UC Berkeley MEng IEOR - FinTech Concentration

Capstone Report

FinTech Group 3

Yue Hu

Chien-Hsun Huang

Ming-Yu Chi

Nicolas Kardous

Yao-chieh Hu

Alex Chen

Executive Summary

Limit order book is the mechanism displaying the intention of buyers and sellers for financial assets and revealing the market equilibriums of supply and demand at each moment. It can extract lots of information from market participants from limit order books. For instance, traders predict short-term trends in stock prices as well as making trading strategies based on the extracted information of dynamics between the limit order book of stocks and market prices. However, most of the research such as traders verifying their trading strategies were done on the only one stream of historical limit order book data that our world has happened. Therefore, data augmentation of limit order book market streams is inevitable for the sake of further studies such as improving performances and reliabilities of high-frequency trading strategies. Combining trending machine learning techniques with financial data, we implement GANs to synthesize data that aim to mimic the real data distribution patterns. Specifically, we employ Sequential GAN, which is adept at dealing with NLP problems, to synthesize our financial limit order book data. Our research shows that Sequential GAN is a viable approach for augmenting limit order book data and the general distribution shape of our synthesized data highly match with the real one despite some different peak value. For further experiments, we could elaborate a more sophisticated neural network structure with our Sequential GAN model to further increase the similarity of the synthesized data with real data as well as compare the performances of our approach with different deep learning approaches.

1. Introduction

The economy is a weighty pillar to support the growth and stability of a nation, no matter a well established one or an emerging one. A meticulous approach to fathom the wellness of the economy is usually through an examination of its financial health. Since the financial sector comprises all corporations, banks, agencies, funds, and businesses, the health of it indirectly reflects the health of the economy itself, which can pose either a threat to humanity or a positive sign to our historic achievements. With an aim to have a better understanding of the financial autonomy, described by The Wealth of Nations of Adam Smith in the age of Enlightenment, March 1776, this paper devotes its focus to dissect and simulate the reaction of the market.

The financial industry is under the supervision of an invisible hand, addressed by Smith, that encourages competition and obliterates monopolies. For the stock markets that enable liquidation of the underlying ownerships of all entities across continents, the value of its study presents significance. From the order book that traders can place various categories of orders, i.e. market orders and limit orders, it could be inconspicuous to address a formulated system to depict the reaction of the market.

As auto-trading bot became a trendy focus, every sector is gearing toward an algorithm that is above the performance average. However, without an in-depth perception of the order book reaction, the sandbox tests on the algorithm would either be inaccurate or overpriced. Specifically, one obvious way is to encapsulate and rerun the historical data, which can be cost-effective while imprecise due to the fact that the impact of the executed orders is overlooked. The impact can be momentous if either the market size is small, or the order size is large, comparatively. The other way is to adopt the algorithm to the field, which may burn lots of cash and intrigue red lights from regulators and other competitors.

2. Literature Review

Since the 1970s, systematic and quantitative modeling approaches have been utilized in the financial industry. Initially, these models were made up of econometric tools and statistical analysis. However, in the last 10 years, machine learning and data science is ubiquitous in the financial industry. For example, machine learning is used in assessing financial news headlines, where natural language processing is used to predict if a stock's price would increase or decrease given the day's news. Additionally, machine learning could be used to simulate market transactions and financial data, and reinforcement learning is used to build automated trading bots. Our Capstone project specifically looks at implementing a combination of supervised learning and reinforcement learning to generate realistic simulations on the behavior of how financial securities are bought and sold.

In researching different techniques, the subcategories of machine learning have been frequently combined to produce outstanding results. Specifically in the problem at hand, our project looks to combine supervised learning with reinforcement learning. Our capstone project very much resonates with the work that is done in the financial industry, because many firms use similar models and techniques in creating predictions and simulations for the financial markets. For example, JP Morgan recently released a research paper titled Get Real: Realism Metrics for Robust Limit Order Book Market Simulations, that uses reinforcement learning to create accurate simulations for buy and sell behavior. This is a very exciting domain because what we are developing for our Capstone project is what is currently researched from firms in the industry today.

When assessing the quantitative applications that are used in the financial industry, the main tools are machine learning, and mathematical and financial models. These mathematical and financial models to a large degree incorporate statistics and time series analysis. Examples of these models are Autoregressive model (AR), Autoregressive moving average model (ARMA), Autoregressive conditional heteroskedasticity (ARCH) and so on. These models were frequently used in the 1980s, however, they aren't really used nowadays. While these models are still applicable, they are not as accurate as mathematical models that utilize machine learning. Because of this, machine learning techniques, whether they are used to a very large or very small degree, are the main models that could compete with our model.

Our implementation is similar to what current competitors and stakeholders use. The difference between our technology and other technologies could be viewed as substitutes. These substitutes would be in the form of different machine learning models that assess different financial problems. For example, machine learning could be viewed as a big toolbox, with multiple tools to solve a problem. In this case, the substitutes are the different tools people could use to solve the problem, where multiple tools could be applied to the same problem. Thus, we hope to address the threat of substitutes by choosing a tool that maximizes our model's accuracy. Additionally, we have to ensure that our model is interpretable, requires minimal computing power, and is customizable where it could be specifically adaptable to the context of the problem.

The bulk of the research papers we looked at investigated the application of machine learning to some financial problem. We will discuss each article in the order found in the sources section found at the end of the document. Because we are looking to implement a GANs model (Generative Adversarial Network), most of the articles we look at implement GANs to some extent.

The three main research articles that we found that are relevant to our technology are:

1. Sequence Generative Adversarial Nets with Policy Gradient
2. Simulation of Credit Default Swap Index Transaction Data
3. Get Real: Realism Metrics for Robust Limit Order Book Market Simulations.

The first article looks at developing a neural network algorithm called SeqGAN (Sequential Generative Adversarial Network), and we plan to use this neural network architecture as the backbone of our model. The second paper looks at utilizing the SeqGAN algorithm on Credit Default Swap data. This paper is a helpful guide to see how the SeqGAN algorithm could be applied to financial data. The third paper looks at implementing reinforcement learning in creating market simulations for limit order book. Our project plans on using the SeqGAN algorithm on the limit order book data. Thus, we are studying the model created in the first paper, assessing how that model is applied to financial data from the second paper, and applying our model to the data that is used in the third paper.

To go about achieving our technical goal, we first need to have a robust understanding of how the SeqGAN works. Given a dataset of real-world structured sequences, we train a θ-parametrized generative model G_θ to produce sequences that mimic the real one. We want G_θ to fit the unknown true data distribution $p_{true}(y_t : Y_{1:t-1})$, which is only revealed by the given dataset $D = \{Y_{1:T}\}$. Our GANs model is made up of a generator (G) and a discriminator (D). The discriminator tries to correctly distinguish the true data and the fake model-generated data. Conversely, the generator tries to generate high quality data to fool the discriminator. Ideally, when D cannot distinguish the true and generated data, G nicely fits the underlying data distribution. All in all, the SeqGAN method is used to effectively train Generative Adversarial Nets for discrete structured sequences generation via policy gradient.

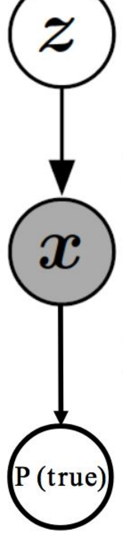


Figure 1: The Figure above shows the basic idea of how the generator and discriminator work with each other. A random probability distribution is passed through the generator, the generator then outputs features. These features are then inputted into the discriminator, and the discriminator should output whether or not the data was real data or fake data.

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(x)] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))]$$

Equation 1: The equation above is the cost function that the Generative Adversarial Network is based on. If the discriminator correctly predicts that the data is real, then it should output a 1, and if it correctly predicts that the data is fake, then it should output a 0. Thus, this equation shows that the discriminator wants a maximum value in the cost function, and the generator wants a minimum value in the cost function.

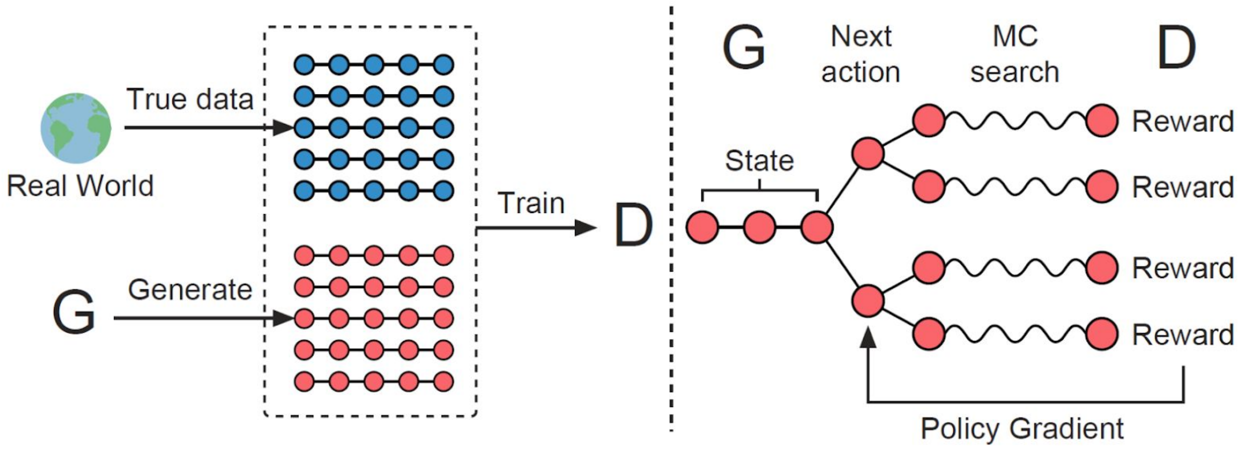


Figure 2: (Yu, 2017) Generator is a reinforcement learning policy $G_\theta(y_t : Y_{1:t-1})$ of generating a sequence. Decides the next data to generate (action) given the previous ones as the state. Discriminator provides the reward (probability of being true data) $D_\phi(Y^n{}_{1:T})$ for the sequence

3. Data Description

3.1 Data Statistics

We have practiced some preliminary data analysis on the limit order book of Apple, Amazon, Google, Intel and Microsoft five companies. We verify if these limit order books match the hypothesis of Roll's model (1984). On the other hand, we are going to use these summary statistics as metrics to verify the effectiveness of our GANs model synthesizing limit order book data.

First, we are going to verify the covariance of the price difference between time i and time $i-1$, that is d_i, with the difference of its one lag term $i-1$, namely d_{i-1}, to see if each of them is negatively correlated. From the table we can conclude that the covariances are all negative for all stocks which coincides with Roll's model.

Unit: 0.01 cent	**Apple**	**Amazon**	**Google**	**Intel**	**Microsoft**
Covariance	-305,382	-242,338	-1,101,870	-1,741	-1,691

Figure 3: Covariance of the price difference

Then, we check if the price difference between time i and time $i-1$, that is d_i, are uncorrelated with the price difference between time j and time $j-1$, that is d_j, for those time j and time i has a time gap no less than 1 time period. The following figure shows that all the d_i and d_j has a covariance around 0 if the number of time gaps between i and j are greater than 1 which again match with the Roll's model.

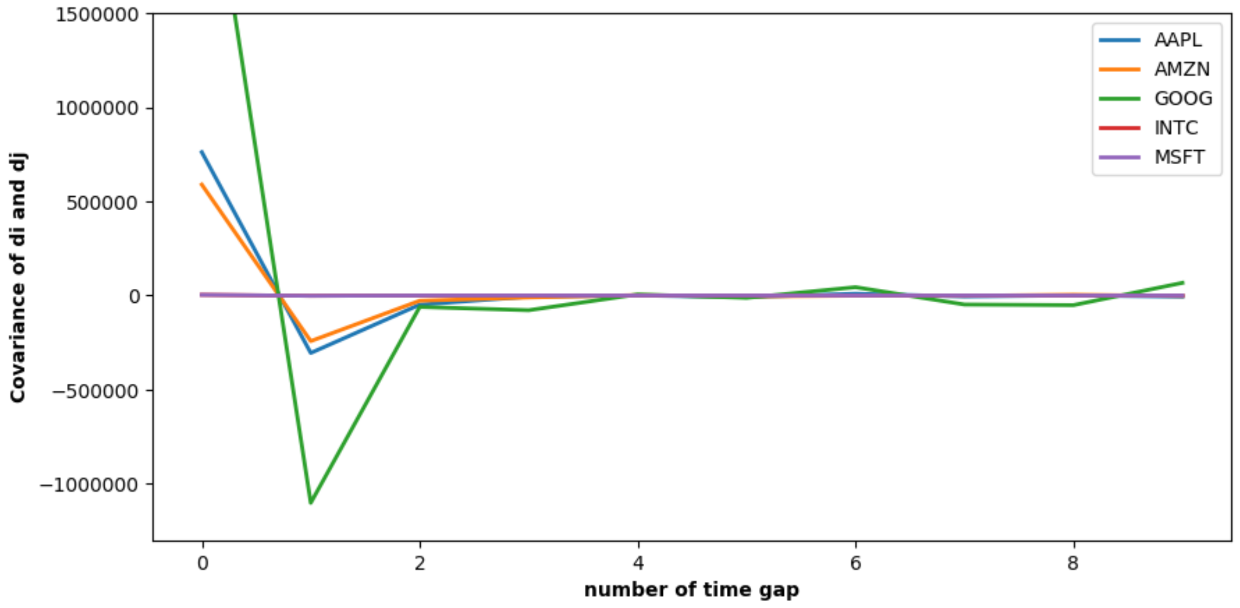


Figure 4: Covariance of di and dj

Next, we are going to check if the bid-ask spread of each stock is constant or relatively constant over transaction time. From the following diagrams we can observe that only Intel and Microsoft are relatively stable on the bid-ask spread while Apple, Amazon and Google are variate on it. Interestingly, even though Apple, Amazon and Google are not constant on their bid-ask spread which does not match the Roll's model, each of the limit order books has a bell shape on the bid-ask spread distribution with some skewness and different center.

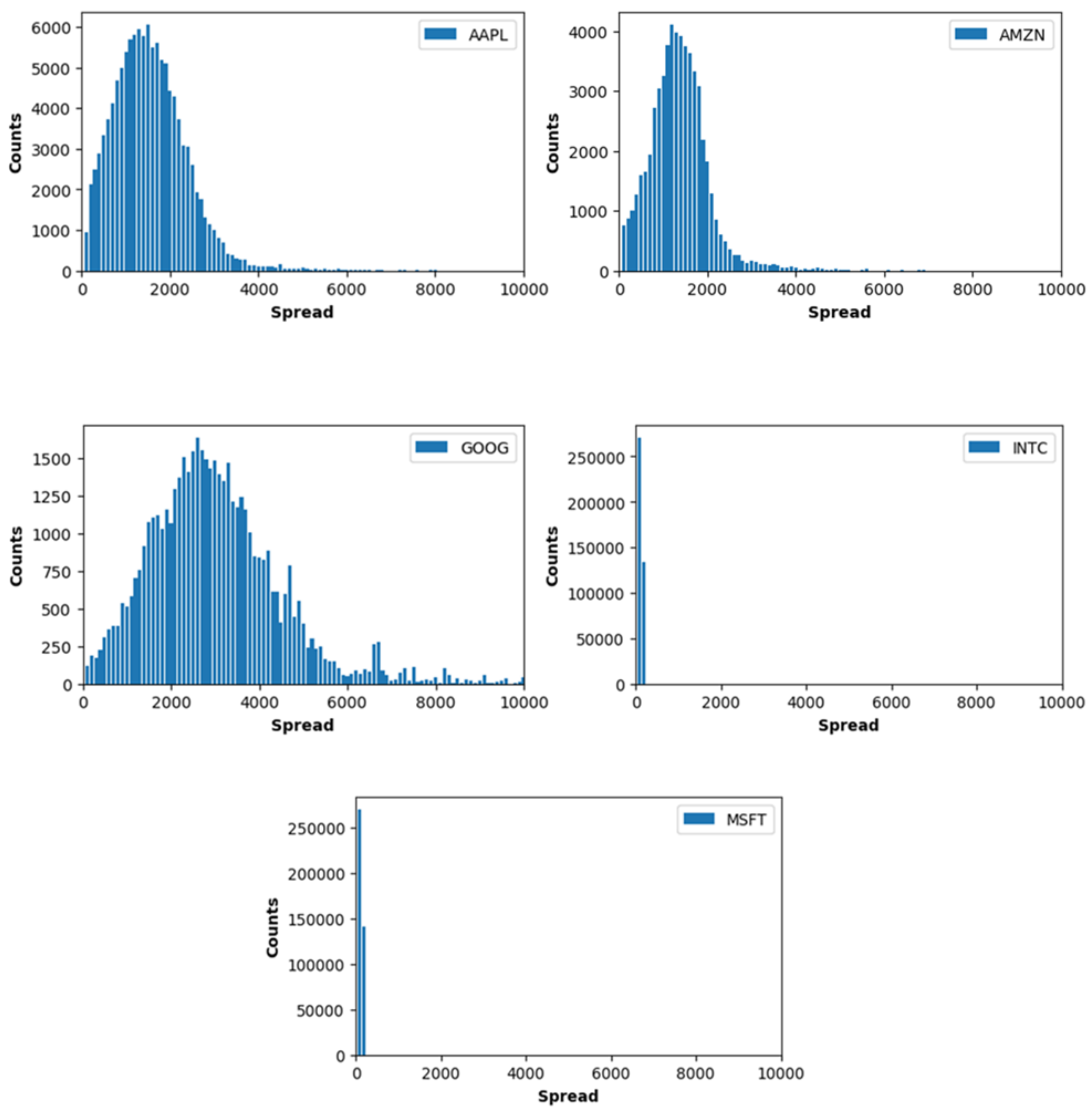


Figure 5: Distribution of stocks

One possible reason that Intel and Microsoft are relatively constant while Apple, Amazon and Google are variate on the bid-ask spread may be that the trading volume of each stock are different. To elaborate, the intuition of the relationship between trading volume and bid-ask spread is that if the market tends to be shallower, then the bid-ask spread tends to be larger and vice versa. This reflects the fact that when there are more participants in the market, the more possible they could offer the liquidity to the market no matter long or short position and whenever there is enough liquidity there should appear least spread. Therefore, when we examine the trading volume from the following table, we could learn that Intel and Microsoft have the most trading volume of their stock while Apple, Amazon and Google have the least trading volume relatively which may lead to the inconsistent of the bis-ask spreads among different stocks.

Unit: share	Apple	Amazon	Google	Intel	Microsoft
Volume	10,122,327	5,148,586	3,791,780	194,887,246	234,974,657

Figure 6: Volume of different stocks

3.2 Bid and Ask Spread

We generated graphs looking at the bid-ask spread for the stock prices of companies Apple, Amazon and Google. We collected the data from the LOBSTER sample dataset provided from the website. As it can be seen from the three figures below, the bid-ask spread for all three companies follows a general distribution where there are greater bid order sizes on the left portion of the graph and greater ask order sizes on the right portion of the graph.

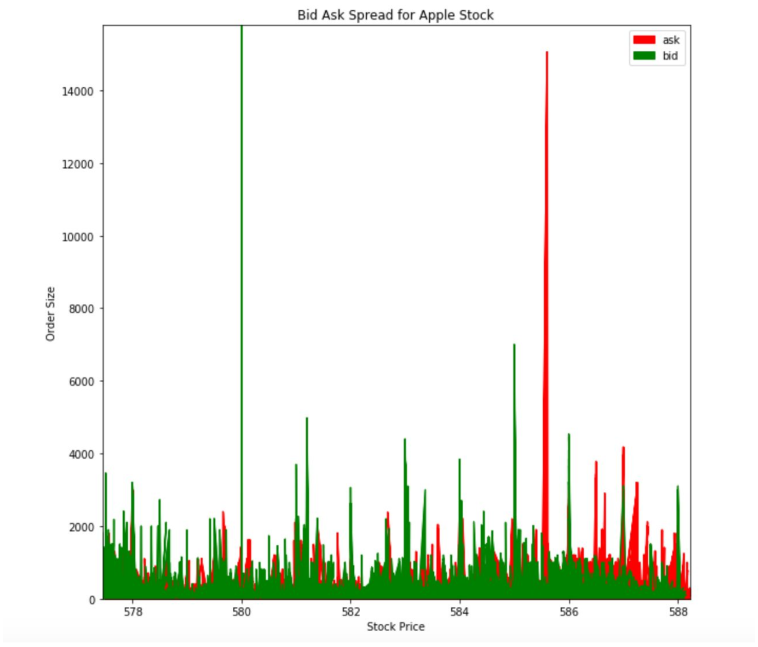


Figure 7: Bid Ask Spread for Apple Stock, data taken from LOBSTER samples

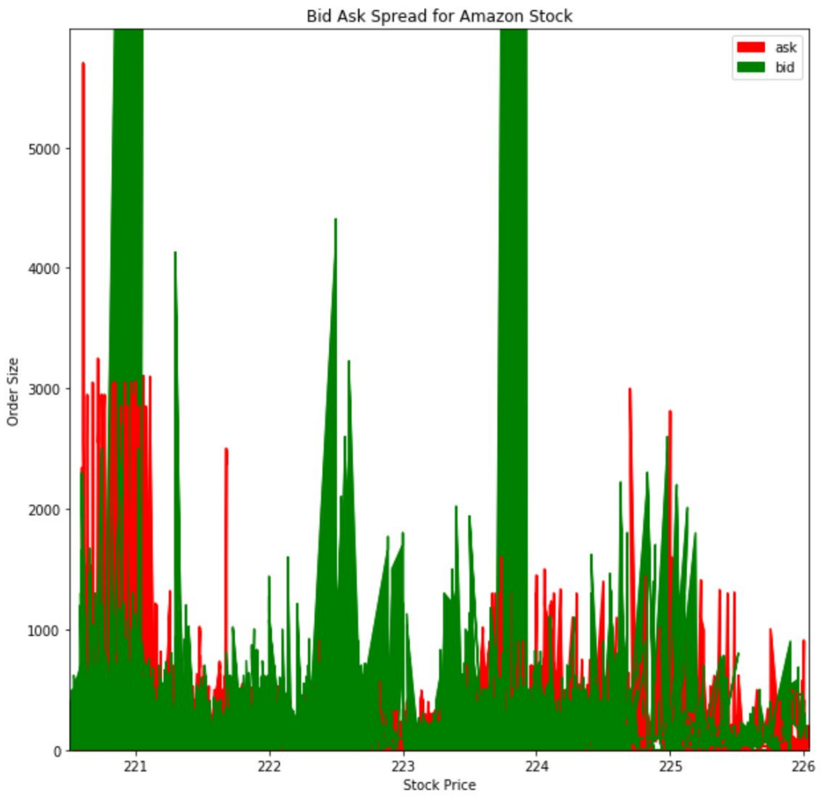


Figure 8: Bid Ask Spread for Amazon Stock, data taken from LOBSTER samples

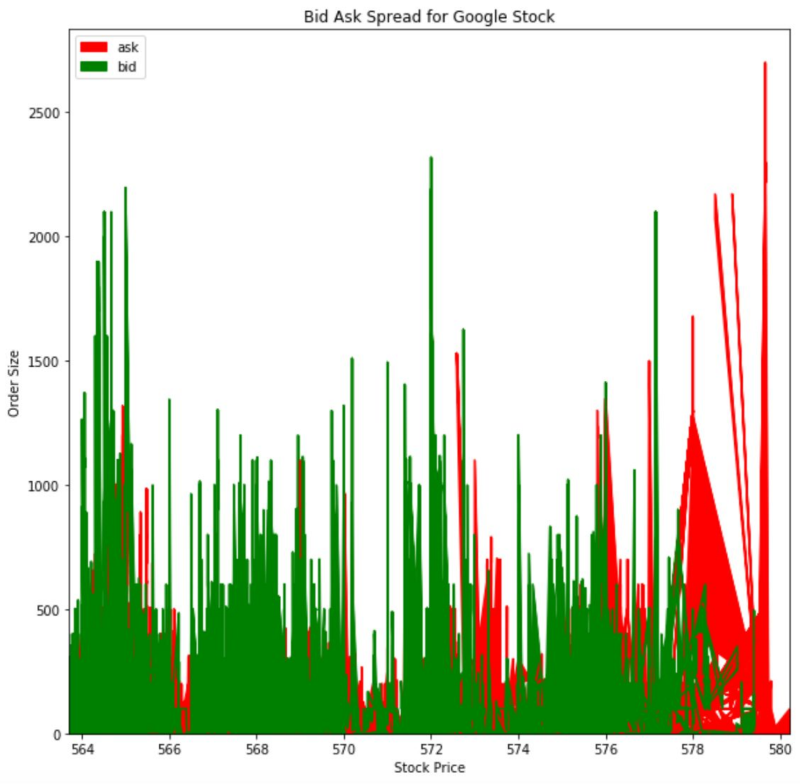


Figure 9: Bid Ask Spread for Google Stock, data taken from LOBSTER samples

4. Models

4.1 Rationale to use SeqGAN

GANs: GANs, generative adversarial networks, is a model that helps develop a generator for data generating. It is a supervised learning model that works by training two sub-models, a generator and a discriminator. The generator starts by randomly generating new datas following an initial distribution. Then the discriminator tries to distinguish between the real data and fake data and give the feedback to the generator. The generator then updates the distribution and then repeats the former steps. Therefore, the generator and discriminator grow stronger together.

SeqGANs: GAN has many applications in the image field. However, there are some limits when dealing with sequential data. The main reason for the poor performance is that GAN encounters two problems when generating continuous discrete sequences: First, because the output of the generator is discrete, the gradient update is more difficult to get to the generator from the discriminator. The second is that the discriminator can only judge when the sequence is completely generated, so the guidance is not very useful at this moment. And if the generator generates the sequence while the discriminator judges, how to balance the score of the current sequence with scores for future sequences are another problem

Improvements from GANs to SeqGANs: To overcome these problems, the SeqGANs model considers the generator as a stochastic process in reinforcement learning, so that SeqGAN can directly avoid gradient problems in the generator through the gradient policy update. At the same time, the discriminator's score on the entire sequence as a reward signal for reinforcement learning can be passed to the intermediate moment of sequence generation through Monte Carlo search. Specifically, it views the process of generating sequences as a sequence decision process in reinforcement learning. The generation model is regarded as an agent. The sequence that has been generated so far represents the current state. The next term to be generated is the action to be taken. The evaluation model's evaluation score for the sequence is the return.

Structure of model: The model contains six main parts, which are Dataloader, Generator, Discriminator, Rollout, target_lstm, main.

I. **Dataloader:** Generate training data. For Generator, we only use dataloader in pre-training to get training data. For Discriminator, we need to use dataloader to get training data in pre-training and adversarial training process. In the eval process, when the similarity determination between the Generator and the oracle model is performed, the dataloader is used to generate data.
II. **Generator :** The generator is used to generate some simulation data which is similar to the distribution of the real data.
III. **Discriminator:** The discriminator is to identify whether a data is real or fake, and then give feedback to the generator. While training the discriminator, the real data are viewed as positive data, and the generated data are viewed as negative data. Then the discriminator train a binary classification model.
IV. **Rollout:** The rollout model is like a generator to solve the sequential problem of GANs. It defines the sampling process when calculating reward.
V. **Target_lstm:** This is the oracle model. It is used to generate the real data at the beginning.
VI. **Main:** This is the code for the main process control. In this part, it first defines the parameters of the dataloader, generator and discriminator, and then generates some real data with target_lstm. Then it starts to pre-train the generator and discriminator. And last, it starts to do the adversarial training.

4.2 Encoding Pipeline

The key issue we encountered is that how to transfer data with features into the input of SeqGAN, even though we had the financial data and SeqGAN technology. So we came up with the idea that we can encode the LOB data and use one number to represent it. Below is our flow.

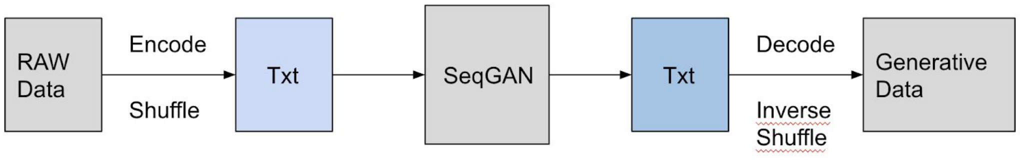
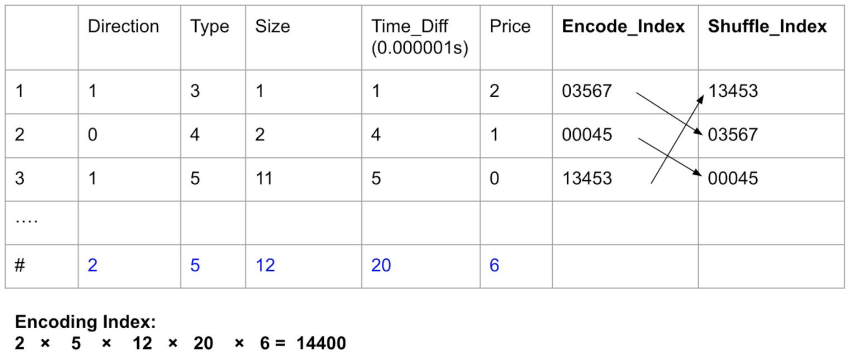


Figure 11: Model Pipeline

First, we encoded our data by our encoding system, which means we encode 5 features into one number. If you got the one number, you can indicate which one data we are pointing to. We have 5 features: *Direction, Type, Size, Time Difference, Price*. They have *2, 5, 12, 20, 6* buckets, respectively, after we categorize them. Then, we combined these 5 numbers to one index number. We also use shuffle to address the feature order problem. By giving the randomness to the index, we can treat different situations equally now. As you can see in the figure 12, we all have 14,400 situations now.



	Direction	Type	Size	Time_Diff (0.000001s)	Price	**Encode_Index**	**Shuffle_Index**
1	1	3	1	1	2	03567	13453
2	0	4	2	4	1	00045	03567
3	1	5	11	5	0	13453	00045
….							
#	2	5	12	20	6		

Encoding Index:
2 × 5 × 12 × 20 × 6 = 14400

Figure 12: Encoding System

Then we arranged numbers to 20 as a sequence and stored it to a txt file. And we treated the txt file as an input and gave it into our model, then we got the txt output file. Then we reversed our encode system and encoded it to become our generative data.

4.3 Result

Here is our result of our first model. As you can see, the left one is the distribution of RAW LOB data (represented by the numbers). And on the right, we got the generative data (also represented by the numbers). As a result, we can see that our generative data nicely captured the distribution of our raw data, which means our model is successful in the method.

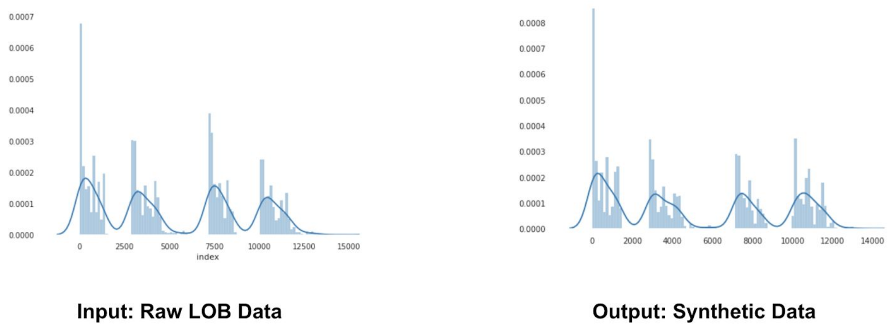


Figure 13: Results of Model

We collected 100,000 LOB data, and generated 200,000 synthetic data. Following is the result of marginal distribution (Categorical Data).

1. **Direction**: We perfectly capture the Direction Distribution.

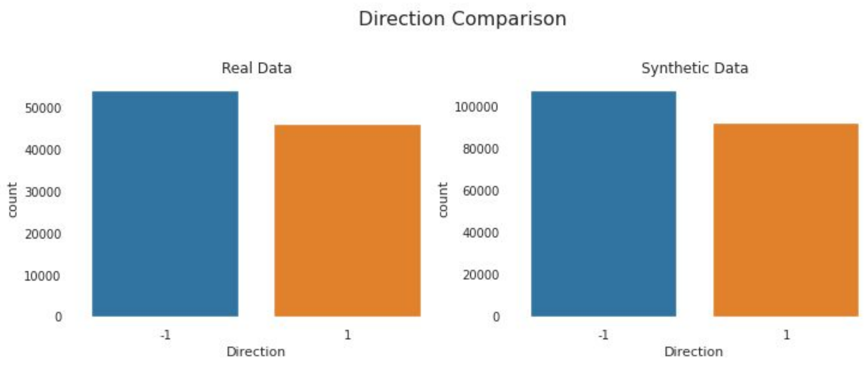


Figure 14: Direction Comparison

2. **Type**: We nicely capture the Type distribution. We can see more type 2 and 5 due to the more data on the synthetic side.

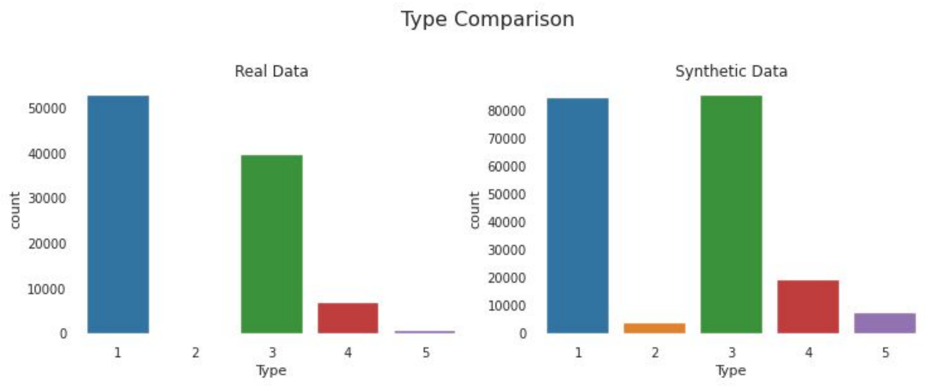


Figure 15: Type Comparison

3. **Size**: We can see the clear pattern in size feature. Most people buy 100, and size 200, 1000, 1200 is the common buying option.

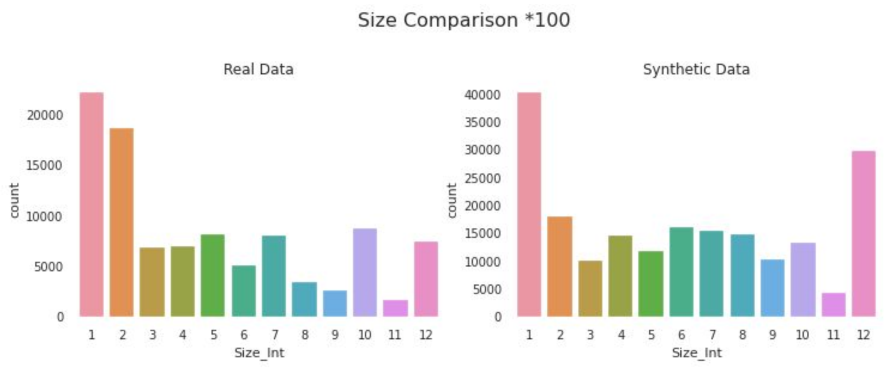


Figure 16: Size Comparison

4. **Time Difference**: There is a clear distribution for time difference here. Most time difference of transactions is 0.000001 second. But we also can see the increase from 0.000010 to 0.000014.

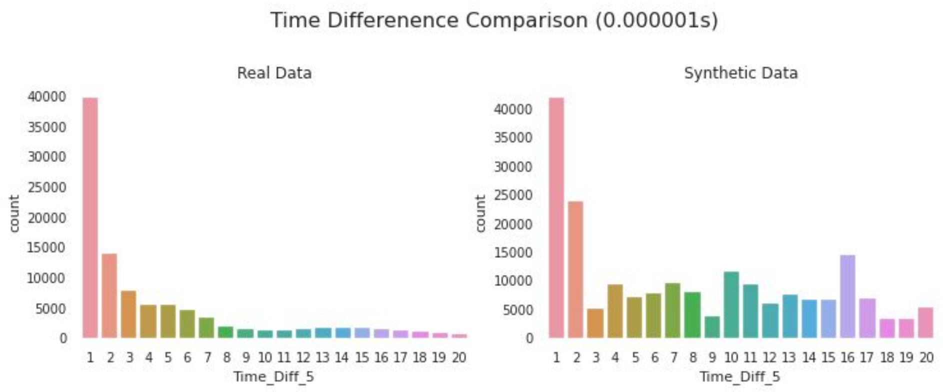


Figure 17: Time Difference Comparison

5. Future Work

To practice the Seq GAN model smoothly for synthesizing data, one should encode the data first. Until now, we have adopted our encoding system then shuffle, however, one hot encoding might also be an efficient alternative approach to try for the encoding problem. Trying to make the result better we are planning to tune the parameters of our Seq GAN model in our next step. That is, to get a better result, we still need to optimize the numbers-size for one sequence as well as determining the bin size for the features in our data based on what we have built previously.

On the other hand, there are still some powerful GANs structures that could be used as synthesizing financial data. The similarities between the real data and synthesized data might be enhanced if we apply different types and structures of GANs. We will also figure out how to predict continuous features and categorical features at the same time. Right now, our Seq-GAN can only deal with categorical features and it can only deal with continuous features for Regular GAN.

Reference:

[1] Li, Jun-Yi, Xintong Wang, Yaoyang Lin, Arunesh Sinha and Michael P. Wellman. “Generating Realistic Stock Market Order Streams.” AAAI 2020 (2020).

[2] Leangarun, Teema, Poj Tangamchit, and Suttipong Thajchayapong. “Stock Price Manipulation Detection Using Generative Adversarial Networks.” *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018. https://doi.org/10.1109/ssci.2018.8628777.

[3] Vyetrenko, Svitlana, David Byrd, Nick Petosa, Mahmoud Mahfouz, Danial Dervovic, Manuela Veloso, and Tucker Hybinette Balch. "Get Real: Realism Metrics for Robust Limit Order Book Market Simulations." arXiv preprint arXiv:1912.04941 (2019).

[4] Wiese, Magnus, Lianjun Bai, Ben Wood, and Hans Buehler. “Deep Hedging: Learning to Simulate Equity Option Markets.” *SSRN Electronic Journal*, 2019. https://doi.org/10.2139/ssrn.3470756.

[5] Wiese, Magnus, Robert Knobloch, Ralf Korn, and Peter Kretschmer. “Quant GANs: Deep Generation of Financial Time Series.” *Quantitative Finance*, June 2020, 1–22. https://doi.org/10.1080/14697688.2020.1730426.

[6] Yu, Lantao, Weinan Zhang, Jun Wang, and Yong Yu. "Seqgan: Sequence generative adversarial nets with policy gradient." In Thirty-First AAAI Conference on Artificial Intelligence. 2017.

[7] Zhang, Kang, Guoqiang Zhong, Junyu Dong, Shengke Wang, and Yong Wang. "Stock Market Prediction Based on Generative Adversarial Network." *Procedia Computer Science* 147 (2019): 400–406. https://doi.org/10.1016/j.procs.2019.01.256.

[8] Zhou, Xingyu, Zhisong Pan, Guyu Hu, Siqi Tang, and Cheng Zhao. "Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets." *Mathematical Problems in Engineering* 2018 (2018): 1–11. https://doi.org/10.1155/2018/4907423.