Deep Learning for the Prediction of Stock Market Trends

Arvand Fazeli
Brock University
Department of Computer Science
St. Catharines, Ontario, Canada
af17tv@brocku.ca

Sheridan Houghten
Brock University
Department of Computer Science
St. Catharines, Ontario, Canada
shoughten@brocku.ca

***Abstract*—In this study, deep learning will be used to test the predictability of stock trends. Stock markets are known to be volatile, prices fluctuate, and there are many complicated financial indicators involved. Various data including news or financial indicators can be used to predict stock prices. In this study, the focus will be on using past stock prices and using technical indicators to increase the performance of the results. The goal of this study is to measure the accuracy of predictions and evaluate the results. Historical data is gathered for Apple, Microsoft, Google and Intel stocks. A prediction model is created by using past data and technical indicators were used as features in the model. The experiments were performed by using long short-term memory networks. Different approaches and techniques were tested to boost the performance of the results. To prove the usability of the final model in the real world and measure the profitability of results backtesting was performed. The final results show that while it is not possible to predict the exact price of a stock in the future to gain profitable results, deep learning can be used to predict the trend of stock markets to generate buy and sell signals.**

I. Introduction

There has been much research on the predictability of stock markets, and although researchers have different opinions, many empirical studies show that some aspects of stock markets can be predicted [21].

There are multiple ways to predict the volatile prices of stocks. Information, including news, tweets, technical indicators, and fundamental indicators, can be processed to later find patterns for predicting future prices. For making investment decisions machine learning models can be incorporated to make such predictions.

In this study, deep learning along with technical indicators are used to predict the price of stocks. A suite of long short-term memory (LSTM) networks is developed for a range of time series prediction. Different architectures are tested to improve the performance, and we propose a new solution to create a profitable model: instead of predicting the prices, we focus on predicting the trends. The difference between the values of LSTM prediction and backtesting is explained and we test the profitability of the model by using backtesting to reach practical results.

The remainder of this paper is structured as follows. Section II gives an introduction to deep learning and technical analysis. LSTM networks are explained and technical indicators are also described. Section III reviews previous related work. Sections IV and V are the primary sections describing the conducted experiments and results. Section IV describes the dataset, and the processing of the data. We also describe the experiment that is designed for building the model and training the dataset. Section V elaborates experimental results from using LSTM network. Section VI provides conclusions and discusses possible future work.

II. Background

Although there are multiple ways to predict stock markets, the most common ones use either technical indicators, which focus on historical trading data, or fundamental data, which focuses on financial statements such as revenue of a company.

A. Technical Analysis

Technical analysis uses past market data to predict the direction of prices [28], and is based on using statistical methods to identify patterns.

Technical indicators are mathematical calculations that use past price and volume to identify the direction and strength of market trends. They can be broken down into four major types: trend, momentum, volume and volatility.

Some indicators are more favored than the others and have been proven more useful in past empirical studies. In [23], the profitability of MACD and RSI (both defined below) are evaluated, and these indicators are concluded to be profitable for some stocks. The author of [29] states that the efficiency of a back propagation neural network was most improved by the addition of MACD. In [22] the authors find that RSI and MACD outperform the buy-and-hold strategy.

1) Relative Strength Index (RSI): The RSI is a momentum indicator that can signal oversold or overbought securities [12]. RSI ranges between 0 and 100; a stock is usually considered overbought when RSI goes above 70 and oversold when it goes below 30. Some analysts use other data ranges such as 80 and 20 or 90 and 10. RSI is typically used on a 14-day time frame and is calculated by $RSI = 100\frac{100}{(1+RS)}$, where $RS = \frac{AverageGain}{AverageLoss}$.

2) Moving Average Convergence Divergence (MACD): MACD is a trend indicator to reveal changes between two moving averages of a security price [6]. It is calculated by subtracting the 26-day exponential moving average (EMA) from the 12-day EMA. The exponential moving average (EMA) is a weighted moving average (WMA) that gives more weight to recent data. The result will be a 9-day EMA of the MACD, also referred to as signal line. This line can be used as a buying signal when the MACD crosses above its signal line. MACD helps investors understand whether the uptrend or downtrend is getting stronger or weaker [6].

3) MACD Histogram: MACD is usually displayed along with a histogram. When the MACD is below the signal line, the histogram will be below the baseline and when it is positive the values are reflected on the MACD histogram [6].

The MACD histogram, created by Thomas Aspray, measures the difference between MACD and its signal line (the 9-day EMA). It was developed to show crossovers in MACD and generate trading signals [7]. The MACD histogram can be used as a potential buy signal when it is below the zero line and begins to converge towards it, and as a potential sell signal when it is above the zero line and begins to converge towards it [8].

4) Williams %R: Williams %R can be used to find entry and exit points in the market, it compares a stock's closing price to the high-low range over a specific period, typically 14 days or more [16]. This indicator is helpful in showing the difference between the period high and closing price within the range of days. Williams %R is calculated as $(-100) * \frac{Highest\ High - Closing\ Price}{Highest\ High - Lowest\ Low}$, where *highest high* is the highest price over the trading period and *lowest low* is the lowest price over the same period.

5) Volatility: Volatility is the rate at which the price increases or decreases for a given set of returns. It is used as an indication of the amount of risk related to a security's value. High volatility value is an indication that the price can fluctuate drastically in either direction. Mathematically, it is the standard deviation calculated over a time period.

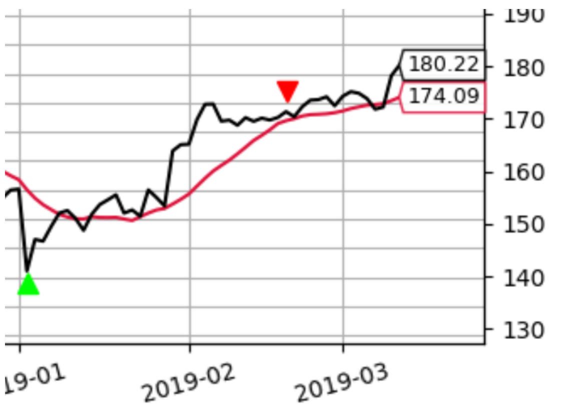

Fig. 1: Buy (green arrow) and sell (red arrow) signals generated by system

B. Backtesting

Backtesting is used to measure the performance of a trading strategy. Backtesting works by simulating trades with past data to determine if the trading strategy is profitable or not. A trading strategy is a strategy to trade stocks based on predefined rules [14]. For this purpose we need to create buy and sell signals, as shown in Figure 1; these signals can be generated by a system. If the results of backtesting are positive it can be an indicator that the trading strategy is successful. After training our neural network, we generate buying and selling signals based on the trading strategy. For backtesting we use a platform to simulate and test the strategy. The results give us insight about the performance of our approach in the real world. Based on the results we can modify our approach or the model to improve the results.

III. Related Work

Predicting the stock market has been the subject of many studies. Dealing with the wide variety of data sources to create a prediction tool is a daunting task. However, deep learning has shown great advantages in processing non-stationary data, and has been used more recently in the finance realm. We briefly review previous work that has used deep learning for stock prediction.

In [26], the authors proposed a method to build deep learning hierarchical decision models that can include complex features. To do so, a framework was set up to train the data, and then a four-step algorithm was used for model construction to build deep portfolios and create an automated process to select portfolios. This method was tested on the IBB Index, and showed that deep learning may have the potential to dramatically improve predictive performance in conventional applications.

In [33], the author analyzed the influence of news articles on stock prices. After downloading the headlines, the stock trend was correlated with headlines, concentrating on predicting whether a stock price rose or fell. Recurrent neural networks were used to map the function between sentiment values and the target price. The performed experiments tried to predict stock prices using information from both numerical analysis and textual analysis. Numerical analysis was performed using long-short term memory (LSTM), and resulted in a mean-squared error (MSE) of 0.00045. Textual analysis was then performed on the news headlines, and the author claimed 78% accuracy in predicting their influence on stock prices. When the results from textual analysis were augmented over the predictions from numerical analysis, MSE improved to 0.00037.

In [20] the authors proposed an investment strategy for creating portfolios based on predicted future fundamental indicators. Fundamental data, such as revenue, operating income and debt, were gathered, and computed features of the reported data were analyzed. Using deep learning, future fundamentals were forecast based on a trailing 5 year window. Quantitative analysis demonstrated a significant improvement in MSE over a naive strategy. On a simulation to assess future financial reports, applying earning yields (EBIT/EV) during a 12 month period achieved a 44% annual return.

In [17], the authors used deep learning to predict one-month-ahead stock returns in a cross-section of the Japanese stock market. The predictive stock returns used information from the past five points of time for 25 factors from the MSCI Japan Index. The performance of a long-short portfolio was compared with support vector regression and random forests. Several patterns of DNN with different numbers of layers were examined, and four patterns of DNN outperformed both other methodologies.

In [32], the researchers used an artificial neural network (ANN) to create a trading system by using technical indicators. A multilayer perceptron (a class of feedforward ANN) was used to predict buy-sell signals by analyzing time-series data, using data from Dow30 stocks for the period 1997-2017. Hold, buy and sell signals were generated based on peak and valley points. In comparison to buy and hold the model provided mixed results. It was suggested that the parameters should be individually tuned for each stock to improve performance.

In [25], the authors used deep learning for a time series prediction problem. Average monthly statistics for the S&P 500 split by industry are gathered from January 1990 until October 2015. LSTM networks were used to determine the directional movements. With daily returns of 0.46% and the Sharpe ratio (a measurement to understand the return of an investment compared to its risk) of 5.8 prior to transaction costs, they found LSTM networks outperformed other methods and showed that deep learning can be deployed in this domain.

In [31], the importance of different network design choices and hyperparameters were tested. It was found that some parameters (e.g. the last layer of the network) had a large impact on performance, while others (e.g. the number of LSTM layers) were of minor importance. They concluded that variational dropout was on all tasks superior to no-dropout or naive dropout. Adam [27] and Adam with Nesterov momentum (Nadam) [24] usually performed the best of those examined. During the experiments, they looked at one dimension for a certain hyperparameter. However it is to be noted that hyperparameters can influence each other.

In [18], the authors searched for accuracy using different statistical measures. After tuning the hyperparameters, they concluded that it may not be possible to predict the adjusted closing price solely based on the open, high, low, close and adjusted closing price. They ran the LSTM algorithm using backtesting data, and although the LSTM model had an accuracy of 80% on predicting the adjusted closing price, they suggested that it would be naive to conclude that their model could do an excellent prediction of the market.

IV. Methodology

In this section we present our approach, which concentrates on trend prediction. Later we optimize the hyperparameters of the model to improve the results. We will describe the metrics for evaluating the performance of the model and the details of the new approach.

A. Data

Our data is chosen from one of the companies from S&P 500. We selected a stock that had a shift in trends for the last 120 days, which is equivalent to our test data size. Data was downloaded from 3/13/2014 until 3/12/2019 for Apple Stock (AAPL) from "Yahoo! Finance" to conduct the experiments.

Our data is the stock price over approximately five years. The data consists of six columns:

1) Open: The price the stock started trading at when the exchange opened.
2) High: The highest price the stock has seen during the day.
3) Low: The lowest price the stock has seen during the day.
4) Close: The stock price at the last close of the market.
5) Volume: The volume is the number of shares that changed hands during a given day.
6) Adjusted Close: The adjusted closing price also factors for dividends.

The data in each column is converted into an array, and technical indicators are created based on these values.

B. Training the model

This process includes comparing the actual value with the value generated by randomly assigned weights. Backpropagation is used to train the network. During this process the weights of neurons are updated based on the previous epoch or iteration.

During training, the number of epochs is counted by multiplying the number of iterations by batch size. An epoch, which is useful for periodic evaluation, is one pass over the entire data [15].

1) Multivariate Time Series: By using LSTM networks it is possible to forecast data using multiple input variables. In a univariate time series, the forecast depends on one time dependant variable. Unlike univariate time series, multivariate time series can have multiple variables. Each of these variables depend on their past value and can be dependent on other values.

2) Features: To create the model we use the following features: opening price, high price, low price, closing price, adjusted closing price, volume, volatility, Williams %R and RSI; the last three of these are technical indicators that were created using the initial data. We will later examine the effect of these indicators on the performance of the model.

3) Feature Scaling: To standardize the range of features, we used scaling. We used min-max scaling, with a range of -1 to 1. By using feature scaling we normalize the range of values, using the formula $x' = \dfrac{x - min(x)}{max(x) - min(x)}$, where x is an original value and x' is the corresponding normalized value.

4) Stacked Long Short Term Memory: Recurrent layers can be stacked on top of each other. In a gated recurrent unit(GRU), the hidden state is passed from one layer to the other. This makes the GRU learn transformations [19]. GRU performances are generally on par with LSTMs [19]. Stacked LSTMs can be defined as multiple LSTM layers used in sequential order. The first layers of LSTM return their full output sequences, but the last one only returns the last step in its output sequence, thus dropping the temporal dimension [3]. In our model four layers of LSTM are used to improve performance. The overall architecture is shown in Fig. 2.

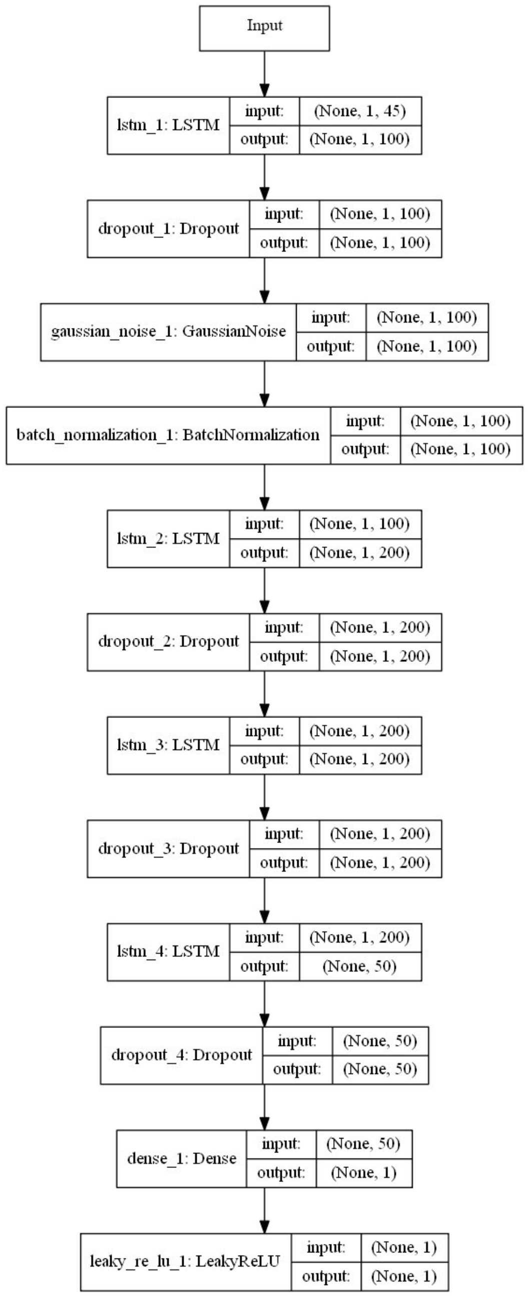

Fig. 2: Architecture of the model

5) Reducing Overfitting: Dropout prevents overfitting and the term refers to removing a hidden unit temporarily from the network. Dropout in our model is a configurable value that indicates the number cells to dropout during the process.

Gaussian Noise is added during training to regularize the layer. The amount of noise added is a configurable hyperparameter. Injected gradient noise causes improvement in different models [30].

6) Overall Architecture: The overall architecture of the network consists of four layers of LSTM networks. To create the input data, features are scaled and sliding windows are created. Then the data is reshaped for input of the LSTM network. The first layer consists of 100 units, with an initial dropout of 0.2; later this value will be optimized through hyperparameter optimization. It should also be noted that zero dropout was tested but not extensively because initial results indicated using it did not improve results. We add gaussian noise of 0.05 to the model and use batch normalization. The second, third and fourth layers each have 200 units with the same dropout value. The input format is $(batch\,size, times\,steps, input\,dimension)$ and the batch size (none) will be set later in training of the model.

C. Frameworks and Libraries

To train the model we used Keras framework [5]. Keras is a Python framework which provides neural networks API by running on top of TensorFlow, CNTK, or Theano [5]. Our framework will also use TensorFlow [10] on top of Keras. TensorFlow is a free software library focused on machine learning. It uses graph structures and each edge between nodes is a multidimensional tensor. Other Python libraries including Pandas [11], Numpy [2], and Scikit-learn [9] were also used to process the information. Backtrader [1] and Talos [13], discussed later, were used for backtesting and hyperparameter optimization.

D. Experiments

The stock we examine is AAPL, selected for the reasons described in Section IV-A. We will later examine the effect technical indicators have on the loss of our model and then optimize the hyperparameters through grid search and measure the performance of the model. To measure the performance of the model in the real world we will use backtesting. To further examine the functionality of the final model, we will test the approach on three other stocks.

1) Initial Experiments: Our initial goal was to predict the price of a stock at a certain point in time. After creating the model and optimizing it, we generated the buy and sell signals. After backtesting was performed, the results showed that the value of the portfolio decreased, even when the commission was set to 0. We tried to penalize the model for loss of value and add threshold but again the results were not satisfactory. One reason for the failure of the experiments was that the loss function was set for the price instead of the profit, so that a lower MSE did not result in higher profit. Another reason could be that the neural networks failed to find an accurate pattern in day to day price changes.

Our next experiments focused on predicting the price change, aiming to predict if the price change would be positive or negative each day. Again, after training the model, the results of the backtesting indicated that the initial portfolio decreased in value.

The third phase of the initial experiments focused on predicting the RSI. That resulted in just one buying signal.When the trading strategy was changed with closer overbought and oversold signals, it did not result in an increase in the portfolio value.

2) Final Experiments: Our later experiments focus on predicting the trend of the market. One good indicator for predicting the trends is the MACD histogram. As mentioned earlier, a trading strategy that uses the MACD histogram can be used to generate a buying signal when the MACD histogram moves from zero to a positive value and a sell signal when the price change crosses below zero.

The data was split into test, training and validation sets. Ten percent of the data, which equals approximately to 120 days, starting from 2018/09/10, is set as test data, and the rest is used for training. The validation will be 10% of the training data.

Before passing the data we must reshape our inputs. The input of the LSTM has three dimensions: number of samples, time samples and number of features. In the model, mean squared error (MSE) was used as the loss function and LeakyRelu was set as the activation function. Adam [27] was set initially as the optimization algorithm. Once we have a suitable model to predict a day ahead, we tune the hyperparameters and select the best set of configurations.

The number of epochs is set to 100 and batch size to 32. Because the optimization is an iterative process, it is necessary to go over the training set multiple times. The data is divided into smaller batches before being fed to the neural networks. We also reduce the learning rate when our metric (MSE) has stopped improving. This is done because it is noted in [4] that models often benefit from doing so. The ReduceLROnPlateau callback in Keras monitors a quantity and the learning rate is reduced if no

Parameter	Values
Dropout	From 0.1 to 0.5 in 5 steps
Optimizer	Adam, Nadam, SGD
Loss Function	Huber Loss, Mean Squared Error
Activation Function	ReLU, LeakyReLU

TABLE I: Parameter space for optimizing

improvement is seen for a defined number of epochs.

3) Hyperparameter Optimization: After creating a working model, we optimize hyperparameters. To do so we must choose which hyperparameters we want to optimize. Based on previous research, we know that a few parameters have a bigger impact on the results than the others. The parameters we focus on are shown in Table I. Based on previous research (including [31]), these values were selected to create a parameter space for grid search.

There are three different optimization strategies: grid search, random search and probabilistic reduction. We use grid search, which scans the data with a set of predefined hyperparameters. It builds a model on each parameter configuration and ultimately selects the model with best performance. The framework that use to optimize the hyperparameters is Talos [13]. Talos is an open source framework, and is used for hyperparameter optimization with Keras models. After an experiment is started, a scan object is created which is used in the main program. By using the parameter space, the framework yields the next permutation through multiple iterations until all permutations of the parameter space are processed.

4) Backtesting: To perform backtesting we used BackTrader [1]. We chose a simple trading strategy that goes long on prices. When the MACD Histogram moves from a negative value toward a positive value we go long on the stock. We also consider a threshold of 5% for crossing the zero-line, to account for mistakes. If the portfolio's value is higher than the initial amount and the buy and hold strategy, we can say the approach was successful.

V. Results

A. Calculating Returns

The focus of the experiment, as explained in Section IV-D2, was on predicting the MACD histogram and using that along with a trading strategy. When the MACD histogram crosses above zero we buy and sell when it crosses below zero. The mean squared error for our experiment without using technical indicators was 0.04057.

To calculate how much this model yields profit, we identify days that cross above or below zero

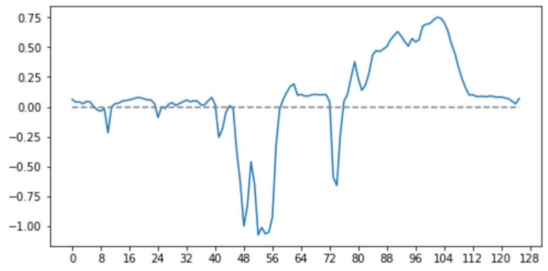

Fig. 3: Predicted MACD histogram with zero line

to generate buy and sell signals. From the data in Fig. 3, these are [6,11,23,26,40,43,44,58,72,75,125]. Since we are only considering to go long the buy signals are [11,26,43,58,75] and the sell signals are [23,40,44,72,125]. To remove sudden fluctuations in our data, we add a threshold of 5% for the zero-line in the MACD histogram, so that the prices have to cross above or below the threshold line. The resulting trading days are the buy and sell signals fed into Backtrader as input data.

Backtrader has a list of configurations that can be set. For testing purposes we are only interested in setting the commission value since it may have the highest impact on the final result. In the first experiment, the commission is set to zero and later the final value is calculated after optimizing the model with a commission to a flat rate of \$10, which is a typical price offered for online trading. After feeding the input data to Backtrader, the program starts simulating the trading. The size for each trade is set to 500. The cost of the investment is equal to the size of trade multiplied by the value of the equity on the first purchase. The return on investment (ROI) is calculated as $ROI = \left(\frac{Net\ profit}{Cost\ of\ investment} * 100\right)$.

The results of backtesting are shown in Fig. 4. Although the final portfolio shows a loss, when compared to the stock price, the values indicate the results can be optimized to create a profitable model.

B. The Effect of Technical Indicators

The goal of this experiment is to examine the use of technical indicators on the performance of our model. We will examine the use of three main technical indicators: RSI, Williams %R and Volatility. Table II shows the effect of these indicators on the loss value. As we can see, by using only RSI we can decrease the loss of the model to the lowest amount.

C. Hyperparameter Optimization Results

To further improve the results and also to examine which hyperparameters have better effects on the

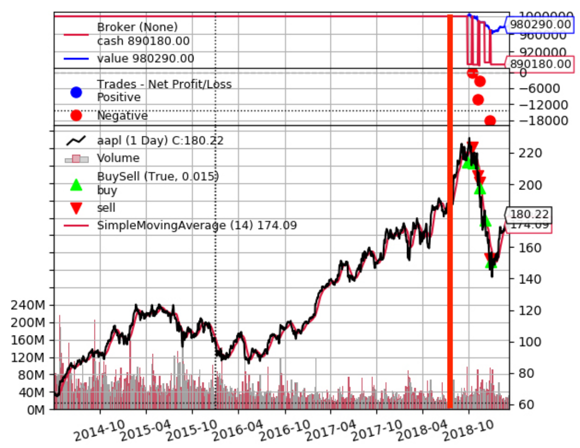

Fig. 4: Backtesting using Backtrader, with buy and sell signals for the test period (after red line).

Technical Indicator	MSE
Volatility	0.02699
William %R	0.03845
RSI	0.02053
All Indicators	0.02598
No Indicator	0.04057

TABLE II: The effect of technical indicators.

model for future research, hyperparameter optimization is performed, running the experiment for 60 different combinations based on the hyperparameter space to find the set of parameters with lowest MSE.

1) Analysis of results: By using LeakyReLU the results improve and MSE comes below 0.1 while by using the ReLU function MSE goes well above 0.2. To summarize the results, the top ten best results are selected and displayed in Table III. The best parameter space is the first row with the MSE of 0.00484. The model is evaluated with these parameters and the results are calculated to examine the performance.

D. Final Experiments

After selecting the best parameters from optimizing the hyperparameters and modifying our model, we run the experiments again to see how it increases the value of our portfolio. Fig. 5 shows the actual and predicted data.

Days [77,78,82,111] cross the zero line. Analyzing the predicted results to generate buy and sell signals we determine [77,82] for buying and [78,111] for selling. Our final portfolio value increased to

MSE	Dropout	Optimizer	Loss fn
0.004845492	0.1	Nadam	MSE
0.004993705	0.1	Nadam	Huberloss
0.005018013	0.18	Nadam	MSE
0.005358382	0.1	Adam	Huberloss
0.005375063	0.18	Nadam	MSE
0.005381555	0.1	Ndam	Huberloss
0.005391314	0.26	Adam	Huberloss
0.005471755	0.18	Aadam	MSE
0.005537316	0.26	Nadam	MSE
0.00564124	0.34	Nadam	MSE

TABLE III: Ten best results of hyperparameter optimization. In all of the ten best cases, Activation was LeakyRelu.

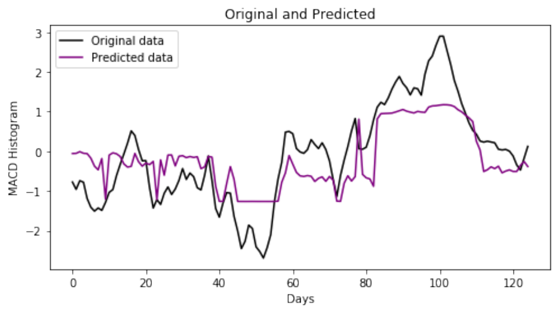

Fig. 5: Original and Predicted data

$1015075.00, which shows that optimizing the hyperparameters has a positive impact for this single testing set. Based on the beginning purchase and the end value, the return on investment for 128 days is equal to 6.1%. It should also be noted that the ROI with the buy and hold strategy for the same period of time is equal to -16.37%.

A summary of the results before and after optimization are shown in Table IV.

Expriment	MSE	ROI
Before Optimization	0.040576865	-18.04%
After Optimization	0.004845492	6.67%

TABLE IV: Comparison of the experiments

Company	Symbol	ROI	Buy & Hold
Apple	AAPL	6.61%	-16.37%
Microsoft	MSFT	25.99%	27.51%
Google	GOOG	5.769%	9.22%
Intel Corp	INTC	16.9%	2.34%

TABLE V: Comparison of ROI for different stocks

E. Testing Other Stocks

To further test our approach we selected three more stocks from the information technology sector of S&P 500. These stocks were selected because they are from the same sector but showed different performance over the time period examined.

We use the same process and select the hyper-parameters from our previous experiment. The final results are shown in Table V. For all of these results (including for the last experiment), the commission was set to \$10. The beginning balance is the amount of money spent to buy 500 shares. Our approach showed a positive ROI for all four stocks. In comparison to buy and hold, the performance of our approach was significantly better for both Apple and Intel, comparable for Microsoft, but worse for Google. During this same time period the overall performance for S&P 500 was +0.05%.

VI. Conclusions and Future Work

In this study, we used long-short term memory (LSTM) networks to predict the trend of markets and generate buy and sell signals to create a profitable model. In predicting the MACD histogram, promising results were seen. This is a suitable indicator to predict the trend of the market.

We chose Apple's stock and created an LSTM network to test our approach and later conducted the experiment for other stocks chosen from S&P 500. The network's architecture was improved by studying previous research and backtesting was used to evaluate the model's outcome in the real world.

We examined the effect of RSI, Williams %R and volatility on the loss of the model. It was shown that by using only RSI, the model's loss was reduced, which contributed to the performance of the model.

We also measured the effect of hyperparameter optimization with the use of grid search to identify the best set of hyperparameters, which is one of the main differences of our work in comparison to previous research using LSTMs. It was shown that choosing a different optimizer and activation function had a substantial effect on the loss of the model. By reducing the loss value it was possible to increase the return on investment from a negative value to 6.67%. This paves the way for future research with the focus of identifying stocks with higher returns and creating portfolios with multiple stocks.

Our results show that deep learning can be integrated with technical analysis to create a profitable portfolio by choosing the correct technical indicator. It should also be noted that the results of the portfolio can be further optimized by choosing more complicated trading strategies.

The achieved results lay the ground for further research. This could include examining the effect of different input data. The data can be gathered from other sectors and different stock markets. Other technical indicators can also be used to decrease the loss value. The research could be further expanded to modify the formulas for technical indicators and also test which indicators have a higher correlation with other factors such as volatility, etc. to categorize the prediction models.

Different neural network architectures may also achieve better results. Our work is consistent with previous research in that we used stacked LSTMs. Although a brief investigation was performed to evaluate other options, the stacked layers produced the best results. Further work could investigate this more fully, including evaluation of the optimal number of layers to use.

Another topic would be to integrate the methodology with other analyses such as sentiment analysis [34]. With the help of sentiment analysis, it may be possible to generate buy signals sooner and integrate that into the model. Another research direction is the use of fundamental indicators such as revenue. The combination of these methods along with the examination of the approach on a wider area of stocks can be a suitable choice for future research.

Acknowledgements

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Backtrader. https://www.backtrader.com/. Last accessed 26 June 2019.

[2] Fundamental package for scientific computing with python. https://numpy.org/. Last accessed 1 September 2019.

[3] Getting started with the keras sequential model. https://keras.io/getting-started/sequential-model-guide/. Last accessed 6 April 2019.

[4] Keras documentation. https://keras.io/callbacks/. Last accessed 20 May 2019.

[5] Keras: The python deep learning library. https://keras.io. Last accessed 10 September 2018.

[6] Macd. https://www.investopedia.com/terms/m/macd.asp. Last accessed 6 April 2019.

[7] Macd historgram. https://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:macd-histogram. Last accessed 6 April 2019.

[8] Macd historgram in technical analysis. https://commodity.com/technical-analysis/macd/. Last accessed 6 April 2019.

[9] Machine learning in python. https://github.com/scikit-learn/scikit-learn. Last accessed 1 September 2019.

[10] An open source machine learning framework. https://github.com/tensorflow/tensorflow. Last accessed 1 September 2019.

[11] Python data analysis library. https://pandas.pydata.org/. Last accessed 1 September 2019.

[12] Relative strength index - rsi. https://www.investopedia.com/terms/r/rsi.asp. Last accessed 10 September 2018.

[13] Talos documentation. https://autonomio.github.io/docs_talos/#introduction. Last accessed 25 May 2019.

[14] Trading strategy. https://www.investopedia.com/terms/t/trading-strategy.asp. Last accessed 02 May 2019.

[15] What does sample, batch, epoch mean? https://keras.io/getting-started/faq/#what-does-sample-batch-epoch-mean. Last accessed 6 April 2019.

[16] Williams %r. https://www.investopedia.com/terms/w/williamsr.asp. Last accessed 10 September 2018.

[17] M. Abe and H. Nakayama. Deep learning for forecasting stock returns in the cross-section. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 273–284. Springer, 2018.

[18] W. Ahmed and M. Bahador. The accuracy of the lstm model for predicting the s&p 500 index and the difference between prediction and backtesting, 2018.

[19] C. Ahuja and L. Morency. Lattice recurrent unit: Improving convergence and statistical efficiency for sequence modeling. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[20] J. Alberg and Z. C. Lipton. Improving factor-based quantitative investing by forecasting company fundamentals. *stat*, 1050:13, 2017.

[21] E. Chong, C. Han, and F. C. Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Elsevier*, 2017.

[22] T. Chong and W. Ng. Technical analysis and the london stock exchange: testing the macd and rsi rules using the ft30. *Applied Economics Letters*, 15(14):1111–1114, 2008.

[23] T. Chong, W. Ng, and V. Liew. Revisiting the performance of macd and rsi oscillators. *Journal of risk and financial management*, 7(1):1–12, 2014.

[24] T. Dozat. Incorporating nesterov momentum into adam. 2016.

[25] T. Fischer and C. Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, October 2018.

[26] J. B. Heaton, N. G. Polson, and J. H. Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.

[27] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[28] C. D. Kirkpatrick and J. R. Dahlquist. *Technical Analysis: The Complete Resource for Financial Market Technicians. Financial Times Press*. Wiley, 2006.

[29] M. Klassen. Investigation of some technical indexes in stock forecasting using neural networks. In *WEC (5)*, pages 75–79. Citeseer, 2005.

[30] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens. Adding gradient noise improves learning for very deep networks. *stat*, 1050:21, 2015.

[31] N. Reimers and I. Gurevych. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *CoRR*, abs/1707.06799, 2017.

[32] O. B. Sezer, A. M. Ozbayoglu, and E. Dogdu. An artificial neural network-based stock trading system using technical analysis and big data framework. In *Proceedings of the SouthEast Conference*, pages 223–226. ACM, 2017.

[33] A. Tipirisetty. Stock price prediction using deep learning. page 60, 2018.

[34] Y. Zhao, B. Qin, T. Liu, et al. Sentiment analysis. *Journal of Software*, 21(8):1834–1848, 2010.