Predicting Prices of Stock Market using Gated Recurrent Units (GRUs) Neural Networks

Mohammad Obaidur Rahman, Md. Sabir Hossain*, Ta-Seen Junaid, Md. Shafiul Alam Forhad, Muhammad Kamal Hossen

Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Bangladesh
*Corresponding Author

Summary
Predicting the stock prices is very much challenging job due to the volatility of the stock market. In this paper, we have proposed a model to predict the future prices of the stock market using Gated Recurrent Units (GRUs) neural networks. We have changed the internal structure of GRUs in order to remove local minima problem, reduce time complexity and others problem of stochastic gradient descent as well as improve the efficiency. We used mini-batch gradient descent, is a good trade-off between stochastic gradient descent and batch gradient descent. We evaluated our result by calculating the root mean square error on the various dataset. After extensive experiments on the real-time dataset, our proposed method predicted the future prices successfully with good accuracy.

Keywords:
Stock Market Prediction, Gated Recurrent Units (GRUS) Neural Networks, Artificial Neural Network, and Deep Learning

1. Introduction

The stock market prediction has entranced a lot of focus from business as well as academia. Because of the complex, volatile and non-linear nature of the market, it's too hard to predict. As the stock markets grow bigger, additional people pay surveillance to develop a logical approach for predicting the stock market.

Stock market prediction is needed for the purpose of predicting the long-term value of company stock. The stock market prediction has moved into the technological realm in this era of the digital computer. By taking advantage of modern digital computation and public economic database which permit individuals to discover the hidden info among these platforms. Artificial intelligence (AI) techniques are effective for detecting non-linear structures in the data of the financial market [1]-[2] in the last few years.

We have a tendency to build gated recurrent units (GRU) neural network [3]. GRU is one of the most advanced architectures of deep learning. And by using it, predict future prices. In this paper, our prime concern is to use deep learning and by using it to predict the longer-term value. Surprisingly there has been no notable previous works which used GRU neural networks on a liquid, massive and survivor bias-free stock universe. We used it to check the performance in large-scale financial market prediction tasks. For financial time series prediction, we provide an in details guide on data processing as well as building, training, and testing of GRU networks. We have a tendency to use the root mean square error measurement to evaluate our proposed model. After preparing the model, we have trained the model on training dataset collected from Yahoo Finance [18]. Then, validate the model on a testing dataset collected from the identical supply.

2. Background

2.1 Gated Recurrent Unit (GRU) Neural Networks

GRU [Fig. 1] is capable of learning some dependencies which are introduced by Cho [3]. It is a special kind of RNN (Recurrent Neural Network). They enormously work on a large type of problems. They are one in every of the most fashionable, powerful and effective neural networks. This is expressly designed to avoid the long-term dependency drawback. GRU has fewer parameters than LSTM and so may train a bit quicker or needless iteration to generalize [10]. Rafal et al. showed that the GRU outperformed the LSTM on most tasks with the exception of language modeling [9].

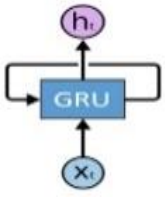


Fig. 1 Rolled GRUs

For every element of a sequence, GRU performs a similar task. That is the reason for which it is called recurrent. And the result being depended on the calculations of the previous. Another think about GRUs is that they have a unit called memory units. The calculated information is captured by it. In Fig. 2, a network, 'GRU' looks at xt input and outputs ht which is also the input of the next step. So a

Manuscript received January 5, 2019
Manuscript revised January 20, 2019

loop permits data to be allowed from one step to the next in the RNN.

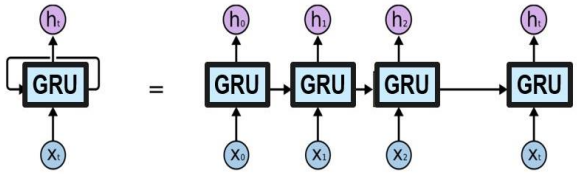


Fig. 2 Unrolled GRUs

Because of the multiple copies of the same network, this neural network is called recurrent. Here, each passing a message to the next network. Recurrent neural networks related to sequences and lists just like stock market data. And that chain-like nature is the reason behind it. To use for extracting the hidden pattern of the stock market, these networks have the natural architecture of the neural network.

2.2 Internal Architecture of GRU

By using reset gate and updates gate t GRU solves a problem of a standard RNN called the vanishing gradient problem. Actually, the reset gate and update gate are two vectors which decide what information should be passed and what information should not be passed to the output. The most important factor is that those can be trained to hold information from long ago. Without remove information or washing it through time which is not relevant to the prediction. With the help of [12], we explain the mathematics behind a single cell of this networks is shown in Fig. 3 and 4.

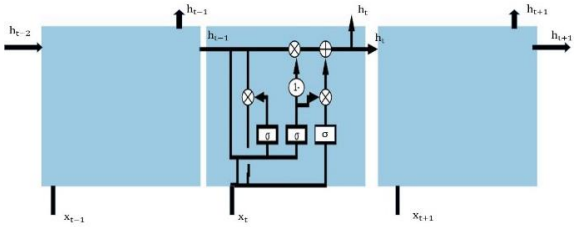


Fig 3. GRUs networks

2.2.1 Update gate

To remove the risk of vanishing gradient problem the update gate helps the network to control how much of the previous information (from past time steps) needs to be passed along to the future which is really effective because the network can decide to recognize all the information from the past [Fig. 5].

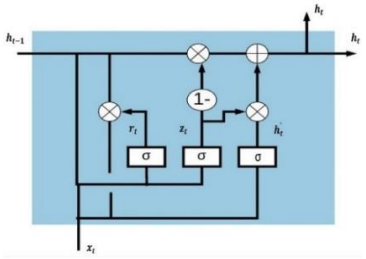


Fig. 4 A single GRUs cell

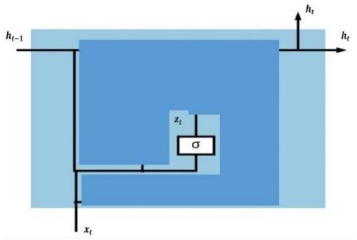


Fig. 5 Update gate

The formula for updating gate is provided in (1).

$$z_t = \sigma\left(W^{(Z)}x_t + U^{(z)}h_{t-1}\right) \quad (1)$$

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \quad (2)$$

Here x_t is the into the network unit, it is multiplied by its own weight $W^{(Z)}$. The same goes for h_{t-1} which holds the information for the previous t-1 units and is multiplied by its own weight $U^{(z)}$. Both results are added together and a sigmoid activation function is applied to squash the result between 0 and 1.

2.2.2 Reset gate

The main purpose of the reset gate in the network is basically to decide how much of the past information to forget shown in Fig. 6. We have reset gate using (2).

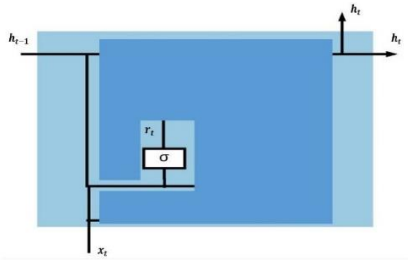


Fig 6. Reset gate

We plug in previous output as input h_{t-1} and x_t, multiply them with their corresponding weights, sum the results and apply the sigmoid function.

2.2.3 Current memory content

This memory content will use the reset gate to store the relevant information from the past. Here we changed our activation function from hyperbolic tangent to sigmoid. Current memory content is calculated using (3).

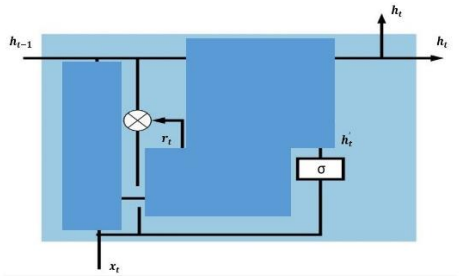


Fig. 7 Current memory content

$$h_t^{'} = \sigma(Wx_t + r_t \cdot Uh_{t-1}) \quad (3)$$

$$h_t = z_t \cdot h_t^{'} + (1 - z_t) \cdot h_{t-1} \quad (4)$$

Here an element-wise multiplication happened between h_{t-1} and r_t line and then sum the result with the input x_t and finally, σ is used to produce $h_t^{'}$.

2.2.4 Final memory at the current time step

Finally, the network needs to determine h_t which is the output of the current unit and passes it down for the next unit. In order to do that the update gate is needed which control what to collect from the current memory content h_t using (4) and what from the previous steps h_{t-1} [Fig. 8].

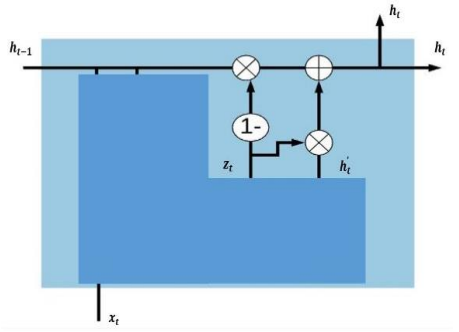


Fig. 8 Final Memory

2.3 Backpropagation

Backpropagation is a backward propagation of errors. It is a technique which is used in deep learning. It determines a gradient and is needed in the computation of the weights to be used in the neural networks. It is generally used by the gradient descent optimization algorithm for the purpose of adjusting the weight of neurons. And it is done by computing the gradient of the cost or loss function.

2.3.1 Loss Function

Here our loss function is a function which is a calculator measures the mean of the squares of the errors and is called mean squared error (MSE).
If $Y_i^{'}$ is n predictions vector and generated from n data points sample on all variables. And if Y_i is an observed values vector of the variable being anticipated. Then the within sample MSE of the predictor is computed using (5).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i^{'})^2 \quad (5)$$

2.4 Mini-batch Gradient Descent

A new approach like mini-batch gradient descent (6), the cost function is averaged over a small set of samples, from around five-five hundred. This is opposed to the size of all the training samples of the batch and the stochastic gradient descent batch size of 1 sample.

$$W = W - \alpha\nabla J(W, b, x^{(z:z+bs)}, y^{(z:z+bs)}) \quad (6)$$

Where weights are denoted by W, the learning rate is represented by α and ∇ denotes the gradient of the cost function $J(W, b)$ in respect of changes in the weights and bs is the mini-batch size.

2.5 Required Tools and Libraries

The programming language we used here is Python 3.7 [13]. For data analysis and data structures purpose, we use Pandas which is an open source BSD-licensed library. We use NumpPy library. And it is used for adding support for large, multi-dimensional arrays, matrices. And for the scientific calculation to operate on these arrays [14]. For collecting historical data at the real-time we use Pandas-datareader. For feature scaling, we use a machine learning library called Scikit-learn [15].
For developing our deep GRU neural networks we used a high-level neural networks API [Keras], written in Python. It is capable of running on top of TensorFlow [16]. To visualize the result we use Matplotlib [17].

3. Related works

A lot of work has been done for predicting stock market trends. Most of those words based on traditional machine learning [4]-[6]. But after better computation power and with lots of data available, deep learning is outclassed and outperformed machine learning in every possible way. Deep learning is a sub-portion of machine learning. It's

alike to how an animal would recognize something, think about it, and then come to a final decision. A deep learning model learns it with its own computing brain by using an ANN [7].

The challenge for time series prediction is also a big factor. There are some researches by using feedforward neural networks like multilayer perception [1] but those networks are not time series model. Feedforward neural networks have done well in classification tasks, however, in a dynamic environment, it needs techniques that account for history. In order to find meaningful statistics and other characteristics of the data, time series analysis is used. Time series model is totally based on the idea which predicts future price behavior.

In [5], authors use the standard RNN to predict the future price. RNN can be used to map input data sequences to output data sequences but practical problems have been detected in training RNN like vanishing gradient problem [11]. Vanishing Gradient Problem occurs when we try to train a Neural Network model using Gradient-based optimization techniques.

Long short-term memory (LSTM) is an excellent neural network to remove the Vanishing Gradient Problem and there are some excellent researches to predict future prices using LSTM like deep learning with long short-term memory networks for financial market predictions [8]. But compare to GRU, LSTM has some redundant information. GRU has fewer parameter than LSTM and thus may train a bit faster or needless iterations to generalize. Writers of the paper 'An Empirical Exploration of Recurrent Network Architectures' showed that the GRU outperformed the LSTM on most tasks with the exception of language modeling [9].

4. Proposed System Architecture

In this Section, we focus on the overall system architecture of the proposed system [Fig. 9] to predict the future prices of the stock market based on an advanced deep neural networks system at real time.

Learning from past history and predict the future is a fundamentally tough challenge. A model may well fit for historical data. But when presented with new inputs, performance is not satisfactory. With gated recurrent neural networks (GRUs), we boost the modeling abilities of artificial neural networks for time series forecasting. With this networks, we also overcome the vanishing gradient problems of recurrent neural networks. When training, we also overcome the local minimum problem of gradient descent by using stochastic gradient descent.

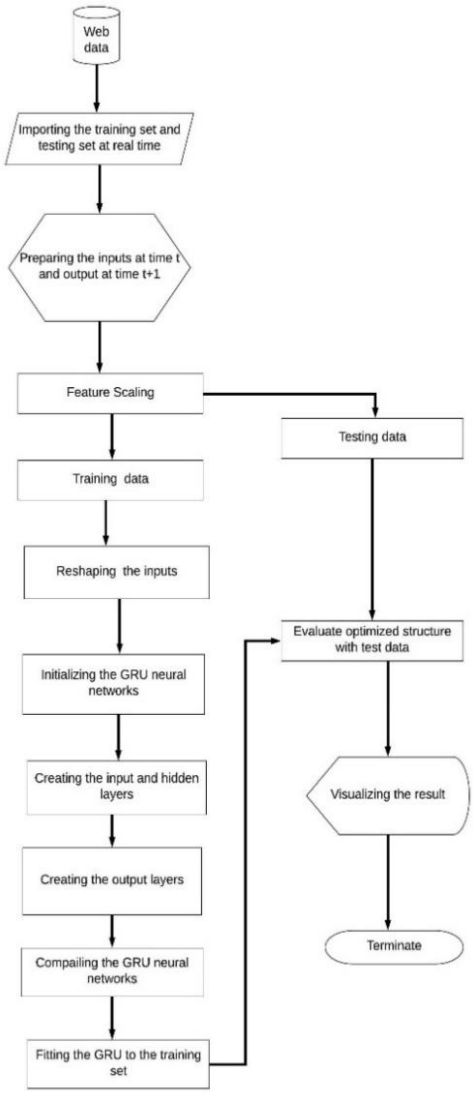


Fig. 9 Overview of Proposed System Architecture

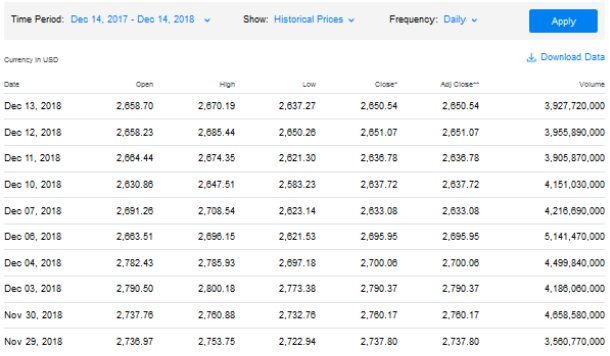
Steps of the proposed encryption algorithm are described in the following:

1) We have to collect data from the web at the real time.
2) We have to select a training set and a testing set.
3) We have to prepare inputs at time t and outputs at time t+1.
4) We have to apply feature scaling on our data so that our data will stay between a range and also between the threshold.
5) After that, we reshape our data so that it has to be into proper form and can be used as the inputs of the networks.
6) Initializes the neural network.
7) Creates the input and hidden layers.
8) Creates the output layer.

9) Compiles the proposed neural networks.
10) Fits our training data into the networks to extract the hidden features.
11) We get the predicted results from the test set.
12) We compare the values with real values.
13) We visualize our data and evaluate the predicted results.

4.1 Web Data Source

The data are collected from Yahoo! Finance at the real-time [Fig. 10]. Yahoo! Finance is a media property that is the part of Yahoo! Network [18].


Time Period: Dec 14, 2017 - Dec 14, 2018 Show: Historical Prices Frequency: Daily Apply

Currency in USD Download Data

Date	Open	High	Low	Close*	Adj Close**	Volume
Dec 13, 2018	2,658.70	2,670.19	2,637.27	2,650.54	2,650.54	3,927,720,000
Dec 12, 2018	2,658.23	2,685.44	2,650.26	2,651.07	2,651.07	3,955,890,000
Dec 11, 2018	2,664.44	2,674.35	2,621.30	2,636.78	2,636.78	3,905,870,000
Dec 10, 2018	2,630.86	2,647.51	2,583.23	2,637.72	2,637.72	4,151,030,000
Dec 07, 2018	2,691.26	2,708.54	2,623.14	2,633.08	2,633.08	4,216,690,000
Dec 06, 2018	2,663.51	2,696.15	2,621.53	2,695.95	2,695.95	5,141,470,000
Dec 04, 2018	2,782.43	2,785.93	2,697.18	2,700.06	2,700.06	4,499,840,000
Dec 03, 2018	2,790.50	2,800.18	2,773.38	2,790.37	2,790.37	4,186,060,000
Nov 30, 2018	2,737.76	2,760.88	2,732.76	2,760.17	2,760.17	4,658,580,000
Nov 29, 2018	2,736.97	2,753.75	2,722.94	2,737.80	2,737.80	3,560,770,000

Fig 10. Web data source

Here every company has a ticker name like 'INTL' for the company name Intel. For every date, that website has opening, highest, lowest, closing, adjusted closing and volume values.

4.2 Real-Time Data Collection

Python's wealthy online connectivity power provide a proper way to fetch data from the Internet, and its capability to save those into CSV files bring a medium for sharing collected data. We use the
pandas_datareader library to collect data at real time.

4.3 Training and Testing Dataset

In deep learning, the models work by developing data-driven decisions. And it is done by using input data to build a mathematical model. We use Pandas library to import training set and testing set.

4.3.1 Training Set

A training data set is a data set of samples used for leaning which means to fit the weights or parameters. Ten years data have been used to train our model.

4.3.2 Testing Set

A test data set is a set of examples used only to evaluate the performance. And the performance is independent of the training dataset. If a model fits the training dataset, it also fits the test data set.

4.4 Preparing the inputs and outputs

Our first task is to prepare the inputs and outputs. Our inputs are opening price, closing price, highest value, and lowest value at time t and output is opening/closing values at time t+1. To do that we shifted our output one timestamp and remove the row containing NAN value. If we have two thousand one rows in our data set that means we have two thousand input vectors each containing the opening price, closing price, highest value, and lowest value at time t. We have output vectors at next time step that means at time t+1 which is basically the opening/closing price of that time.

4.5 Feature Scaling

During the data preprocessing step, feature scaling is generally performed. The main benefit of feature scaling is to avoid dominating attributes greater over smaller numeric ranges. To apply the gradient descent algorithms properly there are some techniques that can be applied to both training set and testing set. If the features applied on input vectors are out of scale then loss space will be somehow stretched and this will make the gradient descent convergence harder or at least slower.

4.6 Method for Feature Scaling

The main adhere are four common methods to perform Feature Scaling.

4.6.1 Min-max normalization

To scale the values between [0, 1], we use min-max normalization (7).

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (7)$$

4.6.2 Mean normalization

To scale the values between [-1, 1], we use mean normalization (8).

$$x' = \frac{x - \text{mean}(x)}{\max(x) - \min(x)} \quad (8)$$

4.6.3 Standardization

After applying those data will have zero mean and unit variance using (9).

$$x' = \frac{x-\bar{x}}{\sigma} \quad (9)$$

4.6.4 Scaling to unit length

Scaling is done using (10) considering the whole feature vector to be of unit length.

$$x' = \frac{\bar{x}}{||x||} \quad (10)$$

4.7 Implementation on collected Dataset

Min-max normalization is used on our data to scale the values between [0, 1]. The formula is given in (11).

$$x' = \frac{x-\min(x)}{\max(x)-\min(x)} \quad (11)$$

As our neural networks need sigmoid activation functions and it scales the values between [0, 1] that is the main reason to scale our data with min-max normalization.

4.8 Reshaping the Inputs

Reshaping is about changing the format of inputs. Before reshaping our input was a two-dimensional array containing the number of observations and features and we have four features. After reshaping our input data is a three-dimensional array as a time step is also included. We implemented this by using the NumpPy library.

4.9 Initializing the GRU Neural Networks

Our data has a continuous sequence. As we are predicting the continuous outcome so we make a regression model. Regression is used when we have continuous outcome and classification is used when we have a categorical outcome. So we called our object regressor and use a sequential class as our data has a continuous sequence.

4.10 Creating the Inputs and Hidden Layers

We changed our hidden layers from traditional GRUs by using sigmoid activation function (12) instead of the hyperbolic tangent (13).

$$A = \frac{1}{1+e^{-x}} \quad (12)$$

$$A = \tanh(x) = \frac{2}{1+e^{-2x}} - 1 \quad (13)$$

As our stock price values never are less than zero and that is the main reason for using sigmoid instead of the hyperbolic tangent. That is why we changed the activation functions. We use Keras to create this model with four input features and one time step using sigmoid activation function.

4.11 Creating the Outputs Layers

We used Keras, Densed class to create the output layers which is opening/closing prices at the next time step.

4.12 Compiling GRU Neural Networks

To compile the GRU neural networks, we first need to know how we train our neural networks. We use BPTT (Backpropagation through Time). And it is used for training our neural networks. In replace of the classical stochastic gradient descent algorithm, adaptive moment estimation algorithm can be used which was presented first by Diederik Kingma. Our loss function is MSE.

4.13 Fitting the GRU to the Training Set

Here we import our inputs and outputs at the neural networks. As we use mini-batch gradient descent, we tested our neural networks on several batch size and find the best result at batch size 32. For good result and good fitting, we iterated our networks 200 times and find a very low lost function which is better for testing.

4.14 Getting the Predicted Result from Test Set

After completing the training, our neural networks learn the hidden patterns of our data, so we test our model at test set. A test data set is a set of samples used only to evaluate the performance that is independent of training dataset but that maintains the same probability distribution as the training dataset. If a model fits the training data set also fits the test data set. We also applied feature scaling at our inputs. Our predicted results are the opening/closing prices of the next day.

4.15 Comparing the Results with Real Values

First, we applied inverse feature scaling on our predicted results. After that, we compare our results with real values.

4.16 Visualizing the Results

We use the pyplot module from matplotlib to plot the real socks price and predicted stocks and visualize the results.

4.17 Evaluating the Results

We have evaluated our proposed model using RMSE (14).

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\left(\frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i')^2\right)} \quad (14)$$

5. Experimental Results and Analysis

In this section, the result obtained from the implementation of the proposed model is analyzed from different perspectives.

5.1 Effect of Epochs to Learn the Patterns

An epoch or iteration is when an entire dataset is passed forward and backward through the neural network only once. Training a deep neural network involves optimizing a large set of parameters which are heavily interdependent. As for this, it can take a lot of scaled training examples before the network even settles into an area of the solution space which is close to the optimal solution or at least, the optimal solution for this training data set. This is exacerbated by the stochastic nature of batch gradient descent as the optimization algorithm is very data-hungry. To summarize, batch gradient descent requires more iterations to converge than one pass over the data set will allow.

To learn the pattern our loss function is mean squared error (MSE) which is an estimator measure the average of the squares of the errors. The less the value of the loss functions, the better our data will fit onto the model and the better our model will learn the patterns.

From Fig. 11, we can see that at fewer epochs we have more loss. But after 150 epochs our loss functions get saturated. We have a pretty low loss function values approximately 1.66E-04. From this, we can conclude that our model will predict the outputs perfectly.

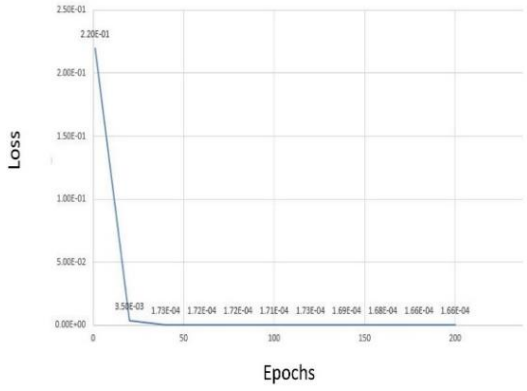


Fig. 11 Loss function values

5.2 Effect of Mini-batch Gradient Descent

In case of complex structure, gradient descent can detect a gradient which might be the gradient of a smaller subpart but that is not the optimal gradient which is known as local minima problem.

At first, recalculating the cost function after each sample. Then, the stochastic gradient descent updates the weight matrix. It can solve the local minimum problem but it takes much computation time.

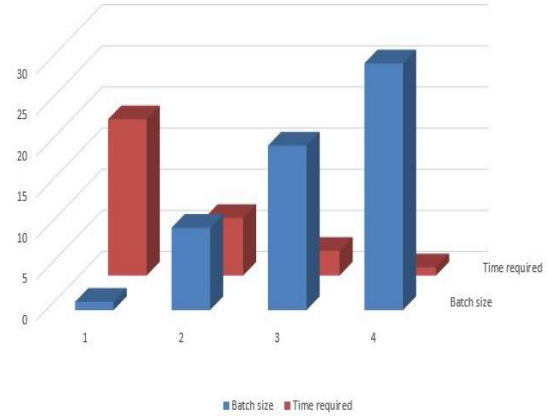


Fig. 12 Effect of batch size

It responds to the effects of each and every sample. And the samples themselves will contain a portion of noisiness. And it will make the result noisy moreover it takes much computation time.

A new approach mini-batch gradient descent technique, the loss function is averaged over a small number of samples, from around five-five hundred.

From Fig. 12, we can see when our batch size is one which is basically stochastic gradient descent, is taking lots of time to compute the networks. And the value mini batch gradient descent which is 30.

5.3 Effect of Training Periods

When the training data set is small, deep neural networks do not perform well. Algorithms of statistical learning cannot learn well from a few examples because of the fundamental principles of their design. From the Fig. 13, we can see when we have 1 year of training data, our model cannot learn properly. The predicted values curve cannot follow the real values curve. From Fig. 13, we can see when we have 5 years of training data, our model can learn properly. The predicted values curve follow the real values curve.

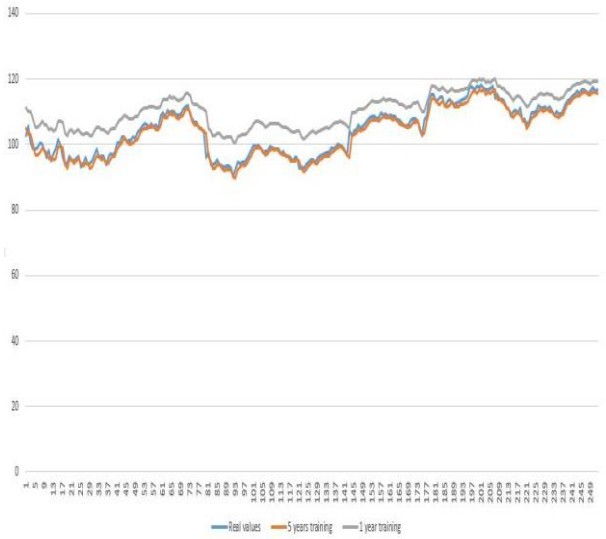


Fig. 13 Effect of training periods

5.4 Efficiency for several companies

Our model is a regression model and regression model is generally evaluated by calculating the root mean square error (RMSE). The comparison of real stock prices and predicted stock prices for Lowe's Companies, Inc. (LOW) is given where the percentage of root mean square error is 0.0127464 [Fig. 14].

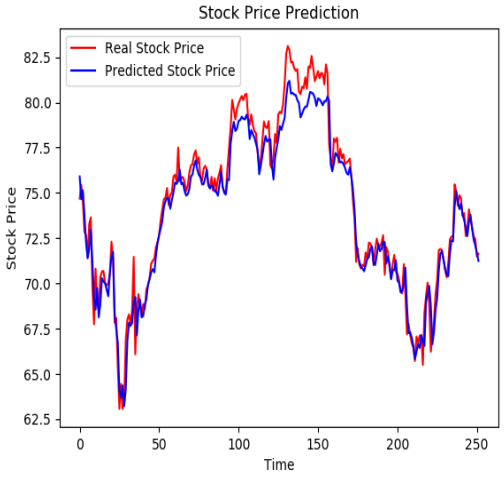


Fig. 14 Predicted prices of Lowe's Companies, Inc.

The comparison of real stock prices and predicted stock prices for The Coca-Cola Company (KO) is given where the percentage of root mean square error is 0.0144508 [Fig. 15].

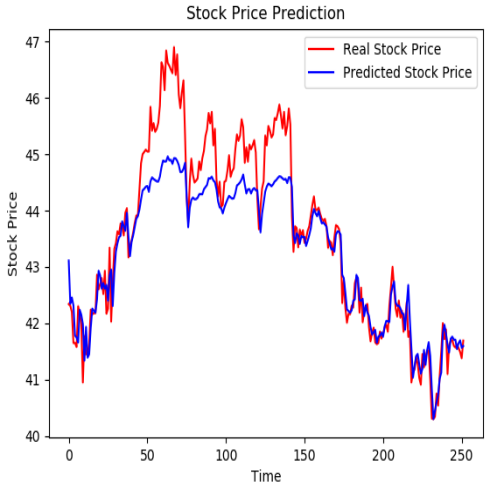


Fig. 15 Predicted prices of The Coca-Cola Company

The comparison of real stock prices and predicted stock prices for Apple Inc. (AAPL) is given where the percentage of root mean square error is 0.013996 [Fig. 16].

6. Conclusion and Future Recommendations

In this paper, we predicted the future prices of the stock market using Gated Recurrent Units (GRUs) neural networks. We predicted the future prices successfully with very good accuracy. The traditional recurrent neural networks have vanishing

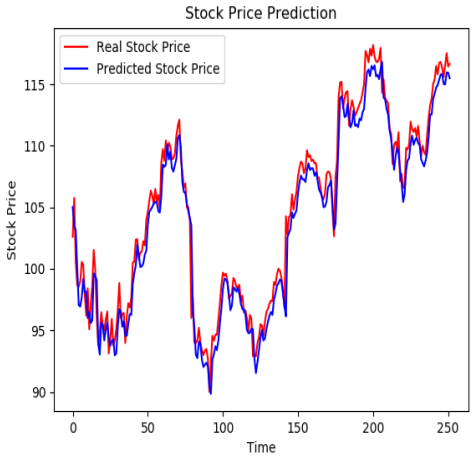


Fig. 16 Predicted prices of Apple Inc.

gradient problems. We overcome those problems by using Gated Recurrent Units (GRUs) neural networks. We also make some small changes inside our Gated Recurrent Units (GRUs) neural networks for more efficiency. We also remove the local minima problem of gradient descent and

time complexity and other problems of stochastic gradient descent. We use mini-batch gradient descent which is a good trade-off between stochastic gradient descent and batch gradient descent.

One of the future implementations of this research will be making smart artificial trading agent. The artificial agent will guide us about when to buy or sell or hold for gaining more profit. Moreover, an automated trading BOTS can save peoples' time and can guide for smart decisions. The proposed system works well for most of the cases but still, it gave bad predictions on some cases. This work may extend to predict far future prices.

References

[1] Guresen, Erkam, et al. "Using Artificial Neural Network Models in Stock Market Index Prediction." Expert Systems with Applications, vol. 38, no. 8, 2011, pp. 10389–10397., doi:10.1016/j.eswa.2011.02.068.

[2] Dixon, Matthew, et al. "Implementing Deep Neural Networks for Financial Market Prediction on the Intel Xeon Phi." Proceedings of the 8th Workshop on High Performance Computational Finance - WHPCF '15, 2015, doi:10.1145/2830556.2830562.

[3] Cho, et al. "Learning Phrase Representations Using

[4] RNN Encoder-Decoder for Statistical Machine

[5] Translation." Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation, 3 Sept. 2014, arxiv.org/abs/1406.1078.

[6] Kazem, Ahmad, et al. "Support Vector Regression with Chaos-Based Firefly Algorithm for Stock Market Price Forecasting." Applied Soft Computing, vol. 13, no. 2, 2013, pp. 947–958., doi:10.1016/j.asoc.2012.09.024.

[7] Bernal, Armando, et al. "Financial Market Time Series Prediction with Recurrent Neural Networks." Stanford University, 14 Dec.2012,cs229.stanford.edu/proj2012/BernalFokPidaparthi

[8] Chen, Yingjun, and Yongtao Hao. "A Feature Weighted Support Vector Machine and K-Nearest Neighbor Algorithm for Stock Market Indices Prediction." Expert Systems with Applications, vol. 80, 2017, pp. 340–355., doi:10.1016/j.eswa.2017.02.044.

[9] Lecun, Yann, et al. "Deep Learning." Nature, vol. 521, no. 7553, 2015, pp. 436–444., doi:10.1038/nature14539.

[10] Fischer, Thomas, and Christopher Krauss. "Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions." European Journal of Operational Research, vol. 270, no. 2, 2018, pp. 654–669., doi:10.1016/j.ejor.2017.11.054.

[11] Józefowicz, Rafal et al. "An Empirical Exploration of Recurrent Network Architectures." ICML, 2015.

[12] Olah,.Christopher.."Understanding.LSTM.Networks.".Colah's.Blog,.colah.github.io/posts/2015-08-Understanding-LSTMs/.

[13] Bengio, Y., et al. "Learning Long-Term Dependencies with Gradient Descent Is Difficult." IEEE Transactions on Neural Networks, vol. 5, no. 2, 1994, pp. 157–166., doi:10.1109/72.279181.

[14] Kostadinov, Simeon. "Understanding GRU Networks – Towards Data Science." Towards Data Science, Towards Data Science, 16 Dec. 2017, towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

[15] "Welcome to Python.org." Python.org, www.python.org/.

[16] "NumPy." NumPy - NumPy,.www.numpy.org/.

[17] "Scikit-Learn." Scikit-Learn: Machine Learning in Python - Scikit-Learn 0.20.0 Documentation, scikit-learn.org/stable/.

[18] "Keras: The Python Deep Learning Library." Keras Documentation, keras.io/.

[19] "Installation." Matplotlib: Python Plotting - Matplotlib 3.0.0 Documentation, matplotlib.org/.

[20] "Yahoo Finance - Business Finance, Stock Market, Quotes, News." Yahoo! Finance, Yahoo!, finance.yahoo.com/.

Mohammad Obaidur Rahman received the B. Sc. Engineering Degree in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh in 1998. He has currently received M. Engineering degree from the Department of CSE, Chittagong University of Engineering and Technology (CUET). From September 2001 to onwards, he has been serving as a faculty member in the Department of CSE, CUET, Bangladesh. He is currently working toward the Ph.D. degree as a part-time basis in the Department of CSE, CUET, Bangladesh. His major research interests include cognitive radio networks, game theory, neural networks, Internet security, and cryptography, etc.

Md. Sabir Hossain received Bachelor degree in Computer Science and Engineering from Chittagong University of Engineering & Technology (2015) with outstanding result. He is now pursuing his M.Sc. degree in Computer Science and Engineering from the same university. His current research interests are Machine Learning, Data Mining, Big data, and Software Engineering. Now, he has been serving as a faculty member in the Department of Computer Science & Engineering (CSE), Chittagong University of Engineering & Technology (CUET), Bangladesh.

Ta-Seen Junaid recently completed his graduation in Computer Science and Engineering from Chittagong University of Engineering and Technology. He is working as "Software Engineer" in BJIT Limited from January 01, 2019. His works focuses specially on Artificial Intelligence. His field of interests are Data Science, Computer Vision, Natural Language Processing, Deep Learning, Artificial Neural Networks, Data Structure and Algorithms.

Md. Shafiul Alam Forhad completed B.Sc. Engg. degree in Computer Science and Engineering from Chittagong University of Engineering & Technology (CUET) in 2014 with outstanding result. He is now pursuing his M.Sc. Engg. degree in Computer Science and Engineering from the same university. His current research concerns are Cryptography, Machine Learning, and Data Mining. He was a lecturer in the Department of Computer Science and Information Technology of Patuakhali Science & Technology University. Now, he has been serving as a faculty member in the Department of Computer Science & Engineering (CSE), Chittagong University of Engineering & Technology (CUET), Bangladesh.

Muhammad Kamal Hossen has received his B. Sc. and M. Sc. in Computer Science & Engineering (CSE) degrees from the department of Computer Science & Engineering of Chittagong University of Engineering & Technology (CUET), Bangladesh in 2005 and 2015, respectively. He is now pursuing his Ph. D. degree in CSE from the same university. Since 2006, he has been serving as a faculty member in the Department of CSE, CUET. His research interests include digital image processing, cryptography, steganography, pattern recognition, and data mining, etc.