Comparison of LSTM and Standard RNN Network on S&P 500 Index Time Series

Written by: Chit Lun, Chow

Abstracts

The experiment is carried out to compare the LSTM and RNN architectures with time series data that carries seasonality, trend, and irregular pattern. The LSTM has shown great prediction power when trying to predict the S&P 500 index during the time in pandemic which shows instabilities. RNN demonstrated its outstanding computational performance for fast training with losing too much prediction performances.

1. Introduction

The Standard & Poor 500 index is a stock market index measured 500 large publicly traded companies in the US introduced in 1957. It provides a reference as a benchmark performance for the US stock market and is sometimes used as a point of reference by investors and analysts. It is also an economic indicator as it reflects the trends of the market and the prospect of the economy. As of its nature and economical meaning, investors might be interested in understanding the economic performance of the market to inform and support any investment decisions. However, [1] has shown that predicting an economical variable with traditional techniques like GLM has proven to be challenging because of their market volatility. [2] Proved that traditional methods like GARCH and ARIMA do give a short-term period forecast, but that is required for dynamical forecasting as time goes on. Hence, the development of a recurrent neural network, which can train with more data and has a memory effect for inferencing, has displayed an opportunity that tackles the problem of long-term forecasting, especially dealing with high-frequency data such as daily stock price.

1.1 Dataset and Initial Analysis

The dataset chosen is a time series data of the Standard & Poor 500 index closing price. With an initial investigation of the data, there's an upward trend in the series. In general, we must employ some general understanding of time series analysis for further exploration and might be required to manipulate the series data to make the prediction accurate.

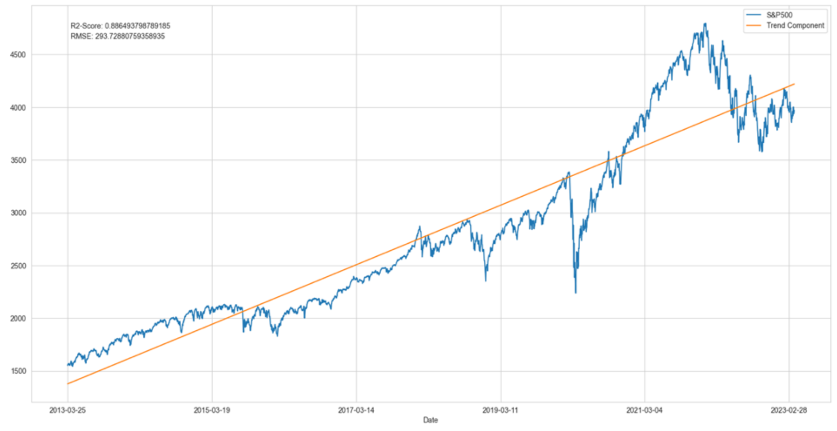

Fig.1 The time series plot and the autocorrelation from lag 1 to 25

Most of the time series possessed some fundamental patterns – including seasonality, trend, and irregular patterns. In the above figure, the orange line represents the overall upward trend of the S&P 500 index concerning time ($R^2 = 0.8865$). To further discuss the component of the time series, it is easy to analyze by decomposing it as a multiplicative model with a period of 252 days (average number of trading days in a year.

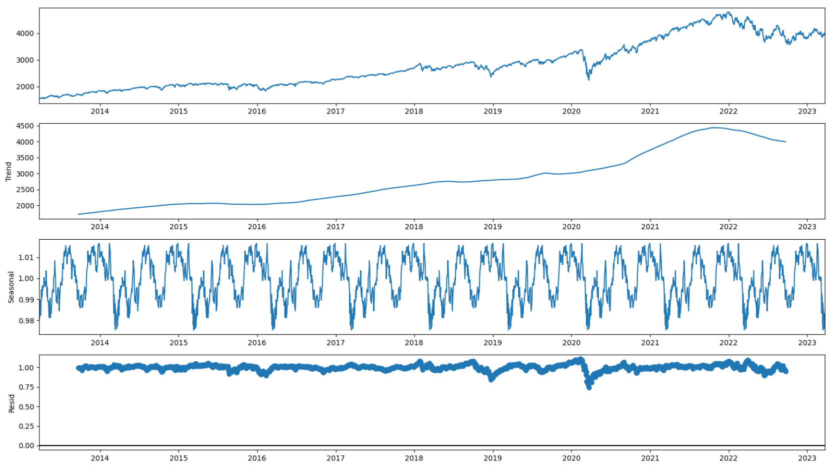

Fig 2. Trend, seasonal and residual component of S&P 500 time series

The above figure shows the result of the decomposed time series with its components, trend, seasonal, and irregular patterns. The figure shows the trend, which starts very stably before 2020. It starts to have a sudden change during the pandemic time which occurs around the middle of 2020. Mentioning the pandemic, we can also see the uncertainty in early 2020 has a big fluctuation. Overall, the seasonality of the S&P 500 index shows a regular patter over the 10 years.

1.2 Choice of Methods

The problem here is a regression problem dealing with sequential data. Therefore, the methods to be used must be able to detect the seasonality and change of trend. The methods employed to solve the problem of time series prediction are Recurrent Neural Network (RNN) and Long-Short Term Memory (LSTM). Both candidates possess similar structures, but LSTM has a forgetting mechanism that helps in memorizing patterns shown in sequential data.

1.2.1 Recurrent Neural Network

The Recurrent Neural Network is an artificial neural network that suits for processing sequential data. In contrast to the other types of neural networks like CNN, it handles sequential data that are variable in length. With one of its features called hidden states, it can maintain the information of the internal state of the neurons and pass it to the next timestep for processing. The hidden state act as a kind of memory, that is beneficial when trying to make a prediction based on a history sequence, for example, predict the next word based on a given sentence. However, the RNN must face the risk of having vanished gradient which will make the networks' parameter forget information from earlier inputs.

1.2.2 Long-Short Term Memory (LSTM)

The LSTM is a variant of the RNN that has a modified hidden state set to overcome the problem of the RNN. The LSTM unit has equipped with an internal memory cell that can store information for a long period. This special cell is composed of several control gates that direct the flow of information into and out of the cell.

2. Methodology

The experiment is designed for a fair comparison of the two methods model and their optimized version. Before the start of training, the data first split into 80% of training and 20% of testing with a sliding window of 100 days looking backward. The testing set won't be seen by the model until training ends.

In the first stage, both candidates will be using a set of baseline parameters, which use the Adam optimizer with a learning rate of 0.01, 1 hidden layer with 10 hidden neurons, to train for 2000 epochs. During the training process, there will be a checkpoint at every 100 epochs to test the model over the validation set, which is 20% of the training set data. Metrics including training time, mean-squared error, and R-squared value will be used for performance comparison. It is also important to have a look at the actual prediction of the original data, so the predictions would be visualized alongside the original data for comparison. This helps in understanding both strengths and weaknesses of the model we've built.

In the second stage, the models are then undergoing a model selection process by using a grid search cross-validation approach. The aim here is to find out the best set of parameters that could maximize the model performance. The cross-validation will search through the settings of the learning rate, the number of hidden layers, and its size. Each of the parameters will have the value set for running the cross-validation:

Parameter	**Value Set**
Number of Layers	1, 2, 3
Dimension of Hidden Layers	10, 20, 50
Learning rate of optimizer	0.1, 0.01, 0.001

Table 1. Parameters Choice

As a result, this stage would produce $3^3 = 27$ models with different parameter settings and are scored by negative mean squared error for comparison. We will then look at the top 10 sets of parameters that have achieved the best score. Each method will then use the best parameters that have been derived by the cross-validation to train a final model for the last competition. Again, it is trained on the 80% training data and will be tested on the testing set.

Finally, the experiment results would be analyzed in both quantitative and qualitative ways. Make a note that performance metrics are not the absolute indicator of seeing whether a model performs well or not, it helps to see the overall picture of their strengths and weaknesses. Therefore, by the end of the experiment, would show a better picture of how these methods would be more suitable to use in their best situations.

3. Results, Findings and Evaluations

3.1 Baseline Models

In terms of computational performance, the RNN method used 47 seconds while LSTM uses 2 mins and 49 seconds, which is about 3.6 times more than the RNN use. The LSTM potentially needs longer training time than RNN needs because of the additional gated mechanism which helps to retain information in the past. Using a GPU could potentially help reduce the difference between the two, but for the compatibility issue with Skorch, CPU runtime is used in this experiment. In terms of prediction, both models have similar predictions. Looking at the whole testing data, both models tend to underestimate the true value. The challenge here is to have accurate prediction over the peak occurring in time step 100 of the testing set, which in terms of date, is around the middle or quarter of the year 2020. After this period, both models perform well for estimating the index with it starts to fall until the year 2023.

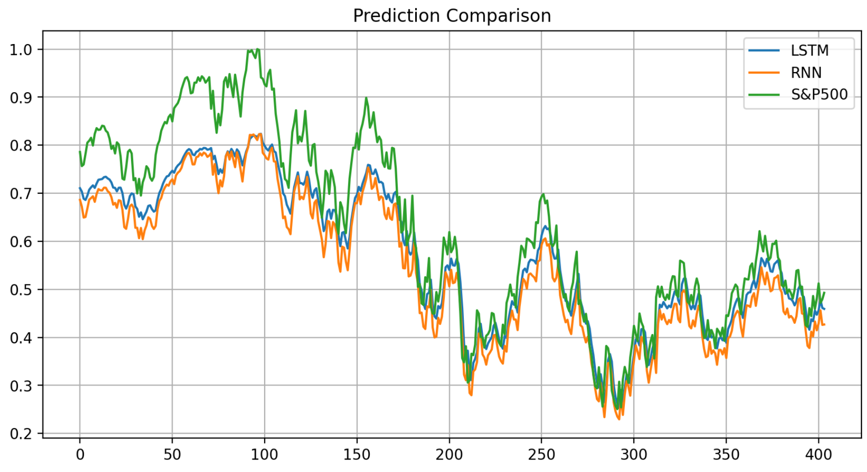

Fig 3. Models Prediction compared to original values.

Baseline Model	R^2	MSE	$RMSE$
RNN	0.59331	0.009517	0.09756
LSTM	0.72047	0.006194	0.07870

Table 2. Baseline Models performances

Although it looks like RNN has a similar performance as the LSTM, it only has R2 = 0.5933, and RMSE = 0.09756. The model has struggled to predict the peak of the data. The prediction of the RNN is mostly less than what LSTM estimated and is the reason why it has a lower R-squared value and higher error. Although the LSTM also plays a strategy of underestimation, it can make a closer prediction than RNN does. As a result, it has shown better performance in the metrics.

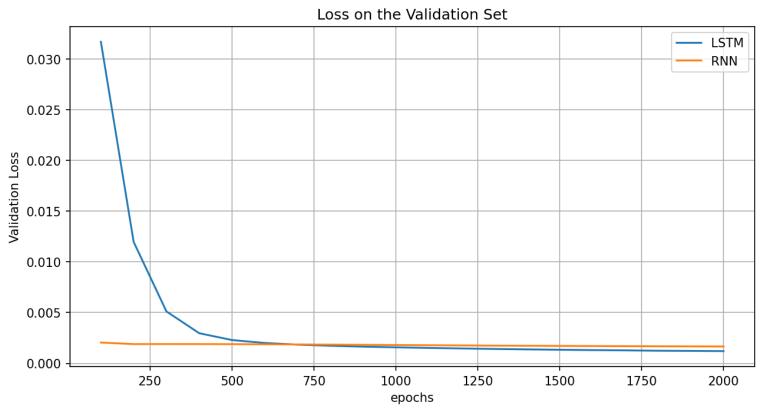

Fig 4. Validation Loss of the models

Looking into the validation loss of each model, we can conclude that the learning styles are different for the two models. The RNN learns very fast, resulting in a fast-dropping validation loss, but that doesn't improve along the 2000 epochs training. This could have two implications: the training can be stopped earlier to prevent overfitting; it might need a slower learning rate when searching for the global minima. In contrast, the LSTM prefers to learn slowly and can reach a smaller error of prediction. In the beginning, the LSTM don't mind having a higher learning rate and it gradually learned and was able to exceed RNN performance after around 750 epochs of training.

3.2 Hyperparameters Selection

The model selection process is first based on the search for parameters for maximizing the performance of prediction. The following tables show the parameters selection results done by 3-fold cross-validation with their mean test score measured in negative mean-squared error and the mean of time for fitting the model.

	Learning Rate	**Hidden Layer Size**	**Hidden Layers Number**	**Mean test score**	**Mean fit time**
1	0.01	20	1	-0.028361	1.581634
2	0.01	50	2	-0.070112	5.812397
3	0.1	50	1	-0.089113	2.861312
4	0.001	50	1	-0.092243	2.835557
5	0.01	20	2	-0.098524	3.028329

Table 3. Top 5 sets of parameters for LSTM

	Learning Rate	**Hidden Layer Size**	**Hidden Layers Number**	**Mean test score**	**Mean fit time**
1	0.001	50	2	-0.003894	1.517669
2	0.001	20	2	-0.012016	0.934585
3	0.001	50	1	-0.018042	0.789092
4	0.01	10	1	-0.019182	0.373473
5	0.01	20	1	-0.030127	0.479589

Table 4. Top 5 sets of parameters for RNN

The cross-validation grid search provides insight into choosing the best parameters for each model. From the above results, we can draw some conclusions for evaluating the computational performance of the models. For their best parameters, the strategy for improving performance is different. The LSTM increased the number of hidden layer sizes only, while the RNN tried to maximize the ability of prediction by increasing both hidden layer size and the number of hidden layers. Although the assumption here is that more neurons imply longer training time, the RNN with 2 hidden layers with 50 neurons each has a slightly faster fitting time than the best candidate of the LSTM, which only has 1 layer of 20 neurons. Mentioning the training time, the LSTM with 2 layers of 50 hidden neurons has the longest training time, which is close to 6 seconds, but the more complex structure cannot rival the simpler model in terms of the test score. In addition, the RNN can achieve a better score by lowering the learning rate to 0.001. As mentioned in the previous session, the RNN have the ability to learn fast, but lowering the learning rate could help increase the chance of reaching global minima.

3.3 Final Model

With the insights drawn from the model selection, the best parameters are used to generate 2 best models by assumption for the final comparison.

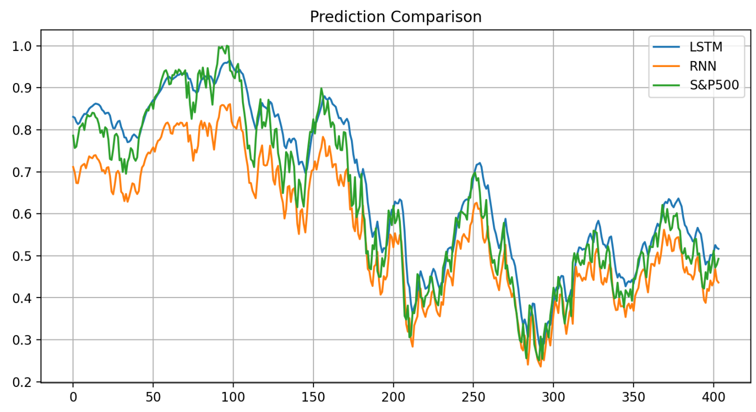

Fig 5. Prediction Comparison of the final model

	R^2_{base}	$R^2_{improved}$	Improvement	$RMSE_{baseline}$	$RMSE_{improved}$	% difference
RNN	0.59331	0.78633	+32.53%	0.09756	0.07574	-22.37%
LSTM	0.72047	0.90583	+25.73%	0.07870	0.05720	-27.32%

Table 5. Performance metrics of final model

Looking at the prediction of the final model, it is clearly shown LSTM's prediction (blue line) can catch-up to the actual value. Something surprising here is that the model tried to "smooth" out the prediction like moving averaged does, instead of trying hard to go close to the true value. Therefore, it can look like predicting by capturing the recent trends. Having an observation at the beginning period, one can see that it can overpredict or underpredict, depending on how the trend is going. Its strategy also benefits its prediction performance as it achieved = 0.90583, the highest score among other competitors. It also achieved a significantly lower RMSE at 0.05720.

Although RNN looks fair in terms of performance, there are still things worth mentioning. The RNN cannot catch up with the actual value in the peak period of the series and shows a significant difference compared to its variant. However, the improvement made by increasing both the number of neurons and the number of layers was more prominent than the optimized LSTM. Comparatively, it requires multiple adjustments to catch up with the performance of the baseline LSTM model with only a layer of 10 hidden neurons, which in terms of model complexity, RNN would need a more complex model to compete with LSTM. Nonetheless, the improvement in terms of minimizing RMSE loss wasn't as good as LSTM does, meaning that the RNN still has a barrier to reaching the minima. This can potentially reveal the bottleneck that an RNN model could be facing when fitting sequential data.

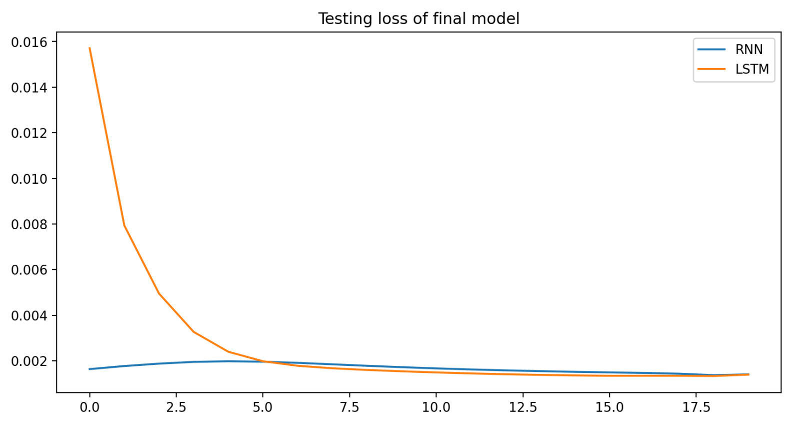

Fig 6. Testing loss of the final models

Something also worth mentioning about RNN is that, given that a lower learning rate has been used, the fast-learning issues mentioned earlier don't resolve. Although the learning rate was lowered by 10 times, the optimizer still converged quickly and don't have significant fluctuations.

Concerning computational performance, the RNN has shown a significant advantage over LSTM. The LSTM uses a total of 6 mins and 10 seconds to train a full model, while in contrast, RNN uses only 1 minute and 35 seconds. The result has clearly shown that RNN was able to compute fast and given the model, complexity has increased, since each RNN unit is just a single neuron passing hidden states.

4. Conclusions, Reflections and Future Works

4.1 Conclusion

Although LSTM and RNN have similarities in terms of structures, however, they can be different in some ways. Considering the prediction results in the experiment, the LSTM has demonstrated its ability for accurate prediction while maintaining the model complexity (number of hidden layers and neurons in each layer). It also possesses the ability to understand trends of the sequential data, which can be achieved by trying to smoothen out the prediction for future prediction. This can be a rather conservative approach which helps to have a better outlook on future timesteps. However, this comes with a tradeoff of computational performance where a longer training time is needed. On the opposite side, RNN gained the advantage of faster computational time, which is only a fraction of the time that LSTM uses, while sacrificing the overall prediction power. The results here imply 2 things: 1) a model that requires precise predictions is better to use LSTM for the inference engine, and 2) cases that require fast training and prediction can utilize RNN without loss of too much accuracy. Applications like voice recognition are better with LSTM while RNN can be used for some online learning application like heart rate anomaly detection.

4.2 Reflections

The experiment was carried out in two phases. However, there are still plenty of rooms of improvements that can be made. First, the experiment only considered 100 days look back period because of the hardware performance bottleneck. It is worth trying for longer period of looking back, say one year, to see how models are going to perform against each other. Also, due to compatibility issue, only CPU is used for the model selection. Having a GPU would potentially help reduce the computational advantage of RNN. Finally, the choice of the optimizer can vary. However, the model performance was sensitive and hence stayed with Adam.

4.3 Future Works

The advantages of RNN and LSTM have purposed to serve different scenarios. It is also worth trying to compare long-term and short-term predictions. RNN's ability might be useful for online learning tasks like heart rate anomalies detection, which is able to detect abnormal high heart rate.

Bibliography

[1] S. Gonçalves and M. Guidolin, “Predictable Dynamics in the S&P 500 Index Options Implied Volatility Surface*,” *The Journal of Business*, vol. 79, no. 3, pp. 1591–1635, May 2006, doi: https://doi.org/10.1086/500686.

[2] T. S. Kvainickas and J. Stankevičienė, “Regional Limitations of Stock Indices Prediction Models Based on Macroeconomic Variables,” *Economics and Culture*, vol. 16, no. 2, pp. 5–20, Dec. 2019, doi: https://doi.org/10.2478/jec-2019-0018.