Novel Deep Learning Model with CNN and Bi-Directional LSTM for Improved Stock Market Index Prediction

Jithin Eapen
Department of Computer Science
California State University
Fullerton, California 92831
jithin.john@csu.fullerton.edu

Abhishek Verma
Department of Computer Science
New Jersey City University
Jersey City, NJ 07305
averma@njcu.edu

Doina Bein
Department of Computer Science
California State University
Fullerton, California 92831
dbein@fullerton.edu

Abstract— **Predicting variations in stock price index has been an important application area of machine learning research. Due to the non-linear and complex nature of the stock market making predictions on stock price index is a challenging and non-trivial task. Deep learning approaches have become an important method in modeling complex relationships in temporal data. In this paper: (i) we propose a novel deep learning model that combines multiple pipelines of convolutional neural network and bi-directional long short term memory units. (ii) Proposed model improves prediction performance by 9% upon single pipeline deep learning model and by over a factor of six upon support vector machine regressor model on S&P 500 grand challenge dataset. (iii) We illustrate the improvement in prediction accuracy while minimizing the effects of overfitting by presenting several variations of multiple and single pipeline deep learning models based on different CNN kernel sizes and number of bi-directional LSTM units.**

Keywords— deep learning; Bi-directional LSTM; stock market prediction; CNN; S&P 500.

I. Introduction

Stock markets are some of the most import financial institutions of any capitalist economy. It allows enterprises to raise money from the investing public to fund their growth and operations. At the same time there is a large amount of risk involved for the investors due to its volatile nature. Hence both private investors and financial institutions like banks, have always attempted to devise ways to predict the changes occurring in the stock prices and make buy and sell decisions early.

The developments in the field of machine learning allowed the use of computers in the task predicting financial markets, by automating the existing statistical methods of data analysis [1]. Machine learning is about building computer systems or programs that can learn from data. Traditional machine learning approaches to stock prediction have focused on improving their performance with different techniques for feature extraction to select the most promising features from a dataset. The most popular is the support vector machine, a type of kernel based supervised machine learning algorithm. While being versatile, they are not as powerful as neural networks, especially the modern deep learning networks. Deep learning networks are powerful machine learning algorithms that make use of many cascading layers in order to learn multiple levels of representations. This corresponds to different levels of abstraction. The levels form a hierarchy of concepts. Thereby deep learning models can achieve high accuracy on prediction and classification tasks.

Once an accurate model is learned from training data, users will be able to feed temporal stock price data of fixed length to the model and be able to get the output as the prediction for the next seven days. It can be used to predict short term price changes, by various stakeholders like investors, financial analysts, investment bankers, as one of the factors affecting their investment decisions. Our primary objective is to analyze a sequence of time-series data and give predicted price for the next seven days. For this purpose, we selected the Standard and Poor's (S&P) 500 stock dataset, publicly available on Yahoo Finance [2].

The paper is organized as follows. In Section II we survey several representative techniques of machine learning and neural networks that are used for stock price prediction. In Section III we describe the dataset used for benchmarking learned models. In Section IV we present our research methodology and proposed novel model, which gives improved performance in comparison with the state of the art models. Section V presents experimental environment. Experimental results are presented in Section VI. Concluding remarks and future work is discussed in Section VII.

II. Related Work

A. *Automation of Stock Market*

Stock market traders today use electronic exchange systems to automate the trading of stocks based on theoretical buy and sell prices, which are calculated based on the current market price of the stock [3]. Such an automated system provides the advantages of better response times to market changes, more accurate trading operations, and reduced risk of loss due to repeated or mistaken operations.

Traders have developed multiple methods for predicting future stock prices: Fundamental stock analysis is the study of the business value of the stock, by considering factors like revenue, operating expenses, assets, products, management, competitors, economic environment etc. Technical analysis is based on the belief that markets tend to move in cycles and accurate predictions can be made by studying the price patterns and price trends of the past. Traders have been using statistical tools to analyze price and volume data. Recent advances in computing and machine learning has enabled them to analyze larger datasets.

978-1-7281-0554-3/19/$31.00©2019 IEEE

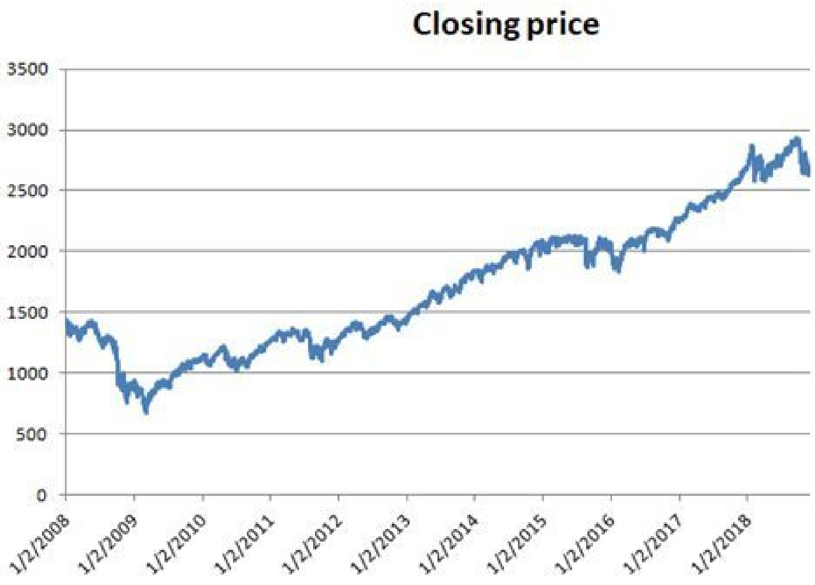

Figure 1. S&P 500 index closing prices from January 2, 2008 to November 27, 2018. [2]

B. Support Vector Machine Regressor

An important characteristic of stock prices is their time series dependency, i.e., stock price at a particular time is dependent on the price during the previous instance. In [4] the authors proposed to use Support Vector Machine (SVM) Regression based model to predict stock prices, as it is a suitable learning technique for recognizing patterns in a time series dataset. Different windowing methods were used to feed more reliable inputs to the regression model. The accuracy of the model depended on choosing the best parameters for the support vector machine. This pre-processing of the input data for rectangular window and flattened window were found to have good prediction for stock prices 1 and 5 days ahead respectively. This model was applied only to the time series dataset of a single company in the stock market.

C. Artificial Neural Network

Artificial neural network (ANN) models have been compared to other statistical models [5] and generally ANN models give better result [6]. The authors then compare various types and hybrids of ANN and multi-layer perceptron models and found that hybrid adaptive neural networks need to be remodeled when training data changes for the time-series forecast. In addition, each layer has to be controlled by reliability tests, as it is the input from the following layers of regression model.

D. Deep Neural Networks

Recurrent Neural Networks (RNN) are a special form of ANN where the weights computed during training are influenced not only by the current training example, but by previous decisions made on earlier training data. They have been specifically designed to deal with sequential or temporal data [7]. In RNNs, the hidden layer receives input not only from its immediate predecessor (input) layer, but also from a previous output layer. Thus, for each input term in the sequential data, the weights calculated also depend on the output prediction fromr the previous term. This architecture conforms perfectly with the time-series nature of stock price data. A hierarchy of Deep RNNs have been shown to perform well for prediction tasks as related to time-series data [8].

One drawback to using the RNNs is that they result in long connections between the final output layer, and hidden layers of previous data in the sequence, thus resulting in the problem of the vanishing gradients. To overcome this problem, mmodifications were made to the RNN units to include gates and memory units to remember the gradients. The most popular such units are Long Short Term Memory (LSTM) units [10]. LSTM units are designed to enhance the ability of RNNs to remember previous values and prioritize learning from newer data in a sequence upon older data. The memory of LSTM units also solves the problem of vanishing gradients, as the memory allows a gradient to move from one hidden layer to the next without being reduced, letting early layers of a neural network train almost as well as later layers [9]. LSTMs were shown to be more effective than conventional RNNs [10].

Bi-directional Recurrent Neural Networks [11] are used to learn from both the forward and backward time dependencies. In a Bi-directional LSTM, each unit is split into two units having the same input and connected to the same output. One unit is used for the forward time sequence and the other for the backward sequence. The common output does an arithmetic operation like sum, average or concatenation on both the units. Bi-directional LSTM is thus useful when learning from long spanning time-series data.

A technique that is suitable for feature extraction on high-dimensional data, as found in stock prices, is convolutional neural networks [12]. A convolutional neural network (CNN) is very similar to an ANN but makes specific assumptions about the input data. That allows it achieve higher invariance when encoding properties of input data into the network

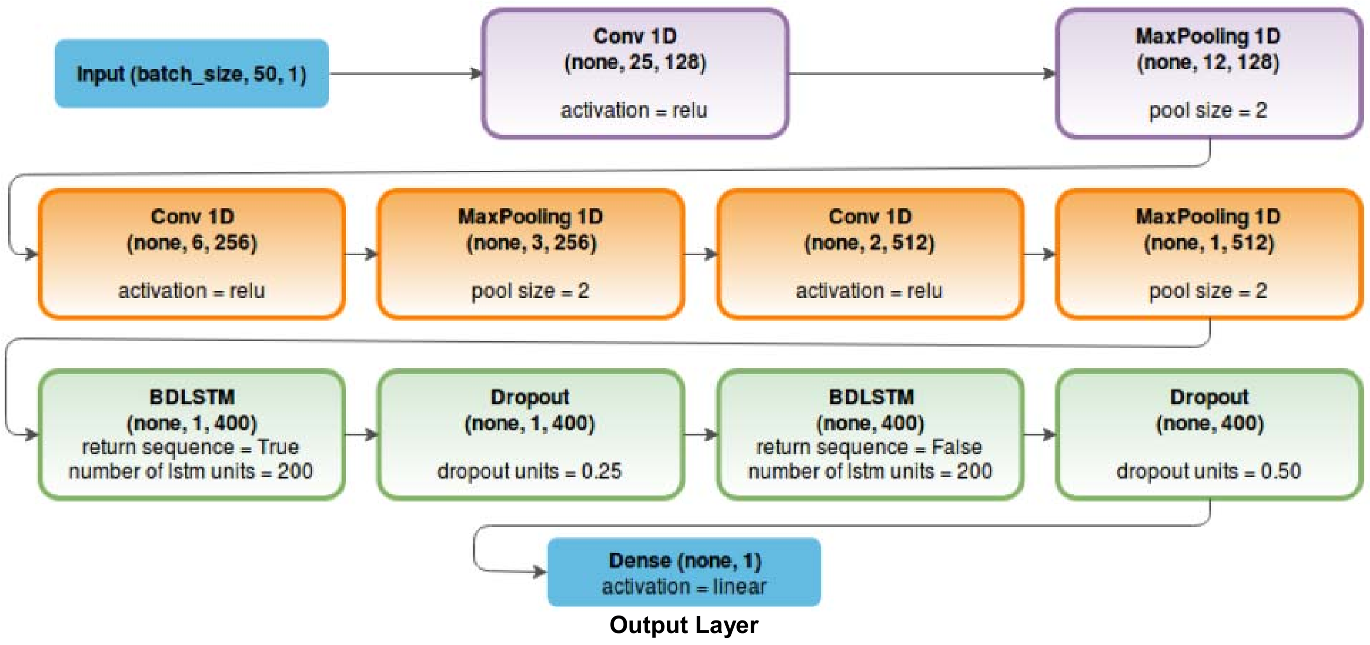
Input (batch_size, 50, 1)
Conv 1D (none, 25, 128) activation = relu
MaxPooling 1D (none, 12, 128) pool size = 2
Conv 1D (none, 6, 256) activation = relu
MaxPooling 1D (none, 3, 256) pool size = 2
Conv 1D (none, 2, 512) activation = relu
MaxPooling 1D (none, 1, 512) pool size = 2
BDLSTM (none, 1, 400) return sequence = True number of lstm units = 200
Dropout (none, 1, 400) dropout units = 0.25
BDLSTM (none, 400) return sequence = False number of lstm units = 200
Dropout (none, 400) dropout units = 0.50
Dense (none, 1) activation = linear
Output Layer

Figure 2. Diagram of basic structure of single pipeline CNN based Bidirectional LSTM network. We use this in our experiments as a baseline deep learning model to compare with proposed novel multiple pipeline deep learning model.

architecture [13]. A typical CNN includes convolution layers, rectified linear unit (ReLU) as activation function, pooling, and fully connected layers. CNNs are the main workhorse to solve modern computer vision and deep learning problems ever since AlexNet achieved first place in the ILSVRC 2012, an annual ImageNet Challenge [14]. CNNs need to be trained on large amounts of training data to generate deep learning models that achieve high generalization accuracy. Generalization accuracy refers to the performance of the models on test data. The hidden units, instead of merely calculating a function on the input, performs a convolution operation using a filter of weights. It is thus capable of exploiting the time structure of the input by performing convolutions on overlapping windows. They also use pooling operations, which combines nearby values in input or feature space through a max, average or histogram operator. The purpose of pooling is to achieve invariance to small local distortions and reduce the dimensionality of the feature space.

Most deep learning architectures are created using a single sequence of multiple layers, each performing a specific operation on the data. A multi-pipeline architecture, consisting of four parallel sequences of layers, has been shown to give lower classification errors on sentiment analysis in the text domain as compared to a single pipeline model [15]. Each pipeline was created using a sequence of CNN, Batch Normalization, ReLU activation, Pooling, and LSTM layers. The output of each pipeline is concatenated to give a single prediction.

III. Description of Dataset

The S&P 500 index (formerly Standard & Poor's 500 Index) is a market-capitalization-weighted index of the 500 largest U.S. publicly traded companies by market value. The index is widely regarded as the best single gauge of large-cap U.S. equities. Thereby it is an excellent measure of the performance of overall stock market.

S&P 500 dataset is publicly available on Yahoo Finance [2]. It consists of historical daily opening and closing stock prices along with a few other additional market parameters. Fig. 1 shows the daily closing price of S&P 500 index from January 2, 2008 to November 27, 2018. It has the daily data for five days in a week for a period of 10 years. It consists of columns like daily Opening price, Closing price, High and Low prices, and daily Volume of stocks traded. The S&P500 was chosen due to its stability compared to individual company stock prices. Statistical analysis shows that the autoregressive moving average model for S&P 500 outperforms the London Stock Exchange, suggesting better potential for predictive models [16].

Stock index data is segmented into sequences of 50 closing prices, with the last price of each sequence being one day after the sequence before it. We construct multiple sequences of length 50, and feed the deep learning model the entire batch as an input. Each sequence is incremented by one day. Additionally, tests were performed using sequences of length 14, 28, and 56, but the length of 50 performed the best out of the three other lengths. Each of the data segments was normalized by subtracting each price in the sequence by the first price in the sequence, then divide by the first price in the sequence. This form of normalization is called relative change. Normalization means that the deep learning network will look at the degree to which the prices changed over time, rather than the raw prices themselves, allowing it to generalize to a wider range of input values.

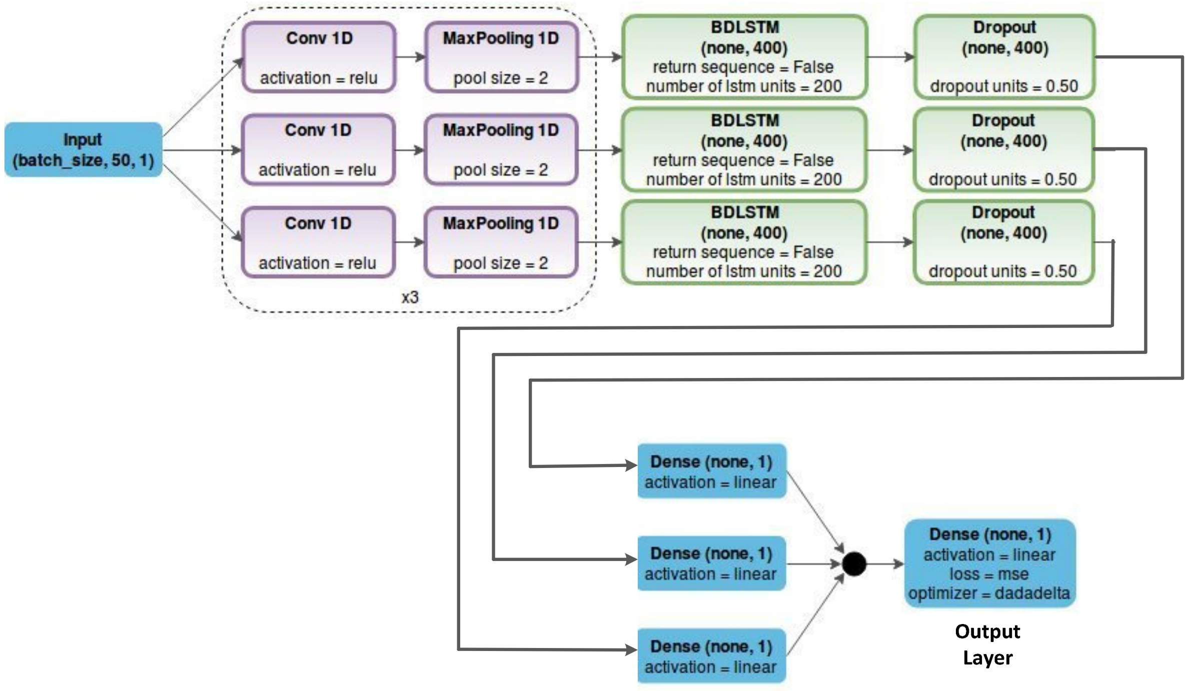
Input (batch_size, 50, 1)
Conv 1D activation = relu
MaxPooling 1D pool size = 2
x3
BDLSTM (none, 400) return sequence = False number of lstm units = 200
Dropout (none, 400) dropout units = 0.50
Dense (none, 1) activation = linear
Dense (none, 1) activation = linear loss = mse optimizer = dadadelta
Output Layer

Figure 3. Diagram of the best performing proposed multiple pipeline model with CNNs followed by Bidirectional LSTMs.

IV. Proposed Multiple Pipeline CNN and Bi-Directional LSTM Model

We constructed various deep learning architectures and compared their performance to find the best performing model. We also implement a Support Vector Machine Regressor model for comparison. A SVM Regressor model was created using Scikit-learn [20] library in python. We perform grid search to find the best parameters for C and gamma for a model using a grid of both radial basis function kernel and linear kernel.

Then, we test several single pipeline deep-learning models using using CNN and Bi-directional LSTM. We find the best single pipeline model, configured as group of 1-dimensional Convolutional layers, followed by two Bi-directional LSTM layers with dropout. The CNN layers use ReLU as the activation function and is followed by a Max-pooling layer with pool size of 2. Fig. 2 diagram shows basic structure of single pipeline CNN based Bi-directional LSTM network. We use this in our experiments as a baseline deep learning model and compare its results with support vector regressor and proposed novel multiple pipeline deep learning model.

In our proposed model, we construct a deep learning architecture consisting of three pipelines of separate layers, see Fig. 3. Each pipeline consists of initial group of three 1-dimensional CNNs with a ReLU activation followed by Max-pooling. The CNNs are used so as to extract the higher level features before passing data to the Bi-directional LSTM neural layers. The group of CNNs are followed by a group of two Bi-directional LSTM layers with dropouts. Each Bi-directional LSTM unit learns from both the forward and backward sequence of data, and uses concatenation to merge the two sequence outputs. Dropout layer is used to prevent overfitting in the model. The use of LSTM also prevents the problem of vanishing gradients.

Each pipeline takes the entire sequence of 50 prices as input, and the CNN layers use 128 kernel windows of size 5 or 9, to return another sequence of smaller size.

The output from the CNN group is fed into the Bi-directional LSTM layer group which:

1) learns using 200 pairs of forward and backward LSTM units.

2) gives an output sequence of length 400

3) drops 50% (dropout) of the weights to generalize the model.

The two dimensional output of this group is then fed to a dense layer with one unit and a linear activation, which then gives the output for each individual pipeline.

The output of each of the three pipeline is concatenated and given to the final layer consisting of a dense layer. The final output is a numeric value of predicted price for seven days into the future. As a typical regression problem, we have used mean squared error (MSE) as the loss function to measure the error in the predicted versus actual data. Adadelta is the gradient descent optimization algorithm for correcting the weights in the model.

TABLE I

RESULTS FROM FOUR BEST PERFORMING SUPPORT VECTOR MACHINE REGRESSOR MODELS ON S&P 500 DATASET

Kernel	C	Gamma	Mean Test Score	Mean Train Score
rbf	0.059636233	0.212095088	0.001619057	0.001367225
rbf	0.040949150	0.372759372	0.001652736	0.001349042
rbf	0.126485521	0.120679264	0.001666914	0.001444660
rbf	0.086851137	0.212095088	0.001672472	0.001433185

Note: rbf is radial basis function. Mean Test Score and Mean Train Score are computed from Mean Squared Error. Lower values are better.

TABLE III

RESULTS FROM FOUR BEST PERFORMING PROPOSED MULTIPLE PIPELINES DEEP LEARNING MODELS WITH CNN AND BI-DIRECTIONAL LSTM ON S&P 500 DATASET

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

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

V. EXPERIMENTAL ENVIRONMENT

We have developed the model using the python programming language. We used the following libraries for various functionalities:

- Keras [17] is a popular open-source library that enables researchers and software engineers to rapidly define and train many deep learning models in a short amount of time. It is used for creating, and training neural networks. It provides a simple interface to existing libraries like Tensorflow [18], which allows us to use GPUs for faster training and prediction. With Keras, we can achieve fast experimentation which is the key component to gain insightful feedback and garner improvements in accuracy of deep learning models for stock prediction. Note that in Keras version 2.0, the Keras team has removed metrics such as recall, precision, and fmeasure to implicitly promote the use of accuracy as the main metric for CNNs that are trained on balanced datasets [19].

- Tensorflow [18] is the backend for Keras, and facilitates the processing of low-level operations such as tensor products and convolutions. It is developed and maintained by Google.

- Scikit-learn [20] was used for performing the grid search of the model to find the best parameters using 5-fold Cross-validation.

- Matplotlib [21] was used for plotting the graphs for the actual time-series as well as the predicted trends.

- Pandas [22] was used for reading value from csv files as DataFrames.

- Numpy [23] used to perform matrix operations like flip, reshape, and create random matrices.

All training and prediction have been performed on several servers with multiple NVIDIA GeForce GTX 1080 Ti GPUs for parallel processing, each GPU has 12GB of VRAM. The servers use Ubuntu 16.10 as the operating system. Main processor used is a collection of 4 Intel Xeon E5-2630 CPUs @ 2.20 Ghz, with 10 cores and 256 GB RAM. Each server uses CUDA toolkit v9.0.176 for utilizing the Nvidia GPUs. We used two such servers with a total of 11 GPUs, allowing to train multiple models simultaneously and consequently drawing on the benefits from parallel architecture the proposed multiple pipeline model.

VI. RESEARCH RESULTS AND ANALYSIS

We have used the grid search to perform five-fold cross

TABLE II

RESULTS FROM FOUR BEST PERFORMING SINGLE PIPELINE DEEP LEARNING MODELS WITH CNN AND BI-DIRECTIONAL LSTM ON S&P 500 DATASET

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

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

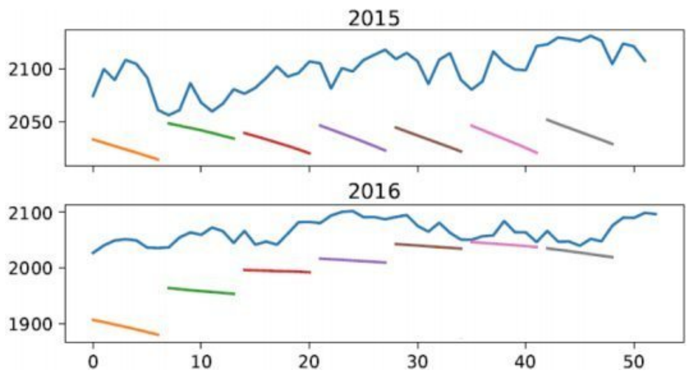

Figure 4: Support Vector Regressor predictions on S&P 500 Dataset. X axis shows the time segment for the index and prediction window. Y axis shows the closing price. Continuous plot is the actual closing price, predictions are shown as disjoint plots for next seven days. Data comes from years 2015 and 2016.

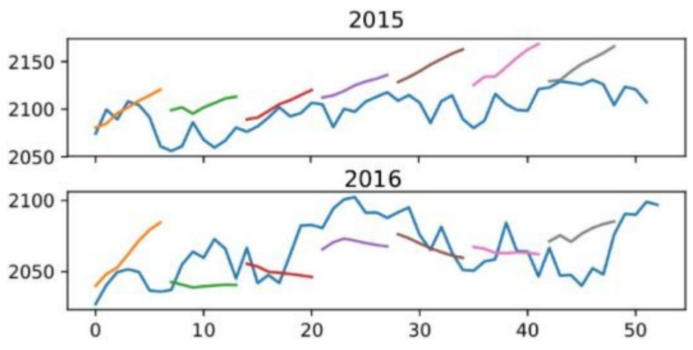

Figure 5: Single Pipeline Deep Learning Model predictions with CNN and Bi-Directional LSTM on S&P 500 Dataset. X axis shows the time segment for the index and prediction window. Y axis shows the closing price. Continuous plot is the actual closing price, predictions are shown as disjoint plots for next seven days window. Data comes from years 2015 and 2016.

validation for each of support vector regressor, single pipeline and multiple pipeline models to find the top four performing models and their parameter values in each of the three aforementioned categories.

A. Support Vector Machine Regressor Models Results

Results from the four best performing Support Vector Regressor models are shown in Table I. Five-fold cross validation was performed along with grid search to select best kernel, C and gamma values. Mean squared error is used as the performance metric. Lower value for mean squared error is considered better. From the list of rbf and linear kernels, the grid search returned rbf kernels as the best performing models. This was an expected result, as radial basis function kernels generally perform better than linear kernels on non-linear data [24]. The mean squared error observed for SVM regressor is in the order of 10^{-3}. From the results it is observed that there is low variance across best four models.

B. Single Pipeline Models Results

Single pipeline deep learning model results are shown in Table II. Various parameter values for CNN's kernel sizes and number of Bi-directional LSTM units were searched to obtain optimal parameters. Kernel sizes of 5 or 9 with Bi-directional LSTM units as 200 or 400 resulted in four best performing models. All mean squared error results given by the deep learning model were in the order of 10^{-4}.

C. Proposed Multiple Pipeline Models Results

Proposed multiple pipeline deep learning model results are shown in Table III. CNN's kernel sizes of 5 or 9 with Bi-directional LSTM units as 200 or 400 resulted in four best performing models. 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}.

D. Comparison of Results Across Models

Multiple and single pipeline deep learning models outperform Support vector machine regressor models by a factor higher than 4.7 for MSE value. Best performing multiple pipeline model outperforms single pipeline model by around 9%. This validates our idea that combining the three pipelines results in improved performance, thereby the fusion is non-redundant. Since, all of the individual pipelines in the multiple pipelines model can be evaluated in parallel; there is negligible difference in training time in comparison to single pipeline model.

Best performing model for each of multiple pipelines, single pipeline, and SVM regressor methodologies is used for constructing prediction graphs in Figs. 4, 5, 6. 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 for next seven days window. Predictions are based on data from past 50 days. Data comes from years 2015 and 2016.

Predictions from the multiple and single pipelines model Figs. 5 and 6 follow the actual prices more closely than the support vector machine regressor model in Fig. 4. The predicted trends in Figs. 5 and 6 give much better stock market direction (up or down) when compared to Fig. 4. In general deep learning models significantly outperform the support vector machine regressor model, both in terms of MSE and prediction trend given in Figs. 4, 5, 6.

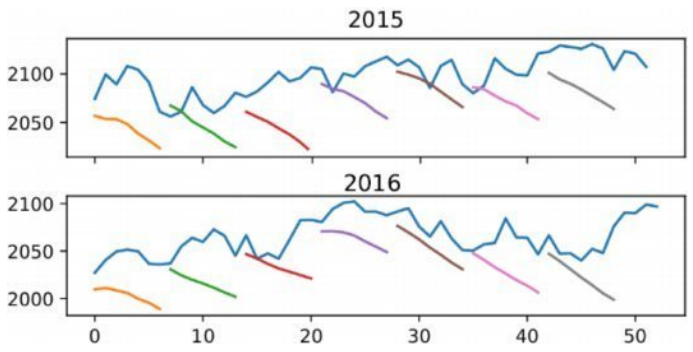

Figure 6: Multiple Pipeline Deep Learning Model predictions with CNN and Bi-Directional LSTM on S&P 500 Dataset. X axis shows the time segment for the index and prediction window. Y axis shows the closing price. Continuous plot is the actual closing price, predictions are shown as disjoint plots for next seven days window. Data comes from years 2015 and 2016.

VII. Conclusion and Future Work

In this paper we proposed a new deep learning model that combines multiple pipelines of CNN for feature extraction from data and Bi-directional LSTM in analyzing temporal data. We observed that CNN layers when combined with Bi-directional LSTM show improved prediction performance from the temporal sequence as compared to the traditional SVM regressor. Furthermore, multiple pipelines of deep layers that concatenate the best of the results from three individual pipelines performs better than single pipeline model. This demonstrates the applicability of deep learning models on predicting both the values and trends in a time-series data. We also presented multiple variations of our model to show how we have increased accuracy and minimized the effects of overfitting.

Future research would apply deep learning toward combining model for predicting price and trends with sentiment analysis on news data to get even better stock price index prediction.

REFERENCES

[1] J. Hall, G. Mani, and D. Barr, "Applying Computational Intelligence to the Investment Process", in *Proc. of 1996 CIFER: Computational Intelligence in Financial Engineering*.

[2] Yahoo Finance, *S&P500 stock data* [Online]. Available: https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC. [Accessed 29 Nov. 2018].

[3] J. M. Marynowski et al., "Automated Trading System in an Electronic Trading Exchange", U.S. Patent 7 177 833B1, Feb 13, 2007.

[4] P. Meesad and R.I Rasel, "Predicting Stock Market Price Using Support Vector Regression," in *Proc. of Int. Conf. Informatics, Electron. and Vision, (ICIEV)*, 2013. doi: 10.1109/ICIEV.2013.6572570.

[5] G. Zhang, B.E. Patuwo, and M.Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *Int. J. Forecasting*, vol. 14, pp. 35-62, Mar. 1998.

[6] E. Guresen, G. Kayakutlu, and T.U Daim, "Using artificial neural network models in stock market index prediction," *Expert Systems with Applications*, vol. 38, pp. 10389-10397, Aug. 2011.

[7] N. V. Bhattacharjee and E. W. Tollner, "Improving management of windrow composting systems by modeling runoff water quality dynamics using recurrent neural network," *Ecological Modelling*, vol. 339, pp. 68-76, Nov. 2016.

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

[9] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735-1780, 1997.

[10] W. Bao, J. Yue and Y. Rao, "A deep learning framework for financial time," *PLoS ONE*, vol. 12, doi: 10.1371/journal.pone.0180944, Jul. 2017.

[11] Y.M. Schuster and K.K. Paliwal, "Bidirectional Recurrent Neural Networks," *IEEE Trans. Signal Process.*, vol. 45, pp. 2673-2681, Nov. 1997.

[12] CM. Langkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Lett.*, vol. 42, pp. 11-24, Jun. 2014.

[13] W. Zhang, "Shift-invariant pattern recognition neural network and its optical architecture," *Proc. annual conf. Japan Society of Applied Physics*,1988.

[14] A.O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vision*, vol. 115, pp. 211–252, Dec. 2015.

[15] A. Yenter and A. Verma, "Deep CNN-LSTM with Combined Kernels from Multiple Branches for IMDb Review Sentiment Analysis," *8th IEEE Annu. Ubiquitous Computing, Electron. & Mobile Commun. Conf.*, *(UEMCON)*, doi: 10.1109/UEMCON.2017.8249013, Oct. 2017.

[16] 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 Applicat.*, vol. 456, pp. 10-21, Aug. 2016..

[17] *Keras: The Python Deep Learning library* [Online]. Available: https://keras.io. [Accessed 29 Nov. 2018].

[18] *TensorFlow: An open source machine learning framework for everyone* [Online]. Available: https://www.tensorflow.org/. [Accessed 29 Nov. 2018].

[19] Keras Team, *Keras 2.0 release notes* [Online]. Available: https://github.com/keras-team/keras/wiki/Keras-2.0-release-notes. [Accessed 29 Nov. 2018].

[20] *Scikit-Learn: Machine Learning in Python* [Online]. Available: https://scikit-learn.org/stable/. [Accessed 29 Nov. 2018].

[21] The Matplotlib development team (2018, Nov. 28). *matplotlib Version 3.0.2* [Online]. Available: https://matplotlib.org/. [Accessed 29 Nov. 2018].

[22] T. Mester (2018, Jul. 10). *Pandas Tutorial 1: Pandas Basics* [Online]. Available: https://data36.com/pandas-tutorial-1-basics-reading-data-files-dataframes-data-selection/. [Accessed 29 Nov. 2018].

[23] *NumPy* [Online], (2018). Available: http://www.numpy.org/. [Accessed 29 Nov. 2018].

[24] C.W. Hsu, CC. Chang, and C.J. Lin, "A practical Guide to Support Vector Classification," Dept. Comp. Sci., Nat. Taiwan Univ., Taipei, Apr. 2010