Examine different neural networks' efficiency in predicting stock prices

Lam Nguyen
Earlham College
Richmond, Indiana
ltnguyen14@earlham.edu

ABSTRACT

There has been a lot of attempts in building predictive models that can correctly predict the stock price. However, most of these models only focus on different in-market factors such as the prices of other similar stocks. This paper discusses about the efficiency/accuracy of three different neural network models (feedforward, recurrent, and convolutional) in predicting stock prices based on external dependencies such as oil price, weather indexes, etc.

KEYWORDS

Machine Learning, Neural Network, Stock Prediction

1 INTRODUCTION

1.1 Outline of the paper

Section 1 is used to provide a brief outline of the paper, as well as discuss goals and results of this project. In order to help the readers to have a better understanding about the models involved in this paper, Section 2 is used to introduce some of the most fundamental concepts and definitions that will be used in this paper. Section 3 is devoted for introducing current advancement in stock prediction methodologies. Section 4 talks about the software architecture used in this project, with the results and discussions in section 5. Section 6 is the conclusion and future work.

1.2 Assumptions and goals

In this paper, I am assuming that there is a strong correlation between the oil price and the oil company's stock price (specifically Exxon Mobil). However, the project is implemented in a way that can easily be scaled with various number of input features, not limited to one. This paper will try to see if the correlation is strong enough that we can predict the stock price trend, and also examine the accuracy of three different basic neural network models in doing this task.

2 TERMINOLOGY AND DEFINITIONS

In order to follow this paper, some fundamental knowledge about neural network models and machine learning in general is advised. Following are some terminologies and definitions that will help the readers to better understand the project.

2.1 Machine Learning and Statistical Classification

In the field of computer science, machine learning is the study of algorithms that can learn and make predictions from data, not from strictly static program instructions. In other words, machine learning gives the computer the ability to learn without being explicitly programmed.

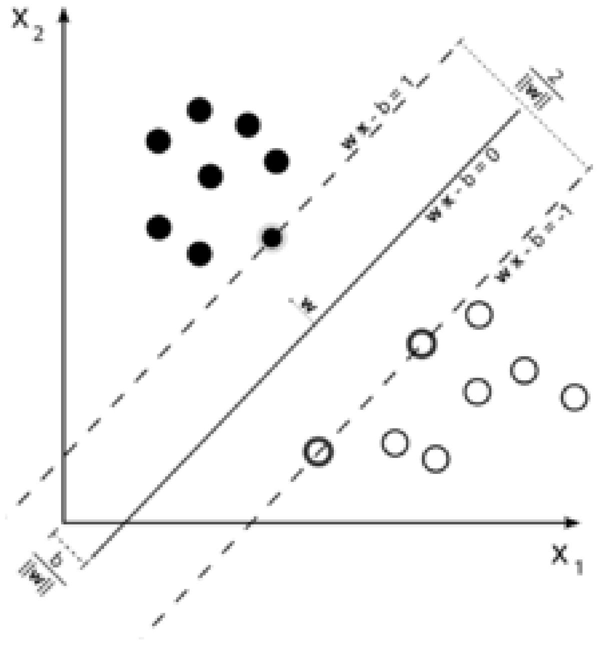

Figure 1: Support Vector Machine [3]

In the field of machine learning, statistical classification is the problem of assigning a new observation into a category based on a training set of data where the category is known. Classification is an example of pattern recognition. An algorithm that implements classification is known as a classifier, which map input data to a category.

Supervised learning is the machine learning task of inferring a function from labeled training data, meaning that the training data consist of a set of training examples. The training data set normally include pair of input and desired output. The algorithm analyzes the training data set that was given and then can be used to map new examples.

2.2 Regression Analysis

In statistical modeling, regression analysis is a process for estimating the relationship among variables. In other words, regression analysis helps us to study how the value of a dependent value changes when we vary one independent variable while keeping the other independent variable constant.

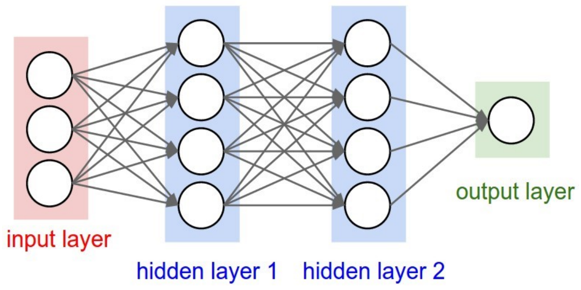

Figure 2: A simple feedforward neural network [2]

2.3 Support Vector Machine

Support vector machines (Figure 1) are supervised learning models that analyze data used for classification and regression analysis. SVM (support vector machine) is a binary linear classifier, which means it's mainly used to classify its inputs into one of two different classes. More specifically, an SVM model constructs a hyperplane, or multiple hyperplanes in order to classify its input into groups, thus providing some useful applications such as outliers detection. Besides performing linear classification, SVM can also perform non-linear classification using a kernel method, implicitly mapping their inputs into high-dimensional feature spaces.

2.4 Feedforward Neural Network

Feedforward Neural Network (Figure 2) is the most basic neural network model that uses back propagation to adjust the weights while training. It was the first and simplest type of artificial neural networks devised. A feedforward neural network tries to mimic what we currently know about a human neural network, with one or multiple hidden layers. Each connection in the model has its own weight and bias, which determine its significance toward the finally output.

2.5 Recurrent Neural Network with Long Short-Term Memory (LSTM)

Recurrent Neural Network (Figure 3) is a neural network model where connections between units form a directed cycle. Unlike feedforward neural networks, recurrent neural networks can use their internal memory to process an arbitrary sequence of inputs, which means the inputs'sizes doesn't have to stay constant. This makes the model very efficient when dealing with languages or time series data.

Long short-term memory (LSTM) can be seen as a very simple neural network that can be used to build a more complete and more complex recurrent neural network. It composed by four main components: a cell, an input gate, an output gate, and a forget gate. As the name suggest, LSTM is responsible for remembering values over arbitrary time intervals. This ensures that the recurrent neural network that is using LSTM is able to remember values that is significant but avoid storing every values which is very inefficient and would reduce the efficiency of the model.

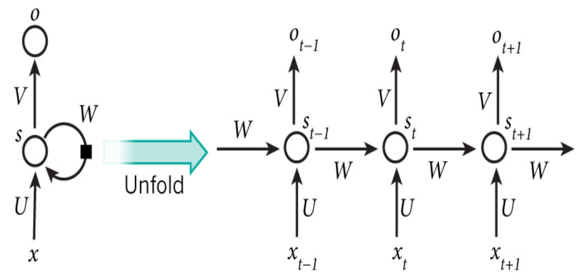

Figure 3: A recurrent neural network with LSTM [4]

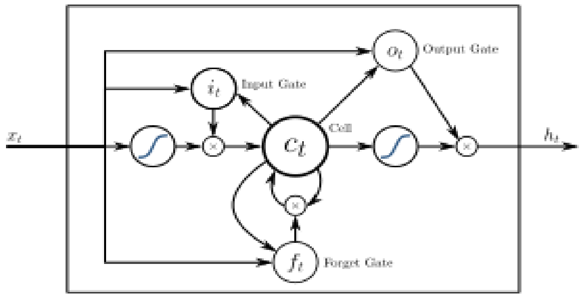

Figure 4: A long short term memory block [4]

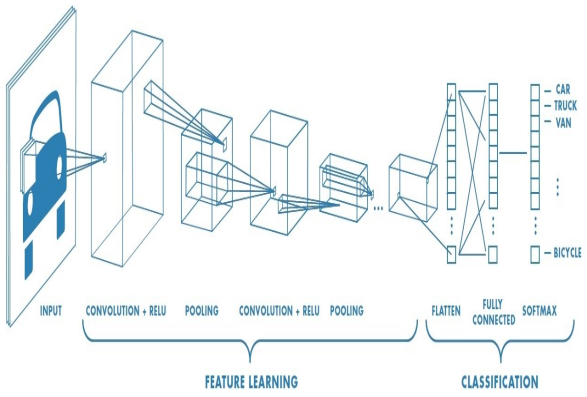

Figure 5: Convolutional Neural Network [1]

2.6 Convolutional Neural Network

Convolutional Neural Network (Figure 5) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. However, the application of Convolutional Neural Network can be found in different field such as music, art, etc. by converting the input into a stream of pixels. This trick enables us to turn music sheets into images, thus giving the network the ability to analyze and even reproduce music that is similar to the input.

In the most basic form, a convolutional neural network has four significant parts: inputs, feature learning, classification (by using fully-connected layers similar to a feedforward neural network), and outputs. As the name suggests, the feature learning part tried to detect significant features that is useful for the classification problem that it's trying to solve. This is done by applying two different methods on the image called pooling and convoluting. Their main use is to simplify the image, thus making it much easier to detect trends, edges, etc.

In the end, a fully-connected layer is applied in order to transform the result from the feature learning part into output. For example, a model that is specialized in detecting if a cat is in an image would produce two outputs: the percentage that a cat is in the image, and the percentage that a cat is not.

3 CURRENT ADVANCEMENT IN STOCK PREDICTION METHODOLOGIES

3.1 Using support vector machine model

Huang et al. [5] and Yang et al. [8] both discusses the possibility of using Support Vector Machine to predict stock market movement direction.

Per Huang [5]), Japanese's economy growth has a close relationship with Japanese export, which in turn makes the United States's (Japanese's main export target) economic condition determines Japan economy. The USA's economy is represented by the S&P 500 Index, while Japan economy is represented by the NIKKEI 225 Index. In this experiment, S&P 500 Index is served as input for the Support Vector Machine model. To evaluate the forecasting ability of SVM, the authors use the random walk model (RW) as a benchmark for comparison. They also included linear discriminant analysis (LDA), quadratic discriminant analysis (QDA) and elman backpropagation neural networks (EBNN). A combined model is also developed using different weights for different classification method. The result is fascinating, with SVM's hit ratio was around 73%, the highest compare to another method. The combined method has even higher hit ratio with 75%. This is very impressive and prove their initial assumption, that is the Japanese and US's economy are heavily dependent on each other. Since the authors only applied their model to S&P 500 Index and KIKKEI 255 Index, it does not actually predicting stock prices but rather the overall movement of the whole market. However, it begs the question of how well would these models work in predicting stock prices or stock trend.

Yang et al. [8] try to apply Support Vector Regression (SVR) to financial prediction tasks. They propose an improved model based on a normal SCR model, which consider margins adaptation. When using SVM in regression tasks, the SVR need to use a cost function to calculate the risk to minimize the regression error. The margin used by those function is very important because when the margin is zero and very small, it is possible to over-fit the data with poor generalization. On the other hand, if the margin is too high, one run into the risk of having higher testing error. For financial data, because of the embedded noise, one must use a suitable margin to obtain a good prediction. Two experiments were conducted to illustrate the effect of FASM (Fixed and Symmetrical Margin), FAAM (Fixed and Asymmetrical Margin), and NASM (Non-fixed and Symmetrical Margin). Through their finding, they concluded that in financial applications, setting a suitable margin is critical to the performance of the prediction tool used.

3.2 Using modular neural network

Kimoto et al. [6] discuss a buying and selling timing prediction system for stocks on the Tokyo Stock Exchange and analysis of internal representation, using a modular neural network algorithm. The input consists of several technical and economic indexes, including some neural networks learned the relationships between the past technical and economic indexes and the appropriate buy/sell time. A prediction system that is made up of modular neural networks was claimed to be correct, with the simulation of buying and selling stocks using the prediction system shows an excellent profit. Below is the graph that the authors used to describe the overall architecture of the prediction system. Using different indexes such as turnover rat, foreign exchange rate, etc. the data is then passed to a preprocessor before passed on to neural networks to predict the right time to buy and sell stocks.

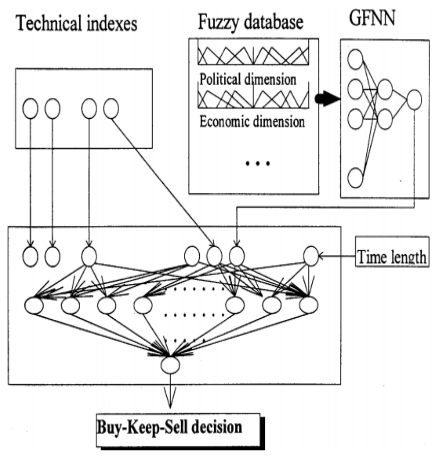

Figure 6: Fuzzy neural network architecture [7]

The authors use high-speed learning algorithm called supplementary learning, which is based on the error back propagation. The algorithm automatically schedules pattern presentation and changes learning constants when needed. In supplementary, the weights are changed according to the sum of error signals after presentation of all learning data.

3.3 Genetic algorithm based fuzzy neural network

Kuo et al. [7] developed a genetic algorithm based fuzzy neural network (GFNN) to formulate the knowledge base of fuzzy inference rules which can measure the qualitative effect (e.g., political effect) on the stock market.

The methodology the authors used is described in figure 6. This study develops an intelligent stock trading decision support system based on the viewpoint of system integration. There are three main parts in this model: factor identification, qualitative model (GFNN), and decision integration.

4 SOFTWARE ARCHITECTURE

4.1 Overview

This project assume that there is a strong correlation between the oil price and the Exxon Mobil stock price, thus tries to derive different predictive models to predict the stock price based on the oil price.

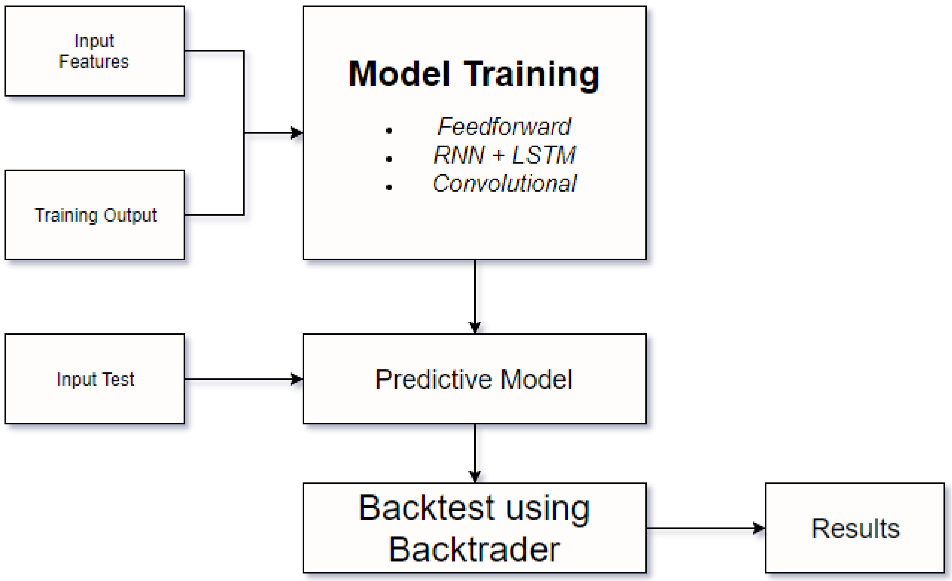

Figure 7: Software architecture

The project has three main parts: data collection and process, creating and training predictive models, and testing these models. The first part, data collect and process is pretty self-explanatory, include collecting data of oil prices and Exxon Mobil's stock prices. The second part derives three different predictive models (feedforward, recurrent, and convolutional neural network) using the data from the first part to train and test. The last part utilizes the models developed in the second part to test on the real world data by using two different methods: an accuracy score and a backtest pipeline.

The project was written in Python using Tensorflow, pandas, numpy module and matplotlib module to plot images. Backtrader module was also used in the project to implement the testing pipeline that determine the efficiency of the models.

4.2 Data collection and process

For this project, the two data sources that was used are the oil price of the last 25 years and the stock price of Exxon Mobile, one of world's biggest oil company. The theory that I wanted to test was that based on the oil price, we can predict the general trend of the stock price using different neural network models, as well as compare the effectiveness of these models.

The data set is split into two subsets: training set (3/4 of the data points) and testing set, which contains the rest of data points. The training set would be used for training and optimization of the models, while the testing set is used to determine the model's accuracy. It is essential to separate the training and testing data sets, since this will help us to avoid the problem of false positive, thus increase the credibility of the accuracy score.

4.3 Introduce lag into the models

Because this is a predictive model, it is important to introduce lag into the model. Lag is simply a fixed period of time that in theory reflect the amount of time it takes of the input features to affect the stock price. To determine the best lag possible, I repeatedly train the models with different lags ranging from 1 to 60 days. To fast-track the learning process, I slightly increased the learning rate of the model. By doing this, the model would be able to learn much faster, but is prone to several problems such as local minimum, etc. However, since I only want to get the general idea of how well a lag would perform regarding the predictive results, increasing the learning rate is acceptable in this case.

Once an optimal lag is determined, I proceed by saving the lag to the disk, which would be handy during the back-testing phase. The lag is then introduced into the model during the actual training phase by realign the training data set. This in theory will give the model just enough information that it would have to predict the stock price.

4.4 Models implemented

4.4.1 Feedforward Neural Network. First model used was a simple feedforward neural network. Since feedforward neural network can only accept a fixed size input, we had to feed the model each data point separately in both training and testing phases. The idea behind this model is to try to find the correlation between the input feature and the output, assuming that the correlation is strong enough for us to predict the stock price in the future using the feature data points and the lag between them.

The learning rate is fixed at 0.001 and multiple instances of the model were created. Since the initial weights were created randomly, it is important to repeat the training phase multiple times in order to get the best result possible. I used rectified linear unit the activation function and mean square error (MSE) as our loss function. MSE is widely used as the loss function in predictive model.

4.4.2 Recurrent neural network with LSTM. The next model implemented was a convolutional neural network with long short term memory. Because of the nature of this model, it is ideal to

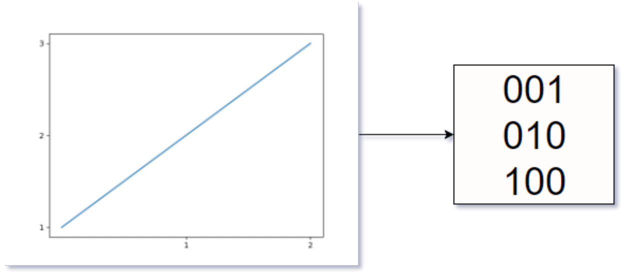

Figure 8: Data preparation for the convolutional neural network

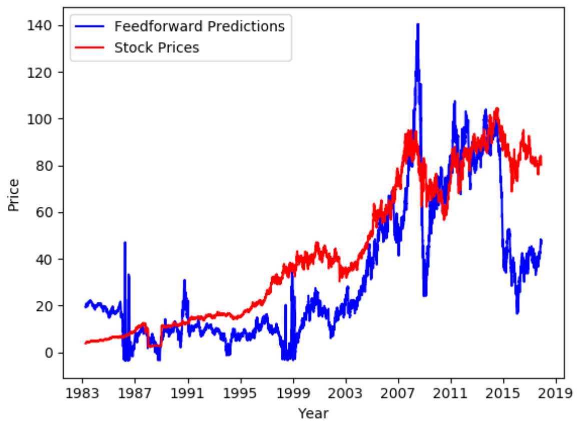

Figure 9: Feedforward neural network predictions

deal with continuous series of data, which is exactly what we have here. This model is expected to outperform the feedforward neural network because this model can take into consideration the relationship between different data points.

Because this model excels at working with time series inputs, the input features are divided into small chunks consist of five days data, since the market is open five day a week. Notice that this approach is different from the approach that we took for the feedforward model, where the inputs were single data points. This allows the model to use the long short term memory to potentially figure out the correlation between the data points in the series.

The setup for this model is very similar to the feedforward neural network, with the learning rate fixed to 0.001, and MSE was used as the loss function.

4.4.3 Convolutional neural network. The last model used was a Convolutional Neural Network. Even though Convolutional Neural Netowrk (CovNet) is used primary in the computer vision field, it is still worthwhile to explore its potential as a stock prediction model. Since it requires picture(s) as input, it was necessary to turn the input data into stream of pixels. To achieve this, I turn the input data into chunks of 20 data points and then turn them into a 4 dimensional arrays (since that's what an image is). A simplified

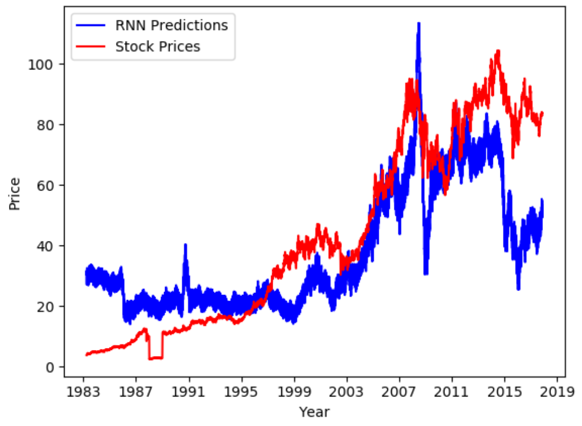

Figure 10: Recurrent neural network predictions

version is demonstrated in figure 8. The output used for training is an array of 20 data points.

I have also added a dropout rate of around 20%. A dropout rate is used to mimic the dead neurons that can be found in our biological brain, and serves as a regularization technique for avoiding overfitting by preventing complex co-adaptations on training data.

4.5 Determine models' accuracy

I implemented two different methods to determine the model's effectiveness: an accuracy score and a backtest pipeline. An accuracy score is calculated by running the model on a testing data set and consider any output that is \$5 within the actual stock price as correct. The second method want to measure how well the model can work in the real world. In order to implement the backtest pipeline, I used Python module Backtrader and develop different strategies that can utilize the model's prediction. The pipeline will have the ability to sell or buy stock without knowing the actual stock price and only knowing the prediction of the model. The strategies that I developed for these models are very simple and basic, but should serve as a good baseline to compare the three models results.

5 RESULTS AND DISCUSSION

5.1 Feedforward Neural Network

The Feedforward Neural Network produced some encouraging results, with the shape of prediction line somewhat follow the trend of the real stock price (Figure 9). The average accuracy for the model is only 12%, with the highest accuracy score recorded was 30%. The reason that the accuracy score changes between runs is because all initial weights and biases were randomized at the beginning, so it makes sense that the performance of the model will change depends on the initial values. Figure 9 also shows that even though the prediction follows the real stock price relatively closely but react very strongly when there is an apparent strong change in the trend of stock price.

When using the backtest pipeline, I implemented a virtual broker that ask for 2% commission to make the transactions more realistic.

Table 1: Models' accuracy score

Model	Average accuracy score (%)	Highest accuracy score (%)
`Feedforward Neural Network`	12	30
`Recurrent Neural Network & LSTM`	23	50
`Convolutional Neural Network`	0	0

Table 2: Models' backtest results

Model	Average backtest result ($)	Highest backtest result ($)
`Feedforward Neural Network`	99 900	100 400
`Recurrent Neural Network & LSTM`	100 390	110 500
`Convolutional Neural Network`	100 000	100 000

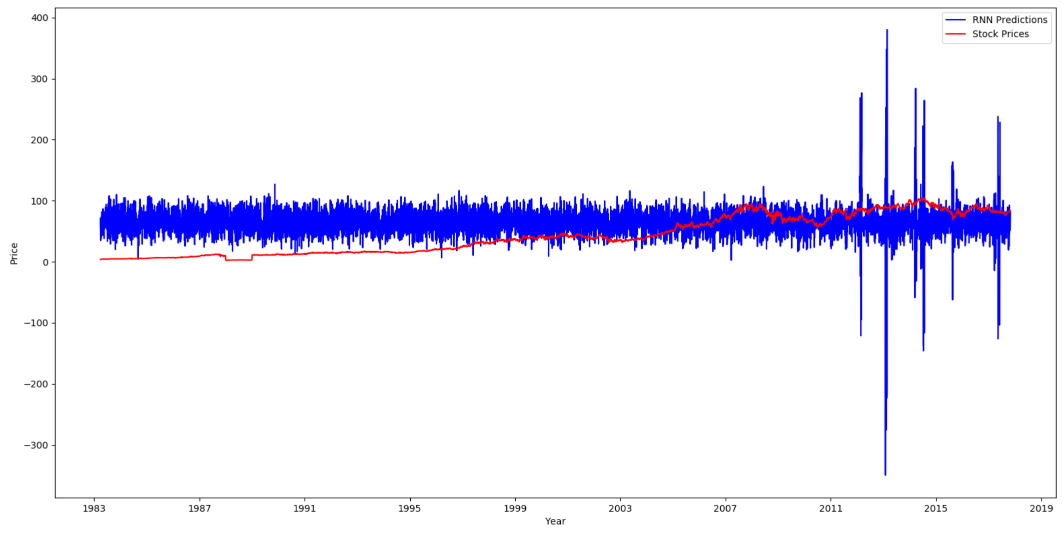

Figure 11: Convolutional neural network predictions

The initial cash amount for all models are $100 000. In this model, in the course of roughly last 7 years, the model is in average end up with the total amount of $99 900, which means the model actually lost money during the backtest process. However, the best run actually end up with $ 100 400, making a profit of $400.

5.2 Recurrent Neural Network with LSTM

The Recurrent Neural Network with LSTM produces the best result out of the three models, scored in average 23% in accuracy score and 50% in the highest run. This makes sense since recurrent neural network is specialized in working with time series data and is perfect for this problem. However, as figure 10 indicates, there are definitely a lot of similarity in the result of this model compared to the feedforward's results as the general shapes of the two are very similar. The explanation is because they both have the same input feature data points (oil price), the result adopts the general shape of the input, thus resulting both outputs to have similar shapes.

Using the backtest pipeline with the same values as I used for the feedforward neural network, the model perform much better than expected. The average result is $100 390, which makes the average profit of $390. The highest profit that the model managed to produce was $10 500, which is more than 10%. Keep in mind that the strategy developed is very conservative and always choose to play safe, which means this model has a lot of potential.

There is also one very interesting phenomena happened in the result of the model, that is the shape of the result has a repeated pattern as in figure 13. This teeth-like pattern repeats throughout the whole prediction results, which suggests one of the two possibilities: it is an artificial product or there is actually a pattern that is happening in the input feature data points. It makes sense if it is

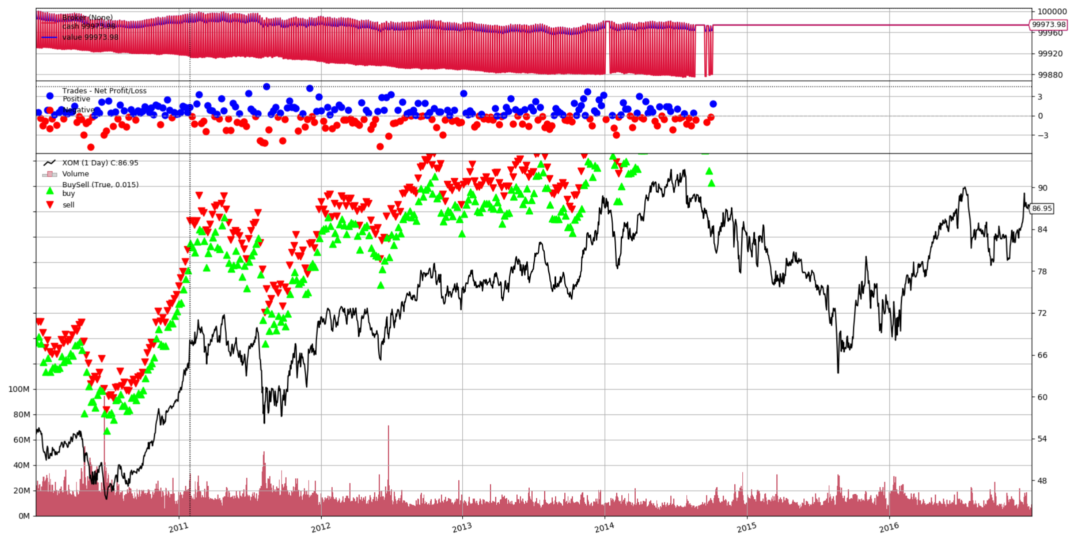

Figure 12: Backtesting result from applying feedforward model

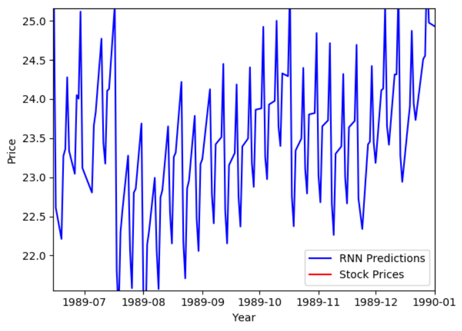

Figure 13: RNN's result teeth pattern

a product of a pattern in the feature data points as the input was divided into chunks of five days, which represents weekly values of input and output data points. However, since the pattern is very consistent, it is very likely to be an artificial product and need to be fixed in the future.

5.3 Convolutional Neural Network

The Convolutional Neural Network performed the worst out of the three models. As figure 11 suggests, the result predictions are very unstable does not very useful in predicting the stock price. Thus the highest accuracy score is 0%, which also means the mean average score is also 0%. This is disappointing but understandable, since convolutional neural network was built in order to process complex images, not data points. Since the backtesting pipeline implemented was very conservative, it did not buy or sell any stock during the whole 7 years, which means the final amount of money remains $100 000. This is some optimizations/changes that can in theory help the model to produce a better result, which I will talk about in the future work.

6 CONCLUSION AND FUTURE WORK

Stock price prediction has always been seen as nearly impossible and provides a lot of challenges. However, it did not stop many people to try out different techniques trying to predict how the stock price will react when a known variable is changed (weather indexes, oil price, etc.). In this paper, I tried three very basic neural network models in order to compare their effectiveness in predicting stock price and get some very interesting results. Feedforward Neural Network, despise its simplicity, performs rather well and was able to make a small profit of 0.3%. A recurrent neural network outperform feedforward neural network with a profit of around 10%. The disappointing model was convolutional neural network which produced 0% profit.

The following things are things that I did not have enough time to implement, but should in theory help the models perform much better.

6.1 Dynamic learning rate

Right now for all three models the learning rate is fixed at 0.001. This is known as a good learning rate for a neural network since it is extremely important to not set it too high, as the model will stuck in local minimum, or set it too low as the model might never converge. However, by implementing a dynamic learning rate (High at the beginning and low after that), the models might learn much faster and will be able to find the best possible result without setting the learning for too low and have to wait an unrealistic amount of time.

6.2 Genetic algorithm to find best strategy

The strategies that I implemented for backtesting pipeline are all manual conditions that determine if the model should buy or sell at a particular time. However, this is not the best approach to the problem. One possible approach to the problem is to use a genetic algorithm to try to find the best strategy based on the prediction that the models produce. By using genetic algorithm on the predictive results, it can still be used as a way to compare the models' efficiency but will reflect the potential of the models much better.

6.3 Convolutional Neural Network Input

To create the input for the Convolutional Neural Network, I had to convert it into a 4 dimensional array as in figure 8. However, it is not the best approach to the problem as it as to round float into the nearest integer, which would in theory reduce the accuracy of the model. A better way to convert the input into an image would be much more preferred and in theory can significantly increase its performance.

7 ACKNOWLEDGEMENT

This work was possible through the guidance and support of my senior capstone advisers/mentors Dave Barbella and Charlie Peck. I would also like to thank the CS Department at Earlham for funding the project, and the tensorflow.org forum community for their valuable discussions.

REFERENCES

[1] [n. d.]. Convolutional Neural Network. ([n. d.]). https://www.mathworks.com/discovery/convolutional-neural-network.html

[2] 2017. Feedforward neural network. (Sept. 2017). https://en.wikipedia.org/w/index.php?title=Feedforward_neural_network&oldid=802923925 Page Version ID: 802923925.

[3] 2017. Support vector machine. (Dec. 2017). https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=814011558 Page Version ID: 814011558.

[4] Denny Britz. 2015. Recurrent Neural Networks Tutorial, Part 1 âĂŞ Introduction to RNNs. (Sept. 2015). http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[5] Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. 2005. Forecasting stock market movement direction with support vector machine. *Computers & Operations Research* 32, 10 (Oct. 2005), 2513–2522. https://doi.org/10.1016/j.cor.2004.03.016

[6] T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka. 1990. Stock market prediction system with modular neural networks. In *1990 IJCNN International Joint Conference on Neural Networks*. 1–6 vol.1. https://doi.org/10.1109/IJCNN.1990.137535

[7] R. J. Kuo, C. H. Chen, and Y. C. Hwang. 2001. An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. *Fuzzy Sets and Systems* 118, 1 (Feb. 2001), 21–45. https://doi.org/10.1016/S0165-0114(98)00399-6

[8] Haiqin Yang, Laiwan Chan, and Irwin King. 2002. Support Vector Machine Regression for Volatile Stock Market Prediction. In *Intelligent Data Engineering and Automated Learning âĂŤ IDEAL 2002 (Lecture Notes in Computer Science)*, Hujun Yin, Nigel Allinson, Richard Freeman, John Keane, and Simon Hubbard (Eds.). Springer Berlin Heidelberg, 391–396. http://link.springer.com/chapter/10.1007/3-540-45675-9_58 DOI: 10.1007/3-540-45675-9_58.