Generating Realistic Stock Market Order Streams

Junyi Li, Xintong Wang, Yaoyang Lin, Arunesh Sinha, Michael P. Wellman

Abstract

We propose an approach to generate realistic and high-fidelity stock market data based on generative adversarial networks (GANs). Our Stock-GAN model employs a conditional Wasserstein GAN to capture history dependence of orders. The generator design includes specially crafted aspects including components that approximate the market's auction mechanism, augmenting the order history with order-book constructions to improve the generation task. We perform an ablation study to verify the usefulness of aspects of our network structure. We provide a mathematical characterization of distribution learned by the generator. We also propose statistics to measure the quality of generated orders. We test our approach with synthetic and actual market data, compare to many baseline generative models, and find the generated data to be close to real data.

1 Introduction

Financial markets are among the most well-studied and closely watched complex multiagent systems in existence. Well-functioning financial markets are critical to the operation of a complex global economy, and small changes in the efficiency or stability of such markets can have enormous ramifications. Accurate modeling of financial markets can support improved design and regulation of these critical institutions. There is a vast literature on financial market modeling, though still a large gap between the state-of-art and the ideal. Analytic approaches provide insight through highly stylized model forms. Agent-based models accommodate greater dynamic complexity, and are often able to reproduce "stylized facts" of real-world markets [LeBaron, 2006]. Currently lacking, however, is a simulation capable of producing market data at high fidelity and high realism. Our aim is to develop such a model, to support a range of market design and analysis problems. This work provides a first step, learning a high-fidelity generator from real stock market data streams.

Our *main contribution* is Stock-GAN: an approach to produce realistic stock market order streams from real market data. We utilize a conditional Wasserstein GAN (WGAN) [Arjovsky *et al.*, 2017] to capture the time-dependence of order streams, with both the generator and critic conditional on history of orders. The *main innovation* in the Stock-GAN network architecture lies in two deliberately crafted features of the generator. The first is a separate neural network that is used to approximate the double auction mechanism underlying stock exchanges. This pre-trained network enables the generator to model order processing and transaction generation. The second feature is inclusion of order-book information in the conditioning history of the network. The order book captures the key features of market state that are not directly apparent from order history segments.

Our second contribution is a mathematical characterization of the distribution learned by the generator. We show that our designed generator models the stock market data stream as arising from a stochastic process with finite memory dependence. The stochastic process view also makes precise the conditional distribution that the generator is learning as well the joint distribution that the critic of the WGAN distinguishes by estimating the earth mover's distance.

Finally, we experiment with synthetic and real market data. The synthetic data is produced using a stock market simulator that has been used in several agent-based financial studies, but is far from real market data. The real market data was obtained from OneMarketData, a financial data provider. We *propose* five statistics for evaluating stock market data, such as the distribution of price and quantity of orders, inter-arrival times of orders, and the best bid and best ask evolution over time. We compare against other baseline generative models such as *recurrent conditional* variational auto-encoder (VAE) and DCGAN instead of WGAN within Stock-GAN. We perform an ablation study showing the usefulness of our generator structure design as elaborated above. Overall, Stock-GAN is able to best generate realistic data. An online appendix provides all additional results and code for our work: https://bit.ly/2GTgYgA.

2 Related Work and Background

WGAN is a well-known GAN variant [Goodfellow *et al.*, 2014]. Most prior work on generation of sequences using GANs has been in the domain of text generation [Press *et al.*, 2017; Zhang *et al.*, 2017]. However, since the space of word representations is not continuous, the semantics change with nearby word representation, and given a lack

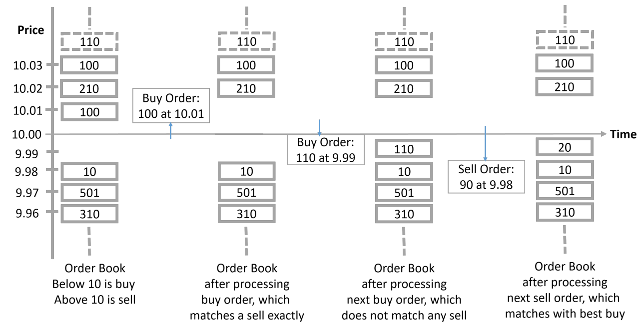


Figure 1: Visual representation and evolution of a limit order book.

of agreement on the metrics for measuring goodness of sentences, producing good quality text using GANs is still an active area of research. Stock market data does not suffer from this representation problem but the history dependence for stock markets can be much longer than for text generation. In a sequence of recent papers, Xiao et al. [2017; 2018] have introduced GAN-based methods for generating point processes. The proposed methods generate the time for when the next event will occur, including application to generate the time for transaction events in stock markets. Our problem is richer as we aim to generate the actual limit orders including time, order type, price, and quantity information.

Deep neural networks and machine learning techniques have been used on financial data mostly for prediction of transaction price [Hiransha *et al.*, 2018; Bao *et al.*, 2017; Qian, 2017] and for prediction of actual returns [Abe and Nakayama, 2018]. As stated, our goal is not market prediction per se, but rather market modeling. Whereas the problems of learning to predict and generate may overlap (e.g., both aim to capture regularity in the domain), the evaluation criteria and end product are quite distinct. GANs have been used for generation of customer buy orders in e-commerce setting [Shi *et al.*, 2019; Kumar *et al.*, 2018], however, stock market orders are much more complex with buys, sells, and cancellations; further we attempt to ensure realism of higher level dynamics like the best bid and ask evolution over time.

Limit order books Nearly all stock markets follow the *continuous double auction* (CDA) mechanism [Friedman, 1993]. Traders submit bids, or *limit orders*, specifying the maximum price at which they would be willing to buy a specified quantity of a stock, or the minimum price at which they would be willing to sell a quantity. The *order book* is a store that maintains the set of active orders: those submitted but not yet transacted or canceled. CDAs are continuous in the sense that when a new order matches an existing (incumbent) order in the order book, the market clears immediately and the trade is executed at the price of the incumbent order—which is then removed from the order book. Orders may be submitted at any time, and a buy order matches and transacts with a sell order when their respective limits are mutually satisfied. For example, as shown in Fig. 1, if a buy order with price \$10.01 and quantity 100 arrives and the best sell offer in the order book has the same price and quantity, then they match exactly and transact. As shown, the next buy order does not match any sell, and the following sell order partially matches what is then the best buy in the order book.

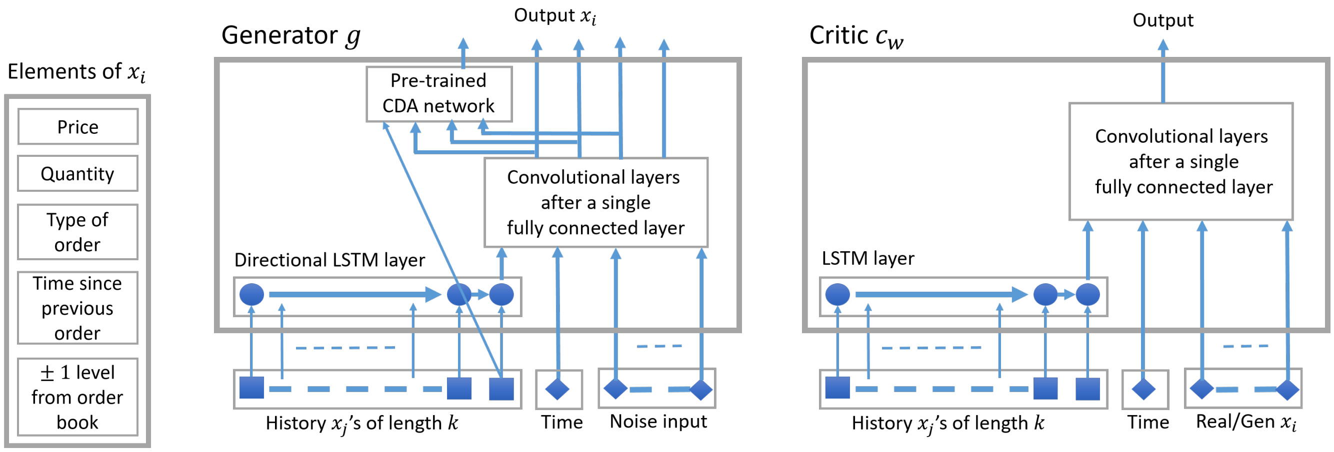
The limit order book maintains the current active orders in the market (or the state of the market), which can be described in terms of the quantity offered to buy or sell across the range of price levels. Each order arrival changes the market state, recorded as an update to the order book. After processing any arrived order every buy price level is higher than all sell price levels, and the *best bid* refers to the lowest buy price level and the *best ask* refers to the highest sell price level. See Fig. 1 for an illustration. The order book is often approximated by few (e.g., ten) price levels above the best bid and ten price levels below the best ask; as these prices are typically the ones that dictate the transactions in the market. There are various kinds of traders in a stock market, ranging from individual investors to large investing firms. Thus, there is a wide variation in the nature of orders submitted. We aim to generate streams of orders that are close in aggregate (not per agent) to real order streams for a given stock.

3 Stock-GAN

We view the stock market orders for a given chunk of time of day Δt as a collection of vector valued random variable $\{\mathbf{x}_i\}_{i \in N}$ indexed by the limit order sequence number in $N = \{1, \ldots, n\}$. $\{\mathbf{x}_i\}$ corresponds to the i^{th} limit order, but, includes more information than the limit order such as the current best bid and best ask. The components of the random vector $\mathbf{x}_i$ include the time interval d_i, type of order t_i, limit order price p_i, limit order quantity q_i, and the best bid a_i and best ask b_i. The time interval d_i specifies the difference in time between the current order i and previous order $i-1$ (in precision of milliseconds); the range of d_i is finite. The type of order can be buy, sell, cancel buy, or cancel sell (represented in two bits). The price and quantity are restricted to lie within finite bounds. The price range is discretized in units of US cents and the quantity range is discretized in units of the equity (non-negative integers). The best bid and best ask are limit orders themselves and are specified by price and quantity. We divide the time in a day into 24 equal intervals and Δt refers to the index of the interval. A visual representation of $\mathbf{x}_i$ is shown in Fig. 2(a).

3.1 Architecture

The architecture is shown in Fig. 2. We use a conditional WGAN [Mirza and Osindero, 2014] with both the generator and critic conditioned on a k length history of $\mathbf{x}_i$'s and the time interval $\Delta \boldsymbol{t}$. We choose $k = 20$. The history is condensed to one vector using a single LSTM layer. This vector and uniform noise of dimension 100 is fed to a fully connected layer followed by 4 convolution layers (see Appendix B for a precise description in form of the code for generator and critic). The generator outputs the next limit order and the critic outputs a real number. Note that when training both generator and critic are fed history from real data, but when the generator executes after training it is fed its own generated data as history. The generator also outputs the best bid and ask. Recall from our description of the stock market that the best bid and ask can be inferred deterministically from the current order and the previous best bid and



(a) $\mathbf{x}_i$ (b) Generator (c) Critic

Figure 2: Stock-GAN architecture

ask (for most orders), we use another neural network (with frozen weights during GAN training) to output the best bid and ask. We call this the CDA network. The CDA network has a fully connected layer layer followed by 3 convolutional layers. Its input is a limit order and the current best bid/ask and the output is the next best bid/ask. The CDA network is trained separately using the orders and order-book data using a standard mean squared error loss (Appendix B has the CDA network code that precisely describes the structure and loss).

The loss function used to train the WGAN is the standard WGAN loss with a gradient penalty term [Gulrajani *et al.*, 2017]. The critic is trained 100 times in each iteration. The *notable* part in constructing the training data is that for each of 64 data points in a mini-batch the sequence of orders chosen (including history) is far away from any other sequence in that mini-batch. This is to break the dependence among data points for the history dependent stock market data. We make this mathematically precise in the next section.

3.2 Mathematical Characterization of Stock-GAN

Stock-GAN models stock market orders as a stochastic process. Recall that a stochastic process is a collection of random variables indexed by a set of numbers. As stated earlier, we view the stock market orders for a given chunk of time of day Δt as a collection of vector valued random variable $\{\mathbf{x}_i\}_{i \in N}$ indexed by the limit order sequence number in $N = \{1, \ldots, n\}$. Following the terminology prevalent for stochastic processes, the above process is discrete time and discrete space (note that discrete time in this terminology here refers to the discreteness of the index set N).

The k length history we use implies a finite history dependence of the current output $\mathbf{x}_i$, that is, $P(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \ldots, \Delta\mathrm{t}) = P(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \ldots, \mathbf{x}_{i-m}, \Delta\mathrm{t})$ for some m. Such dependence is justified by the observation that recent orders mostly determine the transactions and transaction price in the market as orders that have been in the market for long either get transacted or canceled. Further, the best bid and best ask serves as an (approximate) sufficient statistic for events beyond the history length m. While this process is not a Markov chain, it forms what is known as a higher order Markov chain, which implies that the process given by $\mathbf{y}_i = (\mathbf{x}_i, \ldots, \mathbf{x}_{i-m+1})$ is a Markov chain for any given time interval Δt. We assume that this chain formed by $\mathbf{y}_i$ has a stationary distribution (i.e., it is irreducible and positive recurrent). A Markov chain is a *stationary stochastic process* if it starts with its stationary distribution. After some initial mixing time, the Markov chain does reach its stationary distribution, thus, we assume that the process is stationary by throwing away some initial data for the day. Also, for the jumps across two time intervals Δt, we assume the change in stationary distribution is small and hence the mixing happens very quickly. A stationary process means that $P(\mathbf{x}_i, \ldots, \mathbf{x}_{i-m+1} \mid \Delta\mathrm{t})$ has the same distribution for any i. In practice we do not know m. However, we assume that our choice k satisfies $k + 1 > m$, and then it is straightforward to check that $\mathbf{y}_t = (\mathbf{x}_i, \ldots, \mathbf{x}_{i-k})$ is a Markov chain and the claims above hold with $m - 1$ replaced by k.

Given the above stochastic process view of the problem, we show that the generator aims to learn the real conditional distribution $P_r(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \ldots, \mathbf{x}_{i-k}, \Delta\mathrm{t})$. We use the subscript r to refer to real distributions and the subscript g to refer to generated distributions. The real data $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ is a realization of the stochastic process. It is worth noting that even though $P(\mathbf{x}_i, \ldots, \mathbf{x}_{i-k} \mid \Delta\mathrm{t})$ has the same distribution for any i, the realized real data sequence $\boldsymbol{x}_i, \ldots \boldsymbol{x}_{i-k}$ is correlated with any overlapping sequnce $\boldsymbol{x}_{i+k'}, \ldots \boldsymbol{x}_{i-k+k'}$ for $k \geq k' \geq -k$. Our data points for training are sequences $\boldsymbol{x}_i, \ldots \boldsymbol{x}_{i-k}$ and as stated earlier we make sure that the sequences chosen in a batch are sufficiently far apart. In light of the interpretation above, this ensures independence of data points within a batch.

Critic interpretation: When fed real data, the critic can be seen as a function c_w of $\boldsymbol{s}_i = (\boldsymbol{x}_i, \ldots, \boldsymbol{x}_{i-k}, \boldsymbol{\Delta t})$, where w are the weights of the critic network. As argued earlier, samples in a batch that are chosen from real data that are spaced at least k apart are i.i.d. samples of P_r. Then for m samples fed to the critic, $\frac{1}{m}\sum_{i=1}^{m} c_w(\boldsymbol{s}_i)$ estimates $E_{\mathbf{s}\sim P_r}(c_w(\mathbf{s}))$. When fed generated data (with the ten price levels determined from the output order and previous ten levels), by similar reasoning $\frac{1}{m}\sum_{i=1}^{m} c_w(\boldsymbol{s}_i)$ estimates $E_{\mathbf{s}\sim P_g}(c_w(\mathbf{s}))$ when the samples are sufficiently apart (recall that the history is always real data). Thus, the critic computes the Wasserstein distance between the joint distributions $P_r(\mathbf{x}_i, \ldots, \mathbf{x}_{i-k}, \Delta\mathrm{t})$ and $P_g(\mathbf{x}_i, \ldots, \mathbf{x}_{i-k}, \Delta\mathrm{t})$.

Generator interpretation: The generator learns the conditional distribution $P_g(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \ldots, \mathbf{x}_{i-k}, \Delta\mathrm{t})$. Along with the real history, the generator represents the distribution $P_g(\mathbf{x}_i, \ldots, \mathbf{x}_{i-k}, \Delta\mathrm{t}) = P_g(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \ldots, \mathbf{x}_{i-k}, \Delta\mathrm{t}) P_r(\mathbf{x}_{i-1}, \ldots, \mathbf{x}_{i-k}, \Delta\mathrm{t})$.

4 Experimental Results

Evaluating generative models is an inherently challenging task, even in the well-established domain of image generation [Borji, 2019]. To the best of our knowledge, we are the first to generate limit order streams in stock market and as part of the contribution in this paper we propose to measure the quality of generated data using five statistics. These statistics capture various aspects of order streams observed in stock markets that are often studied in finance literature.

Our five proposed statistics are (1) Price. Distribution over price for the day's limit orders, by order type. (2) Quantity. Distribution over quantity for the day's limit orders, by order type. (3) Inter-arrival time. Distribution over inter-arrival duration for the day's limit orders, by order type. (4) Intensity evolution. Number of orders for consecutive T-second chunks of time. (5) Best bid/ask evolution. Changes in the best bid and ask over time as new orders arrive.

For each of these statistics, we also present various quantitative numbers to measure the quality. Due to lack of space, in the main paper the results for price, quantity, inter-arrival distributions are shown only for buy orders. The results for the other types are very similar to buy type results and are presented in the online appendix.

4.1 Synthetic Data

We first evaluate Stock-GAN on synthetic orders generated from an agent-based market simulator. Previously adopted to study a variety of issues in financial markets (e.g., market making [Wah *et al.*, 2017] and manipulation [Wang *et al.*, 2018]), the simulator captures stylized facts of the complex financial market with specified stochastic processes and distributions [Wellman and Wah, 2017]. However, the simulator is still very basic and quite far from real market data. For example, fundamental valuation shocks are generated from a fixed Gaussian distribution (Fig. 4(a)) and quantity is always 1 (Fig. 4(b)), whereas the real market data distributions can be seen to be quite non-smooth (Figs. 5(a)- 5(c)). Thus, we use the output of this basic simulator as our synthetic data (which we call as real in results below).

We use about 300,000 orders generated by the simulator as our synthetic data. These orders are generated over a horizon of 1000 seconds, but the actual horizon length is not important for synthetic data as it can be scaled arbitrarily. The price output by the simulator is normalized to lie in $[-1, 1]$, which is the reason for negative prices in the synthetic data.

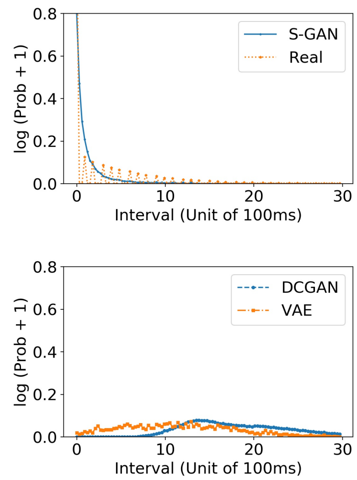


Figure 3: Synthetic inter-arrival dist.

Stock-GAN and baselines: Our first results show the performance of Stock-GAN (S-GAN in graphs) and compares it to baselines, namely to a recurrent variational autoencoder [Chung *et al.*, 2015] (VAE) and the same network as ours, except using a DCGAN [Radford *et al.*, 2015] instead of WGAN. We show results for price distribution (Fig. 4(a)), quantity distribution (Fig. 4(b)), and inter-arrival distribution (Fig. 3—shown in two graphs for clarity). The results show that VAE and DCGAN produce distributions far from the real one. We capture these differences quantitatively using the Kolmogorov-Smirnoff (KS) distance:

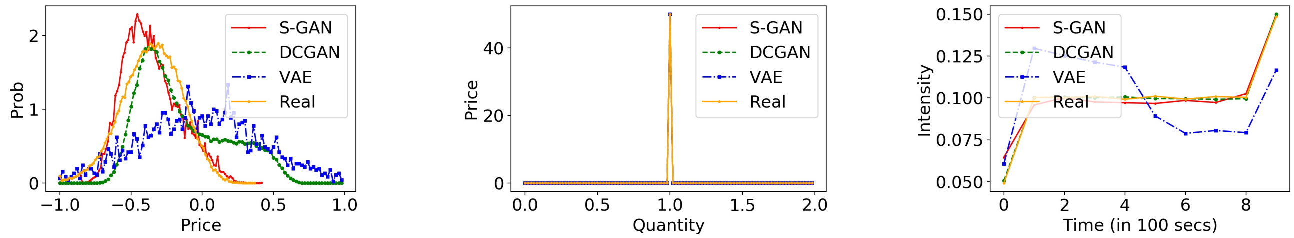
	Real,S-GAN	Real,VAE	Real,DCGAN
Price	0.108	0.502	0.284
Inter-arrival	0.18	0.756	0.923

Table 1: KS distances against real (synthetic)

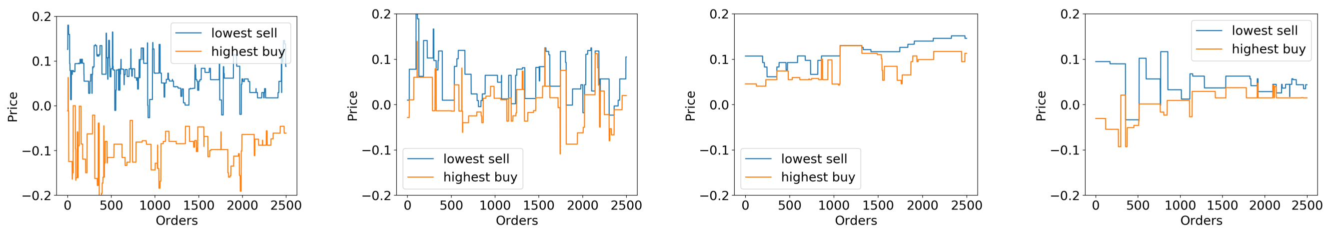
The KS distance always lies between $[0, 1]$. We skip the quantity KS distance as the quantity is always trivially one in the synthetic data. The much smaller KS distance between real and Stock-GAN supports our claim of better performance of our approach compared to VAE and DCGAN.

For intensity, we choose $T = 100$ seconds sized chunks of time and measure intensity as the number of orders in each chunk divided by the total number of orders. Fig. 4(c) shows that VAE completely fails to match the synthetic data intensity. DCGAN has the same flat intensity throughout, whereas Stock-GAN matches the real intensity very closely.

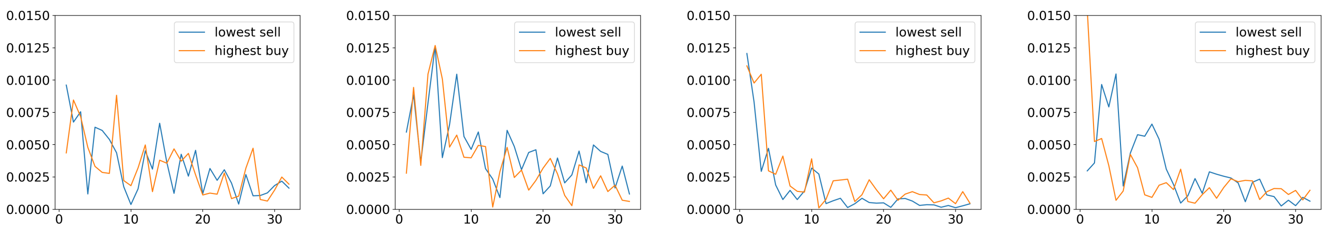
Ablation: The real and Stock-GAN generated best bid/ask evolutions are in Fig. 4(d) and 4(e) respectively. We perform two ablation experiments, one by removing the CDA network (no cda) and one by removing order-book information (no ob), shown in Figs. 4(f) and 4(g) respectively. Differences can be seen in the means of best bid/ask for no cda and no ob compared to the real and Stock-GAN results, but the main distinction is in the spectral densities for these time series shown in Figs. 4(h)–4(k). It can be seen that no cd and no ob have much fewer higher frequency components as compared to real synthetic spectral density, which can also be seen by the smoother variation over time in Figs. 4(f) and 4(g). Stock-GAN's and the real synthetic spectral density match more closely.



(a) Synthetic price distribution (b) Synthetic quantity distribution (c) Synthetic intensity



(d) Synthetic best bid/ask (e) Stock-GAN best bid/ask (f) no CDA network best bid/ask (g) no order book best bid/ask



(h) Synthetic spectral density (i) Stock-GAN spectral density (j) no CDA spectral density (k) no order book spectral density

Figure 4: A comparison of different statistics for generated and real synthetic limit orders. Additional results are in appendix.

4.2 Real Data

We obtained real limit-order streams from OneMarketData, who provided access to their OneTick database for selected time periods and stocks. The provided data streams comprise order submissions and cancellations at millisecond granularity. In experiments, we evaluate the performance of Stock-GAN on a large capitalization stock, Alphabet Inc (GOOG). We also tried a small capitalization stock Patriot National (PN), which has much fewer orders than GOOG. After preprocessing, the PN daily order stream has about 20,000 orders and GOOG has about 230,000. Hence, naturally PN is not a good fit for learning using data hungry neural networks and our results for PN validate this claim. We show the results for PN in the appendix.

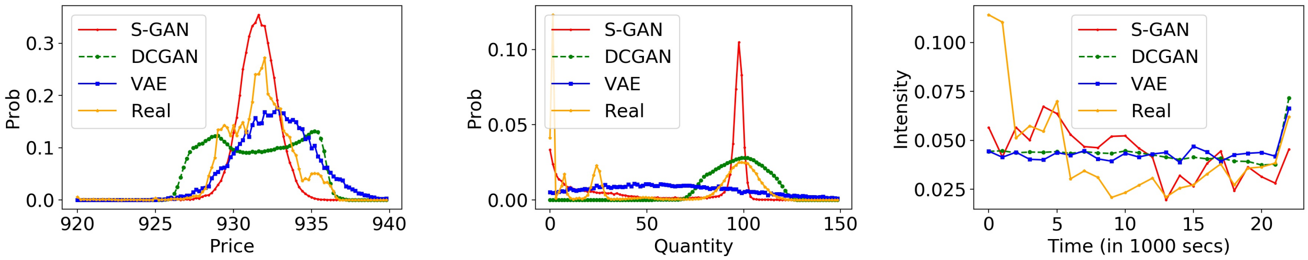
Relative to synthetic data, the real market data is very noisy including many orders at extreme prices far from the range where transactions occur. Since our interest is primarily on behavior that can affect market outcomes, we focus on orders in the relevant range near the best bid and ask. Specifically, in a preprocessing step, we eliminate limit orders that never appear within ten levels of the best bid and ask prices. In the experiment here, we use historical market data of GOOG during one trading day in Aug 2017. Our results for GOOG follow the same evaluation metrics as for synthetic data.

Stock-GAN and baselines: We show the performance of Stock-GAN and compare it to VAE and DCGAN variant of our network. We show these results for price distribution (Fig. 5(a)), quantity distribution (Fig. 5(b)), and inter-arrival times (Fig. 6). As earlier, we capture these differences quantitatively using the KS distance:

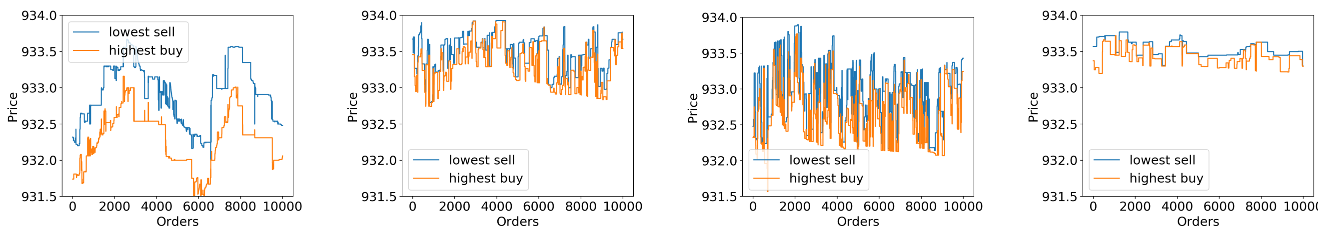
	Real,S-GAN	Real,VAE	Real,DCGAN
Price	0.126	0.218	0.181
Quantity	0.182	0.248	0.471
Inter-arrival	0.066	0.835	0.154

Table 2: KS distances against real (GOOG)

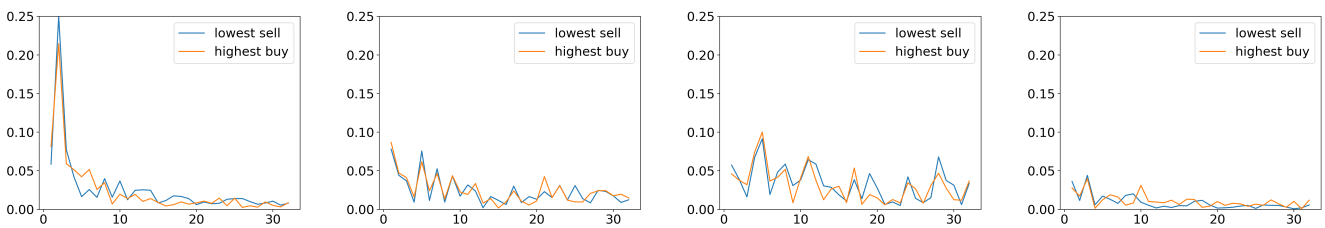
Similar to synthetic data results, the numbers above reveal that Stock-GAN is able to model GOOG data better than the baselines. Intensity is measured in the same way as synthetic data, except that we choose $T = 1000$ seconds sized chunks of time due to the longer horizon of GOOG data. Fig. 5(c) shows much smoother intensity produced by VAE and DCGAN as opposed to Stock-GAN which is much closer to the real intensity.



(a) GOOG price distribution (b) GOOG quantity distribution (c) GOOG intensity



(d) GOOG best bid/ask (e) Stock-GAN best bid/ask (f) no CDA network best bid/ask (g) no order book best bid/ask



(h) GOOG spectral bid/ask (i) Stock-GAN spectral density (j) no CDA spectral density (k) no order book spectral density

Figure 5: A comparison of different statistics for generated and real GOOG limit orders. Additional results are in appendix.

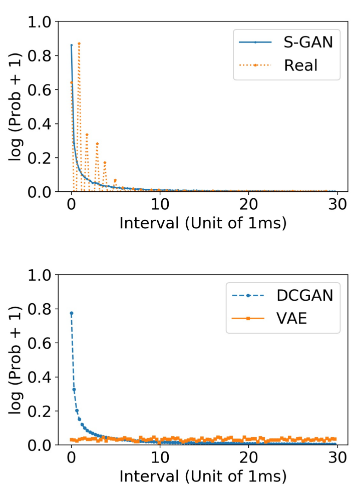


Figure 6: GOOG inter-arrival dist.

Ablation: The real and StockGAN generated best bid/ask evolution are in Figs 5(d) and 5(e) respectively. As for synthetic data, we perform two ablation experiments, one by removing the CDA network (no cda) and one by removing order-book information (no ob), shown in Fig 5(f) and 5(g) respectively. The main distinction again is seen in the spectral densities for these time series shown in Figs. 5(h)-5(k). However, unlike the synthetic data, here it can be seen that no cda has more higher frequency components that real data, which can also be seen by the high variation over time in Fig 5(f). On the other hand, no ob has less higher frequency (or even lower frequency) components which results in the flat shape in Fig. 5(g). The Stock-GAN spectral density, while closest to real one among all alternatives, also misses out on some low frequency components. Nonetheless, Stock-GAN is closest to real data due to our novel structural approach of the CDA network and use of order-book data.

5 Conclusion

We showed the superior performance of Stock-GAN in producing realistic market order streams. However, we point out that we chose our data for dates in which there were no external events, such as financial performance report. Thus, we did not model the effect of exogenous factors on stock market, which we believe is technically possible by just adding another condition for the generator. Notwithstanding these effects, we demonstrated that stock market data can be generated with high fidelity.

References

[Abe and Nakayama, 2018] Masaya Abe and Hideki Nakayama. Deep learning for forecasting stock returns in the cross-section. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 273–284, 2018.

[Arjovsky *et al.*, 2017] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *34th International Conference on Machine Learning*, pages 214–223, 2017.

[Bao *et al.*, 2017] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS One*, 12(7):e0180944, 2017.

[Borji, 2019] Ali Borji. Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[Chung *et al.*, 2015] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

[Friedman, 1993] Daniel Friedman. The double auction market institution: A survey. *The Double Auction Market Institutions, Theories and Evidence, Addison Wesley*, 1993.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[Gulrajani *et al.*, 2017] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[Hiransha *et al.*, 2018] M. Hiransha, E. A. Gopalakrishnan, Vijay Krishna Menon, and K. P. Soman. NSE stock market prediction using deep-learning models. *Procedia Computer Science*, 132:1351 – 1362, 2018. International Conference on Computational Intelligence and Data Science.

[Kumar *et al.*, 2018] Ashutosh Kumar, Arijit Biswas, and Subhajit Sanyal. ecommercegan: A generative adversarial network for e-commerce. *arXiv preprint arXiv:1801.03244*, 2018.

[LeBaron, 2006] Blake LeBaron. Agent-based computational finance. In Leigh Tesfatsion and Kenneth L. Judd, editors, *Handbook of Computational Economics*. Elsevier, 2006.

[Mirza and Osindero, 2014] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[Press *et al.*, 2017] Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. Language generation with recurrent generative adversarial networks without pre-training. *arXiv preprint arXiv:1706.01399*, 2017.

[Qian, 2017] Xin-Yao Qian. Financial series prediction: Comparison between precision of time series models and machine learning methods. *arXiv preprint arXiv:1706.00948*, 2017.

[Radford *et al.*, 2015] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[Shi *et al.*, 2019] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *AAAI*, 2019.

[Wah *et al.*, 2017] Elaine Wah, Mason Wright, and Michael P. Wellman. Welfare effects of market making in continuous double auctions. *Journal of Artificial Intelligence Research*, 59:613–650, 2017.

[Wang *et al.*, 2018] Xintong Wang, Yevgeniy Vorobeychik, and Michael P. Wellman. A cloaking mechanism to mitigate market manipulation. In *27th International Joint Conference on Artificial Intelligence*, pages 541–547, 2018.

[Wellman and Wah, 2017] Michael P. Wellman and Elaine Wah. Strategic agent-based modeling of financial markets. *Russell Sage Foundation Journal of the Social Sciences*, 3(1):104–119, 2017.

[Xiao *et al.*, 2017] Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models. In *Advances in Neural Information Processing Systems*, pages 3247–3257, 2017.

[Xiao *et al.*, 2018] Shuai Xiao, Hongteng Xu, Junchi Yan, Mehrdad Farajtabar, Xiaokang Yang, Le Song, and Hongyuan Zha. Learning conditional generative models for temporal point processes. In *32nd AAAI Conference on Artificial Intelligence*, 2018.

[Zhang *et al.*, 2017] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*, 2017.