Fall 2021

Predicting Stocks with LSTM-based DRNN and GAN

Duy Ngo
San Jose State University

Predicting Stocks with LSTM-based DRNN and GAN

A Project Report

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Class

CS298

By

Duy Ngo

December 2021

The Designated Project Committee Approves the Project Titled

Predicting Stocks with LSTM-based DRNN and GAN

by Duy Ngo

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2021

Dr. Ching-seh Wu, Department of Computer Science

Dr. Thomas Austin, Department of Computer Science

Dr. Wendy Lee, Department of Computer Science

ACKNOWLEDGEMENT

Within the last year of this academic expedition, many challenges have been encountered, but I have gained a wealth of knowledge as a result. This bittersweet feeling will forever be ingrained within me as I continue onward my own path. I will be able to reminisce upon these events and remember the individuals that have given me their support through the process. I am extremely grateful to my advisor Dr. Mike Wu for his guidance and influence during the project, as well as in his classes throughout my master's degree.

I would also like to extend my gratitude towards Dr. Wendy Lee and Dr. Thomas Austin for accepting to be a part of my project committee. Their review and feedback has greatly benefitted me in making my project successful.

My eternal gratitude goes to these great faculty members and also my parents, siblings, and my fiancée, Julie. Everyones' support has been instrumental to the success of this journey.

Abstract

Trading equities can be very lucrative for some and a gamble for others. Professional traders and retail traders are constantly amassing information to be a step ahead of the market to profit off the value of stocks on the market. Some of the tools in their arsenal include different types of calculations based on a variety of data collected on a stock. Technical analysis is a technique for traders to analyze the data of equities presented on charts. Often, the way the price changes over time can be used as an indicator for traders to predict how future prices will move. This practice can be done due to the investing psychology of the masses that indicates a certain sentiment towards a stock. As artificial intelligence and machine learning have developed, researchers have also studied how to utilize this technology to analyze the data and forecast how prices will change.

Most recently, neural networks in deep learning approaches have been shown to outperform traditional machine learning methods. From the deep learning approaches, utilizing Long Short-term Memory (LSTM), a recurrent neural network (RNN) architecture, has been able to use time-series stock data to forecast future stock prices. This project proposes to extend this prediction process by layering another deep learning approach called Generative Adversarial Network (GAN) which pairs two networks to compete and improve each other . The approach was used to train an LSTM network to further improve the performance. The performance of the stacked LSTM-GAN model was compared to the stacked LSTM that had not been improved by the GAN. Results showed that the proposed model was able to outperform the no GAN extension by 22-25%, based on the RMSE and MAE, on the test set of the data used in training. Additionally, on a random set of stock data, the model performed 4-5% better.

Index terms**:** Stock markets**,** generative adversarial networks, long short term memory, deep learning.

TABLE OF CONTENTS

List Of Figures

LIST OF TABLES

I. INTRODUCTION

Stocks are securities that show partial ownership of a company. The value of shares of companies fluctuate constantly as stock traders around the world buy and sell company shares on stock exchanges. The push and pull of supply and demand, as stocks are being traded, is what drives the fluctuation of prices. Traders research companies and analyze trends in price movement to find the optimal time and situation to buy shares of a company at a low price to later sell at a higher price for profit. Furthermore, emotions may get in the way and individual traders start to doubt their technical analyses which leads to lost capital. For example, when actively monitoring price movements, individuals may see consecutive large drops in price and sell their holdings to cut losses. This action creates a chain reaction that causes more drops in the price. Emotions affect decision making and large social media platforms, like Twitter, can be analyzed to recognize these psychology based stock movements [1]. As studies have shown, when groups of individuals all behave the same way to price movement, indicators in a trend can be extracted [2].

With thousands of companies that have been traded back and forth on the stock exchange for years, the large historical data of companies' stock movements can be used to create a model to predict future activity of stocks. Data is available at resources like Yahoo! Finance [3] or MarketWatch [4] contain years of trading data for every publicly traded company or Kaggle [5] which have user uploaded datasets. A prediction model can be used to help less experienced traders check their stock analyses against a predicted model for confirmation or further optimize profits for experienced day traders. Programs that utilize an accurate prediction model can then be used to trade for individuals and cut out the emotions involved.

As such, trends can be predicted based on previous data that led to a similar trend [6]. Various machine learning models have been applied to stock prediction systems from basic Decision Trees and K-Nearest Neighbors (KNN) to many other models including deep learning models like Recurrent Neural Networks (RNN) and Long short-term Memory (LSTM) [7, 8]. Since LSTM performs well with time series data, which stock analysis requires, this approach has been more popular in recent years. Another improvement is to use a Generative Adversarial Network (GAN) to train a model to predict price movements better by training it to detect false trends [9]. Aside from the stock prices themselves, other data points are part of the data used for prediction. Some of which are the volume of the specific stocks being traded, moving averages, and momentum calculations [10]. All of this information on stocks can be utilized and applied to create a prediction system that recognizes trends and supports traders in optimizing their profits.

A. *Research Objective*

The research aims at analyzing a Generative Adversarial Network (GAN) model which uses a deep recurrent neural network (DRNN), that is LSTM-based, for its generator and its effectiveness for stock prediction. Predictions will be made with an LSTM model then the system will employ GAN to optimize the LSTM model further. The model will be compared to current techniques. The archived stock data will be obtained from MarketWatch and preprocessed.

II. BACKGROUND AND RELATED WORK

Research and studies to analyze and predict the stock market has been continually developed where new contributions are constantly being made. Most studies are focused on processing the stock data with values itself and not stock charts [11]. A recent survey by Weiwei Jiang [9] was created for the purpose of acting as a compendium for new researchers to catch up with recent studies and developments. Jiang's research covers the important aspects of stock prediction techniques. The survey covers the important steps of stock market prediction which starts with data acquisition and preparation to deep learning systems used to train, and predict based on the data. Additionally, hundreds of recent papers are cited and categorized for new researchers to not only learn further about different areas and implementations of the study but to also recognize more popular and recent techniques and approaches within the past years, since 2017. This section explores different approaches in machine learning and deep learning that researchers have studied. Some referenced papers may also have been featured in Jiang's survey.

A. *Traditional Approaches*

Traditionally, linear models were used for forecasting time-series data, like stocks. Although research gravitated towards implementing machine and deep learning approaches, the Autoregressive Integrated Moving Average Model (ARIMA) is still an extremely popular linear model that is often used as baseline comparisons to other time-series forecasting approaches [26]. As the name suggests, ARIMA combines two linear methods: autoregressive and moving averages. Autoregression models are ones that input previous time steps into a specific regression equation for forecasting, and moving average is the average of a subset of time steps [27] Before transitioning to modern approaches ARIMA was widely used both in financial and economic time-series predictions. Researchers in [26] implemented ARIMA and compared it to

LSTM in stock prediction where ARIMA had an average RMSE of 511.481 whereas LSTM had an RMSE of 64.213 indicating that LSTM outperforms ARIMA.

B. *Machine Learning*

Various machine learning techniques do not have the limitations that classical linear methods have with non-linear variables. With how much more random the stock market can behave, machine learning techniques can be a better alternative. Often used machine learning models include Logistic regression (Logit), random forest regression (RF), and k-nearest neighbor (KNN) [9].

Logistic regression is similar to a linear model, but generalized [9]. It is used to forecast if the price will go in a certain direction (up or down), and does not give exact predicted values [28]. The process is similar to linear regression where there is data normalization, regression coefficients calculation, matrix transformation, coefficients assignment, then regression equation (Figure 1) applied, except that there is a sigmoid equation (Figure 2) instead of a regression equation to calculate the forecast. Although with decent results, researchers in [29] evaluated a logistic regression implementation that showed accuracy is dependent on the stock selection.

$$Z = a + b\,X + e$$

Z: the predicted value for the dependent variable, response variable
X: a random variable, predictor variable
a: the value of Z when X = 0, intercept
b: rate of change in Z with a unit change in X, regression line slope
e: a random error

Figure 1. Regression Equation [29]

$$W = \frac{1}{1 + e^{-(a+bx)}}$$

W: predicted value for input x
a: value of W when x = 0, intercept
b: rate of change in W with a unit change in X, regression line slope
x: a random variable, predictor variable

Figure 2. Sigmoid Equation [29]

Random forest regression is a technique that contains multiple random trees that utilize different tree structured classifiers. Each trees' outputs are combined and a majority voting procedure is done to decide the majority output as the final prediction. Majority voting is done because the idea is that if many unrelated models output the same prediction, then that prediction

has a higher chance of being more accurate than a minority class [9]. With a random forest implementation with 100 trees, it was shown that the larger the stock's time range, the more accurate the forecasts are [31]. For example, the accuracy of Facebook in a 3 day trading window is 67.59% compared to a 90 day trading window that has an accuracy of 94.76% in predicting the next direction of the stock [31].

Another popular approach is k-Nearest Neighbor which classifies unknown data into labels that have the shortest distance to the known datas' labels. The distance measures that may be used can be the Euclidean distance, Minkowski distance, or Manhattan distances [13]. Researchers [30] implemented the KNN and evaluated the approach with five companies on the Indian stock exchange where KNN resulted in a 63% accuracy using the coefficient of determination, R^2, metric whereas, in comparison, their linear regression approach had a 98% accuracy.

Additionally, AdaBoost is an ensemble method that creates a strong classifier from a number of weak classifiers [20]. The AdaBoost algorithm can be used to optimize a classifier. Researchers in [20] applied the AdaBooster optimization to several popular approaches including long-short term memory (Section C. 1.) by getting the predictions from an ensemble of each approach and passing them through AdaBoost. The AdaBoost-LSTM implementation had a mean-absolute percentage error (MAPE) of 0.80267% on the S*P 500. This was much better than the single LSTM forecast with a MAPE of 1.9168%. AdaBoost was also applied to other combinations of datasets and deep learning and machine learning methods that yielded similar results [20]. This showed that this method could be applied to effectively improve prediction models.

Although these mentioned machine learning approaches are still researched, many of them are compared against deep learning techniques as a baseline due to deep learning techniques' better performance [9]. [29] compared the logistic regression approach against a deep learning approach and found that it outperformed their Logit implementation. Some of these classification methods also do not forecast exact values, but predict the general trend of the price movement.

C. *Deep Learning*

The StockPred framework proposed by M .Sharaf, E.ED. Hemdan, A. El-Sayed and N. A. El-Bahnasawy shown in Figure 3 tests various machine learning and deep learning models with different scenarios [13]. The framework acted as a flowchart for how to process their raw data and summarized their approach for picking and comparing the better models.The scenarios conducted were different time ranges of the stock data as well as varying the epoch size and the batch size for the deep learning models. Although the forecast range can be adjusted to improve accuracy in machine learning models, it was not worth it.

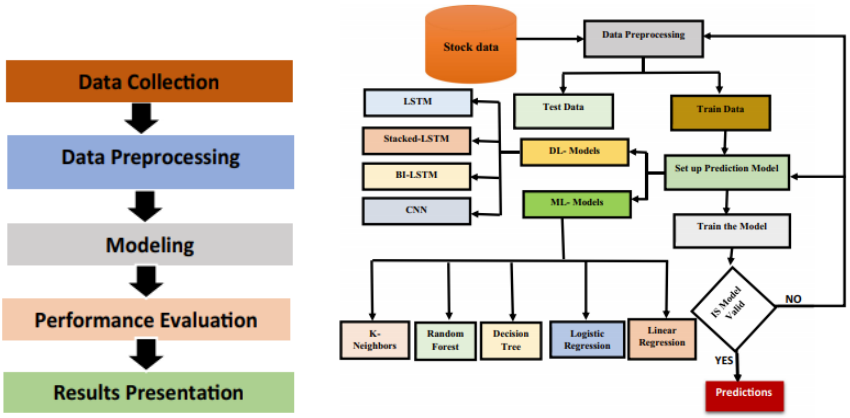


Figure 3. StockPred Framework [13]

Furthermore, they concluded that balancing epoch and batch size hyperparameters is needed to improve performance [13]. These hyperparameters would need to be uniquely tuned

for the specific deep learning model and dataset used. The StockPred framework showed that deep learning models consistently outperformed machine learning models [13]. Deep learning techniques, specifically LSTM approaches, have been found to currently be the best for time series prediction problems [12].

1. Long-Short Term Memory (LSTM)

Between the deep learning models, researchers have leaned towards using Long-Short Term Memory (LSTM). LSTM is a type of recurrent neural network that has special units within them that can process longer time-series data [14]. LSTM has a memorizing mechanism that is useful for time series forecasting, shown in Figure 4 [15]. The sigmoid functions have binary values that determine whether data can pass through or not. The tanh activation functions add weights and keep the values between -1 and 1. In Mehtab and Sen's research, *A Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models*, they found that LSTM out performs the machine learning models [12]. LSTM is a recurrent neural network that allows feedback loops to communicate data from nodes in forward and backwards layers, and it also has a forget gate that helps solve the vanishing gradient problem [12].

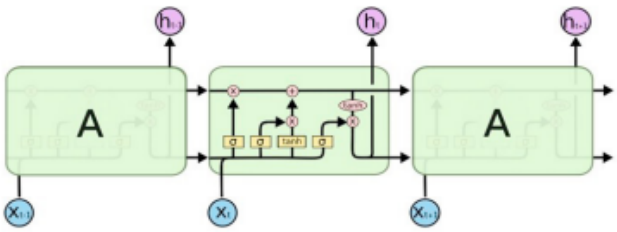


Figure 4. LSTM Structure [15]

Since time series data rely on each other, communicating between forward and backward layers is important. Other researchers further attempt to optimize LSTM by testing various

parameters [16]. Moghar and Hamiche's studies looked at how the chosen data set and preparation affects the prediction model, specifically LSTM [17].

	MV	MARS	DT	BAG	BOOST	RF	ANN	SVM	LSTM
Correlation	**0.99**	**0.99**	0.97	0.97	**0.99**	**0.99**	**0.99**	0.93	**1.00**
RMSE/Mean	13.32	**12.41**	35.34	33.43	23.40	16.26	14.99	53.88	**7.94**
Mismatched Cases	18.67	1.21	13.42	2.95	0.81	**0.00**	1.07	0.27	**0.00**

Figure 5. LSTM vs. Other Models [17]

The study implemented a multilayered LSTM that tested different epochs along with different ranges in the dataset. With Nike and Google datasets, Moghar and Hamiche found that with a longer time period, data becomes less accurate. This may be attributed to different events that are not part of the regular pattern of the dataset. For example, stock splits are not accounted for. Also periods like earnings, where volatility in price is higher can affect training. They found that reasonably less data combined with more epochs was optimal for model training [17]. Figure 5 shows the variety of models that Moghar and Hamiche tested with their Nike and Google datasets, and LSTM outperforms the opposition.

Another view of the LSTM structure in Figure 6 shows that an LSTM unit has three main gates. The input, output, and forget gates control the sequential data being passed through them. The forget gate allows for non-conforming information to be forgotten. These input and output gates can be linked to more units to form a large system of layered LSTMs. Deep recurrent neural network LSTMs, or multi-layer LSTM, implementations further improved accuracy [18]. The LSTM unit is able to keep state as well as managing what information to forget as the data passes through the units making it very useful in processing time-series data like stock prices.

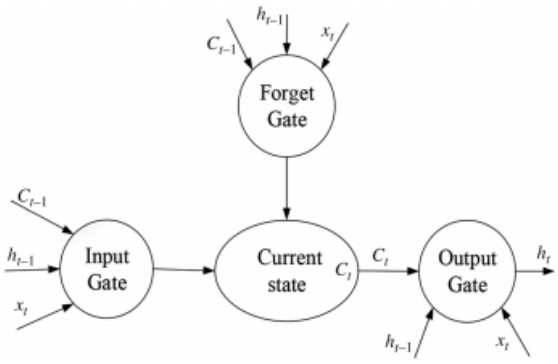


Figure 6. LSTM Unit Structure [19]

In *Study On the Prediction Of Stock Price Based On the Associated Network Model Of LSTM* [19], as part of their study, Ding and Qin implemented a DRNN that layered LSTM units with dropout layers. Figure 7 is the implemented DRNN LSTM structure that Ding and Qin used. There are two LSTM layers and two dropout layers after each LSTM later.

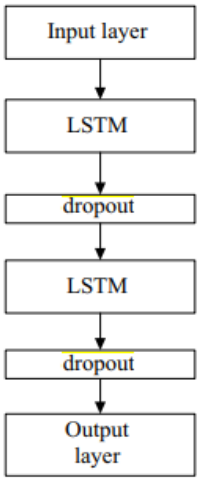


Figure 7. DRNN LSTM Structure [19]

Dropout layers randomly delete some units to solve overfitting [18]. Figure 8 shows a visualization of dropouts in a neural network, some units are dropped out at certain stages. The red-dotted nodes are dropped, and consequently, there are less connected nodes which helps prevent the model from correlating too closely to the specific dataset being used to train it

(overfitting). For example, if the Google stock data was used to train the proposed model without drop out. The result of the model may be too accurate in predicting Google stock data, and forecasting another company's stock price may be wildly inaccurate because the price movement is not similar to Google's, the data used for training. Dropouts are necessary to keep the model more general.

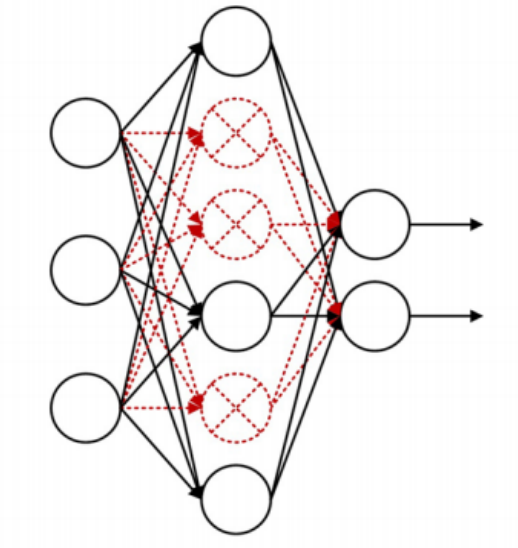


Figure 8. Dropout Visualization [19]

Ding and Qin created three connected DRNN LSTM structures that connected to form an associated network that was used to forecast three different price values for a given day. They trained and tested an individual DRNN LSTM and compared it to a single LSTM model. Figure 7 shows the average accuracy of their study. The DRNN implementation has a higher average accuracy than just the baseline LSTM implementation. They also tested the training times of the baseline LSTM, a DRNN, and their novel associated net. The LSTM was the fastest as it was just one unit compared to the stacked units in the DRNN, and expectedly, the associated network which was three layers of DRNN was the slowest.

Model	LSTM	DRNN	Associated Net		
			Open price	Lowest price	Highest price
Average accuracy	0.787925	0.973754	0.971999	0.956185	0.974434

Figure 9. LSTM vs DRNN LSTM vs Associated Net

Their open price accuracy is similar to their DRNN accuracy due to the open price being the first layer of the associated net, and is essentially just a DRNN (Figure 9). With high accuracy, the researchers then use the output of the subsequent DRNN to forecast the next price target. The open price prediction influences the forecast of the lowest price, and finally, both forecasts are passed onto the third layer for the last prediction. The associated network results also show that connected DRNNs have similar accuracies as just the single DRNN model.

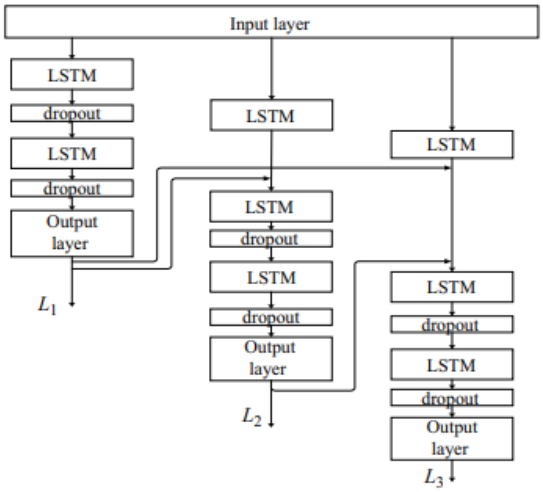


Figure 10. Associated Network DRNN LSTM [19]

The researchers showed that a multi-layered LSTM has improved accuracy while having more layers of stacked LSTMs maintain the same accuracy and was able to be utilized for multiple predictions at once.

From the various research, when implementing an LSTM, the data preparation for training is important. Additionally, the model needs to be correctly balanced and tuned in

regards to the epoch and batch size hyperparameters to optimize the accuracy and performance of the forecasting model [12, 16]. An LSTM implementation will be the main prediction model in the proposed approach for stock forecasting. Research has shown that, currently, LSTM implementations reign in accuracy over other machine learning models.

2. Generative Adversarial Network (GAN)

Also recently, generative adversarial networks (GAN) have been implemented to further improve accuracy. GAN is a framework that has two parts: the generator and discriminator [21]. The generator would be the system that generates predictions. On the other hand, a discriminator's job would be to try to discern whether the value(s) given to it was artificially generated or not. Figure 11 shows the architecture of a GAN system [22]. Real data is separately passed through to the generator and discriminator. The generator passes on its prediction along with the previous real stock data to the discriminator. If the discriminator is able to correctly determine that the predicted value was artificially generated, the GAN system would then fine tune the generator and discriminator to improve both their accuracies [23]. The cycle is then repeated for multiple iterations until both system's are sufficiently optimized.

In [22], the researchers implemented a GAN system with LSTM as the generator to make forecasts for the stock market. On the other hand, the discriminator was created with a Multilayer Perceptron (MLP) [24]. The LSTM and MLP networks will essentially train and improve each other over time.

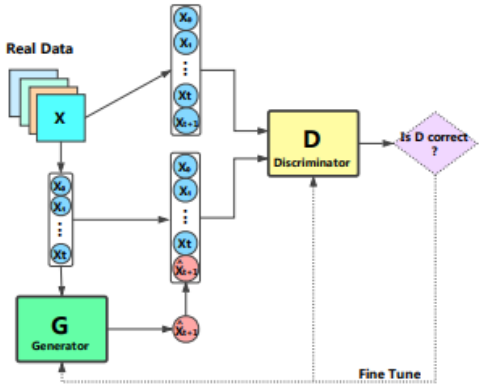


Figure 11. The GAN Architecture [22]

The training stops when an ideal situation where the discriminator cannot tell the difference between the two types of data occurs. Their system used the following features: high price, low price, open price, close price, trade volume, turnover rate, and 5 day moving averages of the stock prices. The predicted data was then concatenated onto the input data and passed to the discriminator that was implemented with MLP and three hidden layers (72, 100, and 10 neurons) with leaky ReLu activation functions [22]. The input data is then either classified as 0 or 1 for fake or real, respectively.

The researchers evaluated the model on stock data from various indexes as well as individual company stocks ranging from the Standard & Poor's 500 (S&P 500 Index), Shanghai Composite Index, International Business Machine (IBM), Microsoft (MSFT), and Ping An Insurance Company of China (PAICC). Their data was split into 90%-95%/5%-10% training and testing. The results of the study, in Figure 12, show LSTM was already an improvement to a baseline artificial neural network (ANN) and support vector regression (SVR) models. On top of that, GAN further improved LSTM, based on the evaluation metrics, where the error metrics from GAN are less than LSTM.

Table 2. The average evaluation on five stock data sets.

Method	MAE	RMSE	MAPE	AR
Our GAN	**3.0401**	**4.1026**	**0.0137**	**0.7554**
LSTM	4.1228	5.4131	0.0145	0.6859
ANN	7.3029	9.1757	0.0808	0.5249
SVR	4.9285	8.2261	0.0452	0.7266

Figure 12. GAN Study Results [22]

The discussed related works in recent stock market prediction approaches act as support and inspiration to this current study. This study aims to improve on these techniques by utilizing an optimized DRNN LSTM based on studies from [8, 9, 10, 12] to implement as a generator in the GAN architecture to further train and improve the DRNN LSTM accuracy with an opposing discriminator, as mentioned in [22]. Within the domain of LSTM research for stock market prediction, only a few have employed GAN to experiment with results in forecasting [2, 25]

III. DATASET AND TECHNOLOGIES

Figure 13 is the general overview of how the proposed GAN and LSTM-RNN will be structured. Before implementing and experimenting with the proposed model, preparations must be made to ensure the prediction models can be developed with the correct tools, and the right dataset is being used for the intended purposes.

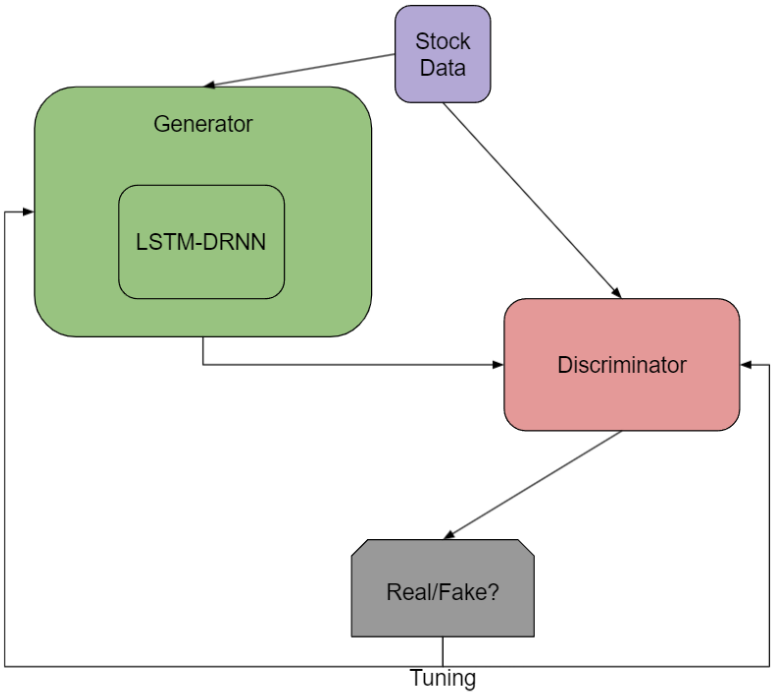


Figure 13. GAN LSTM-RNN

A. *Dataset*

As in Figure 13 and Weiwei's research shows, the first step to any prediction model or deep learning model needs to consider the data being used [9]. In the case of stock data, they are available on many stock related websites. Websites like MarketWatch [4] archive years of market data, and make them readily available to users. Aside from the data, the data collected from these resources usually include 5 main features. For every day, there is an open price, close price, high, low, and volume, shown in Figure 14.

```
Date,Open,High,Low,Close,Volume
04/22/2021,"122.54","124.42","120.37","121.41","7,751,506"
04/21/2021,"117.78","121.82","116.54","121.82","6,537,532"
04/20/2021,"120.13","121.89","117.23","119.00","9,825,747"
04/19/2021,"122.57","123.70","118.94","120.44","11,418,340"
04/16/2021,"126.51","126.62","123.41","124.38","6,910,999"
04/15/2021,"126.84","127.42","125.32","126.62","7,953,875"
04/14/2021,"128.30","130.28","124.51","124.87","9,959,208"
04/13/2021,"123.28","127.71","123.28","127.56","12,014,850"
04/12/2021,"122.69","122.90","120.23","122.47","5,261,325"
04/09/2021,"123.01","123.42","121.94","123.26","5,564,114"
04/08/2021,"122.80","124.41","122.63","124.14","6,610,922"
```

Figure 14. Stock Data from MarketWatch

The open and close prices are respectively the price when the stock market opens on that specific day and the price of the stock when the market closes that same day. Since a stock's price movement changes throughout the day, high and low prices respectively record what the high and low prices of the day are. Lastly, volume value represents the number the stock transacted throughout the day. Figure 15 is a diagram of the candlesticks that represent the price movement within a certain specified timeframe. In the figure, the high and low prices are the wicks of the candle and the body of the candle's open and close prices. Hollow candles, like the left candle in Figure 15, show a positive price movement in the time frame since the stock opened at a lower price and closed at a higher price. The opposite is true for filled candles, like the left, where there is a negative movement from a sell off, and the stock closes at a lower price.

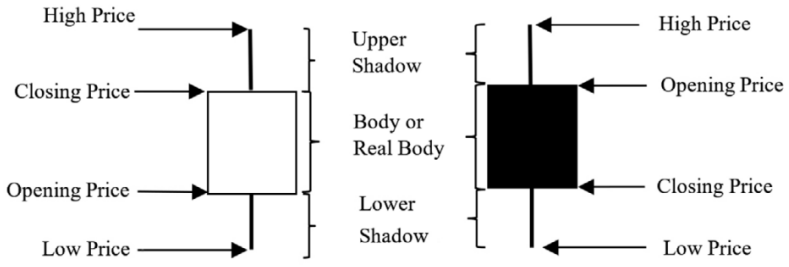


Figure 15. Candlesticks [33]

Collected data features and values are pretty straight forward, but not all may be used in training the prediction model. Initially, the datasets are scanned and checked for format consistency, like in Figure 14. For example, a dataset downloaded from one resource orders

earlier dates first while another source sorted most current dates first. The pandas library in Python will be used to quickly preprocess and modify needed data before passing it on to be used in training.

Other factors that need to be preprocessed include any stock splits that happen anytime throughout the stock data's timeline. Since historical data is used to predict future prices, stock splits, which change the stock prices, will skew the data and training will not be accurate. The approach does not yet have a way to deal with stock splits so it must be manually accounted for.

B. *Technologies and Libraries*

The main technologies utilized are the Python programming language and Google Research's Colaboratory (Colab) service. Python is also a popular language used by the artificial intelligence and data science community so there are many libraries for such use cases. The libraries that are used include: numpy, pandas, matplotlib, sklearn, tensorflow, and more specifically keras.

Numpy and pandas are imported and used for initializing and processing the data and making any necessary manipulations. Although preprocessing is done before reaching the program, further data manipulation will be needed which the mentioned libraries can handle. The matplotlib library is used for analysis purposes throughout the project to visualize results. Another analysis library was sklearn, which is used for error metrics to compare approaches. Sklearn can also be used to normalize the stock data values, and then reverted back to get original prices. The deep learning models were imported from tensorflow's keras library. With the library, a neural network can be set up with only a few lines of code to specify the layers within the network. This allows fast configuration of specific models, like LSTM, which can then be trained with input data to build the model up. Both numpy and tensorflow libraries use a

random seed for certain functions, and to ensure the same result the seed is hardcoded for both numpy and tensorflow.

1. Google Research Colaboratory

Originally, the local machine was used in the initial phase, but then Google Colab was utilized to advance the project further. Google Colab is a service that allows individuals to essentially borrow and use resources over the cloud. In this case, data processing and training machine learning models was the use case for using Google Colab. Colab allows programmers/researchers to utilize stronger hardware for faster processing over the cloud that their local machines may not otherwise be able to process as quickly, and it is a great tool for machine learning. Since machine learning and data intensive processes are often done with this tool, the Google Colab service instances also come initialized with all of the mentioned libraries pre-downloaded so the code only needs to import the specified libraries. Working with a notebook on Google Colab also allows parallelism between working on multiple machines like switching from desktop to laptop without having to set up the environment for both machines.

IV. APPROACH AND IMPLEMENTATION

The proposed objective takes advantage of two artificial intelligence techniques. A deep RNN that uses LSTM has been found to be highly accurate. Furthermore, a generative adversarial network will be able to further improve the accuracy by tuning the LSTM optimally over time as it trains.

A. *Data Preparation*

After raw, cleaned data is loaded, the data needs to be prepared further for training the model. The section will discuss the data preparation and approach for training. The data set being used to train the model is from Starbucks (SBUX) with 1260 days worth of entries, which is about 3.5 years.

Studies like [32] and [33] have shown that consecutive price action of a stock can be used to extrapolate future price movements.

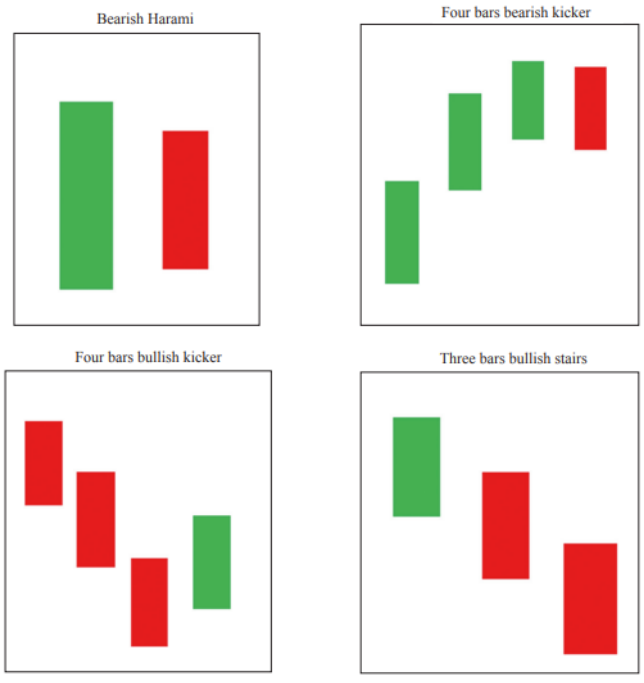


Figure 16. Example of Candlestick Patterns

Figure 16 has a few examples of candlestick patterns that are used to forecast how the next prices will move. Bullish patterns show that the sentiment towards the stock is positive and prices may move higher. On the other hand, bearish patterns are the opposite and investors should expect prices to drop further. The named patterns in Figure 16 also show that there are no set amount of candlesticks used for predicting prices. The bearish harami pattern only uses two candlesticks while the bullish stairs use three followed by the bullish and bearish kickers using four candles.

The data set up for the following LSTM and GAN implementations simplifies the candlestick diagram, and would only be using the closing prices. Furthermore, the experiment uses 5 consecutive prices instead of the 2 to 4 candlesticks used in Figure 16. Since 2 to 4 candlesticks are commonly used in analysis, there may possibly be patterns that can be extracted from 5 candlesticks. With a small set of prices to be learned, it may also prevent overfitting with the dataset used to train the model. For example, using 20 closing prices to predict the stock may cause the prediction model to only accurately predict one set of data and other stock data passed in for forecasting.

In the dataset used, data is accessed and the closing prices extracted during runtime with the pandas library.

X	Y
[1.09785399	0.6921470965254597
0.99916853	
0.87489795	
0.85845037	
0.87489795]	

Figure 17. Sample Scaled Data

For the input data, 5 close prices are stored in an array while the 6th closing price is stored separately as the label. The previous 5 closing prices will be used to predict the 6th price in the timestep. Each set of 5 closing price values are stored in a numpy array for training the model. Furthermore, the train and test split of the data is 80% train and 20% test. As shown in Figure 17, the data was also scaled before training. Since different stocks have different prices, one may trade in a $100-$200 range while another could be $5-$10, scaling data allows the model to learn within a normalized range. When ready, different datasets can then be scaled to the same range and passed in for prediction.

B. *Long Short Term Memory Implementation*

The LSTM model was imported from Keras within the Tensorflow library. The implementation uses two layers of LSTM that are stacked on top of each other and one dropout layer. Each LSTM layer utilizes 5 LSTM units before outputting to the next layer. The dropout layer is used to control overfitting too much of the data used to train the model.

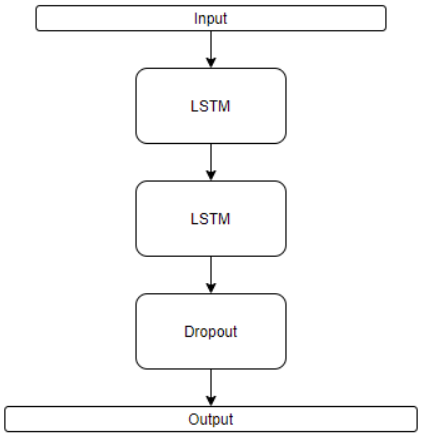


Figure 18. Stacked LSTM

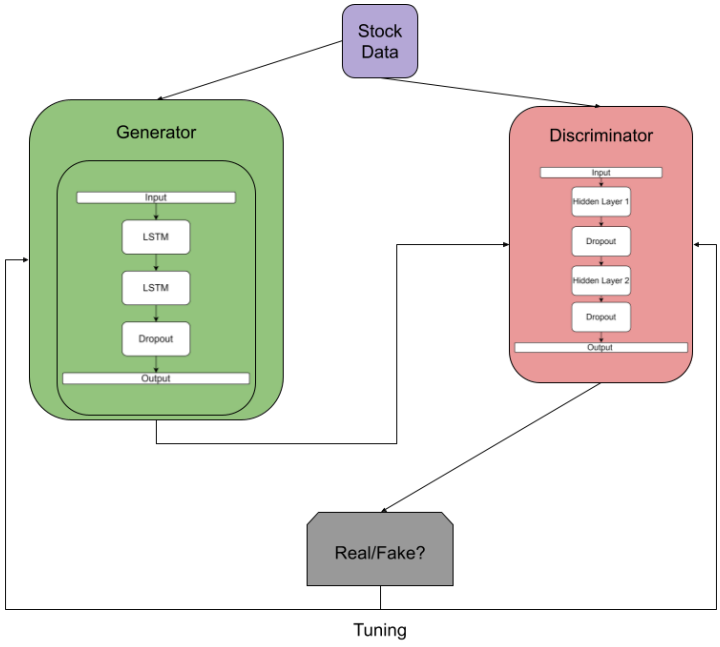
The standalone LSTM RNN model is shown in Figure 18. With Keras, a variable can simply be set for the model and specific layers added using the Keras API to match the specifications of Figure 18. Dropout layers are also added by simply adding the dropout layer

from the Keras API. The discussed hyper parameters are also set at this time. The model is then compiled with the Adam optimizer, a learning rate of 0.0001, and mean squared error as the loss function. The hyperparameters were chosen through trial and error. The model is then trained with 80% training data.

The rest of the data is reserved for testing and validation. 400 epochs were used to train the standalone stacked LSTM model for testing, but varying epoch values were tested when the stacked LSTM was implemented within the GAN system. The LSTM will take in 5 consecutive closing prices and learn to predict the 6th closing price of the stock. A small drop out layer will be used after the second LSTM to prevent overfitting of the price action.

C. *Generative Adversarial Search Implementation*

The basic GAN architecture can be seen from Figure 11. Similarly, the GAN implementation follows this architecture, but uses the previously discussed stacked LSTM structure, with 2 stacked LSTM layers with 5 units each as the generator. On the other hand, the discriminator is a simple multilayer perceptron (MLP) that uses 4 layers using the ReLU activation function in each layer and compiled with the Adam optimizer. Additionally, the GAN will have 400 epochs to train the both generator and discriminator models, and there will also be dropout layers that will keep the model from being overfitted. The epoch was decided through narrowing down and choosing the max epoch possible where the system would yield better performance. Although it was not exact, between 200, 400, 800, and 1000 epochs, 400 yielded the best performance while any higher would only affect time to train and not performance.



Figures 19: GAN Final Layout

The Figure 19 above shows a more detailed view of the GAN architecture with the stacked LSTM and MLP layers as the generator and discriminator, respectively. Implementing the GAN system used Keras to build the specified models in Figure 19 by simply setting variables for the respective models and adding the necessary layers provided by the Keras API. The specific layers are provided, and just the hyper parameters need to be set when initializing these layers

Training the GAN consists of initializing and training both the discriminator and the generator with real data and fake data. With GAN, simply calling the training method and passing the number of epochs like many machine learning models is not sufficient. The training in GAN needs to be implemented where there is some control of how the GAN is trained. To do this, when training, each epoch is looped through and will select a random batch of data from the real data set using the numpy library. Fake data is also created with the numpy library's random function to create a (5, 5, 1) array of random fake data. The discriminator is trained with both

known real and fake data for it to be able to discern between real and fake sequences of closing price data. A target array filled with 1s is passed with the real data to train the discriminator to classify real data. On the other hand, an array with 0s is trained with fake data for the discriminator to classify the random data as fake. The generator is then trained by also using randomly generated data to make predictions, and the discriminator will continuously classify whether the data and predictions are falsely generated every epoch or not. The generator will then be updated and re-trained until it can make predictions against the random data that the discriminator passes as a real data and prediction combination.

V. RESULTS

A. *Metrics*

As seen in many stock market forecasting work, for example [10] and [22], popular evaluation metrics include: root-mean-square error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). These metrics are also evaluated here as well as manual evaluation of the target versus prediction charts.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{1}$$

$$\mathrm{MAE} = \frac{1}{n} \sum_{j=1}^{n} |y_j - \hat{y}_j| \tag{2}$$

$$\begin{aligned} \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n &\text{ are predicted values} \\ y_1, y_2, \ldots, y_n &\text{ are observed values} \\ n &\text{ is the number of observations} \end{aligned} \tag{3}$$

In the variables in (3), the observed values, y, are the known target values that are then compared to the prediction values, $\hat{y}$, which are retrieved from the forecasting model, and n is the number of samples being compared. These equations are given, but the calculations can be done by using the sklearn library in python, and passing in the target and predicted values.

Since different stock datasets have different price ranges, as part of the data preprocessing, all datasets used in the experiments have been normalized to allow for error evaluating and comparing.

RMSE and MAE are both error metrics that measure the average error of a set of predictions and true values. The difference is that with RMSE more weight is given to larger errors, which will reflect on the RMSE value always being larger than the calculated MAE. If the RMSE and MAE are the same, then there are no larger outlier errors that skew the RMSE higher.

With both RMSE and MAE, lower values are more desired [34].

$$\text{MAPE} = \frac{100}{n}\sum_{i}^{n}\frac{y_i - \hat{y}_i}{y_i} \tag{4}$$

MAPE is a metric that measures, on average, the difference between the expected and actual. MAPE values can be big if expected values are small. Although this metric is not ideal for GAN systems, LSTM prediction models often use this metric in literature, so it will be included. The dataset was normalized so the values fall within a small range which outputs a large MAPE value.

B. *Experimental Results*

1. Standalone Stacked LSTM

As previously mentioned, the standalone stacked LSTM was trained with 80 epochs, which seemed enough to optimize the loss of the prediction model. Different epochs were tested, and higher values provided no significant improvements to the loss values.

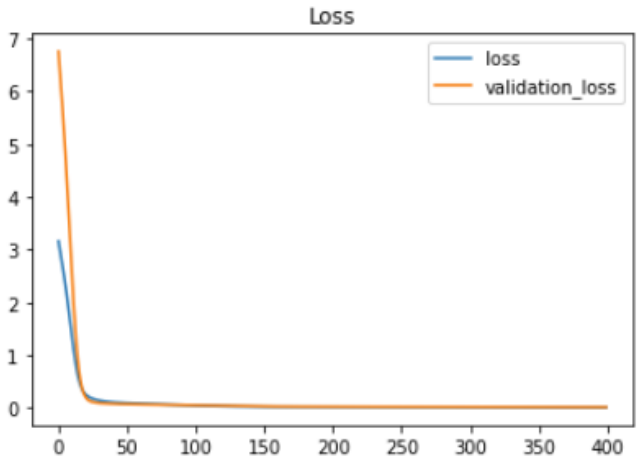


Figure 20. Plotted Loss Per Iteration

Figure 20 shows a graph of the loss over the training iterations of the standalone stacked LSTM. The loss starts off high in the early epochs, and then slowly lower and converges over time,

which shows that the model is performing better over time. The loss of both training and validation data sets stays between 0 and 0.05 after a few epochs, with some variations. Next, the test data set is used to verify and compare between the real 6th closing price and the predictions made through the previous 5 prices.

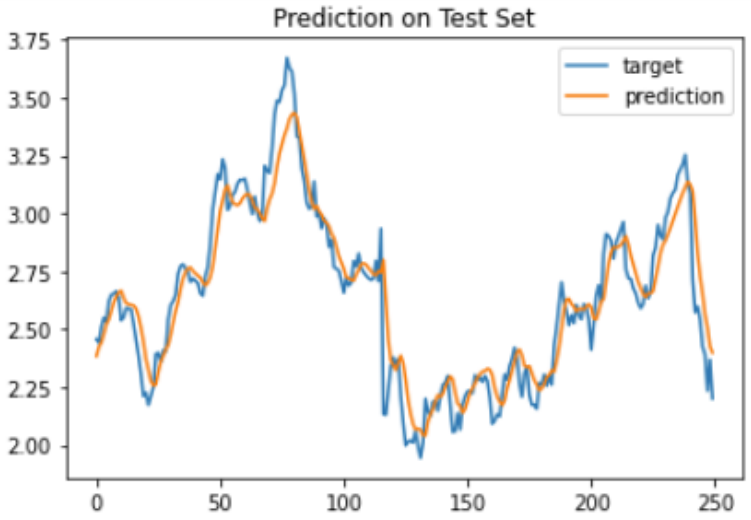


Figure 21. Test Data Expected Target vs Prediction

Figure 21 shows the predictions when the test data set is passed onto the model plotted along with the expected values, which are the target values. The graph shows similar price predictions as targets, but not exact due to the model not being too overfitted, but there is some degree of overfitting since some of the smaller peaks and troughs of price movement are similar to the specific SBUX data. The trained stacked LSTM, or LSTM-based DRNN, model will then be used as the generator within the GAN architecture.

2. LSTM with Generative Adversarial Search

After creating the stacked LSTM, the set up and initialization was carried over to the GAN implementation for the generator. The GAN system was then trained with the prepared SBUX dataset again at 400 epochs. The figure below is the loss chart of the discriminator loss, the discriminator loss with fake data, and the GAN generator loss. This shows a stable GAN loss

that converges around the 0 to 2 levels. As displayed on the chart, the plots for the losses alternate over each other, which is expected. The expectation comes from the fact that the adversarial network is opposing each other. If either the generator or discriminator performs better in an instance, this means that the opposing model must perform worse in that run.

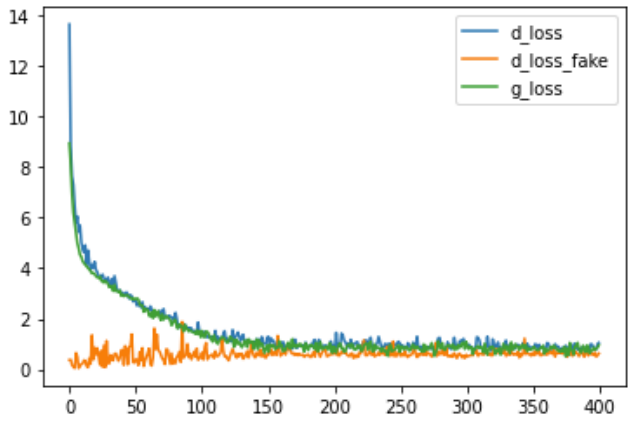


Figure 22. GAN Training Loss

Another visualization of the adversary process is the training accuracy of the discriminator classifying real and fake data. Figure 23 below shows two line plots that are constantly crossing over each other back and forth. This means that the fake data from the generator is catching up to the discriminator and vice versa. They are learning and updating from one another.

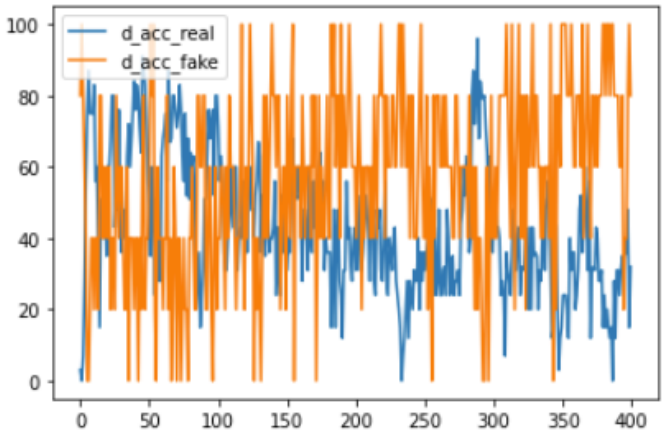


Figure 23. Discriminator Accuracy Plot

Furthermore, a separate dataset from Nike (NKE) stock from 2018 was input into the GAN model for predictions and then plotted on a chart, shown in Figure 24. The prediction follows closely near the target prices on a completely new dataset. This shows that the model is not specific to the dataset (SBUX) that was used to train it. The model is able to capture the market sentiment based on the 5 consecutive price movements, and make close predictions on a separate dataset that it was not trained with. Price action of every 5 timesteps was trained in the model which allowed it to be more generalized and not overfitted to the SBUX dataset only.

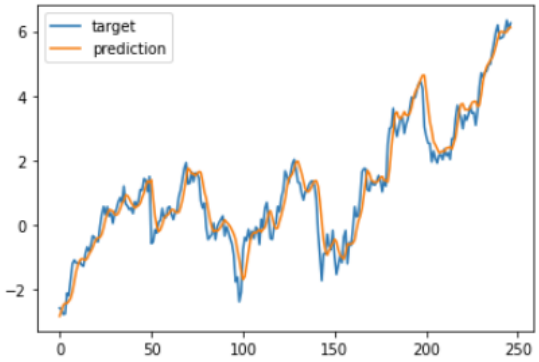


Figure 24. Prediction on NKE 2018 Dataset

With the GAN system having multiple components, there were different hyperparameters that must be configured for optimal results. It is important to correctly tune both the generator and discriminator to get efficient results. If the discriminator learns too slowly, it will have a hard time classifying fake data being passed to it. Consequently, this will negatively affect the generator by not acting as a competitive enough adversary. Other components, like the dropout layer, are also crucial in maintaining a generalization of the prediction and classifier and not overfitting to the data being used to train these models. Below is a table for a few of the notable results and hyperparameters experimented with during the tuning of the GAN.

Manual Tuning	Discriminator Learning Rate	Generator Learning Rate	Discriminator Dropout between layers	Generator Updates per Epoch	RMSE	MAE
A1	0.005	0.001	0.2	**2**	0.42138	0.31697
A2	0.005	0.001	0.2	4	0.43134	0.32564
B1	0.0005	0.0001	**0.5**	8	0.42402	0.31927
B2	0.0005	0.0001	0.1	8	0.54458	0.43681
C1	0.001	0.001	0.8	2	0.51455	0.40903
C2	0.0006	0.001	0.8	2	0.43486	0.34608
C3	0.01	0.001	0.8	2	0.39657	0.31232
C4	0.001	0.001	0.8	2	0.51455	0.40903
C5	**0.004**	**0.001**	**0.8**	**2**	**0.33589**	**0.25421**

Table 1. Hyperparameter Tuning

Table 1 shows some of the hyperparameter tuning that was performed on the GAN with the RMSE and MAE of the resulting predictions used as a performance metric. The notable tunings are grouped alphabetically on the table. In the A group, tuning was done to test the generator update per epoch. In each epoch, the discriminator and generator are both updated based on real and fake data. In this case, the generator was updated twice for every time the discriminator was updated. This allows the generator to more adequately train itself to trick the discriminator. The results showed that having more than two updates was not beneficial to the model. Furthermore, in group B, the value for the dropout layer was tested in the discriminator's classifier. A higher dropout value, like 0.5, seems to be better than a lower, 0.1, value. For group C, a high dropout value of 0.8 was used along with 2 updates per epoch for the generator. Group C tested the learning rates to see which ratio of learning rates worked best. The table shows that having a higher learning rate had better performance. From there, the learning rate between 0.0001 to 0.001 was tested and 0.004 performed the best.

The following Figure 25 is the final tuned GAN along with the No GAN LSTM implementation. Although the No GAN prediction looks pretty accurate from the prediction and target line plot, with the addition of GAN, better performing predictions were made. An example includes the high peaks, or outliers on the charts. In comparison, the stacked LSTM-GAN predicts prices closer to those peaks as well as closer to the troughs.

Figure 25. LSTM vs. LSTM GAN

A previous study from Mehtab and Sen, concluded that LSTM based models outperformed machine learning models as well as some other deep learning approaches [12].

A Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models
by
Sidra Mehtab & Jaydip Sen

	MV	MARS	DT	BAG	BOOST	RF	ANN	SVM	LSTM
Correlation	**0.99**	**0.99**	0.97	0.97	**0.99**	**0.99**	**0.99**	0.93	**1.00**
RMSE/Mean	13.32	**12.41**	35.34	33.43	23.40	16.26	14.99	53.88	**7.94**
Mismatched Cases	18.67	1.21	13.42	2.95	0.81	**0.00**	1.07	0.27	**0.00**

	MV	MARS	DT	BAG	BOOST	RF	ANN	SVM	LSTM
Correlation	**0.99**	**0.99**	0.97	0.98	**0.99**	**0.99**	0.97	0.98	**1.00**
RMSE/Mean	18.84	17.09	37.04	25.70	17.19	**10.82**	36.49	27.92	**4.04**
Mismatched Cases	51.31	4.28	17.38	5.10	4.69	**2.62**	15.31	4.41	**0.00**

	MV	MARS	DT	BAG	BOOST	RF	ANN	SVM	LSTM
Correlation	**0.99**	**0.99**	0.96	0.97	0.97	0.97	0.98	0.83	**0.99**
RMSE/Mean	**18.88**	20.40	56.34	34.91	41.51	32.02	35.29	82.96	**2.36**
Mismatched Cases	51.31	**6.34**	17.38	9.24	6.90	6.48	10.34	13.10	**2.40**

Figure 26. Study of Machine Learning and Deep Learning Models [12]

Figure 26 shows the results of the mentioned study, where LSTM's metrics were better than models like decision trees (DT), random forest (RF), support vector machine (SVM), and others. This led to a follow up study with Mehtab, Sen, and Dutta that focused on using LSTM for stock predictions [10]. Even with the worse tested case for LSTM, the model performed better than the competition. With the study showing great results for LSTMs, LSTM was also used in this study as a baseline to compare a GAN implementation that aims to improve on LSTM predictions.

The final stacked LSTM-GAN is compared to the No GAN stacked LSTM in Table 2. In the SBUX dataset used to train both prediction systems, the GAN is an improvement to the stacked LSTM model. As mentioned from Figure 25, where the GAN implementation predicts the outlier closing prices more accurately, the RMSE supports this observation. RMSE gives more weight to larger error differences. Compared to the RMSE from the No GAN implementation, LSTM-GAN's RMSE is lower. The overall average error of the LSTM-GAN is less, as reflected in the MAE. With the current metrics, the stacked LSTM-GAN has about a 22-25% better performance with the training dataset (SBUX), and about a 4-5% performance increase for a non-trained dataset (NKE).

	RMSE	MAE	MAPE
LSTM (SBUX)	0.50522	0.40040	195.319
LSTM (NKE)	0.55571	0.43412	322.131
Stacked LSTM-GAN (SBUX)	**0.39562**	**0.30253**	213.377
Stacked LSTM-GAN (NKE)	0.54009	0.40697	**139.925**

Table 2. GAN versus No GAN comparison

Throughout literature, researchers have shown that LSTM implements have outperformed traditional approaches as well as other machine learning approaches [12][17][20]. This study builds upon LSTM and extends it with another deep learning approach, GAN. By transitive law, since LSTM-GAN outperforms LSTM, and LSTM outperforms the other approaches shown in Figure 26, the LSTM-GAN approach is better than the machine learning approaches such as DT, RF, SVM, and others. With LSTM already outperforming others, GAN's purpose is to further use adversarial training to strengthen the generator used, which is the stacked LSTM. When implementing a GAN system, it is critical to optimize the system by tuning the hyperparameters to get better results. Overall, implementing GAN to further optimize an LSTM model for stock forecasting is effective.

VI. CONCLUSION AND FUTURE WORK

The goal of this project was to explore a generative adversarial search system applied to a long-short term memory prediction system for stocks. Specifically, a system that predicted closing prices of stocks based on the previous timesteps' prices and movement as a whole. The experiment attempted to imitate how an individual trader would analyze candlesticks and predict future stock prices. Using adversarial training, called GAN, an LSTM prediction model was further tuned and expected to have an increase in performance. From the results, it can be observed that the system can predict the future trend of the stock price based on the previous price action. Previous cited research has shown that LSTM outperforms popular machine learning approaches such as decision trees or random forests for stock predictions [12]. Since the LSTM-GAN implementation was better than LSTM, it also means it can beat the machine learning approaches that were tested in [12] as well. A few other cited literature sources also showed that LSTM was superior than popular machine learning approaches for time series data [17][20]. The stacked LSTM-GAN proposed model successfully demonstrated that it could perform better, and was an improvement to the simple stacked LSTM model. With a 22-25% increase in performance with the same data used to train both models, and 4-5% better performance on untrained data.

Further manual tuning of the hyperparameters in GAN yielded significant varying results. While the metrics are close to the no GAN implementation, finding the correct hyperparameters for the GAN system may significantly improve the performance. Future works may include more in depth analysis on the hyperparameters in GAN that will increase performance. Additionally, extra features from the data set could be explored along with the closing price as input data to the prediction system. For example, volume of the stock being traded is also often used in analysis of

price action in a position. More datasets could also be added into the training data to further train the model to be less overfitting to one dataset. Aside from this, datasets like $SPY or other indexes that track multiple stocks as one may be a better option to use to train the prediction model to predict based off of price actions and movements. Since indexes track multiple stocks, the price movements may be a better generalization of market sentiment than the single company stock used (SBUX). Also, with single company stock data, the high volatility of price action specific to that company around earnings timesteps may skew the prediction model. As discussed, there are a variety modifications possible for future work that may result in a more successful LSTM-GAN prediction model.

REFERENCES

[1] Y. Mao, B. Wang, W. Wei, B. Liu. “Correlating S&P 500 stocks with twitter data.” In: Proceedings of the first ACM international workshop on hot topics on interdisciplinary social networks research, New York, vol 12e16, pp 69–72. 2012.

[2] A Mittal, A Goel. “Stock prediction using twitter sentiment analysis.”. Stanford University. 2011.

[3] “Yahoo Finance - Stock Market Live, Quotes, Business & Finance News,”. *Yahoo! Finance*. [Online]. Available: https://finance.yahoo.com/. [Accessed May 27, 2021]

[4] “MarketWatch: Stock Market News - Financial News - MarketWatch,” *MarketWatch.com*. [Online]. Available: https://www.marketwatch.com/. [Accessed May 27, 2021]

[5] "Kaggle: Your Machine Learning and Data Science Community," *Kaggle.com*. [Online]. Available: http://www.kaggle.com/. [Accessed May 27, 2021]

[6] P. Patil. “Stock Market Prediction Using Ensemble Of Graph Theory, Machine Learning And Deep Learning Models.” San Jose State University. May 2019.

[7] M. Nabipour, P. Nayyeri, H. Jabani, S. S. and A. Mosavi. "Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data; a Comparative Analysis," in *IEEE Access*, vol. 8, pp. 150199-150212. doi: 10.1109/ACCESS.2020.3015966. 2020.

[8] M. U. Gudelek, S. A. Boluk and A. M. Ozbayoglu. "A deep learning based stock trading model with 2-D CNN trend detection." 2017 IEEE Symposium Series on Computational Intelligence (SSCI), 2017, pp. 1-8, doi: 10.1109/SSCI.2017.8285188.

[9] J. WeiWei. “Applications of deep learning in stock market prediction: recent progress.” *ArXiv* abs/2003.01859. 2020.

[10] S. Mehtab, J. Sen, and A. Dutta. “Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models.” *ArXiv* abs/2009.10819. 2020.

[11] M. Velay and D. Fabrice. "Stock chart pattern recognition with deep learning." arXiv preprint arXiv:1808.00418. 2018.

[12] S. Mehtab and J. Sen. "A time series analysis-based stock price prediction using machine learning and deep learning models." arXiv preprint arXiv:2004.11697. 2020.

[13] M. Sharaf, E.ED. Hemdan , A. El-Sayed and N.A. El-Bahnasawy. “StockPred: a framework for stock price prediction”. Multimed Tools Appl. https://doi.org/10.1007/s11042-021-10579-8. 2021.

[14] M. Venkatachalam. “Recurrent Neural Networks.” Medium. Towards Data Science, June 22, 2019. [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce. [Accessed May 27, 2021]

[15] “Understanding LSTM Networks.” Understanding LSTM Networks -- colah's blog. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed May 27, 2021]

[16] H. Jia. "Investigation into the effectiveness of long short term memory networks for stock price prediction." arXiv preprint arXiv:1603.07893. 2016.

[17] A. Moghar, M. Hamiche. “Stock Market Prediction Using LSTM Recurrent Neural Network.” Procedia Computer Science. 2020.

[18] R. Pascanu, G. Caglar, C. Kyunghyun, and B. Yoshua. "How To Construct Deep Recurrent Neural Networks." arXiv preprint arXiv:1312.6026. 2013.

[19] G. Ding, L. Qin. “Study on the prediction of stock price based on the associated network model of LSTM.” Int. J. Mach. Learn. & Cyber. 11, 1307–1317 https://doi.org/10.1007/s13042-019-01041-1. 2020.

[20] Y. Wu, and J. Gao. "AdaBoost-based Long Short-term Memory Ensemble Learning Approach for Financial Time Series Forecasting." Current Science (Bangalore) 115, no. 1 (2018): 159-65. 2018.

[21] J. Brownlee. “A Gentle Introduction to Generative Adversarial Networks (GANs).” Machine Learning Mastery, July 19, 2019. [Online]. Available: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/. [Accessed May 27, 2021]

[22] K. Zhang, G. Zhong, J. Dong, S. Wang and Y. Wang. “Stock Market Prediction Based on Generative Adversarial Network.” Procedia Computer Science. 147. 400-406. 10.1016/j.procs.2019.01.256. 2019.

[23] R. Valle. “Hands-On Generative Adversarial Networks with Keras.” Birmingham: Packt Publishing, Limited. 2019.

[24] “What is a multi-layered perceptron?” Educative. [Online]. Available: https://www.educative.io/edpresso/what-is-a-multi-layered-perceptron. [Accessed May 27, 2021]

[25] X. Zhou, Z. Pan, G. Hu, S. Tang, and C. Zhao. “Stock market prediction on high-frequency data using generative adversarial nets.” Mathematical Problems in Engineering, 2018. 2018.

[26] S. Siami-Namini, N. Tavakoli & A. Siami Namin.” A comparison of arima and lstm in forecasting time series.” In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1394–1401). doi:10.1109/ICMLA.2018.00227. 2018.

[27] H. Pesaran. Time Series and Panel Data Econometrics, Oxford University Press. 2015.

[28] S. Sperandei. “Understanding logistic regression analysis.” Biochemia Medica: Biochemia Med 24(1): 12–18. 2014.

[29] H. Gunduz, Y. Yaslan, and Z. Cataltepe. “Intraday prediction of borsa istanbul using convolutional neural networks and feature correlations.” Knowledge-Based Systems, 137 , 138–148. 2017.

[30] S.P. Poornima, C.N. Priyanka, P. Reshma, S.K. Jaiswal, and K.N. SurendraBabu. "Stock Price Prediction Using KNN and Linear Regression." International Journal of Engineering and Advanced Technology 8, no. 5 pg. 142-45. 2019.

[31] S. Basak, S Kar, S. Saha, L Khaidem, and S.R. Dey. "Predicting the Direction of Stock Market Prices Using Tree-based Classifiers." The North American Journal of Economics and Finance 47 (2019): 552-67. 2019.

[32] G. Cohen, “Best candlesticks pattern to trade stocks,” International Journal of Economics and Financial Issues, vol. 10, no. 2, pp. 256–261, 2020.

[33] W. Hu, Y.-W. Si, S. Fong, and R. Y. Lau, “A formal approach to candlestick pattern classification in Financial Time Series,” Applied Soft Computing, vol. 84, p. 105700, 2019.

[34] N. Vandeput, “Forecast KPI: RMSE, Mae, Mape & Bias,” Medium, 30-Jul-2021. [Online]. Available: https://towardsdatascience.com/forecast-kpi-rmse-mae-mape-bias-cdc5703d242d. [Accessed: 11-Nov-2021].