Time-series Generation by Contrastive Imitation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Consider learning a generative model for time-series data. The sequential setting poses a unique challenge: Not only should the generator capture the *conditional* dynamics of (stepwise) transitions, but its open-loop rollouts should also preserve the *joint* distribution of (multi-step) trajectories. On one hand, autoregressive models trained by MLE allow learning and computing explicit transition distributions, but suffer from compounding error during rollouts. On the other hand, adversarial models based on GAN training alleviate such exposure bias, but transitions are implicit and hard to assess. In this work, we propose a novel framework marrying the best of both worlds: Motivated by a precise moment-matching objective to mitigate compounding error, we optimize a local (but forward-looking) *transition policy*, where the reinforcement signal is provided by a global (but stepwise-decomposable) *energy model* trained by contrastive estimation. In learning, the two components are trained cooperatively, avoiding the instabilities typical of adversarial objectives. Moreover, while the learned policy serves as the generator for sampling, the learned energy naturally serves as a trajectory-level measure for evaluating sample quality. By expressly training a policy to imitate sequential behavior of time-series features in a dataset, our approach embodies "*generation by imitation*". Theoretically, we demonstrate the correctness of our formulation and consistency of our algorithm. Empirically, we evaluate its ability to generate realistic samples using real-world datasets, and verify that it performs at or above the standard of existing benchmarks.

1 Introduction

Time-series data are ubiquitous in diverse machine learning applications, such as financial, industrial, and healthcare settings. At the same time, lack of public access to data is a recurring obstacle to the development and reproducibility of research in domains where datasets are proprietary [1]. Generating synthetic—but realistic—time-series data is a promising solution [2], and has received increasing attention in recent years, driven by advances in deep learning and generative adversarial networks [3,4].

Owing to the fact that time-series features are generated sequentially, generative modeling in the temporal setting faces a two-pronged challenge: First, a good generator should accurately capture the conditional dynamics of *stepwise* transitions $p(x_t|x_1, ..., x_{t-1})$; this is important, as the faithfulness of any conceivable downstream time-series analysis depends on the learned correlations across both temporal and feature dimensions. Second, however, the recursive rollouts of the generator should also respect the joint distribution of *multi-step* trajectories $p(x_1, ..., x_T)$; this is equally important, as synthetic trajectories that inadvertently wander beyond the support of original data are useless at best.

Recent work falls into two main categories. On one hand, *autoregressive models* trained via MLE [5] explicitly factor the distribution of trajectories into a product of conditionals $\prod_t p(x_t|x_1, ..., x_{t-1})$. While this allows directly learning and computing such transitions, with finite data this is prone to *compounding errors* during multi-step generation, due to the discrepancy between closed-loop training (i.e. conditioned on ground-truths as inputs) and open-loop sampling (i.e. conditioned on its own previous

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

outputs) [6]. A variety of methods have sought to counteract this problem of exposure bias, employing auxiliary techniques from curriculum learning [7,8] and adversarial domain adaptation [9,10]; however, such remedies are not without biases [11], and empirical improvements have been mixed [12–14].

On the other hand, *adversarial models* based on GAN training and its relatives [15–17] directly model the distribution of trajectories $p(x_1, ..., x_T)$ [18–20]. To provide a more granular learning signal for the generator, a popular variant matches the induced distribution of sub-trajectories instead, providing stepwise feedback from the discriminator [21,22]. TimeGAN [12] is the most recent incarnation of this, and operates within a jointly optimized latent space. GAN-based approaches alleviate the risk of compounding errors, and have been applied to banking [23], sensors [24], biosignals [25], and smartgrids [26]. However, the conditional dynamics are only *implicitly learned*, yielding no way of inspecting or assessing the quality of sampled transitions nor trajectories. Moreover, the adversarial objective leads to characteristically challenging optimization—exacerbated by the temporal dimension.

Three Desiderata In light of the preceding, we argue that a good probabilistic generative model for time-series data should fulfill the following criteria simultaneously—which no existing technique does. (As a preview of our subsequent developments, Table 1 gives a comparison of related methods):

- **Sampling**: First, it should respect both the stepwise *conditional* distributions of features in the data, as well as the *joint* distribution of full trajectories—minimizing multi-step compounding error.
- **Evaluating**: Second, it should provide generic measures of *sample quality* for both transitions and trajectories—commonly desired for sample comparison, model auditing, or bias correction [27,28].
- **Learning**: Third, unlike black-box GAN discriminators, the evaluator should be *decoupled* from any specific sampler, and the two components should be trainable *non-adversarially* for stability.

Contributions We propose a novel approach satisfying all criteria at once. In the sequel, we first give precise treatment of the "compounding error" problem, thus motivating a specific trajectory-centric optimization objective from first principles (Section 2). To carry it out, we develop a general training framework and practical algorithm, along with its theoretical justification: By training a forward-looking *transition policy* to imitate the sequential behavior of time series using a stepwise-decomposable *energy model* as reinforcement, our approach embodies "*generation by imitation*" (Section 3). To understand its relative simplicity and effectiveness, we compare our method to existing generative models for time-series data, and relate it to imitation learning of sequential behavior (Section 4). Lastly, through experiments with application to real-world time-series datasets with a variety of characteristics, we verify that it performs at or above the standard of comparable benchmarks (Section 5).

2 Synthetic Time Series

2.1 Problem Setup

We operate in the standard discrete-time setting for time series. Let feature vectors $x_t \in \mathcal{X}$ be indexed by time steps t, and let a full trajectory of length T be denoted $\tau \coloneqq (x_1, ..., x_T) \in \mathcal{T} \coloneqq \mathcal{X}^T$. Also, denote with $h_t \coloneqq (x_1, ..., x_{t-1}) \in \mathcal{H} \coloneqq \cup_{t=1}^{T} \mathcal{X}^t$ the history prior to time t. For ease of exposition we shall work with trajectories of fixed lengths T, but our results trivially generalize to the case where T itself is a random variable (for instance, by employing padding tokens up to some maximum length).

Consider a dataset $\mathcal{D} \coloneqq \{\tau_n\}_{n=1}^{N}$ of N trajectories sampled from some true source s. We assume the trajectories are generated sequentially by some unknown transition process $\pi_s \in \Delta(\mathcal{X})^{\mathcal{H}}$, such that features at each step t are sampled as $x_t \sim \pi_s(\cdot|h_t)$. In addition to this stepwise conditional, denote with $\mu_s(h) \coloneqq \frac{1}{T}\sum_t p(h_t = h|\pi_s)$ the normalized occupancy measure—i.e. the distribution of histories induced by π_s. Intuitively, this is the visitation distribution of "history states" encountered by a generator when navigating about the feature space $\mathcal{X}$ by rolling out policy π_s. With slight abuse of notation, we may also write $\mu_s(h, x) \coloneqq \mu_s(h)\pi_s(x|h)$ to indicate the marginal distribution of transitions. Finally, let the joint distribution of full trajectories be denoted by $p_s(\tau) \coloneqq \prod_t \pi_s(x_t|h_t)$.

The goal is to learn a sequential generator π_θ parameterized as θ using samples $\tau \sim p_s$ from $\mathcal{D}$, such that $p_\theta \approx p_s$. Note here that we do not assume stationarity of the time-series data, nor stationarity of the transition conditionals; any influence of t is implicit through the dependence of π_s (and π_θ) on variable-length histories. In line with recent work [14,20], for simplicity we do not consider static metadata as supplemental inputs or outputs, as these are commonly and easily incorporated via an additional conditioning layer or auxiliary generator [12,19]. Lastly, note that much recent

work on sequential modeling is devoted to domain-specific, *architecture*-level designs for generating audio [29,30], text [31,32], and video [33,34]. In contrast, our work is closer in spirit to [12,14] in our agnostic, *framework*-level contribution suitable for generic tabular data in any time-series setting.

Measuring Sample Quality How do we determine the "quality" of a sample? In specialized domains, of course, we often have prior access to *task-specific* metrics such as BLEU or ROUGE scores in text generation [6,35]—then, the generator can simply be optimized for such scores via standard methods in reinforcement learning [36]. In generic time-series settings, however, the challenge is that any such metric must necessarily be *task-agnostic*, and access to it must necessarily come from learning.

So, for any data source s, let us speak of some hypothetical function $f_s : \mathcal{H} \times \mathcal{X} \to [-k, k]$ with $k < \infty$, such that $f_s(h, x)$ gives the quality of any sampled *transition*—that is, any tuple (h, x). Intuitively, we may interpret this as quantifying how "typical" it is for the random process to be in state h and step towards x. Likewise, let as also speak of some function $F_s : \mathcal{T} \to [-kT, kT]$ such that $F_s(\tau)$ gives the quality of any sampled *trajectory*. Naturally, in time-series settings where the underlying process is causally-conditioned, it is reasonable to define this as the decomposition $F_s(\tau) \coloneqq \sum_t f_s(h_t, x_t)$. Now of course, we have no access to the true F_s. But clearly, in learning a generative model p_θ of p_s, we wish that the quality of samples τ drawn from p_θ and p_s be similar in expectation. More precisely:

Definition 1 (Expected Quality Difference) Let $\Delta\bar{F}_s : \Theta \to [-2kT, 2kT]$ denote the *expected quality difference* between p_s and p_θ, where Θ indicates the space of parameterizations for generator π_θ:

$$\Delta\bar{F}_s(\theta) \coloneqq \mathbb{E}_{\tau \sim p_s} F_s(\tau) - \mathbb{E}_{\tau \sim p_\theta} F_s(\tau) \tag{1}$$

Our objective, then, is to learn a generator π_θ that minimizes the expected quality difference $\Delta\bar{F}_s(\theta)$. Two points bear emphasis. First, we know nothing about F_s—beyond it being the sequential aggregate of f_s. This challenge uniquely differentiates our agnostic setting from more popular media-specific applications—for which various predefined measures are readily available for supervision. Second, in addition to matching this *expectation* over samples, we also wish to match the *variety* of samples in the original data. After all, we want p_θ to mimic samples from p_s of different degrees of "typicality". So we should expect to incorporate some measure of entropy, e.g. the commonly used Shannon entropy.

2.2 Matching Local Moments

Recall the apparent tradeoff between autoregressive models and adversarial models. In the spirit of the former, suppose we seek to directly learn *transition conditionals* via supervised learning. That is,

$$\arg\min_\theta \mathbb{E}_{h \sim \mu_s} \mathcal{L}(\pi_s(\cdot|h), \pi_\theta(\cdot|h)) \tag{2}$$

Consider the log likelihood loss $\mathcal{L}(\pi_s(\cdot|h), \pi_\theta(\cdot|h)) \coloneqq \mathbb{E}_{x \sim \pi_s(\cdot|h)} \log \pi_\theta(x|h)$. In the case of exponential family models for $\pi_\theta(\cdot|h)$, a basic result is that this is dual to maximizing its conditional entropy subject to the constraint on feature expectations $\mathbb{E}_{h \sim \mu_s; x \sim \pi_\theta(\cdot|h)} T(x) = \mathbb{E}_{h \sim \mu_s; x \sim \pi_s(\cdot|h)} T(x)$, where $T : \mathcal{X} \to \mathbb{R}$ is some sufficient statistic [37–39]. More generally for deep energy-based models,

$$\arg\min_\theta \left(\mathbb{E}_{\substack{h \sim \mu_s \\ x \sim \pi_\theta(\cdot|h)}} \log \pi_\theta(x|h) + \max_{f \in \mathbb{R}^{\mathcal{H} \times \mathcal{X}}} \left(\mathbb{E}_{\substack{h \sim \mu_s \\ x \sim \pi_s(\cdot|h)}} f(h, x) - \mathbb{E}_{\substack{h \sim \mu_s \\ x \sim \pi_\theta(\cdot|h)}} f(h, x) \right) \right) \tag{3}$$

Note that the moment-matching constraint is *local*—that is, at the level of individual transitions, and all conditioning is based on h from μ_s alone. This is precisely the "exposure bias": The objective is only ever exposed to inputs h drawn from the (perfect) source distribution μ_s, and is thus unaware of the endogeneity of the (imperfect) synthetic distribution μ_θ induced by π_θ. This is not desirable since π_θ is rolled out by open-loop sampling at test time. Specifically, suppose we can bound how well we can enforce the moment-matching constraint; as it turns out, we cannot eliminate error compounding:

Lemma 1 Let $\max_{f \in \mathbb{R}^{\mathcal{H} \times \mathcal{X}}} \left(\mathbb{E}_{\substack{h \sim \mu_s \\ x \sim \pi_s(\cdot|h)}} f(h, x) - \mathbb{E}_{\substack{h \sim \mu_s \\ x \sim \pi_\theta(\cdot|h)}} f(h, x) \right) \leq \epsilon$. Then $\Delta\bar{F}_s(\theta) \in O(T^2 \epsilon)$.

Proof. Appendix A. □

This reveals the problem with modeling conditionals per se: *Not all mistakes are equal*. An objective like Equation 2 penalizes unrealistic transitions (h, x) by treating all conditioning histories h equally—regardless of how realistic h is to begin with. Clearly, however, we care much less about how x looks like, if the current subsequence h is already highly unlikely (and vice versa). Intuitively, earlier mistakes in a trajectory should weigh more: Once π_θ wanders into areas of $\mathcal{H}$ with low support in μ_s, no amount of "good" transitions will bring the trajectory back to high-likelihood areas of $\mathcal{T}$ under p_s.

2.3 Matching Global Moments

Now suppose instead that we seek to directly constrain the *trajectory distribution* p_θ to be similar to p_s:

$$\arg\min_\theta \mathcal{L}(p_s, p_\theta) \tag{4}$$

Consider the Kullback-Leibler divergence $\mathcal{L}(p_s, p_\theta) \coloneqq D_{\mathrm{KL}}(p_s \| p_\theta)$. Like before, we know that in the case of exponential family models for p_θ, this is dual to maximizing its entropy subject to the constraint $\mathbb{E}_{\tau\sim p_s} T(\tau) = \mathbb{E}_{\tau\sim p_\theta} T(\tau)$, where $T : \mathcal{T} \to \mathbb{R}$ is some sufficient statistic [40]. More broadly for deep energy-based models, we have $\arg\min_\theta \left(\mathbb{E}_{\tau\sim p_\theta} \log p_\theta(\tau) + \max_{F\in\mathbb{R}^\mathcal{T}} \left(\mathbb{E}_{\tau\sim p_s} F(\tau) - \mathbb{E}_{\tau\sim p_\theta} F(\tau)\right)\right)$. Now, observe that by definition of occupancy measure μ, for any function $f : \mathcal{H} \times \mathcal{X} \to \mathbb{R}$ it must be the case that $\mathbb{E}_{\tau\sim p} \sum_t f(h_t, x_t) = T\mathbb{E}_{h\sim\mu, x\sim\pi(\cdot|h)} f(h, x)$. Therefore we may equivalently write

$$\arg\min_\theta \left(\mathbb{E}_{\substack{h\sim\mu_\theta \\ x\sim\pi_\theta(\cdot|h)}} \log \pi_\theta(x|h) + \max_{f\in\mathbb{R}^{\mathcal{H}\times\mathcal{X}}} \left(\mathbb{E}_{\substack{h\sim\mu_s \\ x\sim\pi_s(\cdot|h)}} f(h, x) - \mathbb{E}_{\substack{h\sim\mu_\theta \\ x\sim\pi_\theta(\cdot|h)}} f(h, x)\right)\right) \tag{5}$$

Importantly, note that the moment-matching constraint is now *global*—that is, at the level of trajectory rollouts, and π_θ is now conditioned on histories h drawn from its own induced occupancy μ_θ. There is no "exposure bias": In order to respect the constraint, not only does $\pi_\theta(\cdot|h)$ have to be close to $\pi_s(\cdot|h)$ for any given h, but the occupancy μ_θ induced by π_θ also has to be close to the occupancy μ_s induced by π_s. As it turns out, this seemingly minor difference is sufficient to mitigate compounding errors. Specifically, suppose we can bound how well we can enforce the moment-matching constraint:

Lemma 2 Let $\max_{f\in\mathbb{R}^{\mathcal{H}\times\mathcal{X}}} \left(\mathbb{E}_{\substack{h\sim\mu_s \\ x\sim\pi_s(\cdot|h)}} f(h, x) - \mathbb{E}_{\substack{h\sim\mu_\theta \\ x\sim\pi_\theta(\cdot|h)}} f(h, x)\right) \leq \epsilon$. Then $\Delta\bar{F}_s(\theta) \in O(T\epsilon)$.

Proof. Appendix A. □

This illustrates why even *transition-centric* adversarial models such as [12, 21] have shown promise in generating realistic trajectories [23–26]. First, unlike *trajectory-centric* GANs [18, 19] which directly attempt to minimize some form of Equation 4, in transition-centric GANs the objective is to match the transition marginals $\mu_\theta(h, x)$ and $\mu_s(h, x)$—so the discriminator provides more granular feedback to the generator for training. At the same time, we see from Lemma 2 that matching transition marginals is already—indirectly—performing the sort of moment-matching that alleviates compounding error.

Can we be more direct? In Section 3, we shall start by tackling Equation 5 itself. As it turns out, this endeavor yields a novel technique that trains an explicit conditional policy (for sampling), an energy model (for evaluation), and a non-adversarial framework (for learning)—satisfying all our desiderata.

3 Generating by Imitating

First, consider the most straightforward implementation: Let us parameterize $f \in \mathbb{R}^{\mathcal{H}\times\mathcal{X}}$ as ϕ, and swap the min-max order using the minimax theorem, which yields the following adversarial objective:

$$\mathcal{L}(\theta, \phi) \coloneqq \max_\phi \min_\theta \left(\mathbb{E}_{\substack{h\sim\mu_\theta \\ x\sim\pi_\theta(\cdot|h)}} \log \pi_\theta(x|h) + \mathbb{E}_{\substack{h\sim\mu_s \\ x\sim\pi_s(\cdot|h)}} f_\phi(h, x) - \mathbb{E}_{\substack{h\sim\mu_\theta \\ x\sim\pi_\theta(\cdot|h)}} f_\phi(h, x)\right) \tag{6}$$

It is easy to see that this effectively describes variational training of the energy-based model $p_\phi(\tau) \coloneqq \exp(F_\phi(\tau) - \log Z_\phi)$—where $F_\phi(\tau) \coloneqq \sum_t f_\phi(h_t, x_t)$—to approximate the true $p_s(\tau)$, using samples from the variational p_θ. The (outer) energy player is the maximizing agent, and the (inner) policy player is the minimizing agent. The form of this objective naturally prescribes a bilevel optimization procedure in which we perform gradient-based updates for ϕ with (nested) best-response updates for θ.

3.1 Challenges of Learning

Abstractly, of course, training energy models using variational samplers is not new: Multiple works in static domains—such as image modeling—have investigated this approach as a means of bypassing the expense and variance of MCMC sampling [41, 42]. In our setting, however, there is the additional *temporal* dimension: The negative energy $F_\phi(\tau)$ of any trajectory is computed as the sequential composition of stepwise qualities $f_\phi(h_t, x_t)$, and each trajectory sampled from p_θ must be generated as the sequential rollout of stepwise policies $\pi_\theta(x_t|h_t)$. Consider the gradient update for the energy,

$$\nabla_\phi \mathcal{L} = \mathbb{E}_{\substack{h\sim\mu_s \\ x\sim\pi_s(\cdot|h)}} \nabla_\phi f_\phi(h, x) - \mathbb{E}_{\substack{h\sim\mu_\theta \\ x\sim\pi_\theta(\cdot|h)}} \nabla_\phi f_\phi(h, x) \tag{7}$$

and the inner-loop update for the policy,

$$\arg\min_\theta \mathbb{E}_{\substack{h\sim\mu_\theta \\ x\sim\pi_\theta(\cdot|h)}} \log \pi_\theta(x|h) - \mathbb{E}_{\substack{h\sim\mu_\theta \\ x\sim\pi_\theta(\cdot|h)}} f_\phi(h,x) \tag{8}$$

Note that the max-min optimization requires complete optimization within each inner update in order for the outer update to be correct. Otherwise the gradients will be *biased*, and there would be no guarantee the procedure converges to anything meaningful. Yet unlike in the static setting—for which there exists variety of standard approximations for the inner update [41–44]—here the policy update amounts to entropy-regularized *reinforcement learning* [45–47] using $f_\phi(h_t, x_t)$ as reward function. Thus our first difficulty is computational: Repeatedly performing inner-loop RL is simply infeasible.

Now, an obvious alternative is to dispense with complete policy optimization at each step, and instead to employ *importance sampling* to ensure that the gradients for the energy updates are still unbiased:

$$\nabla_\phi \mathcal{L} = \mathbb{E}_{\tau\sim p_s} \nabla_\phi F_\phi(\tau) - \frac{1}{Z_\phi} \mathbb{E}_{\tau\sim p_\theta} \left[\frac{\exp(\sum_t f_\phi(h_t, x_t))}{\prod_t \pi_\theta(x_t|h_t)} \nabla_\phi F_\phi(\tau) \right] \tag{9}$$

where the partition function is computed as $Z_\phi = \mathbb{E}_{\tau\sim p_\theta}[\exp(\sum_t f_\phi(h_t, x_t))/\sum_t \pi_\theta(x_t|h_t)]$, and the sampling policy π_θ is no longer required to be perfectly optimized with respect to f_ϕ. Unfortunately, this strategy simply replaces the original difficulty with a statistical one: As soon as we consider time-series data of non-trivial lengths T, the multiplicative effect of each time step on the importance weights means the gradient estimates—albeit unbiased—will have impractically high variance [48,49].

3.2 Contrastive Imitation

We now describe our proposed framework that sidesteps these difficulties. The key idea is that instead of Equation 7, we shall learn p_ϕ by contrasting (real) "positive" samples $\tau \sim p_s$ and (any) "negative" samples $\tau \sim p_\theta$, which—as we shall see—rids us of the requirement that π_θ be fully optimized at each step for learning to be guaranteed. First, let us establish the notion of a "structured classifier":

Definition 2 (Structured Classification) Recall that the policy-induced distribution p_θ evaluated at any τ is given by $p_\theta(\tau) \coloneqq \prod_t \pi_\theta(x_t|h_t)$. Also denote with $\tilde{p}_\phi$ the *un-normalized* energy-based model such that $\tilde{p}_\phi(\tau) \coloneqq \exp(\sum_t f_\phi(h_t, x_t))$. Define the associated *structured classifier* $d_{\theta,\phi} : \mathcal{T} \to [0, 1]$:

$$d_{\theta,\phi}(\tau) \coloneqq \frac{\tilde{p}_\phi(\tau)}{\tilde{p}_\phi(\tau) + p_\theta(\tau)} \tag{10}$$

That is, unlike a black-box classifier that may be arbitrarily parameterized—such as a generic discriminator d in a GAN—here $d_{\theta,\phi}$ is "structured" in that it is modularly parameterized by the embedded energy and policy functions. Now, we shall train ϕ such that $d_{\theta,\phi}$ discriminates well between $\tau \sim p_s$ and $\tau \sim p_\theta$—that is, so that the output $d_{\theta,\phi}(\tau)$ represents the (posterior) probability that τ is real,

$$\mathcal{L}_{\text{energy}}(\phi;\theta) \coloneqq -\mathbb{E}_{\tau\sim p_s} \log d_{\theta,\phi}(\tau) - \mathbb{E}_{\tau\sim p_\theta} \log \big(1 - d_{\theta,\phi}(\tau)\big) \tag{11}$$

and as before,

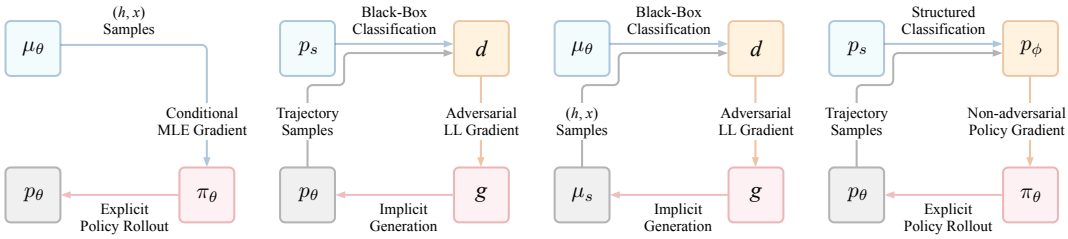
$$\mathcal{L}_{\text{policy}}(\theta;\phi) \coloneqq \mathbb{E}_{\substack{h\sim\mu_\theta \\ x\sim\pi_\theta(\cdot|h)}} \log \pi_\theta(x|h) - \mathbb{E}_{\substack{h\sim\mu_\theta \\ x\sim\pi_\theta(\cdot|h)}} f_\phi(h,x) \tag{12}$$

Why is this better? As we now show formally, each gradient update no longer requires θ to be optimal for the current value of ϕ—nor does it require importance sampling—unlike the procedure described by Equation 6. The only requirement is that p_θ can be sampled and evaluated efficiently, e.g. using learned Gaussian policies as usual, or—should more flexibility be required—with normalizing flow-based policies. As a practical result, this means policy updates can be *interleaved* with energy updates, instead of being *nested* within a repeated inner loop. Specifically, let us establish the following results:

Proposition 3 (Global Optimality) Let $f_\phi \in \mathbb{R}^{\mathcal{H}\times\mathcal{X}}$, and let $p_\theta \in \Delta(\mathcal{T})$ be any distribution satisfying positivity: $p_\theta(\tau) > 0 \Leftrightarrow p_s(\tau) > 0$ (note that this does not require π_θ to be optimal for f_ϕ). Then $\mathcal{L}_{\text{energy}}(\phi;\theta)$ is globally minimized at $F_\phi = \log p_s$, whence $\tilde{p}_\phi$ is self-normalized with unit integral.

Proof. Appendix A. □

This result is intuitive by analogy with noise-contrastive estimation [50,51]: ϕ is learnable as long as negative samples $\tau \sim p_\theta$ cover the support of the true p_s. The positivity condition is mild (e.g. take Gaussian policies π_θ), and so is the realizability condition (e.g. take neural-networks for f_ϕ). Importantly, note that at optimality classifier $d_{\theta,\phi}$ is *decoupled* from any specific value of θ; contrast this with generic discriminators d in GANs, which are only ever optimal for the current generator. Now, in practice we must approximate p_s and p_θ using *finite* samples. In light of this, two questions are immediate: First, does the learned ϕ converge to the global optimum as the sample size increases? Second, what role does the "quality" of the policy's samples play in how ϕ is learned? For the former:



(a) T-Forcing [5] (b) C-RNN-GAN [18] (c) TimeGAN [12] (d) **TimeGCI (Ours)**

Figure 1: *Comparison of Time-series Generative Models*. Examples of (a) conditional MLE-based autoregressive model, (b) trajectory-centric GAN, and (c) transition-centric GAN. (d) Our proposed technique. See also Table 1.

Proposition 4 (Asymptotic Consistency) Let ϕ^* denote the maximizer for $\mathcal{L}_{\text{energy}}(\phi; \theta)$, and let $\hat{\phi}^*_M$ denote the maximizer for its finite-data approximation—that is, where the expectations over p_s and p_θ are approximated by M samples. Then under some mild conditions, as M increases $\hat{\phi}^*_M \xrightarrow{p} \phi^*$.

Proof. Appendix A. □

Now for the second question: Clearly if p_θ were too far from p_s, learning would be slow—the job would be too easy for the classifier $d_{\theta,\phi}$, and it may be able to distinguish samples via basic statistics alone. Indeed, in standard noise-contrastive estimation with a *fixed* noise distribution, learning is ineffective in the presence of many variables [52]. Precisely, however, that is why we continuously update the policy itself as an *adaptive* noise distribution: As p_ϕ moves closer to p_s, so does p_θ—thus providing more "challenging" negative samples. (It is easy to see that minimizing Equation 12 equivalently minimizes the reverse Kullback-Leibler divergence between p_ϕ and p_θ). In fact, should we insist on greedily taking each policy update to optimality, we recover the original max-min gradient from before:

Proposition 5 (Gradient Equality) Let ϕ_k be the value taken by ϕ after the k-th gradient update, and let θ^*_k be the corresponding maximizer for $\mathcal{L}_{\text{policy}}(\theta; \phi_k)$. Then $\nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta^*_k) = \frac{1}{2} \nabla_\phi \mathcal{L}(\theta^*_k, \phi)$. That is, at θ^*_k the energy gradient (of Equation 11) recovers the original gradient (from Equation 7).

Proof. Appendix A. □

In sum, we have arrived at a framework that learns an explicit sampling policy without exposure bias, a decoupled energy model without nested or saddle-point optimization, and is self-normalizing without importance sampling or estimating the partition function. Figure 1 gives a representative comparison.

3.3 Optimization Algorithm

Algorithm 1 Time-series Generation by Contrastive Imitation ▷ Details in Appendix B

Input: source dataset $\mathcal{D} \approx p_s$, mini-batch size M, learning rates λ
Initialize: replay buffer $\mathcal{B}$, energy parameter ϕ, policy parameter θ, critic parameter ψ
for each iteration **do**
  for each policy rollout **do**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\tau \sim p_\theta\}$ ▷ Generate sample
  for each gradient step **do**
    $\phi \leftarrow \phi - \lambda_{\text{energy}} \nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta)$ ▷ Update energy
    $\theta \leftarrow \theta - \lambda_{\text{actor}} \nabla_\theta \mathcal{L}_{\text{actor}}(\theta; \phi, \psi)$ ▷ Update policy
    $\psi \leftarrow \psi - \lambda_{\text{critic}} \nabla_\psi \mathcal{L}_{\text{critic}}(\psi; \phi)$ ▷ Update critic
Output: learned policy parameter θ^* and energy parameter ϕ^*

The only remaining choice is the method of policy optimization. Here we employ *soft actor-critic* [53], although in principle any technique will do—the only requirement is that it performs reinforcement learning with *entropy-regularization* [45–47]. To optimize the policy per Equation 12, in addition to the policy "actor" itself, this trains a "critic" to estimate value functions. As usual, the actor takes soft policy improvement steps, minimizing $\mathcal{L}_{\text{actor}}(\theta; \phi, \psi) := \mathbb{E}_{h \sim \mathcal{B}} \mathbb{E}_{x \sim \pi_\theta(\cdot|h)}[\log \pi_\theta(x|h) - Q_\psi(h, x)]$, where $Q_\psi : \mathcal{H} \times \mathcal{X} \to [-kT, kT]$ is the transition-wise value function parameterized by ψ, and $\mathcal{B}$ is a replay buffer of samples generated by π_θ. The critic is trained to minimize the soft Bellman residual: $\mathcal{L}_{\text{critic}}(\psi; \phi) := \mathbb{E}_{h,x \sim \mathcal{B}}(Q_\psi(h, x) - f_\phi(h, x) - V_\psi(h'))^2$, where the state-values are bootstrapped as $V_\psi(h') := \mathbb{E}_{x' \sim \pi_\theta(\cdot|h')}[Q_\psi(h', x') - \log \pi_\theta(x'|h')]$. By expressly training an *imitation* policy to mimic time-series behavior using rewards from an energy model trained by *contrastive* learning, we call this framework Time-series Generation by Contrastive Imitation (TimeGCI): See Algorithm 1.

Table 1: *Comparison of Time-series Generative Models*. Examples of conditional MLE-based autoregressive models, trajectory-centric GANs, transition-centric GANs, as well as our proposed technique. See also Figure 1.

Type	Examples	Optimization Objective(s)	Generator Signal	Discrim. Signal	No Exposure Bias	Decoupled Discrim.	Non-Adversarial	Explicit Policy	Explicit Energy
Condit. MLE	T-Forcing [5]	Data LL	Stepwise	(N/A)	✗	(N/A)	✓	✓	✗
	Z-Forcing [13]	Data LL (ELBO)	Stepwise	(N/A)	✗	(N/A)	✓	✓	✗
	P-Forcing [10]	Data LL + Class. LL (p_θ v. $\tilde{p}_\theta$)	Stepwise	Global	✗	✗	✗	✓	✗
Traject. GAN	C-RNN-GAN [18]	Classification LL (p_θ v. p_s)	Global	Global	✓	✗	✗	✗	✗
	DoppelGANger [19]	Classification LL (p_θ v. p_s)	Global	Global	✓	✗	✗	✗	✗
	COT-GAN [20]	Sinkhorn Divergence (p_θ v. p_s)	Global	Global	✓	✗	✗	✗	✗
Transit. GAN	RC-GAN [21]	Classification LL (μ_θ v. μ_s)	Stepwise	Stepwise	✓	✗	✗	✗	✗
	T-CGAN [22]	Classification LL (μ_θ v. μ_s)	Stepwise	Stepwise	✓	✗	✗	✗	✗
	TimeGAN [12]	Class. LL (μ_θ v. μ_s) + Data LL	Stepwise	Stepwise	✓	✗	✗	✗	✗
TimeGCI (Ours)		Discrim.: Class. LL (p_θ v. p_s) Generator: Policy Optimization	Stepwise	Global	✓	✓	✓	✓	✓

4 Discussion

Our theoretical motivations are apparent (Sections 2.2–3.1), and the mechanics of optimization are straightforward (Section 3.2–3.3). To understand the relative simplicity and effectiveness of TimeGCI, two questions remain: First, how does this relate to bread-and-butter imitation learning of sequential decision-making? Second, how does this compare with recent deep generative models for time series?

Imitation Perspective In sequential decision-making, *imitation learning* deals with training a policy purely on the basis of demonstrated behavior—that is, with no knowledge of the reward signals that induced the behavior in the first place [54–56]. Consider the standard Markov decision process setting, with states $z \in \mathcal{Z}$, actions $u \in \mathcal{U}$, dynamics $\omega \in \Delta(\mathcal{Z})^{\mathcal{Z}\times\mathcal{U}}$, and rewards $\rho \in \mathbb{R}^{\mathcal{Z}\times\mathcal{U}}$. Classically, imitation learning seeks to minimize the regret $\mathcal{R}_s(\theta) := \mathbb{E}_{\pi_s}[\sum_t \rho(z_t, u_t)] - \mathbb{E}_{\pi_\theta}[\sum_t \rho(z_t, u_t)]$, with $\pi_s, \pi_\theta \in \Delta(\mathcal{U})^{\mathcal{Z}}$ here being the demonstrator and imitator policies, and expectations are taken over episodes generated per $u_t \sim \pi(\cdot|z_t)$ and $z_{t+1} \sim \omega(\cdot|z_t, u_t)$ [57,58]. First, observe that by interpreting h as "states" and x as "actions", our problem setup bears a precise resemblance to imitation learning:

Corollary 6 (Generation as Imitation) Let state space $\mathcal{Z} := \mathcal{H}$, action space $\mathcal{U} := \mathcal{X}$, and reward function $\rho := f_s$. In addition, let the dynamics be such that $\omega(\cdot|h_t, x_t)$ is the Dirac delta centered at $h_{t+1} := (x_1, ..., x_t)$. Then the regret exactly corresponds to the expected quality difference: $\mathcal{R}_s = \Delta\bar{F}_s$.

Proof. Immediate from Definition 1. □

Now, since we want low regret but have no knowledge of the true quality measure (i.e. "reward signal"), we may naturally learn it together. In this sense, TimeGCI is analogous to imitation by *inverse reinforcement learning* (IRL), which seeks to infer rewards that plausibly induced the demonstrated behavior, and to optimize imitating policies on that basis [59,60]. Further, in simultaneously optimizing for variety (cf. entropy) and typicality (cf. energy), TimeGCI is analogous to maximum-entropy IRL [61–63]. Our contrastive approach also bears mild resemblance to stepwise discriminators proposed in this vein [64,65], although our framework focuses on trajectory-wise modeling, and is not adversarial.

There are also crucial differences: In imitation learning, dynamics are generally Markovian; states are readily defined as discrete elements or real vectors, and action spaces are small/discrete. The practical challenge is sample efficiency—to reduce the cost of environment interactions [66,67]. In time-series generation, however, rollouts are free—generating a synthetic trajectory does not require interacting with the real world. But dynamics are never Markovian: The practical challenge is that representations of variable-length histories must be jointly learned; moreover, actions are the full-dimensional feature vectors themselves, which renders policy optimization more demanding than usual (see Appendix B).

Related Work Table 1 summarizes key differentiators of TimeGCI from prevailing techniques. As discussed in Section 1, MLE-based autoregressive models [5,10,13] are easy to optimize, and learn explicit conditional distributions that can be used for inspection, resampling, or uncertainty estimation, but they suffer from exposure bias [8,11,12]. GAN-based adversarial models fall into two camps: For trajectory-centric methods [18,19,68], with only sequence-level signals to guide the generator, they often struggles to converge to the adversarial objective without extensive tuning [12]—with the exception of [68], which proposes Sinkhorn divergences instead. Transition-centric methods [12,21,22] provide more granular signals to guide the generator, but this simply alters the objective of learning p_s to one of learning μ_s, and still inherits the disadvantages of implicit, adversarial learning.

To the best of our knowledge, TimeGCI is the first framework for learning p_s that satisfies all our key criteria. Our analysis is built on ideas from energy-based models (EBMs) [69–71] and reinforcement learning for sequence prediction [72–74]. In particular, our initial formulation (Section 3.1) can be viewed as a temporal extension of variational EBMs [41,42]. Moreover, by adaptively learning π_θ to give negative samples for $d_{\theta,\phi}$, our proposed formulation (Section 3.2) can be viewed as a temporal analogue of noise-contrastive estimation (NCE) [51,75]. More tangentially, conditional EBMs have been trained with NCE for text generation [76–78], and the strength of global normalization has been studied [79]; that said, these are confined to the case where external input tokens are available for conditioning at each step—and not free-running as in our time-series setting. Finally, note that viewing sequence generation as a decision-making problem is present in language modeling [6,35] where task-specific metrics are available as signals. In the absence of predefined signals, GAN-based methods that jointly train discriminators to provide rewards for imitation have been proposed [80–85], although they are adversarial, and all focus on the special case of generating discrete tokens for language modeling.

5 Experiments

Benchmarks We test Algorithm 1 (**TimeGCI**) against the following: The classic Teacher Forcing trains autoregressive networks using ground-truth conditioning (**T-Forcing**) [5]. Professor Forcing uses adversarial domain adaptation by training an auxiliary discriminator to encourage dynamics of the network's free-running and teacher-forced states to be similar (**P-Forcing**) [10]. Trajectory-centric recurrent GANs (**C-RNN-GAN**) directly plug RNNs into the GAN framework as generators and discriminators for full sequences [18]. Causal Optimal Transport GAN (**COT-GAN**) is the latest variant of this [20], proposing to approximate Sinkkorn divergences instead of the standard JS divergence. For transition-centric recurrent GANs (**RC-GAN**), the adversarial loss is computed as the sum of log likelihoods for the stepwise feature vectors conditioned on histories [21], instead of directly as the log likelihood for the entire sequence. Finally, Time-series GAN (**TimeGAN**) is its latest incarnation [12], proposing to generate and discriminate within a jointly optimized embedding space for efficiency.

Datasets We employ five tabular time-series datasets with a variety of different characteristics, such as periodicity, noise level, and correlations: First, we use a synthetic dataset of multivariate sinusoids with different frequencies and phases (**Sines**) [12]. Second, we use a UCI dataset from the monitored energy usage of household appliances in a low-energy house (**Energy**) [86]. Third, we use a UCI dataset from temperature-modulated semiconductor gas sensors for chemical detection (**Gas**) [87]. Fourth, we use a UCI dataset of hourly interstate vehicle volume at a state traffic recording station (**Metro**) [88]. Fifth, we use a medical dataset of intensive-care patients from the Medical Information Mart for Intensive Care (**MIMIC-III**) [89]. All datasets are accessible from their sources, and we use the original source code for preprocessing sines and the UCI datasets by [12], publicly available at [90].

Implementation Experiments for each dataset are arranged as follows: The real trajectories that constitute the original dataset $\mathcal{D}$ are fed as input to train all algorithms. Each algorithm is subsequently used in test mode to generate 10,000 synthetic trajectories. Then, the performance of each algorithm is evaluated on the basis of these generated trajectories. This process is then performed for a total of 10 repetitions, from which we compile the means and standard errors for each reported result. For fair comparison, analogous network components across all benchmarks share the same recurrent architecture: Wherever a generator, policy, discriminator, energy, or critic network applies, we use LSTMs with one hidden layer of 32 units and ELU activations to compute hidden states h, and two fully-connected hidden layers of 32 units each and ELU activations to compute task-specific output variables (i.e. the generator output, policy parameters, discriminator output, energy functions, or critic values). In other respects, we use the publicly available source code to construct the benchmark algorithms—accessible at [90–94]. See Appendix C for additional detail on hyperparameters and implementations.

Evaluation and Results In the tabular data setting, assessing synthetic data generation is inherently tricky [27,95,96]: Unlike in media-specific applications, we have no predefined measures such as music polyphony or BLEU scores, nor can we use human evaluation of realism as done for videos. For tabular time-series, the generally accepted standard for comparing synthetic data is to apply the *Train-on-Synthetic, Test-on-Real* (TSTR) framework, first proposed by [21] and employed by most recent work in synthetic time-series generation [12,14,21,22,26,97], as well as more generally for tabular synthetic data of any kind [95,96,98]. Specifically, we apply the performance measure used by [12,14,97] to quantify how much the synthetic sequences inherit the predictive characteristics of the original dataset (**Predictive Score**): Using synthetic samples, a generic post-hoc sequence-prediction model is learned to forecast next-step feature vectors over training sequences. Then, the trained model

Table 2: *Performance Comparison of TimeGCI and Benchmarks*. Bold numbers indicate best-performing results.

Benchmark	*Metric*	Sines	Energy	Gas	Metro	MIMIC-III
T-Forcing	Predictive Score	0.108 ± 0.002	0.310 ± 0.001	0.035 ± 0.003	0.242 ± 0.001	0.017 ± 0.001
	+3 Steps Ahead	0.115 ± 0.001	0.281 ± 0.001	0.080 ± 0.001	0.244 ± 0.001	0.024 ± 0.007
	+5 Steps Ahead	0.122 ± 0.003	0.270 ± 0.002	0.111 ± 0.001	0.248 ± 0.001	0.018 ± 0.003
	x-Corr. Score	8.369 ± 0.015	194.1 ± 0.043	150.8 ± 0.067	4.222 ± 0.013	400.9 ± 3.203
P-Forcing	Predictive Score	0.105 ± 0.001	0.303 ± 0.002	0.037 ± 0.001	0.241 ± 0.001	0.023 ± 0.006
	+3 Steps Ahead	0.110 ± 0.001	0.268 ± 0.002	0.086 ± 0.002	0.241 ± 0.001	0.018 ± 0.001
	+5 Steps Ahead	0.115 ± 0.001	0.259 ± 0.002	0.121 ± 0.002	0.242 ± 0.001	0.017 ± 0.001
	x-Corr. Score	8.156 ± 0.010	207.6 ± 0.057	150.5 ± 0.023	3.014 ± 0.006	346.6 ± 2.901
C-RNN-GAN	Predictive Score	0.751 ± 0.001	0.500 ± 0.001	0.242 ± 0.001	0.419 ± 0.005	0.019 ± 0.001
	+3 Steps Ahead	0.769 ± 0.001	0.500 ± 0.001	0.243 ± 0.001	0.416 ± 0.002	0.020 ± 0.001
	+5 Steps Ahead	0.786 ± 0.001	0.501 ± 0.001	0.241 ± 0.001	0.416 ± 0.003	0.019 ± 0.001
	x-Corr. Score	10.76 ± 0.012	644.2 ± 0.112	266.4 ± 0.008	18.39 ± 0.003	$1720. \pm 0.339$
COT-GAN	Predictive Score	0.099 ± 0.001	0.259 ± 0.001	0.022 ± 0.001	0.245 ± 0.001	0.014 ± 0.001
	+3 Steps Ahead	0.109 ± 0.001	0.261 ± 0.001	0.050 ± 0.001	0.246 ± 0.001	0.013 ± 0.001
	+5 Steps Ahead	0.110 ± 0.001	0.262 ± 0.001	0.072 ± 0.001	0.245 ± 0.001	0.013 ± 0.001
	x-Corr. Score	3.114 ± 0.038	$\mathbf{67.93 \pm 0.227}$	$\mathbf{25.56 \pm 0.156}$	3.055 ± 0.013	497.7 ± 2.581
RC-GAN	Predictive Score	0.751 ± 0.001	0.498 ± 0.001	0.243 ± 0.001	0.412 ± 0.003	0.019 ± 0.001
	+3 Steps Ahead	0.770 ± 0.001	0.500 ± 0.001	0.244 ± 0.001	0.415 ± 0.004	0.019 ± 0.001
	+5 Steps Ahead	0.786 ± 0.001	0.499 ± 0.001	0.243 ± 0.001	0.418 ± 0.004	0.018 ± 0.001
	x-Corr. Score	5.649 ± 0.012	582.3 ± 0.047	231.2 ± 0.003	19.77 ± 0.001	$1592. \pm 0.192$
TimeGAN	Predictive Score	0.196 ± 0.006	0.261 ± 0.001	0.264 ± 0.011	0.245 ± 0.002	0.502 ± 0.023
	+3 Steps Ahead	0.223 ± 0.006	0.263 ± 0.001	0.251 ± 0.014	0.243 ± 0.001	0.484 ± 0.021
	+5 Steps Ahead	0.246 ± 0.005	0.262 ± 0.005	0.252 ± 0.012	0.242 ± 0.001	0.453 ± 0.020
	x-Corr. Score	17.86 ± 0.001	667.5 ± 0.001	282.5 ± 0.001	17.11 ± 0.001	$2140. \pm 0.010$
TimeGCI (Ours)	Predictive Score	$\mathbf{0.097 \pm 0.001}$	$\mathbf{0.251 \pm 0.001}$	$\mathbf{0.018 \pm 0.000}$	$\mathbf{0.239 \pm 0.001}$	$\mathbf{0.002 \pm 0.000}$
	+3 Steps Ahead	$\mathbf{0.104 \pm 0.001}$	$\mathbf{0.251 \pm 0.001}$	$\mathbf{0.042 \pm 0.001}$	$\mathbf{0.239 \pm 0.001}$	$\mathbf{0.001 \pm 0.000}$
	+5 Steps Ahead	$\mathbf{0.109 \pm 0.001}$	$\mathbf{0.251 \pm 0.001}$	$\mathbf{0.067 \pm 0.001}$	$\mathbf{0.239 \pm 0.001}$	$\mathbf{0.001 \pm 0.000}$
	x-Corr. Score	$\mathbf{1.195 \pm 0.011}$	105.2 ± 0.433	47.91 ± 0.811	$\mathbf{0.738 \pm 0.019}$	$\mathbf{194.3 \pm 0.180}$

is evaluated on the original data, and its predictive performance is quantified in terms of the mean absolute error. We use the original source code for computing this metric, publicly available at [90].

Unlike prior works using this measure, however, we argue that any TSTR evaluation should be more general than just next-step forecasting. After all, the distinguishing characteristic of sequential (vs. static) data generation is that we care about evolution of features *over time*. Hence we additionally compute TSTR metrics for horizons of lengths three and five (**+3 Steps Ahead** and **+5 Steps Ahead**). Importantly, note that a key strength of TSTR evaluation is in its sensitivity to *mode collapse*: If any generation scheme suffers from mode collapse (as GAN methods are prone to), TSTR scores would degrade due to the synthetic data failing to capture the diversity of the real data, which means any prediction model trained on that basis would also fail to capture this variation). Finally, similar to some recent works [19,20], we also compute the cross-correlations of real and synthetic feature vectors, and report the sum of the absolute differences between them, averaged over time (**x-Corr. Score**); this serves to verify if feature relationships are preserved well, in addition to temporal relationships. Table 2 shows the results: With respect to the above metrics, we find that TimeGCI quite consistently produces synthetic samples that perform similarly or better than benchmark algorithms in all datasets. (Note that we do empirically observe several instances of mode collapse in GAN-based benchmarks).

6 Conclusion

In this work, we motivated and presented TimeGCI for time-series generative modeling, combining the flexibility of energy-based models with the tractability of explicit conditional models; this yielded a novel algorithm with advantages in sampling, evaluating, and learning, as well as a new perspective on "generation as imitation". Two caveats are in order: First, while we began from the notion of moment-matching to bound error compounding, in practice there is no guarantee that this is accomplished well during optimization. While our approach dispenses with the instabilities typical of adversarial training, our reliance on policy optimization comes with its own complexities, and has meant that an additional critic network is required. Indeed, should this approach be extended beyond our tabular setting into more high-dimensional media, the policy updates may become quite difficult to perform. By way of conclusion, we reiterate that a perennial challenge in modeling tabular data is in the choice of evaluation metric. While we opted for the most commonly accepted method of TSTR, future work will benefit from a deeper investigation into more sophisticated measures for time series. More broadly, we emphasize that synthetic data generated by agnostic approaches such as ours will often replicate any elements of bias or unfairness present in the original data. In sensitive applications, future work may consider possible approaches to generating synthetic time-series data with fairness constraints.

References

[1] Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*, 2018.

[2] Anna L Buczak, Steven Babin, and Linda Moniz. Data-driven approach for creating synthetic electronic medical records. *BMC medical informatics and decision making*, 2010.

[3] Saloni Dash, Andrew Yale, Isabelle Guyon, and Kristin P Bennett. Medical time-series data generation using generative adversarial networks. *International Conference on Artificial Intelligence in Medicine (AIME)*, 2020.

[4] Qingsong Wen, Liang Sun, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. *arXiv preprint*, 2020.

[5] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989.

[6] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *International Conference on Learning Representations (ICLR)*, 2016.

[7] Yoshua Bengio and Paolo Frasconi. An input output hmm architecture. *Advances in neural information processing systems (NeurIPS)*, 1995.

[8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. *International Conference on Machine Learning (ICML)*, 2009.

[9] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research (JMLR)*, 2016.

[10] Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[11] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *International Conference on Learning Representations (ICLR)*, 2016.

[12] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[13] Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[14] Ahmed M Alaa, Alex J Chan, and Mihaela van der Schaar. Generative time-series modeling with fourier flows. *International Conference on Learning Representations (ICLR)*, 2020.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

[17] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[18] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[19] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Generating high-fidelity, synthetic time series datasets with doppelganger. *ACM Internet Measurement Conference (IMC)*, 2019.

[20] Tianlin Xu, Li K Wenliang, Michael Munn, and Beatrice Acciaio. Cot-gan: Generating sequential data via causal optimal transport. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[21] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint*, 2017.

[22] Giorgia Ramponi, Pavlos Protopapas, Marco Brambilla, and Ryan Janssen. T-cgan: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling. *arXiv preprint*, 2018.

[23] Luca Simonetto. Generating spiking time series with generative adversarial networks: an application on banking transactions. 2018.

[24] Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. Sensegen: A deep learning architecture for synthetic sensor data generation. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 188–193. IEEE, 2017.

[25] Shota Haradal, Hideaki Hayashi, and Seiichi Uchida. Biosignal data augmentation based on generative adversarial networks. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 368–371. IEEE, 2018.

[26] Chi Zhang, Sanmukh R Kuppannagari, Rajgopal Kannan, and Viktor K Prasanna. Generative adversarial network for synthetic time series data generation in smart grids. In *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6. IEEE, 2018.

[27] Ahmed M Alaa, Boris van Breugel, Evgeny Saveliev, and Mihaela van der Schaar. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. *International Conference on Machine Learning (ICML)*, 2021.

[28] Aditya Grover, Jiaming Song, Alekh Agarwal, Kenneth Tran, Ashish Kapoor, Eric Horvitz, and Stefano Ermon. Bias correction of learned generative models using likelihood-free importance weighting. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[29] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *International Conference on Learning Representations (ICLR)*, 2019.

[30] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *International Conference on Learning Representations (ICLR)*, 2019.

[31] Weili Nie, Nina Narodytska, and Ankit Patel. Relgan: Relational generative adversarial networks for text generation. *International Conference on Learning Representations (ICLR)*, 2019.

[32] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *International Conference on Learning Representations (ICLR)*, 2020.

[33] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[34] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[35] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *International Conference on Learning Representations (ICLR)*, 2017.

[36] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. *Advances in Neural Information Processing Systems (NeurIPS)*, 2000.

[37] Peter D Grünwald, A Philip Dawid, et al. Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *Annals of Statistics*, 2004.

[38] Farzan Farnia and David Tse. A minimax approach to supervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[39] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. *Dissertation, Carnegie Mellon University*, 2010.

[40] Michael I Jordan. An introduction to probabilistic graphical models. 2003.

[41] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *International Conference on Learning Representations (ICLR)*, 2016.

[42] Shuangfei Zhai, Yu Cheng, Rogerio Feris, and Zhongfei Zhang. Generative adversarial networks as variational training of energy based models. *arXiv preprint*, 2016.

[43] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. *International Conference on Learning Representations (ICLR)*, 2017.

[44] Rithesh Kumar, Sherjil Ozair, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *arXiv preprint*, 2019.

[45] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2016.

[46] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *International Conference on Machine Learning (ICML)*, 2017.

[47] Wenjie Shi, Shiji Song, and Cheng Wu. Soft policy gradient method for maximum entropy deep reinforcement learning. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[48] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[49] Josiah P Hanna and Peter Stone. Towards a data efficient off-policy policy gradient. *AAAI Symposium on Data Efficient Reinforcement Learning (AAAI)*, 2018.

[50] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research (JMLR)*, 2012.

[51] Ian J Goodfellow. On distinguishability criteria for estimating generative models. *International Conference on Learning Representations (ICLR)*, 2015.

[52] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning. *MIT Press Cambridge*, 2016.

[53] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning (ICML)*, 2018.

[54] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *International conference on artificial intelligence and statistics (AISTATS)*, 2011.

[55] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 2018.

[56] Alexandre Attia and Sharone Dayan. Global overview of imitation learning. *arXiv preprint*, 2018.

[57] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. *International conference on artificial intelligence and statistics (AISTATS)*, 2010.

[58] Umar Syed and Robert E Schapire. A reduction from apprenticeship learning to classification. *Advances in neural information processing systems (NeurIPS)*, 2010.

[59] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. *International conference on Machine learning (ICML)*, 2000.

[60] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. *International conference on Machine learning (ICML)*, 2004.

[61] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. *AAAI Conference on Artificial Intelligence (AAAI)*, 2008.

[62] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. *International conference on machine learning (ICML)*, 2016.

[63] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *NeurIPS Workshop on Adversarial Training*, 2016.

[64] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2018.

[65] Ahmed H Qureshi, Byron Boots, and Michael C Yip. Adversarial imitation via variational inverse reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2019.

[66] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation. *International Conference on Learning Representations (ICLR)*, 2019.

[67] Lionel Blondé and Alexandros Kalousis. Sample-efficient imitation learning via gans. *International conference on artificial intelligence and statistics (AISTATS)*, 2019.

[68] Huan Xu and Shie Mannor. Distributionally robust markov decision processes. *Mathematics of Operations Research*, 2012.

[69] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting Structured Data*, 2006.

[70] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. *International Conference on Machine Learning (ICML)*, 2016.

[71] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[72] Philip Bachman and Doina Precup. Data generation as sequential decision making. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[73] Arun Venkatraman, Martial Hebert, and J Bagnell. Improving multi-step prediction of learned time series models. *AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[74] Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K Reddy. Deep reinforcement learning for sequence-to-sequence models. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2019.

[75] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[76] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *International Conference on Machine Learning (ICML)*, 2012.

[77] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[78] Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[79] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

[80] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[81] Sidi Lu, Lantao Yu, Siyuan Feng, Yaoming Zhu, and Weinan Zhang. Cot: Cooperative training for generative modeling of discrete data. *International Conference on Machine Learning (ICML)*, 2019.

[82] Haiyan Yin, Dingcheng Li, Xu Li, and Ping Li. Meta-cotgan: A meta cooperative training paradigm for improving adversarial text generation. *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[83] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: better text generation via filling in the_. *International Conference on Learning Representations (ICLR)*, 2018.

[84] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[85] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. *International Conference on Machine Learning (ICML)*, 2017.

[86] Luis M Candanedo, Véronique Feldheim, and Dominique Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and buildings*, 2017.

[87] Javier Burgués, Juan Manuel Jiménez-Soto, and Santiago Marco. Estimation of the limit of detection in semiconductor gas sensors through linearized calibration models. *Analytica Chimica Acta*, 2018.

[88] John Hogue. Hourly interstate 94 westbound traffic volume for mn dot atr station 301. *Minnesota Department of Transportation*, 2018.

[89] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Nature Scientific Data*, 2016.

[90] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. https://github.com/jsyoon0823/TimeGAN, 2019.

[91] Tianlin Xu, Li K Wenliang, Michael Munn, and Beatrice Acciaio. Cot-gan: Generating sequential data via causal optimal transport. https://github.com/tianlinxu312/cot-gan, 2020.

[92] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. https://github.com/olofmogren/c-rnn-gan, 2016.

[93] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. https://github.com/ratschlab/RGAN, 2017.

[94] Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor forcing. https://github.com/anirudh9119/LM_GANS, 2016.

[95] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[96] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *International Conference on Very Large Data Bases (VLDB)*, 2018.

[97] Mohammad Navid Fekri, Ananda Mohon Ghosh, and Katarina Grolinger. Generating energy data for machine learning with recurrent generative adversarial networks. *Energies*, 2020.

[98] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. *International Conference on Learning Representations (ICLR)*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code will be made available upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]