Journal of Computer Science

Original Research Paper

Stock price prediction using Generative Adversarial Networks

Hungchun Lin [1], Chen Chen, Amir Jafari and Gaofeng Huang[2]

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

*Corresponding Author:
HungChun Lin,
Department of Data Science,
District of Columbia, United States;
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, we propose 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. We choose 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, we use FinBert to generate a news sentiment index for Apple Inc. as an additional predicting feature. Finally, we compare our GAN model results with the baseline model.

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

Introduction

Stock price prediction is an interesting and challenging topic. Many studies have shown that the stock price is predictable. Stock price prediction is a kind of time series prediction, many classic algorithms such as Long Short-Term Memory (LSTM), ARIMA are used in time series predictions. Generative Adversarial Network (GAN) is one of the most powerful models, the generator and discriminator in the model are adversarial, which help to generate more accurate output. GAN is widely used in image generating, but not in time series prediction. Since there are few studies on time series prediction using GAN, and according to the result of their study, their conclusion is inconsistent. So, in this paper, we decide to use GAN to predict the stock price and to check whether the adversarial system can help improve the time series prediction. We will compare the traditional models, LSTM and GRU with the basic GAN and Wasserstein GAN with Gradient Penalty (WGAN-GP) model.

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 *(Alberto and Romero, n.d.)* 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, and the result showed there are no significant differences between GAN and traditional model LSTM. Accuracy on 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. *(Kang et al. 2019, 400-406)*, 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 baseline traditional model, the accuracy for GAN model is about 75.54% while for baseline LSTM is

Science Publications

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 *(Cho et al. 2014, 1724-1734)*. Unlike a traditional feed-forward neural network, it includes the feedback connections. Furthermore, it can not only be utilized on single-point data but on the sequence of data as well. The basic 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 which means that LSTM has the ability to 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 specific internal structure of the units built in the model. Nowadays, LSTM has been known as a powerful method that is capable of processing, classifying, and making predictions based on time series data.

GRU

Gated recurrent unit (GRU) is a kind of RNN that using gating mechanisms to control the flow of information between cells in the neural network, which was derived from LSTM and is introduced in 2014 By Kyunghyun Cho et al *(Hochreiter and Schmidhuber 1997, 1735-1780)*. GRU is composed by two gates, update gate and reset gate, these gates are used for filtering 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 have a lack of 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. Then nowadays, there are various types of GAN have been proposed, such as Conditional GAN (CGAN) which using extra label information to improve the generator, Wasserstein GAN (WGAN) that includes Wasserstein distance to the loss function, WGAN with Gradient Penalty which add the regularization to their loss function, Cycle GAN, PGGAN and SAGAN which 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, we train discriminator to maximize its objective function, the probability of assigning the correct label to the samples, the objective function for discriminator to maximizing is defined as:

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

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

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

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:

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

The loss function of generator is:

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

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)*:

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

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:

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

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

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2| \quad (7)$$

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 $1(\|\nabla f\|_2 \leq 1)$ 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: Comparison of Basic GAN and WGAN-GP discriminator loss function

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

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

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

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 includes 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 comp, angle of 3 comp, absolute of 6 comp, angle of 6 comp, absolute of 9 comp, angle of 9 comp and News. This project will do the multi-step ahead prediction, therefore, in the generator, we need to define the input step and the output step, and the input of the generator will be a three dimensional data, which are batch size, input-step and features, and the output will be batch size and output-step. For building up a generator with good performance, we use 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 which 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 will be the generated data from the generator. In this discriminator model, it includes three 1D Convolution layers with 32, 64, and 128 neurons separately, and 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 linear function will give a scalar score.

The Architecture of GAN

With the two above structures of our generator and discriminator, we combined these two models as our proposed GAN model.

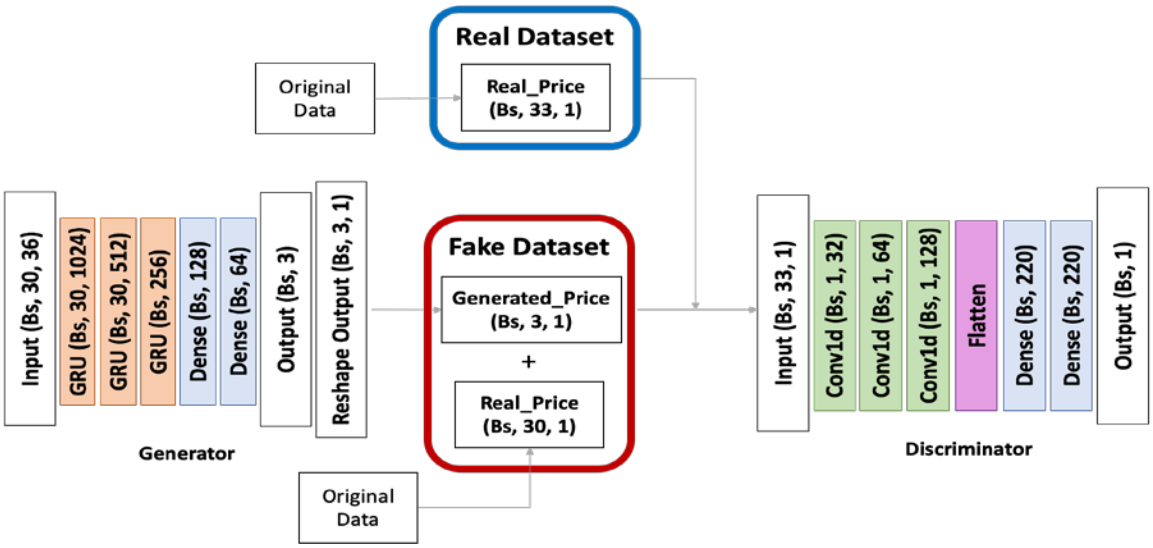

Fig. 1. GAN Architecture

In our GAN model structure, we use cross-entropy to calculate our loss for both generator and discriminator, we already defined the function in 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.

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 2497 observations and 36 variables in the dataset. The train data and test data are split into 7:3.

Feature Engineering

After downloading various asset historical data, we calculated some technical indicators and extracted some trend features. In addition, we use the NLP method to sentiment values of relevant news.

Technical indicators: we calculated 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. We 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.

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, which can help the GRU network pick its prediction trends more accurately *(Banushev 2020)*.

Data Structure

The method we prepared our dataset for supervised learning is to divide the dataset with the rolling window equals 1, the illustration has been shown in figure 2. The original dataset is 2 dimensional, we need to reshape the data to 3 dimensions according to the timesteps.

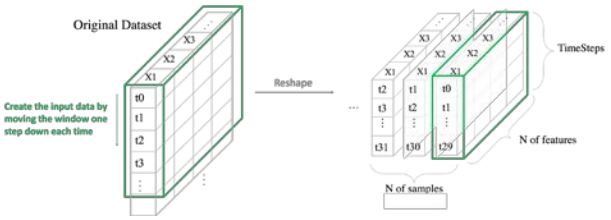

Fig. 2. Input data

Fig 3 illustrates the output we will get from the generator. Here, the number of output units equals 1. In our model, we can modify the timestep in figure 2 and output step in figure 3, in this paper, we built a many

to many models with timestep 30 and output step 3(use 30 days historical price to predict 3 days stock price).

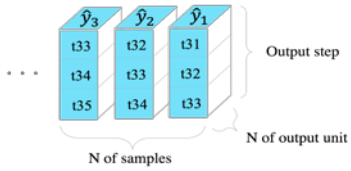

Fig. 3. Output data

Experimental Results

Train the model

The purpose of this paper is to do a prediction for 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 not only the historical closing price but 36 features that might have an effect on 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, we 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, we evaluated the performance of each model by Root Mean Square Error (RMSE), and the indicator is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \widehat{x_i})^2}{N}} \quad (8)$$

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

LSTM

In our LSTM model, we utilized Bidirectional LSTM in the first layer. The optimizer for our models in this work is Adam algorithm with a learning rate 0.001. The batch size is 64, and then we train 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 we split the data to the train set and test set, the testing dataset started on 07/21/2017.

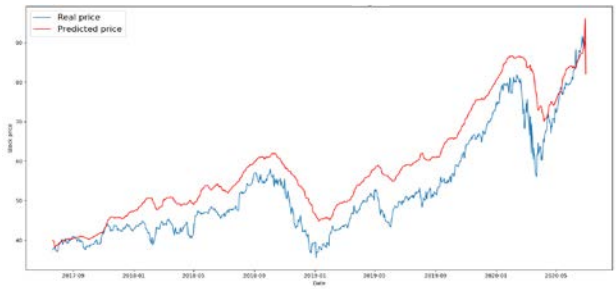

Fig. 4. LSTM test data plot

From Fig 4, we can see the result of LSTM which includes the forecasting of year 2020, the RMSE is 6.60, the blue line is the real stock price, and the red line indicates the predicted stock price. Obviously, all the predicted stock price is slightly higher than the real stock price till the end of May 2020. And after May 2020 the forecasting is much closer to the real stock price. Furthermore, we calculated the result excluding 2020, 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, we utilized 2 layers of GRU, and the optimizer for the GRU model is Adam algorithm with a learning rate 0.0001, and the size of batch is 128, and we train this model for 50 epochs.

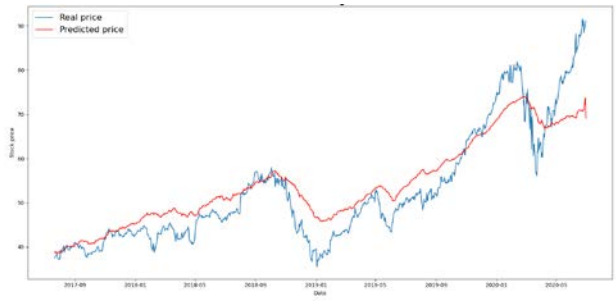

Fig. 5. GRU test data plot

Fig 5 shows the result of GRU including 2020, the RMSE is 5.33, and we can see the GRU model performs better than LSTM model before May 2020. From this figure we can observe the collapse of the forecasting after May 2020. Next, we calculated 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 Adam algorithm with a learning rate 0.00016. The size of batch is 128 and we train the model on this dataset for 165 epochs.

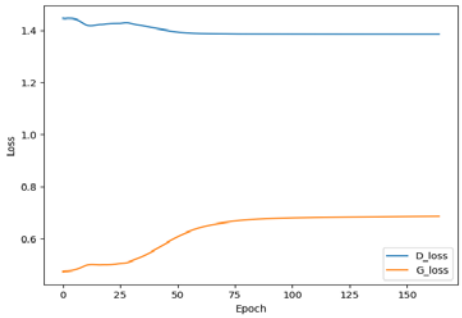

Fig. 6. Basic GAN loss plot

Fig 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.

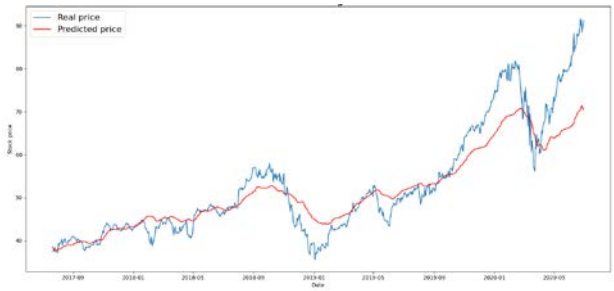

Fig. 7. Basic GAN test data plot

Fig 7 is the predicted result of the basic GAN model, and the RMSE is 5.36, from this figure, we can see the prediction started having a large gap between the real price and the predicted price in 2020 while there is a sudden surge in real 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 without unexpected events, the basic GAN for doing forecasting performs better than both two 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 Adam algorithm with a learning rate 0.0001. The size of batch is 128 and we train the model on this dataset for 100 epochs. And we train the discriminator once and the generator 3 times.

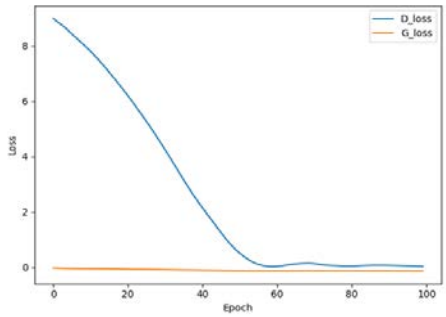

Fig. 8. WGAN-GP loss plot

Fig 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.

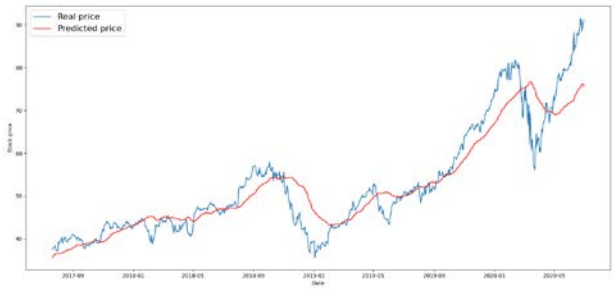

Fig. 9. WGAN-GP test data plot

Fig 9 is the predicted result of the WGAN-GP model, the RMSE is 4.77, which is best one in all the models. Similar with basic GAN in 2020 the prediction starts to have a large gap between the real price and the predicted due to the unexpected event COVID-19. When we remove the test data in year 2020, the RMSE of the forecasting decreases to 3.88, which preforms worse than the basic GAN model.

Evaluation

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

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

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

Conclusion

In this paper, we proposed a GAN which sets GRU as a generator and CNN as a discriminator. According to the experimental result, we have some conclusions. First, compared the GAN model with the traditional models, the GAN model can help to improve the GRU model and LSTM model, both basic GAN and WGAN-GP perform better than traditional 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 in normal periods, basic GAN performs better. However, to our knowledge, a GAN model including RNN is unstable, it is very difficult for these models to tune hyperparameters, without good 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, in each layer and for the whole model, can be tuned more accurately, we believe the result would have been significantly improved.

There are many research teams proposed methods of reinforcement learning for hyperparameter optimization, such as Rainbow *(Matteo et al. 2017)* based on Q-learning and Proximal Policy Optimization (PPO) *(John 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.

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

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

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

Ricardo, A., Carrillo, R.(2019). Generative Adversarial Network for Stock Market price Prediction. Stanford University. United States. http://cs230.stanford.edu/past-projects/#fall-2019

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.

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

Hochreiter, Sepp. "JA1 4 rgen Schmidhuber (1997).“Long Short-Term Memory”." Neural Computation 9, no. 8.

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. DOI: 10.1155/2018/4907423 https://www.researchgate.net/publication/324541558_Stock_Market_Prediction_on_High-Frequency_Data_Using_Generative_Adversarial_Nets

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

Hessel, Matteo, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. "Rainbow: Combining improvements in deep

reinforcement learning." arXiv preprint arXiv:1710.02298 (2017).

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

Jonathan, H., (2020). GAN — Spectral Normalization. https://jonathan-hui.medium.com/gan-spectral-normalization-893b6a4e8f53

Jonathan, H., (2020). GAN — Wasserstein GAN & WGAN-GP. https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490