Recurrent Neural Networks for Stock Price Prediction

Vadde Venkata Kamala Sindhoori
University of Adelaide
Adelaide, South Australia, 5000
a1905610@adelaide.edu.au

Abstract

This paper explores various Recurrent Neural Network (RNN) models for stock price prediction. Various RNN models like vanilla RNN, Gated Recurrent Unit (GRU) and Long short-term memory (LSTM) are built, evaluated and compared to understand which models perform well with stock price prediction. The models are evaluated on the basis of Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Accuracy. The findings of this paper highlight LSTM's superior performance in stock price prediction.

1. Introduction

Stock price prediction is very important in the area of Finance. Deep learning techniques like RNNs are often employed to handle the complexities involved in stock price prediction.

Vanilla RNNs are used for basic sequential data. However, they fail to solve vanishing gradient problem and in turn fail in capturing long-term dependencies [2]. GRUs on the other hand are computationally effective with their gating mechansims and are used for many real-time applications [3]. LSTMs have advanced cell memory, gating mechanisms and are used to manage temporal dependencies and are mainly used for weather forecasting and industrial diagnostics [4, 5].

In this paper various architectures of RNNs namely vanilla RNN, GRU and LSTM are explored. Baseline models are built for each of these models and then hyper-parametric optimisation is performed on these baseline models.

The performance of all the models is evaluated and compared using metrics which are MSE, RMSE and Accuracy. These metrics are essential to understand the performance of each model. After comparison, the most ideal model is selected for stock price prediction.

In Section 2, previous works related to using various RNN models were discussed. Furthermore, in the section 3, the data used, its fetaures and the models used on the data were discussed. Finally, the section 4 involves the discussion the performance of various models on the data and which models perform best in the stock price prediction .

2. Related Works

In recent years, a lot of research has been made on RNNs especially on the models, vanilla RNN, LSTM and GRU. Even though they are simple, vanilla RNNs are known for solving tasks like language modeling and semantic modeling.

Andruszkiewicz and Rychalska [1], have shed light on interpretability and computational effectiveness of vanilla RNNs compared to GRU and LSTM by using it in textual similarity tasks.

According to Cho et al. [3], GRUs are computationally effective and help in overcoming the vanishing gradient problem. Wu et al. [7] focused on using GRU's superior time series modeling capabilities over traditional machine learning methods for Remaining Useful Life (RUL) prediction in aero-engine systems.

Hochreiter and Schmidhuber [5], deduced that with advanced gating mechanisms, LSTMs are very useful in capturing long term dependencies of data. Greff et al. [4] studied the effectiveness of LSTM in time series forecasting especially stock preidctions.

3. Methodolody

The complete code script for this assignment is available on Github ,for review purposes, please refer to the link [6]

3.1. Data

The dataset used for the assignment is taken from Kaggle, which consists of stock price data of Google spanning over different years. The data consists of open, high, low, close and Volume for each trading day. The data is divided into training and testing sets. The training data is further divided into training and validation sets for the purpose of Hyper-parametric optimisation.

3.2. Exploratory Data Analysis

The inspection of null values in training and testing set showed that there are no null values present in both training and testing sets.

The trends in all the features were examined. The figure 1 shows the trend in open stock prices. It indicates that there is a slow and consistent increase in open stock prices. There is an upward movement in the trend indicating the periods with market growth. The High 2 and Low stock prices 3 show the exact same trend as Open stock prrices.

Moreover, the trend curve for closing stock price 4 is different from others. The close price trend graph initially depicts and slow and steady increase over time. However, at around 600 days, the curve has a sudden dip indicating a major event or news. The closed stock prices then recover and indicate a steady growth over time.

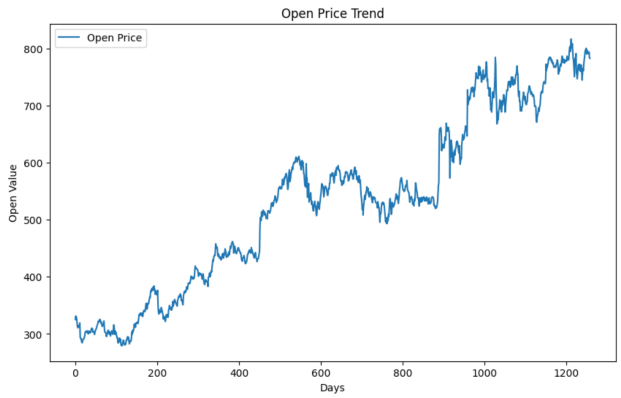

Figure 1. Open Price Trend

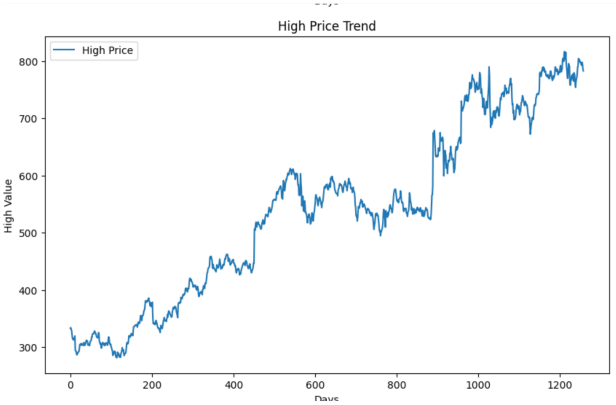

Figure 2. High Price Trend

Further inspection of correlation between variables5indicates the presence of strong and positive correlation between open, high, and low stock prices. The close stock prices show a moderate correlation with open and high stock prices and relatively higher correlation with

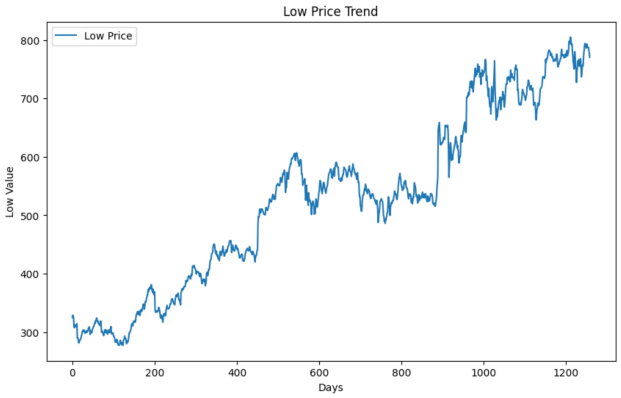

Figure 3. Low Price Trend

Figure 4. Close Price Trend

low stock prices. Volume is negatively correlated with all the other variables indicating that higher volume in trading typically results in decline in stock prices values.

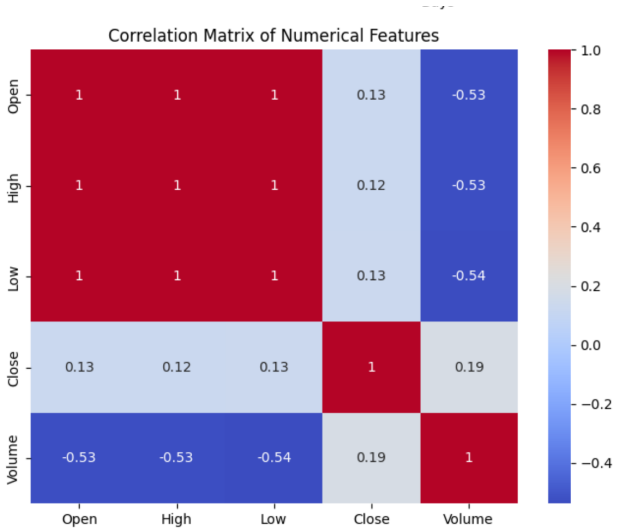

Figure 5. Correlation Matrix between the Variables

Despite having multi-collinearity where various variables in data show strong correlation, none of the variables are removed from analysis. This is because we are exploring deep learning models and these are less affected by multicollinearity. Moreover each variable in the data has unique aspect which can have a significant contribution towards the analysis. Hence, all the variables are used for analysis.

3.3. Sliding Window approach

In this paper, the sliding window approach is used for predicting stock prices. It is implemented by creating a function called 'create_sequences' which displays efficiency in structuring input-output pairs. It uses past 'X' days data to predict for next 'y' days and creates overlapping windows. With this sliding window approach, data usage is maximised and model's learning potential is enhanced. Also, using this approach showcases model's adaptability where different sequence lengths are taken making it an ideal choice for various forecasting scenarios, especially with those involving sequential data.

3.4. Vanilla RNN

Vanilla RNN is one of the basic neural networks designed for processing sequential data. The design retains information from previous steps with the help of a feedback loops in hidden layer. Vanilla RNN is majorly used in areas like series forecasting and sequence classification.

3.4.1 Baseline vanilla RNN model

The baseline vanilla RNN model consists of a single layer followed by a dropout layer(50%). The final output is generated by a dense layer. The model is trained for 50 epochs and is used on sequential data.

3.4.2 Hyper-parametric optimisation vanilla RNN model

The hyper-parametric optimisation of vanilla RNN is performed using keras tuner. The model is trained with RNN units ranging from 10-100 RNN units. The number of trials used are 10. After getting the best hyperparameters, they are applied to the basic model and are trained for 100 epochs with a batch size of 64.

3.5. GRU

GRU is one of RNNs in deep learning which is used for modeling sequential data. It is very effective in reducing vanishing gradient problem. GRU controls flow of information by using gating mechanism which consists of update gate and reset gate. They are very simple and are computationally effective compared to other networks. They are majorly used for tasks like time series prediction and speech recognition.

3.5.1 Baseline GRU model

The baseline GRU model used has a simple architecture. It has a single layer with reduced capacity (20 units). To prevent over-fitting, a dropout layer (50%) is used. The model is trained with a batch size of 64 units and is run for 50 epochs.

3.5.2 Hyper-parametric optimisation GRU model

Similar to Vanilla RNN, the GRU hyper-parametrc optimisation model used keras tuner. They are three hyperparameters used in the model. They are 'GRU Units' ranging from 10-50, 'dropout rate' ranging from 0.4 to 0.7 and 'learning rate' consisting of values 0.1, 0.01, 0.001. The tuner is executed twice for each trial for 10 trials. The best hyper-parameters then obtained are used to train the model for 100 epochs with a batch size of 64.

3.6. LSTM

LSTM is a RNN architecture which learns long term dependencies of sequential data and helps in reducing vanishing gradient problem. Similar to GRU, LSTM consists of a gating mechanism with input, forget and output gates. These gates manage the flow of information. This kind of gating mechanism in LSTM makes them effective for complex patterns such as time series forecasting. LSTMs can be used for capturing both short-term and long-term dependencies in sequential data.

3.6.1 Baseline LSTM model

The baseline LSTM model used has a single LSTM layer with 50 units. These are used to capture temporal dependencies. Then the model is generalised with the help of a dropout layer (20%) foloowed by a dense layer. With a batch size of 64, the model is trained for 50 epochs.

3.6.2 Hyper-parametric optimisation LSTM model

Similar to both Vanilla RNN and GRU, the hyperparametric optimisation of LSTM model is performed using keras tuner. The three hyper-parameters used to tune are 'LSTM units' ranging from 100 to 200, followed by 'dropout rate' whose values are betweem 0.2 and 0.5. Finally the 'learning rate' consisting of these three values 0.0001, 0.00001, 0.000001 is also tuned. The best parameters are found are used in the model and finally the model is trained for 100 epochs.

3.7. Metrics

3.7.1 Mean Squared Error

Mean Squared Error (MSE) is the average of differences between actual and predicted values after squaring them.

The lower the MSE, the higher is the performance. The formula for MSE is given as:

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

where y_i denotes the actual values,
$\hat{y}_i$ represents the predicted values,
n is the number of data points.

3.7.2 Root Mean Squared Error

Root Mean Squared Error (RMSE) is the square root of MSE making the error value in the same units as target variable. The formula for RMSE is given as:

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

3.7.3 Accuracy

Accuracy is one of the most powerful metric which is used to measure the closeness of actual and predicted values in terms of percentage. The formula for Accuracy is given as:

$$\text{Accuracy} = 1 - \frac{|y_i - \hat{y}_i|}{y_i} \times 100 \tag{3}$$

4. Results and Evaluation

4.1. Vanilla RNN

4.1.1 Baseline model

The learning curve of baseline Vanilla RNN model 6 indicates a significant drop in training loss indicating the effectiveness of model in learning the patterns. At around epoch 10, the training loss continues to decreases while validation loss attains stabilisation. The baseline model is well- generalised which is indicated by the small distance between training and validation loss.

The metrics for Vanilla RNN baseline model are as follows:

1. Mean Squared Error (MSE): 5127.2957
2. Root Mean Squared Error (RMSE): 71.6051
3. Accuracy: 94.07%

Finally, the model's predicted vs actual values graph is given in the figure 7

4.1.2 Hyper-parametric optimisation model

From the learning curve of Vanilla RNN's hyper-parametric optimisation model 8, we can understand that both the training and validation loss start to drop from the beginning implying the quickness of model to learn the patterns from the data. Both the losses stabilise at a very low value indicating the presence of a small overfitting. The model is well generalised as indicated by the gap between both the curves and it is robust which is deduced from the consistent low loss.

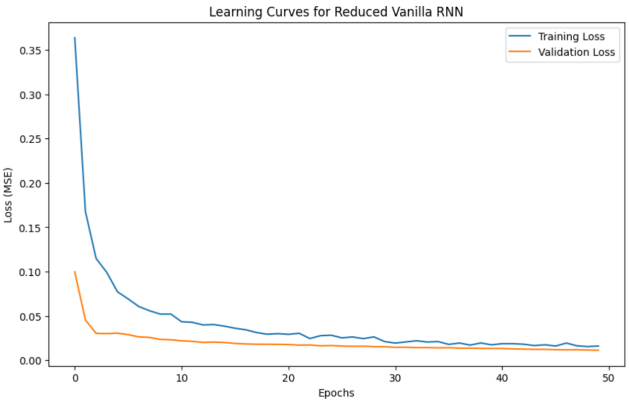

Figure 6. Learning curve for Vanilla RNN baseline model

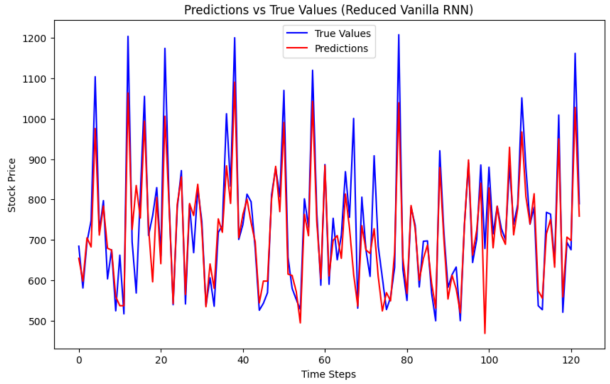

Figure 7. Predicted vs True values graph for Vanilla RNN baseline model

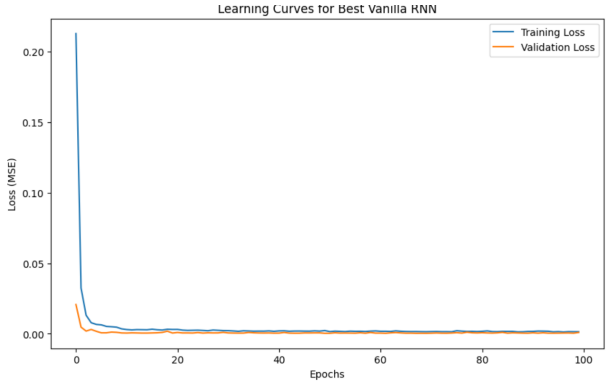

Figure 8. Learning curve for Vanilla RNN Hyper-parametric optimisation model

The metrics for Vanilla RNN Hyper-parametric optimi-

sation model are as follows:

1. Mean Squared Error (MSE): 457.2903
2. Root Mean Squared Error (RMSE): 21.3843
3. Accuracy: 97.97%

The best set of hyper-paramerts are as follows:

1. Best number of units: 60
2. Best dropout rate: 0.30000000000000004
3. Best learning rate: 0.01

The model's predicted vs actual values graph is given in the figure 9

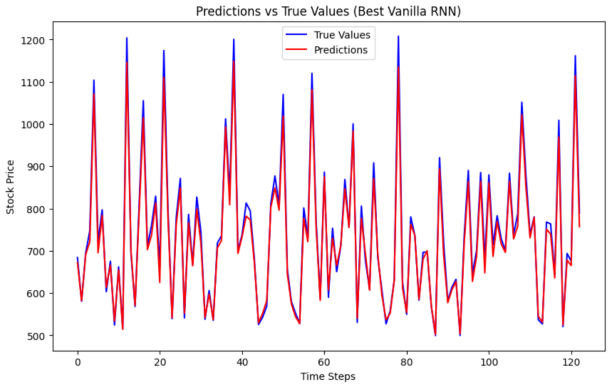

Figure 9. Predicted vs True values graph for Vanilla RNN Hyper-parametric optimisation model

4.2. GRU

4.2.1 Baseline model

From the learning curve of baseline GRU model 10, it is evident that model has rapid learning which is indicated by the sharp decrease in both training and validation loss. After epoch 11, both the curves stabilise and head towards convergence. We can infer that the model generalises too well on the unseen data which is shown by the validation loss being consistently lower than training loss. There is no sign of over-fitting which can be seen in the steady convergence of the losses.

The metrics for GRU baseline model are as follows:

1. Mean Squared Error (MSE): 236.2120
2. Root Mean Squared Error (RMSE): 15.3692
3. Accuracy: 98.38%

Finally, the model's predicted vs actual values graph is given in the figure 11

4.2.2 Hyper-parametric optimisation model

The learning curve for GRU Hyper-parametric optimisation model stabilises around 20 epochs. The learning curve indicates that there is slight over-fitting. Both the training and validation curves stabilise at low loss values.

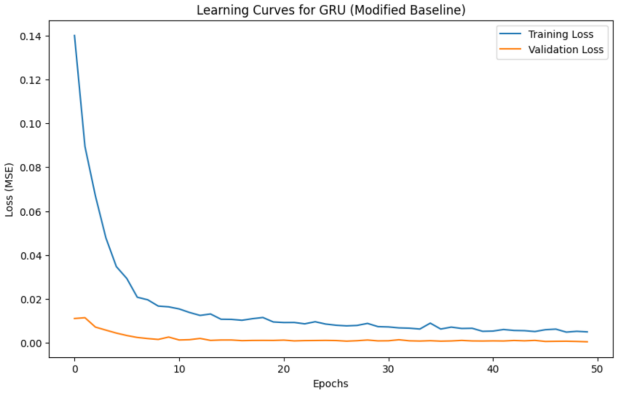

Figure 10. Learning curve for GRU baseline model

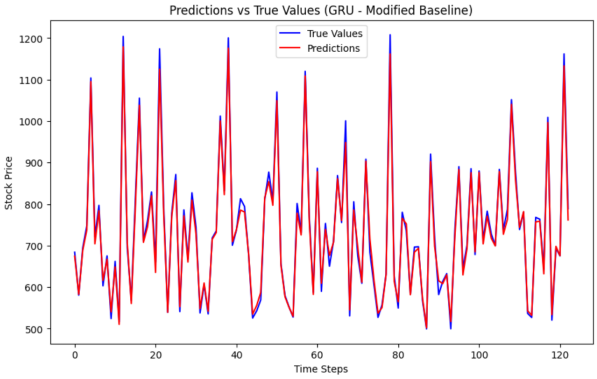

Figure 11. Predicted vs True values graph for GRU baseline model

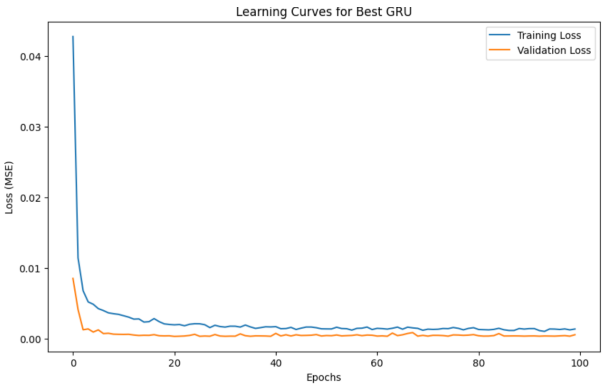

Figure 12. Learning curve for GRU Hyper-parametric optimisation model

The metrics for GRU Hyper-parametric optimisation model are as follows:

1. Mean Squared Error (MSE): 241.9506
2. Root Mean Squared Error (RMSE): 15.5548
3. Accuracy: 98.33%

Also, the best set of hyper-parameters are as follows:

1. Best number of units: 80

2. Best dropout rate: 0.2

3. Best learning rate: 0.001

The model's predicted vs actual values graph is given in the figure 13

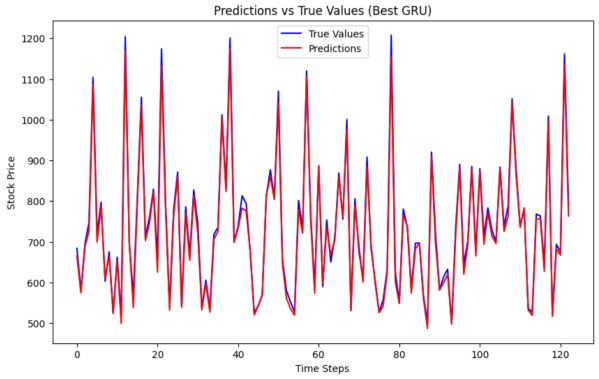

Figure 13. Predicted vs True values graph for GRU Hyper-parametric optimisation model

4.3. LSTM

4.3.1 Baseline model

There is sharp decline in training and validation loss which can be seen from the learning curve 14 of LSTM baseline model. The validation curve being consistently low than training curve indicates that there is no over-fitting. The small gap between both the curves indicates that the model is well-balanced and generalises well.

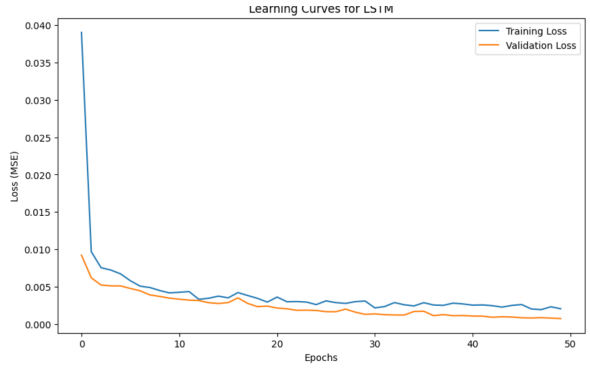

Figure 14. Learning curve for LSTM baseline model

The metrics for LSTM baseline model are as follows:

1. Mean Squared Error (MSE): 278.2855

2. Root Mean Squared Error (RMSE): 16.6819

3. Accuracy: 98.37%

Finally, the model's predicted vs actual values graph is given in the figure 15

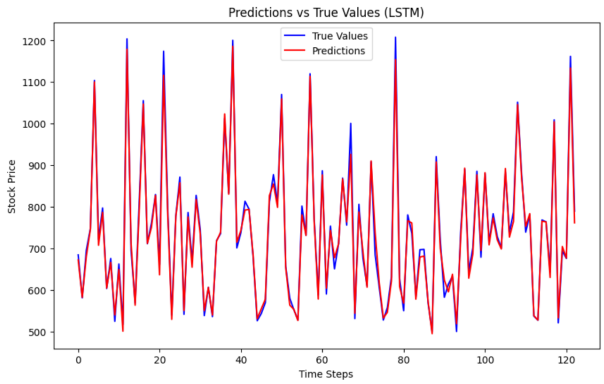

Figure 15. Predicted vs True values graph for LSTM baseline model

4.3.2 Hyper-parametric optimisation model

Similar to other models discussed above, the learning curve 16 of hyper-parametric optimisation model of LSTM shows a sharp decline in the beginning indicating the effectiveness of the model. Both the learning and validation curves plateau around 10 epochs. Both the curves then head towards stabilisation. The model avoids unwanted complexities which can be established from the stability of both the curves.

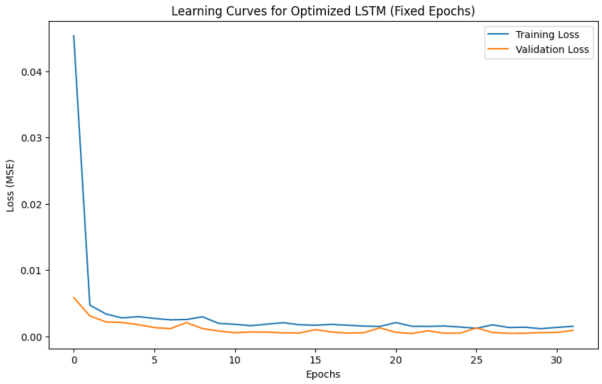

Figure 16. Learning curve for LSTM Hyper-parametric optimisation model

The metrics for LSTM Hyper-parametric optimisation model are as follows:

1. Mean Squared Error (MSE): 163.3679

2. Root Mean Squared Error (RMSE): 12.7815

3. Accuracy: 98.70%

Also, the best set of hyper-parameters are as follows:

1. Best number of units: 50

2. Best dropout rate: 0.1

3. Best learning rate: 0.01

The model's predicted vs actual values graph is given in the figure 17

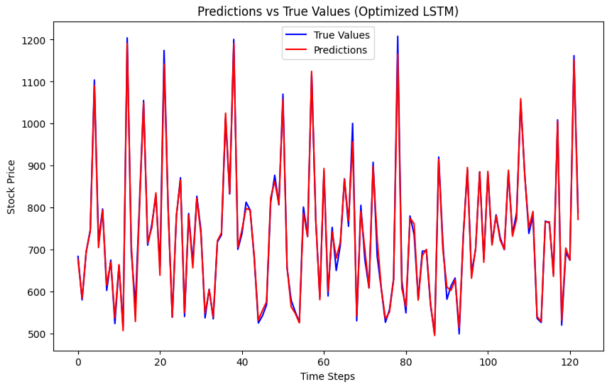

Figure 17. Predicted vs True values graph for LSTM Hyper-parametric optimisation model

5. Conclusion

The three models Vanilla RNN, GRU and LSTM were evaluated on the basis of MSE, RMSE and Accuracy. The baseline and hyper-parametric optimisation models highlighted vaarious aspects of these three models.

The Vanilla RNN baseline model showed a moderate performance with an accuracy of 94.07% mainly because of its restrictions in handling complex patterns. However, after training it with the best hyper-parameters its accuracy increases to 97.97%.

The GRU models performed better compared to the Vanilla RNN model because of their gating mechanisms. The accuracy of both the models are relatively high at 98.38% and 98.33%. However, the accuracy of baseline model is slightly more compared to the hyper-parametric optimisation model and this could be due to the slight underfitting of the hyper-parametric optimisation model which has simple architecture and increase in dropout rates.

Among all the three models, LSTM baseline and hyper-parametric optimisation models achieved the highest accuracy with baseline accuracy being 98.37% and 98.70% due to their advanced gating mechanisms.

Therefore, for predicting stock prices, LSTM with its superior performance can be considered as the best model among all the three RNN models discussed in the paper.

6. Summary and Ideas for Future work

This paper focussed on analysing, evaluating and comparing three major RNN models: Vanilla RNN, GRU and LSTM based on their predictive performance. Because of their ability in handling temporal dependencies both GRU and LSTM outperformed vanilla RNN. For each of three RNN models, a baseline model was constructed followed by the hyper-parametric optimisation of those models.The hyper-parametric optimisation model of GRU did not show any improvement compared to the baseline model because of over-regularisation. Finally, LSTM compared to other models showed high performance with high accuracy values and low error metrics.

The following improvements can be employed to the current study to improve the results:

1. Hybrid models: Combining two or more RNN models to achieve better results.

2. Other optimisers: To obtain faster convergence, optimisers like AdamW, Ranger can be used.

3. Transfer learning: Pre-trained models can be applied for better performance.

References

[1] Piotr Andruszkiewicz and Barbara Rychalska. Vanilla recurrent neural networks for interpretable semantic textual similarity. In *Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation*, pages 703–712, 2022. 1

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. 1

[3] Kyunghyun Cho. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 1

[4] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016. 1

[5] S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997. 1

[6] Venkata Kamala Sindhoori Vadde. Deep learning assignment 1. https://github.com/Sindhu-Vadde/DeepLearning, 2024. Accessed: 2024-10-05. 1

[7] Yuting Wu, Mei Yuan, Shaopeng Dong, Li Lin, and Yingqi Liu. Remaining useful life estimation of engineered systems using vanilla lstm neural networks. *Neurocomputing*, 275:167–179, 2018. 1