MAS-GAN: ADVERSARIAL CALIBRATION OF MULTI-AGENT MARKET SIMULATORS

Anonymous authors
Paper under double-blind review

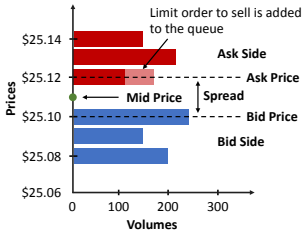
ABSTRACT

We look at the problem of how the simulation of a financial market should be configured so that it most accurately emulates the behavior of a real market. In particular, we address agent-based simulations of markets that are composed of many hundreds or thousands of trading agents. A solution to this problem is important because it provides a credible test bed for evaluating potential trading algorithms (e.g., execution strategies). Simple backtesting of such algorithms suffers from a critical weaknesses, chiefly that the overall market is not responsive to the candidate trading algorithm. Multi-agent simulations address this weakness by simulating *market impact* via interaction between market participants. Calibration of such multi-agent simulators to ensure realism, however, is a challenge. In this paper, we present MAS-GAN – a multi-agent simulator calibration method that allows to tune simulator parameters and to support more accurate evaluations of candidate trading algorithm. Our calibration focus is on high level parameters such as the relative proportions of the various types of agents that populate the simulation. MAS-GAN is a two-step approach: first, we train a discriminator that is able to distinguish between "real" and "fake" market data as a part of GAN with self-attention, and then utilize it within an optimization framework to refine simulation parameters. The paper concludes with quantitative examples of applying MAS-GAN to improve simulator realism.

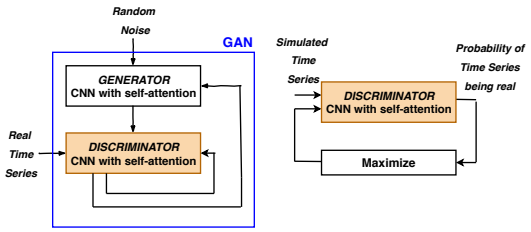
1 INTRODUCTION

1.1 MOTIVATION

Increasingly large market volumes are traded today electronically across multiple asset classes on public exchanges. Development and testing of trading agents for electronic limit order book markets rely on availability of high-quality market simulators. While a traditional approach is to test electronic trading agents against historical data (i.e., backtest), doing so does not consider interaction of the trading agent with other market participants (i.e. *market impact*), which can result in poor performance estimate of an agent and manifest in trading losses once the agent is deployed to production environment. Multi-agent simulation is a paradigm that is conceptually closer to real markets as it allows market impact to emerge as a result of interactions of fully autonomous agents. The challenge is, however, to find realistic agent configurations and prescribe agent behavior in such a way that their actions produce synthetic time series whose statistical properties resemble real markets.



(a) Visualization of the LOB structure.



(b) Schematic illustration of a two-step MAS-GAN approach: first, we train a GAN; and then use its discriminator as an objective function for simulator calibration.

Figure 1

Public exchanges such as NASDAQ and NYSE facilitate the buying and selling of assets by accepting and satisfying buy and sell orders from multiple market participants. The exchange maintains an

order book data structure for each asset traded. The limit order book (LOB) represents a snapshot of supply and demand for the asset at a given time. It is an electronic record of all the outstanding buy and sell limit orders organized by price levels Bouchaud et al. (2018). A matching engine, such as first-in-first-out (FIFO), is used to match incoming buy and sell order interest. Order types are further distinguished between limit orders and market orders. A limit order specifies a price that should not be exceeded in the case of a buy order (bid), or should not be gone below in the case of a sell order (ask). A limit order queues a resting order in the LOB at the corresponding side of the order book. A market order indicates that the trader is willing to accept the best price available immediately. See Figure 1a for visualization of the LOB structure.

In order to have confidence in developed trading strategies, we need to calibrate parametrized multi-agent limit order book market simulator to be as close as possible to real market data. Since the primary goal of using multi-agent market simulators is to model emergent phenomena (such as market impact of trading) and to test trading strategies in hypothetical scenarios (such as COVID market shock) – the simulated price series to is expected to diverge significantly from its historic trajectory and the traditional calibration methods are difficult to apply. To be realistic, limit order book market simulator quantities should display same distributions as those of the real markets Vyetrenko et al. (2019). Such quantities, that are repeated across a wide range of instruments, markets, and time period, are known as *stylized facts* Cont (2001). Some stylized facts originate from traders' behavior, while others are a natural consequence of order book market matching mechanism. Examples of stylized facts include properties of asset return distributions (e.g., asset return distribution have fat tails), volumes and order flow (e.g., top-of-the-book volumes are gamma-distributed), non-stationary patterns (e.g., trading volumes are higher in the morning and in the evening), etc. Optimization with respect to stylized facts can be difficult as it is challenging to design an explicit optimization objective function due to the overlapping nature of stylized facts and lack of clarity over which stylized facts should be given more optimization weight. Therefore, we propose to learn a discriminator (i.e. a classifier) that can distinguish synthetic time series from real, and use it as an implicitly written objective function for the calibration task.

1.2 Background and related work

Modeling the market as an interplay of multiple agents is a natural approach to mimic real market collective emergent behavior, however, justifying the realism of such approach for validating new trading strategies is difficult. Agent-based modeling typically relies on common sense hand-crafted rules (e.g., Palit et al. (2012)), which can be difficult to calibrate as historical data labeled with details about each individual constituent agent behavior is typically not available for public use. Several calibration approaches—e.g. error minimization to find parameters for the asset pricing model with heterogeneous beliefs Tedeschi et al. (2013) and using Bayesian parameter estimation techniques in the simulated context—have been introduced Grazzini et al. (2015). When individual agent- or execution strategy-specific data is available to the researcher, it can be used for the simulator calibration (e.g., Vyetrenko & Xu (2019); Yang et al. (2012)). Other approaches to multi-agent simulator realism can include inverse learning agents' rewards from the market Yang et al. (2012); learning approximate representation of the agent-based simulator first and then using it for parameter optimization either directly Lamperti et al. (2017) or as a part of generative adversarial network Alonso-Monsalve & Whitehead (2018); incorporating feedback from real-time trading into the simulation Ruiz et al. (2019).

Training of GANs (first introduced in Goodfellow et al. (2014)) is, however, quite fragile with mode collapse (when the generator is only capable of producing a limited diversity of samples) being a frequent problem Salimans et al. (2016). Good quantitative metrics to determine the quality of generated images are an area of active research Theis et al. (2015); there is currently no consensus as to what the best set of metrics is – hence, visual inspection of generated images is still a common recommendation. The concept of adversarial training has also been explored to generate realistic time series, however, visual inspection is both impractical and inappropriate, especially for multi-dimensional time series Esteban et al. (2017). GANs have also been used to synthesize financial time series, where stylized facts were used to check the quality of generated time series Wiese et al. (2020); Li et al. (2020).

Multi-agent simulators that are used in trading can by themselves can be viewed as generative models, albeit typically non-differentiable ones. These simulators are typically governed by a small num-

ber of physically meaningful parameters, changing which can provide explanations of the collective system dynamics. Therefore, it is natural to use GANs with generators replaced by domain-specific simulators to manipulate the simulated time series described by the simulator parameters in order to produce most realistic time series – hence, calibrate multi-agent simulators. For instance, the resulting non-differentiable minimax calibration problem was solved by by minimizing variational upper bounds of the adversarial objectives in Louppe et al. (2017). Similar approach to simulator calibration was used in Alonso-Monsalve & Whitehead (2018) with a difference that an emulator neural network, pretrained to approximate non-differentiable simulator, was used as a part of the GAN generator. One, however, needs to note that generating training data using multi-agent simulations can be very expensive, therefore, the above methods might not be feasible in practice.

Transformer's head mechanisms, introduced in Vaswani et al. (2017) have proved their efficiency at capturing global dependencies in multiple domains of applications such as translation, language modeling or object detection. Among the different attention mechanisms, self attention is computing a representation of the next position in a sequence by looking at all the position in the sequence and learning regions of interest. Using a self-attention mechanism for analysing time series has been widely studied. In time series forecasting, Wu et al. (2020) is exploring how to model complex dynamics of the data using transformer. By doing so, they are not processing the data sequentially and are able to capture more complex structures. The use of self attention is also increasingly popular in generative modeling (e.g., GANs). In Xu et al. (2018), authors propose to apply self-attention to input sequences of words, however, it does not intervene directly with the architecture of the GAN. In Zhang et al. (2019), authors introduce the non-local model of self-attention of Wang et al. (2018) directly into the GAN architecture – method called SAGAN – which enables to take into account important relationships between regions that are widely separated from each other. SAGAN is applied to 2D images using non trivial stabilizers like spectral norm layers to facilitate the training phase. In this paper, we adapt SAGAN architecture to the 1D case to encode for complex relationships between the correlated data and build self-attention mechanism into GANs for financial time series.

Most of the GAN literature focuses on training GANs for the purpose of subsequently using only data generator, and discriminator is only viewed as an auxiliary agent that assists with generator training. To complement this, using GAN-trained discriminator for the purpose of time series anomaly detection is discussed in Schlegl et al. (2017); Mattia et al. (2019); Li et al. (2018); Wang et al. (2019). By learning statistical similarities between non-anomalous time series, one can learn to determine what constitutes an anomaly. In this paper, we rely on a similar idea and introduce an application of adversarially trained discriminators to multi-agent simulator parameter optimization.

1.3 Our contributions

In this paper, we propose MAS-GAN – a two-step method for multi-agent market simulator calibration (introduced in Section 2). A discriminator is trained in competition with the generator as a part of GAN to distinguish synthetic time series from real (i.e. historical). Both generator and discriminator are enhanced by self-attention layers for better time series resolution quality. The input of trained discriminator is a synthetic time series, and the output uses sigmoid activation which allows to interpret it as probability of synthetic time series being real. Similar to Liang et al. (2018), the discriminator score is higher if the synthetic time series shares more features with the historical dataset. One can then use the discriminator score as an implicit optimization objective, and optimize simulated model parameters to determine the set that produces most realistic time series (see Figure 1b for schematic).

Using GANs to generate synthetic time series has been widely studied – however, only LSTMs and recurrent neural network architectures have been previously considered (e.g., Esteban et al. (2017)). To the best of our knowledge, using both conv1D and self-attention without recurrence in the architecture to generate synthetic time series is novel. We use conv1D to encode the local correlations in the data. The deeper the network is, the more global encoding will happen. We add self-attention to provide a more targeted way to encode global correlations with a single layer. A 2D version of the self-attention layer we used is well described in Zhang et al. (2019). We provide an implementation of a 1D version of the self-attention layer in the appendix section along with its visualization in Figure 9.

To the best of our knowledge, MAS-GAN is the first method that proposes to use adversarially trained discriminator as an objective function for multi-agent system optimization. Unlike Alonso-Monsalve & Whitehead (2018), our method is model-agnostic and does not require learning a simulator approximation which can be expensive. Because MAS-GAN learns from historical dataset and is unbiased toward synthetic data coming from the simulator, once the discriminator is trained – it can be used to calibrate any time series model (not necessarily multi-agent) that is intended to model that historical dataset. Since simulated data is not used for GAN training, the discriminator is not biased to any particular model and only uses historical data as a 'ground truth' for learning – this provides an additional justification of a two-step as opposed to one-step calibration method. In contrast with Li et al. (2020), our method retains full explainability of multi-agent simulations as GANs in our case are not used for black-box data generation, but rather only for learning a calibration objective given by the discriminator.

2 ADVERSARIAL PARAMETER OPTIMIZATION

GANs were first introduced in Goodfellow et al. (2014) for the purpose of generating realistic looking images (i.e. learning the generator) by having a generator adversary compete against the discriminator during training. When a human learns to distinguish between real and synthetic data, it first learns to recognize what constitutes real data, and then uses this knowledge to decide which features distinguish real data from synthetic. Hence, in human learning the *Anna Karenina principle* is applied: namely, happy families share a common set of attributes which lead to happiness, while any of a variety of attributes can cause an unhappy family. Similarly, a common set of factors distinguishes a real limit order book price time series, whereas, a synthetic time series can be generated in a variety of different ways. Therefore, we use only real time series to adversarially train a classifier to only recognize features that are pertinent to real time series. Similar approach was argued in Sun et al. (2019) for training a discriminator that can recognize data anomalies.

Most multi-agent simulated systems can be represented by a parameter vector that specifies simulator configuration. For instance, multi-agent market simulator can be described by the number of agents of different types and their strategy parameters such as arrival rates, order sizes, etc. In this section, we describe a generic method to find the best configuration for a given multi-agent simulator given by a parameter vector v, and present an example of optimizing a specific simulated environment in Section 3.3. Additionally, because the final simulated time series depend on choices that agents probabilistically make (i.e., exact time of arrival randomly generated according to the probability distribution with a certain agent arrival rate) – we denote the seed that is used to initialize pseudo-random number generator by R.

Let $S_R(v)$ be the simulated mid price time series obtained using the parameter vector v and random seed R. Since the discriminator D outputs probability of time series $S_R(v)$ being real, we are interested in finding parameter vector v^* that maximizes this probability. Formally, we are interested in finding:

$$v^* = \arg\max_v \mathbb{E}_R \left[D(S_R(v)) \right]. \tag{1}$$

We then perform grid-based optimization with respect to the discriminator-defined objective to find most realistic simulation parameters out of the given set. We note here that our approach does not attempt to improve the simulated model but rather finds a set of parameters such that the existing model describes P_{real} most closely.

3 EXPERIMENTAL RESULTS

3.1 GAN TRAINING DETAILS

In real limit order book markets, multiple correlated data streams are produced as a result of participant interaction (e.g., prices and quotes at multiple limit order book levels and traded volumes). Therefore, capturing temporal autocorrelations and cross-correlations of multiple limit order book time series distributions is important for calibration realism. Specifically, we train MAS-GAN on one-minute mid price returns and cumulative one-minute traded volumes of thirty Dow Jones underlying stocks traded on NASDAQ in June 2019, with first three weeks of June 2019 being insample and last week of June 2019 being an outsample dataset. Note that market conditions in June 2019 were characterized by a period of high liquidity and low price volatility.

We are interested in training a GAN for subsequent use of its discriminator part a simulator calibration objective. The specific architecture is depicted in Figure 10 (in the Appendix) – it is using a self-attention layer both in the discriminator and in the generator together with several layers of conv1D, upsampling and dropouts and Relu. The output layer of generator uses linear activation, whereas, the output layer of discriminator uses sigmoid activation. This architecture choice is inspired by Zhang et al. (2019), although we did not need to add the spectral norm layer to stabilize training. The input to the generator is a Gaussian white noise vector and its dimension (i.e., a latent dimension) is a hyper-parameter. We set the latent dimension to 200 after a grid search optimization (for that, we looked at the distribution of the discriminator scores after 20000 iterations).

There are 358 trading minutes in a day between 10 am and 4 pm. To produce the input vector that is passed to the GAN discriminator, we concatenate the mid price returns (a real vector of length 358) and volumes (an integer vector of length 358). This concatenation produces an input vector of size 716. It is done in order for the neural network with self-attention to encode correlations between price and volume time series and to retranscribe them when generating time series from noise. Mode collapse, when generator output shows little to no variability regardless of the input, is a common problem in GAN training. To avoid the mode collapse, we rescaled the volumes so that their order of magnitude is similar to that of the mid price returns. Moreover, we noticed that for some assets the traded volume vectors are very sparse (due the idiosyncratic trading patterns of those stocks) which can lead to a fast collapse at training time. Therefore, we fixed a threshold at 50% of non-zero entries per sample to set a level of acceptance of training vector sparsity.

We note that in order to train a high quality discriminator, the generator needs to be capable to generate a diverse set of realistic time series. Therefore, we train the GAN iteratively and terminate training when the following conditions are satisfied:

1. Generated time series visually show diversity (to prevent mode collapse) – see Figure 3d. We are aware of the issues raised by Ryan et al. (2019) that visual inspection might not be sufficient to demonstrate diversity as generated time series can be simply memorized by GANs. We will leave this question for further work.

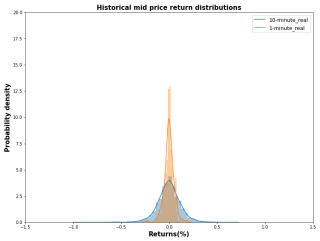
2. Stylized fact properties of the generated time series converge to historical. Specifically, we check for convergence of generated asset return distributions and volume/volatility correlations to historical – see Figure 2.

3. The discriminator cannot distinguish generated time series from real$D\left(x_{real}\right) \approx \frac{1}{2}$ and $D\left(x_G\right) \approx \frac{1}{2}$ – see Figures 3a, 3b, 3c for distribution of discriminator scores during training.

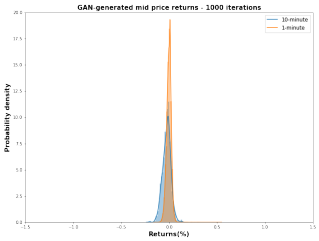
3.2 Ablation study and evaluation

To justify the complexity of the GAN architecture with self-attention in Figure 10, we compare it to the GAN without self-attention that takes same latent dimension as an input, but whose both generator and discriminator are given by feed-forward neural networks with dropout layers and ReLU activation in intermediate layers. This discriminator architecture was studied in Wang et al. (2016) and is depicted in Figure 11 (in Appendix). We train both GANs on mid price return time series and observe that GANs with self-attention produce discriminators with better recognition capabilities, as evident from Figures 4a and 4b. One can see that adding self-attention to GANs reduces the variance of discriminator score distribution and makes it more concentrated around $\frac{1}{2}$ both for real and generator-produced time series. We further observe that by using both mid price return and volume time series for training a GAN with self-attention, we achieve sufficient discrimination accuracy quicker than by using only mid price returns.

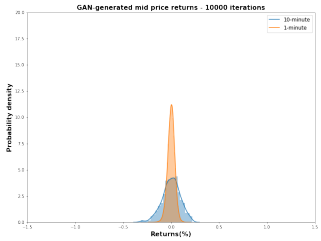
To measure the performance of our model more quantitatively and to study the impact of the latent dimension size, we also conducted analysis using two-sample Kolmogorov-Smirnov test. The historical and generated discriminator scores distributions are compared. The null hypothesis (H0) is that "the two samples are drawn from the same underlying distribution". From Figure 4c, p-values indicates that after 10k iterations, for latent dimensions equal to 100 and 200 we can't reject the null hypothesis. This means that our generator is producing a distribution of time series that follows the same underlying distribution that the historical. For latent dimension equal to 400, the system requires at least 20k iterations to generalize well and to produce realistic distribution.



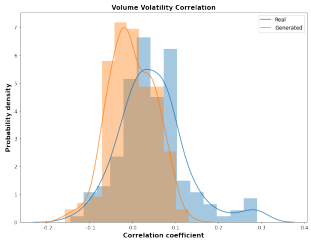
(a) Historical.



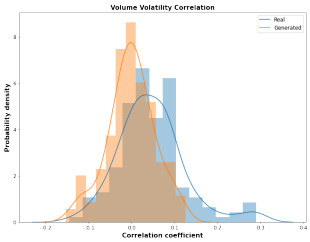
(b) GAN-generated after 1000 iterations.



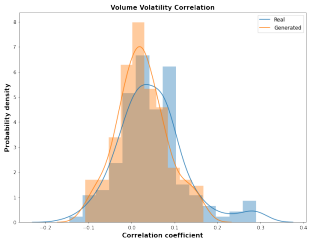
(c) GAN-generated after 10000 iterations. Note that distributions are similar to (a).



(d) Initial.

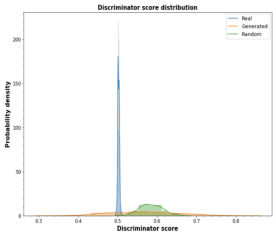


(e) After 1000 iterations.

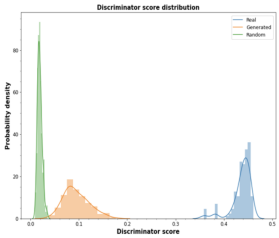


(f) After 10000 iterations. Note that 'real' and 'generated' distributions are overlapping.

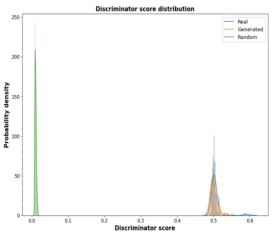
Figure 2: (a) Historical distributions of mid price returns. (b,c) Training progression of 1- and 10-minute mid price return distributions.(d,e,f) Training progression of volume/volatility correlation distributions



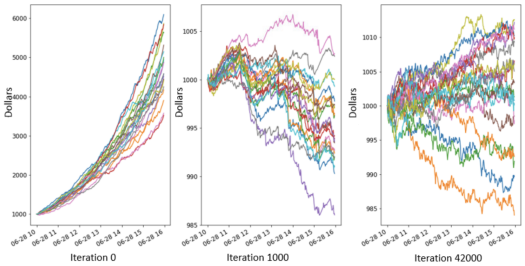
(a) Initial.



(b) After 1000 iterations.



(c) After 10000 iterations. Note that 'real' and 'generated' distributions are overlapping.



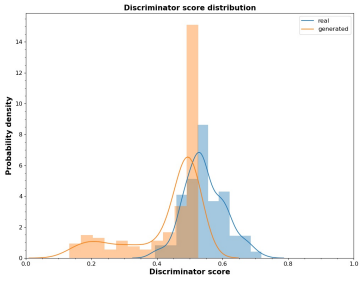
(d) We visually observe diversity of generated price time series with training progression.

Figure 3: (a,b,c) Training progression of distributions of discriminator scores. At convergence, we observe an overlap between 'real' and 'generated' data discriminator scores distributions, centered around 0.5 with low discrimination variance. We also passed random noise to the discriminator to validate it - 'random' label.(d) Training progression of generated mid price returns.

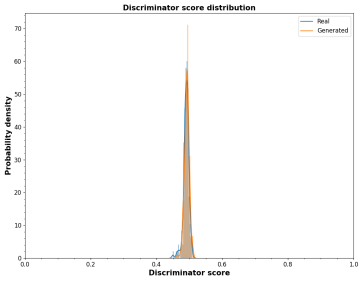
3.3 Simulator parameter optimization

3.3.1 Simulation environment

In order to evaluate the ability of a given agent configuration to reproduce stylized facts about the market, we employ an agent-based interactive discrete event limit order book simulation environment [citation redacted]. The environment provides a NASDAQ-like central exchange agent which manages the flow of time, maintains the limit order book where the orders are inserted and the set of order matching rules, and handles all inter-agent communication. The environment also accepts a single pseudo-random number generator seed at initialization for modeling choices that agents probabilistically make. In our experiments, the simulated agent universe consists of three distinct



(a) Ablation study: GAN without self-attention. Trained with latent dimension 10 using mid-price returns time series only.



(b) Ablation study: GAN with self-attention. Trained with latent dimension 10 using mid-price returns time series only.

Epoch Number	Latent Dim	K-S Stat	P-Value
20k	100	0.10	0.37
	200	0.18	0.05
	400	0.15	0.06
10k	100	0.10	0.40
	200	0.23	0.06
	400	0.10	0.03
1k	100	0.72	10^{-12}
	200	0.44	10^{-33}
	400	0.65	10^{-20}

(c) Two sample Kolmogorov-Smirnov test. At 10k iterations for a latent dimension of 100, we cannot distinguish between historical discriminator scores distribution and generated one based on the two sample test.

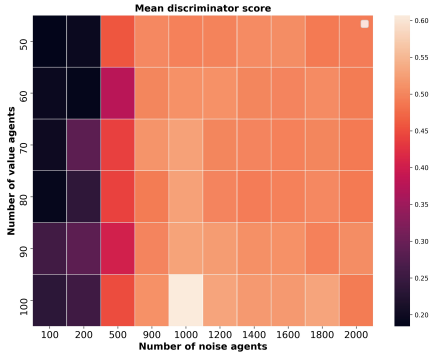
Figure 4

agent behavior types (described below) – our calibration goal is to find the most realistic agent configuration that can be described by them:

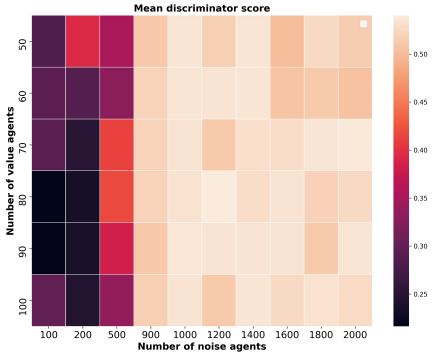
Market maker agent: The market maker agent acts as a liquidity provider by determining the market mid price and placing orders on both sides deep into LOB with a constant arrival rate (we implemented the model of Chakraborty & Kearns (2011)). Its goal is to constantly supplies liquidity to satisfy demand of other market participants.

Value Agents: The value agents are designed to simulate the actions of fundamental traders that trade according to their belief of the exogenous value of a stock, but without any view of the LOB microstructure. The external value of stock price is modeled by a fundamental price stream. In our experiment, fundamental price stream is given by the Ornstein-Uhlenbeck process with megashock events as described in Byrd (2019). Each value agent arrives to the market multiple times according to a Poisson process and chooses to buy or sell a stock depending on whether it is cheap or expensive relative to its noisy observation of a fundamental price. Once the side of an order is determined, the value agent places a limit order at a random level either inside the spread or deeper into the LOB. Therefore, value agents also supply liquidity to the market.

Noise Agents: Noise agents acts as liquidity consumers and are designed to simulate the action of retail traders to trade on demand (e.g. Kyle (1985)). Each noise agent trades once a day by placing a market order. The direction and the size of the trade are chosen randomly.



(a) The discriminator is trained with time series simulated by using $N^* = 1000$ noise agents and $M^* = 100$ value agents for different random seeds.



(b) The discriminator is trained with historical dataset.

Figure 5: Average discriminator score heatmap visualization of parameter optimization procedure with respect to the number of noise and value agents. For each noise and value agent configuration on the rectangular grid, we run 20 simulations with different seeds for initialization of pseudo-random number generator. Lighter colors correspond to higher discrimination score – hence, more realistic simulator configuration.

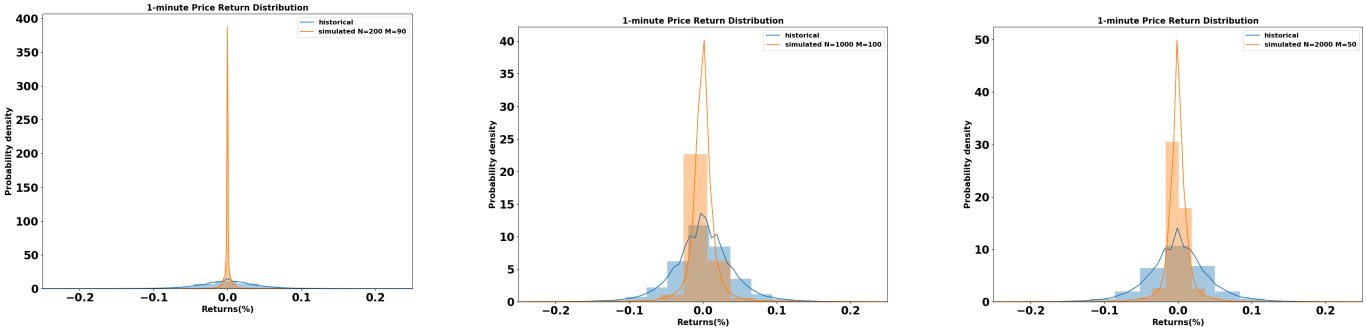
3.3.2 CONFIGURATION 'RECOVERY' EXPERIMENT

All of the above described three agent behavior types are present in real markets, however, the number of agents of each type are difficult to decide on for simulation configuration. It is especially difficult, since a small change to the number of agents can lead to a large change in emergent properties of the simulation. To demonstrate how MAS-GAN works in practice, we consider a set of configurations that consist of a single market maker, N noise agents and M value agents that follow an Ornstein-Uhlenbeck fundamental — with N and M being calibration parameters.

In our first experiment, for fixed N^* and M^* and different seeds for initialization of pseudo-random number generator, one can produce multiple time series samples — that is, simulate trading days with fixed agent participation. Run the following experiment to verify the validity of calibration procedure:

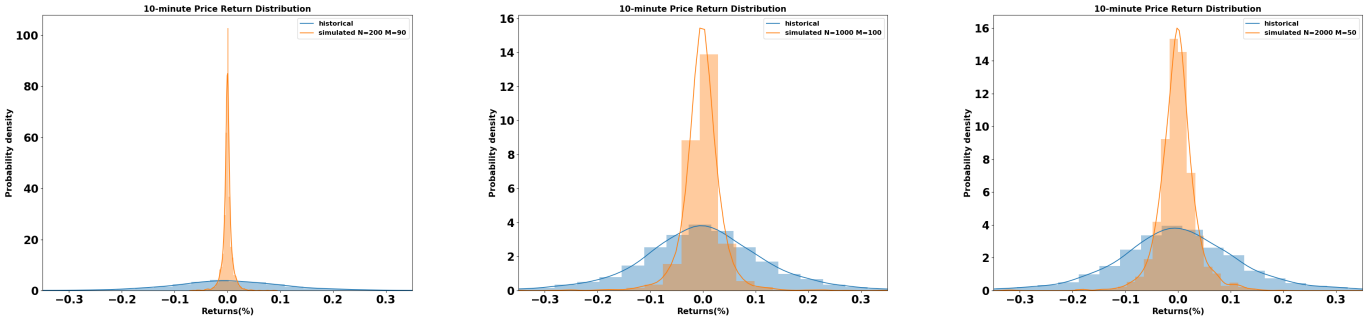
1. Train a discriminator as a part of GAN with self-attention to recognize time series samples for fixed N^* and M^* and varying random seeds.
2. For each choice of N and M over the rectangular grid, apply the trained discriminator to "recognize" N^* and M^* — a configuration that a discriminator was trained against.

Figure 5a illustrates the outcome of the above experiment for N^*=1000 and M^*=100. We observe that applying discriminator to the rectangular grid can identify the configuration that the discriminator was trained against. We also notice that configurations in the immediate neighborhood of N^*=1000 and M^*=100 on the grid are identified as more likely to have trained the discriminator than those that are further away.



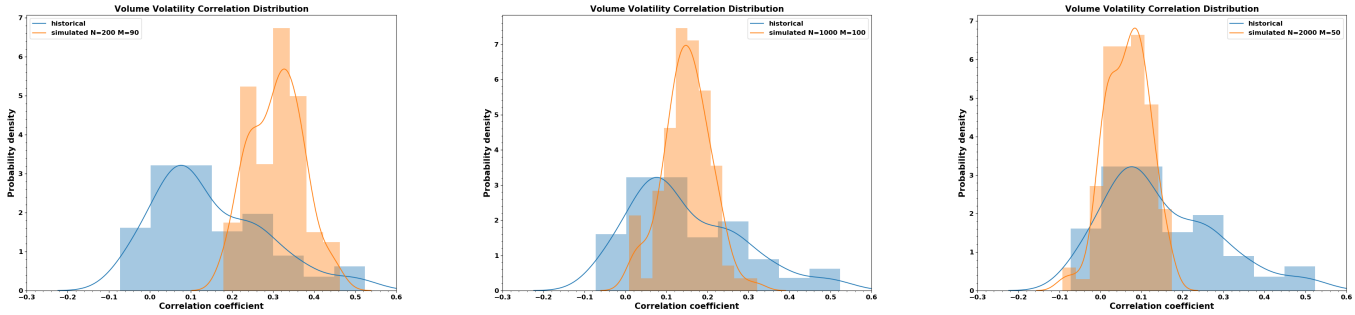
(a) $N = 200$ and $M = 90$ (b) $N = 1000$ and $M = 100$ (c) $N = 2000$ and $M = 50$

Figure 6: Comparison of distributions of 1-minute mid price returns for historical and simulated time series with N noise agents and M value agents.



(a) $N = 200$ and $M = 90$ (b) $N = 1000$ and $M = 100$ (c) $N = 2000$ and $M = 50$

Figure 7: Comparison of distributions of 10-minute mid price returns for historical and simulated time series with N noise agents and M value agents.



(a) $N = 200$ and $M = 90$ (b) $N = 1000$ and $M = 100$ (c) $N = 2000$ and $M = 50$

Figure 8: Comparison of distributions of volume/volatility correlations for historical and simulated time series with N noise agents and M value agents.

3.3.3 Experiment with historical discriminator

In our next experiment, we apply the discriminator trained on historical dataset described in Section 3.1 to the same rectangular grid. Given a set of configurations, we want to perform a calibration task - that is, determine which configurations on the grid can produce most the realistic simulation of the historical dataset. Figure 5b shows the result of applying discriminator trained on historical data to the same rectangular grid of noise and value agent configurations. Although we do not have a "ground truth" in this case, we can examine the stylized facts that these configurations produce in comparison with historical – see Figures 6, 7 and 8 for such comparisons of 1-minute and 10-minute mid price returns and volume/volatility correlation distributions for the three points on the heatmap in Figure 5b. One can see from the charts that the configuration $N = 200$ and $M = 90$, that is identified as least realistic out of the three by the discriminator, has stylized facts that are most dissimilar to historical; whereas, configurations with $N = 1000, M = 100$ and $N = 2000, M = 50$ result in stylized facts that are closer to historical. As in the previous experiment, note that adjacent points on the grid have similar distribution scores.

4 Conclusion

In this paper we present MAS-GAN – a novel adversarial method for multi-agent simulator calibration – we first train a discriminator as a part of GAN, and then use it to optimize the simulation parameters. We note that fidelity of our method depends on the quality of trained GAN (both generator and discriminator). In this work, we represented both generator and discriminator by neural networks with self-attention and showed experimentally that the method can achieve good calibration performance. This work can be extended however. For example, following the idea of Linformer in Sinong et al. (2020), using the fact that self attention can be approximated with a low-rank matrix, it can be computed linearly instead of using $O(n * *2)$ in time and space with respect to the input. Future work could also study to which extent such GAN for time series would memorize the training set or at the contrary produce diverse outputs.

In this paper, we test the MAS-GAN method on limit order book mid price and traded volume time series sampled at one minute. Using that data, we calibrate the simulator over a rectangular grid of noise and value agents (the market maker agent is present in all configurations). In the future, we are planning to look at different time resolutions, as well as to extend the agent universe by different well-known types of agents such momentum traders and heuristic belief learners Gjerstad (2007).

We also want to point out that even though the primary goal of this paper to perform calibration of multi-agent market simulators, applying MAS-GAN to historical data for a given asset over a given time period can help produce behavioral explanations of the market - for instance, how market agent composition changes for high vs. low liquidity assets or high vs. low volatility regimes. This can help understand and model the market better.

REFERENCES

Saúl Alonso-Monsalve and Leigh H. Whitehead. Image-based model parameter optimisation using model-assisted generative adversarial networks. *CoRR*, 2018. URL http://arxiv.org/abs/1812.00879.

Jean-Philippe Bouchaud, Julius Bonart, Jonathan Donier, and Martin Gould. *Trades, quotes and prices: financial markets under the microscope*. Cambridge University Press, Cambridge, 2018.

David Byrd. Explaining agent-based financial market simulation. *arXiv preprint arXiv:1909.11650*, 2019.

Tanmoy Chakraborty and Michael Kearns. Market making and mean reversion. In *Proceedings of the 12th ACM conference on Electronic commerce*, pp. 307–314. ACM, 2011.

Rama Cont. Empirical properties of asset returns: stylized facts and statistical issues. 2001.

Cristobal Esteban, Stephanie Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. 06 2017.

Steven Gjerstad. The competitive market paradox. *Journal of Economic Dynamics and Control*, 2007.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

Jakob Grazzini, Matteo Richiardi, and Efthymios Tsionas. Bayesian estimation of agent-based models. *Journal of Economic Dynamics and Control*, 11 2015.

Albert S Kyle. Continuous auctions and insider trading. *Econometrica: Journal of the Econometric Society*, pp. 1315–1335, 1985.

Francesco Lamperti, Andrea Roventini, and Amir Sani. Agent-based model calibration using machine learning surrogates, 2017.

Dan Li, Dacheng Chen, Jonathan Goh, and See-Kiong Ng. Anomaly detection with generative adversarial networks for multivariate time series. *CoRR*, abs/1809.04758, 2018. URL http://arxiv.org/abs/1809.04758.

Junyi Li, Xintong Wang, Yaoyang Lin, Arunesh Sinha, and Michael P. Wellman. Generating realistic stock market order streams. In *To Appear in the 34th AAAI Conference on Artificial Intelligence*, 2020.

Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1VGkIxRZ.

Gilles Louppe, Joeri Hermans, and Kyle Cranmer. Adversarial variational optimization of non-differentiable simulators, 2017.

Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A survey on gans for anomaly detection, 2019.

Imon Palit, Steve Phelps, and Wing Lon Ng. Can a zero-intelligence plus model explain the stylized facts of financial time series data? pp. 653–660, 06 2012.

N. Ruiz, S. Schulter, and M. Chandraker. Learning to simulate. In *Proceeding of the International Conference on Learning Representations*, New Orleans, LA, 2019.

Webster Ryan, Julien Rabin, Loïc Simon, and Frédéric Jurie. Detecting overfitting of deep generative networks via latent recovery. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/papers/Webster_Detecting_Overfitting_of_Deep_Generative_Networks_via_Latent_Recovery_CVPR_2019_paper.pdf.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, 2017.

Wang Sinong, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. 2020. URL https://arxiv.org/abs/2006.04768.

Yong Sun, Zhentao Xu, and Tianyu Zhang. On-board predictive maintenance with machine learning. In *SAE Technical Paper*. SAE International, 04 2019.

Gabriele Tedeschi, Maria Recchioni, and Mauro Gallegati. A calibration procedure for analyzing stock price dynamics in an agent-based framework. *http://ssrn.com/abstract=2330739*, 09 2013.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017. URL https://arxiv.org/abs/1706.03762.

Svitlana Vyetrenko and Shaojie Xu. Risk-sensitive compact decision trees for autonomous execution in presence of simulated market response. In *ICML 2019 Workshop on AI in Finance*, 06 2019.

Svitlana Vyetrenko, 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, 2019.

Dongyi Wang, Robert Vinson, Maxwell Holmes, Gary Seibel, Avital Bechar, and Shimon Nof. Early detection of tomato spotted wilt virus by hyperspectral imaging and outlier removal auxiliary classifier generative adversarial nets (or-ac-gan). *Scientific Reports*, 03 2019. doi: 10.1038/s41598-019-40066-y.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. *CoRR*, 2016. URL http://arxiv.org/abs/1611.06455.

Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant gans: deep generation of financial time series. *Quantitative Finance*, pp. 1–22, Apr 2020. ISSN 1469-7696. doi: 10.1080/14697688.2020.1730426. URL http://dx.doi.org/10.1080/14697688.2020.1730426.

Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case. 2020. URL https://arxiv.org/pdf/2001.08317.pdf.

Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. URL https://arxiv.org/pdf/1711.10485.pdf.

S. Yang, M. Paddrik, R. Hayes, A. Todd, A. Kirilenko, P. Beling, and W. Scherer. Behavior based learning in identifying high frequency trading strategies. In *2012 IEEE Conference on Computational Intelligence for Financial Engineering Economics (CIFEr)*, 03 2012.

H. Zhang, Goodfellow I., D. Metaxas, and A. Odena. Self-attention generative adversarial networks. *International Conference on Machine Learning*, 2019. URL https://arxiv.org/pdf/1805.08318.pdf.

A APPENDIX

A.1 1D SELF-ATTENTION LAYER

Here we present in more detail the attention mechanism that is accountable for encoding the global correlations of volumes and mid price returns feature maps.

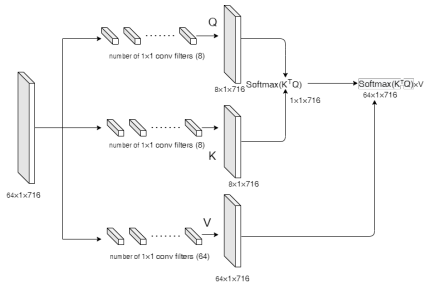


Figure 9: The 1D self-attention mechanism first starts with 1 by 1 convolutional layers along the channel dimension for memory efficiency. Then it follows the standard key, query, values scheme applied to the feature map. Kernel size for the feature spaces f, g is 8 and 8 for feature space h.

```
def call(self, x):
    f = K.conv1d(x,
                 kernel=self.kernel_f,
                 strides=1,
                 padding='same')
    g = K.conv1d(x,
                 kernel=self.kernel_g,
                 strides=1,
                 padding='same')
    h = K.conv1d(x,
                 kernel=self.kernel_h,
                 strides=1,
                 padding='same')
    s = K.batch_dot(g,K.reshape(f,shape=[K.shape(f)[0],\
    K.shape(f)[2], K.shape(f)[1]] ))
    beta = K.softmax(s, axis=1)
    o = K.batch_dot(beta, h)
    o = K.reshape(o, shape=K.shape(x))
    x = self.gamma * o + x
    return x
```

A.2 GAN ARCHITECTURE

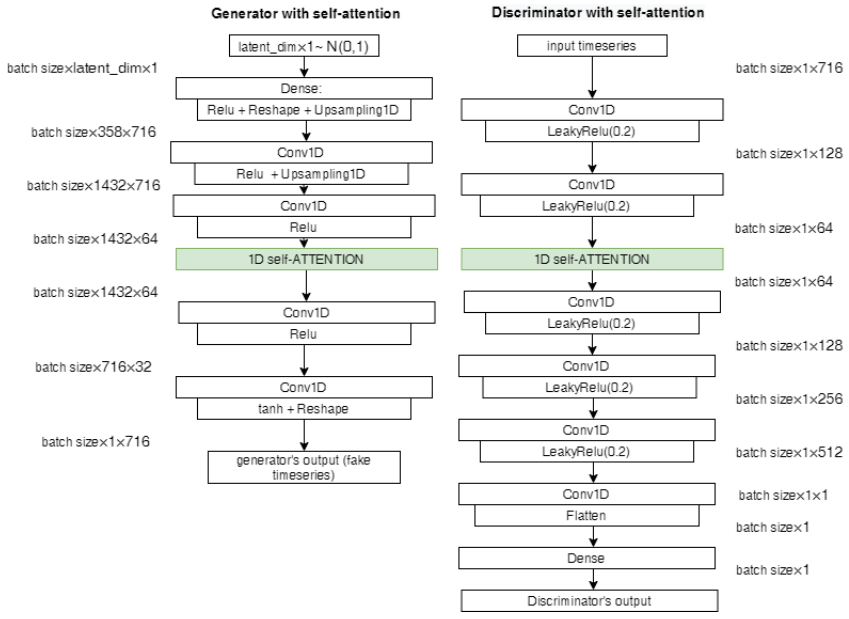


Figure 10: Generative adversarial network architecture with self-attention. The punctured lines represent dropout layers.

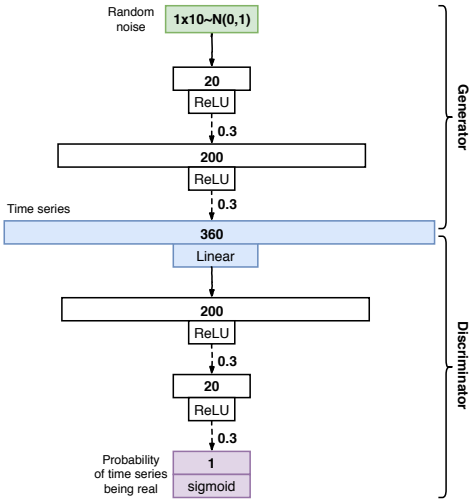


Figure 11: Generative adversarial network architecture without self-attention (used for the ablation study). The punctured lines represent dropout layers.