# STOCK MARKET SIMULATION AND PREDICTION USING GENERATIVE ADVERSARIAL NETWORKS

by

Lin He

Signature Work Product, in partial fulfillment of the
Duke Kunshan University Undergraduate Degree Program

*March 17, 2023*

Signature Work Program
Duke Kunshan University

**APPROVALS**

Mentor: Xing Shi Cai, Division of Natural and Applied Sciences

Marcia B. France, Dean of Undergraduate Studies

# CONTENTS

# ABSTRACT

Applying Generative Adversarial Network (GAN) models in different fields to generate artificial data has been a heated topic lately. In this project, we explore GAN's potential in simulating and predicting stock market time series data. We apply the GAN model with Gated Recurrent Unit (GRU) as the generator, Convolutional Neural Network (CNN) as the discriminator to predict future prices, and the GAN model with Temporal Convolutional Network (TCN) structures as both the generator and the discriminator to simulate historical prices. Our study shows that the GAN model has the ability to approximate the distribution of historical returns and the potential to perform better than the moving average model in stock price prediction. Extensive experiments show that the model trained on one stock can be used to predict the prices of other stocks, even when the companies are in different businesses. Meanwhile, our study also shows that the GAN model does not capture autocorrelations well when simulating stock prices and is very unstable when making future predictions.

# ACKNOWLEDGEMENTS

This endeavor would not have been possible without the guidance of Professor Xing Shi Cai. I am also grateful for all the professors who have taught me and my peers who have given me massive support.

# LIST OF FIGURES

# LIST OF TABLES

Chapter 1

# INTRODUCTION

Using machine learning models to generate artificial data has gained huge popularity lately. The generated data can be a creative product, such as music pieces[6], drawings[37], and poems[48]. The generated data can also be used for data augmentation by producing samples to be added to training sets to enhance performance. The good performance of these generative models not only raises interest in the academic field, but also draws attention from the public. Sequoia Capital, a major venture capital form, predicts that generative Artificial Intelligence (AI) will be a world changer and that it will create trillions of dollars of value[1]. Numerous projects based on generative models has emerged such as the text-to-image generator Stable Diffusion[38] and the text-to-text generator called ChatGPT[34]. Figure 1.1 shows a drawing generated with Stable Diffusion.

Figure 1.1: We thought AI will take over the boring jobs

A generative model is a type of AI that can generate new content based on previous seen data. Many recent generators are built on neural networks, an algorithm designed to capture the relationships and meanings of data by mimicking the brain's activity. Just like humans learn how to write by repeated exercise, neural networks learn to perform a specific type of task by training on the relevant data set. One advantage of neural networks over conventional algorithms is that it has the ability to perform many tasks simultaneously, which greatly increases the processing

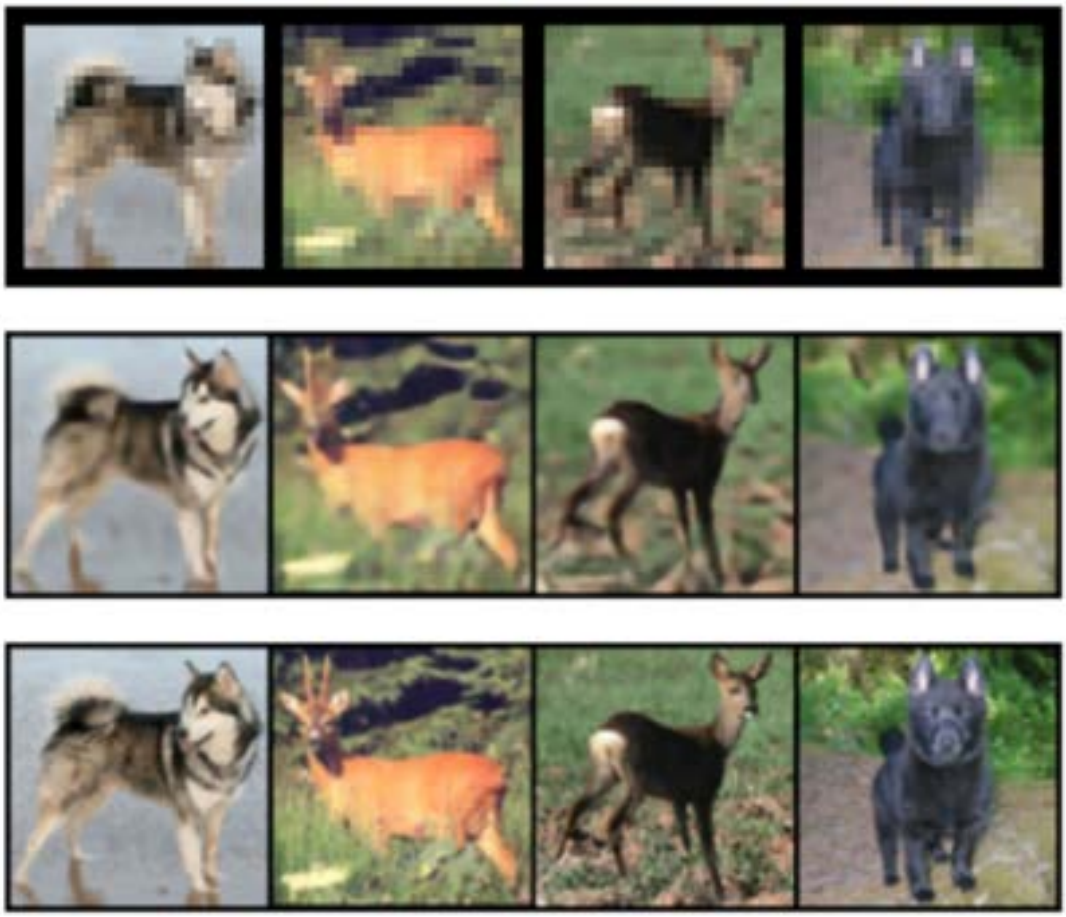


Figure 1.2: Improve the resolution of images using GAN[24]

speed. Additionally, a neural network can capture nonlinear patterns in the given data sets and the interactions between the data, which allows it to handle complex tasks. Because of these advantages, generative neural networks are popular, and many variations of generative neural network models have been invented to perform a wide range of tasks.

One influential model in this field is the generative adversarial network (GAN) introduced by Goodfellow et al.[19]. GAN consists of two neural networks trying to outplay each other. The first network is the generative network, which is trained to generate fake samples. The second network is the discriminative network, which is trained to discriminate real samples from the fake samples created by the generator. The discriminator and the generator are trained simultaneously and the generator is pushed to output more realistic images to fool the discriminator.

GAN is a groundbreaking model originally designed to generate realistic images, but its success has attracted a large amount of research to broaden its applications and to improve its performance. For instance, GAN can be used to generate the crop-out part of an image[13], improve the resolution of a low quality picture[24] (Figure 1.2) and turn a landscape photo into a Monet style painting[50]. While most of the research focuses on leveraging its capacity to create new images, a few studies also investigate its capacity to generate time series data. Kumar et al. [23] introduced WaveGAN to generate audio waveforms. Hartmann, Schirrmeister, and Ball [20] adapted GAN to synthesize Electroencephalography (EEG) time series for brain activity signals. GAN can also generate missing data[30] and detect anomalies[25] in multivariate time series. These studies all showed promising results that favoured the possibility of generating time series data using GAN.

Apart from the applications mentioned above, another area that attracts research interest is the financial time series data, especially the stock price data.

There are two significant subfields in generating stock price data: one is to generate the future price to forecast profits and risks, and the other is to generate the historical data to investigate the capacity of GAN to capture the performance features of a stock. Representative research on predicting future stock prices using GAN was conducted by Zhou et al. [49]. The research used basic technical index data like trading volume and the prices of the previous few minutes as input data for the generator to generate the next minute's stock price. The generated price was compared to the real data, and the results showed that their adapted GAN largely outperformed the baseline models in Direction Prediction Accuracy (DPA) and Root Mean Square Relative Error (RMSRE). On the other hand, Wiese et al. investigated the performance of GAN in mimicking the distributional properties and dependence properties of the returns on stock[46]. The authors used random numbers as input and trained the generator to output log-returns of a stock. Their results showed that the GAN can be adapted to reproduce the distribution of the log-returns satisfyingly.

Since the financial market is a complex, chaotic, noisy, non-stationary, and dynamic system that does not follow random walks[28], it is hard to predict the future prices or recreate historical prices. Traditionally, Autoregressive Integrated Moving Average (ARIMA) generalized by Box et al. [5] is an effective method widely used in generating financial time series. In recent years, machine learning techniques such as Support Vector Machine (SVM) have shown great performance in predicting typically nonlinear, non-stationary financial time series data[41]. With the development of deep learning, techniques such as Long Short-Term Memory and Convolutional Neural Networks (CNNs) have gained more and more interest [42]. Both research on financial time series using GAN above show promising results in generating valuable stock data, which indicates the possibility of adapting GAN to assist stock analysis as an alternative approach apart from the popular financial approaches. Yet, it is important to test the robustness of GAN model in generating realistic stock data. Testing and repeating the experiments is essential for scientific discovery to be accepted as reliable and valid. It is by the repetitions that the possibility of the discovery being a mere coincidence can be ruled out[36].

In reality, reproducing experiments is not easy in economics and computer science. A study conducted by Chang and Li found that less than half of the 67 economic papers published in renowned economic journals can be replicated even with the help of the authors[8]. In machine learning research, even though most parts of the experiment is done by computers and most databases are open source, the difficulty in reproducing consistent results might still occur due to reasons such as disregarding overfitting, improper statistic analysis, and selective results[35]. In extreme cases, such as[9], possible fraudulent data has been found in AI research papers. For the advancement of science, many researchers bring up suggestions for good practices to handle the reproducibility problem. Drummond argues that reproducing the results exactly in the machine learning field is a waste of effort for the community and reproducing the experiment with some changes is a better practice because it is the idea of the experiment that is important[15]. Another problem in reproducibility is the confusion about the precise definition of reproduction and when a robustness test can be considered a replication[11]. To avoid misunderstanding the terminology, this report adopts the classification of replication tests in economics established by Clemens — a *reanalysis test* means modifying the code, and an *extension test* means using new data[11]. Both reanalysis and extension are classified under the robustness test subfield, which focuses on replicating the conceptual idea in contrast to replication tests that require exact results[11].

In this signature project, we tested the performance of GAN in generating historical stock prices and future stock prices. On the one hand, we conducted a combination of reanalysis and

extension of the work of Zhou et al. [49] by adapting their framework with some modifications to predict the Nasdaq stock prices of two tech companies and two companies in the traditional industries. The original work focused only on the stock market in China. Thus, by studying the stock prices from different markets and industries, we tested whether the GAN model is robust in the stock prediction task.

On the other hand, we did an extension on the work of Wiese et al. [46]. The authors used GAN model to approximate S&P 500 returns from 2011 to 2020. In this study, we generated the stock return of a different period, from year 2020 to year 2022, using the same model introduced by Wiese et al. [46]. By doing so, we tested the robustness of GAN in generating historical data under different historical backgrounds. The choice of doing the robustness test instead of the exact replication is due to our interest in testing the practicability of GAN in studying financial time series.

## Chapter 2

# MATERIAL AND METHODS

### 2.0.1 Background Knowledge on Neural Network and GAN

#### Multilayer Perceptrons

The idea of perceptron, which is inspired by brain cells in neural system, lies at the core of neural network models. The mathematical model of the artificial neuron was first introduced by McCulloch and Pitts [32], inspired by the nervous activity. Rosenblatt [39], a psychologist, then improved the model and came up with the concept of perception, a threshold function that maps its input vector $\mathbf{x}$ to a binary output:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

$\mathbf{w}$ represents the weights and b represents the noises. By updating weights and noises to make the output of the model approach the desirable output, the model can learn the right weights and noises from the data to perform tasks that give desirable outcomes.

The idea of using a threshold function is inspired by the mechanism of connectivity between neurons — a neuron only gets activated and passes down signals to other neurons when the signals it received has reached the threshold. However, since the step function gives a strict threshold, it is less flexible and does not do well if we want a more complicated result. Thus, more types of functions, called *activation functions,* are introduced into perceptrons. The common activation functions include Sigmoid, Tanh, rectified linear, etc. The Sigmoid function is one of the most popular nonlinear functions that gives an output between 0 and 1 for any real number input; Tanh is also a popular nonlinear function that gives an output between –1 and 1; Rectified linear is a linear function if the input is positive, but it returns 0 if the input is negative[14]. These activation functions have the extra benefit of being differentiable, which allows the weights and biases to be updated using gradient descent.

When multiple perceptrons are used to form a feedforward network with more than one layer, a Multilayer Perceptrons (MLPs) model is built. The layers between the input and output are called *hidden layers*. In a feedforward network, the inputs are fed into the perceptrons of the first hidden layer, and the outputs are fed into the following hidden layers. This process continues until the output layer is reached (see Figure 2.1). Forming a neuron network gives the model more expressive power. Hornik, Stinchcombe, and White [21] proved the Universal Approximation Theorem which showed that a neural network with as few as one hidden layer can approximate any continuous function. However, this layer must contain a great number of perceptrons if the function is complex. Eldan and Shamir [16] and Telgarsky [44] proved that

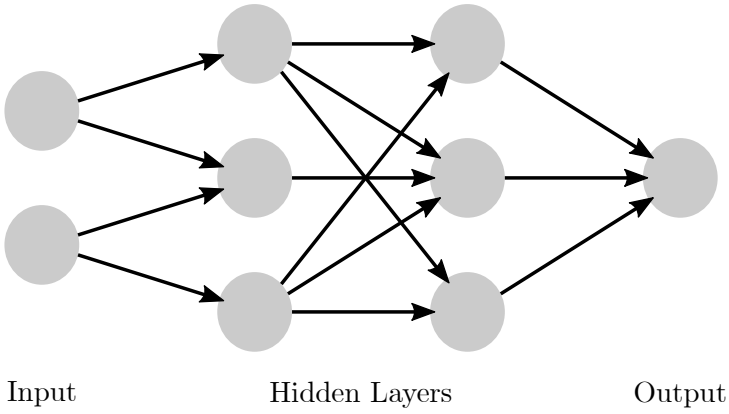



Figure 2.1: Basic Multilayer Perceptrons (MLP) structure[17]

reducing layers may require exponentially more perceptrons to compensate. Adding layers enable the model to handle non-linear functions with fewer perceptrons, therefore making the learning process more efficient. In addition to designing layer wisely, choosing the number of nodes per layer is also important. Lu et al. [29] showed that when the number of perceptrons per layer is restricted, there exist networks that cannot be realized unless the number of layers is substantially large. In essence, a suitable architecture is the key to leveraging the power of the neural network.

#### GAN models

*Minimax* is a turn-based game where each player takes turns to choose the strategy that minimizes their maximum loss. GAN is a minimax game, in which the generative network and the discriminative network compete against each other. The generator tries to minimize the possibility of getting its products recognized by the discriminator, and the discriminator tries to minimize the possibility of making discrimination mistakes. They are trained simultaneously and are updated according to their own loss function. In other words, by competing against each other, both networks adjust their weights to reduce their losses. Different loss functions and network structures have been developed to improve the performance of GAN.

In this project, we use loss functions based on cross-entropy which measures the performance of a discriminator differentiating two classes. When the discriminator is confident in the decision and the decision is correct, the cross-entropy loss is low. When the discriminator gives a wrong classification or gives a right classification with low confidence, the cross-entropy loss is high.

More specifically, we will use the following loss function — Let $Y^i$ be a real label (classification) of an object to be classified and $\tilde{Y}^i$ be a prediction by the discriminator, binary cross-entropy loss function can be defined as:

$$BCE = -\frac{1}{m}\sum_{i=1}^{m}(Y^i \log(\tilde{Y}^i) + (1 - Y^i)\log(1 - \tilde{Y}^i))$$

where $m$ is the number of predication made by the discriminator.

The discriminator in GAN needs to output a score between 0 to 1 to indicate how realistic (in the sense that it is not produced by the generative network) the input it receives. A score close to 1 means that the discriminator believes the input is realistic.

The generator in GAN generates data that aims to fool the discriminator. Let $x^i$ be the input data for the generator, $y^i$ be the target from the real data set, $D$ be the function that maps input

samples to the possibility of the samples being real, and $G(x^i)$ be the synthetic target generated by the generator. The discriminator needs to minimize its loss function —

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

The generator in GAN generates data that aims to fool the discriminator. In other words, generator aims to maximize the binary cross-entropy loss of the discriminator regarding synthetic targets. The loss function of the generator is therefore defined as —

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

### 2.0.2 Generating Historic Return

#### Stylized Facts and Evaluation Metrics

Since stock market is a chaotic system that is impacted by countless of elements, the returns in the past do not necessarily predicate the returns in the future. However, if there are no stable statistical properties, it is pointless to do any statistical analysis[12]. Lin et al. [27] introduced the concept of 'stylized facts' to summarize the statistical facts that characterized economic growthwhich can be used as the base points for economic theoretical modeling. Cont [12] later conducted empirical research on financial market and summarized a list of stylized facts that are common across time periods and a side set of financial assets. The author suggested that these stylized facts could be seen as constraints for a stochastic process to reproduce returns with more realistic statistical properties.

Due to the significance of stylized facts in reproducing the stock returns, Wiese et al. [46] adopted some of them as evaluation metrics for their QuantGAN model. In this signature project, we employed similar evaluation metrics mentioned as Wiese et al. [46], which included Leverage Effect, Earth Mover Distance (EMD) and Auto Correlation Function (ACF) Score.

**Earth Mover Distance** EMD is a measure of the distance between two distributions first introduced by Rubner, Tomasi, and Guibas [40]. It can be seen as the minimum amount of work transfer distribution $X$ distribution $Y$. We used EMD to measure the overall similarity between our generative series and the real return series. Let $\Pi(X, Y)$ be the joint distribution between X and Y, $\|x - y\|_2$ be the $L_2$ norm between $x$ and $y$, the Earth Mover Distance can be written as:

$$EMD(X, Y) = \inf_{\gamma\in\Pi(X,Y)} \mathbb{E}_{(x,y)\sim\gamma}[\,\|x - y\|_2]$$

$L_2$ norm is defined as the square root of the sum of the squares of the values in each dimension.

EMD not only reflects if the generative series captures the mean and variance of the return series, but also indicates if the tails of the generative distribution matches the tail distribution of return distribution. One of the stylized facts mentioned by Cont [12] is that the distribution of return displays a finite but long tail.

**Auto Correlation Function Score** The Auto Correlation Function (ACF) is a widely used method to measure how data points are correlated with the historical data points in a time series. Let $r_{1:T}$ denote the real log-returns from time 1 to time $T$ and $\{\tilde{r}^1_{1:T}, \ldots, \tilde{r}^m_{1:T}\}$ be a set of generated log-returns data. The ACF is a function of time lag $k$ and the log-returns, which measures on average how does the return correlated to the return with time lag $k$ as follows —

$$C(k; r) = Corr(r_{t+k}, r_t)$$

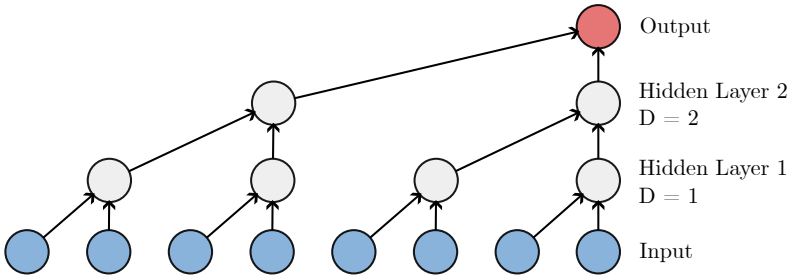



Figure 2.2: Basic TCN structure[33]

The ACF score calculates how much the average ACF of the generative returns differs from the ACF of the real returns —

$$ACF(k) = \left\| C(k,r) - \frac{1}{m}\sum_{i=1}^{m} C(k,\tilde{r}) \right\|_2$$

Cont [12] pointed out that the ACF of absolute returns decays slowly as the time lag increases and this pattern is sometimes interpreted as a sign of long term dependency. Thus, it is meaningful to test if the ACF of the generative returns also show similar patterns.

However, Cont's goal is not having a model that generates the exact ACF is not the goal because the linear autocorrelations are usually insignificant in asset returns, unless the time scales are small (around 20 minutes)[12]. Due to the absence of significant autocorrelations in large time scales, we only focused on the ACF score within a small time scale, and the decay pattern of autocorrelation were compared by visualizing the real and generative ACFs.

It is worth pointing out that the insignificance of correlations does not mean the return fluctuations are identically distributed. Mandelbrot [31] observed that large changes in stock prices tend to happen in a short period of time, which the they call *volatility clustering*. The absolute returns display a positive and slowly decaying ACF —

$$Corr(r_t, r_{t+k}) > 0$$

for a period of time ranging from a few seconds to a few weeks.

**Leverage Effect** Black [4] are among the first groups of people who recorded that the volatility of stock prices tends to be higher in declining stock markets than in rising stock markets. They also suggested an explanation of this volatility asymmetry based on leverage effect — the decrease of stock value increases the debt-to-equity ratio, which increases the risks and volatility. There are many existing ways to estimate the leverage effect, each with its advantages and disadvantages[45]. For simplicity, we used the same leverage effect estimator as the one defined in the paper of Wiese et al. [46]. The estimator is defined as the correlation between the squared, lag log-return and the log-return:

$$L(k;r) = Corr(r_{t+k}^2, r_t)$$

To compare how well the generative model captures the real leverage effect, we computed the leverage effect (EF) score similarly as the ACF score:

$$EF(k) = \left\| L(k,r) - \frac{1}{m}\sum_{i=1}^{m} L(k,\tilde{r}) \right\|_2$$

#### Temporal Convolutional Networks

TCNs are adapted from Multilayer Perceptronss (MLPs) and have recently been shown to perform well on many sequence modeling tasks[2]. In particular, the TCNs capture the long-term dependencies better than other popular recurrent networks[18] such as Long Short-Term Memory (LSTM). Since long-range dependencies and volatility clustering are presented in the market, TCN can be effective in stock return modeling. Wiese et al. [46] constructed and tested a GAN model with both its generator and discriminator being TCN. We have adopted the same approach to generate historical price in this signature work.

The construction key of TCNs is to make the outputs of neurons only depend on past sequences. Figure 2.2 depicts a TCN with a kernel size of $K = 2$ and a dilation factor of 2. We input a series of log-returns from day 1 to day 8 to the TCN described above. The kernel size refers to the number of inputs each neuron will receive, so $K = 2$ means each neuron will receive the inputs from 2 neurons in the previous layer A dilation factor of 2 indicates that the number of neurons in every layer decreases by a factor of 2.

We adapted the code by Icascha's replication experiment in Github repository[22] to construct the GAN for historical price simulation. The TCN has 6 hidden layers, with kernel size $K = 1$ and a dilation factor of 2. Both the generator and the discriminator of our GAN adopted this TCN structure.

#### Data Samples and Preprocessing

Our raw data were downloaded from Yahoo! Finance.[47] Before passing a financial time series to the discriminator, it needs to be preprocessed which includes three steps.

First, calculate the log-return series —

$$r_t = \log\left(\frac{s_t}{s_{t-1}}\right)$$

Second, normalize the data so that we obtain a series with zero mean and unit variance. Third, apply inverse Lambert W to gaussianized the data. In the real market, the log returns do not follow a normal distribution; instead, they usually show heavy-tails[7]. To model the behavior more correctly, Wiese et al. [46] suggested that Lambert W transformation on normalized data helps. Fourth, apply a rolling window to slice up the series. More specifically, we let the length of the time series $x$ be $N$, and the number of segments to be created be $k$. The rolling window creates a group of $k$ series $s_i = x_i, \dots, x_{N-k+i}$.

### 2.0.3 Predicting Future Price

#### Gated Recurrent Unit

We have built the GAN model with a Gated Recurrent Unit (GRU) as the generator and a Convolutional Neural Network (CNN) as the discriminator. GRU is a network introduced by Cho et al. [10] to handle memory in the time series data. The smallest unit in the GRU network consists of a reset gate and an update gate (Figure 2.3). The update gate and the reset gate, though are of different functions, both receive new input and the memory to determine how much information is needed to be stored or neglected in the memory. Many such units can be concatenated together to form a layer, and many such layers can be stacked to create a GRU network.

#### Data Samples and Preprocessing

The data were downloaded from Yahoo! Finance[47]. We selected two stocks, Apple and Microsoft, from tech industry, and two stocks, Canadian National Railway and Exxon. from traditional

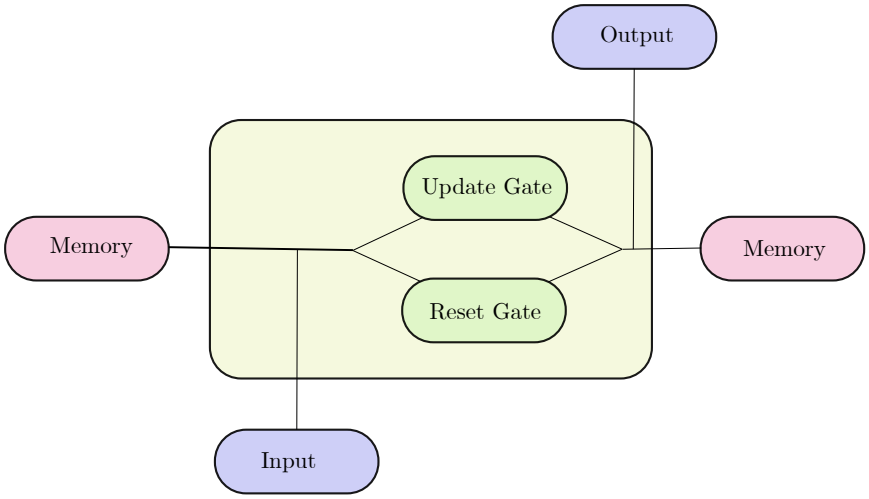



Figure 2.3: The basic unit of Gated Recurrent Unit

industries. The data contains the prices of these four stocks from the beginning of 2016 to the beginning of 2020, before the COVID pandemic hit. It includes the opening price, closing price, highest price of the day, lowest price of the day and trading volume.

The Fourier transform takes a function and decomposes the function into several sine waves whose weighted sum can approximate the original function . Since Fourier transform can help GRU networks to pick up trends more accurately[3], we applied Fourier transform to our normalized closing price series. See Figure 2.4. We used the transforms with 3, 6, and 9 components and inferred the transform with 3 components as the long-term trend, as suggested by Banushev [3].

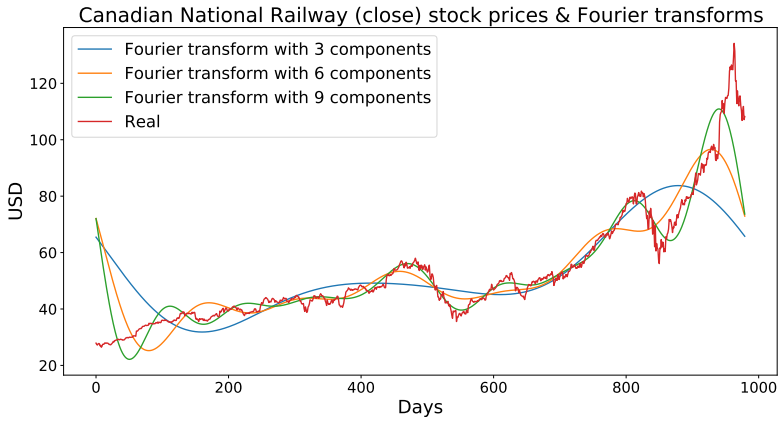



Figure 2.4: Fourier Transform of Canadian National Railway prices

We constructed the input data for the generator using the rolling window. The input data includes the basic features scarped directly from Yahoo Finance! as well as the Fourier transform closing price with 3, 6, and 9 components. The basic features include open price, high price, low price, closing price, and the volume traded of a trading day. The output is a series of predicted closing price within a period following the input. Let the time step be $k$ and the output step be $n$. The former represents the number of consecutive days of data that is used to construct the input, while the latter represents the number of consecutive days of predicted closing prices. The rolling window moves $n$ days forward in the whole data set each time and separates the whole data set into many smaller fractions (Figure 2.5) The output data of generator should be

processed before feeding into the discriminator.

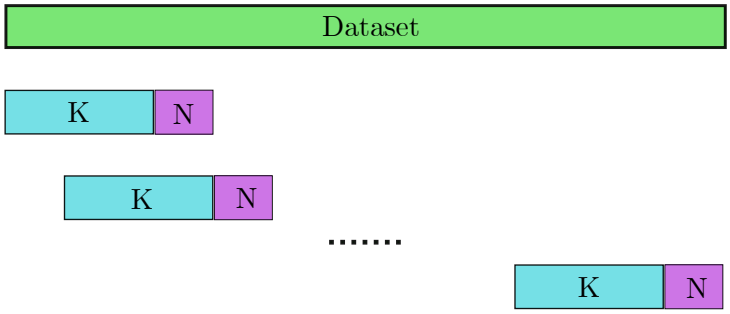



Figure 2.5: Separate data set into small fractions using rolling window

We need to construct both the real data and the synthetic data. The real data is simply the closing prices in the $k + n$ consecutive days. The synthetic data is the combination of the $k$ consecutive real closing prices concatenated with the $n$ predicted closing prices. See Figure 2.6. We adapted the code by Huangchun Lin in Github repository[26] to construct the GAN for future price simulation.

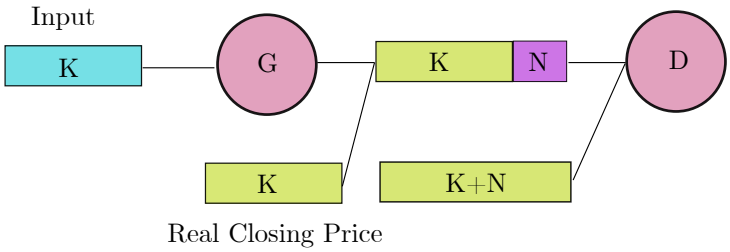



Figure 2.6: GAN architecture

### Baseline Model

The application of machine learning methods to finance time series forecasting has been widely studied[43]. Our aim is not to compete with existing methods, but rather to demonstrate the potential of GANs in this field. Moreover, most of these models are too complicated to implement within the limited time frame of our project. Therefore, we have chosen to use the simple model which we call moving average as our baseline for comparison. More specifically, we used a rolling window with length $k$ on the real closing prices and assumed that the closing prices of the following $n$ days are the average price of the previous $k$ days. The window is moved forward $n$ steps each time on the real data. When the window is moved, a new prediction of $n$ days will be attached to the previous prediction series. The prediction is only based on real data, so the rolling window's job is to attach all the $n$-day predictions into a series.

## Chapter 3

# RESULTS

### 3.0.1 Predicting Future Price

To test the performance of the GAN model in predicting future stock closing prices, Root Mean Square Error (RMSE) between the real closing price and the prediction is used as a criterion. First, a GAN model is trained using the stock data from 2017 to 2019. Then the model is tested on the same stock using data from 2019 to 2020. Note that the stock closing prices are normalized independently. Thus, we can only compare the model's performance within each individual stock.

We first tested the performance of the baseline model using a 3-day average to predict the next day. Figure 3.1 shows the prediction of Apple stock. The predictions follow the real trend well with a slight delay, and the RMSE between the prediction and the real closing price is 1.53. Since 3 days are too short for GAN to capture any long term pattern, and a prediction of one day does not leverage GAN's strength in synthesizing data, we decided to use 30-day data to predict the next 10 days in both the baseline model and GAN.

To check if it is possible to use a model trained on one stock to predict the closing prices of other stocks, we selected the model with relatively better performance to do the test.

We observed that the one trained with Apple stock (AAPL 60) has an Root Mean Square Error (RMSE) 20% smaller than when it is trained with 90 epochs, so we chose AAPL 60 as a representative for the model trained with 60 epochs. We observed that the one trained with Canadian Railway stock (CNI 90) has an Root Mean Square Error (RMSE) 25% smaller than when trained with 60 epochs, so we chose CNI 90 as a representative for the model trained with 90 epochs.

Table 3.1 shows the training results of the models trained with these four stocks, using 60 and 90 epochs. Table 3.2 shows the test results of the four stocks using different models.

Table 3.1: RMSE of the training results of the four stocks with different epochs

| Model | AAPL | MSFT | CNI | XOM |
|---|---|---|---|---|
| 60 epochs | 2.651 | 4.30 | 2.657 | 3.60 |
| 90 epochs | 6.02 | 8.44 | 2.656 | 17.32 |

#### Apple Stock

In the baseline model, the average closing price of the previous 30 days is used as the prediction of the closing price on the following 10 days (Figure 3.2). Compare to the 3-day average model

Table 3.2: RMSE of the test results of the four stocks using different models

| Model | AAPL | MSFT | CNI | XOM |
|---|---|---|---|---|
| Baseline | 8.35 | 11.37 | 5.50 | 5.84 |
| 60 epochs same stock | 7.64 | 10.02 | 8.45 | 7.656 |
| 90 epochs same stock | 9.51 | 10.653 | 6.32 | 8.10 |
| 60 epochs AAPL | 7.64 | 6.82 | 5.02 | 4.92 |
| 90 epochs CNI | 12.656 | 16.656 | 6.32 | 8.60 |

(Figure 3.1), the delay of prediction became more evident and the RMSE raised to 5.42. The error increased to 8.35 as we zoomed in to the 2019-2020 period (Figure 3.3).

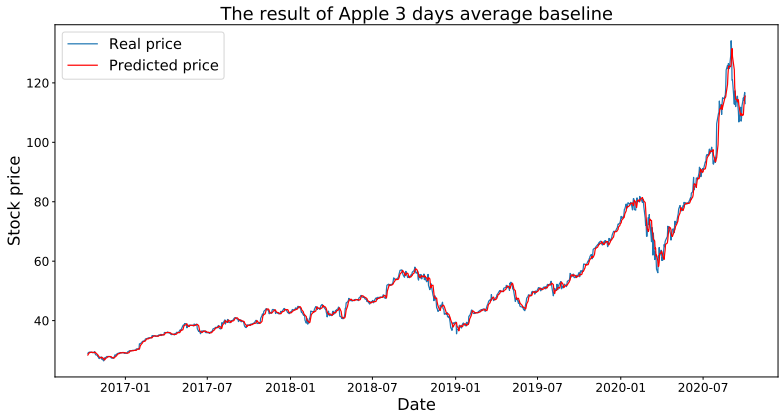



Figure 3.1: Baseline prediction of Apple using 3 days average

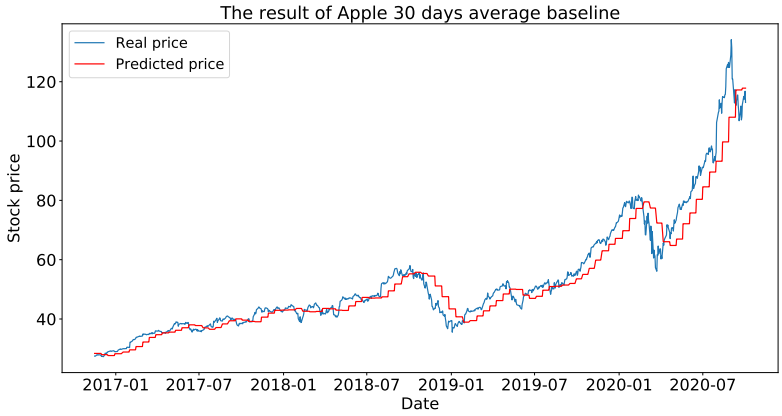



Figure 3.2: Baseline prediction of Apple using 30 days average

As for the GAN model, we trained the model using 90 epochs and saved the model's weights when it finished 60 epochs and 90 epochs, so gave us two models to test. The model with 60 epochs (AAPL 60, Figure 3.4) has a better performance fitting the real training data than the model trained with 90 epochs (AAPL 90, Figure 3.5). AAPL 60 (Figure 3.6) also predicts the test data with smaller error (Figure 3.7). The test results also suggest that AAPL 60 generated stock price trends which are less volatile and follow the real trend better, especially during August 2020, when the price increased suddenly.

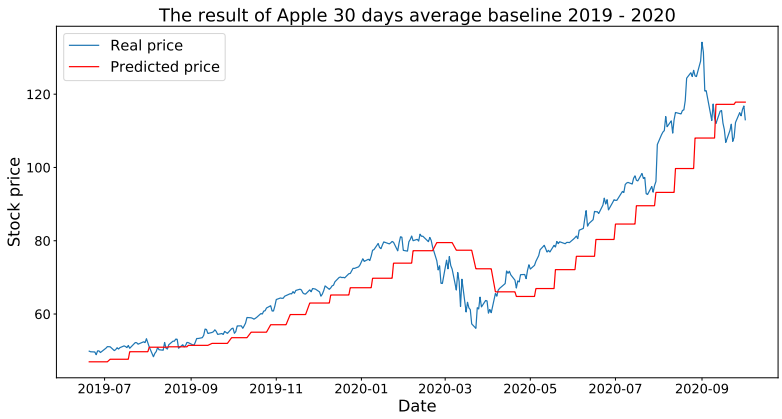



Figure 3.3: Baseline prediction of Apple using 30 days average

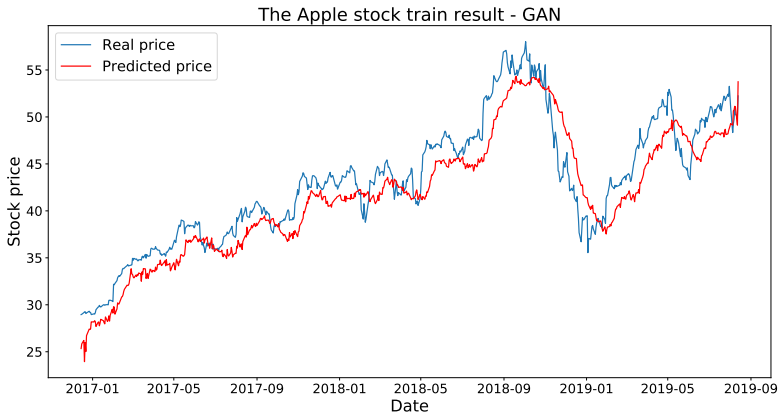



Figure 3.4: Training result of Apple using 60 epochs

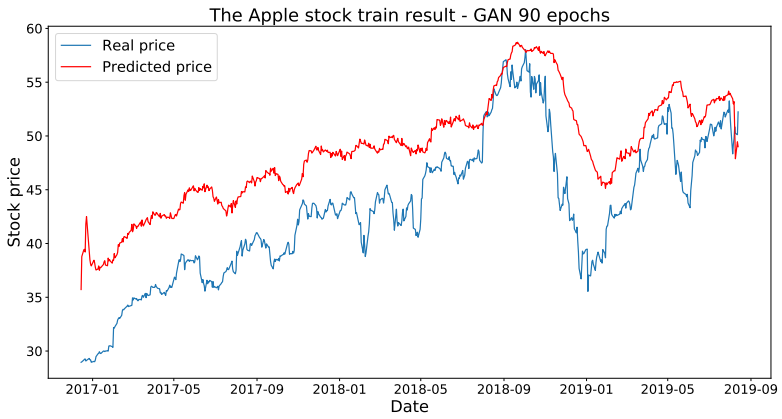



Figure 3.5: Training result of Apple using 90 epochs

### Microsoft Stock

The 30-day-average baseline model does not predict the Microsoft stock price between 2019 and 2020 well. There was a sharp increase and a sharp decrease of the stock price from February 2020 to April 2020, but the baseline model reacted to the change slowly and failed to capture the real trend, causing a high RMSE of 11.37 (Figure 3.8).

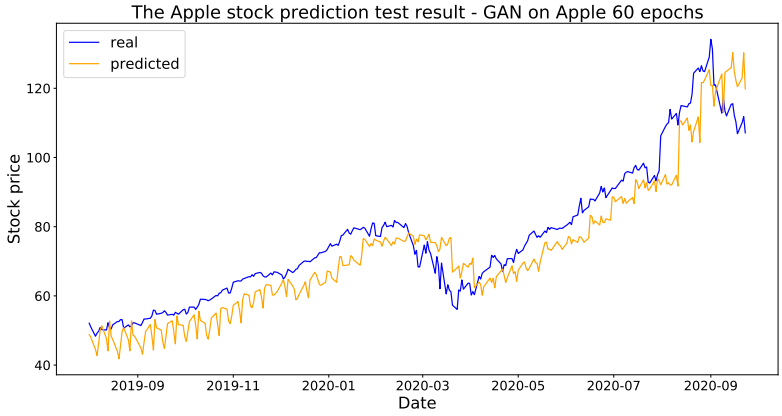



Figure 3.6: Test result of Apple using AAPL 60

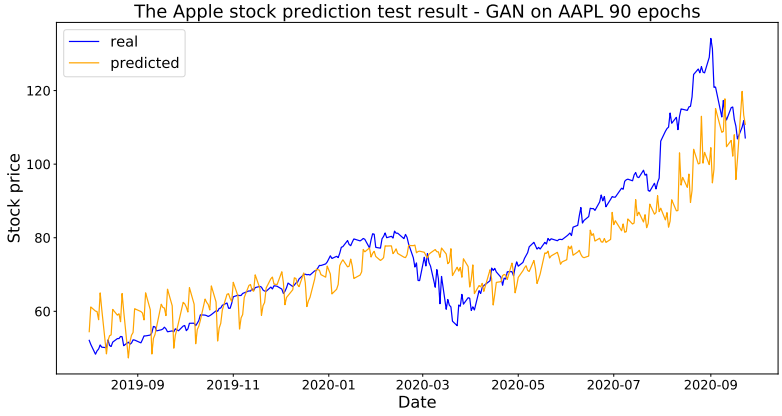



Figure 3.7: Test result of Apple using AAPL 90

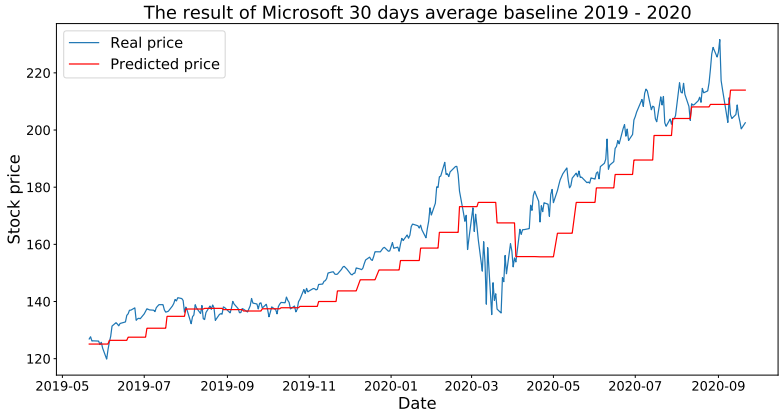



Figure 3.8: Baseline prediction of Microsoft using 30 days average

In terms of the GAN model, the model that was trained with 90 epochs (MSFT 90) fit the train data better than the model trained with 60 epochs (MSFT 60). MSFT 60 (Figure 3.9) predicted a trend line with similar shape as the real trend, but the predicted trend line was around 10 dollars lower than the real trend. Such situation improved when the model was trained with 90 epochs, especially for the section between January 2018 and September 2018 (Figure 3.10). However, neither MSFT 90 (Figure 3.12) nor MSFT 60 (Figure 3.11) performed well when predicting the test stock price. It is apparent from the figures that the predicted trends were very volatile, and

oscillates drastically without observable patterns, especially after May 2019. These oscillations resulted in high errors, and both models had their test errors over 10. The situation improved when AAPL 60 models was used (Figure 3.13).

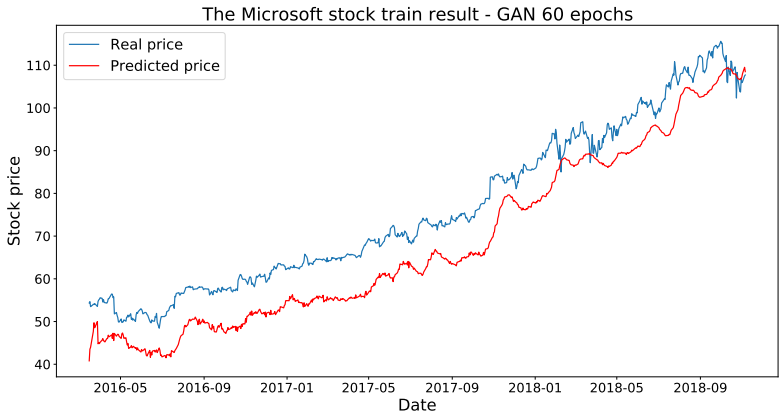



Figure 3.9: Training result of Microsoft using 60 epochs

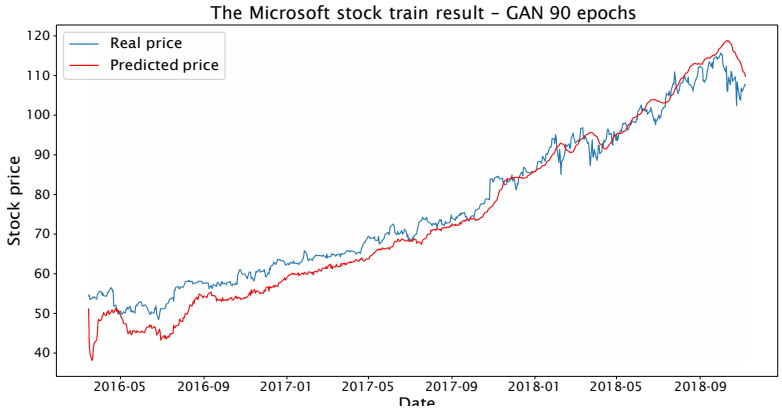



Figure 3.10: Training result of Microsoft using 90 epochs

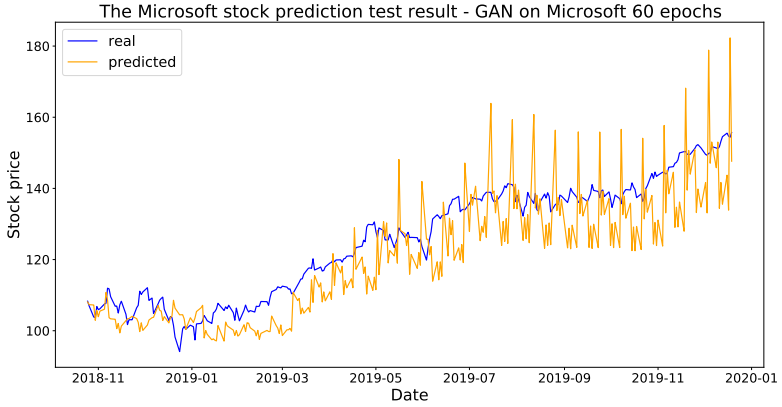



Figure 3.11: Test result of Microsoft using GAN model trained on Microsoft with 60 epochs

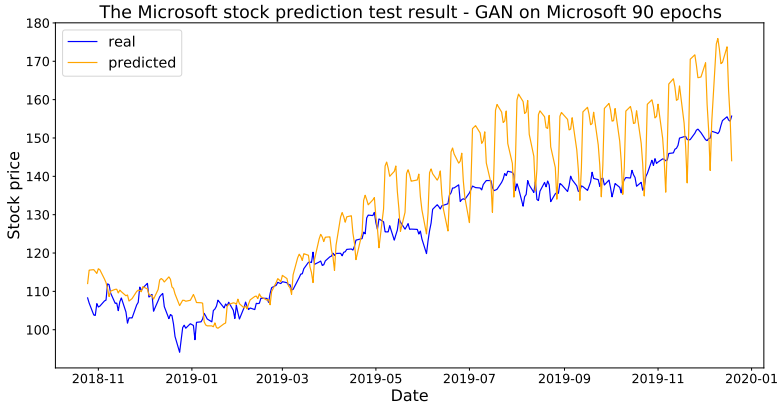



Figure 3.12: Test result of Microsoft using GAN model trained on Microsoft with 90 epochs

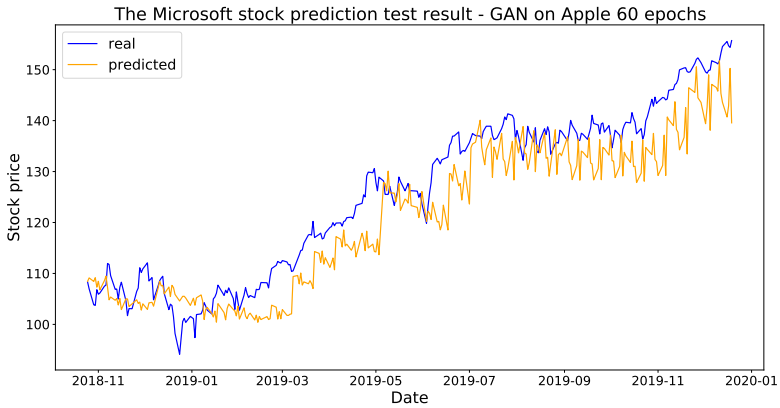



Figure 3.13: Test result of Microsoft using GAN model trained on Apple with 60 epochs

### Canadian National Railway

Compared with Apple and Microsoft, the Canadian National Railway (CNI), a company from the traditional industry, has a more stable stock price. The gap between the highest price and the lowest price of CNI is around 40, while the gaps in Microsoft and Apple are both around 100. Since the absolute change in value is not big, although the baseline model does not capture the drastic drop in price too well, the error is small (Figure 3.14).

The test results of GAN model on the other hand are not as good. Although the CNI 60 model and CNI 90 model fit the training data well (Figure 3.15, 3.16), they did a poor job predicting the test data. The correlation between the real trend and the prediction is weak, and the oscillation problem is obvious (Figure 3.17, 3.18). The problem is mitigated when APPL 60 is used (Figure 3.19).

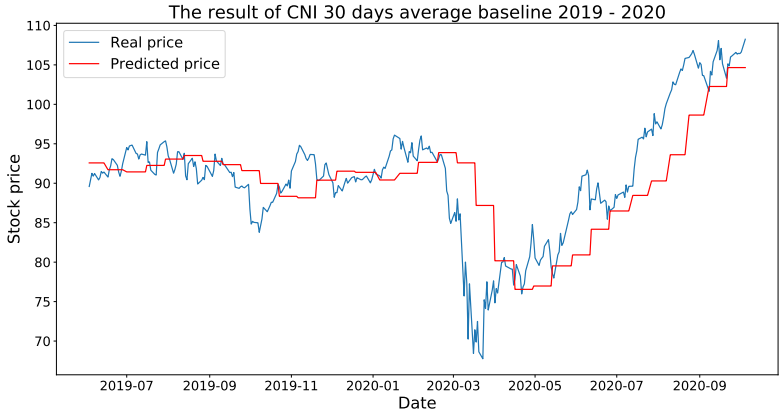



Figure 3.14: Baseline prediction of CNI using 30 days average

To check if the oscillation in the test result is an anomaly due to the volatility of the test data, we used the CNI 90 epochs model to predict the other 3 stocks. Such drastic oscillation occurs in all the predictions (Figure 3.20, 3.21, 3.22)

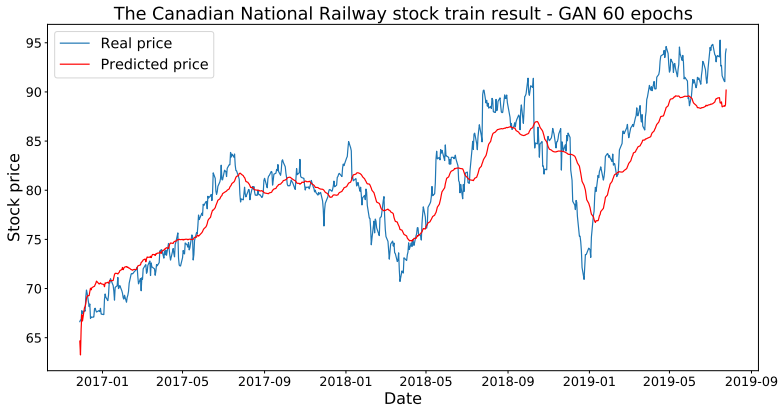



Figure 3.15: Training result of CNI using 60 epochs

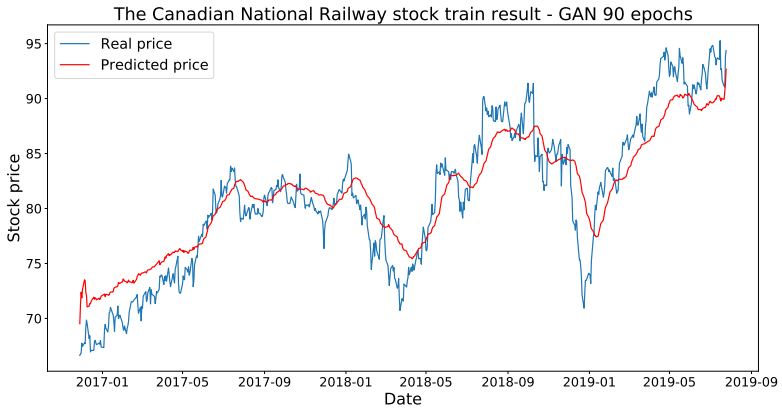



Figure 3.16: Training result of CNI using 90 epochs

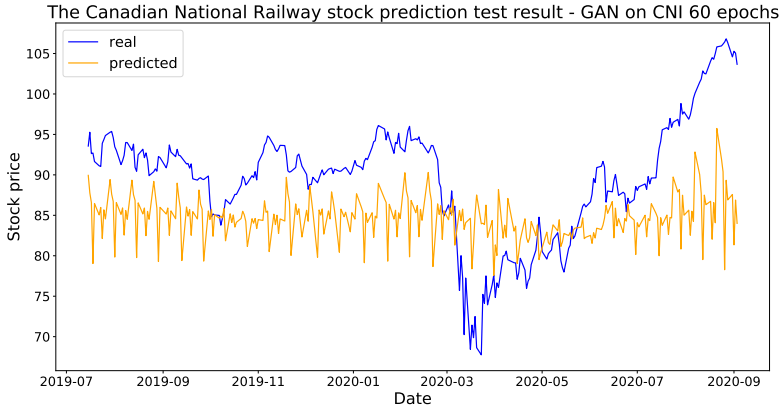



Figure 3.17: Test result of CNI using GAN model trained on Microsoft with 60 epochs

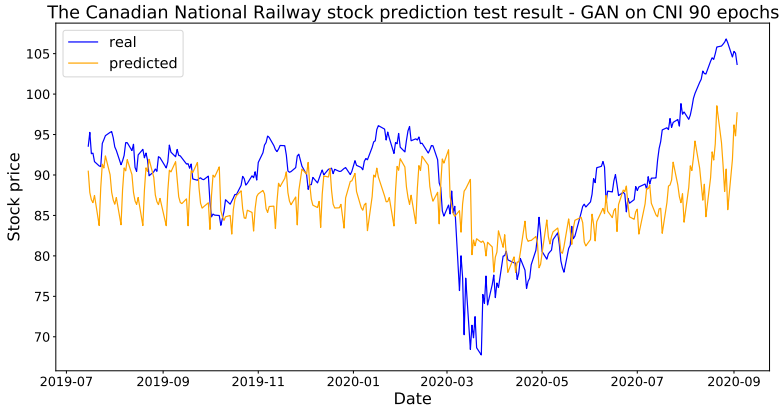



Figure 3.18: Test result of CNI using GAN model trained on Microsoft stock 2017-2019 with 90 epochs

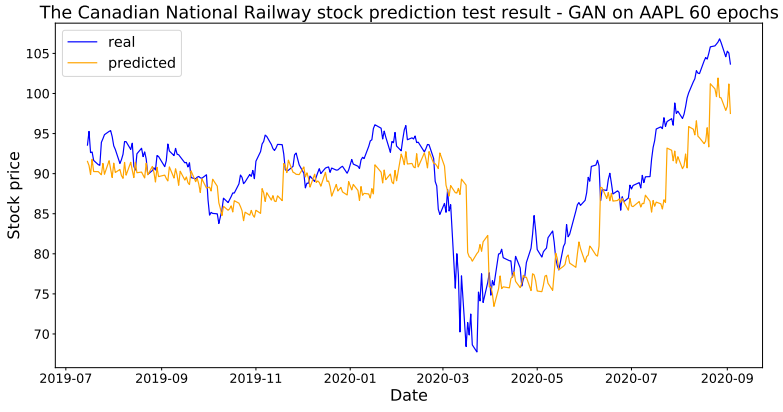



Figure 3.19: Test result of CNI using GAN model trained on Apple with 60 epochs

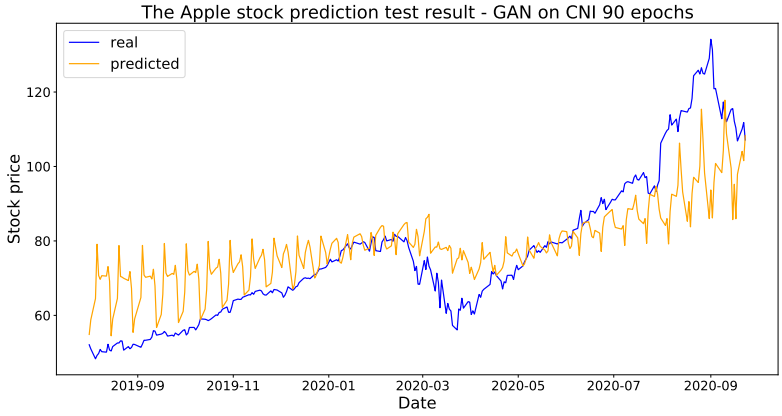



Figure 3.20: Test result of AAPL using GAN model trained on CNI with 90 epochs

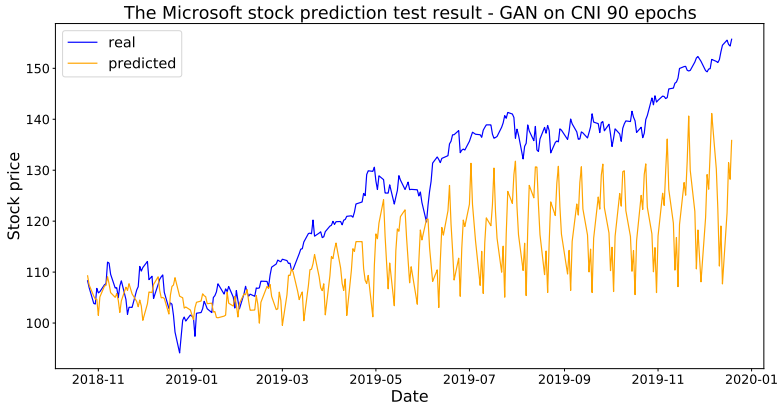



Figure 3.21: Test result of Microsoft using GAN model trained on CNI with 90 epochs

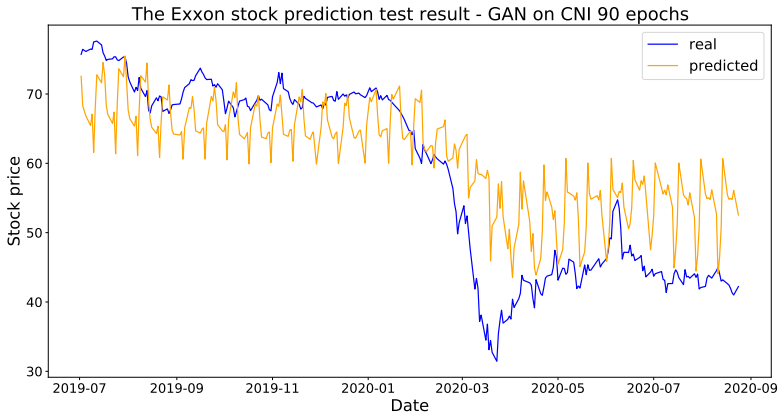



Figure 3.22: Test result of Exxon using GAN model trained on CNI with 90 epochs

### Exxon

A major difference between Exxon test data and the previous test data is that the trend goes downward following a volatile but generally "returning to the mean" training trend. The baseline model is not affected by the data longer than 30 days ago, so the performance is similar to when applied to the previous stocks (Figure 3.23).

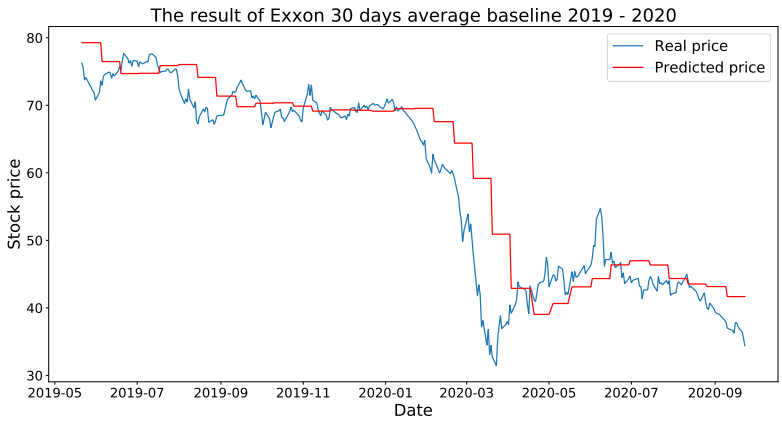



Figure 3.23: Baseline prediction of Exxon using 30 days average

The GAN model exhibits a big gap between prediction and real training data when trained with 90 epochs ( 3.27). The gap is more serious than the one that appears in Microsoft. The problem is mitigated when the model is trained with 60 epochs (Figure 3.24). This kind of gap does not occur in testing. All XOM 60 (Figure 3.26), XOM 90 (Figure 3.27) and APPL 60 (Figure 3.28). The AAPL 60 predicts the Exxon test data the best.

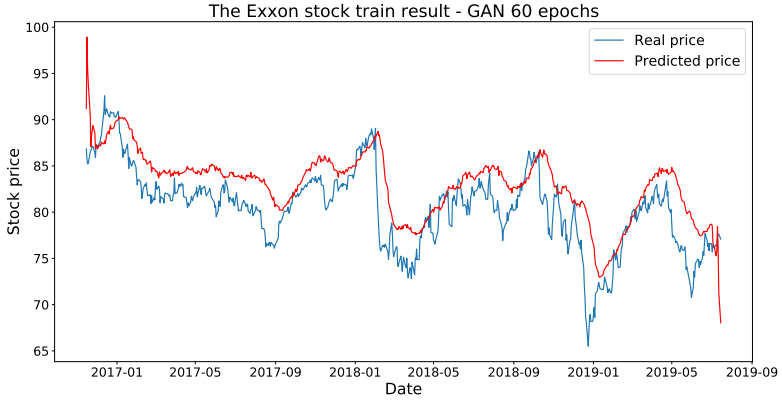



Figure 3.24: Training result of Exxon using 60 epochs

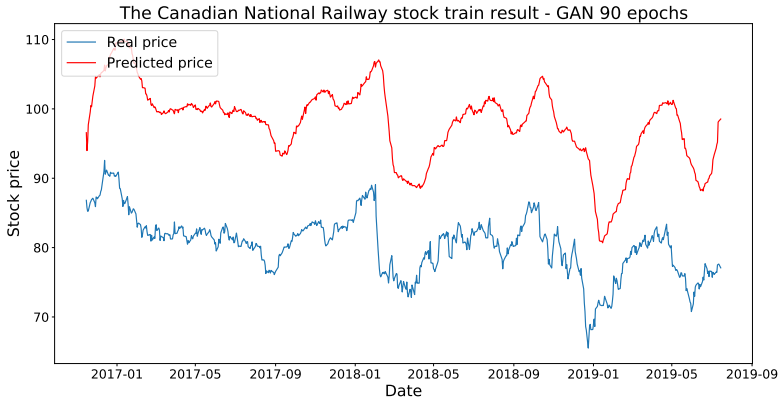



Figure 3.25: Training result of Exxon using 90 epochs

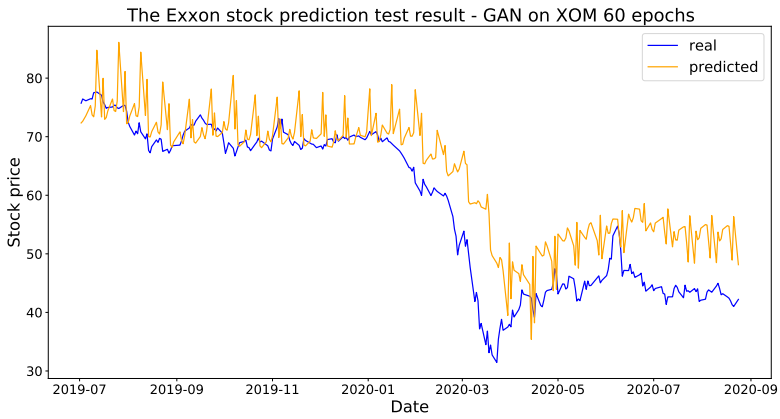



Figure 3.26: Test result of Exxon using GAN model trained on Exxon with 60 epochs

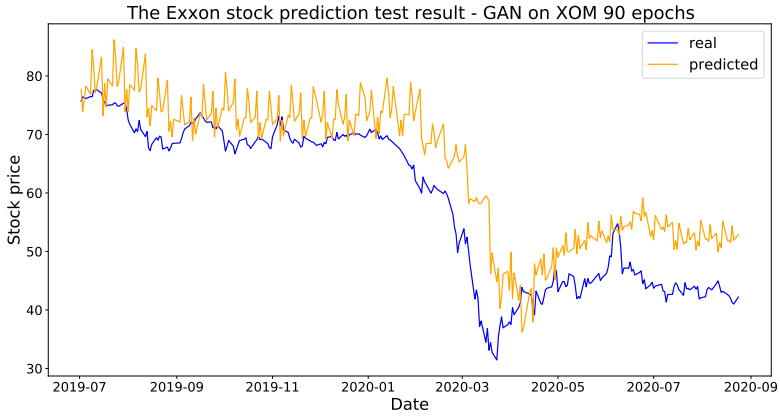



Figure 3.27: Test result of Exxon using GAN model trained on Exxon with 90 epochs

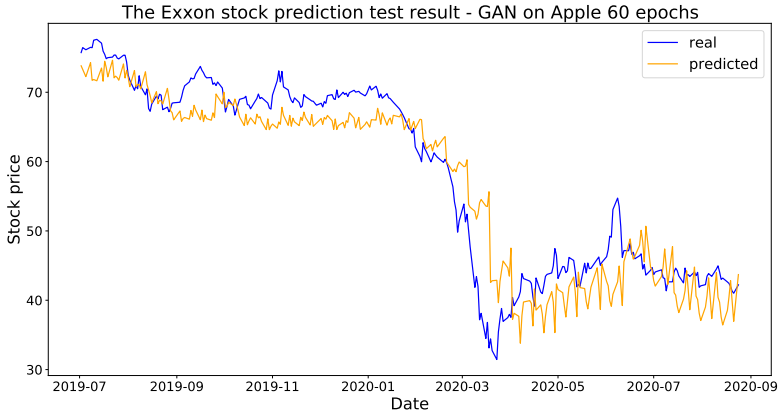



Figure 3.28: Test result of Exxon using GAN model trained on Apple with 60 epochs

### 3.0.2 Generate Historical Prices

We trained the model using the SP500 index closing prices from January 2020 to December 2022 (in total 755 days). The batch size we used is 64. After training, we input a series of random numbers into the generator to generate log return paths with 755 data points from 2020 to 2022 in each path (Figure 3.32). The random numbers serve as seeds for the generator to synthesize data.

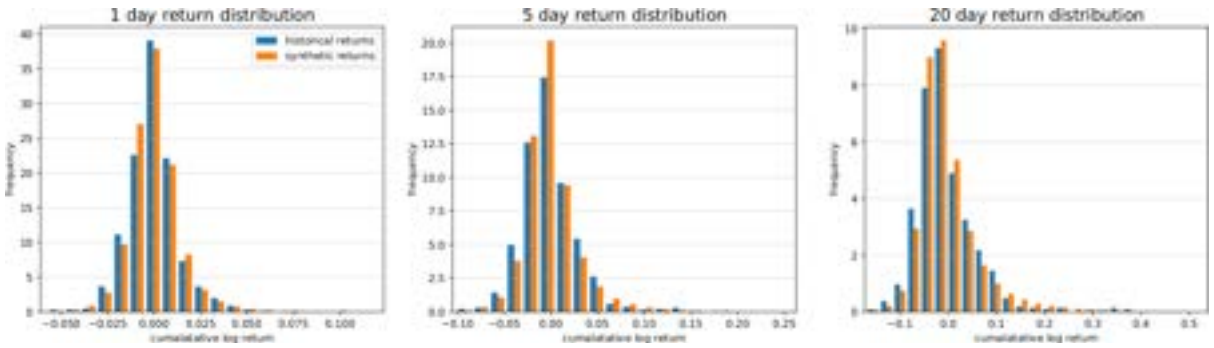



Figure 3.29: The distributions of the synthetic and actual log returns

The guassianized standardized real log returns approximately follow the normal distribution (Figure 3.31). Thus, after the data is guassinized, the model doesn't need to generate the heavy-tail distribution.

The distributions of the synthetic log returns are close to the actual log returns (Figure 3.29). The Earth Mover Distance (EMD) between the actual log returns distribution and the synthetic log returns distribution shows that the smaller the window size, the closer the distance (Table 3.3).

Table 3.3: Earth Mover Distance Between Actual and Synthetic Log-Returns

| Window Size | EMD |
|---|---|
| 1 | 0.000890 |
| 5 | 0.002910 |
| 20 | 0.005828 |
| 100 | 0.045216 |

The Auto Correlation Function (ACF) (Figure 3.34) and the leverage effect (Figure 3.35) of the synthetic log returns do not follow the ones of the actual log returns well. The autocorrelation of the synthetic log returns is smaller than the actual one and it converges to 0 more quickly. The leverage effect of the synthetic log returned, shown in the fourth image of Figure 3.34, also converges to 0 more quickly than the actual ones, and it does not follow the general pattern even when the lag time is small. The volatility clustering of the synthetic returns fits the pattern of the real returns well when the lag time is less than 10 days.

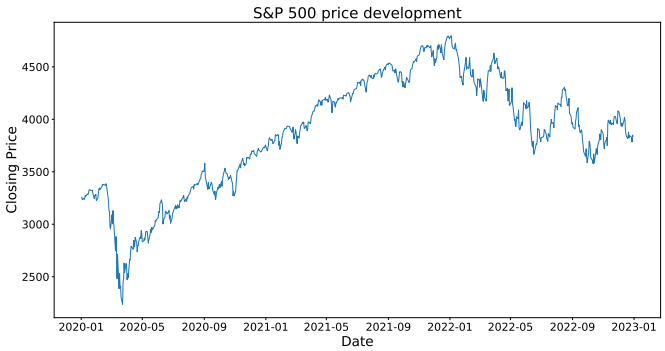



Figure 3.30: The closing prices of S&P 500 from 2020 to 2022

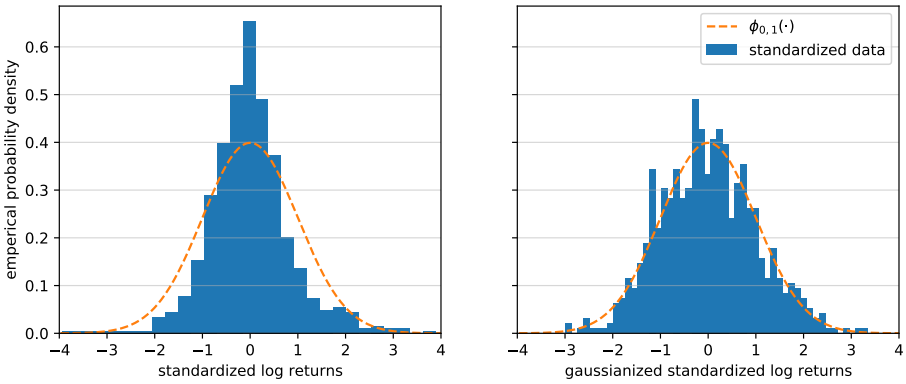



Figure 3.31: The standardized and gaussianized log returns

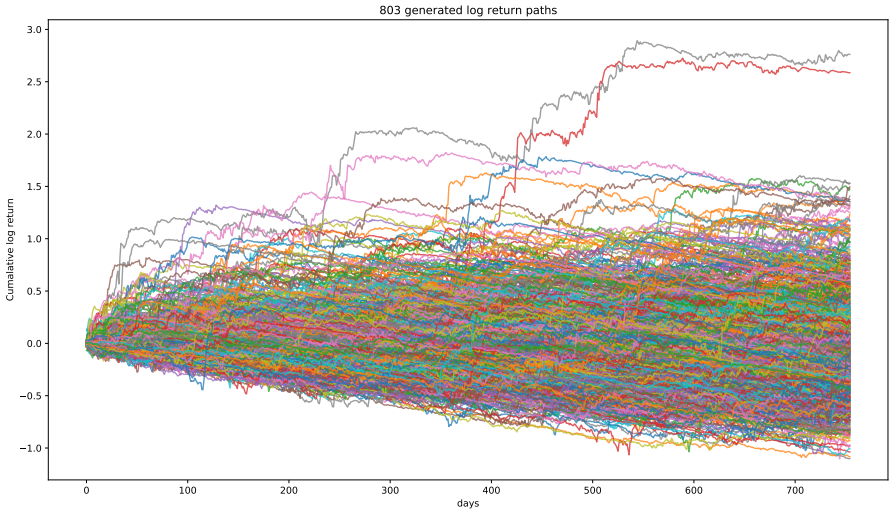
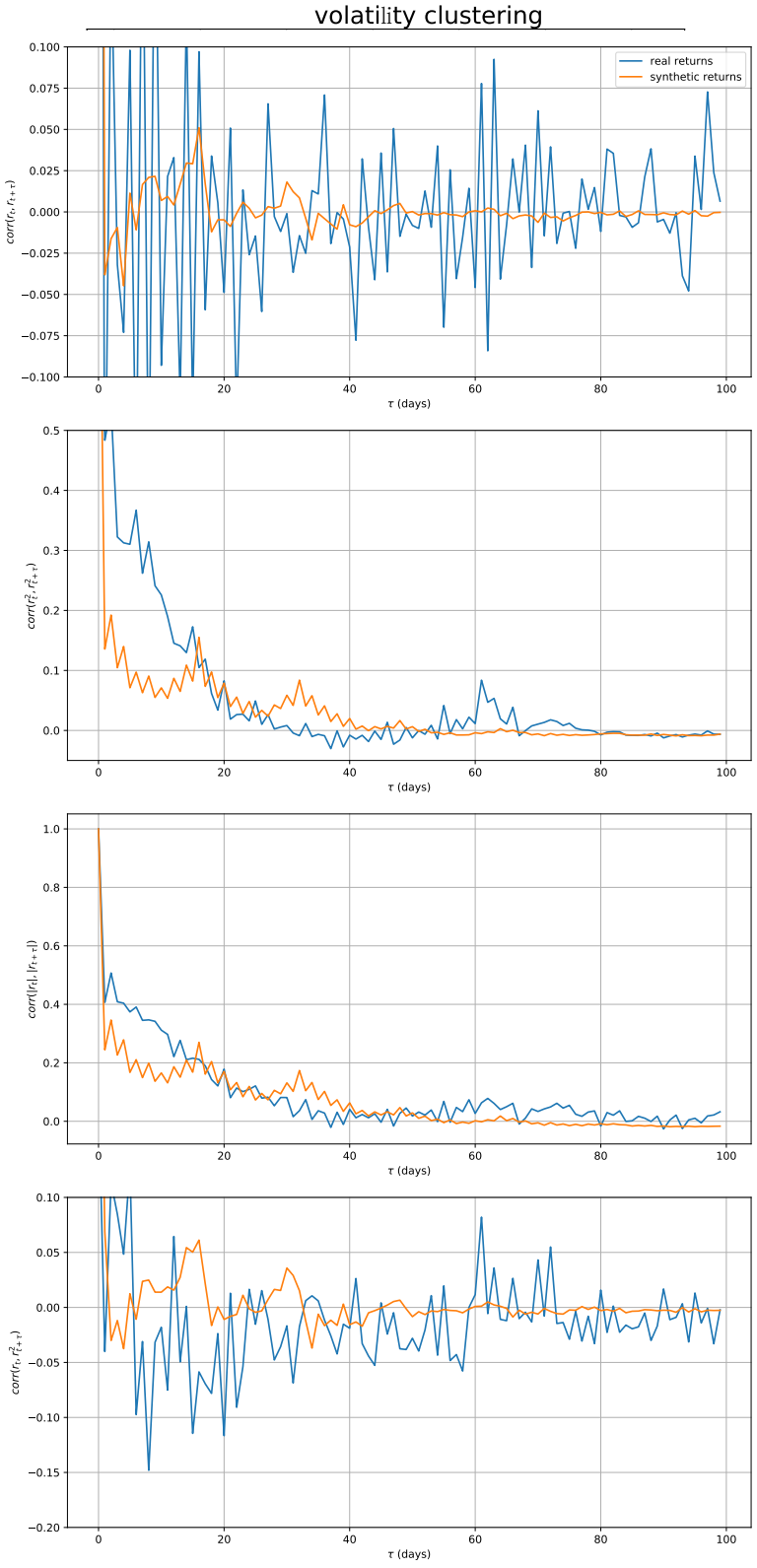



Figure 3.32: The synthetic log return path




Figu rices

s

Figure 3.35: The leverage effect scores of the synthetic and actual prices

Chapter 4

# DISCUSSION

The experimental results show that GAN has the ability to synthesize stock market prices based on previous market data better than the basic moving average model. This is more likely when the target stock has large price fluctuations and the moving average bench-line model reacts slowly. Moreover, the model trained on one stock can be used on other stocks and give good predictions. A well-trained model, such as the one trained with Apple stock data using 60 epochs, achieves the best performance among all the models in predicting all four stocks (Table 3.2). But this is not always the case. The model trained on Canadian National Railway stock using 90 epochs only gives good predictions on the same stock it was trained on. The predictions were poor on other stock compared to other models.

The returns synthetic by GAN can capture some of the stylized facts reflected in the actual returns. The synthetic log returns' distribution matches the actual log returns well. The EMD decreases as the time lag decreases, which indicates that the model simulates the log return distribution better when the log return has a shorter term.

However, the model does not capture the autocorrelation of the stock prices well. The autocorrelation functions of the synthetic return flat out much faster than the actual returns. This indicates a potential problem with the model — it does not capture the fluctuation of the stock market well.

The experimental results also reveal several problems with GAN in generating trend lines using the rolling window approach. For example, the predicted price lines are always volatile and only roughly follow the big trend, which means that most local high points and low points do not have real-world meanings and can not be used as references for investment. One possibility of this volatility might be that the discriminator of GAN learns from volatile data and tends to mark volatile images as authentic images, encouraging the generator to output very volatile data.

Another problem is that GAN occasionally generates trend lines with strong oscillation patterns. This problem might be a sign of mode collapseexplain the term, in which the generator can only produce a small set of outputs even when the input data vary. The test results of the Canadian National Railway and Microsoft stocks suggest that a model with good performance in training can have serious mode collapse problems in testing. Adjusting the number of epochs in the training part help mitigate the problem. But we did not find enough evidence of what training results might indicate mode collapse.

Besides, the trend line predicted by GAN can go parallel but not overlap with the real trend. For instance, the training result of Apple and Exxon stocks. It might result from the discriminator's

failure to capture the mean of the input data and discriminate the generated lines that are visibly higher or lower than the actual ones. This problem might be mitigated by giving higher penalties to the synthetic data whose mean value is significantly different than the actual data.

Chapter 5

# CONCLUSIONS

In this signature project, we showed that GAN model can be used to approximate and predict financial time series in discrete time. In approximating historical prices, the model delivers promising return distribution results but does not capture the autocorrelation of stock prices well. In terms of predicting future prices, GAN model shows its potential to perform better than the classic moving average model and make predictions about stocks other than the stock it is trained on. But the training of the model is unstable and there is no guarantee that it will perform well, which makes it a less ideal model to predict future prices. To better leverage GAN's ability in the stock market in the future, a more detailed evaluation system for the discriminator should be developed so that the average generated prices can approximate the average actual prices better, and the volatility rate of the generated prices can be fine-tuned.

The stock market is an exceedingly complex system that is influenced by numerous factors. It is unrealistic to expect a single AI model or method to accurately predict its behavior. A practical AI system capable of making such predictions must combine various methods and models. Our research has shown that GAN has the potential to be a valuable component of such a system.

# REFERENCES


[1] Akosner. *Generative AI: A Creative New World*. Sequoia Capital US/Europe, 2022.

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. 2018. DOI: 10.48550/ARXIV.1803.01271.

[3] Boris Banushev. "Using the Latest Advancements in AI to Predict Stock Market Movements". In: (2022).

[4] F. Black. "Studies of Stock Price Volatility Changes". In: *American Statistical Association* (1976), pp. 177–181.

[5] George E.P. Box et al. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2015. ISBN: 9781118674925.

[6] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep Learning Techniques for Music Generation – a Survey*. 2017. DOI: 10.48550/ARXIV.1709.01620.

[7] Nicolas Champagnat et al. *An Empirical Analysis of Heavy-Tails Behavior of Financial Data: The Case for Power Laws*. hal.science, 2013.

[8] Andrew C. Chang and Phillip Li. "Is Economics Research Replicable? Sixty Published Papers from Thirteen Journals Say 'Usually Not'". In: (2015).

[9] Huixiang Chen et al. "Retracted on January 26, 2021: 3D-Based Video Recognition Acceleration by Leveraging Temporal Locality". In: *Proceedings of the 46th International Symposium on Computer Architecture*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 79–90. DOI: 10/gnqdh8.

[10] Kyunghyun Cho et al. *Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. DOI: 10.48550/ARXIV.1406.1078.

[11] Michael A. Clemens. "THE MEANING OF FAILED REPLICATIONS: A REVIEW AND PROPOSAL". In: *Journal of Economic Surveys* 31 (2015), pp. 326–342. DOI: 10/f9r3m4.

[12] R. Cont. "Empirical Properties of Asset Returns: Stylized Facts and Statistical Issues". In: *Quantitative Finance* 1 (2001), pp. 223–236. DOI: 10/d4dh8f.

[13] Ugur Demir and Gozde Unal. *Patch-Based Image Inpainting with Generative Adversarial Networks*. 2018. DOI: 10.48550/ARXIV.1803.07422.

[14] Anand Deshpande and Manish Kumar. *Artificial Intelligence for Big Data : Complete Guide to Automating Big Data Solutions Using Artificial Intelligence Techniques*. Packt Publishing, 2018.

[15] Chris Drummond. "Replicability Is Not Reproducibility: Nor Is It Good Science". In: *Proceedings of the Evaluation Methods for Machine Learning Workshop at the 26th ICML*. Vol. 1. National Research Council of Canada Montreal, Canada. 2009.

[16] Ronen Eldan and Ohad Shamir. *The Power of Depth for Feedforward Neural Networks*. proceedings.mlr.press, 2016.

[17] Matt W Gardner and SR Dorling. "Artificial Neural Networks (the Multilayer Perceptron)—a Review of Applications in the Atmospheric Sciences". In: *Atmospheric environment* 32.14-15 (1998), pp. 2627–2636. DOI: 10/ft4hjb.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The Mit Press, 2016.

[19] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. DOI: 10.48550/ARXIV.1406.2661.
[20] Kay Gregor Hartmann, Robin Tibor Schirrmeister, and Tonio Ball. *EEG-GAN: Generative Adversarial Networks for Electroencephalograhic (EEG) Brain Signals*. 2018. DOI: 10.48550/ARXIV.1806.01875.
[21] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural networks* 2.5 (1989), pp. 359–366. DOI: 10/c4wch5.
[22] Icascha. *QuantGANs-replication*. 2021.
[23] Kundan Kumar et al. "MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis". In: 32 (2019).
[24] Christian Ledig et al. *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*. 2016. DOI: 10.48550/ARXIV.1609.04802.
[25] Dan Li et al. "MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks". In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series* (2019), pp. 703–716. DOI: 10/gjq2b2.
[26] Huangchun Lin. *Stock-Price-Prediction-Using-GAN*. 2020.
[27] HungChun Lin et al. "Stock Price Prediction Using Generative Adversarial Networks". In: *Journal of Computer Science* 17 (2021), pp. 188–196. DOI: 10/gm64wx.
[28] Andrew W. Lo and A. Craig MacKinlay. "Stock Market Prices Do Not Follow Random Walks: Evidence from a Simple Specification Test". In: *Review of Financial Studies* 1 (1988), pp. 41–66. DOI: 10/fqc7sk.
[29] Zhou Lu et al. *The Expressive Power of Neural Networks: A View from the Width*. 2017. DOI: 10.48550/ARXIV.1709.02540.
[30] Yonghong Luo et al. "Multivariate Time Series Imputation with Generative Adversarial Networks". In: 31 (2018).
[31] Benoit Mandelbrot. "The Variation of Certain Speculative Prices". In: *The Journal of Business* 36 (1963), p. 394. DOI: 10/bxnm72.
[32] Warren S McCulloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133. DOI: 10/djsbj6.
[33] Aaron van den Oord et al. *WaveNet: A Generative Model for Raw Audio*. 2016. DOI: 10.48550/ARXIV.1609.03499.
[34] Open AI. *ChatGPT*. 2022.
[35] Joelle Pineau et al. "Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program)". In: *Journal of Machine Learning Research* 22 (2021), pp. 1–20.
[36] Karl R Popper. *The Logic of Scientific Discovery*. Routledge, 1959.
[37] Aditya Ramesh et al. *Zero-Shot Text-to-Image Generation*. 2021. DOI: 10.48550/ARXIV.2102.12092.
[38] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. DOI: 10.48550/ARXIV.2112.10752.
[39] Frank Rosenblatt. *PRINCIPLES OF NEURODYNAMICS. PERCEPTRONS AND THE THEORY OF BRAIN MECHANISMS*. 1961.
[40] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. "The Earth Mover's Distance as a Metric for Image Retrieval". In: *International journal of computer vision* 40.2 (2000), pp. 99–121. DOI: 10/fbn8sn.
[41] Nicholas Sapankevych and Ravi Sankar. "Time Series Prediction Using Support Vector Machines: A Survey". In: *IEEE Computational Intelligence Magazine* 4 (2009), pp. 24–38. DOI: 10/bnrrpm.

[42] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. “Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005–2019”. In: *Applied Soft Computing* 90 (2020), p. 106181. DOI: 10/ghrnh2.
[43] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. “Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005–2019”. In: *Applied Soft Computing* 90 (2020), p. 106181. DOI: 10/ghrnh2.
[44] Matus Telgarsky. *Benefits of Depth in Neural Networks*. proceedings.mlr.press, 2016.
[45] Dan Christina Wang and Per A. Mykland. “The Estimation of Leverage Effect with High Frequency Data”. In: *SSRN Electronic Journal* (2011). DOI: 10/fx94mp.
[46] Magnus Wiese et al. “Quant GANs: Deep Generation of Financial Time Series”. In: *Quantitative Finance* 20 (2020), pp. 1419–1440. DOI: 10/gg3c6b.
[47] *Yahoo! Finance*.
[48] Xingxing Zhang and Mirella Lapata. “Chinese Poetry Generation with Recurrent Neural Networks”. In: (2014), pp. 670–680.
[49] Xingyu Zhou et al. “Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets”. In: *Mathematical Problems in Engineering* 2018 (2018), pp. 1–11. DOI: 10/gdhrnx.
[50] Jun-Yan Zhu et al. *Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks*. 2017. DOI: 10.48550/ARXIV.1703.10593.