Improved big data stock index prediction using deep learning with CNN and GRU

Jithin Eapen

Department of Computer Science

California State University

Fullerton, California 92834

Email: jithin.john@csu.fullerton.edu

Abhishek Verma *

Department of Computer Science

New Jersey City University

Jersey City, NJ 07305

Email: averma@njcu.edu

* Corresponding author

Doina Bein

Department of Computer Science

California State University

Fullerton, California 92834

Email: dbein@fullerton.edu

Abstract: Stock index prediction has been a challenging problem due to difficult to model complexities of the stock market. More recently Deep learning approaches have become an important method in modelling complex relationships in time-series data. In this paper: we propose novel deep learning models that combine multiple pipelines of convolutional neural network and uni-directional or bi-directional gated recurrent units. Proposed models improve prediction performance and execution time upon previously published models on large scale S&P 500 dataset. We present several variations of multiple and single pipeline deep learning models based on different CNN kernel sizes and number of GRU units.

Keywords: ***Stock prediction; S&P500; CNN; GRU; Deep learning***

Reference to this paper should be made as follows:. J. Eapen, A. Verma, and D. Bein (2020) 'Improved big data stock index prediction using deep learning with CNN and GRU', *Int. J.of Big Data Intelligence*

Biographical notes:

Jithin Eapen received his MS in Computer Science, from California State University, Fullerton. His research interests are in deep learning, and adaptive machine learning.

Abhishek Verma received his Ph.D. in Computer Science from New Jersey Institute of Technology, NJ, USA. He is presently Associate Professor of Computer Science at New Jersey City University, NJ, USA. His research interests are within the broad area of data science, big data, deep learning, and machine learning.

Doina Bein received her Ph.D. in Computer Science from University of Nevada, Las Vegas, NV, USA. She is presently Associate Professor of Computer Science at California State University, Fullerton, CA, USA. Her research interests are within the broad area of parallel and distributed computing, algorithms, and machine learning.

---

1. INTRODUCTION

The stock market time series data is considered as noisy and dynamic [1]. It is considered noisy as we are not able to model the various factors affecting the stock data like news events or rumours, and hence the effect of these factors gets considered as noise in the data. As the data is non-stationary, representations on the older time-series data may not hold valid to current data. Thus, it becomes very difficult to model and make correct predictions on such moving and noisy data.

In this research paper, we have proposed deep learning based models utilizing single and multiple pipelines of convolutional neural networks with uni-directional or bi-directional gated recurrent units (GRU). Our work improves upon earlier deep learning models [2].

This paper is organized as follows: In section 2, we present the background on stock market analysis and how it is used for automated trading. We describe traditional machine learning algorithms used for stock analysis, and their limitations. Next, we describe various types of layers used in deep learning and listed their advantages. We have also given a small summary of earlier work done by authors on using multiple pipelines of deep learning for various applications. In section 3, we discuss the dataset used, and the reason we selected it. Section 4 is the research methodology, where we describe how we construct, train and make predictions with various deep learning models. In Section 5, we present hardware and software systems that we have utilized for our research work. Section 6 presents the results obtained from each model in terms of mean squared error. We also plot the predicted

trend diagrams from the best performing models. We conclude in Section 7 by summarizing our work, key observations, and analysis.

2. BACKGROUND WORK

2.1 Stock Market Analysis

Attempting to model and predict stock prices has been an important area of interest for traders, investors, and financial institutions. Various models for stock analysis have been developed using both fundamental and technical analysis methods. Fundamental analysis focuses on the business aspects of the company, like profit, cash flow, sales, or news regarding business and political environment etc., while technical analysis uses maths and statistics-based approach to analyse historical data and predict future data. Technical analysis is based on the philosophy that all factors affecting the price, whether fundamentally, politically or otherwise, is reflected in the price itself. Therefore, prices move in trends, and predictions can be based on studying trend charts [3].

2.1.1 Automated Trading

Stock Analysis is also used in computerized trading systems, which automate the process of making sell and buy decisions. Earliest attempts to automate trade involved creation of electronic trading exchanges, which determined the trading decisions based on the theoretical predicted prices which is searched from a look-up table using an index, consisting of a range of current market prices [4]. Advances in the field of machine learning led to its usage in the automated trading systems, where the buy/sell orders are based on pre-set search parameters that are optimized using backtest data. The data feed is received from real-time and historical trading data sources [5]. Such systems expose their APIs for use by brokers, banks, and other institutions to make real-time and immediate trading decisions.

2.1.2 Traditional Machine Learning Algorithms

Researchers have developed decision support systems for dealing with stocks utilizing neural networks and genetic algorithms to make buy and sell decisions for Tokyo based stock indexes [6]. Support Vector Machines emerged as a popular technique due to its ability to recognize patterns in a time-series dataset [7] and learn features from multi-dimensional data using kernel functions that creates a linear hyperplane in the multi-dimensional space [8]. Work done by [9] showed the suitability of support vector machines in forecasting financial time-series data, which outperformed backpropogation based neural networks on criteria like mean square error and others. The kernel methods used by support vector machines scale quadratically i.e. $O(n^2)$ for a training set of size n, as it is implemented using quadratic programming [8].

2.2 Deep Learning

Deep Learning refers to the arrangement of multiple hidden layers of neurons in between the input and output layers. This increases the capability of the network to learn more complex functionalities. It relies on the algorithms of Feed forward and Backpropagation, which fine-tunes the weights, using an optimization algorithm, the objective is to minimize a loss function, or error in output compared to the actual value.

2.2.1 Recurrent Neural Network

Recurrent Neural Networks (RNN) is a type of Artificial Neural Network (ANN) used specifically to learn from sequential data and can be used to predict successive elements in a sequence. In RNNs, the activation on the hidden units for time-step *t-1* is used together with the new input data at *t*, and both influence the activation at time step *t* [10]. This is done by copying the activation of the hidden layer at time-step *t-1* and giving it back as input in time-step *t*. Thus, the new prediction depends upon the historical data, which is similar to our task of predicting stock price index. Using a deep hierarchy of RNNs has been shown to naturally capture the structure of time series data [11].

2.2.2 Bi-directional Recurrent Neural Network

Bi-directional Recurrent Neural Networks are used in the tasks where the prediction depends on the entire input sequence [12]. It consists of layers which combine the unit that learns from the start of the sequence moving forward, with the unit moving backwards from the end of the sequence. This makes it useful to learn from longer sequences of time-series data and makes the output sensitive to the input at time *t*, without using a fixed-sized window around the time *t*, as used in CNNs or a regular RNN with fixed look-ahead buffer [8].

2.2.3 Long-Short Term Memory

Long Short Term Memory (LSTM) units were created to solve the problem of vanishing gradients in recurrent neural networks [13]. In RNNs, due to the long connections, the backpropagation of gradients from the output layer can become smaller and smaller by the time it reaches the initial layers, due to successive derivative operations over activation functions like tanh and sigmoid. This prevents the RNN from learning longer sequences. LSTMs include gates that encode the long-term dependencies existing in sequential data as a simple identity function of 1 or 0 [14]. The three gates called input, forget, and output control the cell states of the units using tanh and sigmoid activations and regulate the information that flows from one time-step to the next.

2.2.4 Gated Recurrent Units

Gated Recurrent Units (GRU) are another variant of RNNs that utilizes two gates called reset and update to control the state units. The key difference between LSTM and GRU being that GRU exposes it memory state fully, while LSTM controls it using output gate. An empirical comparison [15] of traditional RNNs (having tanh units), LSTMs, and

GRUs, was done for sequence modelling tasks like polyphonic music and speech signal modelling. The designed network took 20 consecutive samples of speech signals and predicted the next 10 samples. They observed that LSTM and GRU units clearly out-performed simple RNNs, but they couldn't conclude which among the gated units performed better.

2.2.5 1-Dimensional Convolutional Neural Networks

1-dimensional CNNs can be used as an alternative to RNNs for learning from temporal sequence data, but they are shallow and can learn only from the neighbouring input data because of the convolution operation [8]. However, in the report [16], authors have argued the case for using CNNs in sequence modelling by highlighting the capability of multi-layer CNNs to capture hierarchical representations of input sequence, which means that, the input data that are distant, can interact at the higher layers, and the closer elements interact at the lower layers. This provides shorter paths for dependencies as compared to the chain structures in RNNs. The RNNs due to its longer dependencies have lower efficiency.

2.2.6 Multiple pipeline deep learning models

Instead of a single sequence of layers, as typical in a deep learning model, prior research has used multiple pipelines of layers working on the same input, and learns independently to give different outputs, which is then concatenated as a single output. In [17] a multi-pipeline deep learning model, created with four parallel sequences of layers, performed better than the single pipeline model for the task of classifying text for sentiment analysis.

Another fusion of two deep CNN models was used for improving the prediction of Image Action Recognition [18]. The fusion was done by concatenating two to three layers, consisting of CNN layers followed by a 100 dimensional fully connected layer. Analysing the results showed that the accuracy increased by 2% for each addition of deep CNN layers to the model.

In our earlier work [2], we observed that multiple pipelines of LSTM or Bi-directional LSTM resulted in better predictions of stock indexes as compared to the equivalent single pipeline model.

Design principles have been proposed in [19] that recommend improving the prediction of deep learning models by balancing the increase in both the width and depth of the network. This was shown to give better performance in computer vision tasks with a marginal increase in computational cost, as compared to longer monolithic networks.

3. DATASET DESCRIPTION

As deep learning requires a large dataset for training, we needed a historical dataset of daily stock prices for multiple years. Stock prices are either analysed for each single company registered in the market, or as a combined index, which is the weighted average for a selected group of companies in the market. The S&P 500 index is a combined market-capitalization-weighted index for a collection of 500 large companies and is an excellent measure of the performance of overall U.S. stock market. There are other common indices like Dow Jones Industrial Average, Nasdaq composite, FTSE 100 etc., but we have selected S&P 500 because of its wide availability. Analysis work done [20] showed that the S&P 500 performs better on ARIMA models than London Stock Exchange. The S&P 500 dataset is available for download on Yahoo Finance [21]. It consists of the daily market details like opening and closing price, highest and lowest price, adjusted closing price, volume of trade. We decided to use the daily closing price for our prediction, as the final value for each day.

Experiments done in the paper [22] show the impact on the performance of deep learning networks, by increasing the data size. Performance is shown to increase logarithmically for larger datasets. We decided to use as large a dataset as possible for training our models, depending upon available time and hardware resource. Figure 1 shows the daily closing price of S&P 500 index from January 2, 2008 to November 27, 2018. It is a typical Time-Series based data with sequences of discrete prices recorded during the closing time for each successive day and plotted as a line chart. It has the data for each working day when the market was opened for the selected period of 10 years.

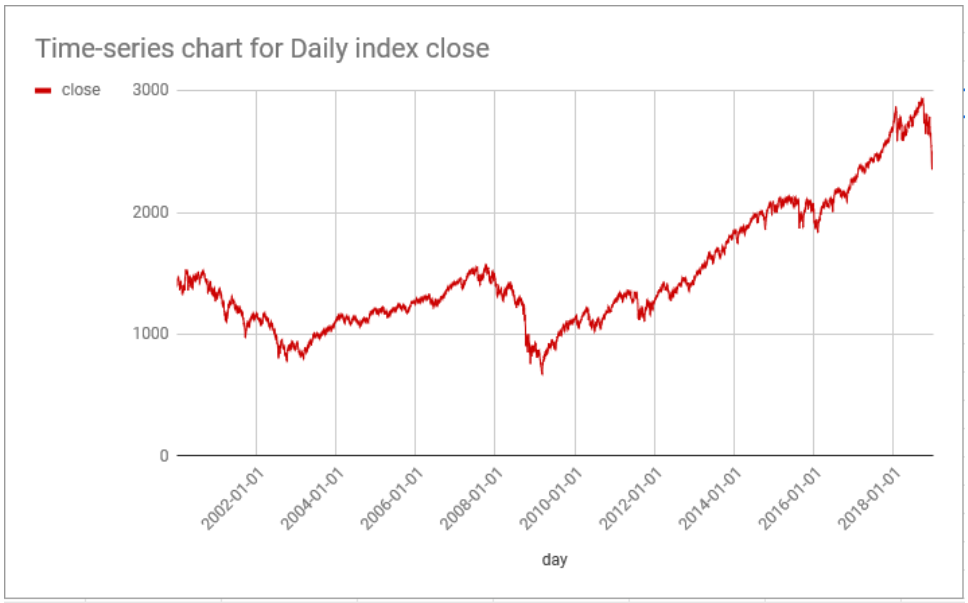

Figure 1: Yahoo Finance S&P 500 Index dataset from 2000 to 2018

4. RESEARCH METHODOLOGY

To find the best performing deep learning models, we created different types of architectures using CNNs and GRU layers. We have utilized both the single pipeline and multi-pipeline methodology.

We use 1-Dimensional CNNs as it is capable of extracting features from sequential data using hierarchical representations. Gated Recurrent Unit networks are also useful for learning from temporal time sequences like stock data. We arrange these layers in different orders and combinations, such as, CNN layers followed by GRU layers. Firstly, we attempt to improve the performance by adding different combinations of Batch-Normalization and reduce overfitting by including Dropout layers. Secondly, we construct several models by varying the number of layers.

4.1 Single Pipeline Deep Learning Model with CNN and Uni-directional or Bi-directional GRU layers

We construct single pipeline model using groups of 1-dimensional Convolutional layers, that is followed with bi-directional or uni-directional gated recurrent unit layers. The CNN layers utilize the ReLU function for its activation and are followed by a Max-pooling of size 2.

Figure 2 shows the architecture of the single pipeline CNN and GRU model. Dropout layers are added after GRUs to reduce overfitting. A Dense layer is used to combine the outputs from the GRU units, and utilizes a linear activation, that gives a numerical output as the predicted stock price index. We use this general architecture on both uni-directional GRU and bi-directional GRU units. We then compare the results of single pipeline models with other multiple pipeline deep learning models.

We use Grid-Search to find the optimal parameters for these models. We tried different Kernel sizes for the convolutional filters like 5, 7 & 9. We also varied the number of GRU units as either 200 or 400. The Grid-search is performed across 5 folds of cross-validation. Grid search utilizes early stopping of the Adadelta optimizer to prevent overfitting. The optimizer algorithm iterates for 100 epochs. We also tried few models with Batch-Normalization to allow for higher learning rates during optimization. A sequence of stock prices for 50 consecutive days is fed to the network and the convolutional operations are done using strides of 3 in one dimension.

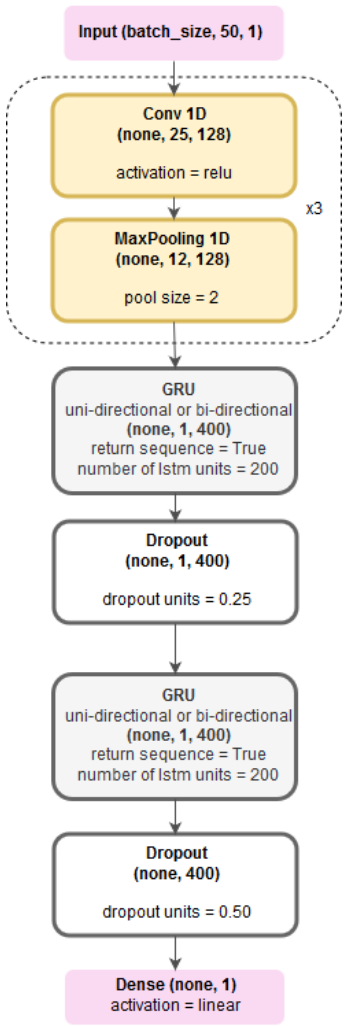

Figure 2: Single pipeline model with uni-directional or bi-directional GRU

4.2 Proposed Multiple Pipeline model with uni-directional or bi-directional GRU

To improve the prediction, we utilized the multiple pipeline deep learning architecture. It uses three parallel layers as shown in figure 3. Each pipeline is made up of an initial group, of 1-dimensional CNN and Max-pooling layers. The CNNs are used to extract the higher-level hierarchical relations in the sequential data, before passing to the GRU layers. The CNNs are followed by another group consisting of GRU layers with dropouts. Similar to the single pipeline model, we tried both the uni-directional GRU and bi-directional GRU layers. The Bi-directional GRU units can learn from both the forward and backward sequence of data and concatenates the output of the two sequences.

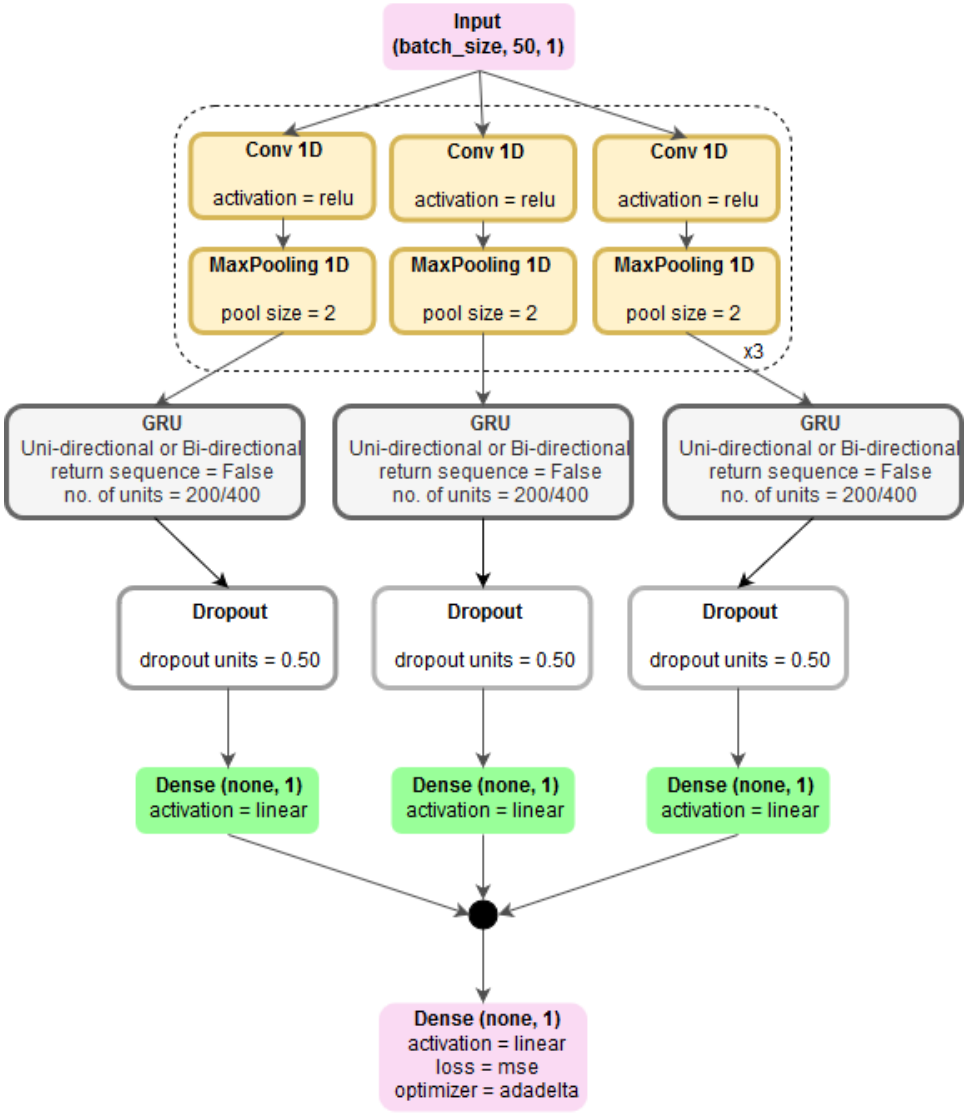

Figure 3: Proposed multiple pipeline model with uni-directional or bi-directional GRU

Within each pipeline, the sequences of 50 input stock indexes are fed to the CNN layers, which utilizes the kernel filters to learn from data of various sequence length. Thus, each individual pipeline within the multiple pipe line setup works on a specific sequence length of data, optimal sequence length is chosen via experimentation. The max-pooling layers then reduces the size of the sequence, before passing it to the GRU layers. A Dense layer reduces the sequence to a single value using the linear activation function. The output from each individual pipeline is concatenated to give a final value. We have used Adadelta as the algorithm for optimizing the weights, which is a form of gradient descent algorithm.

Grid Search is used to find the best parameters for each architecture, and the top 4 models, including the weights and parameters, are saved to be used later for prediction and plotting.

4.3 Comparison of Models

The best performing model is selected for each architecture, based on its mean squared error. The model is used to make predictions, which are then plotted along with the actual value of the stock price index. A sliding window of length 50 days is used as the input to the model, to predict the next 7 days, which is then plotted along with the actual value.

In order to compare the performance time of each model, we noted down the number of weights created in each deep learning model. We also recorded the amount of time required for training the models, as well as for performing the predictions for the entire dataset. To ensure that the execution time is measured under similar conditions, we recorded the time for each model, on the same CPU and GPU, with no other execution load running on the machine.

5. EXPERIMENTAL ENVIRONMENT

5.1 Hardware Environment

We have used high performance GPU based computer server for training and prediction with the deep learning models. Server is equipped with NVIDIA GeForce GTX 1080 Ti GPUs. GPUs allow for parallel processing of matrix operations involved in the deep learning process. Multiple GPUs allow saving time in the experimentation process by running multiple training and prediction jobs simultaneously. The GPUs use the CUDA toolkit v9.0.176 library for executing the parallel threads. The server has multiple Intel Xeon E5-2630 CPUs, having a 256 GB RAM and running at 2.20 GHz.

5.2 Software Environment

The code was developed in the Python language due to ease of coding and good support for deep learning libraries. Out of the large collection of easily available deep learning libraries in Python, we utilized the following -

- Keras [23] is an encapsulating library, with simple, abstract interfaces for construction and training of deep learning models, utilizing back-end framework as Tensorflow. We have used it for defining sequential deep learning pipelines, and specifying the layers, its hyper-parameters and the optimizer algorithms and loss functions.
- Scikit-Learn [24] is a popular library for machine learning tasks. We have used it for preprocessing of data, and for performing the grid-search using cross validation.

- Matplotlib [25] has been used for plotting the various charts to compare the prediction of trend directions.
- Pandas [26] is a data frame based library that has been used for performing disk read and write operations, and especially for importing the dataset from csv file.
- Pickle [27] has been used for serializing and de-serializing to save and load models from the disk in the form of byte streams.
- Numpy [28] library is most suitable for performing various matrix operations, and it is used by Keras for its deep learning operations. We have utilized it for initializing the Grid-Search library with hyper-parameter ranges.

Server login sessions were maintained for each executing model using a linux based command client tool: tmux or terminal multiplexer [29]. It allows multiple SSH sessions to be maintained in parallel for the same user. Each session can be individually connected to, at a later stage to check the results at convenience.

6. Experimental Results and Discussion

Using grid search with five-fold cross validation, we present the top four best performing models and parameters, for single and multiple pipelines of uni-directional GRU and Bi-directional GRU. Grid search library presents the results in the form of rank test score, mean test score, etc. We use rank test score, to sort the models according to their performance rank.

6.1 Results for Single Pipeline Model with CNN followed by Bidirectional LSTM [2]

Table 1 below, shows the results of an earlier work [2], for the single pipeline deep learning, using CNN & Bi-directional LSTM layers, giving a mean squared error in the order of 10^{-4}.

Table 1: Results from four best performing single pipeline deep learning models with CNN and bi-directional LSTM on S&P dataset [2]

LSTM Units	Kernel-size	Mean Train Score	Mean Test Score
200	9	0.000272510	0.000345951
200	5	0.000324599	0.000429588
400	9	0.000333719	0.000431591
400	5	0.000466796	0.000555343

Note: Mean Test Score and Mean Train Score are computed from Mean Squared Error. Lower values are better.

6.2 Results for Single Pipeline Model with CNN followed by uni-directional GRU

Table 2 below, shows the four best performing models for single pipeline deep learning, using a combination of CNN and uni-directional GRU layers. It results in a mean squared error in the order of 10^{-4}. The mean squared error values while using uni-directional GRUs in this research are found to be marginally better than that of Bi-directional LSTMs [2], for the top four models, as well as while comparing the test scores for the same parameter values of Kernel size and number of units. Additionally, this architecture shows low variance and low overfitting as the difference between scores for test and train is not too large.

Table 2: Results from four best performing single pipeline deep learning models with CNN and uni-directional GRU on S&P dataset

GRU Units	Kernel-size	Mean Train Score	Mean Test Score
200	9	0.000254573	0.000318286
400	9	0.000284881	0.000351896
400	5	0.000274086	0.000374044
200	5	0.000413138	0.000505617

Note: Mean Test Score and Mean Train Score are computed from Mean Squared Error. Lower values are better.

6.3 Results for Single Pipeline Model with CNN followed by Bi-directional GRU

Table 3 shows the best models for the single pipeline deep learning architecture using CNN and Bi-directional GRU layers. The best performing model, with number of GRU units 400 and Kernel size 9, outperforms the best performing model of single pipeline architecture using GRU and Bi-directional LSTM with number of LSTM units 200 and Kernel size 9 in Table 1.

Table 3: Results from four best performing single pipeline deep learning models with CNN and Bi-directional GRU on S&P dataset

GRU Units	Kernel-size	Mean Train Score	Mean Test Score
200	9	0.000250285	0.000307375
200	5	0.000350026	0.000448220
400	9	0.000347459	0.000458256
400	5	0.000390678	0.000470128

Note: Mean Test Score and Mean Train Score are computed from Mean Squared Error. Lower values are better.

6.4 Results for Multiple Pipeline Model with CNN followed by Bi-directional LSTM [2]

Table 4 below, shows the results for the multiple pipeline deep learning model consisting of CNN and Bi-directional LSTM layers [2]. The kernel size of 9 for the CNN layer was found to perform the best in combination with the Bi-directional LSTM layers having 200 units. Mean squared error results are in the order of 10^{-4}. The combination of multiple pipelines of layers results in lower mean squared error as compared to the single pipeline CNN and Bi-directional LSTM model.

Table 4: Results from four best performing multiple pipeline deep learning models with CNN and bi-directional LSTM on S&P dataset [2]

LSTM Units	Kernel-size	Mean Train Score	Mean Test Score
200	9	0.000204378	0.000281317
400	5	0.000284712	0.000354261
200	5	0.000234630	0.000359209
400	9	0.000345472	0.000420501

Note: Mean Test Score and Mean Train Score are computed from Mean Squared Error. Lower values are better.

6.5 Results for Proposed Multiple Pipelines Model with CNN followed by uni-directional GRU

As observed from Table 5 below, the multiple pipeline model with CNN and uni-directional GRU layers, performs better than its corresponding single pipeline version in Table 2.

Table 5: Results from four best performing proposed multiple pipeline deep learning models with CNN and uni-directional GRU on S&P dataset

GRU Units	Kernel-size	Mean Train Score	Mean Test Score
400	9	0.000227808	0.000306813
200	5	0.000279954	0.000385774
200	9	0.000345363	0.000444714
400	5	0.000431785	0.000543781

Note: Mean Test Score and Mean Train Score are computed from Mean Squared Error. Lower values are better.

6.6 Results for Proposed Multiple Pipelines Model with CNN followed by Bi-directional GRU

Table 6 below, shows the result for the proposed multiple pipeline model with CNN and bi-directional GRU layers. It has the best performance among all the models and architectures tested. It also shows less difference between Test and Train score for the best 3 models. The results fall in line with those of single pipeline model with CNN and Bi-directional GRU layers and outperforms the corresponding multiple pipeline architectures, i.e., CNN + uni-directional GRU and CNN + Bi-directional LSTM models. Thereby, it combines the strength of multiple pipelines with Bi-directional GRU units.

Table 6: Results from four best performing proposed multiple pipeline deep learning models with CNN and Bi-directional GRU on S&P dataset

GRU Units	Kernel-size	Mean Train Score	Mean Test Score
200	9	0.000213832	0.000278786
400	9	0.000264489	0.000332561
200	5	0.000312285	0.000378146
400	5	0.000350006	0.000455777

Note: Mean Test Score and Mean Train Score are computed from Mean Squared Error. Lower values are better.

6.7 Comparison of execution time

We also noted the execution time for each architecture and model. Table 7 below, shows the time in seconds to complete the training and prediction for the best model in each architecture, using sequences of size 50. As expected, multiple pipeline models take much more time than single pipeline models. This is due to the much larger number of weights that need to be trained in each pipeline using the Adadelta gradient descent algorithm. However, various individual pipelines in a multiple pipeline model could be run in parallel and thus will come close to single pipeline model in terms of execution time. The number of weights used for each model are shown for comparison in Table 8. There are more weights in the LSTM units, as compared to GRU units, this results in lesser execution time for the GRU based models.

Table 7: Comparison of execution time in seconds (including fitting and prediction) for various model on S&P 500 dataset

Architecture	CNN followed by GRU	CNN followed by Bi-directional LSTM	CNN followed by Bi-directional GRU
Single pipeline	173.11	322.18	261.68
Multiple pipeline	360.91	567.97	470.11

Table 8: Comparison of number of weights in each model

Architecture	CNN followed by GRU	CNN followed by Bi-directional LSTM	CNN followed by Bi-directional GRU
Single pipeline	1,489,337	2,923,537	2,397,937
Multiple pipeline	4,730,023	6,869,623	6,014,023

6.8 Comparison of Results across Models

The trend directions were plotted for each deep learning model as shown in figures 4 to 7, X axis representing the time period of the first six months of the year 2016 and 2017. The actual stock index has been plotted in blue line, and the predictions of seven days each are plotted in various colours. The plotted lines can be used to predict if the stock index will rise or fall for the future seven days. Ideally, these predictions should be as close to the actual line in terms of direction and actual value.

Among the various deep learning models shown in figures 4 to 7, they vary in capabilities to give good predictions. The multiple pipeline deep learning models utilizing

Bi-directional GRU, as shown in Figure 7 is the most reliable. All the multiple pipeline models as shown in Figure 6-7, give predicted values closer to the actual stock price line as compared to the single pipeline models in Figures 4-5. The trend directions are more reliable for the models using Bi-directional units (whether single or multiple) as compared to the models with uni-directional units.

As shown in Figure 4 below, the single pipeline model with GRU gives correct predictions when the actual price index is going through a stable phase, as compared to when the stock price index is fluctuating. The trend lines though generally give correct predictions in rise or fall but are far off from the actual index value.

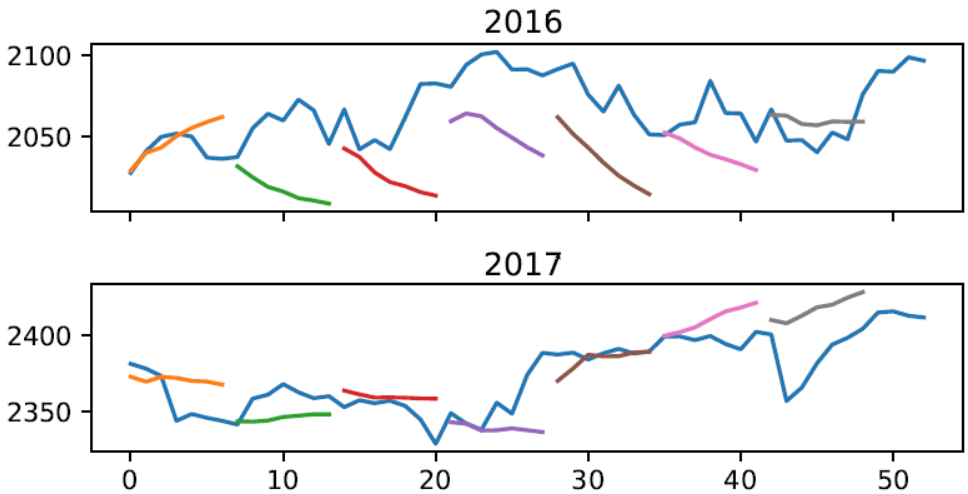

Figure 4: Single Pipeline CNN followed by uni-directional GRU predictions on S&P 500 Dataset. Horizontal axis shows the time segment for the index and prediction window. Vertical axis shows the closing price. Continuous plot is the actual closing price, predictions are shown as disjoint plots.

Figure 5 below, shows similar results for single pipeline model with Bi-directional GRU, which gives prediction lines correctly only for stable durations. Trend lines are generally showing in the downward direction and predicting seven days ahead in a non-stable scenario is challenging.

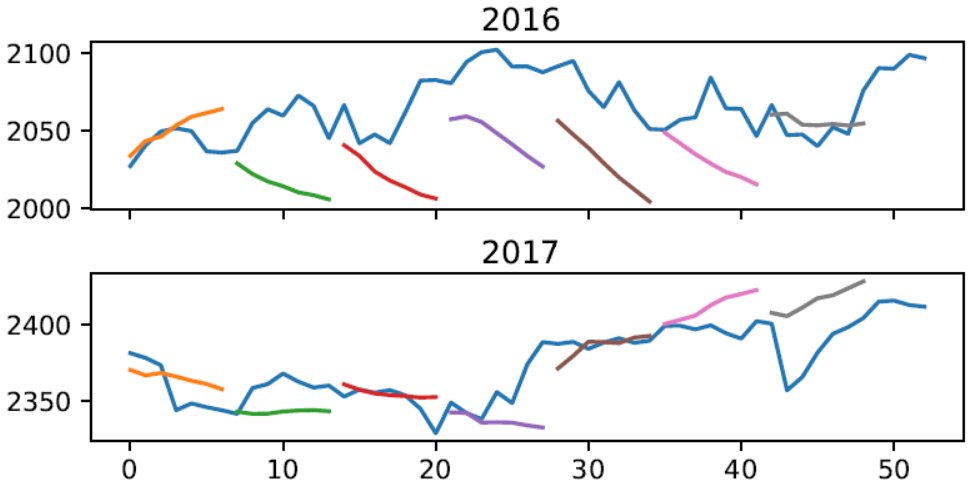

Figure 5: Single Pipeline CNN followed by Bi-directional GRU predictions on S&P 500 Dataset. Horizontal axis shows the time segment for the index and prediction window. Vertical axis shows the closing price. Continuous plot is the actual closing price, predictions are shown as disjoint plots.

Multiple pipeline models with uni-directional GRU and Bi-directional GRU units as shown in Figures 6 & 7, give better results in comparison to single pipeline models.

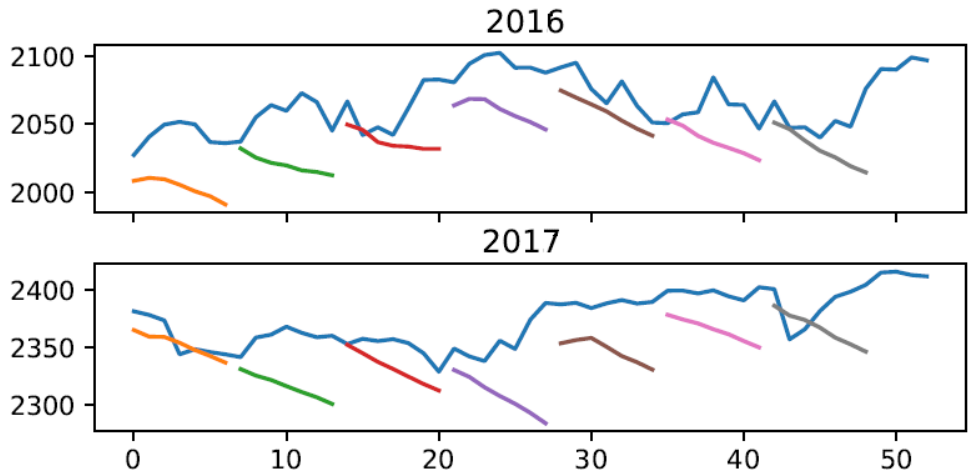

Figure 6: Multiple Pipeline CNN followed by GRU predictions on S&P 500 Dataset. X axis shows the time segment for the index and prediction window. Y axis shows the closing price. Continuous plot is the actual closing price, predictions are shown as disjoint plots.

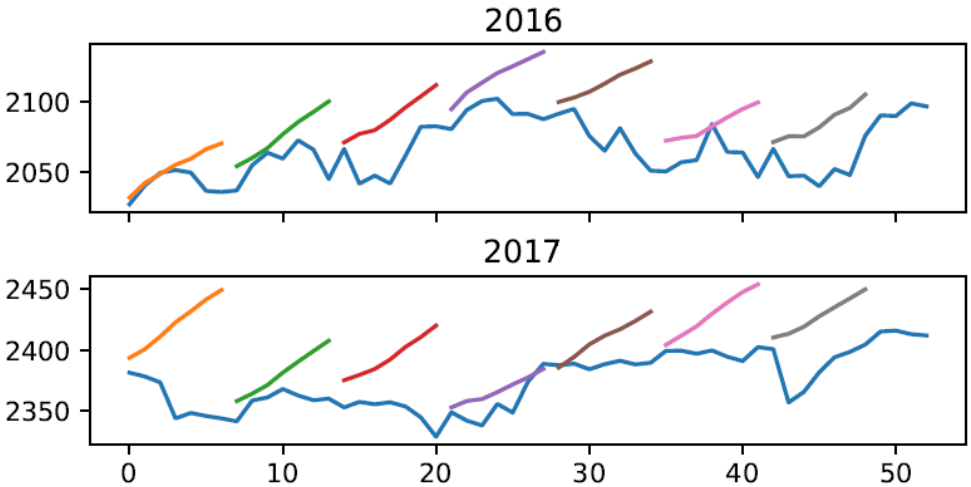

Figure 7: Multiple Pipeline CNN followed by Bidirectional GRU predictions on S&P 500 Dataset. X axis shows the time segment for the index and prediction window. Y axis shows the closing price. Continuous plot is the actual closing price, predictions are shown as disjoint plots.

7. CONCLUSION AND FUTURE WORK

In this paper, for the purpose of predicting stock index trend, we proposed new architectures of multiple pipeline deep learning models, consisting of CNN and uni-directional or bi-directional GRUs. We compared the performance of each proposed model with existing best performing deep learning models created using either single or multiple pipelines of CNN & Bi-directional LSTM units. We observed that multiple pipelines of deep layers that concatenate the best of the results from individual pipelines, perform better than single pipeline model in terms of mean squared error. Multiple pipeline Bi-directional GRUs showed the lowest mean squared error among all the models, due to its capability to learn from sequences of time-series data in both forward and backward direction. GRUs were much faster to train and predict, due to the lesser number of memory gates as compared to LSTM. Future work aims at integrating into the prediction process the impact of news and events through sentiment analysis.

REFERENCES

[1] Y.S. Abu-Mostafa and A. F. Atiya, "Introduction to financial forecasting", *Applied Intelligence,* vol. 6, no. 3, pp 205-213, 1996.

[2] J. Eapen, A. Verma and D. Bein, "Novel Deep Learning Model with CNN and Bi-directional LSTM for Improved Stock Market Index Prediction," 2019 *IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC),* doi. 10.1109/CCWC.2019.8666592.

[3] J. J. Murphy, "Technical analysis of the financial markets: A comprehensive guide to trading methods and applications", New York Institute of Finance, 1999.

[4] J. M. Marynowski, C. D. Voinescu, S. Puscasu, and T. M. O'Donnell, "Automated trading system in an electronic trading exchange", *U.S. Patents,* 7,177,833 B1, Feb 13 2007.

[5] D. Myr, "Machine learning automatic order transmission system for sending self-optimized trading signals", *U.S. Patents,* 7,739,182 B2, Jun 15 2010.

[6] N. Baba, N. Inoue, and Y. Yanjun, "Utilization of soft computing techniques for construting reliable decision support systems for dealing stocks," *Proceedings of the 2002 International Joint Conference on Neural Networks,* doi: 10.1109/IJCNN.2002.1007474

[7] P. Meesad and R. I. Rasel, "Predicting stock market price using support vector regression," *Proceedings of International Conference on Informatics, Electronics and Vision,* 2013, doi: 10.1109/ICIEV.2013.6572570

[8] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," *MIT Press,* [Online]. Available: https://www.deeplearningbook.org/ [Accessed 7th May 2019]

[9] F. E. H. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," Omega: *The international journal of management science,* vol. 29, no. 4, pp. 309-317, 2001.

[10] A. Cleeremans, D. Servan-Schreiber, and J. L. McClelland, "Finite State Automata and Simple Recurrent Networks," *Neural Computation,* vol. 1, no. 3, pp. 372-381. 1989. doi: 10.1162/neco.1989.1.3.372

[11] M. Herman and B. Schrauwen,"Training and Analyzing Deep Recurrent Neural Networks," *Advances in Neural Information Processing Systems 26,* 2013.

[12] Y. M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks," *IEEE Transactions on Signal Processing,* vol. 45, pp. 2673-2681. Nov., 1997.

[13] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty, Fuziness and Knowledge-Based Systems,* vol. 6, no. 2, pp. 107-116, 1998. doi: 10.1142/S0218488598000094

[14] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[15] J. Chung, C. Gulcehre, K. H. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *NIPS 2014 Deep Learning and Representation Learning Workshop,* 2014. arXiv:1412.3555

[16] Y. Zhang, Y. Fang, and X. Weidong, "Deep Keyphrase Generation with a Convolutional Sequence to Sequence Model," *4th International Conference on Systems and Informatics,* pp. 1477-1485. Nov. 2017. doi:10.1109/ICSAI.2017.8248519

[17] A. Yenter and A. Verma, "Deep CNN-LSTM with Combined Kernels from Multiple Branches for IMDb Review Sentiment Analysis," *8th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference,* Oct., 2017. doi: 10.1109/UEMCON.2017.8249013

[18] Y. Lavinia, H. H. Vo, and A. Verma, "Fusion based Deep CNN for Improved Large-Scale Image Action Recognition," *IEEE International Symposium on Multimedia,* Jan., 2017. doi: 10.1109/ISM.2016.0131

[19] C. Sgezedy,V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *The IEEE Conference on Computer Vision and Pattern Recognition,* pp. 2818-2826, 2016.

[20] M. M. Rounaghi and F. N. Zadeh, "Investigation of market efficiency and Financial Stability between S&P 500 and London Stock Exchange: Monthly and yearly Forecasting of Time Series Stock Returns using ARMA model," *Physica A: Statistical Mechanics and Its Applications,* vol. 456, pp. 10-21, 2016.

[21] Yahoo Finance, "S&P500 stock data" [Online]. Available: https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC. [Accessed 7th May 2019].

[22] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," *The IEEE International Conference on Computer Vision,* pp. 843-852, 2017.

[23] "Keras Documentation," [Online]. Available: https://keras.io/ [Accessed 7th May 2019].

[24] "SciKit-Learn," [Online]. Available: https://scikit-learn.org/stable/ [Accessed 7th May 2019].

[25] "matplotlib version 3.0.3," [Online]. Available: https://matplotlib.org/ [Accessed 7th May 2019].

[26] "pandas: Python data analysis library," [Online]. Available: https://pandas.pydata.org/ [Accessed 7th May 2019].

[27] "pickle - Python object serialization," [Online]. Available: https://docs.python.org/3/library/pickle.html [Accessed 7th May 2019].

[28] "NumPy," [Online]. Available: http://www.numpy.org/ [Accessed 7th May 2019].

[29] N. Marriott, "tmux-github," [Online]. Available: https://github.com/tmux/tmux/ [Accessed 7th May 2019].