Stock Prediction with Recurrent Neural Networks

Jacob Oury **Jin Zhang** **Guanjie Huang**

jdo6@psu.edu **zhangjin94@gmail.com** **gzh8@psu.edu**
The Pennsylvania State University
College of Information Sciences and Technology
332 Info Science and Tech
University Park, PA, 16802, USA

Abstract

Stock market prediction is a very important and classic topic in financial economics. A good prediction of a stock's future movement can provide insight about the market behavior over time, and then bring significant profit. We used three types of deep-learning models using recurrent neural networks to compare how well they performed at predicting a stock price. We found that long short-term memory networks and GRU networks performed similarly, though GRU typically had a lower mean square error than the other methods tested. The simple recurrent neural network performed even worse than the linear regression model.

Introduction

The prediction of the future prices of goods has been a problem many people are interested in solving since the birth of the markets themselves. However, the past half century has delivered an immense amount of research towards more effective, efficient tools for us to use in this endeavor. The rise of computing power has launched our ability to predict fluctuations in the market much further than any development prior. In this section, we explore the foundational theories that drive the field of stock prediction and describe some of the tools that have been used to do so.

The Efficient Market Hypothesis

Stock market price prediction is an extremely common focus for research on complex longitudinal data. However, it remains a difficult problem due to the sheer level of complexity and the difficulties involved with integrating the many different variables that can affect its price. Stock market data is nonlinear, uncertain, and non-stationary, further complicating the problem (Längkvist, Karlsson, & Loutfi, 2014). The Efficient Market Hypothesis (EMH) by Fama (1965) is a long-standing theory that has been at the heart of this research. The EMH asserts that stock prices follow a random walk pattern that is minimally affected by the past and present stock market prices. Many studies have validated the predictions of the EMH and it remains a prominent feature in the field (Basu, 1977; Malkiel, 2003). Following the formulation of the EMH, a significant amount of research has been conducted on what factors have the most effect on stock prices. One commonly attributed factor in stock price is the public availability and favorability of news (of various types) about the company (Malkiel, 2003). On top of long-standing "typical" news sources, ubiquitous access to the internet and social media like Twitter has expanded the search to include dynamic changes in sentiment and perception in real-time (Bollen, Mao, & Zeng, 2011). Even with the latest research, however, there still remains some dispute about whether stock prediction has a "hard" cap of 50% accuracy due to random walk pattern of stock growth (Malkiel, 2003; Tsai & Hsiao, 2010).

Stock Markey Prediction Tools and Methods

Group Method of Data Handling

At its most fundamental level, stock prediction is just one of many similar problems revolving around solving high-dimensional, complex problems with lots of data. One of the first (if not the first) methods aimed towards this task is the Group Method of Data Handling (GMDH). GMDH was designed to solve these kinds of problems and make predictions for seemingly random processes (Ivakhnenko, 1970). GMDH is commonly thought of as the first implementation of a neural network.

GMDH relies on heuristic self-organization to solve complex problems. GMDH processes use a gradual growth in complexity to formulate the "best" solution for a given dataset by passing the data through inductive, multi-layered polynomial models (Ivakhnenko, 1970). Since its inception, the GMDH family of algorithms has been used to solve many different problems that require multi-dimensional data prediction. These problems include the identification of physical laws, pattern recognition, and stepwise forecasting among many others. See Ivaknenko & Ivakhnenko (1995) for a more thorough review of the many applications of GMDH.

Neural Networks

Creating financial models with neural networks (NNs) is the most common method for analyzing stock data. Many different types of NNs have been formulated to predict stock prices and solve other similar problems. White (1988) is widely thought to be one of the first to implement a NN on IBM stock to create a model able to predict IBM's stock. White used Feed Forward Neural Networks (FFNNs) on IBM's stock data to demonstrate the ability for NNs to make novel discoveries in a previously inscrutable dataset. This is considered the beginning of the now wide-spread growth of NNs in stock prediction. The basic underlying structure of a neural network is shown below in Figure 1.

The strengths of the NN approach in this field are due to its flexibility and lack of outside interference that could negatively affect the accuracy of the models created. Yoo, Kim, & Jan (2007) offer the following advantages of NNs over other approaches. NN's approximate the relationships between data points by learning from the data itself, even when the data is non-linear or non-parametric. NNs are naturally resistant to the effects of noise and incomplete data on the model's accuracy. Yoo et al. (2007) also notes several drawbacks associated with NNs.

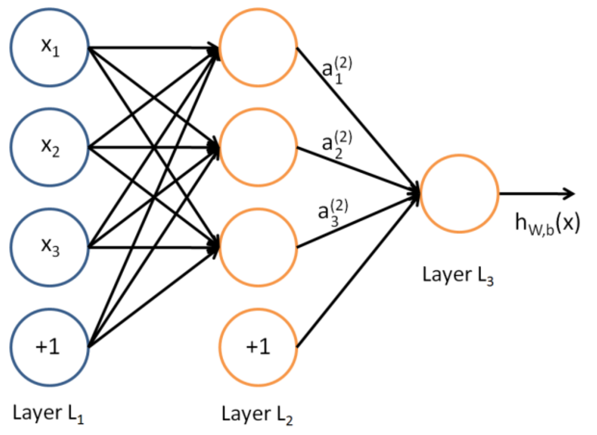

Figure 1: A simple neural network. Circles represent nodes or neurons. This shows how input data in the input layer (L_1) passes through the hidden layers (L_2) before being passed into the output layer (L_3). The nodes labeled "+1" are called bias units and represent the intercept term. Image and accompanying description from http://ufldl.stanford.edu/tutorial/

The first drawback is the lack of transparency for the significance and weight of each variable. This is referred to as the black box problem (Lawrence, 1997). The black box problem limits the future analysis of the findings of any particular NN. Another common issue for NNs is overtraining. Researchers must strike a delicate balance between creating an overly generalizable model and an under-performing, specific model (Yoo et al., 2007).

Types of NNs

One method for classification of NNs is by comparing whether they allow bidirectional transmission of information. The first NNs used only feedforward propagation, thus giving their name of feedforward neural networks (FFNNs). An example is shown in Figure 1 above. Note the lack of connections between nodes that are within the same layer. The simplest form for FFNNs are called *perceptron networks*. A perceptron network maintains only a single-layer with some activation function within the layer to output information. FFNN's are the most common NN used in stock prediction due to their simplicity (Vui, Soon, On, Alfred, & Anthony, 2013). FFNNs have been used in various markets like the US with IBM as well in international markets from Australia (Tilakaratne, Mammadov, & Morris, 2008; White, 1988).

In contrast, recurrent neural networks (RNNs) can use recurrent connections between nodes to expand their historical representation of the input knowledge. There are many different methods for RNNs including long short-term memory (LSTM), and gated recurrent units (GRU). We will explain these methods further in the methods section.

Methods

Prediction of stock market has been a hot topic from industry to academia. A lot of machine learning algorithms, like regression, support vector machine and neural networks, are employed to predict the stock prices. Since the stock price data can be considered as time-series data, recurrent neural networks (RNN) are one powerful model to deal with it owing to its capability of utilizing the sequential nature of the data. Some designs of RNNs were used to predict stock market (Lipton, Berkowitz, & Elkan, 2015; Saad, Prokhorov, & Wunsch II, 1998). Thereby in this project we would like to use the RNN to predict the stock price. Concretely, the Simple RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) will be used.

Data The dataset we have used was retrieved from Wikiprice. We have used Quandl API to get access to company stock prices from WIKI PRICE. The collected data were from the earliest available time to the most recent time, and the input value we have used is the closing value by each day (Holidays are exceptional). For example, the stock price data of Google is shown in Figure 2.

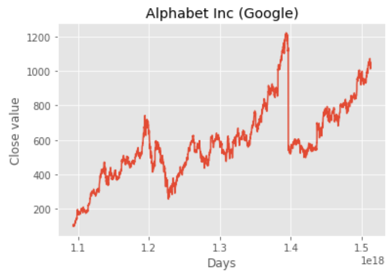

Figure 2: Example data from Wikiprice for Alphabet Inc (Google).

Preprocess

Before applying the model, the preprocess techniques are indispensable. The data is acquired by Quandl API as we described in last section. We first use rolling-window to slice the entire stock data into multiple small bins. And then the normalized data in each bin can be obtained using the equation shown.

,

$$\mathbf{x}_{normalized} = \frac{\mathbf{x} - x_0}{x_0}$$

Where x = ($x_0, x_1, \ldots, x_n$) is a vector and denotes the data in each bin. *n* is the window size. Normalization can help more with model generality/stability when dealing with machine learning/deep learning algorithm. In addition, we use the first 70% of the data as the training set and the remaining 30% as the testing set. And the validation set is 10% of the training set.

Model

Simple Recurrent Neural Network (Simple RNN)

The main idea behind the simple recurrent neural network is to make use of sequential information (Britz, 2015). RNNs are a type of the deep learning that have shown great abilities in neuro-linguistic programming tasks, which was first developed in 1980s. The defining feature of RNNs is their nature as a representation of how human memory works when it comes to the remembering the different things (*e.g.* remembering lyrics to a song, how to spell a word). RNNs will learn and calculate based on a sequence like how the human brain will do the same task. See Figure 3 below for a graphical representation of an RNN.

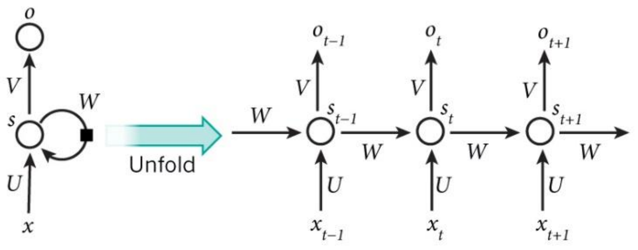

Figure 3: RNNs can be considered as series of forwarding neural network. Image taken from (Britz, 2015)

There are many types of the architectures within the RNN family such as fully recurrent RNNs, long short-term memory (LSTM), Hopfield, and many others. According to McClelland and Rumelhart, a simple RNN utilizes three basic layer, first layer (x) act as input layer this is where the data set is being ingested, the second layer act as hidden layer which take care of calculation and the evaluate the weight of current input combine with the last hidden layer recurrence, and the third layer act as the output or the result layer (McClelland & Rumelhart, 1981). The given input is normally in the sequence order or a fix vector which will contain a fix list for the data point. By evaluating the current data point (input node) with last hidden node and new weight is given to the current hidden node and the timesteps continual more and more data is being evaluated by the hidden layer the output layer will becomes more accuracy. According to an experiment conducted by Mikolov el al. on the RNN network-based language model, RNN can learn more contextual information and have greater efficiency than models with recursion. Furthermore, Its simplicity belies its ability to become highly intelligence given the right information and parameters (Mikolov, Karafiat, Burget, Cernocky, & Khudanpur, 2010).

For our experiment, we built three models using Simple RNNs. They contain either one layer, two layers, and three layers of Simple RNN followed by two fully connected layers. For example, the two-layer RNN one is shown below in Figure 4. The input is the normalized stock data, and the output layer is composed of 5 units with linear activation which is used to predict stock price of the next 5 days. The activation in penultimate layer is ReLU.

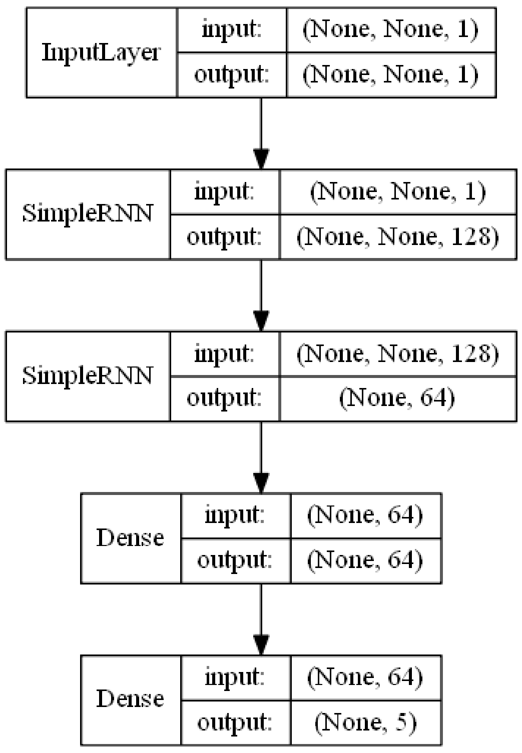

Figure 4: The diagram of our neural network with two layers of Simple RNN.

Long Short-Team Memory

Long short-term memory is a special kind of RNN. It is capable of learning long-term dependencies. More specifically, LSTM is a specific recurrent neural network architecture which was designed to model temporal sequences and their long-range dependencies more accurately in comparison to the conventional recurrent neural network (Sak, Senior, & Beaufays, 2014). According to Sak et al., originally, the LSTM contains memory blocks in the recurrent hidden layer, which contain memory cells. The memory cells contain self-connections, which is used to store the temporal state of the network. In general, LSTM uses the input gate for the data capturing, forget gate to reset the previous hidden state, and an output gate to determine the amount of the internal state for the external network (Zhou, Wu, Zhang, & Zhou, 2016). See Figure 5 below for the equation used for calculating LSTM values. See Figure 6 for the gating process for LSTM.

LSTM (Long Short-Term Memory)

$$\boldsymbol{f}_t = \sigma\left(W_f\left[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t\right] + \boldsymbol{b}_f\right), \tag{4a}$$
$$\boldsymbol{i}_t = \sigma\left(W_i\left[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t\right] + \boldsymbol{b}_i\right), \tag{4b}$$
$$\boldsymbol{o}_t = \sigma\left(W_o\left[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t\right] + \boldsymbol{b}_o\right), \tag{4c}$$
$$\tilde{\boldsymbol{c}}_t = \tanh\left(W_c\left[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t\right] + \boldsymbol{b}_c\right), \tag{4d}$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t, \tag{4e}$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t). \tag{4f}$$

Figure 5: LSTM equation of calculating hidden state. Image from (Zhou et al., 2016).

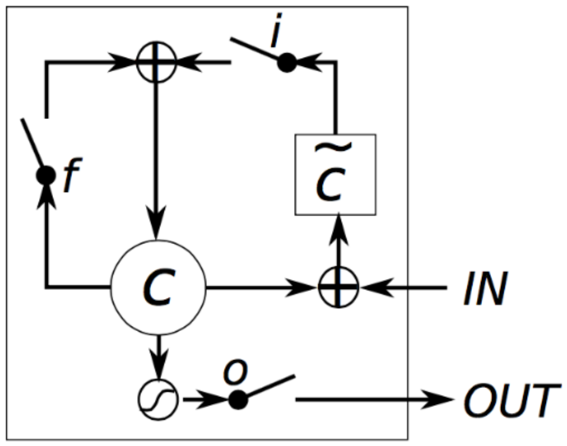

Figure 6: *The LSTM gating process for neural networks. Image from (Britz, 2015).*

Three models of different number of LSTM layer are explored to see its sequence learning power by ourselves. The models contain one layer, two layers, three layers of LSTM followed by two fully connected layers. For example, the figure below is the NN with one LSTM layer. The activation function of the last two layers are ReLU and linear respectively. A diagram of the LSTM model used is shown below in Figure 7.

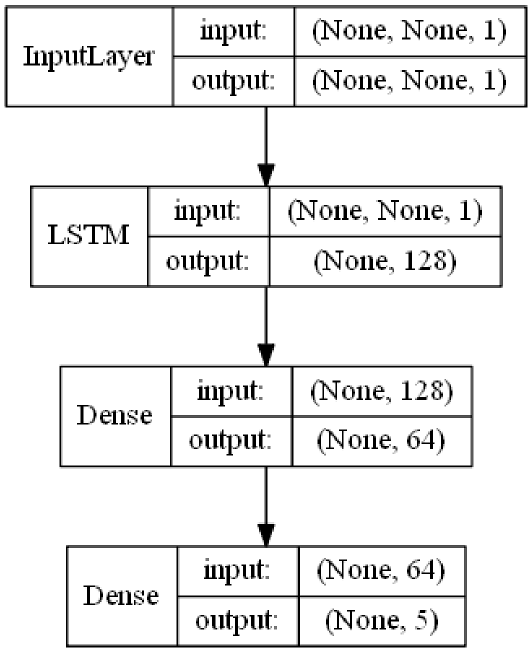

Fig 7: The diagram of our neural network with one LSTM layer.

Gated Recurrent Unit (GRU)

According to Cho et al., Gated recurrent units (GRUs) are an architecture for recurrent neural networks, and were introduced in 2014 by K. Cho to counter the need of the long term dependencies (Cho et al., 2014). Therefore, GRU can be considered as another variant of LSTM . In general, GRU has less parameter settings compared to LSTM, and the key difference that offsets the GRU from LSTM is the way it calculates its hidden layer/state. The GRU only requires two gates, it removes the use of the forget gate and replaces it with a update gate (z), the update gate takes care of the reset of the previous hidden state, it will able to determine how much it needs to reset. Due to having lesser parameter, GRU may be able to produce a result quicker compared to LSTM, but according to Chung, the advantages are minor (Chung, Gulcehre, Cho, & Bengio, 2014).

In our project, three models of different number of GRU layer are used. Specifically, we employ one layer, two layers, three layers of GRU followed by two fully connected layers. For instance, the NN with three GRU layers is shown below. In addition, the activation function of the last two layers are ReLU and linear separately. See Figures 8 and 9 below for the equations and graphical representation of GRU networks.

GRU (Gated Recurrent Unit)

$$z_t = \sigma\left(W_z\left[h_{t-1}, x_t\right] + b_z\right) , \tag{5a}$$
$$r_t = \sigma\left(W_r\left[h_{t-1}, x_t\right] + b_r\right) , \tag{5b}$$
$$\tilde{h}_t = \tanh\left(W_h\left[r_t \odot h_{t-1}, x_t\right] + b_h\right) , \tag{5c}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t . \tag{5d}$$

Figure 8: GRU equation of calculating hidden state From (Zhou et al., 2016).

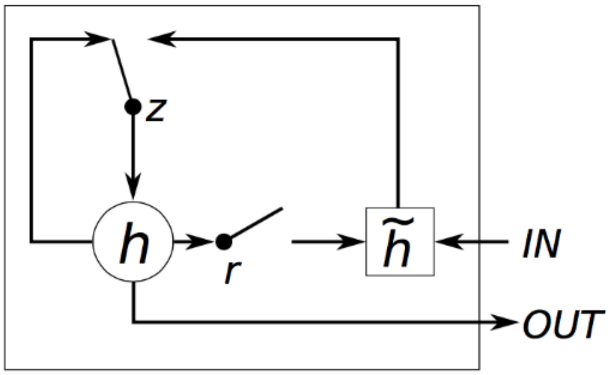

Figure 9: The GRU Gating process. Image from (Chung et al., 2014).

Experiment and Results

First, we tried to predict the stock price of next day by using the data of previous 100 days. We first used NN with one Simple RNN layer to predict. The loss function curve is shown in the figure 10. We can see that both of the curves of the validation and training loss are decrease with more epochs. And the Figure 11 shows the prediction curve and the ground truth curve. The prediction one is drawn by concatenating all the prediction of each day, and it is pretty match the ground truth. In addition, since our prediction is the stock value install of the class label, we cannot use accuracy or precision as our evaluation metric. The Mean Square Error (MSE) is employed as our evaluation metric.

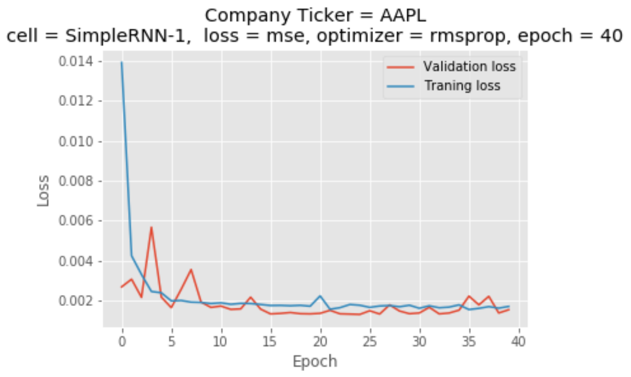

Fig 10: The diagram of loss vs epoch

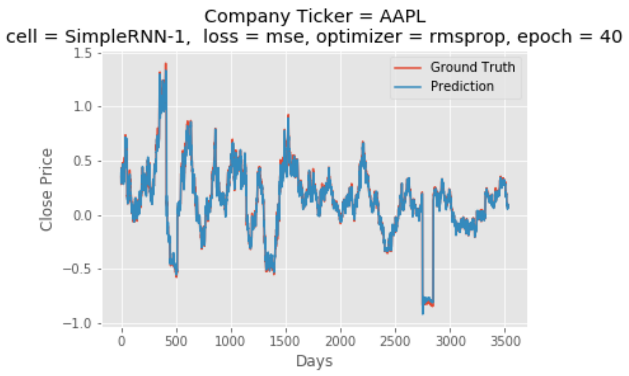

Fig 11: Prediction vs ground truth

Next, the models which are introduced in last section are used to predict next five days' stock prices according to the previous 100 days. And the performance of these models are compared with a classic machine learning method (i.e. linear regression). The results of these tests are shown below in Table 1.

Table 1: The mean square error of the models tested compared to the MSE of the linear regression.

Model	**MSE**
Linear Regression	0.0168
RNN-1	0.0173
RNN-2	0.0183
RNN-3	0.0184
LSTM-1	0.0169
LSTM-2	0.0175
LSTM-3	0.0167
GRU-1	0.0168
GRU-2	0.0170
GRU-3	0.0166
Pretrain-LSTM	0.0186

Discussion

In Table 1, We can find that the LSTM is slightly better than Simple RNN. This may because that the Simple RNN suffers from vanishing and exploding gradients problem. LSTM can solve this problem by using input and forget gates, which has the capability of controlling over the gradient flow. Also, GRU is slightly better than LSTM. It may because that the GRU are simpler, and thus faster to train and perform better than LSTM with less training data. Hence, the NN with 3 GRU layers achieves the best performance among these models. By the way, when training the models, we tried a variety of different optimization algorithms, all of which are in the Stochastic Gradient Descent (SGD) family of algorithms. Specifically, we tried RMSProp, Adam, and AdaGrad, all of which converged to the same general solution. We also found that allow the learning rate to decay yielded better performance.

In addition, since currently we feed the deep learning model by using one single company's stock data, the data may not be big enough to train the model. That is, with limited data, it is hard to train a very deep NN and its performance is difficult to outperform the classic machine learning method, like linear regression.

Since one single company's data is limited, we can use the other companies' data to pre-train the model, and then feed the data of the company which we want to predict to this pre-train model. Ideally, the performance can improve with the pre-train model. By the way, we tried to train a pre-model by using the data of other 3000 and 500 companies separately on ICS server. But they all exceeded the maximum walltime on ICS server and failed. So we only can reduce our training set for pre-train model, it achieves MSE of 0.0186, which is not as good as other mode. It is because the training set for pre-train model is not big enough. So it cannot benefit the generalization and would play a role as noise in the training process. But we believe it is able to achieve better performance if we can use more data to train the model.

Conclusion

In this project, we study a variety of methods to predict the stock price. We examined the performance of classic machine learning method and different RNN models to see their capability of predicting the stock value. Even with limited training data, the RNN model can achieve similar performance as the classic machine learning method. So we think that if we can train a deeper RNN model with more data, it would be likely to outperform the classic machine learning method.

References

Basu, S. (1977). Investment Performance of Common Stocks in Relation to Their Price-Earnings Ratio: A Test of the Efficient Market Hypothesis. *The Journal of Finance*, *32*(3), 663–682.

Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, *2*(1), 1–8. https://doi.org/10.1016/j.jocs.2010.12.007

Britz, D. (2015). Recurrent Neural Networks Tutorial, Part 1 - Introduction to Neural Networks. Retrieved from http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. https://doi.org/10.3115/v1/D14-1179

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, 1–9. Retrieved from http://arxiv.org/abs/1412.3555

Fama, E. F. (1965). The Behavior of Stock-Market Prices. *The Journal of Business*, *38*(1), 34–105. https://doi.org/10.1017/CBO9781107415324.004

Ivakhnenko, A. G. (1970). Heuristic self-organization in problems of engineering cybernetics. *Automatica*, *6*(2), 207–219. https://doi.org/10.1016/0005-1098(70)90092-0

Ivakhnenko, A. G., & Ivakhnenko, G. A. (1995). The review of problems solvable by algorithms of the group method of data handling (GMDH). *Pattern Recognition and Image Analysis*, *5*(4), 527–535.

Längkvist, M., Karlsson, L., & Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, *42*(1), 11–24. https://doi.org/10.1016/j.patrec.2014.01.008

Lawrence, R. (1997). *Using Neural Networks to Forecast Stock Market Prices*. Retrieved from http://people.ok.ubc.ca/rlawrenc/research/Papers/nn.pdf

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. *Proceedings of the ACM International Conference on Multimedia - MM '14*, 675–678. https://doi.org/10.1145/2647868.2654889

Malkiel, B. G. (2003). The Efficient Market Hypothesis and Its Critics. *The Journal of Economic Perspectives*, *17*(1), 59–82.

McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception. *Psychological Review*. https://doi.org/10.1037/0033-295X.88.5.375

Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., & Khudanpur, S. (2010). Recurrent Neural Network based Language Model. *Interspeech*, (September), 1045–1048.

Saad, E. W., Prokhorov, D. V, & Wunsch II, D. C. (1998). Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks*, *9*(6), 1456–1470. https://doi.org/10.1109/72.728395

Sak, H., Senior, A., & Beaufays, F. (2014). Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. *Interspeech 2014*, (September), 338–342.

https://doi.org/arXiv:1402.1128

Tilakaratne, C. D., Mammadov, M. A., & Morris, S. A. (2008). Predicting Trading Signals of Stock Market Indices Using Neural Networks. In W. Wobcke & M. Zhang (Eds.), *AI 2008: Advances in Artificial Intelligence: 21st Australasian Joint Conference on Artificial Intelligence Auckland, New Zealand, December 1-5, 2008. Proceedings* (pp. 522–531). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-89378-3_53

Tsai, C. F., & Hsiao, Y. C. (2010). Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems*, *50*(1), 258–269. https://doi.org/10.1016/j.dss.2010.08.028

Vui, C. S., Soon, G. K., On, C. K., Alfred, R., & Anthony, P. (2013). A review of stock market prediction with Artificial neural network (ANN). In *2013 IEEE International Conference on Control System, Computing and Engineering* (pp. 477–482). Penang: IEEE. https://doi.org/10.1109/ICCSCE.2013.6720012

White, H. (1988). Economic prediction using neural networks: The case of IBM daily stock returns. *Neural Networks, 1988., IEEE International Conference on*, 451–458. https://doi.org/10.1109/ICNN.1988.23959

Yoo, P. D., Kim, M. H., & Jan, T. (2007). Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation. *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, *2*, 835–841. https://doi.org/10.1109/CIMCA.2005.1631572

Zhou, G. B., Wu, J., Zhang, C. L., & Zhou, Z. H. (2016). Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, *13*(3), 226–234. https://doi.org/10.1007/s11633-016-1006-2