Novel Deep Learning Model with Fusion of Multiple Pipelines for Stock Market Prediction

Andrew Quintanilla

Department of Computer Science

California State University

Fullerton, California 92834

Email: quint.andrew@gmail.com

Abhishek Verma

Department of Computer Science

New Jersey City University

Jersey City, NJ 07305

Email: averma@njcu.edu

Abstract: ***Deep learning has become a powerful tool in modeling complex relationships in data. Convolutional neural networks constitute the backbone of modern machine intelligence applications, while long short-term memory layers (LSTM) have been widely applied towards problems involving sequential data, such as text classification and temporal data. By combining the power of multiple pipelines of CNN in extracting features from data and LSTM in analyzing sequential data, we have produced a novel model with improved performance in stock market prediction by 20% upon single pipeline model and by five times upon support vector regressor model.***

We also present multiple variations of our model to show how we have increased accuracy while minimizing the effects of overfitting. Specifically, we show how changes in the parameters of our model affect its scores for training and testing, and compare the performance of a multiple pipelines model using three different kernel sizes versus a single pipeline model.

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

Reference to this paper should be made as follows:. A. Quintanilla and A, Verma (xxxx) 'Novel Deep Learning Model with Fusion of Multiple Pipelines for Stock Market Prediction', *Int. J.of Advanced Intelligence Paradigms*

Biographical notes:

Andrew Quintanilla received his MS degree in Computer Science from

A. Quitanilla and A. Verma

California State University, Fullerton. His research interests are in deep learning, model ensembling, and financial market analysis.

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, and machine learning. Deep learning on big datasets such as deep convolutional nets, model ensembling, image/video/speech recognition, natural language processing, financial market analysis, sentiment analysis. Computer vision on big datasets in video and image. Fusing multiple modalities from video/images/text/speech. Data mining, artificial intelligence, and biometrics for surveillance and security.

1 INTRODUCTION

Stock market prediction has been the subject of intense research. On one end, the Random Walk Hypothesis states that prices evolve according to random price changes, and the Efficient-Market Hypothesis states that prices reflect all currently available information, which would mean that prediction of stock prices is impossible [1]. There are, of course, numerous investment managers who claim that they will outperform the market, and are paid handsomely by their clients due to that claim. They may not be knowingly selling false products, but the use of inappropriate statistical tools and lenient standards for evaluating trading strategies can lead to inflated estimates of performance [2].

More modern approaches to stock prediction include the use of deep learning algorithms such as neural networks. One problem with the application of neural networks towards financial problems is about the types of tasks they are suited for. Typically, neural nets have been applied towards tasks to which humans are well-suited, such as recognizing the content of an image, or converting speech into text. However, in the case of stock prediction, humans have a dubious ability to predict stock performance. The hope is that deep learning approaches will be able to model complex nonlinear relationships between variables that are too difficult for humans – and traditional statistical models – to understand [3].

The rest of this paper is organized into six main parts. Section 2 reviews related literature in the field of machine learning, neural networks, and stock prediction. Section 3 defines the details of the S&P500 stock index dataset. Section 4 presents the detailed structure of the model layer by layer. Section 5 describes the experimental environment. Section 6 describes the training and configuration of the model. Section 7 discusses the results and challenges. Section 8 concludes the paper.

Novel Deep Learning Model with Fusion of Multiple Pipelines for Stock Market Predict.

2. BACKGROUND WORK

The following subsections review the literature behind neural networks, time-series forecasting, and stock market prediction.

2.1 Neural Networks

2.1.1 Convolutional Neural Networks

Convolutional Neural Networks, or CNNs, operate by extracting features from data [4]. Unlike fully-connected layers, each convolutional layer is connected only to a small region of neurons from the previous layer. This region is determined by the kernel size and stride parameters. They are most commonly used to analyze spatial data, especially images and video [5]. Although, the functionality that CNNs perform could technically be performed using traditional fully-connected layers, it quickly becomes prohibitively expensive to do so, due to traditional dense layers needing to relearn a feature at every input position. One paper found that, after using five fully connected layers with a total of 12,174,500 weights, the model's performance in predicting the direction of financial market movements was only slightly better than random selection [6]. The use of CNNs allows for a much more efficient use of computing resources, due to the limited number of connections and weights needed compared to fully connected layers.

2.1.2 Recurrent Neural Network

Recurrent Neural Networks, or RNNs, differ from traditional neural networks in that, during training, they are influenced not only by the current training example, but by previous decisions made on earlier training samples. This memory makes it well suited for capturing non-linear relationships in data that is sequential in nature, such as text or time series data [7]. However, recurrent neural networks face the problem of vanishing gradients, where training becomes exponentially difficult with the addition of hidden layers [8]. This happens during training, as a gradient is back-propagated through the model. As it travels through the layers, the gradient is reduced, such that the earlier layers see a much smaller gradient than later layers, and train much slower.

2.1.3 Long-Short Term Memory

Long short-term memory (LSTM) units are used to build recurrent neural networks (RNNs). LSTMs enhance the ability of RNNs by being able to remember previous values and prioritize learning from newer data in a sequence over older data. This allows it to solve problems involving long time lags that were difficult to solve with older approaches, such as language processing. 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]. Evidence has shown LSTMs to be more effective than conventional RNNs [10].

2.2 Stock Market Prediction

2.2.1 Automated Trading

Modern stock trading has moved away from human stock brokers towards automated computer systems that trade stocks using a predefined set of rules based on a trading strategy. This allows for a much faster trading speeds, and a more precise development of trading strategy using testing on historical stock data [11]. Modern high-frequency

trading measure time intervals between trades by the millisecond, and competing algorithms fight for fractions of a cent in profit.

2.2.2 Traditional Machine Learning Algorithms

Traditional machine learning approaches to stock prediction include the use of non-neural network supervised learning algorithms, such as support vector machines. Due to these tools being versatile, but not quite as powerful as neural networks, many traditional machine learning approaches focus on improving their performance with different techniques for feature extraction to select the most promising features from a dataset.

One example of such an approach is the use of independent component analysis to extract important features from a dataset, which is used to train support vector machines [12]. Independent component analysis was shown to improve results over a pure support vector machine approach [12]. Unlike the dataset used for the model described in the rest of this paper, the dataset examined a number of technical indicators in combination with stock closing prices [12].

Another approach using support vector machines attempts to improve performance using Particle Swarm Optimization, a technique that uses a population of software agents to seek out optimizations in the solution space [13]. Although our approach uses neural networks, there are similarities in that the particles in particle swarm optimization have a memory that allows them to revert to previous and more advantageous positions [13], which is reminiscent of how LSTM networks can remember previous values to make it more resilient to the vanishing gradient problem.

Chen et al. proposed the McDSL algorithm for discovering causalities on high-dimensional discrete data, which would help reduce the problem space of high-dimensionality data, such as stock market data [14]. McDSL follows a two-phase structure, which aims to first find relationships between variables, and then establish the direction of cause-effect between those variables.

Traditional trading strategies often involve creating and evaluating stock portfolios. By applying a Mean-Variance framework to stock portfolios, researchers were able to evaluate the efficiency of anomaly-based trading strategies [15]. In order to simplify the problem space, we chose to look at only a single index.

A benefit of traditional machine learning algorithms over neural network approaches is that algorithms such as support vector machines can be more interpretable than those of neural nets [16]. For short term forecasting (defined as one or two weeks), support vector regression has been shown to perform as well or better than a basic neural network approach using multilayer perceptrons [17]. However, with the advent of more advanced types of neural networks, such as convolutional neural networks, deep learning approaches to tough regression problems such as stock prediction are showing more promise in their use.

2.2.3 Deep Learning Algorithms

Some approaches to stock prediction with neural networks include the use of LSTM to overcome limitations with traditional RNNs when it comes to vanishing gradients [10] [18].

Another approach uses deep belief networks, combined with an oscillation box trading strategy, to perform automated trading. Unlike our approach, which looks at the

overall index price, the authors of this paper chose to look at 400 individual stocks in the S&P500 [19].

2.2.4 Technical Analysis

Technical analysis involves the use of technical indicators and the study of charts to make predictions. One paper compared different methods of stock prediction and found that their neural net approach attained better accuracy than an approach utilizing analysis of technical indicators [20]. Technical indicators are calculated using various sequence lengths of previous stock prices. By applying CNNs to longer sequences of stock prices, we hope to have the neural net learn relevant features from the prices themselves, rather than relying on technical indicators developed by humans to generate predictions.

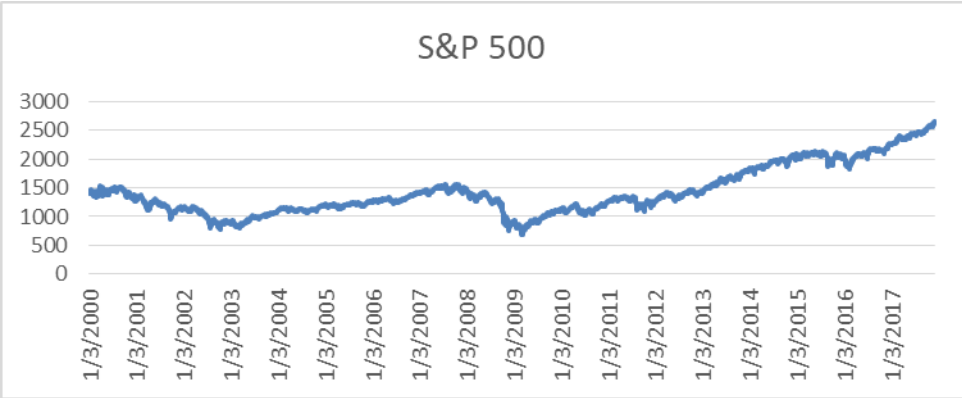

Figure 1: Historical S&P500. Y axis measures the closing price.

3. DATASET DESCRIPTION

The dataset used for this research consists of historical stock prices for the S&P500. Specifically, it uses the daily closing price of the index from January 3, 2000 to December 1, 2017, as shown in Figure 1. The S&P500 was chosen due to it being a good representation of the US economy, and due to its stability compared to individual company stock prices. Statistical analysis shows that the autoregressive moving average model for S&P500 outperforms the London Stock Exchange, suggesting better potential for predictive models [23]. An analysis of the variance of the S&P500 also rejects the Random Walk Hypothesis, which makes it a more promising subject for the training of the neural network [1].

4 RESEARCH METHODOLOGY

4.1 Data Segmentation

The stock market data was converted from daily closing prices into sequences of 50 closing prices, with the last price of each sequence being one day after the sequence before it. Tests were performed using sequences of length 20, 50, and 100, with 50 performing the best out of the three lengths.

4.2 Data Normalization

Each of the segments was normalized by subtracting each price in the sequence by the first price in the sequence, then dividing 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.

4.3 Single Pipeline Deep Learning Model

The single-pipeline deep learning model as shown in fig. 2 consists of three CNN layers, feeding into two LSTM layers, feeding a dense output layer. The first CNN layer

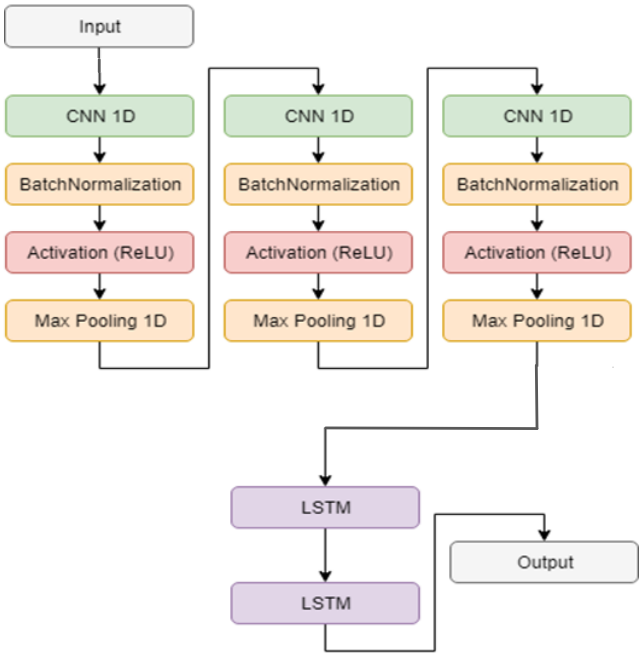

Figure 2: Single-Pipeline deep learning model

consists of 128 filters. The second CNN layer consists of 256 filters. The third CNN layer consists of 512 filters. All three layers use a kernel size of 5, as well as a stride of 2. Batch normalization occurs after each layer, which is followed by a rectified linear unit. Max pooling follows the ReLU, which uses a pool size of 2.

The two LSTM layers have identical numbers of units. The model was tested using 200 and 400 LSTM units. During training, the first LSTM layer uses a dropout value of 0.25, while the second layer uses a dropout value of 0.5, to reduce the amount of overfit in the model. This then feeds into a linear activation unit for output. We compare later in this paper results of single-pipeline model with the proposed multiple pipelines deep learning model.

4.4 Proposed Multiple Pipelines Deep Learning Model

The multiple pipelines model is shown in fig. 3. It combines the three most promising single pipeline models to use the best predictive aspects of all three. CNN layer is followed by batch normalization, activation layer that uses rectified linear units, and max pooling layer. Due to the increase in training time needed for the multiple pipelines model, the multiple pipelines model uses a single 400 unit LSTM layer instead of two 300 unit LSTM layers. Also, to save training time, rather than vary the parameters of the CNN layers of each pipeline independently, we tested one parameter that represents how much the kernel size increases between pipelines. For example, with a step size of two and a starting kernel size of five, a three-pipelines model would have kernel sizes of 5, 7, and 9 for each of its CNN layers within a pipeline.

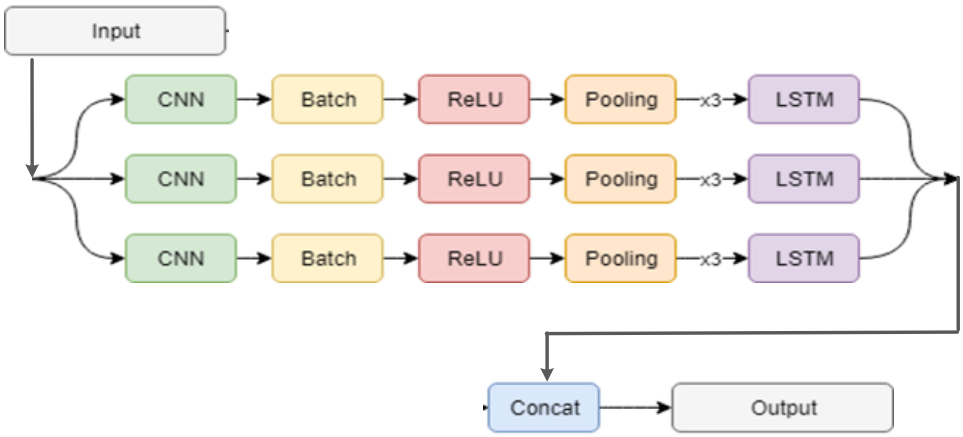

Figure 3: Proposed Multiple Pipelines deep learning model.

Combination of the pipeline predictions is done using concatenation of the outputs of each pipeline, which was found to improve the model's performance vs. averaging the outputs of the pipelines.

5. EXPERIMENTAL ENVIRONMENT

The machine used for training was a Windows 10 PC, with 8 GB of RAM, an Intel 4770K CPU, and an Nvidia 780 GTX GPU. The training and testing procedures were coded in Python 3, with the models being constructed using Keras [26] and TensorFlow [25].

The Nvidia 780 GTX is a gaming GPU, released in 2013, with 3 GB of video memory running at a frequency of 6.0 Gbps, and 2304 cores running between 863 and 900 MHz, depending on the card's temperature [24]. It is normally used for hardware acceleration of 3D graphics for video games. However, GPUs have enormous computational power, particularly with highly parallel calculations, and with Nvidia's

Table 1: Results from Support Vector Regressor on S&P 500 Dataset

Kernel	C	Gamma	Mean Test Score	Mean Train Score
rbf	0.126485522	0.120679264	0.001659199	0.001449364
rbf	0.059636233	0.212095089	0.001668910	0.001416561
rbf	0.086851137	0.212095089	0.001696734	0.001458230
rbf	0.040949151	0.372759372	0.001717464	0.001417301

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

Table 2: Results from Single Pipeline Deep Learning Model on S&P 500 Dataset

Kernel Sizes	LSTM Units	Mean Test Score	Mean Train Score
5, 9, 13	400	0.000290009	0.000278847
5, 7, 9	400	0.000292874	0.000281922
5, 7, 9	200	0.000348505	0.000341443

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

release of CUDA, that power became available for uses outside graphics programming, such as cryptocurrency mining and the training of deep learning models. GPUs are well-suited to training neural networks due to the parallel nature of calculating neuron weights during back propagation.

TensorFlow [25] is an open-source machine learning library developed by Google Brain, a section of Google dedicated to AI research. It uses dataflow graphs to represent computations in terms of the dependencies between individual operations [25]. It has uses in a variety of machine learning and artificial intelligence applications. Using their Python API directly gives us more control over how it uses hardware, and the ability to use multiple GPUs during training.

Keras is a high-level API specifically for neural networks written in Python [26]. We chose TensorFlow to use as its backend. It operates at a higher level of abstraction than TensorFlow, and allows us to stack deep learning network layers with simple API calls. This in turn made it easier to change the model parameters so that we could experiment with many different combinations of values.

Other software libraries used include Numpy, a Python library that adds a number of scientific computing functions, and SciKit-Learn [21], which added a number of functions used during training, such as grid search and cross validation. It was also used for the development of the support vector regressor model used as a comparison for the deep learning network models. Pandas was used to handle loading data from the CSV.

Table 3: Results from Proposed Multiple Pipelines Deep Learning Model on S&P 500 Dataset

Kernel Size	LSTM Units	Mean Test Score	Mean Train Score
9	400	0.000240905	0.000232545
5	200	0.000273995	0.000262546
5	400	0.000340446	0.000335832

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

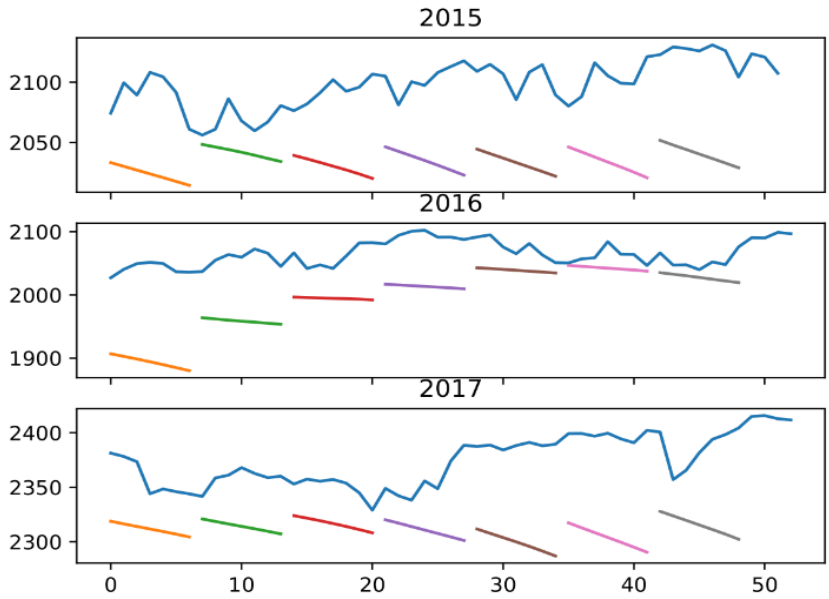

Figure 4: Support Vector Regressor predictions on S&P 500 Dataset. X axis shows the time segment for the index and prediction window. Y axis shows the closing price. Continuous plot is the actual closing price, predictions are shown as disjoint plots.

6. TRAINING OF SINGLE PIPELINE AND PROPOSED MULTIPLE PIPELINES DEEP LEARNING MODELS

Grid search using five-fold cross-validation was used to tune the parameters for the single pipeline and multiple pipelines models. For each training run of single-pipeline models, five models could be compared at a time. Due to the increased number of weights, only two to four multiple pipelines models could be compared for each run, usually varying a single model parameter. To compensate for this, many training runs needed to be conducted to make enough comparisons between models of different parameters. Early stopping was also used to reduce the time spent training, with patience being set to 20 epochs.

Adadelta was used as the optimization method for training, with mean square error set as the loss function. Adadelta has several advantages over traditional optimization methods. Unlike Stochastic Gradient Descent, Adadelta does not require setting a training rate, which eliminates having to optimize an additional parameter [22]. It is also robust to different model architecture choices [22], which is important in our case due to our having to compare models with varying numbers of layers, as well as the parameters for those layers.

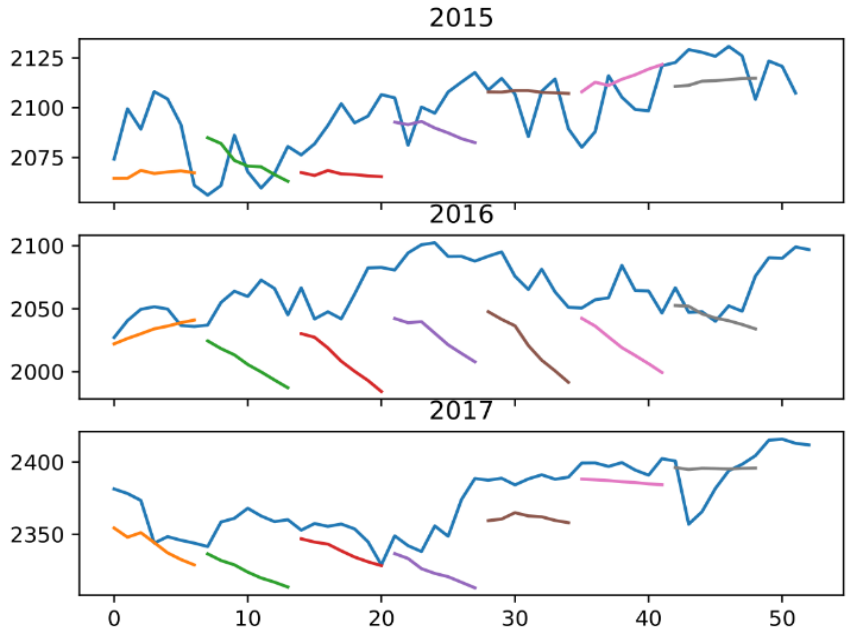

Figure 5: Single Pipeline Deep Learning Model predictions on S&P 500 Dataset. X axis shows the time segment for the index and prediction window. Y axis shows the closing price. Continuous plot is the actual closing price, predictions are shown as disjoint plots.

7. EXPERIMENTAL RESULTS AND DISCUSSION

7.1 Support Vector Regressor Model Results

Considering that in the past support vector regressor has received much attention due to its success in stock market prediction, we decided to compare it's results with deep learning models. The results from top four Support Vector Regressor models are displayed in Table 1. A total of 5000 models were tested and five folds cross validation was performed using different kernels and different values for C and gamma. Performance metric is mean squared error (MSE).

The radial basis function (rbf) was the best performing kernel out of the three tested: RBF, linear, and polynomial, with the degree set to two for the latter. This was expected, as RBF is better suited to handling data that is non-linear in nature compared to the linear and polynomial kernels.

7.2 Single Pipeline Deep Learning Model Results

While searching for optimized parameters, we found that certain parameter values made the model more sensitive to changing values for other parameters. For example, looking at the MSE results of training the single pipeline model in Table 2, we found that setting the CNN layers to have a kernel size of 9 gave best performing models, out of the three compared, with the top and bottom models set to 400 and 200 LSTM units respectively.

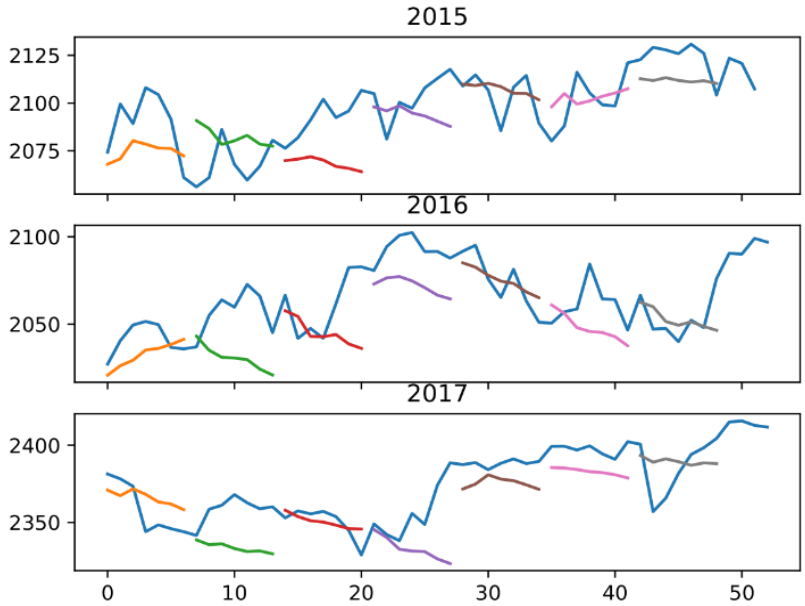

Figure 6: Proposed Multiple Pipelines Deep Learning Model predictions on S&P 500 Dataset. X axis shows the time segment for the index and prediction window. Y axis shows the closing price. Continuous plot is the actual closing price, predictions are shown as disjoint plots.

7.3 Proposed Multiple Pipelines Deep Learning Model Results

Examining Table 3, containing the multiple pipelines model results, we observe the best performing model outperforms the single pipeline model. The top performing model used 400 units, while the bottom used 200. Interestingly, with the 400 unit models, the one with larger step sizes performed better, while with the 200 unit model, the one with the smaller step size had a better test score.

7.4 Comparison of Results across Models

Multiple pipelines models generally outperformed the single pipeline models and support vector regressor. Best performing multiple pipelines model obtained 20% better MSE than single pipelining model and improved by a factor of five compared to best results from support vector regressor. Our idea that using multiple pipelines would allow the model to combine the strengths of the three different kernel sizes was reflected in the results. Best performing single pipeline model outperformed support vector regressor by a factor of four.

The top-scoring models were used to generate graphs of the predictions for each year from January 1st to June 1st, with each segment representing the next seven predictions based on the previous fifty points. See figs. 4, 5, 6. The predictions from the multiple pipelines model fig. 6 seem to follow the actual prices more closely than the single pipeline model fig. 5 and support vector regressor fig 4 and is willing to make larger changes in direction.

Both deep learning models greatly outperformed the support vector machine approach, both in terms of after examining the MSE and from looking at graphs of the predictions. Even comparing the performance of the best support vector machine model to the worst multiple pipelines model, the deep learning approach performed roughly three times better. Much of this is due to the simplicity of a pure support vector machine approach.

8 CONCLUSION AND FUTURE RESEARCH

In this paper we proposed a new deep learning model that combined the power of multiple pipelines of CNN in extracting features from data and LSTM in analyzing sequential data. Our model improved accuracy in stock market prediction upon the popular support vector regressor model and single pipeline deep learning models. We also presented multiple variations of our model to show how we have increased accuracy.

Future research would apply deep learning towards classification of stock direction, rather than using it for regression. Classification of direction would result in a greatly simplified problem space compared to prediction of stock prices. Another area of research to pursue would be to include stock sentiment and event analysis in stock market prediction.

REFERENCES

[1] N. Musunuru and J. Patton, "Examining Random Walk Hypothesis on Major World Financial Indices," *Indian Journal of Economics & Business,* vol. 11, no. 2, pp. 587-602, 2012.

[2] C. R. Harvey and Y. Liu, "Evaluating Trading Strategies," *Journal of Portfolio Management,* vol. 40, no. 5, pp. 108-118, 2014.

[3] J. B. Heaton, N. G. Polson and J. H. Witte, "Deep Learning for Finance: Deep Portfolios," *Applied Stochastic Models in Business and Industry,* pp. 3-12, 2017.

[4] Y. Zheng, Q. Liu, E. Chen, Y. Ge and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks," *Frontiers of Computer Science,* pp. 96-112, 2016.

[5] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM,* vol. 60, no. 6, pp. 84-90, 2017.

[6] M. Dixon, D. Klabjan and J. H. Bang, "Classification-based Financial Markets Prediction using Deep Neural Networks," *Algorithmic Finance,* 2016.

[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, 2016.

[8] J. A. Tepper, M. S. Shertil and H. M. Powell, "On the importance of sluggish state memory for learning long term dependency," *Knowledge-Based Systems,* vol. 96, pp. 104-114, 2016.

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

[10] W. Bao, J. Yue and Y. Rao, "A deep learning framework for financial time," *PLoS One,* 2017.

[11] P. Konana and S. Ram, "Semantics-Based Transaction Processing for Real-Time Databases: The Case of Automated Stock Trading," *INFORMS Journal on Computing,* vol. 11, no. 3, pp. 299-315, 1999.

[12] H. Grigoryan, "A Stock Market Prediction Method Based on Support Vector Machines (SVM) and Independent Component Analysis (ICA)," *Database Systems Journal,* no. 1, pp. 12-21, 2016.

[13] "Robust Stock Value Prediction using Support Vector Machines with Particle Swarm Optimization," *Evolutionary Computation (CEC), 2015 IEEE Congress on,* pp. 3327-3331, 2015.

[14] W. Chen, Z. Hao, R. Cai, X. Zhang and Y. Hu, "Multiple-cause discovery combined with structure learning," *Soft Computing,* vol. 20, no. 11, pp. 4575-4588, 2016.

[15] V. Galvani and S. Gubellini, "Mean–variance dominant trading strategies," *Finance Research Letters,* vol. 10, no. 3, pp. 142-150, 2013.

[16] C. Pereira and A. Dourado, "On the Complexity and Interpretability of Support Vector Machines for Process Modeling," *Proceedings of the 2002 International Joint Conference on Neural Networks,* vol. 3, pp. 2204-2209, 2002.

[17] H. Ince and T. B. Trafalis, "Short term forecasting with support vector machines and application to stock price prediction," *International Journal of General Systems,* vol. 37, no. 6, pp. 677-687, 2008.

[18] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang and C. W. Cottrell, "A Dual-Stage Attention-Based Recurrent Neural Network," pp. 2627-2633, 2017.

[19] C. Zhu, J. Yin and Q. Li, "A Stock Decision Support System based on DBNs," *Journal of Computational Information Systems,* pp. 883-893, 2014.

[20] A. A. Bhat and S. S. Kamath, "Automated Stock Price Prediction and Trading Framework for Nifty Intraday Trading," *Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on,* pp. 1-6.

[21] "SciKit-Learn," [Online]. Available: http://scikit-learn.org/. [Accessed 16 5 2018]

[22] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *arXiv:1212.5701 [cs.LG],* 2012.

[23] 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.

[24] Nvidia, "GeForce GTX 780 Graphics Card - Quiet, Powerful, Fast PC Gaming |Nvidia," 23 May 2013. [Online]. Available: http://www.nvidia.in/object/geforce-gtx-780-in.html#pdpContent=2. [Accessed 16 May 2018].

[25] Google LLC, "Graphs and Session | TensorFlow," 28 April 2018. [Online]. Available: https://www.tensorflow.org/programmers_guide/graphs#why_dataflow_graphs. [Accessed 16 May 2018].

[26] "Keras Documentation," [Online]. Available: https://keras.io/. [Accessed 16 5 2018].

[27] S. Boonpeng and P. Jeatrakul, "Decision Support System for Investing in Stock Market by using OAA-Neural Network," *Advanced Computational Intelligence (ICACI), 2016 Eighth International Conference on,* pp. 1-6, 2016.