Effectiveness of Generative Adversarial Network Architecture for the Modeling of Stock Market Data

Cade Harger[1] and Ayden Tanner[1]

Trinity University, San Antonio TX 78212, USA

Abstract. It is quite difficult to predict the stock market and its trends, though such predictions are highly valuable and sought after. Through a multitude of models and techniques, AI has been applied to this complex task in an attempt to predict future prices based on historical stock data. Though there exist multiple academic papers exploring different implementations of one such model, a generative adversarial network, or GAN, this method has still been underutilized. In this paper, we create a GAN to predict some number of future trading days of a given stock. We used data on the stocks comprising the NASDAQ index over a period of multiple years. Using a combination of convolutional and dense layers in both portions of the GAN, we found that we were unable to replicate the accuracy produced by other similar models.

1 Background

1.1 Introduction

In the financial world, the prediction of stock market data is a hugely sought-after objective. Such predictions would be hugely lucrative as knowing when and how the value of stocks will change would lead to more advantageous trades. However, the stock market is a complex and dynamic system affected by a multitude of factors, making it difficult to accurately predict its movements. However, with the increasing availability of data and advancements in machine learning techniques, predicting stock prices using neural networks has become a promising area of research, one that has been explored already in myriad papers. In this paper specifically, we will explore the use of different architectures in both the generators and discriminators of generative adversarial networks (GANs) to predict stock market data. We will then examine their performance and accuracy compared to methods brought forth in other studies. More specifically, we will analyze the effectiveness of these different architectures in predicting future stock prices, and discuss their potential applications in real-world trading scenarios.

1.2 Literature Review

Stock Market Prediction Based on Generative Adversarial Network (Kang Zhang et al.) [6] This paper proposes a novel architecture of Generative Adversarial Network (GAN) using Multi-Layer Perceptron (MLP) as the discriminator and Long Short-Term Memory (LSTM) as the generator for forecasting the closing price of stocks. The generator is built by LSTM to mine the data distributions of stocks and generate data in the same distributions, whereas the discriminator built using an MLP aims to discriminate the real stock data and generated data. This model was trained using 20 years of stock data from multiple indexes and firms including the Standard and Poor's 500, Shanghai Composite Index in China, IBM, and Microsoft, among others. The experiment's results show that the proposed GAN model outperforms other machine learning and deep learning models in predicting the closing price of stocks in a wide range of training periods. This conclusion was made by evaluating the models on their Mean Absolute Error (MAE), their Root Mean Square Error (RMSE), their Mean Absolute Percentage Error (MAPE), and their Average Return (AR). The average percentage error (MAPE) for this model was 0.0137%. Other evaluated models and their errors included: LSTM (0.0145%); ANN (0.0808%); SVR (0.0452%).

Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets (Xingyu Zhou et al.) [5] The paper proposes a generic framework that utilizes Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) for adversarial training to forecast high-frequency stock market data. The model takes publicly available index data from the Chinese stock market in 2016, specifically a CSI 300 stock. The model also puts significant emphasis on the effects of a changing model update cycle. In this particular case, it was found that shorter model update cycles led to more accurate short-term predictions. These experimental results indicated that the approach proposed can improve stock price prediction accuracy effectively. This model and its competitors were evaluated using Root Mean Squared Relative Error (RMSRE); when using their most effective model parameters, the study's model's error varied between 0.48% and 1.49% depending on the stock that was being evaluated.

Stock Price Forecasting by a Deep Convolutional Generative Adversarial Network (Alessio Staffini) [3] This paper proposes a different, novel architecture for a Generative Adversarial Network (GAN); this model was termed a Deep Convolutional Generative Adversarial Network (DCGAN) and was tasked with producing the closing price of stocks over different time intervals. The model was trained on stocks from the Financial Times Stock Exchange Milano Indice di Borsa (FTSE MIB), with data from 2005 to 2021. The generator was constructed from a convolutional layer, a bi-directional long short-term memory (Bi-LSTM) layer, and an artificial neural network (ANN). The discriminator was constructed of two convolutional layers and an ANN. The model had a unique training scheme wherein it trained the discriminator five times more than the generator: only training the generator every fifth iteration. In both single-step and multi-step forecasting, the model performed better than traditional linear and non-linear methods such as an autoregressive integrated moving average (ARIMA) or Long Short-Term Memory (LSTM), though these structures were not being used inside of a GAN model. These models had their effectiveness determined by their Root Mean Squared Error (RMSE), their Mean Absolute Error (MAE), and their Mean Absolute Percentage Error (MAPE). Depending on the stock, there were significant differences in the amount of error the models produced with the percentage error (MAPE) of their model (the most effective one) ranging from 2.8% to 5.9%.

2 GAN Applicability

Machine learning techniques, particularly Generative Adversarial Networks (GANs), have been relatively underutilized in the realm of stock market prediction. Nonetheless, their potential for generating more actionable insights for stock market traders is immense. Emphasizing probability distributions over single-point predictions proves to be far more advantageous; understanding the probability of distinct closing price ranges or the likelihood of varying degrees of positive or negative change empowers traders to make better-informed purchasing decisions or explore suitable options. Recognizing that a stock has an 80% chance of marginally increasing, a 10% chance of remaining stable, and a 10% chance of experiencing a substantial drop is decidedly more valuable than merely obtaining a single output price that's marginally lower than the current one. While that singular output price indeed represents the average expected price, it fails to convey the complete picture of possible market outcomes.

One might then ask, why not have a traditional system (one that outputs a single estimation) output a price distribution rather than what it thinks is the most likely price? When predicting over a series of days, a typical model would take in the previous days as input and output the distribution. However, there is no way to feed the distribution it just produced back into itself to predict the day after that. Therefore, if the first day was predicted to have had an extreme output relative to previous days there would be no way for the model to take that into account when predicting the days following.

GANs do not suffer from this issue: given a series of length $d-n$ previous prices in the market, any number of length output price patterns of a given length n can be generated. Therefore, this model generates any number of continuations of length d, which represents the input market data in addition to n additional predicted days. This allows us to determine a probability distribution given any previously predicted prices by generating a large number of predictions for a single input of d - n prices. We can also take the subset of generated continuations that reach a given price on a day $(d+x-n)$, and use it to create a distribution of all the following $(n-x)$ prices. This creates a model that captures the maximal amount of information as it can grasp the dependence of one day and the next as well as understanding the full distribution of prices instead of the most likely one.

3 Data

3.1 Data Set

We used the 'Stock Market Dataset' from Kaggle, which can be found at the following link: https://www.kaggle.com/datasets/jacksoncrow/stock-market-dataset. This dataset contains daily, historical data for all stocks included in the NASDAQ index up to April 1st, 2020. This dataset contains data on the opening price, maximum price, minimum price, closing price, closing price adjusted for dividends, and the volume, or number of shares that were traded that day. We cut out the most recent 50 days of the dataset, as there were significant outliers in the data that reflected the COVID-19 Pandemic, something that we do not expect our model to try to account for.

3.2 Selection of Training and Test Set

This model makes predictions on the maximum and minimum price and the volume of traded shares for each day; therefore, only those relevant measures are included from the data set.

Not all stocks have the same number of available days of information and so they needed to be standardized to fit the network input shape. Using some constant number of days t, stock histories are selected and pruned: any stock with less than this number of days is rejected for not having enough information, and any stock with more has its excess days cut off.

Because all of these stocks' data points are from the same stretch of time, market trends must be accounted for. While individual stocks in the market go up or down to varying degrees on any given day, there are significant, market-wide trends that can be identified in the data. However, we are trying to identify and extrapolate based on the unique trends of a specific stock. If the market-wide trends are not accounted for, the GAN may memorize these market-wide trends and use that as the primary driver of its predictive output, leading to similar predictions for most stocks as it extrapolates these wide-scale trends and puts less emphasis on stock-specific details. Of the t days of data, some continuous subset of that, with a length of d days, is taken starting from a random location. This will ensure the market-wide trends do not match up across different inputted sets. These d days make up the training data for the GAN.

For the testing data, days that have not been used in the training of the model, either for inputs or labels, will be predicted. Each continuous subset of d days will be converted to a training sample. The entire d days will be used as a label. For inputs, the initial $d - n$ days of the sample will be taken, and n days of random noise will be added, effectively masking the predicted days. This same technique will be performed for the test evaluation dataset, but instead of a randomly selected continuation, it will be the most recent continuation.

We utilized Min-Max Normalization to normalize our data. As stock prices and volume can reach very high numbers, normalization is important to obtain an accurate model. The minimum and maximum from the input k days is used for normalization on both the input k days as well as the d days of continuation used as our label. By not including the extrema reached in the label continuation in our normalization, we avoid leaking the extrema information to the generator as it could learn to use the maximum input values to determine the maximum output values, which does not reflect real-world capabilities.

4 Methodology

As seen in *Fig. 1*, we include the $d - n$ days that are given to the generator in its output and apply only Mean Squared Error (MSE) loss through these output neurons. This is done to jumpstart training in the generator by forcing it to encode and preserve the stock's information throughout the architecture. This was proven to be a better approach for quicker training than leaving the given days out of the output entirely early on during testing. In turn, the discriminator loss is only applied to output neurons responsible for the predicted n days. This ensures we retain the benefits of GANs addressed in section 2 without convoluting the training process. The quality of these gradients is ensured by feeding in the n days appended to the original input $d - n$ days to the discriminator. This prevents it from disqualifying the prediction based on information that is not integral to our training goals.

Our generator architecture consists of 3 consecutive Dense layers, each followed by a Leaky ReLU activation, followed by a Transpose Convolution and Leaky ReLU to generate the output. This architecture proved successful due to its simplicity and information bandwidth. The convolutional layer allows for the output graph to be smoother and captures the inductive bias of stock prices being dependent on their neighboring data.

Our discriminator architecture consists of four consecutive convolutional layers, each with Leaky ReLU activation. This is followed by three dense layers with ReLU and finally sigmoid activation. This architecture allows for the monitoring of the inductive bias of stock information through the initially convolutional layers. This information can then be processed using the following dense layers.

5 Models

5.1 Our GAN

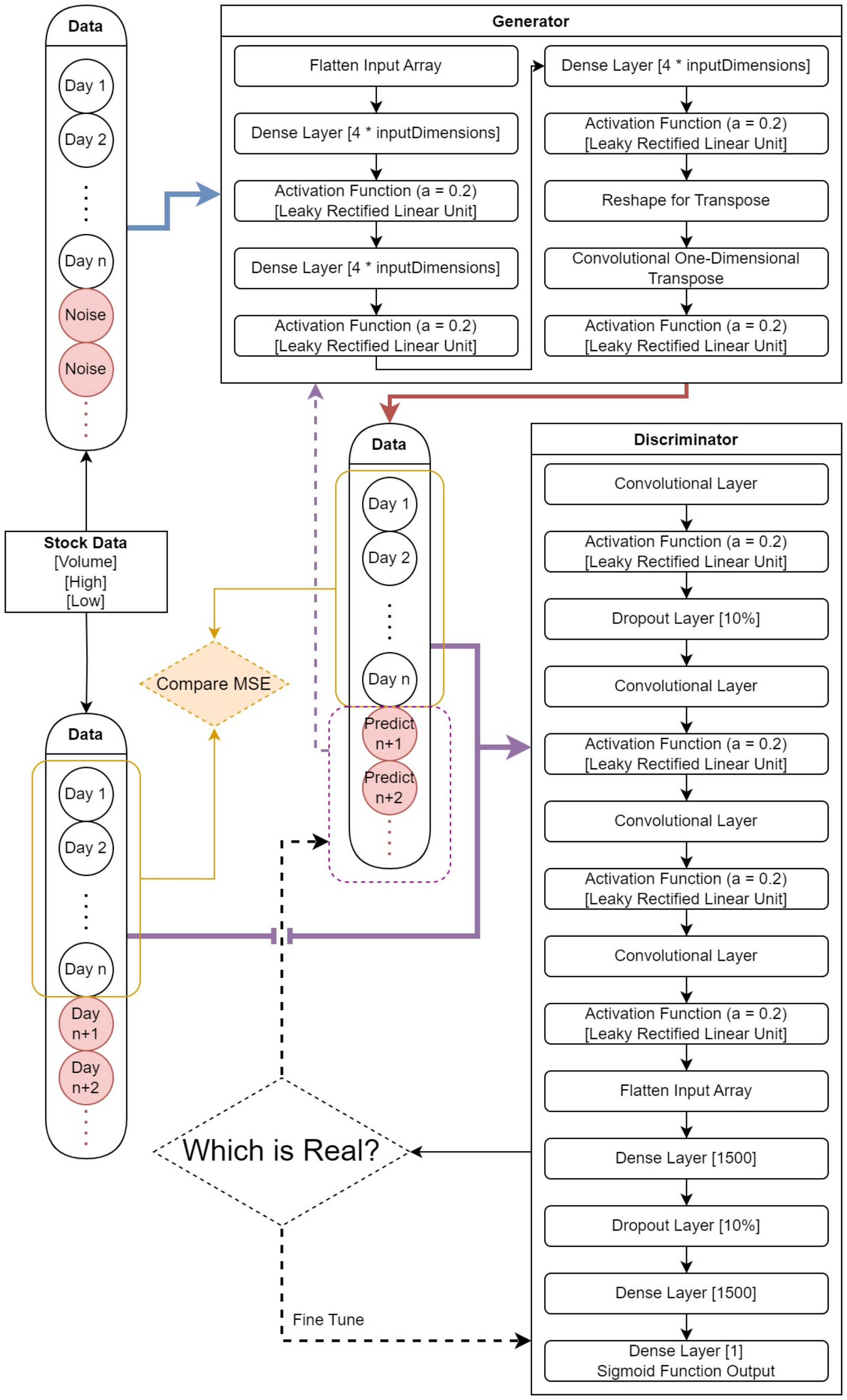
Data
Day 1
Day 2
Day n
Noise
Noise
Generator
Flatten Input Array
Dense Layer [4 * inputDimensions]
Activation Function (a = 0.2) [Leaky Rectified Linear Unit]
Dense Layer [4 * inputDimensions]
Activation Function (a = 0.2) [Leaky Rectified Linear Unit]
Dense Layer [4 * inputDimensions]
Activation Function (a = 0.2) [Leaky Rectified Linear Unit]
Reshape for Transpose
Convolutional One-Dimensional Transpose
Activation Function (a = 0.2) [Leaky Rectified Linear Unit]
Stock Data [Volume] [High] [Low]
Compare MSE
Data
Day 1
Day 2
Day n
Predict n+1
Predict n+2
Data
Day 1
Day 2
Day n
Day n+1
Day n+2
Discriminator
Convolutional Layer
Activation Function (a = 0.2) [Leaky Rectified Linear Unit]
Dropout Layer [10%]
Convolutional Layer
Activation Function (a = 0.2) [Leaky Rectified Linear Unit]
Convolutional Layer
Activation Function (a = 0.2) [Leaky Rectified Linear Unit]
Convolutional Layer
Activation Function (a = 0.2) [Leaky Rectified Linear Unit]
Flatten Input Array
Dense Layer [1500]
Dropout Layer [10%]
Dense Layer [1500]
Dense Layer [1] Sigmoid Function Output
Which is Real?
Fine Tune

Fig. 1. Our GAN

6 Notable Pursuits

After reading *Resolution-robust large mask Inpainting with Fourier Convolutions* [4], the application of Fast Fourier Convolutions (FFC) [1] to stock market data seemed promising, as they had been shown to be superior to many traditionally applied architectures in the field of image inpainting. In stock market prediction, you are effectively inpainting a region of the stock graph. On top of this, predictable behavior that would emerge in the graph of a stock would come in the form of a recurring pattern, something that a Fourier transformation would seem to excel at.

Unfortunately, training in an architecture consisting of mostly FFCs resulted in significantly poorer performance. This seemed to be due to the complexity of stock data and difficulties with combining it with our GAN structure. FFCs applied to stock market data seem like it would be a good avenue for future research.

7 Results

7.1 Comparison Table

Table 1. GAN Accuracy Metrics for 5 Day Predictions on Different Architectures

Metric	Stock Price Forecasting[3]	Stock Market Prediction[5]	Our GAN Model
RMSE	1.0687	4.1026	0.7704
MAE	0.7868	3.0401	1.5306
MAPE	3.5585	0.0137	155.92

–Note– X. Zhou et al.'s *Stock market prediction on high-frequency data using generative adversarial nets*[6] only asseses accuracy based on the Root Mean Squared Relative Error (RMSRE) metric. As we do not have access to the values this error is in relation to, we are unable to translate this metric into any of the other metrics. Therefore, we are unable to include this group's model in our accuracy assessment.

7.2 Assessment and Discussion

In both the previous studies' models, different models performed better in different metrics. The model of A. Staffini[3] had a much lower root mean squared error and mean average error than K. Zhang et al.'s model[5] but was completely outclassed in terms of mean average percent error. One would expect that a better model would perform equal to or better on all metrics compared to competing models; however, this does not seem to be the case given these numbers. We do not know which model is necessarily "better", but it is interesting that one model would be significantly better on two metrics and significantly worse on another.

One reason that could explain this discrepancy is a difference in tested values. Multiple data normalization schemes exist. Both RMSE and MAE are based on values: that is, they are both proportionally as large as the values they are comparing. For example, the difference between five and ten is 5 and the difference between one and two is one, but in both examples, the relative difference is a doubling. It could be that Zhang et al.'s model[5] uses numbers that are significantly larger than that of A. Staffini's model[3]. It could be that one model is min-max normalized like our model is, and so has values ranging from 0-1 while the other model used some other normalization scheme. If Zhang et al.'s model[5] does indeed use significantly higher values it would then make sense why their RMSE and MAE could be larger compared to the other model even while their mean average percentage error is significantly smaller.

After using the values of 405 for d and 5 for n, we put our model on an even playing field with the other papers, as we predict a similar amount of data. Data normalization seemed to affect how comparable our metrics are to similar papers. While our MAPE error was extremely higher in comparison, our other two errors were comparable and even better than the other papers. In future work, perhaps new metrics should be created that allows for better comparability, or the models should be implemented with a trading strategy that demonstrates their gain. The primary advantage of using a GAN would lie in utilizing the predicted distribution of plausible results in advanced trading strategies, which is something that these metrics completely fail to represent.

7.3 Mode Collapse

While training several versions of the model, our GAN experienced the problem that plagues many GAN models: mode collapse. Predicted outputs would look almost identical despite completely different stocks. This problem still partially affects some of the model's outputs, but it has been mostly alleviated. One of the ways this issue was resolved was by increasing the size of the discriminator's architecture. This way, it can memorize examples that the generator repeatedly uses and can ensure that it knows they are not real stock continuations.

8 Conclusion

Ultimately, we were unable to produce accuracy results similar to other stock prediction GAN models. Of the ones that were used, the accuracy metric that best represents the performance of the model is mean average percentage error (MAPE) due to it not being proportional to the magnitude of tested values. Using this metric, our model had a MAPE of 155.92, which is significantly worse than that of K. Zhang et al.'s model[5] or A. Staffini's model[3]. In addition to the explanations offered in the preceding section, there are two other issues that may have led to such poor results. For one, we may have not had data of the same quality. A. Staffini's model[3] was trained on stock market data from 2005 to 2021, a 16-year span. K. Zhang et al.'s model[5] not only uses 20 years' worth of data, but also uses data from "Standard & Poor's 500 (S&P 500 Index), Shanghai Composite Index in China, International Business Machine (IBM) from New York Stock Exchange (NYSE), Microsoft Corporation (MSFT) from National Association of Securities Dealers Automated Quotation (NASDAQ), [and] Ping An Insurance Company of China (PAICC)"[5]. This dataset represents a significant breadth of data points. In comparison, our model used about 700 days of stock data to train the model, which is a little less than two years. Though we did use NASDAQ data, which includes thousands of stocks, it is interesting to note that the accuracy of the models seems to be proportional to the quality of the data used to train it, with K. Zhang et al.'s model[5], trained on 20 years of diverse data, being the most accurate by the assessed metrics. In addition to the dataset explanation, the accuracy may also be affected by the computational demands required to train a complex GAN. As a GAN has both a discriminator and a generator element, it is more computationally demanding to create a model complex enough to capture the intricacies of stock data relative to a traditional deep neural network. When attempting to tackle a problem as complex as stock predictions the datasets are also necessarily very large. Even though we only ended up having three features, there were still three data points for every one of thousands of stocks, for every one of hundreds of days. Despite this, the other models from other papers trained on even larger datasets[3][5]. A contributing factor was that we do not have access to computational power equivalent to what was likely used to train these models; therefore, it is unlikely we could have produced a model as accurate as what they were able to produce unless we had devised a breakthrough novel architecture, which we entirely did not do. Unfortunately, the smaller, less diverse dataset and lack of equivalent computational power meant that it would have been incredibly difficult to match the accuracy of other models on a task as complex as this.

References

1. Chi, L., Jiang, B., & Mu, Y. (2020). Fast Fourier Convolution. Advances in Neural Information Processing Systems, 33. https://papers.nips.cc/paper/2020/hash/2fd5d41ec6cfab47e32164d5624269b1-Abstract.html
2. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014, June 10). Generative Adversarial Nets. arXiv.org e-Print archive. https://arxiv.org/pdf/1406.2661.pdf
3. Staffini, A. (2022). Stock price forecasting by a deep Convolutional generative adversarial network. Frontiers in Artificial Intelligence, 5. https://doi.org/10.3389/frai.2022.837596
4. Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., & Lempitsky, V. (2022). Resolution-robust large mask Inpainting with fourier convolutions. 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). https://doi.org/10.1109/wacv51458.2022.00323
5. Zhang, K., Zhong, G., Dong, J., Wang, S., & Wang, Y. (2019). Stock market prediction based on generative adversarial network. Procedia Computer Science, 147, 400-406. https://doi.org/10.1016/j.procs.2019.01.256
6. Zhou, X., Pan, Z., Hu, G., Tang, S., & Zhao, C. (2018). Stock market prediction on high-frequency data using generative adversarial nets. Mathematical Problems in Engineering, 2018, 1-11. https://doi.org/10.1155/2018/4907423