Multiscale Generative Models: Improving Performance of a Generative Model Using Feedback from Other Dependent Generative Models

Changyu Chen,[1] **Avinandan Bose,**[2] **Shih-Fen Cheng,**[1] **Arunesh Sinha**[1]

[1] Singapore Management University

[2] Indian Institute of Technology Kanpur

cychen.2020@phdcs.smu.edu.sg, avibose@iitk.ac.in, sfcheng@smu.edu.sg, aruneshs@smu.edu.sg

Abstract

Realistic fine-grained multi-agent simulation of real-world complex systems is crucial for many downstream tasks such as reinforcement learning. Recent work has used generative models (GANs in particular) for providing high-fidelity simulation of real-world systems. However, such generative models are often monolithic and miss out on modeling the interaction in multi-agent systems. In this work, we take a first step towards building multiple interacting generative models (GANs) that reflects the interaction in real world. We build and analyze a hierarchical set-up where a higher-level GAN is conditioned on the output of multiple lower-level GANs. We present a technique of using feedback from the higher-level GAN to improve performance of lower-level GANs. We mathematically characterize the conditions under which our technique is impactful, including understanding the transfer learning nature of our set-up. We present three distinct experiments on synthetic data, time series data, and image domain, revealing the wide applicability of our technique.

Introduction

Realistic simulations are an important component of training a reinforcement learning based decision-making system and transferring the policy to the real world. While typical video game-based simulations are, by design, realistic, the same is very difficult to claim when the problem domain is a real-world multi-agent system such as stock markets and transportation. Indeed, recent works have used generative models to design high-fidelity simulators using real world data (Li et al. 2020a; Sun et al. 2020; Shi et al. 2019; Chen et al. 2019), supposedly providing a higher degree of fidelity than traditional agent-based simulators. However, such generative models do not provide a fine-grained simulation of each agent in a multi-agent system, instead modeling the system as a single generative model. A fine-grained simulation can be informative in the design of interventions to optimize different aspects of the problem, e.g., how traders react to a new stock market policy. In this work, we take a first step towards building multiple interacting generative models (specifically, GANs), one for each type of agent, providing basic building blocks for a fine-grained high-fidelity simulator for complex real-world systems.

To the best of our knowledge, no prior work has built multi-agent simulations with multiple interacting generative models. Our work focuses on utilizing the interaction in a multi-agent system for better (generative) modeling of an individual agent for whom the data is limited or biased. Reflecting the multi-agent interaction in the real world, we set up a hierarchy of generative model instances, called Multiscale Generative Models (MGM), where the higher level is a conditional model that generates data conditioned on data generated by multiple distinct generative model instances at the lower level. We particularly note that our hierarchy is among different GAN instances (with distinct generative tasks), unlike hierarchy among multiple generators or discriminators for a single generative task that has been explored in the literature (Hoang et al. 2018).

Our *first contribution* is an architecture of how to utilize the feedback from a higher-level conditional GAN to improve the performance of a lower-level GAN. The higher-level GAN takes as input the output of (possibly multiple) lower-level GANs, mirroring agent interaction in the real world, e.g., prices offered in multiple regions (apart from other random factors) determining the total demand of electricity. We repurpose the discriminator (or critic in WGANs) of the higher-level model to provide an additional feedback (as an additional loss term) to the lower-level generator.

As our *second contribution*, we provide a mathematical interpretation of the interacting GAN set-up, thereby characterizing that the coupling between the higher-level and the lower-level models determines whether the additional feedback improves the performance of the lower-level GANs or not. We identify two scenarios where our set-up can be especially helpful. One is where there is a small amount of data available per agent at the lower-level and another where data is missing in some biased manner for the lower-level agents. Both these scenarios are often encountered in the real-world data, for example, in the use case of modeling electricity market, the amount of demand data for a new consumer might be small and the feedback from a higher-level model about such consumer's expected behaviors helps in building a more realistic behavioral model. Moreover, the case of small amount of data can also be viewed as a few-shot generation problem.

Finally, we provide three distinct sets of experiments revealing the fundamental and broadly applicable nature of our technique. The first experiment with synthetic data explores the various conditions under which our scheme provides benefits. Our second experiment with electric market time series data from Europe explores a set-up where electricity prices from three countries (plus random factors) determine demands, where we show improved generated price time series from one of the lower-level data generator using our approach. In our third experiment, we demonstrate the applicability of our work for a few-shot learning problem where the higher-level CycleGAN is a horse to zebra convertor trained using ImageNet, and a single lower-level GAN generates miniature horse images using feedback from the CycleGAN and small amount of data. Overall, we believe that this work provides a basic ingredient of and a pathway for building fine-grained high-fidelity multi-agent simulations. All missing proofs are in appendix.

Related Work

Generative Models: There is a huge body of work on generative modeling, with prominent ones such as VAEs (Kingma and Welling 2013) and GANs (Goodfellow et al. 2014). The closest work to ours is a set-up in federated learning where multiple parties own data from a small part of the full space and the aim is to learn one generator over the full space representing the overall distribution. Different approaches have been proposed to tackle this, where multiple discriminators at multiple parties train a single generator (Hardy, Le Merrer, and Sericola 2019) or multiple generators at different parties share generated data (Ferdowsi and Saad 2020) or network parameters (Rasouli, Sun, and Rajagopal 2020) to learn the full distribution. Our motivation, set-up, and approach is very different. Our motivation is fine-grained multi-agent simulation where we exploit the learned knowledge of the real world interaction (embedded in the higher level conditional GAN) to improve a lower level agent's generative model. Dependency among agents is crucial for our work, whereas these other approaches have independent parties.

Other non-federated learning works also employ multiple generators or discriminators to improve learning stability (Hoang et al. 2018; Durugkar, Gemp, and Mahadevan 2016) or prevent mode collapse (Ghosh et al. 2018). Pei et al. (2018) proposed multi-class discriminators structure, which is used to improve the classification performance cross multiple domains. Liu and Tuzel (2016) proposed to learn the joint distribution of multi-domain images by using multiple pairs of GANs. The network of GANs share a subset of parameters, in order to learn the joint distribution without paired data. Again, our main insight is to exploit dependencies among agents which is not considered in these work.

Few-shot Image Generation: Few-shot image generation aims to generate new and diverse examples whilst preventing overfitting to the few training data. Since GANs frequently suffer from memorization or instability in the low-data regime (e.g., less than 1,000 examples), most prior work follows a few-shot adaptation technique, which starts with a source model, pretrained on a huge dataset, and adapts it to a target domain. They either choose to update the parameters of the source model directly with different forms of regularization (Wang et al. 2018; Mo, Cho, and Shin 2020; Ojha et al. 2021; Li et al. 2020b), or embed a small set of additional parameters into the source model (Noguchi and Harada 2019; Wang et al. 2020; Robb et al. 2020). Besides these, some works also exploit data augmentation to alleviate overfitting (Karras et al. 2020a; Zhao et al. 2020) but this approach struggles to perform well under extreme low-shot setting (e.g., 10 images) (Ojha et al. 2021). Different from these work, our method exploits abundant data that is dependent on the target domain data and provides effective feedback to the GAN that performs adaptation (lower level GAN), leading to higher quality generation results. Note that our MGM does not have any specific requirement for the lower level GAN so it can benefit a bunch of existing work.

Preliminaries and Problem Formulation

Although our approach is independent of the type of generative model used, to explain our main solution approach concretely, we adopt a well-known generative model called WGAN, which is briefly introduced below. Every WGAN has a generator network G_θ and a critic network C_w, where θ, w are weights of the networks. The generator generates samples $y = G_\theta(z, c) \in Y$ from a learned conditional distribution $\mathbb{P}^g_{y|c}$, where z is some standard noise (e.g., Gaussian) and $c \in C$ is a condition. The goal is to learn the true distribution $\mathbb{P}^r_{y|c}$. In an unconditional WGAN, the critic simply outputs an estimate of the Wasserstein-1 distance, $W(\mathbb{P}^r, \mathbb{P}^g)$, between the real and generated distributions, given samples from each. For the conditional case, the condition c forms part of the input (y, c) for the critic. Let $\mathbb{P}^r_{joint}$ denote the distribution over $Y \times C$ with $c \sim \mathbb{P}^r_c$ and $y \sim \mathbb{P}^r_{y|c}$, and similarly $\mathbb{P}^g_{joint}$ denotes the distribution with $c \sim \mathbb{P}^r_c$ and $y \sim \mathbb{P}^g_{y|c}$. With the condition as input, the critic estimates the distance between joint distributions, $W(\mathbb{P}^r_{joint}, \mathbb{P}^g_{joint})$. The expected loss for the generator is $-E_{z,c\sim\mathbb{P}^r_c}[C_w(G_\theta(z, c)]$ and for the critic is[1]

$$E_{(y,c)\sim\mathbb{P}^g_{joint}}[C_w(y, c)] - E_{(y,c)\sim\mathbb{P}^r_{joint}}[C_w(y, c)] + \lambda \mathcal{J}_{C_w,\mathbb{P}^g_{joint}}$$

where $\mathcal{J}$ is the gradient penalty regularizer in WGAN-GP (Gulrajani et al. 2017) with weight λ. The actual loss optimized is an empirical version of the above expectations, estimated from data samples. A subtle implementation issue (unaddressed in prior work) arises with the $\mathcal{J}$ term when the condition is continuous; we discuss how we address this in the appendix.

Problem Formulation: Let there be n agents, where the agent in i^{th} position produces data distributed in the space $X_i \subset \mathbb{R}^{d_i}$ for some positive integer dimension d_i. Corresponding to position i, let $\mathbb{P}^r_i$ be a probability distribution and the agent generates $x_i \in X_i$ distributed as $x_i \sim \mathbb{P}^r_i$. Denote the data generated by all the agents as a single vector $x = \langle x_1, \ldots, x_n \rangle$. These lower-level agents generate data that feeds into a higher-level agent, which we call as *mixer*

[1]As a heuristic, generated sample is mixed with real sample in WGAN-GP; we ignore this in writing for ease of presentation.

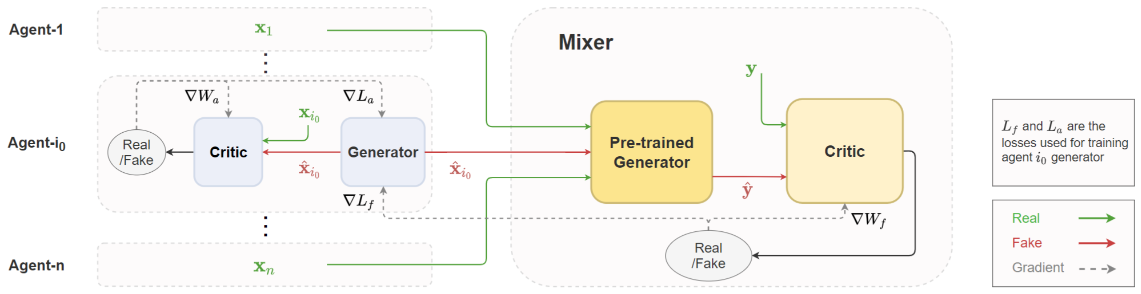

Figure 1: Architecture of the proposed MGM model.

to distinguish it from the lower-level agents. The mixer generates data in space $Y \subset \mathbb{R}^d$ for some positive integer dimension d with a conditional distribution $\mathbb{P}^r_{y|x}$. A pictorial representation of the set-up is shown in Figure 1.

Our goal is to obtain a fine-grained model of the agents as well as the mixer. With sufficient data, building multiple GAN models for the mixer and every agent is not very hard. However, the data for some lower-level agents are often limited or even corrupted; e.g., when an agent has *newly arrived* into the system. For example, this could happen with a new electricity supplier in the electricity market, or a new trader in a stock market, or even a new image generation task with low or biased amount of data. Thus, we assume one of the n agents, indexed by i_0 has newly arrived and replaces the prior agent at index i_0. This leads to the technical problem that we solve in this paper.

Problem Statement: Our goal is to utilize the higher-level mixer GAN to improve upon the generative model of a lower-level agent GAN, namely one that is newly arrived.

We believe that the above is tractable since the mixer model is trained on and relies on the output distribution of every kind of agent, and hence encodes information about the agent-level model within its own model. This information can be used to guide the training of an agent-level GAN.

Examples: We provide two possible real world practical applications of a solver of the aforementioned problem. This is apart from the practical few-shot learning experiment we present later. First, a biased data scenario. Consider three transportation network companies (TNCs), such as Uber, providing online ride-hailing services in a city with multiple districts. They would like to model the average behavior of drivers in different districts to help with optimized vehicle dispatching. This can be done by training a generative model for every district (each district as an agent) if a TNC has all the data. However, a TNC only has part of the drivers' data, that is, a biased dataset. E.g., 30% drivers work for TNC-1, while 40% and 30% work for TNC-2 and TNC-3 respectively. Due to the competitive relationship, TNCs do not share data with each other. They agree on providing their data to a third party, where all the data is used to pre-train a mixer that models a city-level output (e.g., total supply). When a TNC trains an agent, the agent can obtain extra feedback by feeding its own generator's output to the third part (mixer) and receiving the gradient feedback (as explained in the approach section). During the process, the third party does not share any data with a TNC.

Next, consider a low data scenario. Continuing from the above example, consider a new TNC (a fourth TNC) that starts to provide service in the city. As it is a new TNC, it has very limited data. In this case, the fourth TNC can also improve its own model by the feedback from the mixer.

Approach

Our approach, which we call MGM, to the problem stated above relies upon the architecture in Fig. 1. We choose GANs as our generative model (different types of GANs for different problems). As shown in the architecture, the mixer GAN is a conditional GAN in which the conditions are the output of the agent GANs. We use the notation: x_{-i_0} as a vector in $\times_{j \neq i_0} X_j$ and $\langle x_{-i_0}, x_{i_0} \rangle$ denotes the full vector with the i_0 component.

The i^{th} agent generator takes as input noise z_i to produce a sample $x_i = G_{\theta_i}(z_i)$ from its learned distribution. The mixer generator takes as input (1) noise z_{mix} and (2) the output $x = \langle x_1, \ldots, x_n \rangle$ of the agent generators as a condition to produce a sample $y = G_{\theta_{mix}}(z_{mix}, x)$ from its learned distribution. The conditional mixer GAN is trained in the traditional manner using available data for all the agents. We assume that there is plenty of data to train the mixer GAN effectively. That is, the learned mixer generator G_{mix} models the distribution $\mathbb{P}^r_{y|x}$ accurately.

We focus on the training of one *newly arrived* agent i_0, which replaces the prior agent i_0. The position i_0 has a true average distribution $\mathbb{P}^r_{i_0}$. This new arrival happens after the mixer GAN is trained. $D^r_y \subset Y$ is the real output data corresponding to the mixer. We also have datasets for every agent $j \neq i_0$, given by $D^r_j \subset X_j$ and $D^r_{i_0} \subset X_{i_0}$. In particular, $D^r_{i_0}$ has low amount of or biased data that prevents learning the true distribution $\mathbb{P}^r_{i_0}$ just using $D^r_{i_0}$.

Our technical innovation is in training generator of agent i_0. The framework for this is shown in Algorithm 1. The agent trainer has access to the datasets stated above, D^r_y, D^r_j for all agents j, as well as the pre-trained G_{mix} of the mixer. The agent i_0 generator is trained using an empirical version L_a (i.e., sample average estimate) of its own expected loss $\mathcal{L}_a$ and also using an empirical version L_f of the feedback $\mathcal{L}_f$ from the mixer critic (line 16-17). There are two ways to combine these loss terms: (1) form one combined loss as $\alpha L_a + (1-\alpha) L_f$ where $\alpha \in (0, 1)$ is a tunable hyperparameter and (2) train with L_a for some iterations and then

L_f for some iterations (alternate updating). In experiments, we use the former for synthetic and time series data, and the latter for image domain.

The empirical estimates for $\mathcal{L}_a, \mathcal{L}_f$ are obtained using samples in line 14-15. Additionally, $\mathcal{L}_a, \mathcal{L}_f$ depend on the agent level critic $C_{w_{i_0}}$ and mixer level critic $C_{w_{mix}}$ respectively (see example below). $C_{w_{i_0}}$ and $C_{w_{mix}}$ are trained with the empirical form W_a and W_f of the expected loss $\mathcal{W}_a$ and $\mathcal{W}_f$, respectively (line 10 and 12). These empirical losses W_a and W_f are built from the samples in line 5-9.

Concrete Example: We use the WGAN-GP loss terms here. With WGAN-GP, the loss $\mathcal{L}_a$ and $\mathcal{L}_f$ would be $-E_{x_{i_0} \sim \mathbb{P}^g_{x_{i_0}}}[C_{w_{i_0}}(x_{i_0})]$ and $-E_{y \sim \mathbb{P}^g_y}[C_{w_{mix}}(y)]$, respectively. $\mathcal{W}_a$ is the standard WGAN-GP loss

$$E_{x_{i_0} \sim \mathbb{P}^g_{x_{i_0}}}[C_{w_{i_0}}(x_{i_0})] - E_{x_{i_0} \sim \mathbb{P}^r_{x_{i_0}}}[C_{w_{i_0}}(x_{i_0})] + \lambda \mathcal{J}_{C_{w_{i_0}}, \mathbb{P}^g_{x_{i_0}}}$$

For $\mathcal{W}_f$, we first introduce some notation. Let $\mathbb{P}^r_y$ be the real marginal distribution over Y, marginalized over all agents distributions. Let $\mathbb{P}^g_y$ be the marginal generated distribution over Y formed by the following process: $x_j \sim \mathbb{P}^r_j$ for $j \neq i_0$, $x_{i_0} \sim \mathbb{P}^g_{i_0}$, $y \sim \mathbb{P}^r_{y|x}$. Then, $\mathcal{W}_f$ is

$$E_{y \sim \mathbb{P}^g_y}[C_{w_{mix}}(y)] - E_{y \sim \mathbb{P}^r_y}[C_{w_{mix}}(y)] + \lambda_{mix} \mathcal{J}_{C_{w_{mix}}, \mathbb{P}^g_y}$$

In the above expressions, all expectations are replaced by averages over samples to form the counterpart empirical losses. These samples are obtained as follows: D^r_y are samples from $\mathbb{P}^r_y$ (line 9) and samples from $\mathbb{P}^g_y$ are generated by (1) forming x, by obtaining x_j from D^r_j for $j \neq i_0$ (line 6), obtaining x_{i_0} by running generator for i_0 (line 7), and (2) finally obtaining y by running the pre-trained mixer generator (which is assumed to be accurate) conditioned on x (line 8).

We reiterate that Algorithm 1 is a framework, and in the above paragraph we provide a possible instantiation with WGAN loss. The framework can be instantiated with CycleGAN and StyleGAN as well as a time series generator COT-GAN. These are used in the experiments, and we provide the detailed losses in the appendix due to space constraint.

Mathematical Characterization

We derive mathematical result for WGAN. As the high level idea of all GANs are similar, we believe that the high level insight obtained at the end of sub-section hold for any type of GAN. From the theory of WGAN (Arjovsky, Chintala, and Bottou 2017), the loss term $\mathcal{L}_f$ estimates the Wasserstein-1 distance between $\mathbb{P}^r_y$ and $\mathbb{P}^g_y$. As these two distributions differ only in the way x_{i_0} is generated, it is intuitive that $\mathcal{L}_f$ provides meaningful feedback. However, there are many interesting conclusions and special cases that our mathematical analysis reveals, which we show next.[2]

Background on Wasserstein-1 metric: Let $\Pi_{\mathbb{P}_i, \mathbb{P}_j}$ be the set of distributions (also called couplings) whose marginal distributions are $\mathbb{P}_i$ and $\mathbb{P}_j$, i.e., $\int_{X_i \times A} d\pi(x_i, x_j) = \mathbb{P}_j(A)$

[2]We work in Euclidean spaces and standard Lebesgue measure. Probability distribution refers to the probability measure function.

Algorithm 1: *MGM Template*

```
1  Inputs: Pre-trained $G_{mix}$, datasets $D^r_y, D^r_j \ \forall j$,
2  $N(0, I)$ is standard normal dist., $U(\cdot)$ is uniform dist.
3  while $\theta_{i_0}$ has not converged do
4      for $m$ times do
5          Sample $\{z_k\}_{k=1}^K, \{z'_k\}_{k=1}^K \sim N(0, I)$
6          Sample $\{x^r_{j,k}\}_{k=1}^K \sim U(D^r_j)$
7          $x^g_{i_0,k} = G_{\theta_{i_0}}(z_k)$ , $x^g_k = \langle x^r_{-i_o,k}, x^g_{i_0,k} \rangle$
8          $y^g_k = G_{mix}(z'_k, x^g_k)$
9          Sample $\{y^r_k\}_{k=1}^K \sim U(D^r_y)$
10         Form loss $W_a$ using $\{x^g_k, x^r_k\}_{k=1}^K$
11         Update $w_{i_0}$ using $\nabla_{w_{i_0}} W_a$
12         Form loss $W_f$ using $\{y^g_k, y^r_k\}_{k=1}^K$
13         Update $w_{mix}$ using $\nabla_{w_{mix}} W_f$
14     Sample $\{z_k\}_{k=1}^K, \{z'_k\}_{k=1}^K \sim N(0, I)$
15     Sample $\{x^r_{j,k}\}_{k=1}^K \sim U(D^r_j)$
16     Form loss $L_a$ using $G_{\theta_{i_0}}(z_k), C_{w_{i_0}}$
17     Form loss $L_f$ using
         $G_{mix}(z'_k, \langle x^r_{-i_o,k}, G_{\theta_{i_0}}(z_k) \rangle), C_{w_{mix}}$
18     Update $\theta_{i_0}$ using $\nabla_{\theta_{i_0}} L_a$ and $\nabla_{\theta_{i_0}} L_f$
```

and $\int_{A \times X_j} d\pi(x_i, x_j) = \mathbb{P}_i(A)$ for any event A in the appropriate σ-algebra, for any $\pi \in \Pi_{\mathbb{P}_i, \mathbb{P}_j}$. The Wasserstein-1 metric is:

$$W(\mathbb{P}_i, \mathbb{P}_j) = \inf_{\pi \in \Pi_{P_i, P_j}} \int_{X_i \times X_j} d(x_i, x_j)\, d\pi(x_i, x_j) \quad (1)$$

It is known that the Wasserstein-1 metric is also equal to

$$\sup_{\substack{f(x_i) + g(x_j) \leq d(x_i, x_j) \\ f \in \mathcal{C}_b(X_i), g \in \mathcal{C}_b(X_j)}} \int_{X_i} f(x_i)\, d\mathbb{P}_i(x_i) + \int_{X_j} g(x_j)\, d\mathbb{P}_j(x_j)$$

where $\mathcal{C}_b(X)$ is the space of bounded and continuous real valued functions on X. The functions f and g are also the dual functions (Villani 2009) as they correspond to dual variables of the infinite linear program that Eq. 1 represents. It is known that the minimizer of Eq. 1 and maximizer functions of the dual form exists (Villani 2009).

Analysis of $\mathcal{L}_f$: We are given the system generator that represents the distribution $\mathbb{P}^r_{y|x}$, with probability density function given by $P^r_{y|x}(\cdot)$. We also have true distributions $\mathbb{P}^r_j$ for all j. The data that we are interested in generating is from $\mathbb{P}^r_{i_0}$, and the corresponding agent level generator generates data from $\mathbb{P}^g_{i_0}$. We aim to have $\mathbb{P}^g_{i_0}$ close to $\mathbb{P}^r_{i_0}$. Towards this end, we repurpose the critic of the mixer GAN. In training, the critic is fed real data that is distributed according to the density $P^r_y(y) = \int_X P^r_{joint}(y, x) dx$ where $P^r_{joint}(y, x) = P^r_{y|x}(y|x) \times \prod_j P^r_j(x_j)$. The mixer critic is additionally fed generated data that is distributed according to the density $P^g_y(y) = \int_X P^g_{joint}(y, x) dx$ where $P^g_{joint}(y, x) = P^r_{y|x}(y|x) \times P^g_{i_0}(x_i) \times \prod_{j \neq i_0} P^r_j(x_j)$. Thus, the repurposed mixer critic estimates the Wasserstein-1 dis-

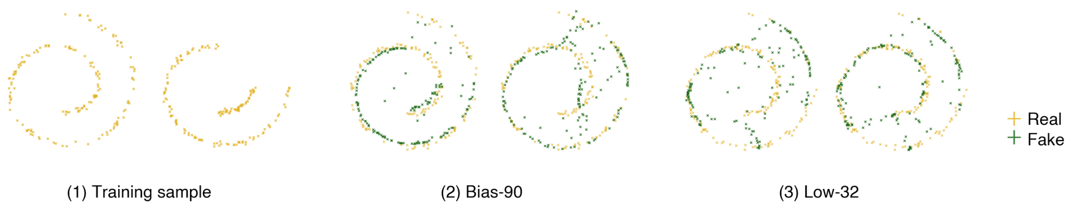

Figure 2: (1) Training sample: left fig. is full training data, right fig. shows missing data in the top right area (biased data); (2) given biased data, left fig. shows the baseline, right fig. shows WGAN-GP-C; (3) given low data (32 samples), left fig. shows the baseline, right fig. shows WGAN-GP-C. Note that all results shown here use $\beta = 0.7$.

tance between $\mathbb{P}^r_y(\cdot)$ and $\mathbb{P}^g_y(\cdot)$ given as

$$W(\mathbb{P}^r_y, \mathbb{P}^g_y) = \inf_{\pi \in \Pi_{\mathbb{P}^r_y, \mathbb{P}^g_y}} \int_{Y \times Y} d(y, y')\ d\pi(y, y')$$

In order to analyze the above feedback from the mixer to the agent generator, we define some terms. Consider

$$W(\mathbb{P}^r_{y|\hat{x}}, \mathbb{P}^r_{y|\hat{x}'}) = \inf_{\pi \in \Pi_{\mathbb{P}^r_{y|\hat{x}}, \mathbb{P}^r_{y|\hat{x}'}}} \int_{Y \times Y} d(y, y')\ d\pi(y, y')$$

Dual functions $f^*_{\hat{x}}$ and $g^*_{\hat{x}'}$ exist here such that $f^*_{\hat{x}}(y) + g^*_{\hat{x}'}(y') \leq d(y, y')$ for all y, y' as well as for any $\hat{x}, \hat{x}'$ and

$$W(\mathbb{P}^r_{y|\hat{x}}, \mathbb{P}^r_{y|\hat{x}'}) = \int_Y f^*_{\hat{x}}(y) P^r(y)\ dy + \int_Y g^*_{\hat{x}'}(y) P^g(y)\ dy$$

Let $\pi^*_d(x_j, x'_j)$ denote the distribution (or coupling) that achieves the minimum for the Wasserstein-1 distance below between the same distributions $\mathbb{P}^r_{i_0}$ and $\mathbb{P}^g_{i_0}$

$$\pi^*_d = \operatorname*{arginf}_{\pi \in \Pi_{\mathbb{P}^r_{i_0}, \mathbb{P}^g_{i_0}}} \int_{X_{i_0} \times X_{i_0}} d(x_{i_0}, x'_{i_0})\ d\pi(x_{i_0}, x'_{i_0})$$

Note that π^* takes d as a subscript, i.e., it depends on the distance metric d. We denote this explicitly since we later deal with different distance metrics. We will also use subscript d for $W(\cdot, \cdot)$, if not obvious. $\otimes_{j \neq i_0} \mathbb{P}^r_j$ denotes the joint probability distribution (product measure) over $\times_{j \neq i_0} X_j$. The next result provides a mathematical formulation of the Wasserstein-1 distance feedback provided from the mixer critic.

Theorem 1. *Assume that (1) all spaces X_j's, Y are compact, (2) all probability density functions are continuous, bounded and > 0 over its domain, and (3) $f^*_x(y), g^*_x(y)$ are jointly continuous in x, y. Then, the following holds:*

$$W(\mathbb{P}^r_y, \mathbb{P}^g_y) = \int_{X_{i_0} \times X_{i_0}} E_{x_{-i_0} \sim \otimes_{j \neq i_0} \mathbb{P}^r_j} \Big[W(\mathbb{P}^r_{y|\langle x_{-i_0}, x_{i_0} \rangle}, \mathbb{P}^r_{y|\langle x_{-i_0}, x'_{i_0} \rangle}) \Big]\ d\pi^*_d(x_{i_0}, x'_{i_0}),$$

where d is any metric on X_{i_0}.

We provide further explanation of the above mathematical result. The following result aids in the explanation:

Lemma 1. *Define $M_{x_{-i_0}}(\cdot)$ as the function $M_{x_{-i_0}}(x_{i_0}) = \mathbb{P}^r_{y|\langle x_{-i_0}, x_{i_0} \rangle}$ and assume all conditions in Theorem 1 hold. If $M_{x_{-i_0}}(\cdot)$ is an injection for all x_{-i_0}, then*

$$d_0(x_{i_0}, x'_{i_0}) = E_{x_{-i_0} \sim \otimes_{j \neq i_0} \mathbb{P}^r_j} \big[W(\mathbb{P}^r_{y|\langle x_{-i_0}, x_{i_0} \rangle}, \mathbb{P}^r_{y|\langle x_{-i_0}, x'_{i_0} \rangle}) \big]$$

is a distance metric on X_{i_0}. If in the above case, $M_{x_{-i_0}}(\cdot)$ is not an injection then d_0 is a pseudo-metrics.

Using the above result, if $M_{x_{-i_0}}(\cdot)$ is an injection $\forall x_{-i_0}$, then we can check that by choosing $d = d_0$ in Thm. 1 we get

$$W(\mathbb{P}^r_y, \mathbb{P}^g_y) = W_{d_0}(\mathbb{P}^r_{i_0}, \mathbb{P}^g_{i_0})$$

In words, under the injective assumption, the feedback $\mathcal{L}_f$ from the mixer critic is the Wasserstein-1 distance between real and fake distributions of position i_0 but measured in a different underlying distance metric d_0 on X_{i_0} induced by the mapping $M_{x_{-i_0}}(\cdot)$. When $M_{x_{-i_0}}(\cdot)$ is not an injection, then it induces a pseudo-metric on X_{i_0}. A pseudo-metric satisfies all properties of a distance metric except that two distinct points may have distance 0. This provides less fine-grained feedback with a push to match probability density only over those points with $d_0 > 0$, and in the extreme case when $M_{x_{-i_0}}(\cdot)$ is a constant function then $W(\mathbb{P}^r_{y|\langle x_{-i_0}, x_{i_0} \rangle}, \mathbb{P}^r_{y|\langle x_{-i_0}, x'_{i_0} \rangle})$ is zero for any x_{i_0}, x'_{i_0} and thus the feedback $\mathcal{L}_f$ from the mixer critic is zero. Indeed, $M_{x_{-i_0}}(\cdot)$ constant is the case when the mixer output distribution $\mathbb{P}^r_y$ does not depend on x_{i_0} and hence the mixer model does not encode any useful information about agent i_0.

At a high level, how the mixer depends on the agent i_0 (captured in $M_{x_{-i_0}}(\cdot)$) determines the nature and magnitude of the feedback. This feedback combined with the agent i_0's own loss still aims to bring $\mathbb{P}^g_{i_0}$ closer to $\mathbb{P}^r_{i_0}$.

Experiments

We showcase the effectiveness of MGM on three different domains[3]: (1) synthetic data modeling relationship between a mixer and two agents, (2) time series generation task, and (3) few-shot image generation that reveals the transfer learning nature of the MGM set-up. We use different specialized GANs for each domain.

For the synthetic and image data, we ran the experiments on a server (Intel(R) Xeon(R) Gold 5218R CPU, 2x Quadro

[3]https://github.com/cameron-chen/mgm : code and data are available at this url

RTX 6000, 128GB RAM) on the GPU. For time series data, we ran the experiments on a cloud instance (14 vCPUs, 56GB memory, 8x Tesla K80) on the GPU. For all experiments, we use Adam optimizer with a constant learning rate tuned between 1e-5 and 1e-3 by optuna (Akiba et al. 2019).

Evaluation: Our evaluation methodology has the following common approach: we first train a mixer GAN (or use pre-trained one) using full data available. We train the agent i_0 GAN using low or biased data only and additionally with feedback from the mixer. We then compare the performance of agent i_0 with and without the feedback from the mixer. The evaluation metrics vary by domain as described below.

Synthetic Data

We first show our results on synthetic data to explore various conditions under which the MGM provides benefits. The synthetic data scenario models a system where the output of the mixer is linearly dependent on the outputs of two agents. Agent 1 acts as the new arrival in this experiment.

Experimental Setup: We generate 128,000 samples for all agents, output of agent 1, $x_1 \in \mathbb{R}^2$, is drawn from Swiss roll (see Fig. 2 (1)), and output of agent 2, $x_2 \in \mathbb{R}^2$, is drawn from a 2D Gaussian distribution. The data for the mixer is a linear combination of agent outputs, $y = \beta x_1 + (1-\beta)x_2$, where the parameter $\beta \in [0, 1]$ controls the dependence of y on x_1. We use Wasserstein-1 distance, computed by POT package (Flamary et al. 2021), to evaluate the model.

We use unconditional WGAN-GP to model the agents and conditional WGAN-GP to model the mixer. Each WGAN-GP consists of a generator network with 3 fully-connected layers of 512 nodes with Leaky ReLU activations and a critic network of the same structure as the generator except for input and output layer. We choose the weight of gradient penalty term $\lambda_{i_0} = 0.1$ for agent GAN and $\lambda_{mix} = 1$ for mixer GAN based on a grid search. This experiment used the combined loss $\alpha L_a + (1-\alpha)L_f$, thus, the MGM trained model is denoted as WGAN-GP-C in Table 1.

Biased data: We remove the data of agent 1 from the top right area (see Fig. 2(1)) either completely (bias-100) or 90% (bias-90). We train with a minibatch size of 256. Then, the model is evaluated against a test set of size 2,000 that contains data in all area.

Low data: We sample random subsets of the training data for agent 1, of size 32 and 64. We train with a minibatch size of 32 or 64, depending on the size of the training data. Then, same as the biased data scenario, the model is evaluated with a test set of size 2,000.

Results Discussion: We visualize the trained distribution for bias-90 and low-32 ($\beta = 0.7$) in Fig. 2 (2) and 2 (3) respectively. We report quantitative results in Table 1. The results show that tighter coupling (higher β) results in better performance gain of agent 1's generator. This is consistent with the inference derived from our theoretical results (text after Lemma 1). We observe that agent 1's generator improves more in the case where the learned distribution by a single GAN is further away from the true distribution. On average, low-32 improves 8.0% compared with 4.4% for low-64, while bias-100 improves 15.7% compared with 9.0% for bias-90, where a single WGAN-GP (baseline) performs worse for low-32 and bias-100. More results obtained on multiple different random low datasets can be found in appendix.

Real-World Time Series Data

Prior work has investigated synthesizing time series data of a single agent via GANs (Xu et al. 2020; Yoon, Jarrett, and van der Schaar 2019; Esteban, Hyland, and Rätsch 2017). We model the time series data of an agent who is interacting in an electricity market. The European electricity system enables electricity exchange among countries. The demand for the electricity in one country is naturally affected by the electricity prices in its neighboring countries within Europe because of the cross-border electricity trade. For example, the demand for electricity in Spain is affected by the electricity prices in France, Portugal, and Spain itself.

We model the price in France as agent 1, the prices in Spain and Portugal as agents 2 and 3 respectively, and the demand in Spain as the mixer. The intuition is that the dependency between the demand for electricity in Spain and the price in France is able to provide extra feedback for agent 1 and hence help with better modeling of agent 1.

Experimental Setup: We use the electric price and demand data publicly shared by Red Eléctrica de España, which operates the national electricity grid in Spain (Eléctrica 2021). The data contains the day-ahead prices in Spain and neighboring countries, and the total electricity demand in Spain. We use the data between 5 Nov, 2018 and 13 July, 2019 aggregated over every hour and split by days. We use the difference between autocorrelation metric (ACE) (Xu et al. 2020) to evaluate the output.

For each of 249 days, agent 1 outputs the price every hour in France, $x_1 \in \mathbb{R}^{24}$, while other agents output the price every hour in Spain and Portugal, $x_2, x_3 \in \mathbb{R}^{24}$. The mixer takes as input the prices in the three countries and outputs the total demand in Spain, $y \in \mathbb{R}^{24}$. We use COT-GAN (Xu et al. 2020) to model the agents and its conditional version to model the mixer, using a minibatch size of 32, over 20,000 iterations. For other settings, we follow COT-GAN defaults.

As stated in preliminaries, when *pre-training* the mixer, the critic of the conditional COT-GAN should take as input the condition as well. However, we find that such pre-trained mixer performs poorly. Instead, we try an *alternate pre-training* where we do not feed the condition into the critic, thus, the critic measures the distance between the marginal distribution of the mixer output and real data (discussed more in appendix); we denote the corresponding MGM model by COT-GAN-CM (C for combined loss, M for marginal distribution). We denote the former MGM model by COT-GAN-CJ (J for joint distribution). Empirical results show that COT-GAN-CM works better than COT-GAN-CJ.

Biased data: Modeling missing data in a biased manner is tricky in time series data. We take the simple approach of dropping the second half of the time series sequences, after which we have 124 time series sequences (call this Half) from Nov 2018 to Feb 2019, that may exhibit seasonality.

Low data: We randomly sample 64 and 124 time series sequences (out of the 249) once for agent 1 and treat these

Scenario	Baseline (WGAN-GP)	WGAN-GP-C over $\beta \in [0, 1]$					
		0	0.1	0.3	0.5	0.7	1.0
Bias-100	0.705(0.009)	0.702(0.010)	0.655(0.016)	0.618(0.020)	0.585(0.010)	0.521(0.004)	0.483(0.014)
Bias-90	0.650(0.011)	0.642(0.009)	0.630(0.016)	0.629(0.022)	0.610(0.030)	0.566(0.021)	0.473(0.002)
Low-32	0.641(0.006)	0.639(0.011)	0.614(0.008)	0.593(0.006)	0.586(0.004)	0.565(0.005)	0.541(0.004)
Low-64	0.582(0.010)	0.570(0.009)	0.567(0.012)	0.559(0.010)	0.552(0.007)	0.553(0.007)	0.539(0.008)

Table 1: MGM vs baseline in Wasserstein-1 distance (mean and std. dev. of 16 runs; each run is a sample from the final trained generator) for different scenarios and varying β.

($\times 10^{-2}$)	Full	Half	Low-64	Low-124
COT-GAN	9.7(1.0)	13.9(1.6)	18.9(0.8)	10.2(1.1)
COT-GAN-CJ	9.1(0.7)	12.6(1.0)	17.3(1.6)	**8.5(0.9)**
COT-GAN-CM	**8.8(0.8)**	**12.2(0.9)**	**12.8(1.4)**	9.4(0.9)

Table 2: Model comparison using ACE ($\downarrow$) (avg and std. dev. of 16 sample means, each of 16 runs; each run is a sample from the final trained generator) for different scenarios. Bold shows best results.

	5-Shot	10-Shot	30-Shot
FSGAN	60.01(0.47)	49.85(0.71)	**47.64(0.69)**
FSGAN-A	**56.28(1.14)**	**48.98(0.52)**	48.41(1.10)
FreezeD	78.07(0.95)	53.69(0.38)	47.94(0.87)
FreezeD-A	**77.61(1.78)**	**51.92(0.73)**	**46.27(0.56)**

Table 3: FID scores ($\downarrow$) where MGM uses alternate updating. Std. dev. are computed across 5 runs.

two subsets (low-64 and low-124) as examples of low data.

Results Discussion: The results are summarized in Table 2. We observe the two MGM models consistently outperform the baseline and COT-GAN-CM outperforms COT-GAN-CJ most of the times. The results provide evidence that our MGM approach works for time series data. Additional supporting results are in the appendix.

Few-Shot Image Generation

The few-shot image generation problem is to generate images with very limited data, e.g., 10 images. Many works have proposed to solve this problem by adapting a pre-trained GAN-based model to a new target domain (Mo, Cho, and Shin 2020; Ojha et al. 2021; Li et al. 2020b; Wang et al. 2020). As low data is one of the scenarios in which our approach provides benefits, we explore few-shot image generation problem in a MGM set-up.

We consider a single agent and a mixer. The agent generates images of the *miniature horses*, while the mixer is a converter from horse to zebra images. The intuition here is that the mixer can provide feedback about the high-level concept of a horse structure, such that it mitigates the overfitting of the agent due to limited training data.

Experiment Setup: We collected images of miniature horses from the search engine *yandex.com* by keyword "miniature horse". All the images were resized to 256×256. After processing, the dataset contains 1061 images, 34 of them forming training set, the remainder forming test set. The dataset will be released publicly. We employ the widely used FID (Heusel et al. 2017) score to evaluate the output.

The agent model produces images of miniature horse, x. The mixer takes input x to produce zebra images y, where $x, y \in \mathbb{R}^{256\times256\times3}$. We select two models, FSGAN (Robb et al. 2020) and FreezeD (Mo, Cho, and Shin 2020) as the agent model; both of these fine tune a pre-trained StyleGAN2 (Karras et al. 2020b) to the miniature horse domain. We select the prior pre-trained CycleGAN (Zhu et al. 2017) (horse to zebra) as the mixer model. Recall that we train agent with L_f followed by L_a (alternate updating) in this experiment. We thus denote the MGM version of FSGAN and FreezeD by FSGAN-A and FreezeD-A respectively.

We follow the defaults from FSGAN, using minibatch size of 16, stopping at iteration 1250 and choosing the better of the model at iteration 1000 or 1250. We evaluate the model in the same manner as Ojha et al. (2021).

Results Discussion: The quantitative results in Table 3 with 5, 10, 30 shot for FSGAN and FreezeD show that the MGM approach improves the baseline FSGAN and FreezeD, especially for lower shot settings. Also, we show some clear cases of improved quality of images in Fig. 3.

Ablation Study

To further investigate how the mixer improves the agent GAN model, we perform an ablation study for synthetic and image data. For synthetic data, instead of combined loss, we train the agent generator with L_a and L_f alternately. We denote this alternate by WGAN-GP-A and show results in Table 4. We observe that the alternate updating strategy has less improvement than the combined loss (in Table 1) in all cases, except low-64 with $\beta = 0.7$. This reveals combined loss strategy is more effective for synthetic data. For image data, we perform ablation by using combined loss. We denote this alternate by FSGAN-C and FreezeD-C, and present results in Table 5. The results show that the alternate updating strategy is more effective on image data. Also, we found the training of FreezeD-C to be unstable, which is reflected in the results.

Scenario	Baseline (WGAN-GP)	WGAN-GP-A over $\beta \in [0, 1]$					
		0	0.1	0.3	0.5	0.7	1.0
Bias-100	0.705(0.009)	0.735(0.008)	0.664(0.018)	0.622(0.018)	0.603(0.018)	0.591(0.018)	0.488(0.008)
Low-64	0.582(0.010)	0.571(0.007)	0.571(0.008)	0.575(0.005)	0.559(0.006)	0.551(0.004)	0.551(0.007)

Table 4: Ablation study, MGM vs baseline in Wasserstein-1 distance (over 16 runs) for 2 scenarios and varying β.

	5-Shot	10-Shot	30-Shot
FSGAN	60.01(0.47)	49.85(0.71)	**47.64(0.69)**
FSGAN-C	**55.79(0.82)**	**49.19(0.56)**	49.30(0.73)
FreezeD	78.07(0.95)	**53.69(0.38)**	**47.94(0.87)**
FreezeD-C	**67.50(1.68)**	66.03(1.01)	69.45(0.45)

Table 5: Ablation study, FID scores (↓) where MGM uses single combined loss. Std dev. are computed across 5 runs.

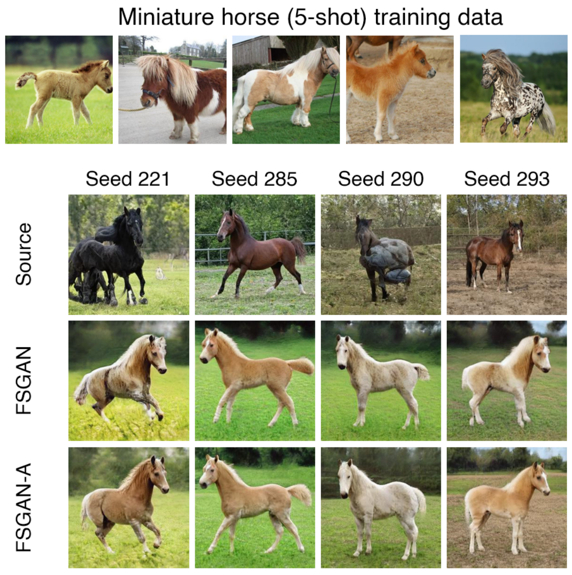

Figure 3: Source is the pretrained Style-GAN2, FSGAN is tuned Source, FSGAN-A is MGM tuned FSGAN. Our MGM approach (FSGAN-A) produces more natural images w.r.t. horse body, tail and clearer background.

Conclusion

In this paper, we demonstrate that we can significantly improve the performance of generative models by incorporating real-world interactions. We presented and analyzed our MGM approach both theoretically and empirically, and we show that our approach is particularly effective when data available is small. A possible future work could be to explore the approach with generative models other than GANs.

Acknowledgement

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-RP-2020-017). We are thankful for the help from Tianlin Xu on COT-GAN.

References

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, 214–223.

Chen, X.; Li, S.; Li, H.; Jiang, S.; Qi, Y.; and Song, L. 2019. Generative adversarial user model for reinforcement learning based recommendation system. In *International Conference on Machine Learning*, 1052–1061. PMLR.

Durugkar, I.; Gemp, I.; and Mahadevan, S. 2016. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*.

Eléctrica, R. 2021. International interconnections. Available online at: https://www.ree.es/en/activities/operation-of-the-electricity-system/international-interconnections, last accessed on 01 Sep, 2021.

Esteban, C.; Hyland, S. L.; and Rätsch, G. 2017. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*.

Ferdowsi, A.; and Saad, W. 2020. Brainstorming generative adversarial networks (BGANs): Towards multi-agent generative models with distributed private datasets. *arXiv preprint arXiv:2002.00306*.

Flamary, R.; Courty, N.; Gramfort, A.; Alaya, M. Z.; Boisbunon, A.; Chambon, S.; Chapel, L.; Corenflos, A.; Fatras, K.; Fournier, N.; Gautheron, L.; Gayraud, N. T.; Janati, H.; Rakotomamonjy, A.; Redko, I.; Rolet, A.; Schutz, A.; Seguy, V.; Sutherland, D. J.; Tavenard, R.; Tong, A.; and Vayer, T. 2021. POT: Python Optimal Transport. *Journal of Machine Learning Research*, 22(78): 1–8.

Ghosh, A.; Kulharia, V.; Namboodiri, V. P.; Torr, P. H.; and Dokania, P. K. 2018. Multi-agent diverse generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8513–8521.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*.

Hardy, C.; Le Merrer, E.; and Sericola, B. 2019. Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. In *2019 IEEE international parallel and distributed processing symposium (IPDPS)*, 866–877. IEEE.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Hoang, Q.; Nguyen, T. D.; Le, T.; and Phung, D. 2018. MGAN: Training generative adversarial nets with multiple generators. In *International conference on learning representations*.

Karras, T.; Aittala, M.; Hellsten, J.; Laine, S.; Lehtinen, J.; and Aila, T. 2020a. Training Generative Adversarial Networks with Limited Data. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 12104–12114. Curran Associates, Inc.

Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2020b. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8110–8119.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Li, J.; Wang, X.; Lin, Y.; Sinha, A.; and Wellman, M. 2020a. Generating realistic stock market order streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 727–734.

Li, Y.; Zhang, R.; Lu, J.; and Shechtman, E. 2020b. Few-shot image generation with elastic weight consolidation. *arXiv preprint arXiv:2012.02780*.

Liu, M.-Y.; and Tuzel, O. 2016. Coupled generative adversarial networks. *Advances in neural information processing systems*, 29: 469–477.

Mo, S.; Cho, M.; and Shin, J. 2020. Freeze the discriminator: a simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*.

Noguchi, A.; and Harada, T. 2019. Image generation from small datasets via batch statistics adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2750–2758.

Ojha, U.; Li, Y.; Lu, J.; Efros, A. A.; Lee, Y. J.; Shechtman, E.; and Zhang, R. 2021. Few-shot Image Generation via Cross-domain Correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10743–10752.

Pei, Z.; Cao, Z.; Long, M.; and Wang, J. 2018. Multi-adversarial domain adaptation. In *Thirty-second AAAI conference on artificial intelligence*.

Rasouli, M.; Sun, T.; and Rajagopal, R. 2020. Fedgan: Federated generative adversarial networks for distributed data. *arXiv preprint arXiv:2006.07228*.

Robb, E.; Chu, W.-S.; Kumar, A.; and Huang, J.-B. 2020. Few-shot adaptation of generative adversarial networks. *arXiv preprint arXiv:2010.11943*.

Shi, J.-C.; Yu, Y.; Da, Q.; Chen, S.-Y.; and Zeng, A.-X. 2019. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4902–4909.

Sun, H.; Deng, Z.; Chen, H.; and Parkes, D. C. 2020. Decision-aware conditional gans for time series data. *arXiv preprint arXiv:2009.12682*.

Villani, C. 2009. *Optimal transport: old and new*, volume 338. Springer.

Wang, Y.; Gonzalez-Garcia, A.; Berga, D.; Herranz, L.; Khan, F. S.; and Weijer, J. v. d. 2020. Minegan: effective knowledge transfer from gans to target domains with few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9332–9341.

Wang, Y.; Wu, C.; Herranz, L.; van de Weijer, J.; Gonzalez-Garcia, A.; and Raducanu, B. 2018. Transferring gans: generating images from limited data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 218–234.

Xu, T.; Wenliang, L. K.; Munn, M.; and Acciaio, B. 2020. COT-GAN: Generating Sequential Data via Causal Optimal Transport. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 8798–8809. Curran Associates, Inc.

Yoon, J.; Jarrett, D.; and van der Schaar, M. 2019. Time-series Generative Adversarial Networks. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zhao, S.; Liu, Z.; Lin, J.; Zhu, J.-Y.; and Han, S. 2020. Differentiable augmentation for data-efficient gan training. *arXiv preprint arXiv:2006.10738*.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2223–2232.