International Journal of Engineering & Technology, 7 (4.15) (2018) 25-29

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper



Long Short Term Memory Recurrent Network for Standard and Poor's 500 Index Modelling

Said Jadid Abdulkadir[1]*, Hitham Alhussian[1], Muhammad Nazmi[1], Asim A Elsheikh[2]

[1]Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Malaysia
[2]Department of Computer Science, Tabuk University, Saudi Arabia
**Corresponding author E-mail: saidjadid.a@utp.edu.my*

Abstract

Forecasting time-series data are imperative especially when planning is required through modelling using uncertain knowledge of future events. Recurrent neural network models have been applied in the industry and outperform standard artificial neural networks in forecasting, but fail in long term time-series forecasting due to the vanishing gradient problem. This study offers a robust solution that can be implemented for long-term forecasting using a special architecture of recurrent neural network known as Long Short Term Memory (LSTM) model to overcome the vanishing gradient problem. LSTM is specially designed to avoid the long-term dependency problem as their default behavior. Empirical analysis is performed using quantitative forecasting metrics and comparative model performance on the forecasted outputs. An evaluation analysis is performed to validate that the LSTM model provides better forecasted outputs on Standard & Poor's 500 Index (S&P 500) in terms of error metrics as compared to other forecasting models.

Keywords: *Financial Time-Series; Long Short Term Memory, Recurrent Network; Standard and Poor's 500 Index.*

1. Introduction

Forecasting is the prediction of a condition or situation at some future time [1]. Business decisions especially financial related decisions depend heavily on forecasts of future events [2] of which investors and stakeholders are particularly interested in the projections of future events. The main difficulty in forecasting financial data is the size of the data obtained is massive, and usually chaotic in nature, hence affecting the ability of the prediction model to generate robust forecasts.

Statistical models have been used for time-series forecasting, however they are limited to linear problems unlike real-world problems which are non-linear [3-6]. Non-linear models have also been implemented in time-series forecasting such as support vector machines (SVM) [7], artificial neural network [8] and genetic algorithm [9].

Forecasting time-series can be categorized as long term or short term. The problem with artificial neural networks which are usually considered good for short term forecasting, and its precise nature, do not have loops which can provide feedback from the previous network output to the next network input [10]. While for long term forecasting, a loop is needed to feed the neural network with previous forecasted values as its next input. Several previous forecasted values are used as the next input for the network which depends on the time steps that have been set, to generate the next forecasted value. When the network continuously feeds using previous forecasted values, the network can generate an infinite process of forecasting, which also leads to a more complex modelling process than short term forecasting [11].

The goal of a forecaster is to generate best results using minimum required input data and a least complex model [12]. With this in mind, there is a need to have analysis on current non-linear models which are dynamic models that can help businesses to make better decision using accurate forecasting outcomes.

Recurrent neural network (RNN) is one type of neural network that is able to perform forecasting as its structure includes previous values being fed into the next step, hence giving it the ability to forecast with higher accuracy. Unfortunately, the problem of vanishing gradient limits its ability to forecast further in the future, as the weight value is diminishing and does not contribute to improvise the recurrent neural network model. Theoretically, RNN can do long-term dependencies. But unfortunately, in practice, it cannot be able to improve the accuracy of the model and becomes stagnant as the weight update process is very low.

Long short term memory (LSTM) model is one of the successors of RNN, which is designed to solve the problem of vanishing gradient that exists in RNN [13]. The idea of LSTM is to have several gates, which are input, forget, and output gate to control which subject or value that contributes to the output, and need to be forgotten.

LSTMs are able to handle the long-term dependencies [13-14], hence improve the accuracy to predict and give robust value. In this paper, the LSTM model will be evaluated and compared to other forecasting models, to validate the effectiveness of LSTM. Most forecasting models will use 70 percent of the datasets as training data, and another 30 percent of the datasets as test data to give reliable accuracy and measurements. The evaluation will be compared based on the value of Mean Square Error (MSE) of each model. The lower the value of MSE of the model, the better the prediction generated. The layout of this paper is structured as follows, the problem statement in Section 2, more detailed explanation of Long Short Term Memory model in Section 3, experimental results and discussions in Section 4. The last section of this paper will summarize the findings and conclusion.

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

2. Problem Statement

Recurrent network models are powerful neural network models that specialize in sequential learning, and time-series data. The problem with recurrent neural network model is the vanishing gradient problem, which makes the model to forget the far previous states hence making it less reliable for long-term forecasting. Vanishing gradient is a major problem in a recurrent neural network as it makes the gradient to easily vanish and the learning process for the model becomes much more difficult.
For example, in case of the RNN model trained using long sequential data (e.g. minimum of 100 time steps), the gradient will vanish easily due to the long sequences of the gradients being back propagated during the updating of weights through the recurrent layers. This creates a problem in the forecasting model, hence affecting the performance of the model in generating long-term forecasts.
To overcome this problem, an upgraded version of recurrent neural network model is built to solve this vanishing gradient problem, called Long Short Term Memory network model. LSTM networks are a variant of RNNs which have the ability to store information and propagate losses over long sequences of data [13-14].

3. Methodology

Recurrent Neural Network helps to connect previous information to the present task and create a neural network model that can understand the current information based on historical information. Occasionally, we just need to look at a few windows of past information to forecast the present or several next tasks. For example, in text generation, if we try to predict the last word of "the sky is blue", we do not need to have seen all the context of the paragraph, instead just take few words before as it is already obvious that the next is going to be "blue". While in some cases, we might need to see more context or maybe all the contexts. An example, for this is to assume that the first sentence in the paragraph is "I am from Malaysia", and we try to predict the last word for the paragraph, which the sentence is "that's why I can speak Malay". Based on the recent text given, the next word should be the language spoken. But to know which language, we need to go further back to the first sentence of the paragraph. As the gap is very large, RNN has difficulties to learn about the missing information.
Theoretically, RNN can do long-term dependencies but unfortunately, the result is not the same when it is being applied in practice. Practically, this cannot be achieved because of an old and fundamental problem. According to [15-21], learning algorithms used by RNN are usually based on gradient cost function, which are typically used when performing the backpropagation process within the network. The problem with backpropagation when having long term dependencies is the gradient vanishing problem, which causes difficulty in training of the RNN using gradient based learning methods. As a result of the gradient update process being very small the network barely learns anything from it. LSTM (Long Short Term Memory) is applied to overcome this problem.
Long Short Term Memory network is a special kind of RNN, designed to be able to learn long-term dependencies. It is also explicitly designed to avoid the old problem of vanishing gradient in RNN [22], and then later refined by a lot of researchers in application to different types of time-series data [23-24]. The normal RNN structure only has a simple structure with a single sigmoid layer or activation function as shown in Figure 1.

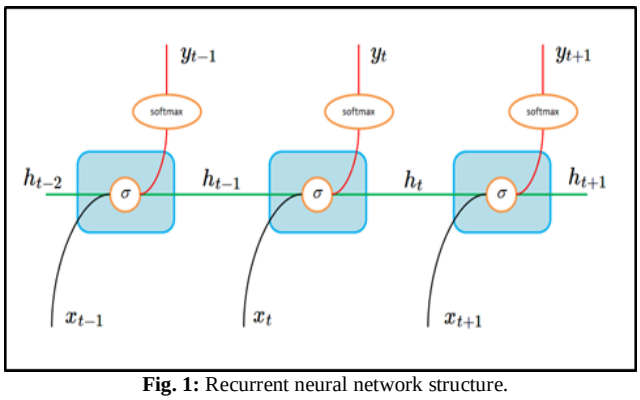


Fig. 1: Recurrent neural network structure.

LSTMs have the same structure as RNN, but instead of having a single sigmoid layer to be activated, LSTM has four calculations inside a single layer of LSTM as shown in Figure 2

.

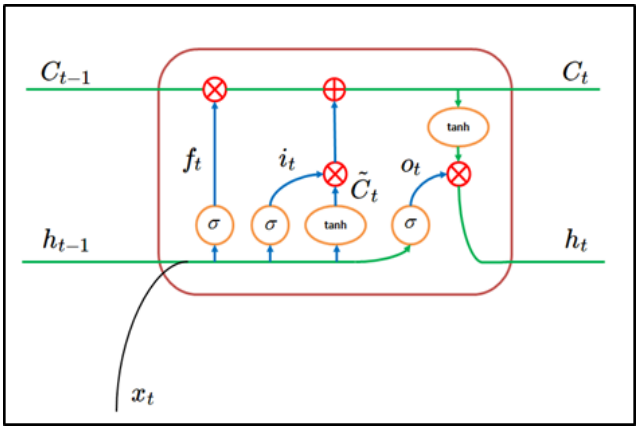


Fig. 2: Long short term memory structural model.

An LSTM block or cell can contain three or four gates that protect and control the information coming through. The gates are named as input gate, forget gate, and output gate. In the forget gate layer as shown in Figure 3, it receives output from the previous cell, cell state from the previous cell and the current input. The cell state from the previous cell will give a number between 0 and 1, where 1 translates to the process of storing the value while 0 translates to getting rid of the previous information. The cell state decides whether to remember or to forget the previous cell. This is useful when the model tries to forecast the next step based on the previous step, whenever given a new input and we want to forget the old inputs.

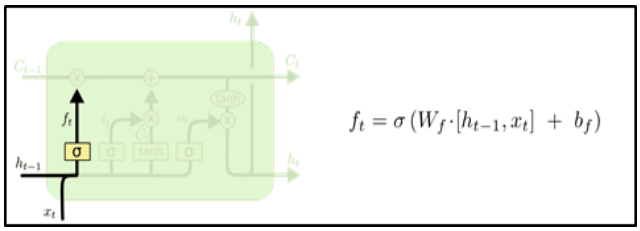


Fig. 3: Forget gate layer within the LSTM structure (activation function includes bias and weights with inputs).

After deciding whether to remember or forget the previous information, we need to decide what new information is needed to be stored. This task is given to the input gate as shown in Figure 4. In here, a sigmoid activation is implemented on concatenated input and previous output to decide which values that needs to be updated. Tanh activation function creates a vector of candidate values of cell states. These two values are then combined to create an update to the next cell state.

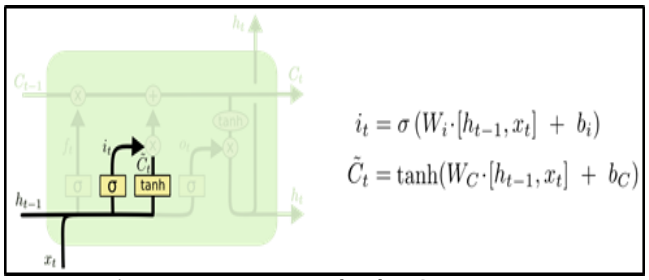


Fig. 4: Input gate part 1 within the LSTM structure.

To update the old cell state, result from the previous step which is in the forget gate, is then multiplied to the old cell state, forgetting the things that are decided to be forgotten and then added to the result from the multiplication of sigmoid and tanh activation functions in the earlier step (vector of candidate values of cell state). The new candidate value is scaled by how much we decide to update each state value. In here, we have already dropped the information about the old subject and replace it with new information as shown in Figure 5.

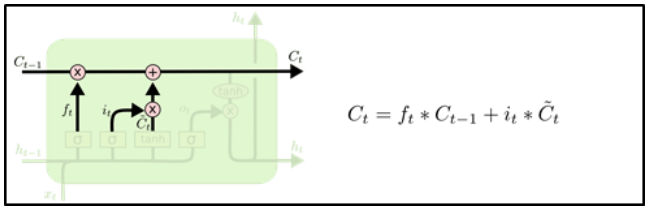


Fig. 5: Input gate part 2 within the LSTM structure.

Until the last step, we already have the cell state, and now we need to decide on the output. The output is based on the cell state, and the concatenated input and previous output that has been generated using the sigmoid function. The cell state is obtained using a tanh function and then multiplied with the output of the sigmoid gate, and to generate an output as shown in Figure 6.

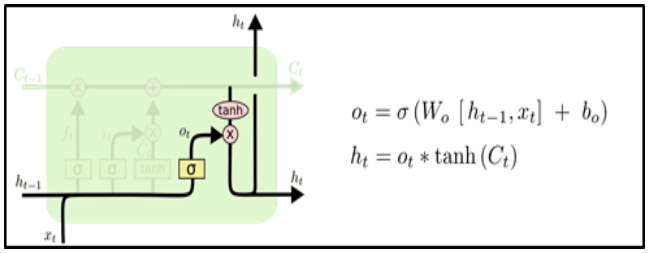


Fig. 6: Output gate layer within the LSTM structure results and discussions.

4. Results and Discussion

The S&P 500 or the Standard and Poor's 500 [25] is a stock market index based on the market capitalizations of 500 largest companies listed on the NYSE [15] or NASDAQ [26].

The S&P 500 index components and their weightings are determined by S&P Dow Jones Indices. It differs from other U.S. stock market indices, such as the Dow Jones Industrial Average or the Nasdaq Composite index because of its diverse constituency and weighting methodology. It is one of the most commonly followed equity indices, and many consider it one of the best representations of the U.S. stock market.

The National Bureau of Economic Research [24] has classified common stocks as a leading indicator of business cycles. S&P 500 financial time-series (16140 points) are selected as shown in Figure 7. For comparative model analysis, 70 percent of the datasets are used as training data, and the remaining 30 percent of the datasets as test data. The evaluation will be compared based on the value of Mean Square Error (MSE) of each model. The lower the value of MSE of the model, the better the prediction is generated.

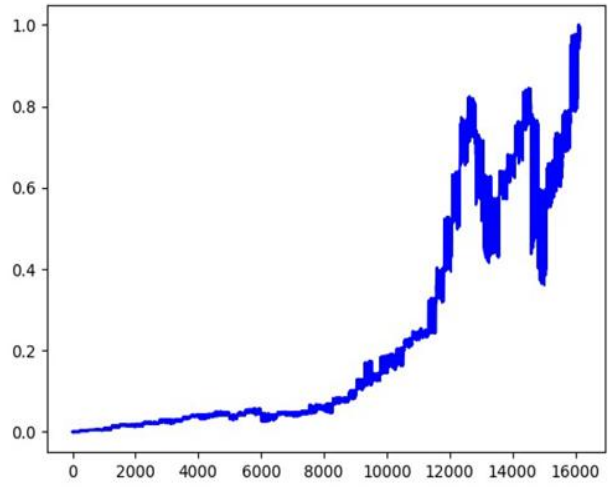


Fig. 7: 16,140 daily closing price for S&P 500 dataset.

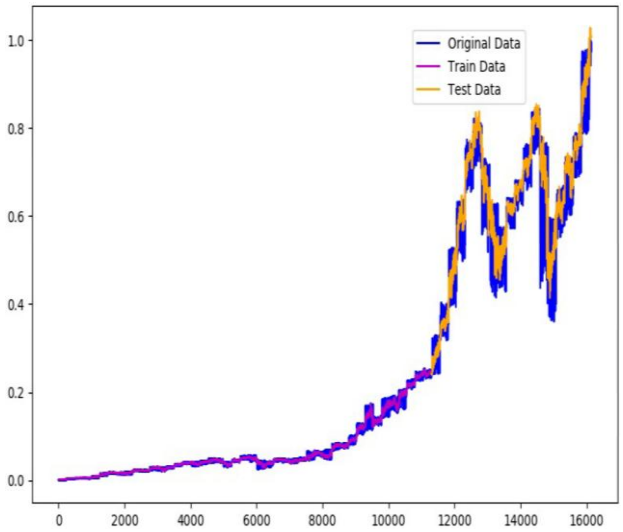


Fig. 8: Long Short Term Memory neural network implementation using S&P 500 data.

In data pre-processing step, 11,287 samples of S&P 500 are divided in the training process and the remaining 4,832 samples are used for testing of the LSTM model as shown in Figure 8. The experimental process is done and the quantitative metrics obtained using 10 epochs. The first epoch has a mean squared error of 0.0079 and the 10th epoch had a mean squared error of 0.0017.

For comparative analysis, evaluation performance between the LSTM model and other commonly used models such as linear regression, support vector machine and recurrent neural network trained using gradient boosted regression algorithm is performed.

Evaluation performance using the same number of training and testing samples are done (11,287 samples for training and 4,832 samples for testing). The graphical representation when S&P 500 is applied using linear regression is as shown in Figure 9.

Two other models namely support vector machine and a gradient boosted regression model are also tested using the same number of training and testing samples as applied to the long term short memory model and the forecasted outcomes from the S&P 500 dataset are as shown in Figures 10 and 11.

The other evaluation process apart from comparative analysis is done by performing a quantitative analysis from the forecasted outcomes obtained from the four models (LSTM, gradient boosted, SVM and linear regression). The quantitative metric applied is obtained using the mean square error from an average of 10 epochs from each model as shown in Figure 12. Based on the MSE quantitative values, the better the model the lower the MSE value of which LSTM has an MSE value of 0.00159.

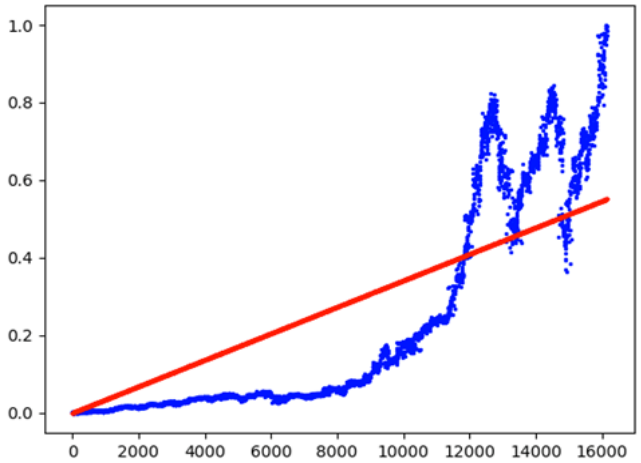


Fig. 9: Forecasting S&P 500 using Linear Regression.

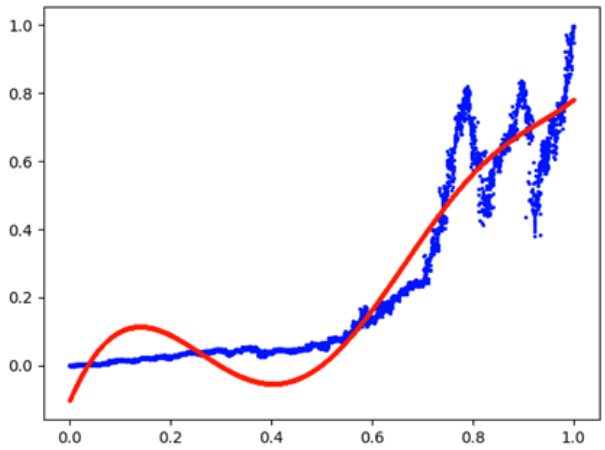


Fig. 10: Forecasting S&P 500 using Support Vector Machine.

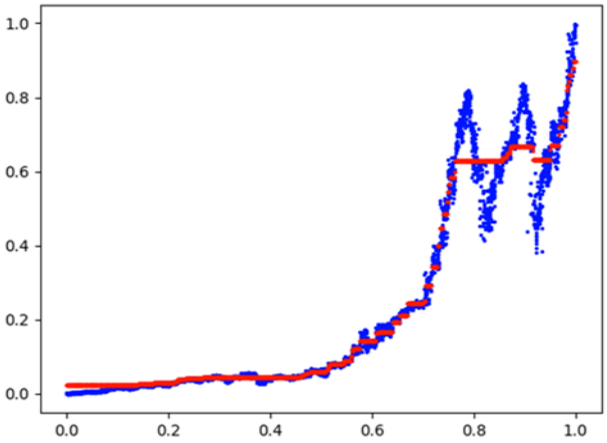


Fig. 11: Forecasting S&P 500 using Gradient Boosted Regression.

```
MSE - Mean Square Error, the lower the better
=============================================
LSTM MSE : 0.00159
Gradient Boosted MSE : 0.00181
SVM MSE : 0.00735
Linear Regression MSE : 0.02657
```

Fig. 12: Mean Square Error of each model implemented using S&P 500.

5. Conclusion

The standard recurrent neural network has a simple structure with a single sigmoid layer or activation function. Long short term memory (LSTM) model, is one of the successors of RNN, which is designed to solve the problem of vanishing gradient that exists in RNN. The idea of LSTM is to have several gates, which are input, forget, and output gate to control which subject or value that contributes to the output, and need to be forgotten.

Long Short Term Memory recurrent networks have the same structure as the RNN, but instead of having a single sigmoid layer to be activated, LSTM has four calculations inside a single layer of LSTM. An LSTM block or cell contains three or four gates that protect and control the information coming through. The gates are named as input gate, forget gate, and output gate. In the forget gate layer, it receives output from the previous cell, cell state from the previous cell and the current input. The cell state decides whether to remember or to forget the previous cell. This is useful when the model tries to forecast the next step based on the previous step, whenever given a new input and we want to forget the old inputs.

LSTMs are able to handle long-term dependencies, hence improving the accuracy in prediction providing robust outputs. In this paper, the LSTM model is evaluated and compared with other forecasting models, to validate the effectiveness of LSTM. Most forecasting models will use 70 percent of the datasets as training data, and another 30 percent of the datasets as test data to give reliable accuracy and measurements. The evaluation is compared based on the value of Mean Square Error (MSE) of each model. The lower the value of MSE, the better the prediction generated by the model.

In this study, S&P 500 time-series data is applied to differentiate the performance of the comparative models stated in the results section. From the experimental results, it can be concluded through quantitative and comparative model performance that the forecasting performance of long sort term memory network is better as compared to Support Vector Machine, Gradient Boosted Regressor and Linear Regression. The structural modification that differentiates the long short term memory network to the standard recurrent neural network improves the learning of long term dependencies hence generating more accurate long term forecasts.

Acknowledgement

This research was supported by STIRF grant (0153AA-F73), funded by Universiti Teknologi PETRONAS (UTP).

References

[1] Coelho, I. M., Coelho, V. N., Luz, E. J. D. S., Ochi, L. S., Guimarães, F. G., & Rios, E. (2017). A GPU deep learning metaheuristic based model for time series forecasting. Applied Energy, 201, 412-418.

[2] Gilliland, M. (2017). Changing the paradigm for business forecasting. Foresight: The International Journal of Applied Forecasting, 2017(44), 29-35.

[3] Abdulkadir, S. J., & Yong, S. P. (2013). Unscented Kalman filter for noisy multivariate financial time-series data. Proceedings of the International Workshop on Multi-Disciplinary Trends in Artificial Intelligence, pp. 87-96.

[4] Abdulkadir, S. J., Shamsuddin, S. M., & Sallehuddin, R. (2012). Moisture prediction in maize using three term back propagation neural network. International Journal of Environmental Science and Development, 3(2), 199-204.

[5] Abdulkadir, S. J., Yong, S. P., & Zakaria, N. (2016). Hybrid neural network model for metocean data analysis. Journal of Informatics and Mathematical Sciences, 8(4), 245-251.

[6] Abdulkadir, S. J., Yong, S. P., & Alhussian, H. (2016). An enhanced ELMAN-NARX hybrid model for FTSE Bursa Malaysia KLCI index forecasting. Proceedings of the IEEE 3rd International Conference on Computer and Information Sciences, pp. 304-309.

[7] Ahmad, A. S., Hassan, M. Y., Abdullah, M. P., Rahman, H. A., Hussin, F., Abdullah, H., & Saidur, R. (2014). A review on applications of ANN and SVM for building electrical energy consumption

forecasting. Renewable and Sustainable Energy Reviews, 33, 102-109.
[8] Abdulkadir, S. J., & Yong, S. P. (2015). Scaled UKF–NARX hybrid model for multi-step-ahead forecasting of chaotic time series data. Soft Computing, 19(12), 3479-3496.
[9] Liu, D., Niu, D., Wang, H., & Fan, L. (2014). Short-term wind speed forecasting using wavelet transform and support vector machines optimized by genetic algorithm. Renewable Energy, 62, 592-597.
[10] Abdulkadir, S. J., Yong, S. P., Marimuthu, M., & Lai, F. W. (2014). Hybridization of ensemble Kalman filter and non-linear auto-regressive neural network for financial forecasting. In R. Prasath, P. O'Reilly, & T. Kathirvalavakumar (Eds.), Mining Intelligence and Knowledge Exploration, Lecture Notes in Computer Science, vol 8891. Cham: Springer, pp. 72-81.
[11] He, M., Yang, L., Zhang, J., & Vittal, V. (2014). A spatio-temporal analysis approach for short-term forecast of wind farm generation. IEEE Transactions on Power Systems, 29(4), 1611-1622.
[12] Abdulkadir, S. J., & Yong, S. P. (2014). Empirical analysis of parallel-NARX recurrent network for long-term chaotic financial forecasting. Proceedings of the IEEE International Conference on Computer and Information Sciences, pp. 1-6.
[13] Manashty, A., & Thomson, J. L. (2017). A new temporal abstraction for health diagnosis prediction using deep recurrent networks. Proceedings of the ACM 21st International Database Engineering and Applications Symposium, pp. 14-19.
[14] Bengio, Y., Frasconi, P., & Simard, P. (1993). The problem of learning long-term dependencies in recurrent networks. Proceedings of the IEEE International Conference on Neural Networks, pp. 1183-1188.
[15] Alfi, V., Coccetti, F., Petri, A., & Pietronero, L. (2007). Roughness and finite size effect in the NYSE stock-price fluctuations. European Physical Journal B, 55(2), 135-142.
[16] Lin, T., Horne, B. G., & Giles, C. L. (1998). How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies. Neural Networks, 11(5), 861-868.
[17] Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Proceedings of the Fifteenth Annual Conference of the International Speech Communication Association, pp. 338-342.
[18] Bengio, Y. (2009). Learning deep architectures for AI. Foundations and Trends® in Machine Learning, 2(1), 1-127.
[19] Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249-256.
[20] Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 4580-4584.
[21] Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. Proceedings of the International Conference on Machine Learning, pp. 1310-1318.
[22] Squartini, S., Hussain, A., & Piazza, F. (2003). Attempting to reduce the vanishing gradient effect through a novel recurrent multiscale architecture. Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 2819-2824.
[23] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157-166.
[24] Neumann, M., & Skiadopoulos, G. (2013). Predictable dynamics in higher-order risk-neutral moments: Evidence from the S&P 500 options. Journal of Financial and Quantitative Analysis, 48(3), 947-977.
[25] Carrion, A. (2013). Very fast money: High-frequency trading on the NASDAQ. Journal of Financial Markets, 16(4), 680-711.
[26] Jeanne, O., & Korinek, A. (2010). Excessive volatility in capital flows: A Pigouvian taxation approach. American Economic Review, 100(2), 403-407.