Journal of Computer Science

Original Research Paper

Stock price prediction using Generative Adversarial Networks

[1]HungChun Lin, [1]Chen Chen, [2]GaoFeng Huang and [1]Amir Jafari

[1]Department of Data Science, Columbian College of Arts & Sciences, The George Washington University, District of Columbia, USA
[2]Department of Algorithm, Carina Medical LLC, Maryland, USA

Article history
Received: 03-12-2020
Revised: 24-02-2021
Accepted: 05-03-2021

Corresponding Author:
HungChun Lin
Department of Data Science, Columbian College of Arts and Sciences, The George Washington University, District of Columbia, *USA*
Email: hungchun_lin@gwmail.gwu.edu

Abstract: Deep learning is an exciting topic. It has been utilized in many areas owing to its strong potential. For example, it has been widely used in the financial area which is vital to the society, such as high-frequency trading, portfolio optimization, fraud detection and risk management. Stock market prediction is one of the most popular and valuable areas in finance. In this paper, it proposes a stock prediction model using Generative Adversarial Network (GAN) with Gated Recurrent Units (GRU) used as a generator that inputs historical stock price and generates future stock price and Convolutional Neural Network (CNN) as a discriminator to discriminate between the real stock price and generated stock price. Different from the traditional methods, which limited the forecasting on one-step-ahead only, by contrast, using the deep learning algorithm is possible to conduct the multi-step ahead prediction more accurately. In this study, it chose the Apple Inc. stock closing price as the target price, with features such as S&P 500 index, NASDAQ Composite index, U.S. Dollar index, etc. In addition, FinBert has been utilized to generate a news sentiment index for Apple Inc. as an additional predicting feature. Finally, this paper compares the proposed GAN model results with the baseline model.

Keywords: Stock price prediction; GAN; WGAN-GP; NLP

Introduction

Stock price prediction is an interesting and challenging topic as a time series prediction. Many studies have shown that the stock price is predictable and many classic algorithms such as Long Short-Term Memory (LSTM) and ARIMA are used in time-series predictions. Generative Adversarial Network (GAN) is one of the most powerful models to conduct prediction. The generator and discriminator in the model are adversarial, which helps increase the result's accuracy. GAN is widely used in image generating, but not in time series prediction. Since there are few studies on time series prediction using GAN, their conclusions are inconsistent according to their studies. This paper aims to use GAN to predict the stock price and check whether the adversarial system can help improve the time series prediction. Also, it includes the comparison between the traditional models, LSTM and GRU with the basic GAN and Wasserstein GAN with Gradient Penalty (WGAN-GP) model.

The main contribution of this paper can be summarized in the followings:

- In our experiments, we found the adversarial training with a 1D-CNN discriminator will enhance the performance of basic recurrent models; and the GRU-based generator is recommended in the aspects of training stabilization and testing performance.
- Different from the related work, most of them used a plain GAN model. This project applied the loss function from WGAN-GP. Our model is more stable and gives good performance even when making multi-step ahead predictions.
- This project also extracted the daily news topic through the Natural Language Processing, which is one of the vital indexes in the features, especially if there is an unexpected incident like COVID-19. Thus, this paper compared the model performance in the normal periods and the COVID-19 period.

Related Works

Stock prediction is widely used in traditional models such as LSTM, Gated Recurrent Units (GRU) and ARIMA. But there are few studies that make the prediction using GAN. And the result of using GAN to make the stock prediction is inconsistent. For example, Ricardo and Carrillo (2019) compared the performance of

the GAN model with traditional deep learning model LSTM. They used LSTM as the generator and Convolutional Neural Network (CNN) as the discriminator. Their specific goal was to predict whether the price would increase one day after the sample period. The result showed no significant differences between GAN and traditional model LSTM. Accuracy on the GAN model is 72.68% compared with 74.16% on shallow LSTM. The performance of GAN is even a little bit worse. However, according to the study from Zhang *et al.* (2019), they proposed a GAN model with the LSTM as the generator and Multi-Layer Perceptron (MLP) as the discriminator to forecasting the one day closing price of the stock, and also compared the result with baseline LSTM. The result showed that GAN performs better than their traditional baseline model. The GAN model's accuracy is about 75.54%, while for baseline, LSTM is 68.59%.

Theoretical Background

LSTM

Long short-term memory (LSTM) is a specific recurrent neural network (RNN) architecture. It was proposed in 1997 by Hochreiter and Schmidhuber (Hochreiter *et al.*, 1997). Unlike a traditional feed-forward neural network, it includes feedback connections. Furthermore, it can be utilized on single-point data and the sequence of data as well. The essential components of LSTM are an input gate, an output gate and a forget gate, and the LSTM network was developed to resolve the vanishing gradient problem while training the traditional RNNs. LSTM is a cell memory unit that means that LSTM can remove or add information to the cell state.

LSTM has overcome the vanishing gradients and the exploding gradients problem that appeared in RNN through the units' specific internal structure built in the model. Nowadays, LSTM has been known as a powerful method capable of processing, classifying, and making predictions based on time series data.

GRU

Gated recurrent unit (GRU) is a kind of RNN that uses gating mechanisms to control the flow of information between cells in the neural network derived from LSTM and was introduced in 2014 By Kyunghyun Cho *et al.* (1997). GRU is composed of two gates, an update gate and a reset gate. These gates are used to filter out what information should remain and what should be disposed of. Different from traditional RNN, GRUs solve the vanishing and exploding gradient problems. Unlike LSTM, GRU has fewer parameters than LSTM due to the lack of one gate. Another difference is that GRUs also lack the cell state from LSTM so that GRU can only store both long and short-term memory in the hidden state. Recently, GRUs have been shown to perform better than LSTM on certain smaller and less frequent datasets.

GAN

Generative adversarial network (GAN) is a minimax problem, which is based on zero-sum non-cooperative games. In general, GAN is composed of two components, a generator and a discriminator. The generator is aimed to generate examples that can look as real as possible, and the goal for the discriminator is to distinguish the examples as real or fake (generated).

GAN is a technique that has been rapidly explored in the deep learning field. Based on the basic structure, people are developing different methods for improving the result by adjusting the structure and loss function. Nowadays, various types of GAN have been proposed. Conditional GAN (CGAN) uses extra-label information to improve the generator. Wasserstein GAN (WGAN) includes Wasserstein distance to the loss function. WGAN with Gradient Penalty, which adds the regularization to their loss function. Cycle GAN, PGGAN and SAGAN change the structure.

Basic GAN

In the original GAN, the loss function is based on KL-JS divergence, in the training process, the GAN model will use cross-entropy loss to minimize the difference between two distributions which is equivalent to minimizing the KL-JS divergence.

In this project, the objective of discriminator is to maximize the probability of assigning the correct label to the samples. The mathematical objective function for discriminator is defined as:

$$\hat{V} = \frac{1}{m}\sum_{i=1}^{m} logD\left(y^i\right) + \sum_{i=1}^{m}\left(1 - log\ D\left(G\left(x^i\right)\right)\right) \quad (1)$$

and then we train generator to minimize its objective function which is:

$$\hat{V} = \frac{1}{m}\sum_{i=1}^{m}\left(1 - log\ D\left(G\left(x^i\right)\right)\right) \quad (2)$$

Where x is the input data for generator, y is the target from the real dataset, $G(x_i)$ is the generated data (fake target) from the generator.

For present the calculating through the training process in GAN, the loss function of discriminator is:

$$-\frac{1}{m}\sum_{i=1}^{m} log\ D\left(y^i\right)-\frac{1}{m}\sum_{i=1}^{m}\left(1-log\ D\left(G\left(x^i\right)\right)\right) \quad (3)$$

The loss function of generator is:

$$-\frac{1}{m}\sum_{i=1}^{m}\left(log\ D\left(G\left(x^i\right)\right)\right) \quad (4)$$

Through the training process, it always needs to minimize the loss function to get the better result.

WGAN-GP

The discriminator in Basic GAN is not powerful enough. The training process is known to be slow and unstable. WGAN-GP is proposed to help stabilize and improve the training of GAN.

WGAN-GP proposed the Wasserstein distance to solve this problem. The Wasserstein distance (or Earth-Mover Distance (EMD)) is the minimum cost of transporting mass in converting the data distribution to the data distribution. The Wasserstein distance for the real data distribution P_r and the generated data distribution P_g is mathematically defined as the greatest lower bound (infimum) for any transport plan (i.e., the cost for the cheapest plan) (Zhou *et al*., 2018):

$$W\left(P_r,P_g\right)=\inf_{\gamma\in\Pi\left(P_r,P_g\right)}E_{(x,y)\sim\gamma}\left[\|x-y\|\right] \quad (5)$$

Where $\Pi(P_r, P_g)$ denotes the set of all joint distributions between P_r and P_g, Π contains all the possible transport plan γ. Using the Kantorovich-Rubinstein duality, we can simplify the calculation to:

$$W\left(P_r,P_g\right)=\sup_{\|f\|_L\le 1}E_{x\sim P_r}\left[f\left(x\right)\right]-E_{x\sim P_g}\left[f\left(x\right)\right] \quad (6)$$

where sup is the least upper bound and f is a 1-Lipschitz function following Lipschitz constraint:

$$\left|f\left(x_1\right)-f\left(x_2\right)\right|\le\left|x_1-x_2\right| \quad (7)$$

WGAN-GP uses gradient penalty to enforce the Lipschitz constraint. A differentiable function f is 1-Lipschitz if and only if it has gradients with norm at most 1($\|\nabla f\|_2\le 1$) everywhere (Zhou *et al*., 2018). The model is penalized if the gradient norm moves away from its target norm value 1.

Compared with Basic GAN, the network is without the sigmoid function and outputs a scalar score rather than a probability. This score can be interpreted as how real the input data are (Zhou *et al*., 2018). In addition, a gradient penalty is used in the discriminator. Table 1 and 2 shows the comparison between the Basic GAN and WGAN-GP with their loss function of discriminator and generator.

Table 1: Comparison of Basic GAN and WGAN-GP discriminator loss function

	Discriminator loss
GAN	$-\frac{1}{m}\sum_{i=1}^{m}\left[logD\left(y^i\right)+\log\left(1-D\left(G\left(x^i\right)\right)\right)\right]$
WGAN-GP	$\frac{1}{m}\sum_{i=1}^{m}\left[D\left(y^i\right)-D\left(G\left(x^i\right)\right)+\lambda E\left(\|\nabla D_{y^i\sim x^i}\|_2-1\right)^2\right]$

Table 2: Comparison of Basic GAN and WGAN-GP generator loss function

	Generator loss
GAN	$-\frac{1}{m}\sum_{i=1}^{m}\log\left(D\left(G\left(x^i\right)\right)\right)$
WGAN-GP	$-\frac{1}{m}\sum_{i=1}^{m}D\left(G\left(x^i\right)\right)$

Methodology

The Generator

In our GAN model, we set the GRU as the generator according to its stability. Our dataset includes the past 10 years' history of the stock price and also consists of 36 features, includes Open, High, Low, Close, Volume, NASDAQ, NYSE, S&P 500, FTSE100, NIKKI225, BSE SENSEX, RUSSELL2000, HENG SENG, SSE, Crude Oil, Gold, VIX, USD index, Amazon, Google, Microsoft, MA7, MA21, MACD, 20SD, upper_band, lower_band, EMA, log momentum, absolute of 3 comps, angle of 3 comps, absolute of 6 comps, angle of 6 comps, absolute of 9 comps, angle of 9 comps and News. This project will make the multi-step ahead prediction. Therefore, in the generator, the input step and the output step need to be defined. The input of the generator will be three-dimensional data: batch size, input-step and features, and the output will be two-dimensional data: batch size and output-step. For building up a generator with good performance, this model uses three layers of GRU, the numbers of the neuron are 1024, 512 and 256, and then add two layers of Dense, and the neuron number of the latest layer will be the same as the output step we are going to predict.

The Discriminator

The discriminator in our GAN model is a Convolution Neural Network that aimed to distinguish whether the input data of the discriminator is real or fake. The input for the discriminator will be from the original data or the generated data from the generator. This discriminator includes three 1D Convolution layers with 32, 64, and 128 neurons separately add three other Dense layers in the end, which have 220, 220 and 1 neuron. The Leaky Rectified Linear Unit (ReLU) has been set as the activation function among all layers, but not in the output layer which is with the Sigmoid activation function for Basic GAN and linear activation for WGAN-GP. The sigmoid function will give a single scalar output, 0 and 1, which means real or fake, and the linear function will give a scalar score.

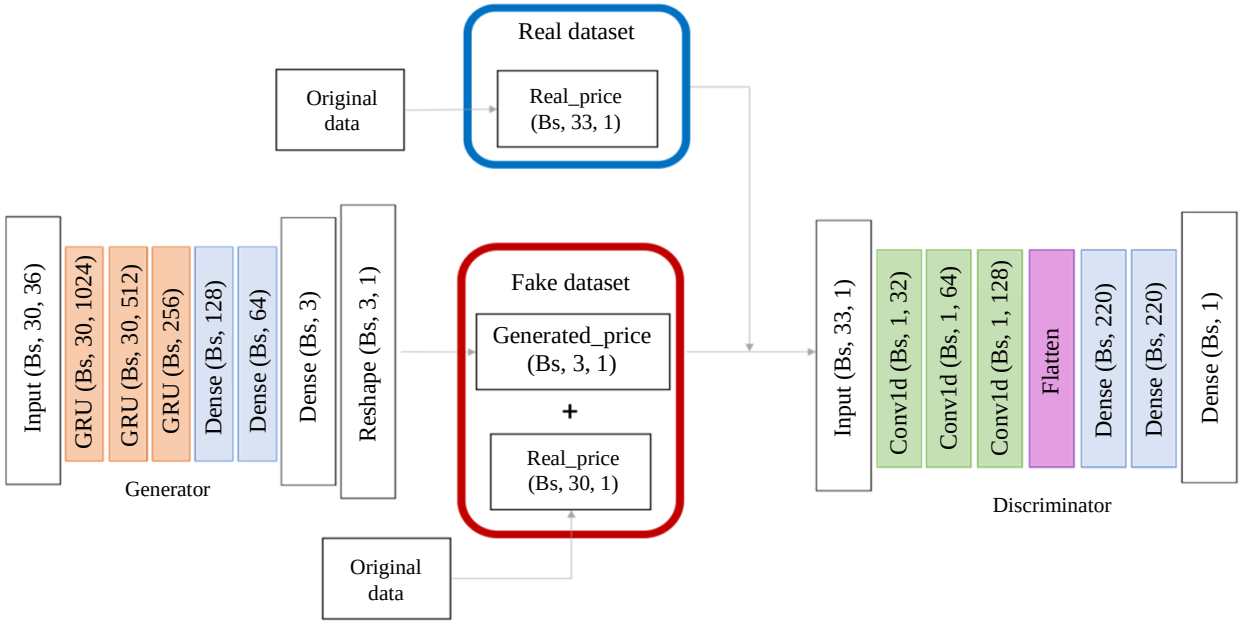

Fig. 1: GAN Architecture

The Architecture of GAN

The combination of the above generator and discriminator consists of our proposed GAN model. Figure 1 indicated the architecture of GAN which had been utilized.

In our GAN model structure, cross-entropy has been used to calculate the loss for both generator and discriminator, and the function has been defined in the Theoretical Background section. Especially in the discriminator, we combined the generated stock price with the historical stock price of input steps as our input for the discriminator, this step enhances the data length and increases the accuracy for the discriminator to learn the classification.

Hyperparameter tuning

In a machine learning algorithm, each model has a certain number of parameters that need to be defined through the training process, the number of layers, the learning rate, the number of the neuron, and some other parameters, that contain in the different types of the layers. Hyperparameter optimization is a process of tuning hyperparameters to achieve the highest performance score with the limited time.

In this study, Bayesian Optimization has been utilized in the training, which uses Bayes Theorem to find parameters that can produce the minimum or maximum score of the given objective function.Our model tuned the learning rate between 0.0001 to 0.0008, the number of epochs between 100 to 300, and the batch size between 64 to 512.

Dataset and Features

Dataset Descriptions

The stock price data and stock index data are from Yahoo Finance, the dollar index is from Fred, and the news data are scrapped from SeekingAlpha. The target stock price in the model is Apple.Inc. Stock closing price. The statistical data are calculated using the stock closing price. There are a total of 2497 observations and 36 variables in the dataset. The train data and test data are split into 7:3.

Feature Engineering

Some technical indicators have been calculated by downloading various asset historical data and some extracted trend features. In addition, this project uses NLP methods to sentiment values of relevant news.

Technical indicators: this study calculated the most popular technical indicators for investors (7 days and 21 days moving average, exponential moving average, momentum, Bollinger bands, MACD.).

News sentiment analysis: news can indicate potential stock price movements. This project scrapped all the daily news of Apple.Inc, and used FinBert to analyze the news into positive, neutral or negative by giving a score between -1 to 1.

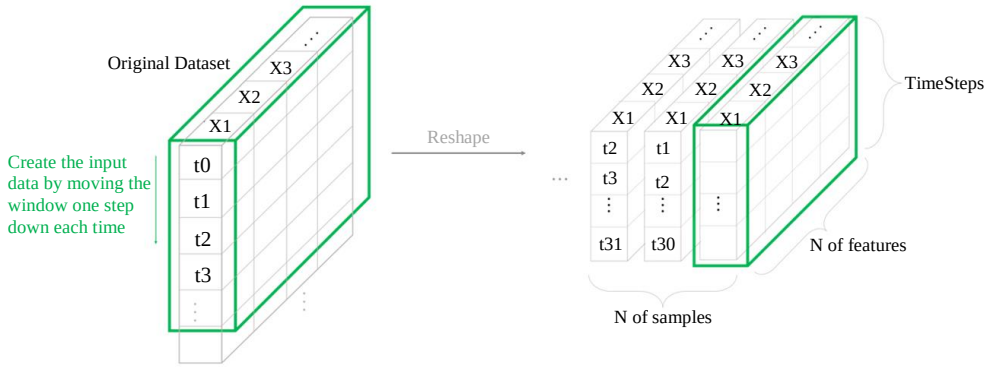

Fig. 2: Input data

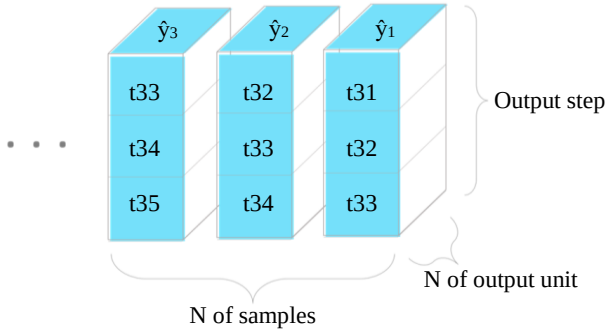

Fig. 3: Output data

Fourier transforms: along with the daily closing price, we created Fourier transforms to extract long term and short-term trends in the Apple stock. Fourier transforms take a function and create a series of sine waves; when combined, these sine waves approximate the original function, helping the GRU network pick its prediction trends more accurately (Banushev, 2020).

Data Structure

The method this model prepared the dataset for supervised learning is to divide the dataset with the rolling window equals 1. The illustration has been shown in Fig. 2. The original dataset is 2 dimensional, and it needs to be reshaped to 3 dimensions according to the timesteps.

Figure 3 illustrates the output of the dataset that will be obtained from the generator. Here, the number of output units equals 1. In our model, the time step can be modified in Fig. 2 and the output step in Fig. 3. This paper built a many to many model with timestep 30 and output step 3 (use 30 days historical price to predict 3 days stock price).

Experimental Results

Experimental Setup

The framework of this experiment has been set up with Keras with Tensorflow backend. The computational devices contain one NVIDIA Tesla M60 GPU with 8GB memory and one CPU with 16 GB memory. For the activation functions, linear (non-activation) is used in the generator, while Leaky ReLU with a low threshold of 0.01 in the discriminator experimentally results in a good performance.

Train the Model

The purpose of this paper is to predict the stock closing price in the following three days with the data of the past 30 days. For training the forecasting model, this project will input the historical closing price and 36 features that might affect the price. In the training process, the dataset will be split into a training set and a testing set as 70% (1726 data) and 30% (739 data). During the testing process, this study will do two different parts, a prediction with an unexpected event, and a prediction without an unexpected event, in this project, the unexpected event is COVID-19 for 2020.

Experimental and Results

In this paper, it evaluated the performance of each model by Root Mean Square Error (RMSE), and the indicator is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(x_i - \hat{x}_i\right)^2}{N}} \tag{8}$$

The N is the number of the data points, x_i is actual stock price, and $\hat{x}_i$ denotes the predicted stock price, to evaluate the models we built in this project, this study compared all the models of their RMSE on testing data (with 2020 and without 2020).

LSTM

In our LSTM model, Bidirectional LSTM has been utilized in the first layer. The optimizer for our models in this work is the Adam algorithm with a learning rate of 0.001. The batch size is 64, and then the model trains 50 epochs on this stock price dataset.

As the input of the GAN model, in this baseline model, the whole dataset includes the past 10 years' historical data and 36 correlative features. After splitting the data into the train set and test set, the testing dataset started on 07/21/2017.

Figure 4 shows that the result of LSTM includes the forecasting of the year 2020, the RMSE is 6.60, the blue line is the actual stock price, and the red line indicates the predicted stock price. Obviously, all the predicted stock price is slightly higher than the actual stock price till the end of May 2020. And after May 2020, the forecasting is much closer to the actual stock price. Furthermore, we calculate the result excluding 2020, and then the RMSE is increased to 9.42, which is much higher than the result that includes 2020.

GRU

GRU model, the second basic model in this paper. Building this model utilized 2 layers of GRU, and the optimizer for the GRU model is the Adam algorithm with a learning rate of 0.0001, and the size of the batch is 128, and then train this model for 50 epochs.

Figure 5 shows the result of GRU, including 2020, the RMSE is 5.33, and from the result, it indicates that the GRU model performs better than the LSTM model before May 2020. From this figure, we can observe the collapse of the forecasting after May 2020. Next, the result excluding 2020 for GRU, the RMSE is 4.08. The GRU model performs better when making predictions without predicting unexpected events.

Basic GAN

The structure of the GAN model in this paper has been proposed in the methodology section. In this model, the optimizer for our models in this paper is the Adam algorithm with a learning rate of 0.00016. The batch size is 128 and then the model on this dataset has been trained for 165 epochs.

Figure 6 is the loss plot of the basic GAN model, and the blue line is the loss path of the discriminator and the orange line is the loss path of the generator. From the beginning, the loss of discriminator is higher than the loss of generator, and through the training process, both loss paths are becoming flat.

Figure 7 is the predicted result of the basic GAN model, and the RMSE is 5.36. This figure shows the prediction started having a large gap between the actual price and the predicted price in 2020 while there is a sudden surge in actual price, which might be due to the unexpected event, COVID-19. In this model, we calculated the result of the basic GAN model excluding 2020 forecasting as well, and the RMSE turned out to decrease to 3.09. It indicates that the basic GAN for forecasting performs better without unexpected events than both basic models.

WGAN-GP

The structure of the WGAN-GP model in this paper has been proposed in the methodology section. In this model, the optimizer is also an Adam algorithm with a learning rate of 0.0001. The batch size is 128 and the model has been trained on this dataset for 100 epochs. In this study, we trained the discriminator once and the generator three times.

Figure 8 is the loss plot of the WGAN-GP model, the blue line is the loss path of the discriminator and the orange line is the loss path of the generator. The discriminator loss decreases towards 0. Compared with the Basic GAN, the discriminator in WGAN-GP learns better.

Figure 9 is the predicted result of the WGAN-GP model. The RMSE is 4.77, which is the best one in all the models. Like basic GAN in 2020, the prediction starts to have a large gap between the actual price and the predicted price due to the unexpected COVID-19. When removing the test data in 2020, the RMSE of the forecasting decreases to 3.88, which performs worse than the basic GAN model.

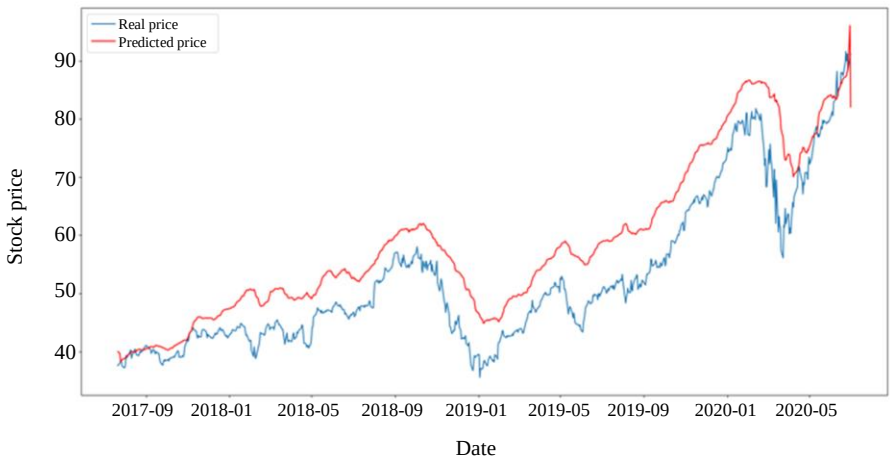

Fig. 4: LSTM test data plot

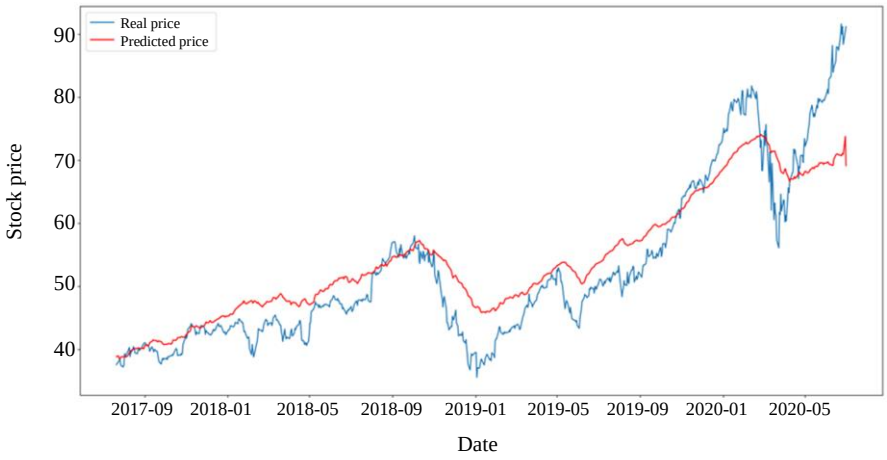

Fig. 5: GRU test data plot

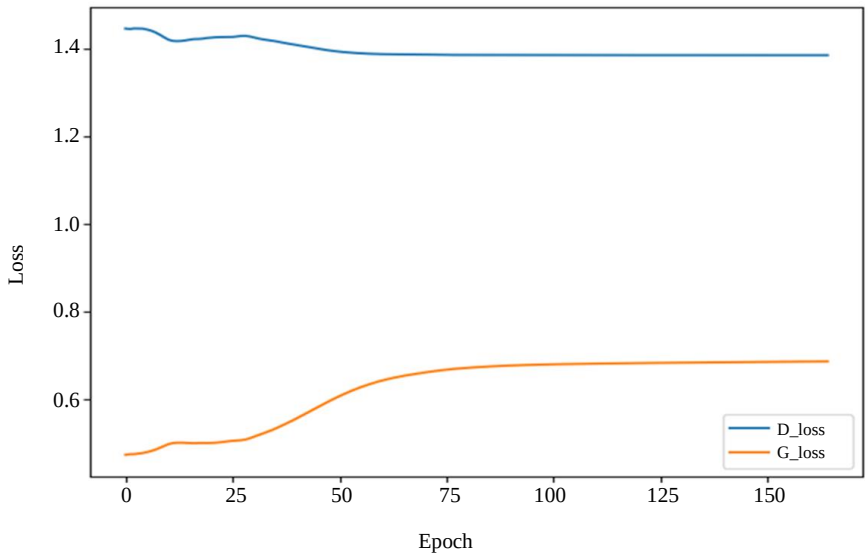

Fig. 6: Basic GAN loss plot

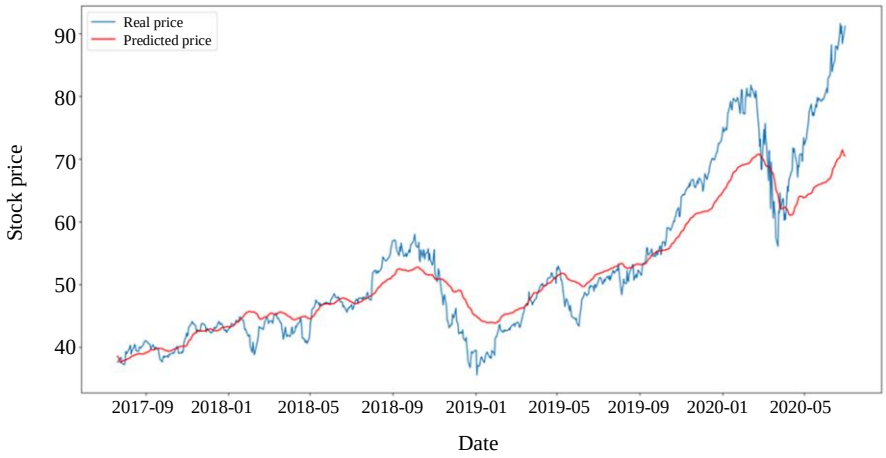

Fig. 7: Basic GAN test data plot

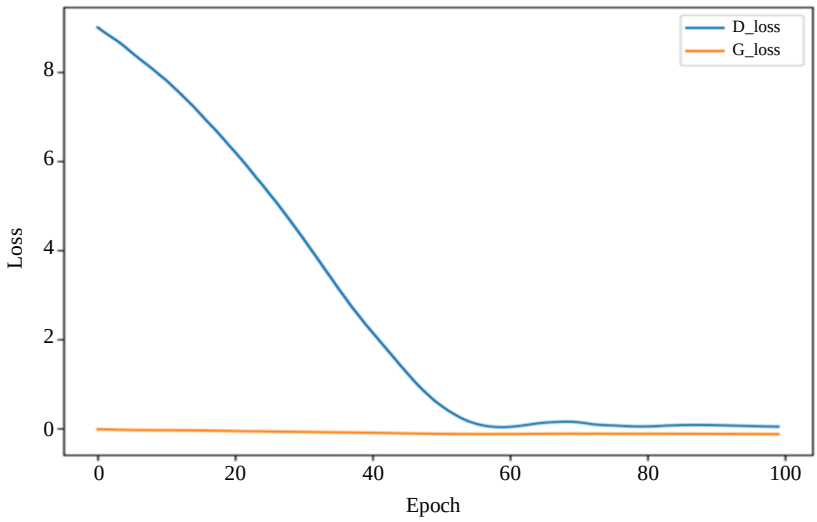

Fig. 8: WGAN-GP loss plot

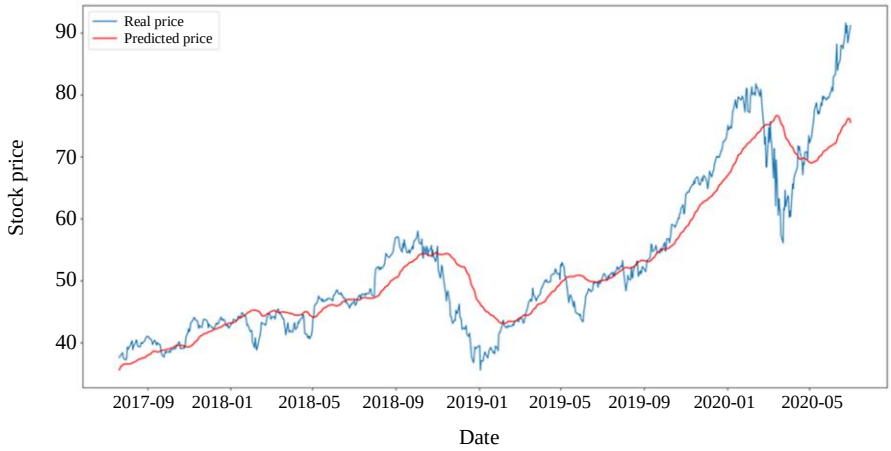

Fig. 9: WGAN-GP test data plot

Evaluation

Table 3: Model summary RSME

	LSTM	GRU	Basic GAN	WGAN-GP
RMSE (Train data)	1.52	1.00	1.64	1.74
RMSE (Test data include 2020)	6.60	5.33	5.36	4.77
RMSE (Test data exclude 2020)	9.45	4.08	3.09	3.88

The Table 3 compares the training RSME and testing RMSE for different models.

For the training dataset, GRU performs the best. While, for the testing dataset, when including COVID-19 period data, WGAN-GP performs the best, when excluding that period, basic GAN performs the best. But overall, GANs models perform better than the baseline traditional models according to our result.

Conclusion

This paper proposed a GAN that sets GRU as a generator and CNN as a discriminator. According to the experimental result, some conclusions have been summed. First, compared to the GAN model with the traditional models, the GAN model can improve the GRU and LSTM models. Both basic GAN and WGAN-GP perform better than conventional models. One of the key findings of this work is that when there is an unexpected event like COVID-19, WGAN-GP performs better than basic GAN, but basic GAN performs better in normal periods. However, to our knowledge, a GAN model including RNN is unstable. It is challenging for these models to tune hyperparameters. Without suitable parameters, you may have bad results.

Future research should be devoted to the development of hyperparameter tuning. In the GAN model, if each of the

parameters can be tuned more accurately in each layer and for the whole model, we believe the result would have been significantly improved.

Many research teams proposed methods of reinforcement learning for hyperparameter optimization, such as Rainbow (Hessel *et al*., 2018) based on Q-learning and Proximal Policy Optimization (PPO) (Schulman *et al*., 2017). Based on the basic structure in this paper and exploring further reinforce learning, we hope the GAN model with RNN can produce much more reliable forecasting of the stock price.

Funding Information

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Author's Contributions

HungChun Lin: Supervised the project as a whole and drafted the manuscript. Designed, presented and wrote the LSTM, GRU and Basic GAN model used in this study. Also did the data analysis work of the model.

Chen Chen: Supervised the project as a whole and drafted the manuscript. Designed, presented and wrote the Basic GAN and WGAN-GP model used in this study. Also did the data analysis work of the model.

GaoFeng Huang: Contributed with his deep learning and coding knowledge in this research work for improving the result of the research.

Amir Jafari: Contributed with his expertise in this research work. Guided all of us with his outstanding experience in deep learning.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Banushev, B. (2020). Using the latest advancements in AI to predict stock market movements. https://github.com/borisbanushev/stockpredictionai

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., ... & Silver, D. (2018, April). Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Ricardo, A., Carrillo, R. (2019). Generative Adversarial Network for Stock Market price Prediction. Stanford University. United States. https://cs230.stanford.edu/projects_fall_2019/reports/26259829.pdf

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

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.

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.