Decision-Aware Conditional GANs for Time Series Data

He Sun
Harvard University
Boston, MA, USA
he_sun@g.harvard.edu

Zhun Deng
Columbia University
New York, NY, USA
zhun.d@columbia.edu

Hui Chen
MIT
Cambridge, MA, USA
huichen@mit.edu

David C. Parkes
Harvard University
Boston, MA, USA
parkes@eecs.harvard.edu

ABSTRACT

We introduce the *decision-aware time-series conditional generative adversarial network* (DAT-CGAN), a method for the generation of time-series data that is designed to support decision-making. The framework adopts a multi-Wasserstein loss on decision-related quantities and an overlapped block-sampling approach for sample efficiency. We characterize the generalization properties of DAT-CGAN and in application to a multi-period portfolio choice problem and financial time series data, we demonstrate better training stability and generative quality in regard to both raw data and decision-related quantities than strong GAN-based baselines.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning theory**; **Neural networks**; **Generative and developmental approaches**; • **Mathematics of computing** → **Time series analysis**; *Stochastic models*; • **Applied computing** → *Banking*.

KEYWORDS

Time Series, Simulator, Decision Aware, GAN

ACM Reference Format:
He Sun, Zhun Deng, Hui Chen, and David C. Parkes. 2023. Decision-Aware Conditional GANs for Time Series Data. In *4th ACM International Conference on AI in Finance (ICAIF '23), November 27–29, 2023, Brooklyn, NY, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3604237.3626855

1 INTRODUCTION

High-fidelity data generators ("simulators") are desirable across many domains as a result of paucity of data or because of high stakes in the deployment of automated decision methods. A good simulator can be used to improve and evaluate the performance of decision methods. A typical approach to the design of data generators to use GANs but a gap in current approaches is that they are not *decision-aware* but focus instead on modeling the raw data distribution [12, 29]. A problem with this is that the approximation error in the raw synthetic data distribution can be further amplified, after the transformation through decision-related functions, making the approximation to the underlying decision-related quantities unreliable.

We show that taking decision-related quantities into account during training improves the effectiveness of GANs in support of decision making and helps to stabilize the training process. We do this by introducing *decision-related quantities* to the loss function. We focus on time-series data, and consider settings where decisions are made over time and based on multi-step inference.

In such settings, data scarcity can be a particular challenge, either due to a limited sample size or non-stationarity. [1] Previous work has applied bootstrap methods to improve the sample efficiency of estimators for time-series data [8], but without introducing decision-related quantities or providing finite-sample guarantees on generalization error.

Another challenge when training sequential generative models is *exposure bias* [22]. This arises when models are trained to predict one-step forward using previous ground truth observations, whereas at test time they are used to generate an entire sequence. As a result, the generated data distribution can diverge from the training distribution, with accumulating errors. Although exposure bias has received attention in language models [5, 23], this problem remains present in other generative model applications [15].

Our Contributions. We propose a novel, *decision-aware time-series conditional generative adversarial network (DAT-CGAN).* The training procedure is made decision aware by adopting a *multi-Wasserstein loss structure* on decision-related quantities in addition to raw data. In particular, the generator needs to capture the structural relationship between different decision-related quantities. We provide the discriminator with access to the same amount of conditioning information as the generator to avoid it being too strong relative to the generator. We also address exposure bias by *aligning training and evaluation with the same number of look-ahead steps* and we improve sample efficiency by adopting an *overlapped block-sampling mechanism.*

We provide a theoretical characterization of DAT-CGAN, giving finite-sample generalization bounds.

In experimental results, we evaluate DAT-CGANs on a portfolio choice problem for a risk-averse investor [19]. A high-quality

ICAIF '23, November 27–29, 2023, Brooklyn, NY, USA

ACM ISBN 979-8-4007-0240-2/23/11…$15.00
https://doi.org/10.1145/3604237.3626855

[1] For example, weekly financial data series only provide around 50 observations per year, and pooling across multiple years is not effective due to potential distributions shifts. Even for high-frequency data, distribution shift remains a concern due to the potential changes in the composition of market participants and trading rules over time.

simulator will help an investor characterize the distribution of portfolio returns more reliably than relying on simplistic parametric assumptions such as normality. The results show that introducing portfolio-decision-related quantities into the loss function, in addition to asset returns, provides better fit to quantities of interest than strong GAN-based baselines.

Related Work. The literature on GANs for time-series data does not consider decision awareness, even for the non-time series context, and does not provide theoretical guarantees for decision-related quantities or conditional GANs with overlapped sampling schemes [21, 26, 28–30]. In the context of financial markets, Li et al. [15] introduce *stock-GAN* for the generation of order streams, and evaluate their approach on stylized facts about market microstructure. Koshiyama et al. [12] study the use of GANs for generating additional synthetic samples to help with model calibration and aggregation. Neither method is decision aware, and they are similar to the baselines we use in ablations (specifically, they are akin to a "1-step, return-data only" baseline). There is also work that makes use of GANs to perform anomaly detection in time series data [4, 11, 13, 14], for imputation for multivariate time series [16, 17], and to study causal effects in economic models [3]. In the context of enhacing performance of deep learning, other research introduces auxilary loss to an intermediate layer of the Inception Network and use it as a regularizer during the training process [25]. In regard to decision-focused learning, Mandi et al. [18] frame this as a ranking problem and introduce ranking loss, whereas Elmachtoub and Grigas [10] introduce a loss function that measures the decision errors induced by predictions.

2 WASSERSTEIN GANS FOR TIME SERIES GENERATION

A Wasserstein GAN uses the *Wasserstein distance* as the loss function. The Wasserstein distance between two random variables, r and r', distributed according to $\mathcal{P}_r$ and $\mathcal{P}_{r'}$, is

$$W(\mathcal{P}_r, \mathcal{P}_{r'}) = \inf_{\Gamma \in \Pi(\mathcal{P}_r, \mathcal{P}_{r'})} \mathbb{E}_{(r,r')\sim\Gamma}[\|r - r'\|], \tag{1}$$

where $\|\cdot\|$ is the L^2 norm, and $\Pi(\mathcal{P}_r, \mathcal{P}_{r'})$ is the set of all joint distributions whose marginals equal to $\mathcal{P}_r$ and $\mathcal{P}_{r'}$. According to the Kantorovich-Rubinstein duality [27], the dual form can be written as:

$$\sup_{\{h:\|h\|_L \leqslant 1\}} \mathbb{E}_{r\sim\mathcal{P}_r}[h(r)] - \mathbb{E}_{r'\sim\mathcal{P}_{r'}}[h(r')], \tag{2}$$

where $\|h\|_L$ is defined as $\sup_{x,x'} |h(x) - h(x')|/\|x - x'\|$. For Wasserstein GANs, the goal is to minimize the Wasserstein distance between the non-synthetic data and synthetic data.[2] Following Mirza and Osindero [20], we work with *Conditional GANs*, allowing for conditioning variables. For functions D_θ and G_η, the *discriminator* parameterized by θ and the *generator* parametrized by η, and with conditioning variable x, the CGAN problem is

$$\min_\eta \max_\theta \mathbb{E}_{r\sim\mathcal{P}(r|x)}[D_\theta(r, x)] - \mathbb{E}_{z\sim\mathcal{P}(z)}[D_\theta(G_\eta(z, x), x)],$$

where $\mathcal{P}(r|x)$ and $\mathcal{P}(z)$ denote the distribution of non-synthetic data and input random seed, respectively. Here, the synthetic data comes from the generator, with $r' = G_\eta(z, x)$ conditioning on x.

[2]We use Wasserstein loss because it tends to improve the learning process stability relative to other choices, for example in regard to mode collapse, and to yield interpretable learning curves [1].

3 DECISION-AWARE TIME SERIES CONDITIONAL GENERATIVE ADVERSARIAL NETWORK

Let $(r_1, \ldots, r_T)$ denote a multivariate time series, where r_t in time t is a d-dimensional column vector. Let x_t denote an m-dimensional *time-series information vector*, summarizing relevant information up to time t. Let $R_{t+1:t+k} = (r_{t+1}, \ldots, r_{t+k})$ denote a *k-length block after time t*, where $k \in \{1, \ldots, K\}$ is the kth look-ahead step, and K is the total number of *look-ahead steps* to generate. To accommodate settings where the sample size is very limited, we allow the $R_{t+1:t+k}$ blocks to overlap with each other for different t and k, in order to fully utilize the samples.[3]

To model the decision process of an end user, let $f_{j,k}(R_{t+1:t+k}, x_t)$ denote the jth decision-related quantity (a scalar, vector, or matrix), for $j \in \{1, \ldots, J\}$, in look-ahead step k, and define $f_k(R_{t+1:t+k}, x_t)$ as

$$(f_{1,k}(R_{t+1:t+k}, x_t), \ldots, f_{J,k}(R_{t+1:t+k}, x_t))^\intercal. \tag{3}$$

This represents the J decision-related quantities at look-ahead step k given data $R_{t+1:t+k}$ and information x_t. In finance, $f_{j,k}(R_{t+1:t+k}, x_t)$ could be the estimated co-variance of asset returns, or the portfolio weights, both determined using the information up to time $t + k$.

Multi-Wasserstein loss. Let $r'_{t+k|t}$ denote the *synthetic data* generated from information vector x_t for look-ahead step k, for $k \in \{1, \ldots, K\}$. We use notation $r'_{t+k|t}$ rather than notation r'_{t+k} because there is a difference, for example, between $r'_{12|9}$ and $r'_{12|10}$, where $r'_{12|9}$ is the synthetic data generated for day 12 conditioning on information up to day 9, and $r'_{12|10}$ is the synthetic data for day 12 conditioning on information up to day 10. For all t, all k, we want the conditional distribution on synthetic data, $\mathcal{P}(r'_{t+k|t}|x_t)$, where x_t is discrete, to match the conditional distribution on the non-synthetic data, $\mathcal{P}(r_{t+k}|x_t)$. Similarly, for all t, all k, we want the conditional distribution on decision-related quantities for synthetic data, $\mathcal{P}(f_{j,k}(R'_{t+1:t+k}, x_t)|x_t)$, where $R'_{t+1:t+k} = (r'_{t+1|t}, \ldots, r'_{t+k|t})$, to match the conditional distribution, $\mathcal{P}(f_{j,k}(R_{t+1:t+k}, x_t)|x_t)$, on quantities computed for non-synthetic data; see Figure 1. It will be convenient to write $\mathcal{P}(R'_{t+1:t+K}|x_t)$ for $\{\mathcal{P}(r'_{t+k|t}|x_t)\}_{k\in\{1,\ldots,K\}}$.

Adopting a separate loss term for each quantity and each look-ahead step k, we define the following multi-Wasserstein objective (written here for conditioning, x_t):

$$\begin{aligned}
&\inf_{\mathcal{P}(R'_{t+1:t+K}|x_t)} \sum_{k=1}^{K} \omega_k L^r_k + \sum_{k=1}^{K}\sum_{j=1}^{J} \lambda_{j,k} L^f_{j,k}, \\
&L^r_k = W\big(\mathcal{P}(r_{t+k}|x_t), \mathcal{P}(r'_{t+k|t}|x_t)\big) \\
&L^f_{j,k} = W\big(\mathcal{P}(f_{j,k}(R_{t+1:t+k}, x_t)|x_t), \\
&\quad \mathcal{P}(f_{j,k}(R'_{t+1:t+k}, x_t)|x_t)\big),
\end{aligned} \tag{4}$$

[3]In finance, r_{t+1} could be the asset returns at day $t + 1$, x_t the past asset returns, volatility, and other technical indicators, and $R_{t+1:t+k}$ the k-days forward asset returns (see Figure 1).

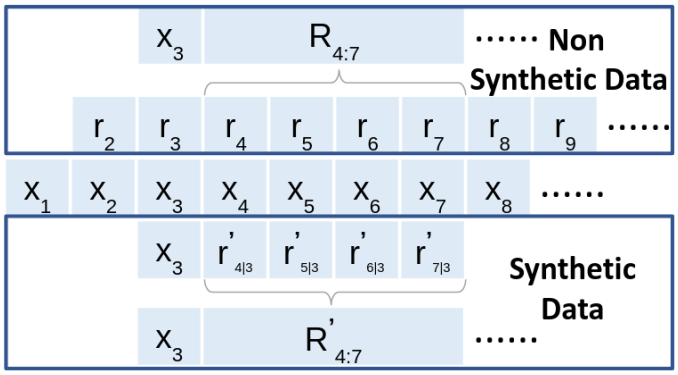


Figure 1: Non-synthetic data and synthetic data generated by conditional GANs, shown for $K = 4$.

where L_k^r denotes the loss for data at k steps forward, $L_{j,k}^f$ the loss for decision-related quantity j at k steps forward, and where values $\omega_k > 0$ and $\lambda_{j,k} > 0$ are weights.[4]

Surrogate loss. Let D_{γ_k} denote the discriminator at look-ahead step k, with parameters γ_k, and let $D_{\theta_{j,k}}$ denote the discriminator for decision-related quantity j at look-ahead step k, with parameters $\theta_{j,k}$. Let $r'_{t+k|t} = G_\eta(z_{t,t+k}, x_t)$ denote the synthetic data at look-ahead step k, where G_η is the generator with parameters η, and with noise $z_{t,t+k} \sim N(0, I_d)$. Let $Z_{t,t+k} = (z_{t,t+1}, \ldots, z_{t,t+k})$ denote a k-length block of random seeds after t. We define the following quantities:

$$\mathbb{E}_k^r = \mathbb{E}_{r_{t+k} \sim \mathcal{P}(r_{t+k}|x_t)}[D_{\gamma_k}(r_{t+k}, x_t)], \tag{5}$$

$$\mathbb{E}_k^{G_\eta} = \mathbb{E}_{z_{t,t+k} \sim \mathcal{P}(z_{t,t+k})}[D_{\gamma_k}(r'_{t+k|t}, x_t)], \tag{6}$$

$$\mathbb{E}_{j,k}^{f,R} = \mathbb{E}_{R_{t+1:t+k} \sim \mathcal{P}(R_{t+1:t+k}|x_t)}[D_{\theta_{j,k}}(f_{j,k}(R_{t+1:t+k}, x_t), x_t)], \tag{7}$$

$$\mathbb{E}_{j,k}^{f,G_\eta} = \mathbb{E}_{Z_{t,t+k} \sim \mathcal{P}(Z_{t,t+k})}[D_{\theta_{j,k}}(f_{j,k}(R'_{t+1:t+k}, x_t), x_t)], \tag{8}$$

where (5) and (7) are defined on non-synthetic data and (6) and (8) on synthetic data, and (7) and (8) are defined on derived, decision-related quantities.

To formulate the DAT-CGAN training problem, we use the Kantorovich-Rubinstein duality for each Wasserstein distance in (4) and sum over the dual forms [27]. This provides a *surrogate loss*, upper bounding the original objective. The surrogate problem is a min-max optimization problem, with the *discriminator loss* defined as

$$\inf_\eta \sup_{\gamma_k, \theta_{j,k}} \sum_{k=1}^K \omega_k (\mathbb{E}_k^r - \mathbb{E}_k^{G_\eta}) + \sum_{k=1}^K \sum_{j=1}^J \lambda_{j,k} (\mathbb{E}_{j,k}^{f,R} - \mathbb{E}_{j,k}^{f,G_\eta}), \tag{9}$$

and the *generator loss* defined as

$$\inf_\eta - \sum_k \omega_k \mathbb{E}_k^{G_\eta} - \sum_{k,j} \lambda_{j,k} \mathbb{E}_{j,k}^{f,G_\eta}. \tag{10}$$

[4]An alternative formulation would impose the Wasserstein distance on a vector concatenating all quantities. We justify this design choice in the experimental results section.

Algorithm 1 . Learning Rate $\alpha = 1e-5$, $\omega_k = \lambda_{j,k} = 0.8^k$, $s_D = 1$, $s_G = 5$, clipping $l_b = -0.5, u_b = 0.5$, $T = 3500$, look-ahead step $K = 4$, batch size $I = 32$, training steps $N = 2e5$.

1: **Require:** $\gamma_{k,0}$ and $\theta_{j,k,0}$, initial discriminator parameters; η_0, initial generator parameters.
2: **for** $t = 1, k = 1$ **to** T, K **do**
3: Compute $R_{t+1:t+k}$ and $f_{j,k}(R_{t+1:t+k}, x_t), \forall j$
4: **end for**
5: **while** $n < N$ **do**
6: **for** $s = 0$ **to** s_D **do**
7: Make I samples of K-size time blocks.
The ith sample ($1 \leqslant i \leqslant I$) ranges from time $t_i + 1$ to $t_i + K$, and consists of data $(r_{t_i+k}, f_k(R_{t_i+1:t_i+k}, x_{t_i}), x_{t_i})_{k=1}^K$
8: **for** $i = 1, k = 1$ **to** I, K **do**
9: Sample $z_{t_i,t_i+k} \sim \mathcal{P}(z_{t_i,t_i+k})$; Compute $r'_{t_i+k|t_i} = G_\eta(z_{t_i,t_i+k}, x_{t_i})$; Compute $f_{j,k}(R'_{t_i+1:t_i+k}, x_{t_i}), \forall j$
10: **end for**
11: **for** $k = 1$ **to** K **do**
12: $\gamma_k \leftarrow \text{clip}(\gamma_k + \alpha\omega_k \nabla_{\gamma_k} \sum_{k=1}^K [\hat{\mathbb{E}}_k^r - \hat{\mathbb{E}}_k^{G_\eta}], l_b, u_b)$
13: $\theta_{j,k} \leftarrow \text{clip}(\theta_{j,k} + \alpha\lambda_{j,k} \nabla_{\theta_{j,k}} \sum_{k=1}^K [\hat{\mathbb{E}}_{j,k}^{f,R} - \hat{\mathbb{E}}_{j,k}^{f,G_\eta}], l_b, u_b), \forall j$
14: **end for**
15: **end for**
16: **for** $s = 0$ **to** s_G **do**
17: **for** $i = 1, k = 1$ **to** I, K **do**
18: Sample $z_{t_i,t_i+k} \sim \mathcal{P}(z_{t_i,t_i+k})$; Compute $r'_{t_i+k|t_i} = G_\eta(z_{t_i,t_i+k}, x_{t_i})$; Compute $f_{j,k}(R'_{t_i+1:t_i+k}, x_{t_i}), \forall j$
19: **end for**
20: $\eta \leftarrow \eta - \alpha\omega_k \nabla_\eta \sum_{k=1}^K [\hat{\mathbb{E}}_k^{G_\eta} - \hat{\mathbb{E}}_{j,k}^{f,G_\eta}], \forall j$
21: **end for**
22: **end while**

We also write $\tilde{L}_k^r = \mathbb{E}_k^r - \mathbb{E}_k^G$ and $\tilde{L}_{j,k}^f = \mathbb{E}_{j,k}^{f,R} - \mathbb{E}_{j,k}^{f,G_\eta}$, to denote the *discriminator loss* for the raw data and decision-related quantities, respectively.

Training procedure. See Algorithm 1. Lines 2-3 prepare the data. Lines 6-15 train the discriminators: Line 7 performs K-length block sampling; Lines 8-10 generate synthetic block samples for each time block, conditioning on the information vector; Lines 11-14 update the discriminators. Lines 16-21 train the generators: Lines 17-19 generate synthetic block samples for each time block, conditioning on the information vector; Line 20 updates the generators. We define sample estimates for expectations (5), (6), (7), (8), as $\hat{\mathbb{E}}_k^r$, $\hat{\mathbb{E}}_k^{G_\eta}$, $\hat{\mathbb{E}}_{j,k}^{f,R}$ and $\hat{\mathbb{E}}_{j,k}^{f,G_\eta}$, respectively. Quantities $(r_{t_i+k}, f_k(R_{t_i+1:t_i+k}, x_{t_i}), x_{t_i}), \forall i$ are obtained by an overlapped block sampling scheme (see Figure 1), where different blocks of samples can overlap with other blocks.

4 THEORETICAL RESULTS

In this section, we give a finite-sample generalization bound for DAT-CGAN. Previous techniques for generalization bounds for

GANs with i.i.d. data [2] have not considered the overlapping structures or derived quantities in our algorithm and to provide a generalization bound we need to extend the previous arguments to the present case. We only keep the assumptions and main theorems here, but leave the proof details to Sun et al. [24].

Arora et al. [2] have shown that training results for GANs that appear successful may be far from the target distribution in terms of standard metrics, such as Jensen-Shannon (JS) divergence or Wasserstein distance—even though the synthetic data distribution is close to the empirical distribution induced by the samples, it can still be far from the underlying true distribution under those metrics. For example, the Wasserstein distance between two empirical distributions $\hat{\mu}$ and $\hat{\nu}$, both induced by m training samples, can be 0, while the distance between the true underlying corresponding distributions μ and ν can be larger than a constant unless m is exponentially large w.r.t. the data dimension, which is usually very high. However, in practice, generalization occurs with respect to a weaker version of distance, i.e. *neural network distance*, defined in Definition 4.1. In practice, when training WGAN, we are not optimizing the real Wasserstein distance between synthetic and real data, but a distance approximated by neural networks:

$$\min_{\eta}\max_{\theta}\mathbb{E}_{r\sim\mathcal{P}_r}[D_\theta(r)] - \mathbb{E}_{z\sim\mathcal{P}(z)}[D_\theta(G_\eta(z))], \tag{11}$$

where D_θ and G_η are instantiated as neural networks, parameterized by θ and η. Here, r is the real data while $G_\eta(z)$ is the synthetic data with random seed z. Arora et al. [2] consider the following, weaker metric:

Definition 4.1 (Neural Net Distance for WGAN)**.** Given a family of neural networks $\{D_\theta : \theta \in \Theta\}$ for a set Θ, for two distributions μ and ν, the corresponding *neural network distance* for the Wasserstein GAN is defined as,

$$\mathscr{D}_\Theta(\mu,\nu) = \sup_{\theta\in\Theta}\{\mathbb{E}_{x\sim\mu}[D_\theta(x)] - \mathbb{E}_{x\sim\nu}[D_\theta(x)]\}. \tag{12}$$

With this, Arora et al. [2] build a generalization theory for WGANs under the following generalization property.

Definition 4.2. Let $\mathcal{P}_{\text{data}}$ denote the distribution of non-synthetic data and $\mathcal{P}_G$ denotes the generated distribution, and let $\hat{\mathcal{P}}_{\text{data}}$ and $\hat{\mathcal{P}}_G$ denote the corresponding empirical versions, the generalization gap for WGAN is defined as

$$|\mathscr{D}_\Theta(\hat{\mathcal{P}}_{\text{data}},\hat{\mathcal{P}}_G) - \mathscr{D}_\Theta(\mathcal{P}_{\text{data}},\mathcal{P}_G)|. \tag{13}$$

A natural question in our setting is the following: *for DAT-CGAN, can we give a generalization property guarantee under the neural network distance?*

To build such a theory for DAT-CGAN, instead of dealing with i.i.d. data as in Arora et al. [2], we need to deal with time series and overlapping block sampling as well as the conditioning information. In this section, we will show how to conquer such issues and provide a theoretical guarantee. Instead of considering a multi-step multi-loss, it is WLOG to consider the case when $k = K$ and a single decision-related quantity (the raw data can also be viewed as a decision-related quantity, where the corresponding f is the mapping picking the last element in $R_{t_i+1:t_i+K}$). For multiple but finite values of k, and multiple but finite decision-related quantities, we can use a uniform bound to obtain the corresponding generalization bounds. Given this, we can simplify notation: let $\hat{\mathcal{P}}_R(I)$ and $\hat{\mathcal{P}}_{G_\eta,Z}(I)$ denote the empirical distribution induced by data set $\{(f(R_{t_i+1:t_i+K},x_{t_i}),x_{t_i})\}_{i=1}^I$ and $\{(f(R'_{t_i+1:t_i+K},x_{t_i}),x_{t_i})\}_{i=1}^I$, respectively. Recall

$$R'_{t_i+1:t_i+K} = (G_\eta(z_{t_i,t_i+1},x_{t_i}),\ldots,G_\eta(z_{t_i,t_i+K},x_{t_i})), \tag{14}$$

and define

$$\mathscr{D}_\Theta(\hat{\mathcal{P}}_R(I),\hat{\mathcal{P}}_{G_\eta,Z}(I)) = \sup_{\theta\in\Theta}[\hat{\mathbb{E}}^{f,R} - \hat{\mathbb{E}}^{f,G_\eta}], \tag{15}$$

$$\hat{\mathbb{E}}^{f,R} = (1/I)\sum_{i=1}^{I}[D_\theta(f(R_{t_i+1:t_i+K},x_{t_i}),x_{t_i})],$$

$$\hat{\mathbb{E}}^{f,G_\eta} = (1/I)\sum_{i=1}^{I}[D_\theta(f(R'_{t_i+1:t_i+K},x_{t_i}),x_{t_i})].$$

Here, Θ and Ξ are parameter sets. Before formally stating the theoretical results, we need to understand the convergence point of $\mathscr{D}_\Theta(\hat{\mathcal{P}}_R(I),\hat{\mathcal{P}}_{G_\eta,Z}(I))$. Notice that for the surrogate loss, taking expectation with respect to $\mathcal{P}(r_{t+K}|x_t)$, for any realization of x_t, i.e. $x_t = c$ for constant vector c, we need enough samples for r_{t+K} given $x_t = c$ so that the empirical distribution $\hat{\mathcal{P}}(r_{t+K}|x_t = c)$ can well represent the ground-truth distribution $\mathcal{P}(r_{t+K}|x_t = c)$. However, in applications, we would not normally have enough samples for any arbitrary value c, and especially considering that x_t may be a continuous random vector instead of a categorical one. It is even possible that for all $\{t_i\}_{i=1}^I$, the $\{x_{t_i}\}_{i=1}^I$ values are different from each other. Thus, we need to understand what $\mathscr{D}_\Theta(\hat{\mathcal{P}}_R(I),\hat{\mathcal{P}}_{G_\eta,Z}(I))$ converge to as $I\to\infty$. We show that $\mathscr{D}_\Theta(\hat{\mathcal{P}}_R(I),\hat{\mathcal{P}}_{G_\eta,Z}(I))$ converges to a "weaker" version for a given η under certain conditions, i.e., that it converges to

$$\mathscr{D}_\Theta(\mathcal{P}_R,\mathcal{P}_{G_\eta,Z}) = \sup_{\theta\in\Theta}[\mathbb{E}^{f,R} - \mathbb{E}^{f,G_\eta}], \tag{16}$$

where $\mathcal{P}_R$ and $\mathcal{P}_{G_\eta,Z}$ are the distribution of $(f(R_{t+1:t+K},x_t),x_t)$ and $(f(R'_{t+1:t+K},x_t),x_t)$, respectively, and

$$\mathbb{E}^{f,R} = \mathbb{E}_{x_t}\mathbb{E}_{R_{t+1:t+K}\sim\mathcal{P}(R_{t+1:t+K}|x_t)}[D_\theta(f(R_{t+1:t+K},x_t),x_t)], \tag{17}$$

$$\mathbb{E}^{f,G_\eta} = \mathbb{E}_{x_t}\mathbb{E}_{Z_{t,t+K}\sim\mathcal{P}(Z_{t,t+K})}[D_\theta(f(R'_{t+1:t+K},x_t),x_t)]. \tag{18}$$

Compared with the surrogate loss mentioned previously, such as Eq. (5), there is an extra expectation over x_t in $\mathscr{D}_\Theta(\mathcal{P}_R,\mathcal{P}_{G_\eta,Z})$, which comes from sampling over different $\{x_{t_i}\}$'s. We can view this as an average version of the surrogate losses under different realizations of x_t's. Now we are ready to state a generalization bound regarding

$$|\mathscr{D}_\Theta(\hat{\mathcal{P}}_R(I),\hat{\mathcal{P}}_{G_\eta,Z}(I))) - \mathscr{D}_\Theta(\mathcal{P}_R,\mathcal{P}_{G_\eta,Z})|.$$

In order to conquer the issues with non-i.i.d. data and overlapping sampling, we introduce a framework for defining suitable *mixing conditions*. This kind of framework is commonly used in time-series analysis [7].

Mixing condition framework. Let $X_i \in S$ for some set S, and $X = (X_1,,\cdots,X_n)$. We further denote $X_i^j = (X_i,X_{i+1},\cdots,X_j)$ as a random vector for $1 \leqslant i < j \leqslant n$. Correspondingly, we let $x_i^j = (x_i,x_{i+1},\cdots,x_j)$ be a subsequence for the realization of X, i.e. $(x_1,x_2,\cdots,x_n)$. We denote the set $C = \{\boldsymbol{y} \in S^{i-1}, w, w' \in S : \mathbb{P}(X_1^i = \boldsymbol{y}w) > 0,\ \mathbb{P}(X_1^i = \boldsymbol{y}w') > 0\}$, and write $\bar{\eta}_{i,j}(\{X_i\}_{i=1}^n) =$

$\sup_C \eta_{i,j}(\boldsymbol{y}, w, w')$, where
$\eta_{i,j}((\{X_i\}_{i=1}^n, \boldsymbol{y}, w, w')$ denotes

$$TV\Big(\mathcal{P}(X_j^n|X_1^i = \boldsymbol{y}w), \mathcal{P}(X_j^n|X_1^i = \boldsymbol{y}w')\Big). \tag{19}$$

Here, TV is the *total variation distance* and $\mathcal{P}(X_j^n|X_1^i = \boldsymbol{y}w)$ is the distribution of X_j^n conditioning on $\{X_1^i = \boldsymbol{y}w\}$.

Assumptions and implications. We make a number of natural boundedness assumptions. We assume the time series data have bounded support with a universal B_r, such that $\max\{\|r_i\|_\infty, \|r_i\|\} \leqslant B_r$,

where the boundedness of $\|r_i\|$ is implied by the boundedness of $\|r_i\|_\infty$ since the dimension of r_i is finite. We assume boundedness of conditioning information $\{x_t\}_t$ with a universal B_x, such that $\max\{\|x_t\|_\infty, \|x_t\|\} \leqslant B_x$.

For the discriminators D_γ and D_θ, where $\theta \in \Theta \subseteq \mathbb{R}^p$, we assume w.l.o.g. that Θ is a subset of unit balls with corresponding dimensions.[5] Similarly, for the generative model G_η, $\eta \in \Xi$, we assume Ξ is a subset of unit ball.

We also require L-Lipschitzness of D_θ and G_η with respect to their parameters, i.e. $\|D_{\theta_1}(x) - D_{\theta_2}(x)\| \leqslant L\|\theta_1 - \theta_2\|$ for any x (similar for G_η), as well as the boundedness of the output range of G, with a Δ such that $\max\{\|G_\eta(x)\|, \|G_\eta(x)\|_\infty\} \leqslant \Delta$ for any input x. To characterize the mixing conditions, we assume there exists a universal function β, such that
$\max\{\bar{\eta}_{i,j}(\{(r_i, x_i)\}_{i=1}^T), \bar{\eta}_{i,j}(\{x_i\}_{i=1}^T)\} \leqslant \beta(|j-i|))$,
and with $\Delta_\beta = \sum_{k=1}^\infty \beta(k) < \infty$, where β's are the *mixing coefficients*. Lastly, and as holds for the Wasserstein GAN, there exists a constant $\tilde{L}$, such that $\|D_\theta(x) - D_\theta(x')\| \leqslant \tilde{L}\|x - x'\|$ for all θ. We first claim the boundedness of the decision-related quantities in DAT-CGAN. We defer the proofs of Lemma 4.3 and Theorem 4.4 to Sun et al. [24].

Lemma 4.3 (Boundedness of decision-related quantities)**.** *Under the assumptions above, the decision-related quantities we considered are all bounded, where the bounds are universal and only depend on B_r.*

Let B_f denote the bound of the decision-related quantity, $\max\{\|f(R_{t_i+1:t_i+K}, x_{t_i})\|, \|f(R'_{t_i+1:t_i+K}, x_{t_i})\|\} \leqslant B_f$ for all i. By Lemma 4.3, we obtain the following generalization bound for $|\mathcal{D}_\Theta(\hat{\mathcal{P}}_R(I), \hat{\mathcal{P}}_{G_\eta,Z}(I))) - \mathcal{D}_\Theta(\mathcal{P}_R, \mathcal{P}_{G_\eta,Z})|$, for each iteration of the training process (referring to each round of the mix-max optimization of CGANs).

Theorem 4.4. *Under the assumptions above, suppose $G_{\eta_1}, G_{\eta_2}, \cdots, G_{\eta_M}$ be the M generators in the M iterations of the training, let $B_* = \sqrt{B_f^2 + B_x^2}(K + \Delta_\beta)$, then*

$$\sup_{j\in[M]} |D_\Theta(\hat{\mathcal{P}}_R(I), \hat{\mathcal{P}}_{G_{\eta_j},Z}(I)); \eta) - D_\Theta(\mathcal{P}_R, \mathcal{P}_{G_{\eta_j},Z})| \leqslant \varepsilon,$$

with probability of at least

$$1 - C\exp\Big(p\log(\frac{pL}{\varepsilon})\Big)(1+M)\exp\Big(-\frac{I\varepsilon^2}{\tilde{L}^2 B_*^2}\Big),$$

for some constant $C > 0$.

[5] We can always rescale the parameter properly by changing the parameterization as long as Θ is bounded. The boundedness of Θ is naturally satisfied since the training algorithm of the Wasserstein GAN requires weight clipping.

Theorem 4.4 provides, whether for raw data or a decision-related quantity, that the distribution on non-synthetic data is close to the generated distribution at every iteration in training. As with Arora et al. [2], we obtain an exponential tail bound, in our case with a constant that also involves the mixing coefficient and sampling block size.

5 DAT-CGAN FOR PORTFOLIO CHOICE

We apply the DAT-CGAN to portfolio choice, where an investor wants to understand the properties of a portfolio strategy. A good generator ("simulator") should not only generate synthetic asset return data but also support the high-fidelity generation of decision-related quantities that are relevant for portfolio choice.

Specifically, we assume the investor formulates a *mean-variance portfolio optimization problem* in choosing the portfolio weights. They want to invest across a number of assets, considering the portfolio return and the portfolio risk. Let r_{t+k+1} denote the *asset return vector* at time $t+k+1$, for look-ahead step $k+1$. Let x_t denote the conditioning variables at time t. We use $w_{t+k|t}$ to denote the portfolio weights decided at time $t+k$, and traded on at time $t+k+1$. For non-synthetic data, the portfolio optimization problem at time $t+k$ can be written as:

$$\max_{w_{t+k|t}^\intercal \mathbf{1}=1} \quad w_{t+k|t}^\intercal \hat{u}_{t+k|t} - \phi \cdot w_{t+k|t}^\intercal \hat{\Sigma}_{t+k|t} w_{t+k|t}, \tag{20}$$

where $\phi > 0$ is the risk preference parameter, and with estimated mean and co-variance of asset returns, $\hat{u}_{t+k|t}$ and $\hat{\Sigma}_{t+k|t}$, respectively.

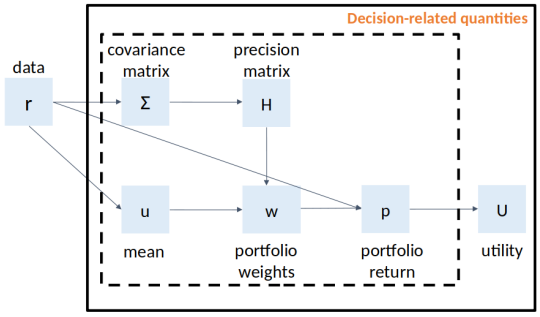
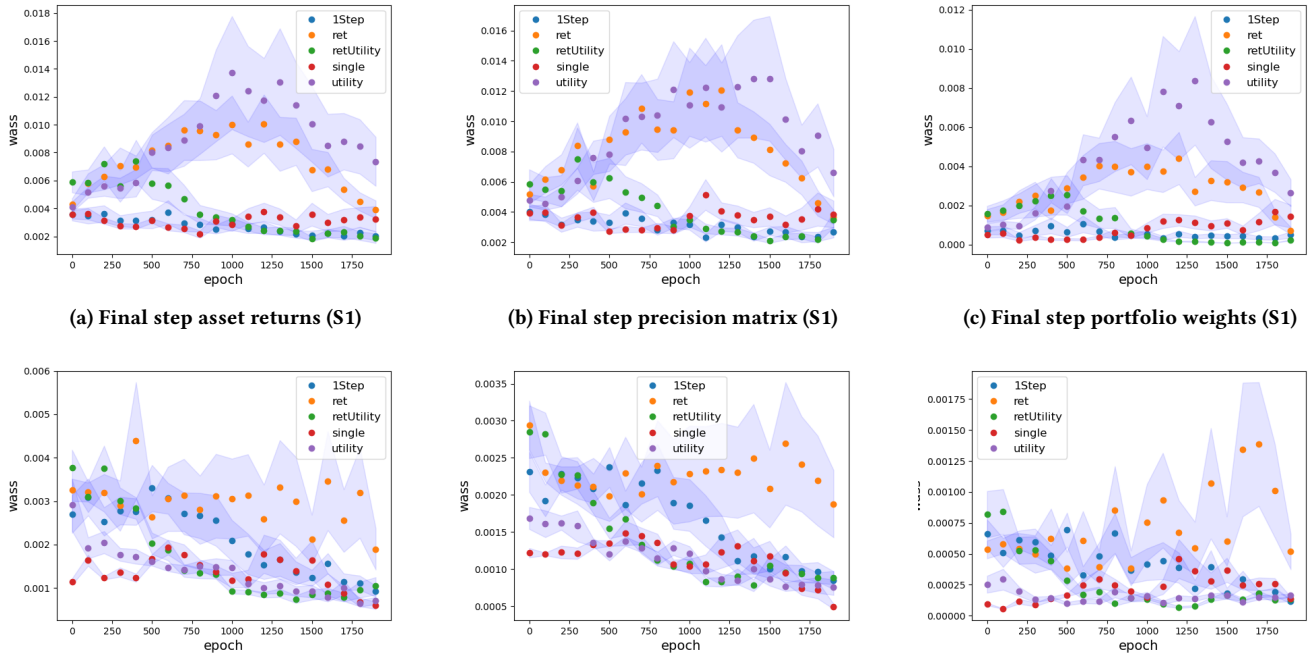


Figure 2: Decision-related quantities in the portfolio selection problem.

These estimators are defined on non-synthetic asset returns as, $\hat{u}_{t+k|t} = f_{u,k}(R_{t+1:t+k}, x_t) = \mathrm{MA}_\zeta(r_{t+k})$ and $\hat{\Sigma}_{t+k|t} = f_{\Sigma,k}(R_{t+1:t+k}, x_t) = \mathrm{MA}_\zeta(r_{t+k}r_{t+k}^\intercal) - \hat{u}_{t+k|t}^2$. Here, $\mathrm{MA}_\zeta(r_{t+k}) = \zeta \cdot \mathrm{MA}_\zeta(r_{t+k-1}) + (1-\zeta)\cdot r_{t+k}$ is a moving average operator, and $\zeta > 0$ a smoothing parameter. We set $\zeta = 0.74$. The analytical solution to the investment problem is,

$$w_{t+k|t} = \frac{2\hat{H}_{t+k|t}}{\phi}(\hat{u}_{t+k|t} - \frac{\mathbf{1}^\intercal \hat{H}_{t+k|t}\hat{u}_{t+k|t}\mathbf{1} - 2\phi\mathbf{1}}{\mathbf{1}^\intercal \hat{H}_{t+k|t}\mathbf{1}}), \tag{21}$$

where $\hat{H}_{t+k|t}$ is the estimated precision matrix ($\hat{\Sigma}_{t+k|t}^{-1}$) of asset returns. $\hat{H}_{t+k|t} = ((1-\tau)\hat{\Sigma}_{t+k|t} + \tau\Lambda)^{-1}$ using the shrinkage method [9], where Λ is the identity matrix and $\tau > 0$ is a shrinkage parameter.



(a) Final step asset returns (S1)

(b) Final step precision matrix (S1)

(c) Final step portfolio weights (S1)

(d) Final step asset returns (ETF)

(e) Final step precision matrix (ETF)

(f) Final step portfolio weights (ETF)

Figure 3: Wasserstein Distance between non-synthetic and synthetic data for artificial environments S1 (first row), and for real data comprised of four U.S. ETFs (second row). 3a and 3d asset returns; 3b and 3e estimated precision matrix; 3c and 3f portfolio weights. An epoch is one full pass of the data. Confidence bands are computed over 5 runs.

The investor is interested in the *realized portfolio return*, $p_{t+k+1|t} = w_{t+k|t}^{\intercal} r_{t+k+1|t}$, and the realized *utility of the portfolio return* given the risk preference, defined as $U_{t+k+1|t} = p_{t+k+1|t} - \phi p_{t+k+1|t}^2$.

We give the relationship between the various decision-related quantities in Figure 2.

These decision-related quantities are generated based on conditioning variables that reflect market conditions. We use the chain rule to take derivatives through the portfolio optimization problem during training, making use of the closed-formed solution (21).

6 EXPERIMENTAL RESULTS

We study two different kinds of environments. A first kind of environment is an artificial environment, while the second kind of environment makes use of real time series data that comprises a basket of ETF time series. *To avoid ambiguity, we emphasize here that we adopt the phrase "artificial" to refer to the first environment. We always using "synthetic" to refer to the data generated by the DAT-CGAN and other baselines, whether in the first or second kind of environment.*

Experimental setup. We study a generation problem with $K = 4$ lookahead horizon steps. We assume the risk-preference parameter of an investor is $\phi = 1$, and adopting shrinkage parameter $\tau = 0.01$ when estimating the precision matrix to avoid issues with a degenerate co-variance matrix.

Our proposed methods are as follows:

• (*Ret-Utility-GAN*) Asset returns as raw data and with the realized utility of the portfolio as the decision-related quantity.

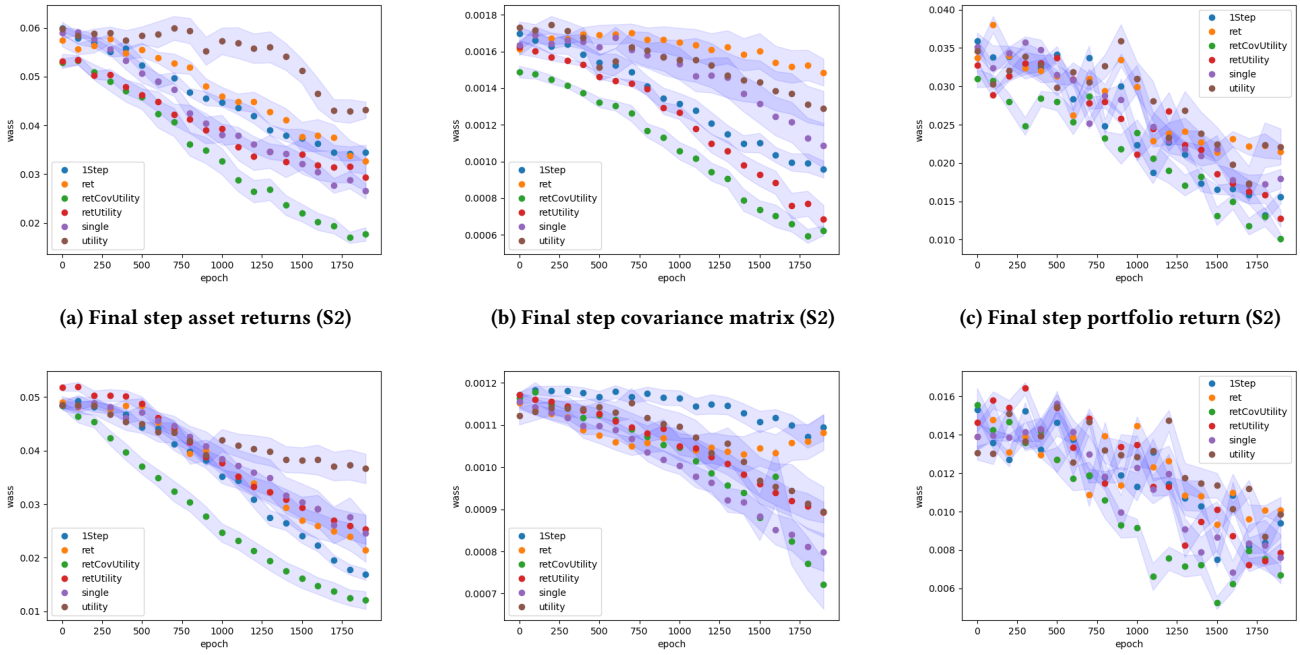
• (*Ret-Cov-Utility-GAN*) Asset returns as raw data, with estimated covariance matrix as intermediate quantity and with the realized utility of the portfolio as the decision-related quantity.

Utility is the decision-related quantity in Ret-Utility-GAN, since this comes at the end of the decision chain and controls all the decision-related quantities; in particular, the derivative of this quantity also involves the derivative of earlier quantities, via the chain rule, and thus controls multiple quantities of interest. Estimated covariance matrix is the intermediate quantity in Ret-Cov-Utility-GAN, and is adopted as it provides additional supervision to control the errors of intermediate quantity, which further control the errors of all quantities.

We find in our experiments that *Ret-Utility-GAN* and *Ret-Cov-Utility-GAN* provide good fidelity across different risk preference parameters (See Figure 3 and Figure 7 in Sun et al. [24]) and across different data generating processes comparing with the baselines. (see the first row of Figure 3 and 4).

Baselines. We compare our methods with the following approaches:

• (*Ret-GAN*) Asset returns as raw data. This is a standard model [12, 30] with loss imposed only on the asset return. This GAN generates synthetic raw asset returns $R'_{t+1:t+K}$ for each t, and the training process uses the sum of K Wasserstein losses, one for each look-ahead step.



(a) Final step asset returns (S2) (b) Final step covariance matrix (S2) (c) Final step portfolio return (S2)

(d) Final step asset returns (S3) (e) Final step covariance matrix (S3) (f) Final step portfolio return (S3)

Figure 4: Wasserstein Distance between non-synthetic and synthetic data for artificial environments S2 (first row) and S3 (second row). 4a and 4d asset returns; 4b and 4e estimated covariance matrix; 4c and 4f portfolio return. An epoch is one full pass of the data. Confidence bands are computed over 5 runs.

• (*1 Step-GAN*) Asset returns as raw data. This is a 1-step version of the Ret-Utility GAN: a GAN with a 1-step look-ahead asset return and utility (this is similar to Li et al. [15], but with an additional decision-aware loss). This GAN generates synthetic data $R'_{t+1:t+1}$ (i.e. $r'_{t+1|t}$) and synthetic derived quantity $U'_{t+1|t}$, i.e., utility. The training process uses the sum of two Wasserstein losses, defined on $R'_{t+1:t+1}$ and $U'_{t+1|t}$.

• (*Single-GAN*) Asset returns as raw data. This is a single Wasserstein loss defined on a vector of $2K$ quantities coming from stacking the asset returns, $R'_{t+1:t+K}$, and utility quantities $U'_{t+k|t}$, for each k and t.

• (*Utility-GAN*) Asset returns as raw data. The utility is the only loss of this model. (i.e., no loss on the asset returns). This GAN generates the synthetic derived quantities, $U'_{t+k|t}$, for each k and t, i.e., the utility quantities. The training process uses the sum of K Wasserstein losses, one for each look-ahead step.

Generator Network Architecture. For the generator, we use a feed-forward neural network for each asset, where the outputs are predicted asset returns and used to compute decision-related quantities. The generator network architecture for a particular asset (asset 2) is shown in Figure 5. For period t and look-ahead step k, there is a network with $(5 + 32)$ inputs, with 5 hand-crafted conditioning variables and 32 random seeds. $x_{t,i,s}$ is an input node, with index i $(1 \leqslant i \leqslant 5)$ corresponding to the asset return in the last day and each of four rolling-average returns (see below), and s being the index for the asset. Inputs $z_{t,t+k,j}$ $(1 \leqslant j \leqslant 32)$ are the random seeds, which are shared for each asset given the same k. The network has a single hidden layer with 16 ReLU units, denoted $h_{t+k|t,l,s}$ units $(1 \leqslant l \leqslant 16)$. The output units $r'_{t+k|t,s}$ provide the synthetic asset return for asset s. We then concatenate $r'_{t+k|t,s}$ to form $r'_{t+k|t}$.

After obtaining $r'_{t+k|t}$, we compute quantities of interest, i.e., $\hat{u}'_{t+k|t} = f_{u,k}(R'_{t+1:t+k}, x_t)$, $\hat{\Sigma}'_{t+k|t} = f_{\Sigma,k}(R'_{t+1:t+k}, x_t)$, $\hat{H}'_{t+k|t}$, $w'_{t+k|t}$, $p'_{t+k+1|t}$, and $U'_{t+k+1|t}$ (see Section 5). We use the same generator architecture for each asset s, each period t, and each look-ahead step k.

Discriminator Network Architecture. There is one discriminator for each quantity of interest (e.g., raw data, or a decision-related quantity). Figure 6 illustrates the discriminator network for the utility quantity. As with other quantities of interest, this is a feed-forward neural network. In the case of utility, the input to the network for period t and look-ahead step k is either the synthetic utility $U'_{t+k|t}$ or the non-synthetic utility U_{t+k}, together with $x_{t,i}$, for $1 \leqslant i \leqslant 5$, where x represents the conditioning variables. There are 32 hidden nodes (ReLU activations), these shown as $m_{t+k|t,i}$ units (for $1 \leqslant i \leqslant 32$). There is a single output node, denoted $D_{U,t+k|t}$, and representing the discriminator value. We use the same discriminator architecture for each decision quantity of interest, each period t, and each look-ahead step k.

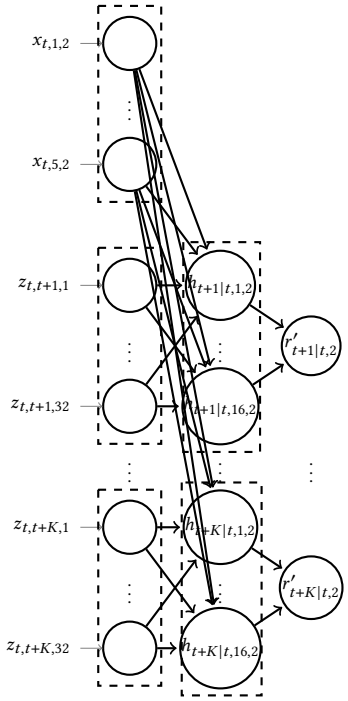


Figure 5: Generator Network Architecture for Asset 2, shown for period t and for look-ahead step $k = 1$ and $k = K$.

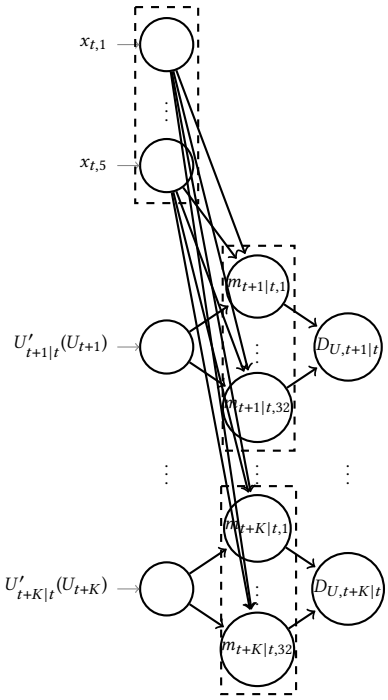


Figure 6: Discriminator Network Architecture for Utility, shown for period t and for look-ahead step $k = 1$ and $k = K$.

Features. For both the discriminator and the generator, we use five features as the conditional variables for each asset. These are the *asset return of the last week*, and four different moving-average-based features, computed by taking the average of asset returns in the past few weeks. The moving average operators are defined as $\mathrm{MA}_{\psi_i}(r_t) = \psi_i \mathrm{MA}_{\psi_i}(r_{t-1}) + (1-\psi_i) r_t$, where r_t is the raw asset return and the value of the *smoothing parameter* ψ_i ($0 < \psi_i < 1$) controls the smoothing for a particular feature. We adopt this small and fixed set of features, rather than some more general approach such as recurrent neural nets to represent history because the sample size is small when working with real data and we want to avoid over-fitting.

Evaluation. For evaluation, we calculate the Wasserstein distances between the raw asset returns, the estimated covariance matrix, the estimated precision matrix, the portfolio weights (the decision variables), the portfolio return and their respective synthetic counterparts. We do this for quantities that correspond to each of the final lookahead steps (i.e., steps 3 for the estimated covariance matrix, precision matrix and portfolio weights, and steps 4 for the raw data and portfolio return). We have chosen to present evaluation metrics exclusively for the final step quantities, which represent the culmination of the chain and therefore pose a greater challenge for yielding positive results. By demonstrating favorable outcomes on these difficult-to-master quantities, we implicitly suggest the proficiency of our proposed method for the earlier, less complex stages of the chain as well.

Results: Artificial environment. We first present results on three distinct artificial time series. Here, the data-generating process is given by $r_{t+1} = b_0 \cdot r_t + \sum_{i=1}^{4} b_i \cdot \mathrm{MA}_{\zeta_i}(r_t) + \epsilon$, where r_t is the asset return vector, $\mathrm{MA}_{\zeta_i}(r_t)$ the moving average operator, $\zeta_i > 0$ the smoothing parameter, b_i the coefficient, and ϵ the noise.

We summarize the datasets as follows:

• (*S1*) We set $\zeta_1 = 0.55, \zeta_2 = 0.74, \zeta_3 = 0.86, \zeta_4 = 0.92$, and $b_0 = 0.3, b_1 = 0.1, b_2 = 0.2, b_3 = 0.1$, and $b_4 = 0.1$. We use a *multivariate t-distribution* to model the noise, with location parameter $\mu = [0, 0, 0, 0]^\intercal$, shape matrix $\Sigma =$ $[1, 0.6, 0, 0; 0.6, 1, 0.6, 0; 0, 0.6, 1, 0.6; 0, 0, 0.6, 1]$, and d.o.f., $\nu = 100$. ν simulates the tail level of asset returns [6].

• (*S2*) We keep all parameters of data generating process from S1 the same, and only let $b_0 = -0.3$.

• (*S3*) We keep all parameters of data generating process from S1 the same, and only let $\nu = 5$, which simulates a heavy tail distribution.

The first row of Figure 3 and both rows of Figure 4 show that the Ret-Utility-GAN and Ret-Cov-Utility-GAN have generally better performance in terms of minimizing the Wasserstein distance for each of asset returns, estimated covariance matrix, estimated precision matrix, portfolio weights and portfolio return. In the first row of Figure 3, Ret-Utility-GAN performs (1) better than Ret-GAN, confirming that introducing utility provides useful moderation on the distribution of synthetic raw data; and (2) better than the Utility-GAN, which shows that including the asset return loss also helps; (3) better than the Single-GAN, which shows that imposing loss for each quantity is more effective than a single loss on stacked quantities (also, Ret-Utility-GAN is more computationally efficient than Single-GAN, since Single-GAN Wasserstein loss is imposed

on higher dimensions of stacked quantities, which increases the discriminator capacity compared to the unstacked version); (4) comparably to the 1-Step GAN. We also observe that Ret-GAN doesn't perform as well as the Ret-Utility-GAN even on the raw asset returns.

Our hypothesis is that, by introducing additional loss, Ret-Utility-GAN provides more information for the gradient during the training process, leading to more effective training stability and better generalization on evaluation data. In addition, the Ret-Utility-GAN, 1-Step GAN, and Single-GAN have losses imposed on both asset return and utility quantities, and they outperform Ret-GAN and Utility-GAN, showing that having both raw quantities and derived quantities are important for good performance.

In each row of Figure 4, Ret-Cov-Utility-GAN performs the best among all methods by a large margin, and Ret-Utility GAN is also competitive for raw data and the covariance matrix metric. Ret-GAN does not have good performance for the convariance matrix, and Utility-GAN does not have good performance for raw data. Ret-GAN and Utility-GAN are the two worst methods for the portfolio return metric. Together, these results confirm that introducing derived and decision-related quantities helps for stabilizing the training process and achieving better test performance.

Results: Real ETF time series. We also give results for training DAT-CGAN on real ETF time series, where we use weekly price data for each of four U.S. ETFs[6] from 1999 to 2016. The data includes the end-of-day price for each ETF. The entire dataset has more than 3,500 data points (17 years $\times$ 52 weeks $\times$ 4 ETFs). We divide the data into a training set from 1999–2006, and a test set from 2007–2016. As a reference, Yoon et al. [29] make use of 4,000 data points for their experiment, thus our sample size is comparable. We generate future weekly returns for each of $K = 4$ future weeks (1 month).

Figures 3, second row, shows that the performance of Ret-Utility-GAN is much better than Ret-GAN in regard to the Wasserstein loss in regard to each of the asset returns, precision matrix, and portfolio weights. Ret-Utility-GAN performs as well as Single-GAN in regard to the Wasserstein loss on all metrics. Ret-Utility-GAN also performs better than the 1Step-GAN in terms of its training stability, which shows that the Ret-Utility-GAN is effective in addressing exposure bias. In this experiment, Utility-GAN, also performs well. We again observe that Ret-GAN does not perform as well as Ret-Utility-GAN, and even on the raw asset return data, which further emphasizes that the introduction of a decision related quantity, such as utility, provides moderation on the distribution of synthetic raw data.

7 CONCLUSION

In this paper, we have introduced DAT-CGAN, which is a novel, decision-aware time series conditional generative adversarial network for generating time-series data. The method incorporates decision-related quantities into a multi-loss structure, avoids exposure bias by aligning look-ahead steps during training and testing, and alleviates problems with data scarcity through an overlapped-block sampling scheme. We characterize the finite-sample generalization properties of DAT-CGANs for generating the raw data as well as decision-related quantities. In an application to financial portfolio selection, we demonstrated better generative quality for decision-related quantities, such as estimated precision matrix and portfolio weights, as well as raw data than other strong, GAN-based baselines. We have established this on real, ETF time series data as well as within three different artificial environments.

[6] the *Material (XLB)*, *Energy (XLE)*, *Financial (XLF)*, and *Industrial (XLI)* ETFs.

REFERENCES

[1] Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 214–223.

[2] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. 2017. Generalization and Equilibrium in Generative Adversarial Nets (GANs). In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 224–232.

[3] Susan Athey, Guido W Imbens, Jonas Metzger, and Evan Munro. 2021. Using wasserstein generative adversarial networks for the design of monte carlo simulations. *Journal of Econometrics* (2021).

[4] MA Bashar and R Nayak. 2020. TAnoGAN: Time series anomaly detection with generative adversarial networks. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*.

[5] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*. 1171–1179.

[6] Bollerslev. 1987. A conditional heteroskedastic time series model for speculative prices and rates of return. *Review of Economics and Statistics* 69 (1987), 542–547.

[7] R.C. Bradley. 2007. Introduction to Strong Mixing Conditions. In *Kendrick Press, Heber City (Utah)*.

[8] Peter Bühlmann. 2002. Bootstraps for Time Series. *Statist. Sci.* 17 (2002), 52–72.

[9] J.B. Copas. 1983. Regression, Prediction and Shrinkage. In *Journal of the Royal Statistical Society.*

[10] Adam N Elmachtoub and Paul Grigas. 2022. Smart "predict, then optimize". *Management Science* 68, 1 (2022), 9–26.

[11] A Geiger, D Liu, and S Alnegheimish. 2020. TadGAN: Time series anomaly detection using generative adversarial networks. In *2020 IEEE International Conference on Big Data*.

[12] Adriano Soares Koshiyama, Nick Firoozye, and Philip C. Treleaven. 2019. Generative Adversarial Networks for Financial Trading Strategies Fine-Tuning and Combination. *CoRR* abs/1901.01751 (2019). http://arxiv.org/abs/1901.01751

[13] D Li, D Chen, and J Goh. 2018. Anomaly detection with generative adversarial networks for multivariate time series. In *arxiv*.

[14] D Li, D Chen, and B Jin. 2019. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *Advances in Neural Information Processing Systems 31*.

[15] Junyi Li, Xintong Wang, and Yaoyang Lin. 2020. Generating Realistic Stock Market Order Streams. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. 727–734.

[16] Y Luo, X Cai, and Y Zhang. 2018. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems 31*.

[17] Y Luo, Y Zhang, and X Cai. 2019. E2gan: End-to-end generative adversarial network for multivariate time series imputation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*.

[18] Jayanta Mandi, Vıctor Bucarey, Maxime Mulamba Ke Tchomba, and Tias Guns. 2022. Decision-focused learning: through the lens of learning to rank. In *International Conference on Machine Learning*. PMLR, 14935–14947.

[19] Harry Markowitz. 1952. Portfolio Selection. In *The Journal of Finance*.

[20] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *CoRR* abs/1411.1784 (2014). http://arxiv.org/abs/1411.1784

[21] Hao Ni, Lukasz Szpruch, and Magnus Wiese. 2020. Conditional Sig-Wasserstein GANs for Time Series Generation. In *arxiv*.

[22] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *4th International Conference on Learning Representations*.

[23] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-Critical Sequence Training for Image Captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 1179–1195.

[24] He Sun, Zhun Deng, Hui Chen, and David Parkes. 2023. Technical Materials for Decision-Aware Conditional GANs for Time Series Data. (2023). https://drive.google.com/file/d/1fcNsQULSuO8NZPHii2GoQn8RlSEPrcON/view?usp=sharing

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[26] S Takahashi, Y Chen, and K Tanaka-Ishii. 2019. Modeling financial time-series with generative adversarial networks. In *Physica A: Statistical Mechanics and its Applications*.

[27] Cédric Villani. 2009. Optimal Transport: Old and New. In *Springer, Berlin*.

[28] Magnus Wiese, Robert Knobloch, and Ralf Korn1. 2019. Quant GANs: Deep Generation of Financial Time Series. In *arxiv*.

[29] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. 2019. Time-series Generative Adversarial Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*. 5509–5519.

[30] Xingyu Zhou, Zhisong Pan, Guyu Hu, Siqi Tang, and Cheng Zhao. 2018. Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets. *Mathematical Problems in Engineering* (2018).