Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 167 (2020) 2091–2100

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2019)

Optimizing LSTM for time series prediction in Indian stock market

Anita Yadav[a,*], C K Jha[a], Aditi Sharan[b]

[a]Department of Computer Science, AIM & ACT, Banasthali Vidyapith, Rajasthan- 304022, India
[b]School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi - 110067, India

Abstract

Long Short Term Memory (LSTM) is among the most popular deep learning models used today. It is also being applied to time series prediction which is a particularly hard problem to solve due to the presence of long term trend, seasonal and cyclical fluctuations and random noise. The performance of LSTM is highly dependent on choice of several hyper-parameters which need to be chosen very carefully, in order to get good results. Being a relatively new model, there are no established guidelines for configuring LSTM. In this paper this research gap was addressed. A dataset was created from the Indian stock market and an LSTM model was developed for it. It was then optimized by comparing stateless and stateful models and by tuning for the number of hidden layers.

Keywords: LSTM; Hyperparameters; Stateful Stateless; Hidden layers; Time series prediction

1. Introduction

Stock price prediction is the process by which future stock prices are forecast on the basis of past prices. Stock price prediction is useful for investors to increase the profits from stock trading. There are two traditional approaches to prediction: technical and fundamental analysis. Technical analysis tries to identify some patterns from historical data while fundamental analysis focuses on overall economy, the company's financial condition and its management [1]. Stock price prediction is a very challenging task as it is highly volatile, non-linear and dynamic and is impacted by various factors like political conditions, economic situation, trend, seasonality, investor psychology etc.

* Corresponding author. Tel.: +91-9899782208.
E-mail address: anitayadav_07@hotmail.com

Nomenclature

ANN	Artificial Neural Network
DL	Deep Learning
KNN	K-nearest neighbors
LSTM	Long Short Term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
NB	Naive Bayes
NLP	Natural Language Processing
NSE	National Stock Exchange
RAF	Random Forest
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SVM	Support Vector Machine

Due to information explosion in recent years, a large amount of data has become available online and manually analyzing this data is not feasible. Various supervised and unsupervised machine learning techniques have been developed to automatically gather and analyze data and make predictions. Deep learning techniques have become the hottest tool delivering state-of-the art results in areas as diverse as computer vision, speech recognition and NLP problems. But deep learning techniques also present a unique set of challenges.

The performance of deep learning models depends upon a number of parameters called hyper-parameters. Hyper-parameters are manually set training variables which are determined before training the model. Some automatic techniques for tuning hyper-parameters have been proposed but these methods don't provide insight to the end-users about the interactions between different hyper-parameters and their relative importance [2]. For the deep learning model used in this paper, namely LSTM, the important hyper-parameters are [3] activation function (sigmoid, tanh, softmax etc.), optimizer (Adam, Adadelta, RMSprop etc.), batch size, number of epochs, number of hidden layers etc. LSTM can be stateful or stateless. By creating a dataset of stock prices from the Indian stock market and building an LSTM model to predict stock prices, various configurations of LSTM can be tested and compared to optimize the model.

2. Literature Review

Neural networks have been widely used for forecasting time series. Foster et al. [4] compared the forecasting of noisy time series using neural networks with linear regression and exponential smoothing methods. Initially the forecast accuracy of linear regression was better. The forecast accuracy of neural networks improved after removing seasonal variation from the series. [5] proposed the use of independent component analysis and support vector regression using two stage modeling in forecasting financial time series. It was opined that the most important challenge of working with financial time series was the presence of noise and the need to make them stationary.

Hamzacebi et al. [6] used artificial neural network for multi periodic forecasting using iterative and direct methods and compared their results using grey relational analysis. [7] applied deep neural networks (DNNs) for prediction of financial market. The trading strategy was back-tested on commodity and futures markets. [8] used data from France metropolitan's electricity consumption data for electric load forecasting. The number of hidden neural layers and optimal time lags were tuned for LSTM using genetic algorithm. It was found to give better performance as compared to machine learning models.

[9] used genetic algorithms to optimize technical analysis parameters. These are then passed on to a deep neural network for predictions. Enhanced stock trading performance was reported. [10] used a special type of artificial neural network called CEFLANN. Stock trading prediction was modeled as a classification problem and model performance was compared with other ML techniques like SVM, NB, KNN and DT. [11] used LSTM for out-of-

sample prediction of financial time series. It was found to outperform logistic regression, DNN and RAF. A short term trading strategy was devised based on the LSTM outputs.

[12] compared LSTM, RNN, CNN and MLP with linear and non-linear regression models. The networks were trained with one company from NSE and then tested on five companies from NSE and NYSE. CNN was found to outperform other models. [13] was one of the early papers that addressed the question of how many hidden layers should be there in a neural network. Using remote sensing data, a near optimal solution was discovered with genetic algorithm. [14] evaluated eight LSTM variants on three tasks: music modeling, handwriting recognition and speech recognition. It was found that none of the variants significantly outperformed the basic LSTM architecture and that the hyper-parameters were virtually independent. [15] compared LSTM with SVM, backpropagation and Kalman filter for stock market for different number of epochs varying from 10 to 100. LSTM was found to have high accuracy and low variance. [16] compared manual search and grid search techniques of hyper-parameter optimization for neural networks and deep belief networks. It was found that the former is able to find as good or even better models than the latter within lesser computation time. It was suggested that manual search be used as a baseline for judging hyper-parameter optimization algorithms.

3. Long Short Term Memory (LSTM)

LSTM is a type of recurrent neural network (RNN). RNNs are a powerful type of artificial neural network that can internally maintain memory of the input. This makes them particularly suited for solving problems involving sequential data like a time series. However RNNs frequently suffer from a problem called vanishing gradient which leads to the model learning becoming too slow or stopping altogether. LSTMs were created in the 1990's to solve this problem. LSTMs have longer memories and can learn from inputs that are separated from each other by long time lags.

An LSTM has three gates: an input gate which determines whether or not to let the new input in, a forget gate which deletes information that is not important and an output gate which decides what information to output. These three gates are analog gates based on the sigmoid function which works on the range 0 to 1. These three sigmoid gates can be seen in Fig. 1 below. A horizontal line that can be seen running through the cell represents the cell state.

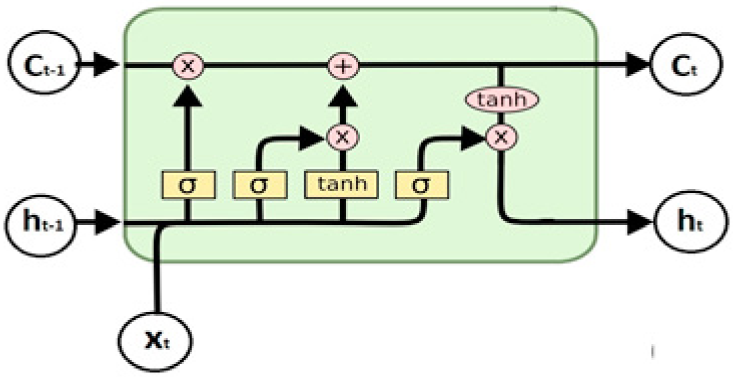

Fig. 1. LSTM Architecture [17]

3.1. Proposed System

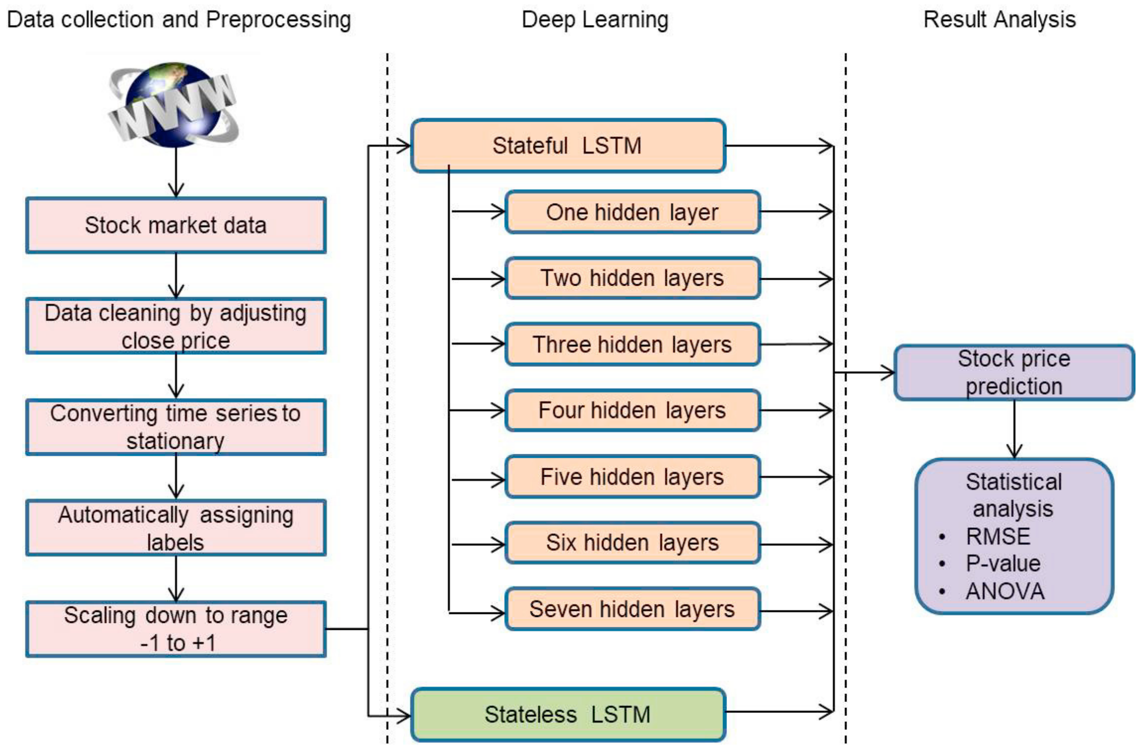

Fig. 2. Main steps of the proposed model

3.2. Tuning Hyper-parameters

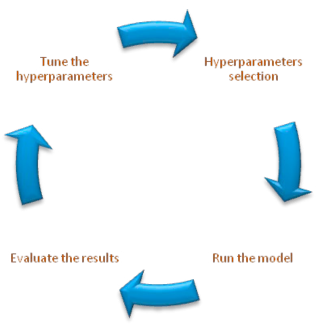

Fig. 3. Hyper-parameter tuning process

Fig. 3 shows a typical development cycle for any neural network. For large neural nets, training times can be long so repeatedly running the network and evaluating the results can take days or weeks. For faster development of neural network models, it is important to explore hyper-parameter tuning and develop recommendations that can provide a good initial starting point for the kind of network being developed.

4. Experimental Setup

4.1. Selection of companies

The most widely used benchmark for stocks in India is the NIFTY 50. This is provided by the National Stock Exchange (NSE) which is the largest Indian stock exchange. As the name signifies, NIFTY 50 is made up of fifty stocks. For this paper, stocks were picked from different sectors so that LSTM performance could be viewed on time series that were influenced by different factors. The sector with the highest weightage of 37.94% in NIFTY is financial services. Out of this ICICI Bank, which is the largest private sector bank in India, was selected. The sector with the next highest weightage of 15.31% is Energy. Here Reliance, which is the first Indian company to reach $100 billion in market capitalization, was chosen. The next sector is IT with a weightage of 14.46%. The biggest company here is Tata Consultancy services Limited (TCS) which was chosen. The fourth biggest sector is Automobiles where again the biggest company, Maruti, was chosen. The stock price data for these companies was downloaded from the World Wide Web as shown in Fig. 2.

Keeping in mind easier allocation of batch size later while configuring LSTM, data was extracted for these four companies for 2560 days for the period 29-12-2008 to 24-05-2019 which is over ten years of data. An 80:20 ratio was implemented for training and testing data, yielding 512 days of testing.

4.2. Adjustment of closing price

The last quoted price for a stock during a day is called its closing price. This is the standard price considered during financial time series analysis. This, however, needs to be adjusted for corporate actions before doing any analysis on historical data. These actions could be stock splits, dividends or rights offerings. All of these affect the "nominal" price of the stock though none of them affects its "true" price. For proper training of the LSTM, the price effect of these actions needs to be removed from the nominal stock price. The closing price after these adjustments is called the adjusted close price and has been used in our analysis.

4.3. Setting up the environment

Python 3.6.8 was used as the programming environment. A virtual environment was created to run the experiments for this paper. In this virtual environment, the following packages were installed:

- Tensorflow 1.13.1
- Keras 2.2.4-tf
- Pandas 0.24.2
- Sklearn 0.21.1
- Numpy 1.16.3
- Matplotlib 3.0.3

4.4. Data Preparation

Before applying time series techniques, the data series needs to be made stationary. This means that it should be a flat looking series without any long term trend or seasonality. A common method to do this is differencing.

$$Y_i = Z_i - Z_{i-1} \tag{1}$$

The resulting series has one element less because the first item has nothing to difference it from and has to be skipped. For this reason 2560+1 data points were included in the dataset. After forecasting through LSTM this difference has to be added back to get the original series.

The next step is to add an output component to the data. LSTM assumes that there are input values (time series) which are to be used to predict an output value. Since the time series data only had an input series, the stock price value from time t-1 was used as input for predicting the stock price value from time t as the output. A new input series was created by pushing the values on the original series back by one time period. The original times series contains the output variables.

For the LSTM default activation function, hyperbolic tangent (tanh) was used, whose output values range between -1 and 1. The input data was scaled down to fit it within the range of the activation function. Again, once the forecasts have been made, the forecasts must be scaled back up to the original scale.

4.5. LSTM Model

Experiment 1: Data is fed to LSTM in batches. Batch size is decided by the user (64 in our case) and defines the number of rows from the dataset processed by the model before it updates its weights. While a batch is being processed, LSTM maintains its state. Between batches, the state can be maintained or cleared. By default, the state is cleared. This is called "Stateless." In our first experiment this was compared with the other option of maintaining state called "Stateful" to see which one yields better results.

Experiment 2: An important hyper-parameter in configuring LSTM is the number of neural layers. There are three types of these layers in any LSTM – input layer, output layer and hidden layer. Out of these, input and output have a single layer and this number is not configurable. The number of hidden layers can be configured. In the second experiment LSTM was run with hidden layers varying from one to seven and the performance was compared. This has been done for two of the companies – ICICI and TCS.

In both experiments, "mean_squared_error" was used as the LSTM loss function and ADAM as the optimization algorithm. Root Mean Squared Error (RMSE) was used to evaluate the performance of the various models. A batch size of 64 was used for training and 1 for testing. The number of epochs was fixed at 30 and the tests were repeated 30 times to ensure consistency.

5. Results and Analysis

The input data was visualized by plotting a combined line plot for the four companies. It is shown in Fig. 4.

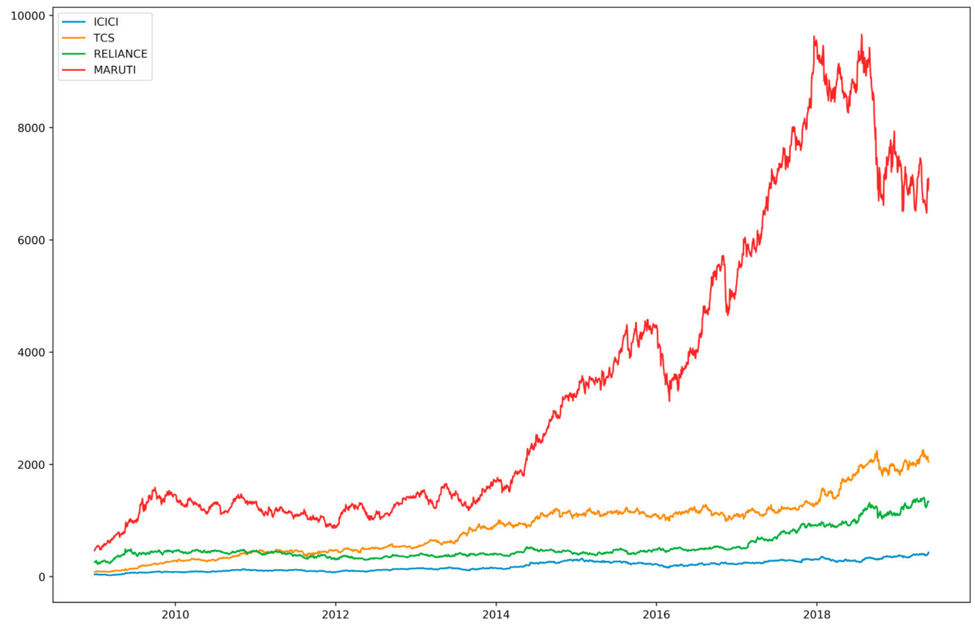

Fig. 4. Combined plot of input data for four companies

Experiment 1: The input data was run through the LSTM and predictions were made on the test dataset. This was done for Stateless and Stateful LSTM. The results of 30 repetitions were then statistically analyzed. These are presented below where results for stateful are labelled A and those for stateless are labelled B.

Table 1. Comparison of Stateful (A) and Stateless (B) LSTM

	ICICI-A	ICICI-B	TCS-A	TCS-B	RELIANCE-A	RELIANCE-B	MARUTI-A	MARUTI-B
count	30	30	30	30	30	30	30	30
mean	5.851305	5.852057	25.596914	25.531364	17.00645	16.789178	112.526895	112.428441
std	0.026454	0.024332	0.247032	0.162271	0.298829	0.147629	0.383043	0.347485
min	5.824255	5.818488	25.374967	25.396562	16.715145	16.624479	111.932105	111.976718
25%	5.830218	5.829164	25.411047	25.412839	16.842389	16.668147	112.299	112.201765
50%	5.843647	5.849717	25.463858	25.457608	16.883783	16.752088	112.496621	112.320372
75%	5.86248	5.868858	25.78003	25.623765	17.104022	16.859496	112.682002	112.569707
max	5.920165	5.897414	26.257963	26.102652	18.08476	17.147931	113.44813	113.425736

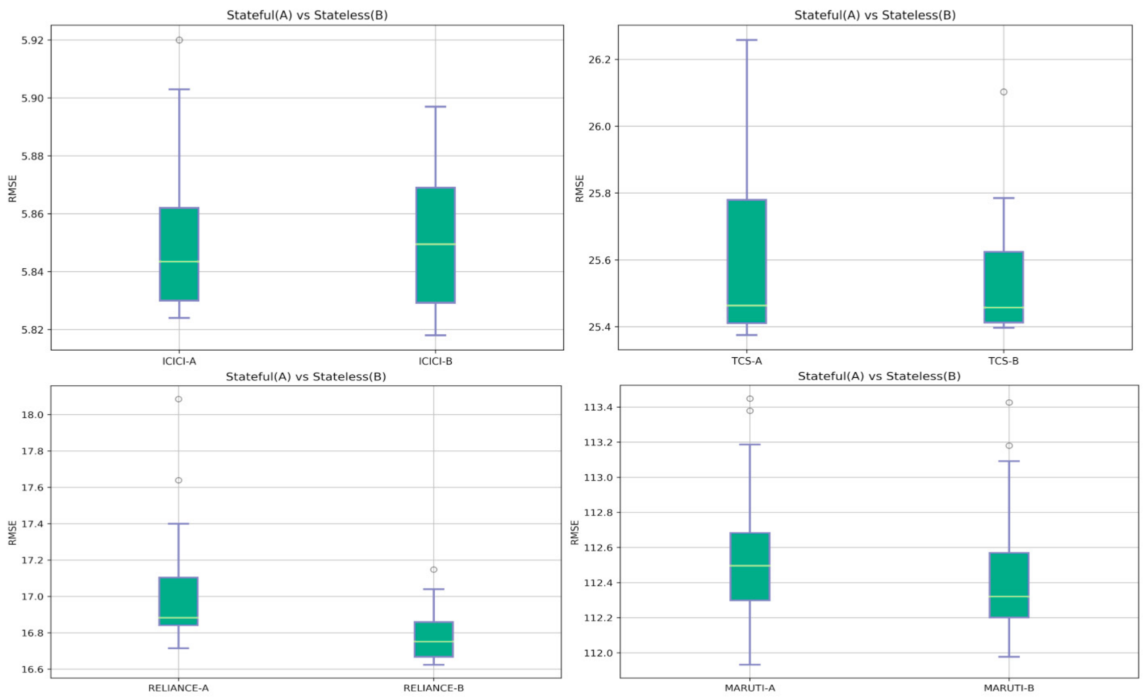

Fig. 5. Box plots for Stateful (A) and Stateless (B) LSTM

From table 1 and Fig. 5, no major difference is found between the performance of stateful and stateless LSTM. Mean RMSE for ICICI is lower for stateful as compared to stateless but for TCS, RELIANCE and MARUTI stateless gives a lower mean RMSE. To find out if these differences are meaningful, a P-value test for statistical significance was applied for a 5% significance level (alpha). The P-values were as follows: ICICI 0.898, TCS 0.216, RELIANCE 0.002 and MARUTI 0.263. The values for three companies out of four (except RELIANCE) are greater than the significance level (0.05) so it can be concluded that the differences in prediction accuracy between stateful and stateless LSTM are not statistically significant.

Experiment 2: Other hyper-parameters were kept constant while the number of hidden layers (n) was varied. The experiment was conducted with n values from 1 to 7. The results are given in table 2.

Table 2. Comparison of number of hidden layers for ICICI

	One	Two	Three	Four	Five	Six	Seven
count	30	30	30	30	30	30	30
mean	5.851305	5.860783	5.867778	5.880775	5.881493	5.88441	5.892269
std	0.026454	0.015801	0.019696	0.011612	0.011708	0.011299	0.008682
min	5.824255	5.827461	5.832743	5.855847	5.860987	5.854128	5.87459
25%	5.830218	5.851372	5.85345	5.873032	5.871773	5.878954	5.886606
50%	5.843647	5.858564	5.864112	5.881874	5.88012	5.885006	5.890901
75%	5.86248	5.871088	5.879801	5.887195	5.889274	5.891017	5.896691
max	5.920165	5.890493	5.908064	5.902414	5.903165	5.903842	5.909898

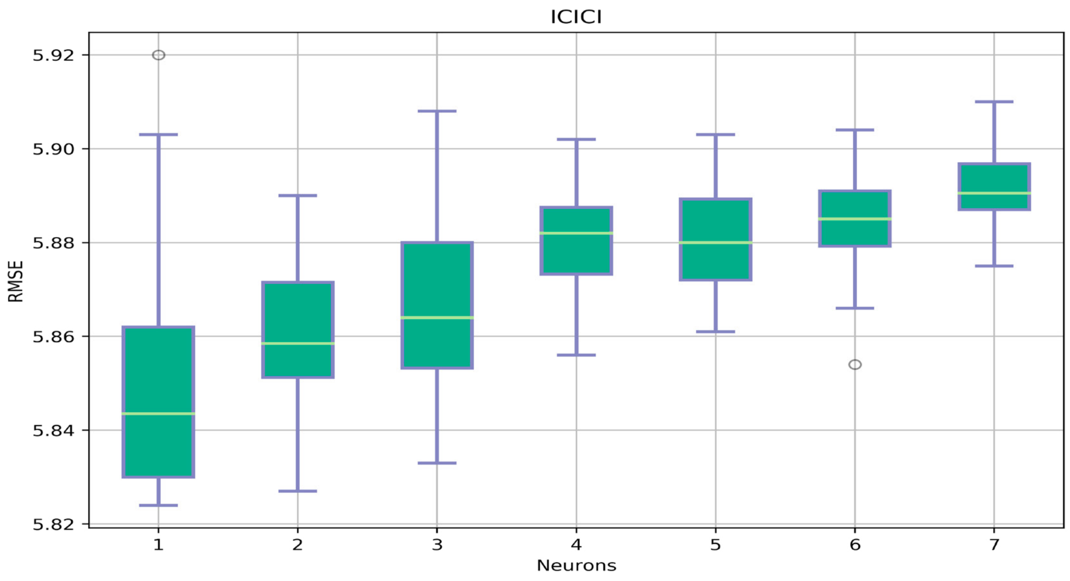

Fig. 6. Box plots for number of hidden layers for ICICI

Table 3. Comparison of number of hidden layers for TCS

	One	Two	Three	Four	Five	Six	Seven
count	30	30	30	30	30	30	30
mean	25.59691	25.47586	25.49538	25.45115	25.48475	25.48042	25.47779
std	0.247032	0.106443	0.101739	0.073064	0.059278	0.062683	0.046589
min	25.37497	25.3578	25.36995	25.35989	25.38053	25.39786	25.39795
25%	25.41105	25.4084	25.40937	25.38487	25.44425	25.42941	25.44357
50%	25.46386	25.45043	25.46922	25.43292	25.47954	25.46087	25.48252
75%	25.78003	25.50188	25.57136	25.50593	25.51148	25.52651	25.50313
max	26.25796	25.8193	25.72406	25.61197	25.62057	25.61386	25.57116

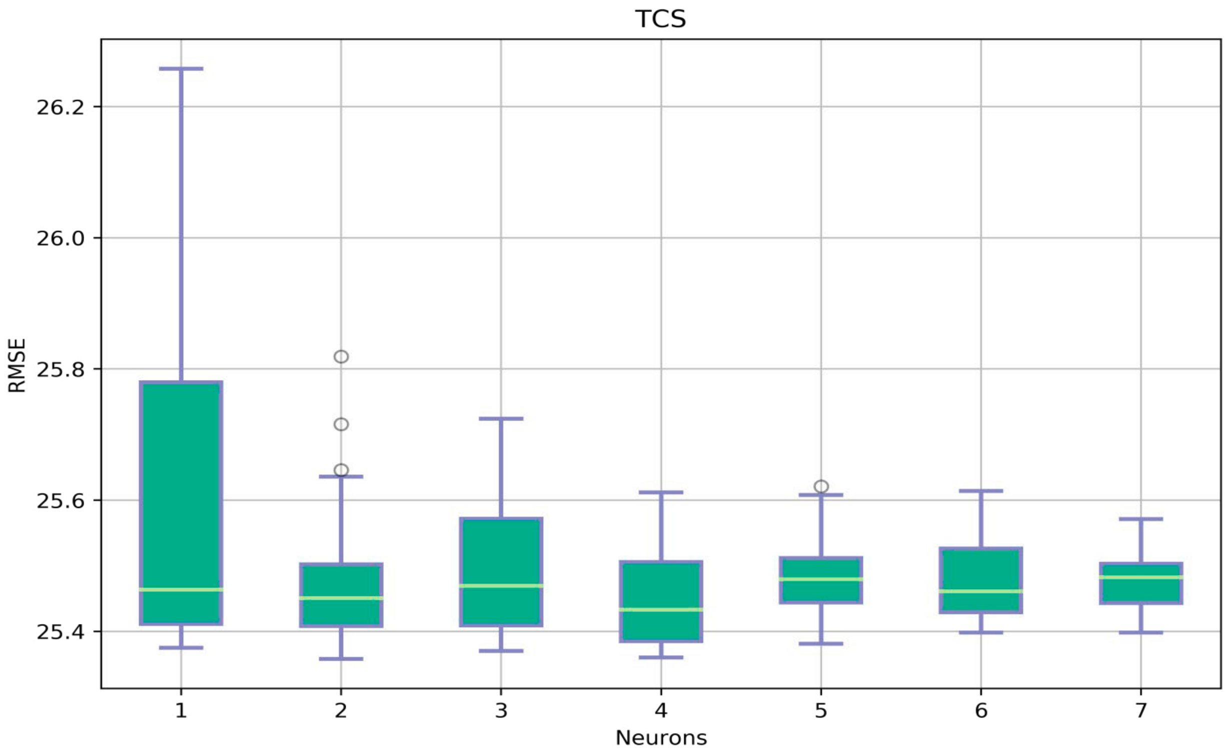

Fig. 7. Box plots for number of hidden layers for TCS

Fig. 6 for ICICI shows very clearly that n = 1 is the best configuration as far as the mean RMSE is concerned. This is less clear from Fig. 7 for TCS where there is not much to differentiate the various configurations. To validate this statistically, one-way ANOVA test was applied. It was found that for ICICI there was a statistically significant difference between groups as determined by one-way ANOVA (F(6,203) = 24.80, p = 5.54x10^{-22}). For TCS, again, there was a statistically significant difference between groups as determined by one-way ANOVA (F(6,203) = 4.76, p = 1.45x10^{-4}). Another thing that stands out clearly from all the figures is that the standard deviation decreases drastically as n increases from 1 to 7.

6. Conclusion

In Experiment 1, stateful and stateless LSTMs were compared for four companies - ICICI, TCS, Reliance and Maruti. The P-values showed that the differences between stateful and stateless LSTM for the stock price prediction problem chosen for the experiment, are statistically insignificant. Most of the difference in values can be accounted for by the random seeding that occurs for every LSTM run that produces small variations in output. The standard deviation values and the spread in the box and whisker plot diagram shows that the deviation or spread for stateless is less than that for stateful. This suggests that stateless LSTM is more stable compared to stateful LSTM. It can be concluded that for time series prediction problems, a stateless LSTM model is preferable due to higher stability. If one is doing language modelling where successive batches are successive chunks of text with possible interconnections, it makes sense to go for stateful LSTM.

In Experiment 2, the number of hidden layers was varied from one to seven. The results show that n = 1 appears to be the best configuration as far as mean RMSE is concerned. This was also validated by the one-way ANOVA test. The recommendation here would be to go with one hidden layer for most problems due to better accuracy, easier training and less risk of over-fitting. If there is a problem that is very complex and cannot be efficiently modelled with one layer, one may want to go with higher number of layers but such cases are expected to be rare.

An advantage of increasing the number of hidden layers is that the LSTM becomes more stable as revealed by the decreasing standard deviation values and the spread in the box and whisker plot diagram.

7. Future Work and limitations

Deep learning techniques are highly computation intensive. The present work has been performed on Google Colab which provides eight Tensor processing units (TPUs). If dedicated computational resources are available, more experiments can be run with more data and higher epochs. This may provide more clarity on hyper-parameter tuning.

The present work is limited to tuning the basic LSTM architecture. Different types of LSTM models can be similarly tuned.

References

[1] Lam, Monica. (2004) "Neural network techniques for financial performance prediction: integrating fundamental and technical analysis." *Decision Support Systems* **37 (4)**: 567-581.

[2] Hutter, Frank, Holger Hoos, Kevin Leyton-Brown. (2014) "An Efficient Approach for Assessing Hyperparameter Importance." *Proceedings of the 31st International Conference on Machine Learning (PMLR)* **32 (1)**: 754-762.

[3] Persio, Luca Di, and Oleksandr Honchar. (2016) "Artificial Neural Networks architectures for stock price prediction: comparisons and applications." *International Journal of Circuits, Systems and Signal Processing* **10**: 403-413.

[4] Foster, W. R., F. Collopy, and L. H. Ungar. (1992) "Neural network forecasting of short, noisy time series." *Computers & Chemical Engineering* **16 (4)**: 293-297.

[5] Lu, Chi-Jie, Tian-Shyug Lee, and Chih-Chou Chiu. (2009) "Financial time series forecasting using independent component analysis and support vector regression." *Decision Support Systems* **47 (2)**: 115–125.

[6] Hamzaebi, Coskun, Akay Diyar, and Kutay Fevzi. (2009) "Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting." *Expert Systems with Applications* **36 (2):** 3839-3844.

[7] Dixon, Matthew, Diego Klabjan, and Jin Hoon Bang. (2017) "Classification-based financial markets prediction using deep neural networks." *Algorithmic Finance* **6**: 67–77.

[8] Bouktif, Salah, Ali Fiaz, Ali Ouni, and Mohamed Adel Serhani. (2018) "Optimal Deep Learning LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches." *Energies* **11 (7)**: 1636.

[9] Sezer, Omer Berat, Murat Ozbayoglu, and Erdogan Dogdu. (2017) "A Deep Neural-Network Based Stock Trading System Based on Evolutionary Optimized Technical Analysis Parameters." *Procedia Computer Science* **114**: 473-480.

[10] Dash, Rajashree, and Pradipta Kishore Dash. (2016) "A hybrid stock trading framework integrating technical analysis with machine learning techniques." *The Journal of Finance and Data Science* **2 (1)**: 42-57.

[11] Fischer, Thomas, and Christopher Krauss. (2018) "Deep learning with long short-term memory networks for financial market predictions." *European Journal of Operational Research* **270 (2)**: 654-669.

[12] Hiransha, M, Gopalakrishnan E. A., Vijay Krishna Menon, and Soman K. P. (2018) "NSE Stock Market Prediction Using Deep-Learning Models." *Procedia Computer Science* **132**: 1351-1362.

[13] Stathakis, D. (2009) "How many hidden layers and nodes?" *International Journal of Remote Sensing* **30 (8)**: 2133–2147.

[14] Greff, Klaus, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. (2017) "LSTM: A Search Space Odyssey." *IEEE Transactions on Neural Networks and Learning Systems* **28 (10)**: 2222 - 2232.

[15] Karmiani, D., R. Kazi, A. Nambisan, A. Shah, and V. Kamble. (2019) "Comparison of Predictive Algorithms: Backpropagation, SVM, LSTM and Kalman Filter for Stock Market." *Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates*: 228-234.

[16] Bergstra, James, and Yoshua Bengio. (2012) "Random Search for Hyper-Parameter Optimization." *Journal of Machine Learning Research* **13**: 281-305.

[17] Olah, Christopher. (2015) Understanding LSTM Networks [Blog post]. Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/