# The difference between Long-short Term Memory and traditional Recurrent Neural Network

"To what extent is a Long-short Term Memory module more accurate than a traditional Recurrent Neural Network module in predicting stock prices?"

Computer Science Extended Essay

Word count: 3824 words

918 uncounted words

# Contents

# Introduction

Artificial intelligence (AI) refers to the ability of computers to do tasks that requires human intelligence. In the 21st century, AI is widely used in face scan authorization, video recommendations, decision making, predicting stock market prices and etcetera (Copeland). One of the huge risk that portfolio managers have to face is the loss of principal risk, which is a danger that an investor may not get their money back, or that the value or at least a portion of the initial investment will be lost (Mehta). This essentially makes it more difficult for them to meet the client's requirement at the end of the year. Utilizing AI modules in making financial decisions will be beneficial since the chance of losing will be smaller. Although people argued that whether AI can make better investment decisions than us humans, it is undoubtedly true that AI can work with data at a much faster rate than us. This shows that AI will be able to use a wider data range than us when predicting the stock market prices, thus increasing the chance of it being accurate.

This investigation will explore if it is better for porfolio managers to use Long-short Term Memory (LSTM) or to continue using its forebear, Recurrent Neural Network. These two modules are one of the most commonly used modules in predicting the stock market performance so that essential financial decisions can be made.

A traditional recurrent neural network (RNN) module is a neural network module that takes output data from the output layer and improves its hidden layer. Figure 1 below shows the structure of a traditional RNN module.

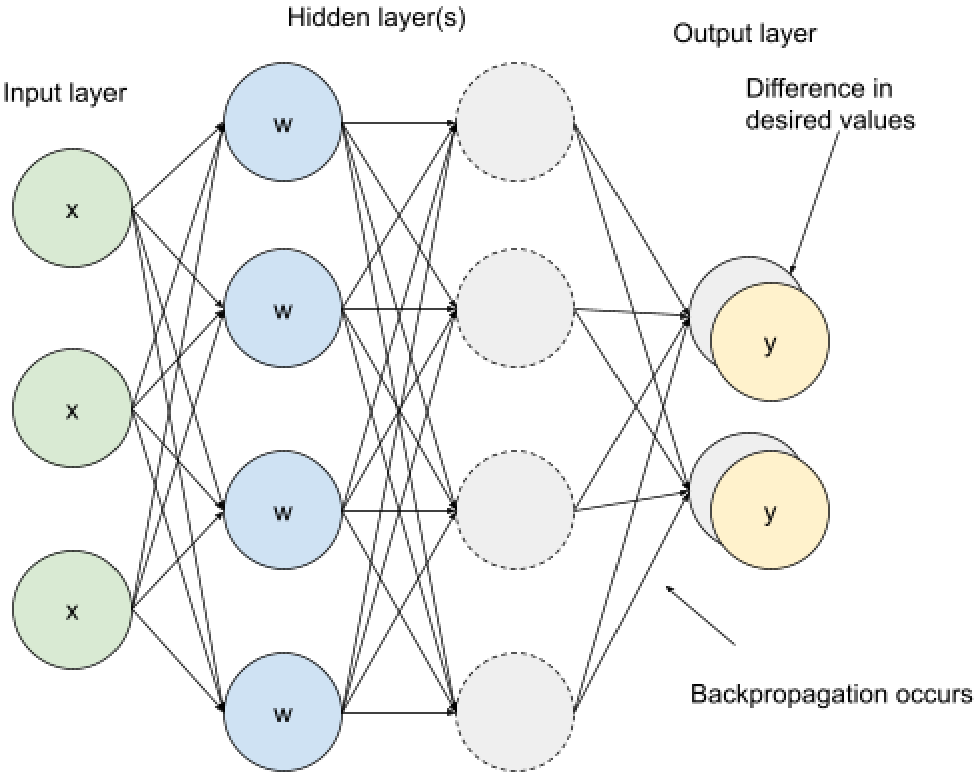


**Figure 1.** Traditional RNN module (Diagram drawn by myself)

As seen in Figure 1 above, the dataset containing all the data is input one by one through the input layer. This input is then passed to the hidden layer, where all the data processing of the RNN occurs. It is called the hidden layer because it is not directly noticeable from the input and output layers. Hidden layers are necessary for artificial neural networks if and only if the data must be segregated non-linearly (Gad). There might be a lot of hidden layers, which is drawn as grey circles in Figure 1, to process the data and this depends on the module and applications of the RNN. After the input data has gone through all the hidden layers, it will be output in the output layer. This data will be compared with the desired value (the actual price of the share) and this difference is backpropagated back to the hidden layer to correct the hidden layers (Shiriram et al.). This essentially allows RNN to analyze trends which can be used to predict the opening, and closing prices of shares the next day. Therefore, we can conclude that the more data that the RNN module is given to process, the more accurate the prediction of the actual stock price will be since it is being trained more often. However, this is not always the case. In the event of exploding or vanishing gradient which will be explored later, RNN will suffer from a loss of accuracy.

Figure 2 below shows the structure of typical structure of the LSTM module, which is a RNN module with three additional gates and a cell state.

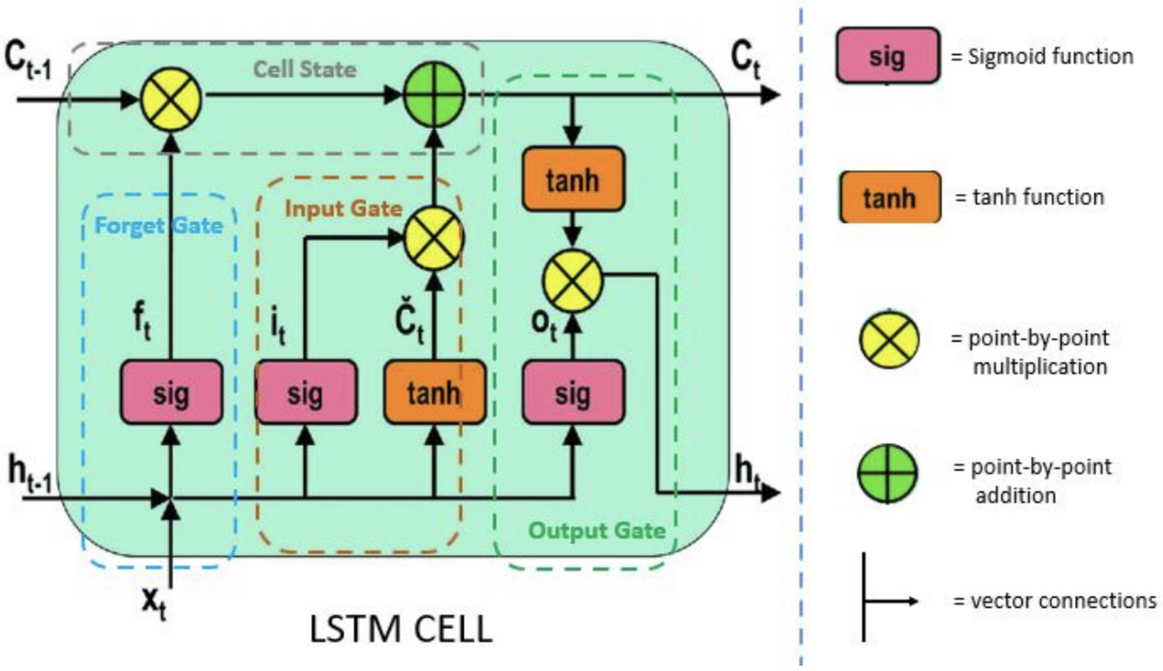


**Figure 2.** LSTM Cell Diagram taken from a website (Singhal)

As shown in Figure 2 above, LSTM module is a type of RNN module but with a cell state and three additional gates: forget gate, input gate, output gate. Firstly, the LSTM contains a forget gate which is responsible for deciding which values are needed and which are optional. The sigmoid function receives data from the current input and the hidden state. The sigmoid function then determines if the part of the old output is required by producing values between 0 and 1, with 1 indicating that it is required. The forget gate output is then utilised in point-by-point multiplication.

Secondly, The input gate determines which data may be entered into the network. The cell status is updated by sending the current and prior concealed states into the second sigmoid function. The values are translated between 0 and 1, with 0 being the most significant. The same data will be sent into the tanh function, which will generate a vector with values ranging from -1 to 1. This is then used for multiplication by points.

The last gate in LSTM module is the output gate. The next hidden state for prediction is finalised by the output gate. The third sigmoid function is fed data from the current and prior hidden states. To determine which data the current state should contain, this value is multiplied by the new cell state obtained by sending the cell state via the tanh function.

In short, LSTM module is the improved version of its traditional predecessor which has additional logic gates which addresses the vanishing gradient directly and the exploding gradient indirectly. The solution to each of these problems will be discussed in the body of this paper. This is important in the prediction of stocks because stocks in the market are volatile and priced based on the trader’s confidence.

# Vanishing gradient problem in RNN

The vanishing gradient problem normally occurs during backpropagation. When training RNN, network weights are updated based on the value of the gradient after each run. Network weight is a neural network parameter that modifies input data inside the network's hidden layers (“Weights and Biases - AI Wiki”). When the gradient value is too small, it will be getting smaller and smaller during backpropagation to the input layer (Basodi et al.). This will prevent the module continue to train itself because the network weight is too small there will be little to no modifications done to correct the hidden layer, which leads to the module to stop training. The stock price is dependent on the numbers of people selling it and the number of people buying it which means that the stock prices are very volatile and hard to predict since humans are not known to act rationally. Furthermore, individual stocks are also highly influenced by stock manipulators, even though stock manipulation is illegal in most countries, it is hard for regulators to detect and prove. This means that when giving a dataset consisting of the past opening and closing prices of a particular stock, the prediction obtained from RNN will not be very accurate and slightly lower accuracy will lead to huge investment lost which can damage the reputation of wealth management companies since they are all using artificial intelligence to make financial decisions like buy in or sell out. This is demonstrated when the artificial intelligence made a stock prediction and lost $440 million in less than an hour in August 2012 (Alam and Kendall). The sudden change in stock prices the day before caused confusion in the algorithm which made it buy millions of shares in over a hundred stocks. Since the number of shares in the market decreased, the price of the shares was overvalued and when the Knight Capital Group discovered this, they had to sell the overpriced shares at a lower price, causing a huge lost. This shows the danger of the vanishing gradient problem for RNN.

LSTM on the other hand addresses the vanishing gradient problems which cause the learning rate of RNN to gradually decrease until it is no longer updating the hidden layers.

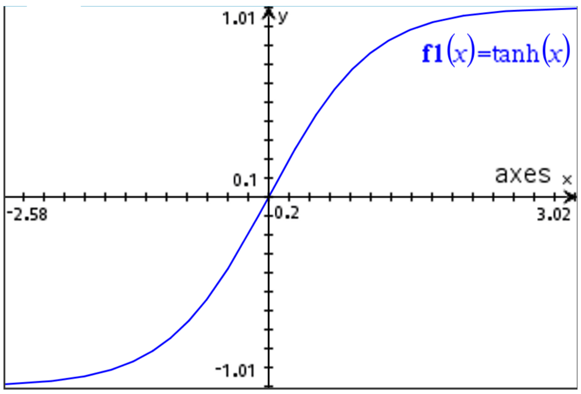


**Figure 3.** tanh graph plotted on GDC

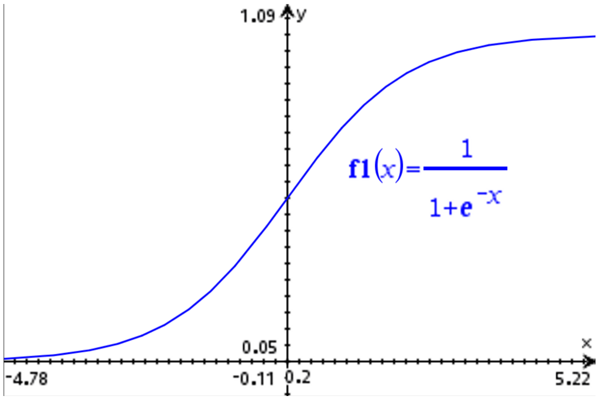


**Figure 4.** sigmoid graph plotted on GDC

LSTM typically has 2 activation functions: tanh and sigmoid functions. As seen in Figure 2, the tanh will give values between -1 to 1. This demonstrates that LSTM may employ tanh as a non-linear activation function to control the data flowing through the network. Since the tanh's second derivative is able to survive longer, this allows the LSTM module to avoid data fading (Singhal). Another non-linear function found in LSTM is the sigmoid function. As seen in Figure 3, the value returned from a sigmoid function is between 0 and 1. This allows the sigmoid function to be implemented into the gate for deciding whether the data is important or not, 0 being unimportant and can be completely forgotten. This will allow LSTM to detect for data that is no longer required for accurate prediction and remove it to prevent it from affecting the probability prediction. Take example from the August 2012 incident, LSTM will recognised that the older data before was irrelevant and the sigmoid function will return 0 to the forget gate, telling it to remove the data which prevents fail identification of trend.

The most common place where this problem will happen is where neural network modules are predicting highly fluctuating stocks. The techniques used by LSTM suggests that it will be unaffected by the vanishing gradient problem and continue making accurate predictions regarding the stock market prices. This is because the forget gate is able to prevent false data affecting its ability identify trend. On the other hand, RNN will be heavily affected by the vanishing gradient problem. Due to high volatility, it is very likely that RNN's network weights will become so insignificant that there will be negligible updates done to correct the hidden layer. Hence, producing predictions that are way off from the actual value, leading to huge loss in portfolios.

# Exploding gradient problem

Exploding gradient problem, similar to the vanishing gradient problem, is a problem that occurs in the training of an AI neural network with gradient-based learning methods and backpropagation. RNN is a learning system that employs a network of functions to comprehend and transform the input into a certain output. This sort of learning algorithm is intended to imitate the way neurons in the human brain behave. When huge error gradients build and result in very large modifications to neural network module weights during training, this is known as exploding gradients. Gradients are utilized to update network weights during training, although this process works best when the changes are minimal and regulated. When the magnitudes of the gradients add up, an unstable network is likely to form, resulting in poor prediction results or possibly a module that returns nothing meaningful at all. There are techniques for dealing with expanding gradients, such as gradient clipping and weight regularisation (“Exploding Gradient Problem Definition”).

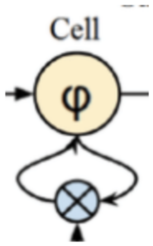
Exploding gradients can create issues with artificial neural network training. When gradients explode, an unstable network might form, and learning cannot be completed. The weights' values can potentially get so enormous that they overflow, resulting in NaN (Not a Number) values. NaN values are values that represent undefined or unrepresentable values (“Exploding Gradient Problem Definition”). Although employing a minimal learning rate, scaling the target variables, and using a standard loss function can help to avoid explosive gradients, exploding gradients will still occur in RNN with a large number of input time steps (stock prediction machines).

Although there is a forget gate in the LSTM module which can fix the vanishing gradient problem mentioned before, there is actually no mechanism in the LSTM that addresses the exploding gradient problem. After a large number of inputs, exploding gradient problem might still occur. However, in the event of exploding gradient, LSTM will use a technique known as clipping (Khurana). LSTM moves in the direction of the gradient but since the gradient is now a lot steeper, it will update its parameters with a smaller magnitude. This prevents the LSTM module to update its network weight too fast, hence providing an alternative approach to exploding gradient problem. Although this is not the ultimate solution to the problem, the LSTM module will be less affected by the problem compared to traditional RNN modules, hence reducing the probability of stock prediction machines making inaccurate predictions.

Similarly to the cause of vanishing gradient problem mentioned above, exploding gradient problem is also very likely to happen in extremely fluctuating stocks. Through clipping, LSTM will be less affected by exploding gradients than RNN when predicting prices of highly volatile stocks. Although it might still encounter this issue after a large number of inputs as mentioned above, it is less risky when comparing to investors who still uses RNN as their forecasting system.

# Solving long-term temporal problems

As shown in figure 1, a traditional RNN module normally contains three layers: input layer, hidden layer, and output layer. The hidden layer is where all the training happens. Inside the hidden layer, each input is multiplied by a weight, which is then given to a neuron. The neuron adds the product of input and weight with a bias value. This output is then output to the activation function. The activation function prevents the neural network from becoming a linear function by performing slight alterations to the output. Another advantage of the hidden layer is that it allows RNN to save information in its memory over time. However, information stored in RNN memory will not be enough if it was required to solve long-term temporal problems.



Cell State

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t$$

Cell State Equation

**Figure 5.** Cell state from (Kang)

LSTM however has cell states that can store the data for a very long time, allowing it to capture past stock price trends. The recursive nature of the cell is indicated by the looping arrows. This permits past interval information to be kept within the LSTM cell (Kang). The forget gate, which is positioned underneath the cell state, modifies it, and the input modulation gate adjusts it. The previous cell state forgets by multiplying with the forget gate and adds new information through the output of the input gates, as shown in the equation (Kang). This essentially allows LSTM to solve long-term temporal problems since past data are stored inside the memory, unlike RNN.

Since stock market is sequential and may sometimes require data from a long time ago to make accurate predictions, LSTM should be more suitable for prediction than RNN since it is built to solve temporal problems such as predicting price movements based on long sequential price data.

# Computational cost

The computational cost is the amount of time necessary to accomplish a specific task. The higher the computational cost, the longer it takes to complete the operation. The sum of parameters W in a basic LSTM module with a cell for every memory block, neglecting the biases, is calculated using the equation below:

$$W = n_c \times n_c \times 4 + n_i \times n_c \times 4 + n_c \times n_o + n_c \times 3$$

**Equation 2.** Numbers of parameters (Sak et al.)

where $n_c$ is the number of memory cells, $n_i$ is the number of input units, $n_o$ is the number of output units present in the LSTM module. This shows that as the time progresses, $n_c$ will increase since more data will be stored in the LSTM module, hence requiring more memory cells. Since $n_c$ is increasing, the number of parameters will also increase since it is proportional to the number of memory cells. Through the function stochastic gradient descent O(W) function, increasing the number of memory cells will have a direct effect on the computational complexity per time step of the LSTM module (Sak et al.). This suggests that for procedures which are required to store large temporal data (analysing trend of a stock), the computational cost of LSTM will be very high.

On the other hand, as seen in Figure 1 above, RNN does not have memory cells to remember the information. Although this can be a downside when it comes to solving temporal problems which is stated before, this also suggests that RNN will be using less resources when processing sequential data compared to LSTM, thus making RNN less computational expensive. In an ideal scenario where both modules are unaffected by the problems mentioned above, RNN will perform much better than LSTM.

Since RNN is less computational expensive compared to LSTM, it also suggests that RNN might potentially perform better than LSTM on predicting slightly fluctuating shares when a lot of inputs are taken into account. When large numbers of data are being input, LSTM will need more memory cells to store the data due to the nature of its architecture. This might negatively impact LSTM’s performance and leading to RNN outperform LSTM. Another reason why RNN might outperform LSTM in this scenario is that it is very unlikely that vanishing and exploding gradients will occur in slightly fluctuating shares, resulting in RNN less likely to produce highly inaccurate predictions.

# Experiment

In order to investigate the effectiveness of both modules in the prediction of stock market prices further, both modules are recreated using TensorFlow. Three types of datasets are chosen: slightly fluctuated, moderately fluctuated, and highly fluctuated stocks. The prediction was done in 3 intervals: one day, three days and five days. Figure 6 below shows the methodology used to perform the experiment.

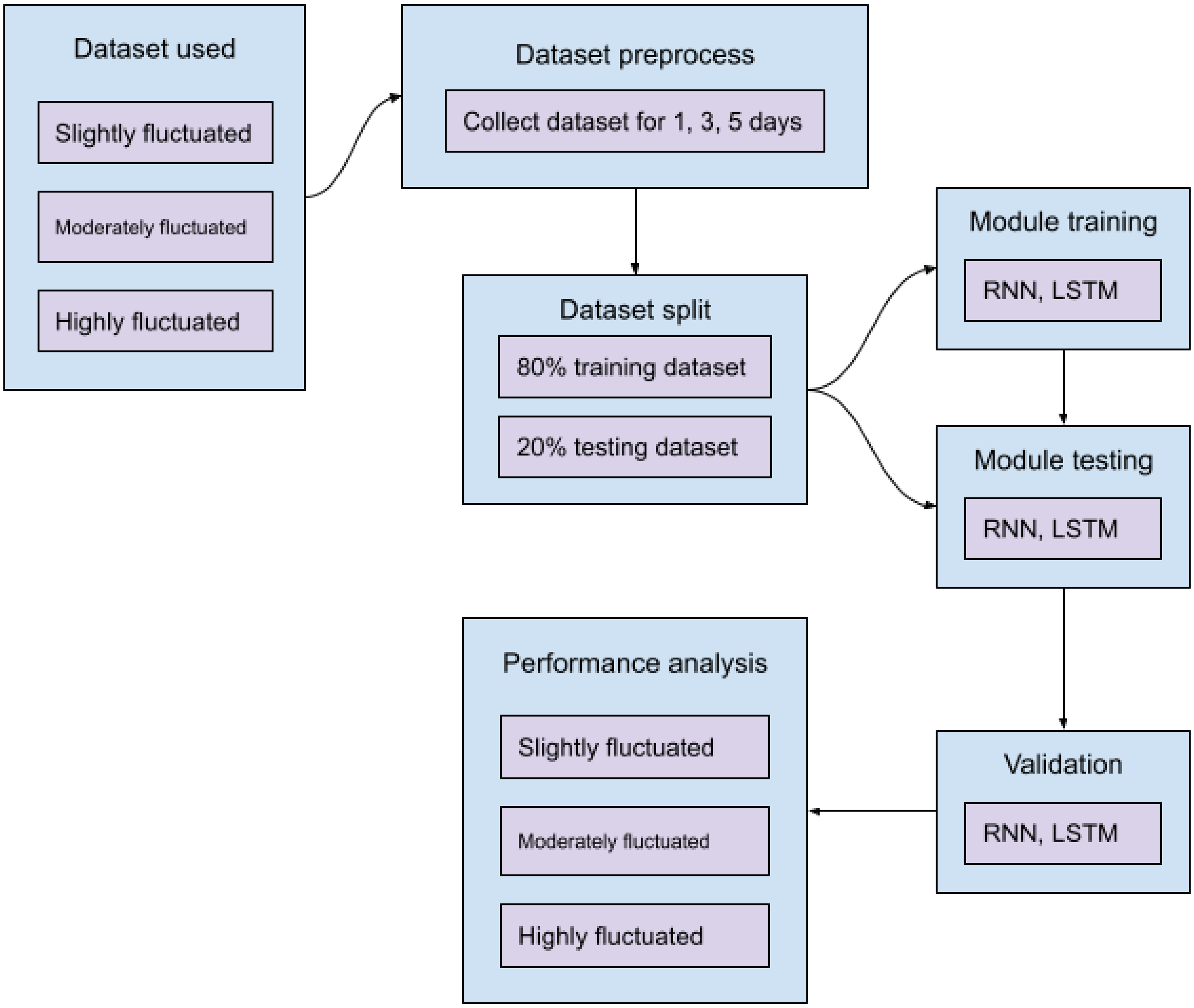


**Figure 6.** System design for predicting stock prices in various frequency domains

The volatility of stocks is used to select stocks with different levels of fluctuation. This percentage is calculated using the two equations below:

$$R_i = \ln ( C_i / C_{i-1} )$$

**Equation 3.** Equation for calculating Logarithmic Returns ("Historical Volatility (HV) Calculator")

where $R_i$ is the return of a given stock over the period i, $C_i$ is the closing price at the end of the period i, $C_{i-1}$ is the closing price at the end of the period immediately before period i.

$$HV_{Std.Dev.} = \sqrt{N}\sqrt{\frac{\sum_{i=1}^{n}(R_i - R_{avg})^2}{n-1}}$$

**Equation 4.** Equation for calculating Standard Deviation of Logarithmic Returns ("Historical Volatility (HV) Calculator")

where N is the number of periods per year.

As slightly fluctuating data, Honda Motor Company (HMC) is considered. Oracle Corporation (ORCL) is considered moderately fluctuating data whereas Intuit Inc (INTU) is considered highly fluctuating data. These data are collected from Yahoo Finance. Each dataset contains the date, opening price, highest price, lowest price, closing price, adjusted closing price and traded volume. Datasets are collected for 20 years from June 2000 to July 2010.

**Table 1.** Module architecture for slightly fluctuating data (HMC)

| Time Interval | Module | First Layer | Second Layer |
|---|---|---|---|
| 1 day | RNN | 64 | 64 |
| | LSTM | 256 | 256 |
| 3 days | RNN | 128 | 128 |
| | LSTM | 256 | 256 |
| 5 days | RNN | 128 | 32 |
| | LSTM | 256 | 128 |

**Table 2.** Module architecture for moderately fluctuating data (ORCL)

| Time Interval | Module | First Layer | Second Layer |
|---|---|---|---|
| 1 day | RNN | 256 | 64 |
| | LSTM | 256 | 128 |
| 3 days | RNN | 64 | 128 |
| | LSTM | 256 | 256 |
| 5 days | RNN | 256 | 64 |
| | LSTM | 256 | 256 |

**Table 3.** Module architecture for highly fluctuating data (INTU)

| Time Interval | Module | First Layer | Second Layer |
|---|---|---|---|
| 1 day | RNN | 64 | 256 |
| | LSTM | 256 | 128 |
| 3 days | RNN | 128 | 128 |
| | LSTM | 1024 | 512 |
| 5 days | RNN | 256 | 128 |
| | LSTM | 256 | 512 |

To validate the performance of the two neural network modules, I decided to use the classic method of finding the R squared value ($R^2$) as it is a statistical fit metric that quantifies the proportion of a dependent variable's variance that can be accounted for along the x-axis in a regression (Fernando). This basically allows me to evaluate how close the prediction curves of the two modules are to the actual value. This method is also used in finance to find the return of a financial asset after being adjusted by risks. The closer the R squared value is to 1, the more accurate the prediction is to the actual price, hence showing a better performance.

## Performance evaluation of HMC

Figures 7 and 8 are graphs plotting the performance of each neural network module and the actual value; the results are summarised in table 4. Both modules are performing well in the sense that they both have $R^2$ values extremely close to 1 (RNN is 0.98321; LSTM is 0.98346). However, the $R^2$ value of LSTM is much closer to 1 than RNN, this suggests that LSTM can make predictions on stock market prices more accurately than RNN if the data are given in daily intervals.

**Figure 7.** RNN prediction vs HMC actual (one-day)

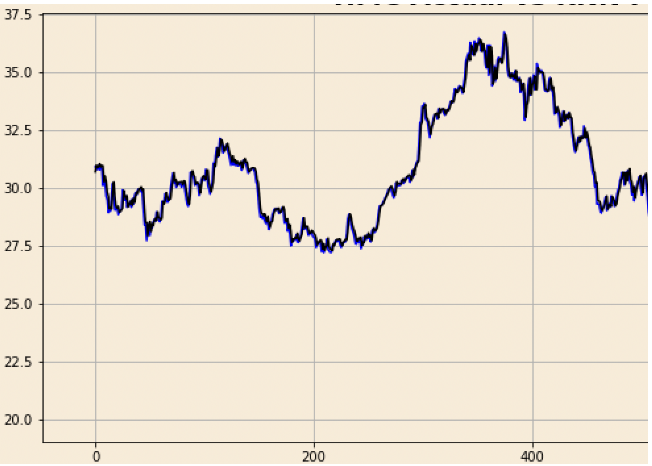


**Figure 8.** LSTM prediction vs HMC actual (one-day)

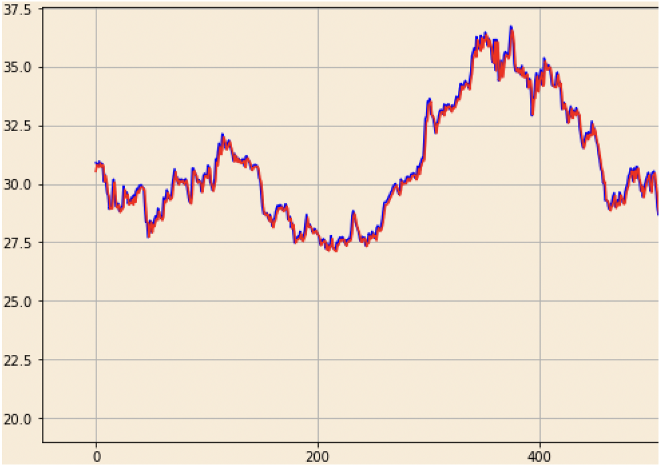


**Table 4.** Performance comparison of RNN and LSTM in one-day intervals (HMC)

| Module | $R^2$ |
|---|---|
| RNN | 0.98321 |
| LSTM | 0.98346 |

Figures 9 and 10 are graphs plotting the performance of each neural network module and the actual value; the results are summarised in table 5. Compared to their performance for one-day intervals, there is a decrease in the accuracy of their performance as both modules have a decrease in R$^2$ value. RNN also happens to make a better prediction due to its higher R$^2$ value.

**Figure 9.** RNN prediction vs HMC actual (three-day)

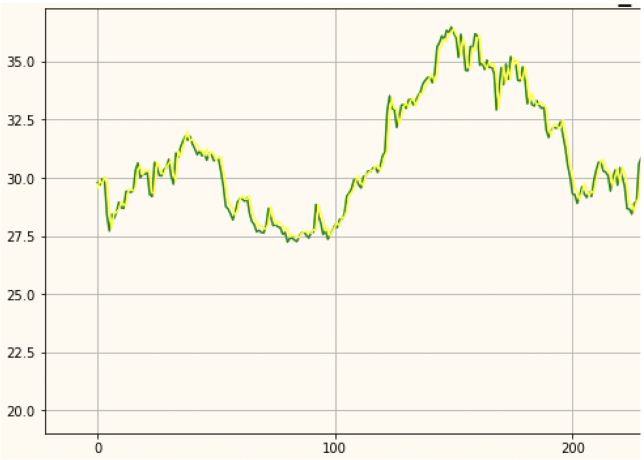


**Figure 10.** LSTM prediction vs HMC actual (three-day)

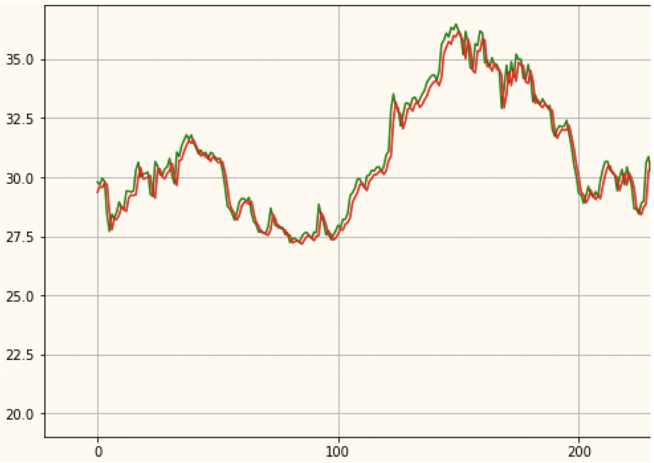


**Table 5.** Performance comparison of RNN and LSTM in three-day intervals (HMC)

| Module | R$^2$ |
|---|---|
| RNN | 0.96305 |

|  |  |
|---|---|
| LSTM | 0.96204 |

Figures 11 and 12 are graphs plotting the performance of each neural network module and the actual value; the results are summarised in table 6. Compared to their performance with previous intervals, they are performing worse, as predicted. This is because there are less data available for the two modules, thus less training can be done leading to less accurate predictions. Just like in three-day intervals, RNN is performing better than LSTM at predicting stock market prices.

**Figure 11.** RNN prediction vs HMC actual (five-day)

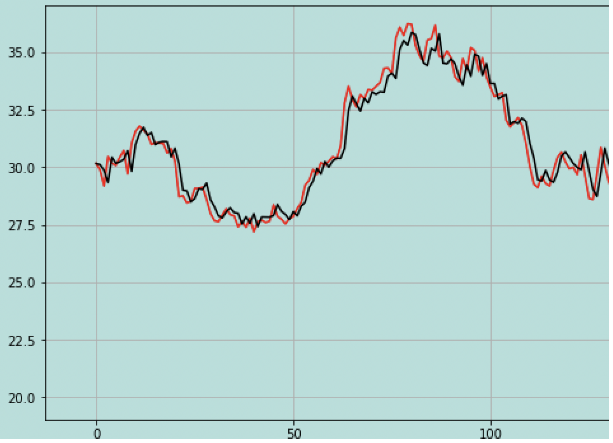


**Figure 12.** LSTM prediction vs HMC actual (five-day)

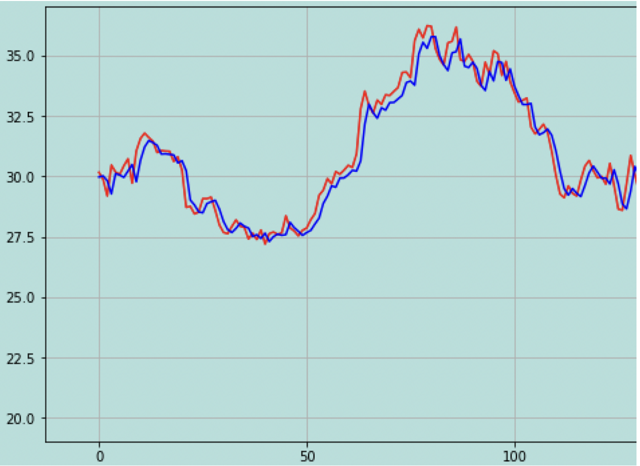


**Table 6.** Performance comparison of RNN and LSTM in five-day intervals (HMC)

| Module | $R^2$ |
|---|---|
| RNN | 0.94284 |
| LSTM | 0.94093 |

Overall, for slightly fluctuating stock, RNN would be the ideal neural network module as it is performing better than LSTM under conditions where data is given in three days and five days intervals.

## Performance evaluation of ORCL

Figures 13 and 14 are graphs plotting the performance of each neural network module and the actual value; the results are summarised in table 7. Both modules are performing well in the sense that they both have $R^2$ values extremely close to 1 (RNN is 0.96749; LSTM is 0.97345). However, the $R^2$ value of LSTM is much closer to 1 than RNN, this suggests that LSTM can make predictions on stock market prices more accurately than RNN if the data are given in daily intervals.

**Figure 13.** RNN prediction vs ORCL actual (one-day)

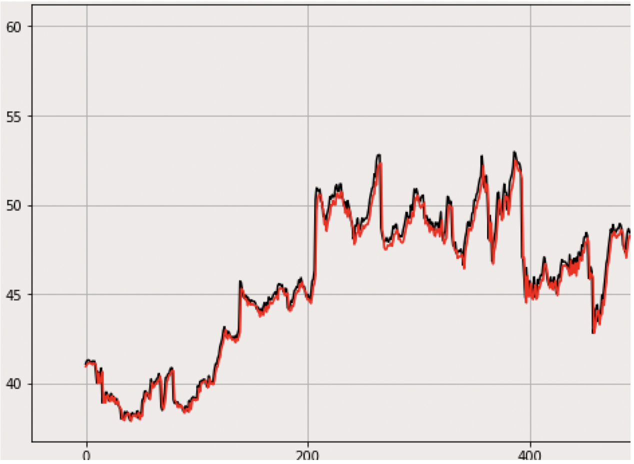


**Figure 14.** LSTM prediction vs ORCL actual (one-day)

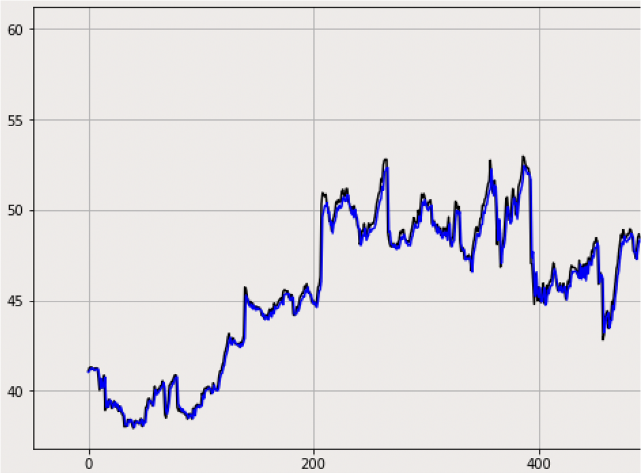


**Table 7.** Performance comparison of RNN and LSTM in one-day intervals (ORCL)

| Module | $R^2$ |
|---|---|
| RNN | 0.96749 |
| LSTM | 0.97345 |

Figures 15 and 16 are graphs plotting the performance of each neural network module and the actual value; the results are summarised in table 8. Compared to their performance for one-day intervals, there is a decrease in the accuracy of their performance as both modules have a decrease in $R^2$ value.

**Figure 15.** RNN prediction vs ORCL actual (three-day)

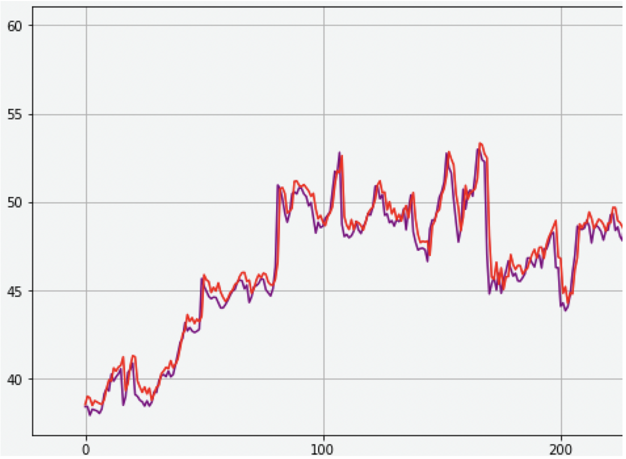


**Figure 16.** LSTM prediction vs ORCL actual (three-day)

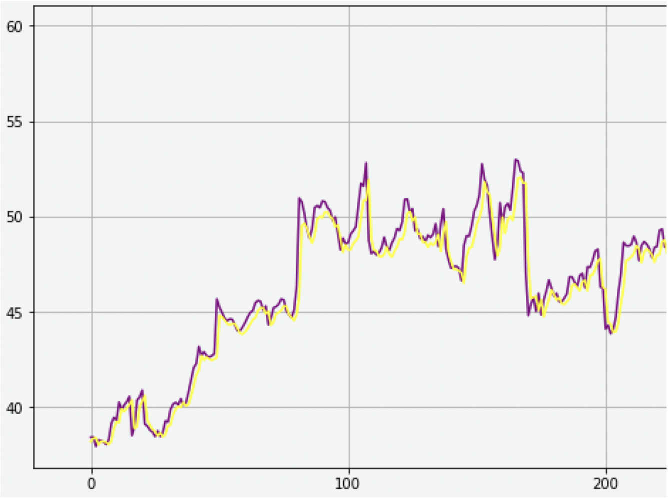


**Table 8.** Performance comparison of RNN and LSTM in three-day intervals (ORCL)

| Module | $R^2$ |
|---|---|
| RNN | 0.95136 |
| LSTM | 0.95364 |

Figures 17 and 18 are graphs plotting the performance of each neural network module and the actual value; the results are summarised in table 9. Compared to their performance with previous intervals, they are performing worse, as predicted. This is because there are less data available for the two modules, thus less training can be done leading to less accurate predictions.

**Figure 17.** RNN prediction vs ORCL actual (five-day)

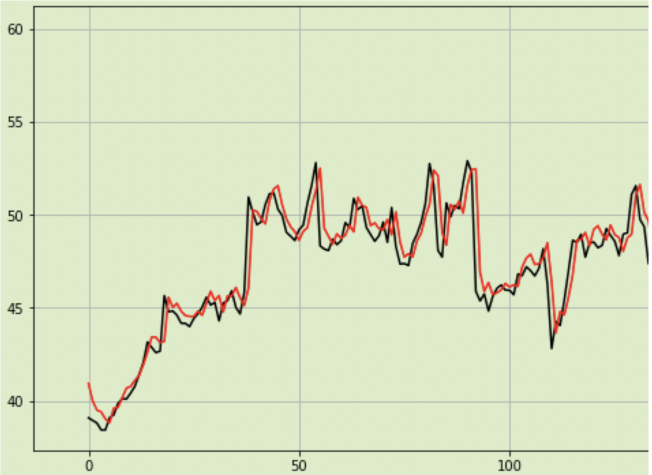


**Figure 18.** LSTM prediction vs ORCL actual (five-day)

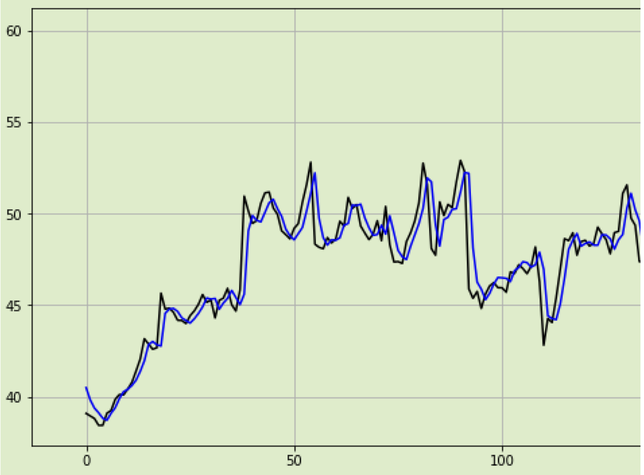


**Table 9.** Performance comparison of RNN and LSTM in five-day intervals (ORCL)

| Module | $R^2$ |
|---|---|
| RNN | 0.90040 |
| LSTM | 0.90153 |

Overall, for moderately fluctuating stock, LSTM outperforms RNN in all three intervals: one-day, three-day and five-day. This could be due to the difference in the architectures of LSTM and RNN modules. Due to higher fluctuation than HMC, this could result in problems in the training of RNN, leading to minimal learning progress.

## Performance evaluation of INTU

Figures 19 and 20 are graphs plotting the performance of each neural network module and the actual value; the results are summarised in table 10. Both modules are performing well in the sense that they both have $R^2$ values extremely close to 1 (RNN is 0.98555; LSTM is 0.98046). The $R^2$ value of RNN is much closer to 1 than LSTM, this suggests that RNN can make predictions on stock market prices more accurately than LSTM if the data are given in daily intervals.

**Figure 19.** RNN prediction vs INTU actual (one-day)

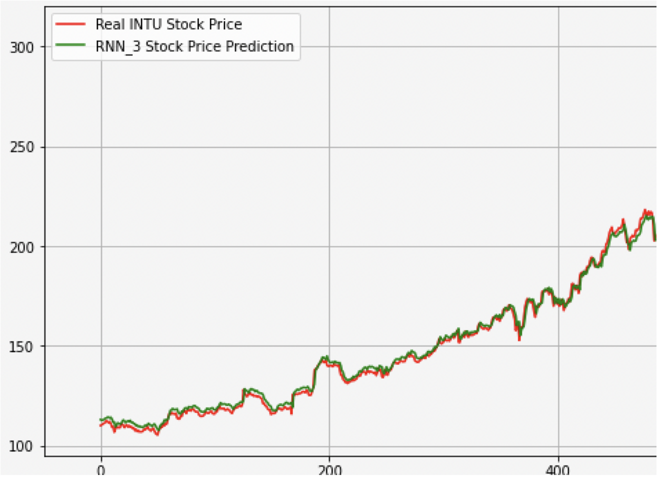


**Figure 20.** LSTM prediction vs INTU actual (one-day)

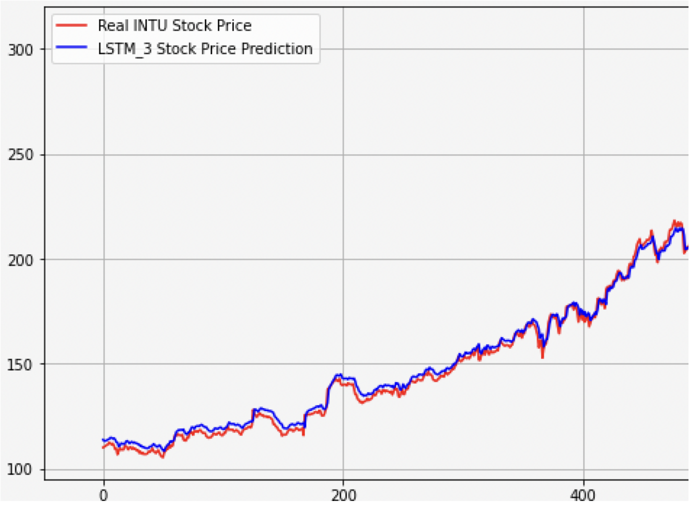


**Table 10.** Performance comparison of RNN and LSTM in one-day intervals (INTU)

| Module | $R^2$ |
|---|---|
| RNN | 0.98555 |
| LSTM | 0.98046 |

Figures 21 and 22 are graphs plotting the performance of each neural network module and the actual value; the results are summarised in table 11. Compared to their performance for one-day intervals, there is a decrease in the accuracy of their performance as both modules have a decrease in $R^2$ value. However, the $R^2$ value of LSTM is a lot larger than the $R^2$ value of RNN, which suggests that LSTM can make predictions on (high) prices more accurately if the data are given in three-day intervals.

**Figure 21.** RNN prediction vs INTU actual (three-day)

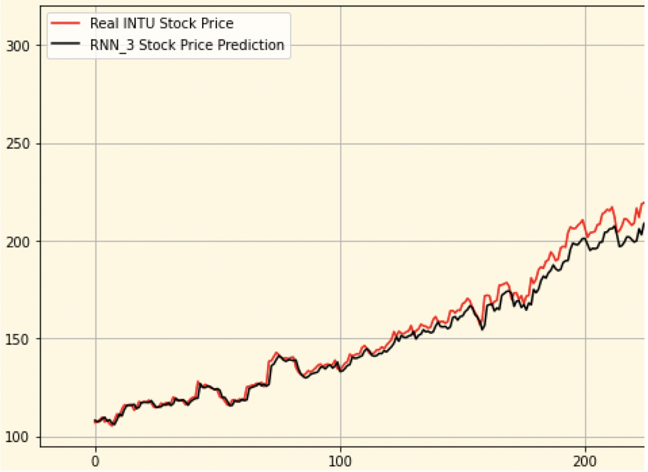


**Figure 22.** LSTM prediction vs INTU actual (three-day)

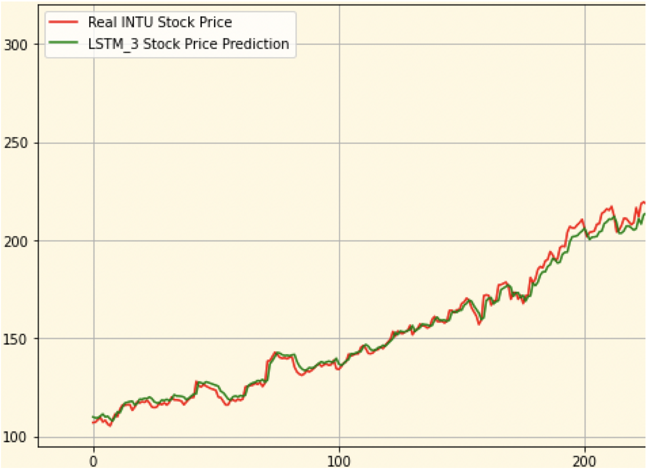


**Table 11.** Performance comparison of RNN and LSTM in three-day intervals (INTU)

| Module | $R^2$ |
|---|---|
| RNN | 0.94280 |
| LSTM | 0.97453 |

Figures 23 and 24 are graphs plotting the performance of each neural network module and the actual value; the results are summarised in table 12. Compared to their performance with previous intervals, they are performing worse, as predicted. This is because there are less data available for the two modules, thus less training can be done leading to less accurate predictions. Just like in three-day intervals, LSTM is performing a lot better than RNN at predicting stock market prices, with a less decrease in $R^2$ compared to RNN.

**Figure 23.** RNN prediction vs INTU actual (five-day)

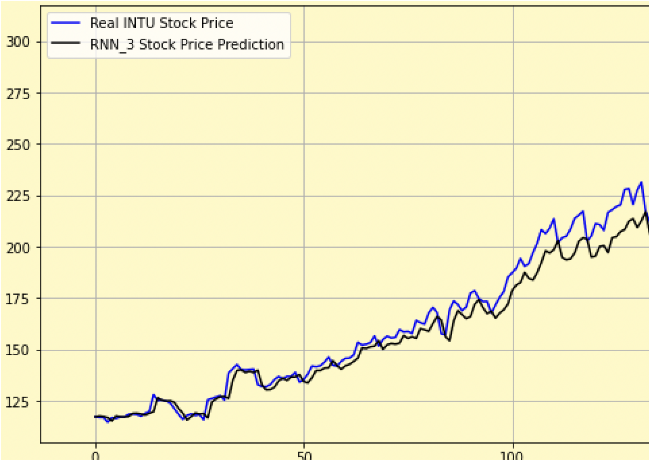


**Figure 24.** LSTM prediction vs INTU actual (five-day)

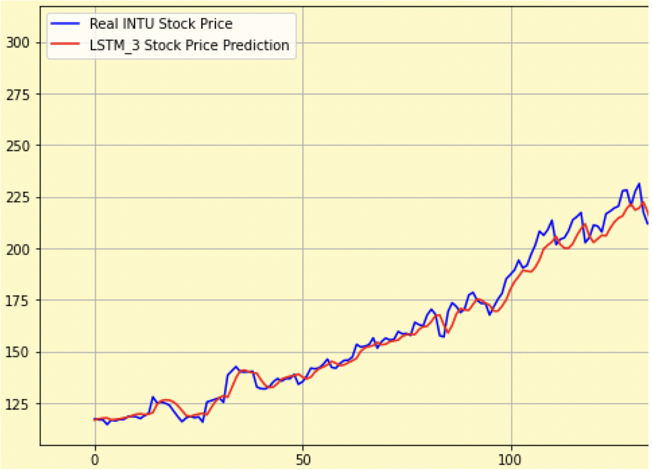


**Table 12.** Performance comparison of RNN and LSTM in five-day intervals (INTU)

| Module | $R^2$ |
|---|---|
| RNN | 0.90405 |
| LSTM | 0.96641 |

Overall, for highly fluctuating stock, LSTM would be the ideal neural network module as it is performing better than RNN under conditions where data is given in three days and five days intervals. However, based on the performance metrics, if companies can collect data in daily intervals, RNN would be the better choice as it outperformed LSTM for one-day intervals. However, judging by the huge difference in both three-day and five-day intervals, it is very likely that the chance of RNN performing better than LSTM is a coincidence, hence it is recommended that companies use highly fluctuating stock when making financial decisions.

# Conclusion

Portfolio managers and investors are constantly looking for new strategies to predict market movements and generate significant returns without putting too much effort in analysing trends manually. To satisfy their needs, researchers continue their study in the field of artificial intelligence, especially the signal processing part. In this extended essay, I evaluated the effectiveness of both RNN and LSTM architectures in time series analysis for the purpose of predicting future prices. Three stock types have been used for experiments: one stock's price changes very little, another's price changes little, and the third stock's price changes dramatically. In order to determine how both modules perform, three time intervals were taken into account. While RNN works slightly better when the data is mildly fluctuating, LSTM performs better when the pattern is neither too flat or too sharp and a lot better when it is substantially variable. Based on overall performance and the average fluctuations of stocks, LSTM will be a more suitable choice for portfolio managers and investors compared to RNN since RNN has a vanishing gradient problem as mentioned above, which is a common problem in moderately and highly fluctuating stocks. In the case of slightly fluctuating stocks, there is only a minor difference between the $R^2$ of the two modules, thereby resulting in negligible differences in the price movements of those kind of stocks. In addition, since these kinds of stocks will only change prices slightly over time, the probability of portfolio managers and investors losing huge amounts of money due to error in the prediction of LSTM modules will be a lot lower than them losing huge amounts of money due to error in the prediction of highly volatile stocks using RNN modules. As a result, investors should experiment with LSTM rather than traditional RNN modules when forecasting stock price movements.

# Works cited

Alam, Nafis, and Graham Kendall. “Are robots taking over the world's finance jobs?” *The Conversation*, 29 June 2017, https://theconversation.com/are-robots-taking-over-the-worlds-finance-jobs-77561. Accessed 22 January 2022.

Basodi, S., et al. “Gradient amplification: An efficient way to train deep neural networks.” *IEEE Xplore Full-Text: PDF*, in Big Data Mining and Analytics, vol. 3, no. 3, pp. 196-207, September 2020, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9142152. Accessed 20 June 2022.

Brownlee, Jason. “How to Choose an Activation Function for Deep Learning.” *Machine Learning Mastery*, 18 January 2021, https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/. Accessed 22 June 2022.

Copeland, BJ. “artificial intelligence | Definition, Examples, Types, Applications, Companies, & Facts.” *Encyclopedia Britannica*, https://www.britannica.com/technology/artificial-intelligence. Accessed 20 June 2022.

“Exploding Gradient Problem Definition.” *DeepAI*, https://deepai.org/machine-learning-glossary-and-terms/exploding-gradient-problem. Accessed 10 September 2022.

Fernando, Jason. “R-Squared Formula, Regression, and Interpretations.” *Investopedia*, https://www.investopedia.com/terms/r/r-squared.asp. Accessed 13 November 2022.

Gad, Ahmed. “Beginners Ask “How Many Hidden Layers/Neurons to Use in Artificial Neural Networks?”” *Towards Data Science*, 27 June 2018, https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e. Accessed 22 June 2022.

“Historical Volatility (HV) Calculator.” *Good Calculators*, https://goodcalculators.com/historical-volatility-calculator/. Accessed 29 September 2022.

Hochreiter, Sepp. “Long Short-term Memory.” *nc.dvi - lstm.pdf*, 3 April 2016, https://www.researchgate.net/profile/Sepp-Hochreiter/publication/13853244_Long_Short-term_Memory/links/5700e75608aea6b7746a0624/Long-Short-term-Memory.pdf?origin=publication_detail. Accessed 20 June 2022.

Kang, Eugine. “Long Short-Term Memory (LSTM): Concept | by Eugine Kang.” *Medium*, https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359. Accessed 10 September 2022.

Khurana, Parveen. “How LSTMs solve the problem of Vanishing Gradients?” *Parveen Khurana*, https://prvnk10.medium.com/how-lstms-solve-the-problem-of-vanishing-gradients-ea88f08c78ca. Accessed 11 September 2022.

Mehta, Sher. “Types of Portfolio Risks - Financial Edge.” *Financial Edge Training*, 8 July 2021, https://www.fe.training/free-resources/portfolio-management/types-of-portfolio-risks/. Accessed 29 September 2022.

Sak, Has¸im, et al. “1402.1128v1 [cs.NE] 5 Feb 2014.” *arXiv*, 5 February 2014, https://arxiv.org/pdf/1402.1128.pdf. Accessed 29 September 2022.

Shiriram, S., et al. “Future Stock Price Prediction using Recurrent Neural Network, LSTM and Machine Learning – IJERT.” *IJERT*, 1 April 2021, https://www.ijert.org/future-stock-price-prediction-using-recurrent-neural-network-lstm-and-machine-learning. Accessed 26 February 2022.

Singhal, Gaurav. “Introduction to LSTM Units in RNN.” *Pluralsight*, 9 September 2020, https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn. Accessed 21 June 2022.

“Weights and Biases - AI Wiki.” *Paperspace*, July 2021, https://machine-learning.paperspace.com/wiki/weights-and-biases. Accessed 22 June 2022.