and potential predictors for future stock price movements, which are further explored through modeling in the report. Also, these plots were essential to do stationarity checks of the stock prices.

2.2. Data Pre-Processing

Data Pre-processing is an important step in ensuring the quality and consistency of the input data for our RNN models. The first step in data pre-processing involved syn-

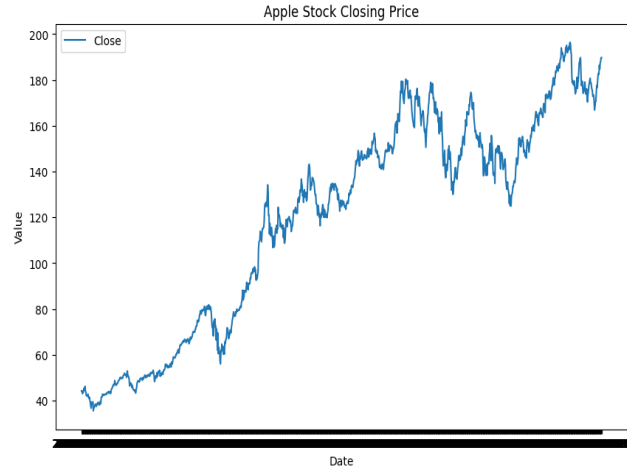


Figure 1. Apple Time Series

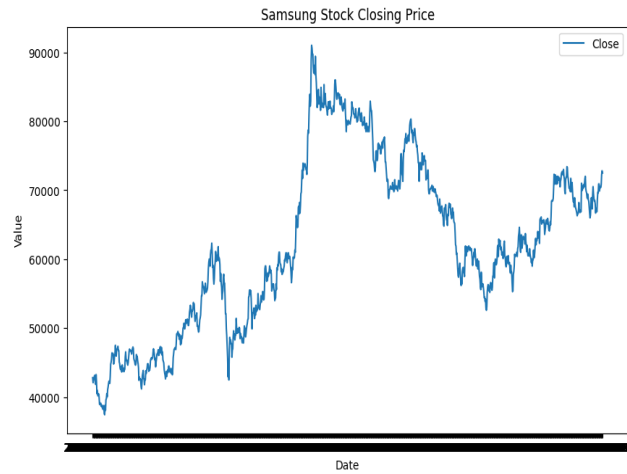


Figure 2. Samsung Time Series

chronization of our Apple and Samsung datasets as discrepancies were found between both datasets. The end dates of the datasets were different and there were 107 mismatched dates. These inconsistencies were rectified by aligning both datasets on common dates and removing the non-overlapping entries to ensure a uniform time series for comparative analysis.

After Synchronization, the datasets were subjected to scaling and normalization using the min-max scaler to transform the features into a bounded range of $[0, 1]$. Furthermore, the data was partitioned into training, validation and test sets in a 70/15/15 ratio. This splitting ensures that the model has a substantial amount of data for learning, a dedicated subset for hyper-parameter tuning during validation, and a final test set for unbiased evaluation of the model's predictive power.

In the final phase of pre-processing, we structured the

data into sequences to serve as input for the RNN models. Each sequence comprised 30 days of data, representing a month's stock information for the models to process at a time to capture short-term temporal dependencies. This pre-processed data was then utilized to train the models, setting the foundation for robust and reliable time series forecasting.

2.3. Vanilla RNN Model

The baseline model in our study is a Vanilla Recurrent Neural Network model (RNN) which comprises 2651 trainable parameters and serves as the foundational model for this study. The simplicity of the Vanilla RNN model makes it an ideal candidate for a baseline model which can act as a starting point for our comparative analysis of stock price prediction algorithms. In mathematical notation, a Vanilla Recurrent Neural Network (RNN) can be described using the following equations for each time step t in the input sequence [5]: The basic computations that happen inside the layers of Vanilla RNN are as follows:

Hidden State Computation:

$$\text{hidden}_t = F(\text{hidden}_{t-1}, \text{input}_t) \quad (1)$$

Initially, the hidden state is often a zero matrix, which, combined with the first input, is processed by the RNN cell. The hidden state and input are multiplied with initialized weight matrices, followed by an activation function like tanh [5].

Updated Hidden State:

$$\text{hidden}_t = \tanh(\text{weight}_{\text{hidden}} * \text{hidden}_{t-1} + \text{weight}_{\text{input}} * \text{input}_t) \quad (2)$$

Output Generation:

$$\text{output}_t = \text{weight}_{\text{output}} * \text{hidden}_t \quad (3)$$

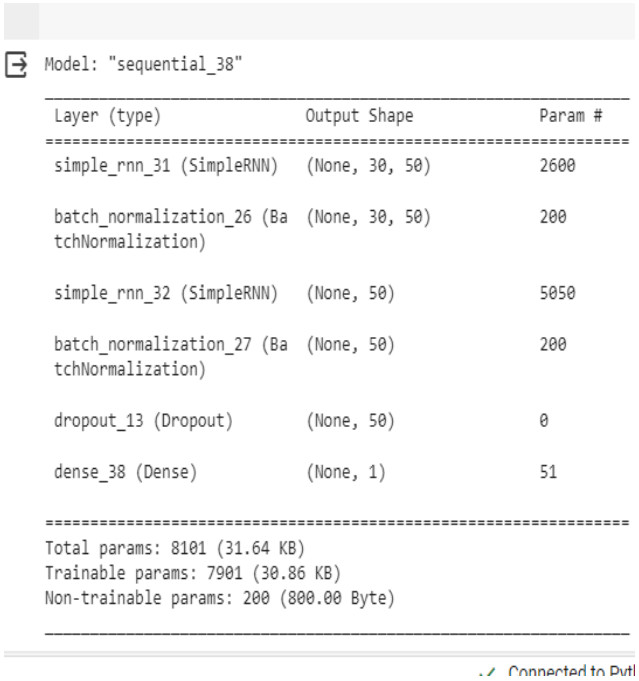
The generated hidden state is then used to produce an output at each time step [5]. This output can be obtained by passing the hidden state through a linear layer or multiplying it by another weight matrix. The newly produced hidden state is fed back into the RNN cell along with the next input, and this process continues through the sequence [5].

2.4. Enhanced RNN Model

As part of our experiment, we added more layers to the baseline model to improve its performance. Building upon the basic structure of the Vanilla RNN, this enhanced model incorporates additional layers like Dropout and Batch Normalization. Often with machine learning models, there is always a possibility of over-fitting. To combat this, we've added the dropout layer. The Dropout layer serves as a powerful tool for preventing over-fitting, by randomly omitting

a subset of features at each iteration during training. This technique ensures that the model does not become overly reliant on any specific feature and enhances its generalization capability.

On the other hand, the batch normalization layer aids in stabilizing and accelerating the learning process. By normalizing the inputs in each mini-batch, this layer ensures a consistent distribution of the data flowing through the network, which often results in faster convergence and improved overall performance. Figure 3 depicts the structure of our enhanced Baseline model.



The screenshot shows a Jupyter Notebook cell with the title "Model: 'sequential_38'". It displays a table of the model's layers, their types, output shapes, and the number of parameters. Below the table, it shows the total, trainable, and non-trainable parameters with their respective sizes in KB and bytes. At the bottom right, there is a status bar indicating "Connected to Pu..."

Layer (type)	Output Shape	Param #
simple_rnn_31 (SimpleRNN)	(None, 30, 50)	2600
batch_normalization_26 (Batch Normalization)	(None, 30, 50)	200
simple_rnn_32 (SimpleRNN)	(None, 50)	5050
batch_normalization_27 (Batch Normalization)	(None, 50)	200
dropout_13 (Dropout)	(None, 50)	0
dense_38 (Dense)	(None, 1)	51

=====
 Total params: 8101 (31.64 KB)
 Trainable params: 7901 (30.86 KB)
 Non-trainable params: 200 (800.00 Byte)

Connected to Pu...

Figure 3. Enhanced baseline model Architecture

2.5. Long Short-Term Memory (LSTM) Model

2.5.1 Need for LSTM

Traditional Recurrent Neural Network (RNN) models are very efficient at capturing temporal dependencies in sequence data, making them highly effective for tasks involving time series analysis. However, there is one notable issue in the traditional RNN models, the vanishing gradient problem. The vanishing gradient problem arises when gradients become exceedingly small during back-propagation, making it challenging to train the network effectively.

To address this, the Long Short-Term Memory (LSTM) model, a variant of RNNs, has been developed. LSTMs are designed to overcome this hurdle by maintaining a more consistent gradient flow across many time steps. Their key

advantage lies in their ability to bridge long time intervals, surpassing other RNNs, hidden Markov models, and various sequence learning methods in this regard. The term "long short-term memory" encapsulates the model's unique feature: providing a form of short-term memory for RNNs that persists across extensive sequences, potentially spanning thousands of time steps. [6]

2.5.2 LSTM Architecture

Long Short-Term Memory (LSTM) networks are an advanced type of Recurrent Neural Networks designed to address the vanishing gradient problem in traditional RNNs [7]. They are structured as a chain of cells, each containing four interconnected neural network layers that manipulate the cell's memory. The core of LSTM's effectiveness lies in its unique memory blocks or cells, which are adept at retaining information over long sequences [8].

The working of an LSTM cell is governed by three specialized gates: the forget gate, input gate, and output gate. The forget gate determines which information is discarded from the cell state, using a combination of the current input and the previous output, passed through a sigmoid function. The input gate adds new, relevant information to the cell state, again influenced by the current input and the previous output but also incorporating a tanh function to regulate the information flow. Finally, the output gate extracts the necessary information from the current cell state to form the output, which also serves as an input for the next cell in the sequence. These gates work in tandem to ensure that the LSTM cell can both maintain essential historical information and adapt to new data, making it exceptionally proficient for complex sequential tasks [9]. The figure 4 from [6] demonstrates all the above mentioned gates.

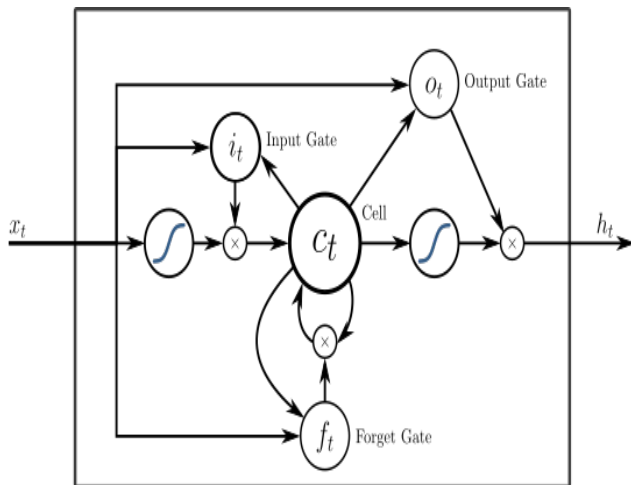


Figure 4. LSTM model Architecture

In our assignment, the LSTM model is specifically designed for the task of stock price prediction. The model architecture includes LSTM layers that are excellent at learning and remembering information over long sequences, making them particularly suitable for the time-series data of stock prices. These LSTM layers are configured to process data sequence-wise, capturing the intricate patterns and trends within the stock market. The model also incorporates dense layers following the LSTM layers to aid in the process of mapping the learned features to the desired output, which in this case is the predicted stock price.

2.6. Gated Recurrent Unit Model (GRU)

Gated Recurrent Units (GRUs) are another variant of Recurrent Neural Network models that also aim to solve the vanishing gradient problem just like LSTM. GRU architecture is more streamlined as compared to the LSTM model. They simplify the LSTM model by combining the forget and input gates into a single update gate [10]. The cell state and the hidden state are also merged resulting in a more efficient learning process with fewer parameters, which can be particularly advantageous in certain applications where model complexity is a concern. [11]

In our analysis, we have employed a GRU model for the prediction of stock prices, taking advantage of its ability to capture temporal dependencies efficiently. It starts with a GRU layer of 50 units, adept at capturing temporal dependencies, followed by a batch normalization layer to stabilize the learning process. A second GRU layer, also with 50 units, enhances the model's ability to identify complex patterns in the data. To minimize over-fitting, a dropout layer is added, providing robustness and generalization. The final output is generated by a dense layer with a single unit, making the model well-equipped for precise time-series forecasting. Figure 5 shows the architecture of our model. This compact yet effective structure, totaling 23,701 parameters, balances computational efficiency with the task of predicting stock market trends.

The idea is to explore more complex architectures for our datasets to identify the best model out of all of them. GRUs maintain a balance between computational efficiency and the capability to handle long-term dependencies, making them a compelling choice for financial forecasting tasks. Recent studies have shown the efficiency of GRU models in various sequence modeling tasks, including financial time series prediction. For instance, a study by [12] depicts the effectiveness of GRUs in predicting stock market movements, highlighting their potential to capture complex temporal relationships in the data [3].

Model: "sequential_43"		
Layer (type)	Output Shape	Param #
=====		
gru_14 (GRU)	(None, 30, 50)	7950
batch_normalization_36 (Batch Normalization)	(None, 30, 50)	200
gru_15 (GRU)	(None, 50)	15300
batch_normalization_37 (Batch Normalization)	(None, 50)	200
dropout_18 (Dropout)	(None, 50)	0
dense_43 (Dense)	(None, 1)	51
=====		
Total params: 23701 (92.58 KB)		
Trainable params: 23501 (91.80 KB)		
Non-trainable params: 200 (800.00 Byte)		

Figure 5. GRU Architecture

2.7. Experimental Analysis

2.7.1 Primary Objective

The primary objective of this study is to analyze different Recurrent Neural Network Architectures and their corresponding hyper-parameters tuning for the prediction of stock market prices for both Apple and Samsung datasets. The models mentioned in the methodology were extensively analyzed under various settings to identify the optimal solution for our use case. The analysis involves Data Visualization, Data Pre-Processing, Vanilla RNN Baseline Modeling, Enhanced RNN model, Long Short Term Memory Modeling (LSTM), and GRU modeling. In the results section, we will explain the steps taken and the results achieved through those steps in chronological order.

2.7.2 Results

Analyzing the data: In the experimental analysis of our project, we first focused on data visualization to understand the underlying patterns and trends of the stock prices of Apple and Samsung. Time series plots were generated for the closing prices of both datasets, providing a visual representation of their trends over time. Figure 1 and Figure 2 shows the time series plot of the closing price of both datasets. As we can see, the plots are non-stationary in nature indicating variability in mean and variance over time. To address the issue of non-stationarity, we applied log transformation to the closing prices. However, this transformation did not ef-

fectively stabilize the series, leading to the decision to proceed with the original non-stationary data for further analysis. This decision was based on the understanding that while non-stationarity poses challenges, the advanced capabilities of our chosen models could still potentially extract meaningful insights from the data.

The preprocessing phase involved scaling and normalizing the data using a MinMax scaler, followed by splitting it into training, validation, and testing sets with a 70/15/15 ratio. The data was then structured into sequences to be fed into the models, ensuring that each sequence encapsulates at least 30 days of data crucial for accurate forecasting.

Modeling the data: The next step of our experiment analysis involved modeling of both datasets using models such as the Vanilla RNN, Enhanced Vanilla RNN, LSTM, and GRU. Each model was rigorously evaluated on the validation data using the Mean Square Error as the primary metric. This metric is crucial as it quantifies the average squared difference between the estimated values and the actual value, with a lower MSE indicating higher accuracy.

For Apple's stock price prediction, the Vanilla RNN model achieved an MSE of 0.0004, indicating a high level of accuracy in its predictions. The Enhanced RNN model followed closely with an MSE of 0.0005. In comparison, the more complex LSTM and GRU models exhibited higher MSEs of 0.0633 and 0.0525, respectively. This suggests that while LSTM and GRU models are typically more capable of capturing long-term dependencies, in this specific scenario, the simpler Vanilla and Enhanced RNN models were more effective in modeling Apple's stock price data. The table 1 shows the Validation Mean Square Error values of the Apple Data.

Similarly, in the case of Samsung's stock price prediction, the Vanilla RNN again outperformed the other models with the lowest MSE of 0.0003. The GRU model, with an MSE of 0.0004, also performed well, followed by the Enhanced RNN and LSTM models, which had MSEs of 0.0008 and 0.0012, respectively. The table 2 shows the Validation Mean Square Error values of the Samsung Data.

Table 1. Validation MSE of Apple data models

Model	MSE
Baseline RNN	0.0004
Enhanced RNN	0.0005
LSTM	0.0633
GRU	0.0525

Hence, we finalized the Vanilla RNN as our best model. Upon finalizing, we evaluated the Vanilla RNN model on

Table 2. Validation MSE of Samsung data models

Model	MSE (%)
Baseline RNN	0.0003
Enhanced RNN	0.0008
LSTM	0.0012
GRU	0.0004

the test data. The Mean Squared Error (MSE) for Apple's predictions was impressively low at 0.0003329, while for Samsung, it was also minimal at 0.0003621, indicating a high level of accuracy in the model's predictions compared to the actual stock prices.

We plotted the prediction curves on the test data for both brands. Figure 6 and Figure 7 shows the prediction plots for our dataset.

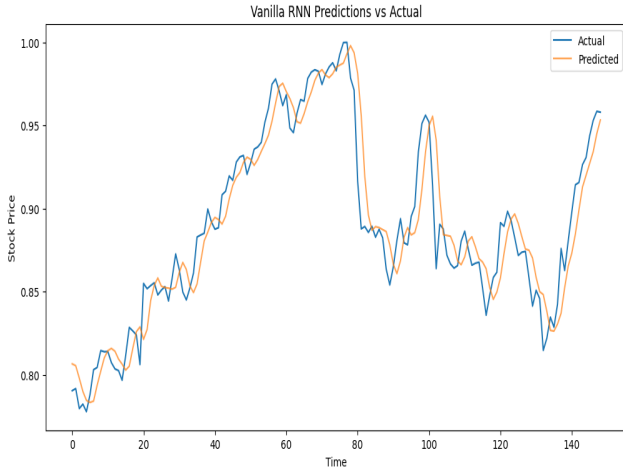


Figure 6. Prediction vs actual values for Apple

2.8. Code

The code is implemented using Python framework executed on specialized GPU in a Google Collab environment and relies on the TensorFlow and Keras libraries for building the Recurrent Neural Network models. The code is uploaded to the GitHub public repository. Here's the link to the same: <https://github.com/rohanpadaya/Deep-Learning.git>

2.9. Conclusion

This paper investigated the influence of different Recurrent Neural Network Architectures for the task of stock price prediction on the historical data of two premium brands, Apple and Samsung. The project involved a comprehensive analysis of different RNN models, including

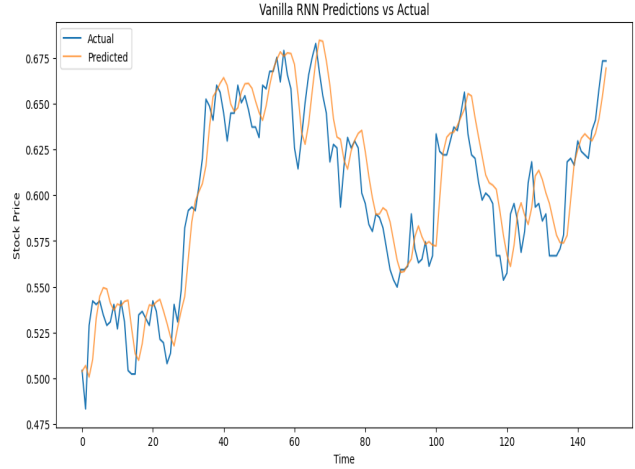


Figure 7. Prediction vs actual values for Samsung

Vanilla RNN, Enhanced RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) networks.

The findings of the study suggest that the Vanilla RNN model outperformed all the other models in terms of the Mean Squared Error (MSE) as the primary metric. The Vanilla RNN model achieved the lowest MSE on the test as well as validation data for both Apple and Samsung stock prices. This highlights the model's ability to capture the critical temporal patterns in the stock price data without the need for the additional complexity of LSTM and GRU architectures. For the specific characteristics of the financial time series data in this study, the Vanilla RNN model provided the best balance between model complexity and predictive capability. While the Vanilla RNN model was the standout in this study, further exploration can be done into other RNN variations and deep learning architectures. Due to constraints such as time and computational resources, the scope of the study was limited. In conclusion, our study contributes valuable knowledge to the field of financial time series forecasting, depicting the importance of model selection in relation to data characteristics and the trade-offs between model simplicity and complexity. These conclusions can guide future studies aiming to push the boundaries of predictive accuracy in the vibrant field of machine learning and financial data analytics.

References

- [1] J. Zhao, D. Zeng, S. Liang, H. Kang, and Q. Liu, "Prediction model for stock price trend based on recurrent neural network," *Journal of Ambient Intelligence and Humanized Computing*, 05 2020. 1
- [2] R. Rajak, "Share price prediction using rnn and lstm," Analytics Vidhya, 03 2022. [Online]. Available: <https://medium.com/analytics-vidhya/share-price-prediction-using-rnn-and-lstm-8776456dea6f> 1
- [3] E. Hoseinzade and S. Haratizadeh, "Cnnpred: Cnn-based stock market prediction using a diverse set of variables," *Expert Systems with Applications*, vol. 129, pp. 273–285, 09 2019. 1, 4
- [4] M. S. Mottaghi and M. H. Chehreghani, "A deep comprehensive model for stock price prediction," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, pp. 11 385–11 395, 06 2023. 1
- [5] M. Saeed, "An introduction to recurrent neural networks and the math that powers them," Machine Learning Mastery, 09 2021. [Online]. Available: <https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/> 3
- [6] "Long short-term memory," Wikipedia, 11 2023. [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory#cite_note-hochreiter1991-2 4
- [7] R. Zazo, P. Sankar Nidadavolu, N. Chen, J. Gonzalez-Rodriguez, and N. Dehak, "Age estimation in short speech utterances based on lstm recurrent neural networks," *IEEE Access*, vol. 6, pp. 22 524–22 530, 2018. 4
- [8] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang, "A cnn-lstm-based model to forecast stock prices," *Complexity*, vol. 2020, pp. 1–10, 11 2020. 4
- [9] A. Chugh, "Deep learning — introduction to long short term memory," GeeksforGeeks, 01 2019. [Online]. Available: <https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/> 4
- [10] L. Jing, C. Gulcehre, J. Peurifoy, Y. Shen, M. Tegmark, M. Soljagic, and Y. Bengio, "Gated orthogonal recurrent units: On learning to forget," *Neural Computation*, vol. 31, pp. 765–783, 04 2019. 4
- [11] "Gated recurrent unit networks," GeeksforGeeks, 07 2019. [Online]. Available: <https://www.geeksforgeeks.org/gated-recurrent-unit-networks> 4
- [12] Y. Gao, R. Wang, and E. Zhou, "Stock prediction based on optimized lstm and gru models," *Scientific Programming*, vol. 2021, pp. 1–8, 09 2021. 4