Time-series Generation by Contrastive Imitation

Daniel Jarrett
University of Cambridge, UK
daniel.jarrett@maths.cam.ac.uk

Ioana Bica
University of Oxford, UK
Alan Turing Institute, UK
ioana.bica@eng.ox.ac.uk

Mihaela van der Schaar
University of California, Los Angeles
University of Cambridge, UK
Alan Turing Institute, UK
mv472@cam.ac.uk

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 study a generative framework that seeks to combine the strengths of both: Motivated by a 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. At training, the two components are learned cooperatively, avoiding the instabilities typical of adversarial objectives. At inference, the learned policy serves as the generator for iterative sampling, and the learned energy 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, this approach embodies "*generation by imitation*". Theoretically, we illustrate the correctness of this formulation and the consistency of the algorithm. Empirically, we evaluate its ability to generate predictively useful samples from real-world datasets, verifying that it performs at 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.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

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 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 Operations Consider a probabilistic generative model p for some dataset $\mathcal{D}$. We are generally interested in performing one or more of the following operations: (1) *sampling* a time series $\tau \sim p$, (2) *evaluating* the likelihood $p(\tau)$, and (3) *learning* the model p from a set of i.i.d. samples τ. In light of the preceding, we investigate a generative framework that attempts to fulfill the following criteria:

- Samples should respect both the stepwise *conditional* distributions of features, as well as the *joint* distribution of full trajectories; unlike pure MLE, we wish to avoid multi-step compounding error.
- Evaluating likelihoods should be possible as generic measures of *sample quality* for both transitions and trajectories—often desired for sample comparison, model auditing, or bias correction [27, 28].
- Unlike black-box GAN discriminators, we wish that the evaluator be *decoupled* from any specific sampler, such that the two components can be trained *non-adversarially*, thus may be more stable.

Contributions In the sequel, we explore an approach that seeks to satisfy these criteria. 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: We train a forward-looking *transition policy* to imitate the sequential behavior of time series using a stepwise-decomposable *energy model* as reinforcement, giving a method that embodies "*generation by imitation*" (Section 3). Importantly, to understand its strengths and limitations, we compare the 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, we verify that it generates predictively useful samples that perform at 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 $\tau := (x_1, ..., x_T) \in \mathcal{T} := \mathcal{X}^T$. Also, denote with $h_t := (x_1, ..., x_{t-1}) \in \mathcal{H} := \cup_{t=1}^{T} \mathcal{X}^t$ 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 $\mathcal{D} := \{\tau_n\}_{n=1}^{N}$ 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 $\mu_s(h) := \frac{1}{T}\sum_t p(h_t = h|\pi_s)$ 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 $\mu_s(h, x) := \mu_s(h)\pi_s(x|h)$ to indicate the marginal distribution of transitions. Finally, let the joint distribution of full trajectories be denoted by $p_s(\tau) := \prod_t \pi_s(x_t|h_t)$.

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 being an agnostic, *framework*-level study applicable to 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 $f_s : \mathcal{H} \times \mathcal{X} \to [-c, c]$ with $c < \infty$, 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 $F_s : \mathcal{T} \to [-cT, cT]$ 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 $\Delta\bar{F}_s : \Theta \to [-2cT, 2cT]$ 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 this 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, we have (however, recall that strong duality does not generalize to the nonlinear case; see Appendix A):

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

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. Now, although at the global optimum the moment-matching discrepancy must be zero (i.e. the equality constraint is enforced), in practice there may be a variety of reasons why this is not perfectly achieved (e.g. error in estimating expectations, error in

function approximation, error in optimization, etc). Suppose we could bound how well we are able to enforce the moment-matching constraint; as it turns out, we cannot eliminate error compounding:[1]

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 $\mathcal{L}(p_s, p_\theta) \coloneqq D_{\text{KL}}(p_s \| p_\theta)$. 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 $\arg\min_\theta \left(\mathbb{E}_{\tau \sim p_\theta} \log p_\theta(\tau) + \max_{F \in \mathbb{R}^{\mathcal{T}}} \left(\mathbb{E}_{\tau \sim p_s} F(\tau) - \mathbb{E}_{\tau \sim p_\theta} F(\tau) \right) \right)$ (but again, recall here that strong duality does not generalize to the nonlinear case; see Appendix A). 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 measure μ_θ. There is no longer any "exposure bias" here: 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 measure μ_θ induced by π_θ also has to be close to the occupancy measure μ_s induced by π_s. As it turns out, this seemingly minor difference is sufficient to mitigate compounding errors. As before, although at the global optimum the moment-matching discrepancy must be zero, in practice this may not be perfectly achieved. Now, suppose we could bound how well we are able to enforce the moment-matching constraint; but we now have:

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 we shall see, this endeavor gives rise to a technique that trains a conditional policy (for sampling), an energy model (for evaluation), and a non-adversarial framework (for learning)—addressing our three initial criteria.

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 begin with the primal form of Equation 5, which yields the following adversarial learning objective:

[1] Lemmas 1 and 2 are similar in spirit to results for error accumulation in imitation by behavioral cloning and distribution matching. See Appendix A; this analogy with imitation learning is formally identified in Section 4.

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

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 of ϕ with nested (best-response) updates of 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,

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

and the inner-loop update for the policy,

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

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:

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

where the partition function is computed as $Z_\phi = \mathbb{E}_{\tau\sim p_\theta}[\exp(\sum_t f_\phi(h_t, x_t))/\prod_t \pi_\theta(x_t|h_t)]$, 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].

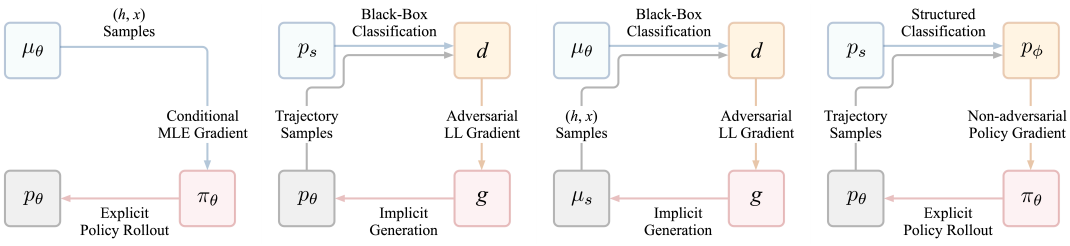
3.2 Contrastive Imitation

We now investigate a generative framework that seeks to avoid 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":[2]

Definition 2 (Structured Classification) Recall the π_θ-induced distribution $p_\theta(\tau) \coloneqq \prod_t \pi_\theta(x_t|h_t)$. Denote with $\tilde{p}_\phi$ the *un-normalized* energy-based model such that $\tilde{p}_\phi(\tau) \coloneqq \exp(\sum_t f_\phi(h_t, x_t))$, and let Z_ϕ be folded into ϕ as a learnable parameter. Define the *structured classifier* $d_{\theta,\phi} : \mathcal{T} \to [0, 1]$:

$$d_{\theta,\phi}(\tau) \coloneqq \frac{\frac{1}{Z_\phi}\tilde{p}_\phi(\tau)}{\frac{1}{Z_\phi}\tilde{p}_\phi(\tau) + p_\theta(\tau)} \quad (10)$$

[2]The idea that density estimation can be performed by logistic regression goes back at least to [50], and formalized as negative sampling [51] and noise-contrastive estimation [52]. Structured classifiers have been studied in the context of imitation learning [53, 54] by analogy with GANs. In the time-series setting, however, we shall see that this approach is equivalent to noise-contrastive estimation with an adaptive noise distribution.



(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.

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,

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

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: $p_s(\tau) > 0 \Rightarrow p_\theta(\tau) > 0$ (this does not require π_θ be optimal for f_ϕ). Then $\mathcal{L}_{\text{energy}}(\phi;\theta)$ is globally minimized at $F_\phi(\cdot) - \log Z_\phi = \log p_s(\cdot)$, whence p_ϕ is self-normalized with unit integral.

Proof. Appendix A. □

This result is intuitive by analogy with noise-contrastive estimation [52, 55]: 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:

Proposition 4 (Asymptotic Consistency) Let ϕ^* denote the minimizer for $\mathcal{L}_{\text{energy}}(\phi;\theta)$, and let $\hat{\phi}^*_M$ denote the minimizer 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 [56]. 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.[3] In fact, should we insist on greedily taking each policy update to optimality, we recover a "weighted" version of the original max-min gradient from before:

[3] It is easy to see that minimizing Equation 12 equivalently minimizes the reverse KL div. between p_ϕ and p_θ.

Proposition 5 (Gradient Equality) Let ϕ_k be the value taken by ϕ after the k-th gradient update, and let θ_k^* denote the associated minimizer for $\mathcal{L}_{\text{policy}}(\theta; \phi_k)$. Suppose p_ϕ is already normalized; then

$$\nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta_k^*) = -\tfrac{T}{2} \nabla_\phi \mathcal{L}(\theta_k^*, \phi)$$

That is, at θ_k^* the energy gradient (of Equation 11) recovers the original gradient (from Equation 7). In the general case, suppose p_ϕ is un-normalized, such that $p_{\theta_k^*} = p_\phi / K_\phi$ for some constant K_ϕ; then

$$\nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta_k^*) = \tfrac{TK_\phi}{K_\phi+1} \mathbb{E}_{\substack{h\sim\mu_{\theta_k^*} \\ x\sim\pi_{\theta_k^*}(\cdot|h)}} \nabla_\phi f_\phi(h,x) - \tfrac{T}{K_\phi+1} \mathbb{E}_{\substack{h\sim\mu_s \\ x\sim\pi_s(\cdot|h)}} \nabla_\phi f_\phi(h,x)$$

Proof. Appendix A. □

This "weighting" is intuitive: If p_ϕ were un-normalized such that $K_\phi > 1$, the energy loss automatically places higher weights on negative samples $h \sim \mu_{\theta_k^*}, x \sim \pi_{\theta_k^*}(\cdot|h)$ to bring it down; conversely, if p_ϕ were un-normalized such that $K_\phi < 1$, the energy loss places higher weights on positive samples $h \sim \mu_s, x \sim \pi_s(\cdot|h)$ to bring it up. (If p_ϕ were normalized, then $K_\phi = 1$ and the weights are equal). 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, regularization coefficient κ, 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**
    $\theta \leftarrow \theta - \lambda_{\text{actor}} \nabla_\theta \mathcal{L}_{\text{actor}}(\theta; \phi, \psi) + \kappa \nabla_\theta \mathcal{L}_{\text{mle}}(\theta)$ ▷ Update policy
    $\phi \leftarrow \phi - \lambda_{\text{energy}} \nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta)$ ▷ Update energy
    $\psi \leftarrow \psi - \lambda_{\text{critic}} \nabla_\psi \mathcal{L}_{\text{critic}}(\psi; \phi)$ ▷ 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* [57], 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 $\mathcal{L}_{\text{actor}}(\theta; \phi, \psi) \coloneqq \mathbb{E}_{h\sim\mathcal{B}} \mathbb{E}_{x\sim\pi_\theta(\cdot|h)}[\log \pi_\theta(x|h) - Q_\psi(h,x)]$, where $Q_\psi : \mathcal{H} \times \mathcal{X} \to \mathbb{R}$ is the transition-wise soft value function parameterized by ψ, and $\mathcal{B}$ is a replay buffer of samples generated by π_θ. For stability, the actor is regularized with the conditional MLE loss $\mathcal{L}_{\text{mle}}(\theta) \coloneqq \mathbb{E}_{x\sim\pi_s(\cdot|h)} \log \pi_\theta(x|h)$. The critic is trained to minimize the soft Bellman residual: $\mathcal{L}_{\text{critic}}(\psi; \phi) \coloneqq \mathbb{E}_{h,x\sim\mathcal{B}}(Q_\psi(h,x) - f_\phi(h,x) - V_\psi(h'))^2$, where the state-values are bootstrapped as $V_\psi(h') \coloneqq \mathbb{E}_{x'\sim\pi_\theta(\cdot|h')}[Q_\psi(h',x') - \log \pi_\theta(x'|h')]$. 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.

4 Discussion

Our theoretical motivations are apparent (Sections 2.2–3.1), and the practical mechanics of optimization are straightforward (Section 3.2–3.3). To understand the strengths and limitations 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 [58–60]. 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 $\mathcal{R}_s(\theta) \coloneqq \mathbb{E}_{\pi_s}[\sum_t \rho(z_t,u_t)] - \mathbb{E}_{\pi_\theta}[\sum_t \rho(z_t,u_t)]$, 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 $z_{t+1} \sim \omega(\cdot|z_t,u_t)$ [61,62]. First, observe that by interpreting h as "states" and x as "actions", our problem setup bears a precise resemblance to imitation learning:

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	✓	✓	✓	✓	✓

Corollary 6 (Generation as Imitation) Let state space $\mathcal{Z} \coloneqq \mathcal{H}$, action space $\mathcal{U} \coloneqq \mathcal{X}$, and reward function $\rho \coloneqq f_s$. In addition, let the dynamics be such that $\omega(\cdot|h_t, x_t)$ is the Dirac delta centered at $h_{t+1} \coloneqq (x_1, ..., x_t)$. 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 [63–66]. Further, in simultaneously optimizing for variety (cf. entropy) and typicality (cf. energy), TimeGCI is analogous to maximum-entropy IRL [67, 68]. Our contrastive approach also bears mild resemblance to stepwise discriminators studied in this vein [54, 69], although our framework focuses on trajectory-wise modeling, and is not adversarial (see Appendix D for more discussion on how TimeGCI relates to popular imitation learning methods).

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 [70, 71]. 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); beyond the tractable tabular settings we experiment in, higher-dimensional data may prove challenging.

Related Work Table 1 summarizes the 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, 72], 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 [72], which utilizes 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.

Our analysis is built on ideas from energy-based models (EBMs) [73–75] and reinforcement learning for sequence prediction [76–78]. 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 $d_{\theta,\phi}$, the formulation we study (Section 3.2) is equivalent to a temporal analogue of noise-contrastive estimation (NCE) [55, 79]. More tangentially, conditional EBMs have been trained with NCE for text generation [80–82], and the strength of global normalization has been studied [83]; 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 studied [84–89], 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**) [90]. Third, we use a UCI dataset from temperature-modulated semiconductor gas sensors for chemical detection (**Gas**) [91]. Fourth, we use a UCI dataset of hourly interstate vehicle volume at a state traffic recording station (**Metro**) [92]. Fifth, we use a medical dataset of intensive-care patients from the Medical Information Mart for Intensive Care (**MIMIC-III**) [93]. 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 [94]. Table 2 shows summary statistics for the datasets used in the experiments.

Table 2: *Summary Statistics for Datasets Used.*

Dataset	Dimension	Length	Autocor.	+3 Lag	+5 Lag
Sines	5	24	0.875	0.623	0.377
Metro	9	24	0.429	0.200	0.029
Gas	20	24	0.656	0.382	0.170
Energy	29	24	0.702	0.411	0.176
MIMIC-III	52	24	0.532	0.212	0.059

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 to compute hidden states for representing histories 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 [94–98]. 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, 99, 100]: 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, 101], as well as more generally for tabular synthetic data of any kind [99, 100, 102]. Specifically, we apply the performance measure used by [12, 14, 101] 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 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 [94].

Further to prior works using this measure, we additionally believe that synthetic data evaluation should be more general than just next-step TSTR forecasting. After all, the distinguishing characteristic of sequential (vs. static) data generation is that we care about evolution of features *over time*. Hence we also compute TSTR metrics for horizons of other lengths (**+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

Table 3: *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	$\mathbf{0.738 \pm 0.019}$	$\mathbf{194.3 \pm 0.180}$

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 3 shows the results: With respect to these metrics, we find that TimeGCI somewhat 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 invite an explicit analogy between time-series generation and imitation learning, and explore a framework that fleshes out this connection. Two caveats are in order: First, while we began from the notion of moment-matching to address the error compounding problem, in practice there is no guarantee that this is accomplished well during optimization. In particular, *scalability* is a major limitation beyond the range of feature dimensions and sequence lengths considered in our experiments. Sample-based estimates could rapidly degrade with the horizon, especially if transitions are highly stochastic. A relevant question is whether or not training on fixed subsequence lengths could potentially alleviate this concern for longer sequences. In addition, while our approach seeks to dispense with the instabilities typical of adversarial training, we are instead left with the difficulties of policy optimization, which may prove a prohibitive challenge in higher-dimensional feature spaces. For the datasets we consider, we find that pre-training and regularizing the policy with maximum likelihood, combined with a small enough learning rate, had the most impact in promoting stability and learning. Second, we reiterate that a perennial challenge in modeling tabular data is in choosing the metric for *evaluation*. While we opted for the most commonly accepted method of TSTR, this may not be general enough to capture the range of downstream tasks that may be performed on the synthetic data. Future work will benefit from a deeper investigation into more sophisticated measures for time series, such as contrastive methods and how to evaluate different aspects of the "quality" of the generated trajectories.

Acknowledgments

We would like to thank the reviewers for all their invaluable feedback. This work was supported by Alzheimer's Research UK, The Alan Turing Institute under the EPSRC grant EP/N510129/1, the US Office of Naval Research, as well as the National Science Foundation under grant number 1722516.

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] James Jordon, Daniel Jarrett, Evgeny Saveliev, Jinsung Yoon, Paul Elbers, Patrick Thoral, Ari Ercole, Cheng Zhang, Danielle Belgrave, and Mihaela van der Schaar. Hide-and-seek privacy challenge: Synthetic data generation vs. patient re-identification. *NeurIPS 2020 Competition and Demonstration Track*, 2021.

[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] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised as supervised learning. *The Elements of Statistical Learning*, 2009.

[51] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[52] 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.

[53] 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.

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

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

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

[57] 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.

[58] 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.

[59] 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.

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

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

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

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

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

[65] Ioana Bica, Daniel Jarrett, Alihan Hüyük, and Mihaela van der Schaar. Learning what-if explanations for sequential decision-making. *International Conference on Learning Representations (ICLR)*, 2021.

[66] Daniel Jarrett, Alihan Hüyük, and Mihaela Van Der Schaar. Inverse decision modeling: Learning interpretable representations of behavior. *International Conference on Machine Learning (ICML)*, 2021.

[67] 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.

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

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

[70] 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.

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

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

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

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

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

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

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

[78] 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.

[79] 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.

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

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

[82] 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.

[83] 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.

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

[85] 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.

[86] 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.

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

[88] 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.

[89] 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.

[90] 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.

[91] 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.

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

[93] 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.

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

[95] 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.

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

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

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

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

[100] 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.

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

[102] 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.

[103] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. *International Conference on Machine Learning (ICML)*, 2002.

[104] Alekh Agarwal, Nan Jiang, and Sham M Kakade. Reinforcement learning: Theory and algorithms. 2019.

[105] Gokul Swamy, Sanjiban Choudhury, Steven Wu, and Andrew Bagnell. Of moments and matching tradeoffs and treatments in imitation learning. *International Conference on Machine Learning (ICML)*, 2021.

[106] Tian Xu, Ziniu Li, and Yang Yu. Error bounds of imitating policies and environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[107] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems (NeurIPS)*, 2016.

[108] Nir Baram, Oron Anschel, and Shie Mannor. Model-based adversarial imitation learning. *International Conference on Machine Learning (ICML)*, 2017.

[109] Wonseok Jeon, Seokin Seo, and Kee-Eung Kim. A bayesian approach to generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[110] Xin Zhang, Yanhua Li, Ziming Zhang, and Zhi-Li Zhang. f-gail: Learning f-divergence for generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[111] Oleg Arenz and Gerhard Neumann. Non-adversarial imitation learning and its connections to adversarial methods. *arXiv preprint*, 2020.

[112] Tianwei Ni, Harshit Sikchi, Yufei Wang, Tejus Gupta, Lisa Lee, and Benjamin Eysenbach. F-irl: Inverse reinforcement learning via state marginal matching. *Conference on Robot Learning (CoRL)*, 2020.

A Proofs of Propositions

For Lemmas 1 and 2, we first introduce some additional quantities to enable more compact notation; in particular, we adopt the value-function terminology from imitation learning. The following definitions are standard and immediate from the mapping given by Corollary 6, but are explicitly stated here for completeness. Recall that $f_s : \mathcal{H} \times \mathcal{X} \to [-c, c]$ for some finite c. At any state h_t, define the "value function" to be the (forward-looking) expected sum of future quantities $f_s(h_u, x_u)$ for $u = t, ..., T$. Specifically, let $V^{\pi_\theta}_{s,t}(h) : \mathcal{H} \to [-cT, cT]$ and $Q^{\pi_\theta}_{s,t}(h, x) : \mathcal{H} \times \mathcal{X} \to [-cT, cT]$ be given as follows:

$$V^{\pi_\theta}_{s,t}(h) \coloneqq \mathbb{E}_{\tau \sim p_\theta}[\Sigma_{u=t}^{T} f_s(h_u, x_u) | h_t = h] \tag{13}$$

$$Q^{\pi_\theta}_{s,t}(h, x) \coloneqq \mathbb{E}_{\tau \sim p_\theta}[\Sigma_{u=t}^{T} f_s(h_u, x_u) | h_t = h, x_t = x] \tag{14}$$

where the notation for both $V^{\pi_\theta}_{s,t}$ and $Q^{\pi_\theta}_{s,t}$ is explicit as to their dependence on the policy π_θ being followed, the source s under consideration, and the time t—unlike in typical imitation learning, we operate in a non-stationary (and non-Markovian) setting. For Lemma 1, we require an additional result:

Lemma 7 (Expected Quality Difference) $\Delta \bar{F}_s(\theta) = T\mathbb{E}_{\substack{h \sim \mu_s \\ x \sim \pi_s(\cdot|h)}} Q^{\pi_\theta}_{s,t}(h, x) - T\mathbb{E}_{\substack{h \sim \mu_s \\ x \sim \pi_\theta(\cdot|h)}} Q^{\pi_\theta}_{s,t}(h, x)$.

Proof. From Definition 2,

$$\begin{aligned}
\Delta \bar{F}_s(\theta) &= \mathbb{E}_{\tau \sim p_s} \textstyle\sum_t f_s(h_t, x_t) - \mathbb{E}_{\tau \sim p_\theta} \textstyle\sum_t f_s(h_t, x_t) && (15)\\
&= \mathbb{E}_{\tau \sim p_s} \textstyle\sum_t (f_s(h_t, x_t) + V^{\pi_\theta}_{s,t}(h_t) - V^{\pi_\theta}_{s,t}(h_t)) - \mathbb{E}_{\tau \sim p_\theta} \textstyle\sum_t f_s(h_t, x_t) && (16)\\
&= \mathbb{E}_{\tau \sim p_s} \textstyle\sum_t (f_s(h_t, x_t) + V^{\pi_\theta}_{s,t+1}(h_{t+1}) - V^{\pi_\theta}_{s,t}(h_t)) && (17)\\
&= \mathbb{E}_{\tau \sim p_s} \textstyle\sum_t (Q^{\pi_\theta}_{s,t}(h_t, x_t) - V^{\pi_\theta}_{s,t}(h_t)) && (18)\\
&= T\mathbb{E}_{h \sim \mu_s, x \sim \pi_s(\cdot|h)} (Q^{\pi_\theta}_{s,t}(h, x) - V^{\pi_\theta}_{s,t}(h)) && (19)\\
&= T\mathbb{E}_{h \sim \mu_s, x \sim \pi_s(\cdot|h)} Q^{\pi_\theta}_{s,t}(h, x) - T\mathbb{E}_{h \sim \mu_s, x \sim \pi_\theta(\cdot|h)} Q^{\pi_\theta}_{s,t}(h, x) && (20)
\end{aligned}$$

where (16) to (17) telescopes terms, and we use the fact $V^{\pi}_{s,T+1}(h) = 0$. This derivation can be viewed as a non-stationary, non-Markovian analogue of the "performance difference" result in [103]. □

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. From Lemma 7,

$$\begin{aligned}
\Delta \bar{F}_s(\theta) &= T\mathbb{E}_{h \sim \mu_s, x \sim \pi_s(\cdot|h)} Q^{\pi_\theta}_{s,t}(h, x) - T\mathbb{E}_{h \sim \mu_s, x \sim \pi_\theta(\cdot|h)} Q^{\pi_\theta}_{s,t}(h) && (21)\\
&= T\mathbb{E}_{h \sim \mu_s} [\mathbb{E}_{x \sim \pi_s(\cdot|h)} Q^{\pi_\theta}_{s,t}(h, x) - \mathbb{E}_{x \sim \pi_\theta(\cdot|h)} Q^{\pi_\theta}_{s,t}(h, x)] && (22)\\
&\leq \max_{Q \in [-cT, cT]^{\mathcal{H} \times \mathcal{X}}} T\mathbb{E}_{h \sim \mu_s} [\mathbb{E}_{x \sim \pi_s(\cdot|h)} Q(h, x) - \mathbb{E}_{x \sim \pi_\theta(\cdot|h)} Q(h, x)] && (23)\\
&\leq \max_{f \in \mathbb{R}^{\mathcal{H} \times \mathcal{X}}} T\mathbb{E}_{h \sim \mu_s} [\mathbb{E}_{x \sim \pi_s(\cdot|h)} Tf(h, x) - \mathbb{E}_{x \sim \pi_\theta(\cdot|h)} Tf(h, x)] && (24)\\
&\leq T^2 \epsilon && (25)
\end{aligned}$$

where the final inequality applies the assumption from the lemma. Note that this is similar in spirit to various results for error accumulation in imitation learning through behavioral cloning. The most well-known one is [61], where a quadratic bound is given with respect to the *probability* that the learned policy makes a small mistake. Another well-known one is in [104], where the bound is given with respect to *sample complexity*. Here, in order to motivate our perspective from the notion of expected quality difference, our bound is given with respect to the *moment-matching* discrepancy, and can be interpreted as a non-Markovian variant of the "off-policy upper bound" result in [105]. □

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. From Definition 1,

$$\begin{aligned}
\Delta \bar{F}_s(\theta) &= \mathbb{E}_{\tau \sim p_s} \textstyle\sum_t f_s(h_t, x_t) - \mathbb{E}_{\tau \sim p_\theta} \textstyle\sum_t f_s(h_t, x_t) && (26)\\
&= T\mathbb{E}_{h \sim \mu_s, \pi_s(\cdot|h)} f_s(h, x) - T\mathbb{E}_{h \sim \mu_\theta, \pi_\theta(\cdot|h)} f_s(h, x) && (27)\\
&\leq \max_{f \in [-c, c]^{\mathcal{H} \times \mathcal{X}}} (T\mathbb{E}_{h \sim \mu_s, x \sim \pi_s(\cdot|h)} f(h, x) - T\mathbb{E}_{h \sim \mu_\theta, x \sim \pi_\theta(\cdot|h)} f(h, x)) && (28)\\
&\leq \max_{f \in \mathbb{R}^{\mathcal{H} \times \mathcal{X}}} (T\mathbb{E}_{h \sim \mu_s, x \sim \pi_s(\cdot|h)} f(h, x) - T\mathbb{E}_{h \sim \mu_\theta, x \sim \pi_\theta(\cdot|h)} f(h, x)) && (29)\\
&\leq T\epsilon && (30)
\end{aligned}$$

where the final inequality applies the assumption from the lemma. Note that this is similar in spirit to various results for error accumulation in imitation learning through distribution matching. For instance, [106] shows a bound in terms of *divergences* in occupancy measures, while [104] shows a bound in terms of *sample complexity*. Here, in order to motivate our perspective from the notion of expected quality difference, our bound is given with respect to the *moment-matching* discrepancy, and can similarly be interpreted as a non-Markovian variant of the "reward upper bound" in [105]. □

For Propositions 3 and 4, we use the fact that training the structured classifier (Definition 2) using the energy loss (Equation 11) amounts to a specific form of (sequence-wise) noise-contrastive estimation, and where the "noise" p_θ employed happens to be adaptively trained via the policy loss (Equation 12):

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: $p_s(\tau) > 0 \Rightarrow p_\theta(\tau) > 0$ (this does not require π_θ be optimal for f_ϕ). Then $\mathcal{L}_{\text{energy}}(\phi;\theta)$ is globally minimized at $F_\phi(\cdot) - \log Z_\phi = \log p_s(\cdot)$, whence p_ϕ is self-normalized with unit integral.

Proof. Briefly, a noise-contrastive estimator [79] operates as follows: Suppose we have some data $y \in \mathcal{Y}$ distributed as $p_{\text{data}}(y)$. Consider that we wish to learn a model distribution p_{model}, as follows:

$$p_{\text{model}}(y;a,b) \coloneqq \tilde{p}_{\text{model}}(y;a)\exp(b) \tag{31}$$

parameterized by a and b, where we emphasize that the model is not necessarily normalized as b is simply a learnable parameter. Also denote any noise distribution that can be sampled and evaluated:

$$p_{\text{noise}}(y;c) \tag{32}$$

parameterized by c. Now, define a classifier $d(\cdot\,;a,b,c)$ as follows, which we shall train to discriminate between p_{data} and p_{noise}—that is, given some y, to represent the (posterior) probability that it is real:

$$d(y;a,b,c) \coloneqq \sigma(\log p_{\text{model}}(y;a,b) - \log p_{\text{noise}}(y;c)) \tag{33}$$

where σ indicates the usual sigmoid function, i.e. $\sigma(u) \coloneqq 1/(1+\exp(-u))$ for any $u \in \mathbb{R}$. The noise contrastive estimator maximizes the likelihood of the parameters a, b in d given p_{data} and p_{noise}:

$$\mathcal{L}_{\text{class}}(a,b;c) \coloneqq -\mathbb{E}_{y\sim p_{\text{data}}}\log d(y;a,b,c) - \mathbb{E}_{y\sim p_{\text{noise}}}\log\big(1-d(y;a,b,c)\big) \tag{34}$$

In this optimization problem, a basic result is that $\mathcal{L}_{\text{class}}$ attains a minimum at $\log p_{\text{model}} = \log p_{\text{data}}$ and that there are no other minima if p_{noise} is chosen such that $p_{\text{data}}(y) > 0 \Rightarrow p_{\text{noise}}(y) > 0$ holds: see the "nonparametric estimation" result in [52]. Now, let us consider the following correspondence:

$$\big(\mathcal{Y}, p_{\text{data}}, \tilde{p}_{\text{model}}(\cdot\,;a), b, p_{\text{noise}}(\cdot\,;c)\big) \coloneqq \big(\mathcal{T}, p_s, \tilde{p}_\phi, -\log Z_\phi, p_\theta\big) \tag{35}$$

In other words, let the underlying space be that of trajectories $\mathcal{Y} \coloneqq \mathcal{T}$; let the data distribution be $p_{\text{data}} \coloneqq p_s$; let the model distribution be given by the un-normalized energy model $\tilde{p}_{\text{model}}(\cdot\,;a) \coloneqq \tilde{p}_\phi$ and partition function $b = -\log Z_\phi$; and let the noise distribution be given by rollouts of the policy, $p_{\text{noise}}(\cdot\,;c) \coloneqq p_\theta$. Then it is easy to see that the classifier and its loss function correspond as follows:

$$\begin{aligned} d_{\theta,\phi}(\cdot) &= d(\cdot\,;a,b,c) \\ \mathcal{L}_{\text{energy}}(\phi;\theta) &= \mathcal{L}_{\text{class}}(a,b;c) \end{aligned} \tag{36}$$

But then the optimality result above directly maps to the statement that $\mathcal{L}_{\text{energy}}$ is globally minimized at $F_\phi(\cdot) - \log Z_\phi = \log p_s(\cdot)$ assuming that the positivity condition $p_s(\tau) > 0 \Rightarrow p_\theta(\tau) > 0$ holds. Technicality: Note that here $\tilde{p}_\phi$ is constrained as $F_\phi(\tau) \coloneqq \sum_t f_\phi(h_t, x_t)$ instead of being arbitrarily parameterizable, but this does not affect realizability as we assumed that $F_s(\tau) \coloneqq \sum_t f_s(h_t, x_t)$. □

Proposition 4 (Asymptotic Consistency) Let ϕ^* denote the minimizer for $\mathcal{L}_{\text{energy}}(\phi;\theta)$, and let $\hat{\phi}^*_M$ denote the minimizer 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. Continuing the exposition above, let $\mathcal{L}^M_{\text{class}}(a,b;c)$ indicate the finite-data approximation of $\mathcal{L}_{\text{class}}(a,b;c)$—that is, by using M samples to approximate the true expectations over p_{data} and p_{noise}:

$$\mathcal{L}^M_{\text{class}}(a,b;c) \coloneqq -\tfrac{1}{M}\textstyle\sum_{m=1}^M \log d(y^{(m)}_{\text{data}};a,b,c) - \tfrac{1}{M}\sum_{m=1}^M \log\big(1-d(y^{(m)}_{\text{noise}};a,b,c)\big) \tag{37}$$

where the samples are drawn as $y^{(m)}_{\text{data}} \sim p_{\text{data}}$ and $y^{(m)}_{\text{noise}} \sim p_{\text{noise}}$. Consider the following conditions:

1. Positivity: $p_{\text{data}}(y) > 0 \Rightarrow p_{\text{noise}}(y) > 0$;
2. Uniform convergence: $\sup_{a,b} |\mathcal{L}^M_{\text{class}}(a, b; c) - \mathcal{L}_{\text{class}}(a, b; c)| \xrightarrow{p} 0$; and
3. The following matrix is full-rank: $\mathcal{I} \coloneqq \int g(y)g(y)^\top p_{\text{data}}(y)p_{\text{noise}}(y)/(p_{\text{data}}(y)+p_{\text{noise}}(y))dy$, where $g(y) \coloneqq \nabla_{(a,b)} \log p_{\text{model}}(y; a, b)|_{(a^*,b^*)}$ and a^*, b^* denote the optimal values of the model.

Note that (1) is same as before, and (2) and (3) are analogous to standard assumptions in maximum likelihood estimation. Let $\hat{a}^*_M, \hat{b}^*_M$ denote the minimizers for $\mathcal{L}^M_{\text{class}}(a, b; c)$. Under the preceding conditions, another basic result is that $(\hat{a}^*_M, \hat{b}^*_M)$ converges in probability to (a^*, b^*) as M grows: see the "consistency" result in [52]. But continuing the correspondence from before, it is easy to see that

$$\mathcal{L}^M_{\text{energy}}(\phi; \theta) = \mathcal{L}^M_{\text{class}}(a, b; c) \tag{38}$$

where we similarly define $\mathcal{L}^M_{\text{energy}}(\phi; \theta)$ to be the finite-data approximation of $\mathcal{L}_{\text{energy}}(\phi; \theta)$—that is, by using M samples to approximate the true expectations over p_s and p_θ, and $y_s^{(m)} \sim p_s$ and $y_\theta^{(m)} \sim p_\theta$:

$$\mathcal{L}^M_{\text{energy}}(\phi; \theta) \coloneqq -\tfrac{1}{M}\textstyle\sum_{m=1}^M \log d_{\theta,\phi}(\tau_s^{(m)}) - \tfrac{1}{M}\sum_{m=1}^M \log\big(1 - d_{\theta,\phi}(\tau_\theta^{(m)})\big) \tag{39}$$

which directly maps the above convergence result to the statement that as M increases $\hat{\phi}^*_M \xrightarrow{p} \phi^*$. □

Proposition 5 (Gradient Equality) Let ϕ_k be the value taken by ϕ after the k-th gradient update, and let θ^*_k denote the associated minimizer for $\mathcal{L}_{\text{policy}}(\theta; \phi_k)$. Suppose p_ϕ is already normalized; then

$$\nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta^*_k) = -\tfrac{T}{2}\nabla_\phi \mathcal{L}(\theta^*_k, \phi)$$

That is, at θ^*_k the energy gradient (of Equation 11) recovers the original gradient (from Equation 7). In the general case, suppose p_ϕ is un-normalized, such that $p_{\theta^*_k} = p_\phi / K_\phi$ for some constant K_ϕ; then

$$\nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta^*_k) = \tfrac{TK_\phi}{K_\phi+1}\mathbb{E}_{\substack{h\sim\mu_{\theta^*_k} \\ x\sim\pi_{\theta^*_k}(\cdot|h)}} \nabla_\phi f_\phi(h, x) - \tfrac{T}{K_\phi+1}\mathbb{E}_{\substack{h\sim\mu_s \\ x\sim\pi_s(\cdot|h)}} \nabla_\phi f_\phi(h, x)$$

Proof. From Equation 11,

$$\nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta^*_k) = \nabla_\phi\big(-\mathbb{E}_{\tau\sim p_s} \log d_{\theta^*_k,\phi}(\tau) - \mathbb{E}_{\tau\sim p_{\theta^*_k}} \log\big(1 - d_{\theta^*_k,\phi}(\tau)\big)\big) \tag{40}$$

$$= \nabla_\phi\big(-\mathbb{E}_{\tau\sim p_s} \log \frac{p_\phi(\tau)}{p_\phi(\tau) + p_{\theta^*_k}(\tau)} - \mathbb{E}_{\tau\sim p_{\theta^*_k}} \log \frac{p_{\theta^*_k}(\tau)}{p_\phi(\tau) + p_{\theta^*_k}(\tau)}\big) \tag{41}$$

$$= -\mathbb{E}_{\tau\sim p_s}\nabla_\phi(\log p_\phi(\tau) - \log(p_\phi(\tau) + p_{\theta^*_k}(\tau))) + \mathbb{E}_{\tau\sim p_{\theta^*_k}}\nabla_\phi \log(p_\phi(\tau) + p_{\theta^*_k}(\tau)) \tag{42}$$

$$= -\mathbb{E}_{\tau\sim p_s}\big[\nabla_\phi \log p_\phi(\tau) - \frac{\nabla_\phi p_\phi(\tau)}{p_\phi(\tau) + p_{\theta^*_k}(\tau)}\big] + \mathbb{E}_{\tau\sim p_{\theta^*_k}} \frac{\nabla_\phi p_\phi(\tau)}{p_\phi(\tau) + p_{\theta^*_k}(\tau)} \tag{43}$$

$$= -\mathbb{E}_{\tau\sim p_s}\big[\nabla_\phi \log p_\phi(\tau) - \frac{p_\phi(\tau)\nabla_\phi \log p_\phi(\tau)}{p_\phi(\tau) + p_{\theta^*_k}(\tau)}\big] + \mathbb{E}_{\tau\sim p_{\theta^*_k}} \frac{p_\phi(\tau)\nabla_\phi \log p_\phi(\tau)}{p_\phi(\tau) + p_{\theta^*_k}(\tau)} \tag{44}$$

$$= -\mathbb{E}_{\tau\sim p_s}\big[\nabla_\phi \log p_\phi(\tau) - \tfrac{1}{2}\nabla_\phi \log p_\phi(\tau)\big] + \tfrac{1}{2}\mathbb{E}_{\tau\sim p_{\theta^*_k}}\nabla_\phi \log p_\phi(\tau) \tag{45}$$

$$= \tfrac{1}{2}\mathbb{E}_{\tau\sim p_{\theta^*_k}}\nabla_\phi \log p_\phi(\tau) - \tfrac{1}{2}\mathbb{E}_{\tau\sim p_s}\nabla_\phi \log p_\phi(\tau) \tag{46}$$

$$= \tfrac{1}{2}\mathbb{E}_{\tau\sim p_{\theta^*_k}}\nabla_\phi(\log \tilde{p}_\phi(\tau) - \log Z_\phi) - \tfrac{1}{2}\mathbb{E}_{\tau\sim p_s}\nabla_\phi(\log \tilde{p}_\phi(\tau) - \log Z_\phi) \tag{47}$$

$$= \tfrac{1}{2}\big(\mathbb{E}_{\tau\sim p_{\theta^*_k}}\nabla_\phi \textstyle\sum_t f_\phi(h_t, x_t) - \mathbb{E}_{\tau\sim p_s}\nabla_\phi \sum_t f_\phi(h_t, x_t)\big) \tag{48}$$

$$= \tfrac{1}{2}\big(T\mathbb{E}_{h\sim\mu_{\theta^*_k}, x\sim\pi_{\theta^*_k}(\cdot|h)}\nabla_\phi f_\phi(h, x) - T\mathbb{E}_{h\sim\mu_s, x\sim\pi_s(\cdot|h)}\nabla_\phi f_\phi(h, x)\big) \tag{49}$$

$$= -\tfrac{T}{2}\nabla_\phi \mathcal{L}(\theta^*_k, \phi) \tag{50}$$

where the fourth and fifth lines repeatedly use the identity $\nabla_z p_z \equiv p_z \nabla_z \log p_z$ for any p_z parameterized by z, and the sixth line uses the fact that the current value of θ (i.e. θ^*_k) is the minimizer for $\mathcal{L}_{\text{policy}}(\theta; \phi)$ at the current value of ϕ (i.e. ϕ_k), hence it must be the case that $p_\theta = p_\phi$ at those values. Note that this assumes that p_ϕ is normalized; in practice this will be approximately true, for instance if we pre-train ϕ beforehand, using a fixed π_θ pre-trained by maximum likelihood (see Appendix B).

In the more general case of any arbitrary un-normalized p_ϕ, we only know $p_{\theta_k^*} = \frac{1}{K_\phi} p_\phi$ for some constant K_ϕ; then we recover a generalized "weighted" version of Equation 7. From the fifth line above,

$$\nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta_k^*) = -\mathbb{E}_{\tau \sim p_s}\big[\nabla_\phi \log p_\phi(\tau) - \frac{p_\phi(\tau)\nabla_\phi \log p_\phi(\tau)}{p_\phi(\tau) + p_{\theta_k^*}(\tau)}\big] + \mathbb{E}_{\tau \sim p_{\theta_k^*}} \frac{p_\phi(\tau)\nabla_\phi \log p_\phi(\tau)}{p_\phi(\tau) + p_{\theta_k^*}(\tau)} \quad (51)$$

$$= -\mathbb{E}_{\tau \sim p_s}\big[\nabla_\phi \log p_\phi(\tau) - \frac{K_\phi}{K_\phi+1}\nabla_\phi \log p_\phi(\tau)\big] + \frac{K_\phi}{K_\phi+1}\mathbb{E}_{\tau \sim p_{\theta_k^*}} \nabla_\phi \log p_\phi(\tau) \quad (52)$$

$$= \frac{K_\phi}{K_\phi+1}\mathbb{E}_{\tau \sim p_{\theta_k^*}} \nabla_\phi \log p_\phi(\tau) - \frac{1}{K_\phi+1}\mathbb{E}_{\tau \sim p_s} \nabla_\phi \log p_\phi(\tau) \quad (53)$$

$$= \frac{K_\phi}{K_\phi+1}\mathbb{E}_{\tau \sim p_{\theta_k^*}} \nabla_\phi (\log \tilde{p}_\phi(\tau) - \log Z_\phi) - \frac{1}{K_\phi+1}\mathbb{E}_{\tau \sim p_s} \nabla_\phi (\log \tilde{p}_\phi(\tau) - \log Z_\phi) \quad (54)$$

$$= \frac{K_\phi}{K_\phi+1}\mathbb{E}_{\tau \sim p_{\theta_k^*}} \nabla_\phi \textstyle\sum_t f_\phi(h_t, x_t) - \frac{1}{K_\phi+1}\mathbb{E}_{\tau \sim p_s} \nabla_\phi \sum_t f_\phi(h_t, x_t) \quad (55)$$

$$= \frac{TK_\phi}{K_\phi+1}\mathbb{E}_{h \sim \mu_{\theta_k^*}, x \sim \pi_{\theta_k^*}(\cdot|h)} \nabla_\phi f_\phi(h, x) - \frac{T}{K_\phi+1}\mathbb{E}_{h \sim \mu_s, x \sim \pi_s(\cdot|h)} \nabla_\phi f_\phi(h, x) \quad (56)$$

This "weighting" is intuitive: If p_ϕ is un-normalized such that $K_\phi > 1$, the energy loss automatically places higher weights on negative samples $h \sim \mu_{\theta_k^*}, x \sim \pi_{\theta_k^*}(\cdot|h)$ to bring it down; conversely, if p_ϕ is un-normalized such that $K_\phi < 1$, the energy loss places higher weights on positive samples $h \sim \mu_s, x \sim \pi_s(\cdot|h)$ to bring it up. (If p_ϕ is normalized, then $K_\phi = 1$ and the weights are equal). □

Note on Duality Lemmas 1 and 2 are for building intuition, and we are *not* implying that Equations 3 and 5 are direct generalizations of the duality between maximum likelihood and maximum entropy. To be clear, we shall explain Equation 5; Equation 3 is similar. Consider first the case of finite $\mathcal{X}$ (therefore finite $\mathcal{T}$). In the usual linear case, given basis functions $T(\tau)$, the optimization problem is:

$$\arg\min_\theta E_{\tau \sim p_\theta} \log p_\theta(\tau) \quad \text{s.t.} \quad E_{\tau \sim p_s} T(\tau) = E_{\tau \sim p_\theta} T(\tau) \quad (57)$$

Internalizing the constraint, we may write $\arg\min_\theta(E_{\tau \sim p_\theta} \log p_\theta(\tau) + \max_F(\langle F, E_{\tau \sim p_s} T(\tau)\rangle - \langle F, E_{\tau \sim p_\theta} T(\tau)\rangle))$. To generalize to the nonlinear case, let us now specifically define the feature vector $T(\tau)$ to be the indicator function (i.e. a finite-length vector, each zero-one entry of which corresponds to each element in $\mathcal{T}$). Then note that $\langle F, E_{\tau \sim p} T(\tau)\rangle = \langle F, p\rangle = E_{\tau \sim p} F(\tau)$, therefore:

$$\arg\min_\theta(E_{\tau \sim p_\theta} \log p_\theta(\tau) + \max_F(E_{\tau \sim p_s} F(\tau) - E_{\tau \sim p_\theta} F(\tau))) \quad (58)$$

Finally, to arrive at Equation 5 we use the fact that $E_{\tau \sim p} \sum_t f(h_t, x_t) = T E_{h \sim \mu, x \sim \pi(\cdot|h)} f(h, x)$, as already noted. All in all, this is a "generalization" from the case of linearity in *known* basis functions, to the case of *unknown* basis functions—where we "linearize" the expression using indicator functions. But we cannot use the same logic to claim the same for *infinite* $\mathcal{X}$ (which is the setting we operate in).

B Details on Algorithm

Policy Optimization Recall the policy update (Equation 12); this corresponds to entropy-regularized reinforcement learning using $f_\phi(h, x)$ as transition-wise reward function. Here we give a brief review of entropy-regularized reinforcement learning [45–47] in our context, as well as the practical method we employ (i.e. soft actor-critic). First, we introduce some standard notation. At any state h, define the (soft) "value function" to be the (forward-looking) expected sum of future rewards $f_\phi(h, x)$ as well as entropies $H(\pi(\cdot|h))$. Specifically, let $V_\phi^{\pi_\theta}(h)$ and $Q_\phi^{\pi_\theta}(h, x)$ be given as follows (we omit explicit notation for t, as any influence of time is implicit through dependence on variable-length histories):

$$V_\phi^{\pi_\theta}(h) \coloneqq \mathbb{E}_{\tau \sim p_\theta}[\textstyle\sum_{u=t}^T f_\phi(h_u, x_u) + H(\pi_\theta(\cdot|h_u))|h_t = h] \quad (59)$$

$$Q_\phi^{\pi_\theta}(h, x) \coloneqq f_\phi(h, x) + \mathbb{E}_{\tau \sim p_\theta}[\textstyle\sum_{u=t+1}^T f_\phi(h_u, x_u) + H(\pi_\theta(\cdot|h_u))|h_t = h, x_t = x] \quad (60)$$

Let π_{θ^*} denote the optimal policy (i.e. that minimizes loss $\mathcal{L}_{\text{policy}}$), and $Q_\phi^{\pi_{\theta^*}}$ its corresponding value function. An elementary result is that the optimal policy assigns probabilities to x proportional to the exponentiated expected returns of energy and entropy terms of all trajectories that begin with (h, x):

$$\pi_{\theta^*}(x|h) = \frac{\exp(Q_\phi^{\pi_{\theta^*}}(h, x))}{\int_{\mathcal{X}} \exp(Q_\phi^{\pi_{\theta^*}}(h, x))dx} \quad (61)$$

Now, for any transition policy π_θ (i.e. not necessarily optimal with respect to f_ϕ), the value function $Q_\phi^{\pi_\theta}$ is the unique fixed point of the following (soft) Bellman backup operator $\mathbb{B}_\phi^{\pi_\theta} : \mathbb{R}^{\mathcal{H} \times \mathcal{X}} \to \mathbb{R}^{\mathcal{H} \times \mathcal{X}}$:

$$(\mathbb{B}_\phi^{\pi_\theta} Q)(h,x) \coloneqq f_\phi(h,x) + \mathbb{E}_{x' \sim \pi_\theta(\cdot|h')}[Q(h',x') - \log \pi_\theta(x'|h')] \tag{62}$$

and hence—in theory—$Q_\phi^{\pi_\theta}$ may be computed iteratively by repeatedly applying the operator $\mathbb{B}_\phi^{\pi_\theta}$ starting from any function $Q \in \mathbb{R}^{\mathcal{H} \times \mathcal{X}}$; this is referred to as the (soft) "policy evaluation" procedure.

Using $Q_\phi^{\pi_\theta}$, we may then perform (soft) "policy improvement" to update the policy π_θ towards the exponential of its value function, and is guaranteed to result in an improved policy (in terms of $Q_\phi^{\pi_\theta}$):

$$\theta' \leftarrow \arg\min_\theta D_{\mathrm{KL}}\Big(\pi_{\theta'}(\cdot|h) \Big\| \frac{\exp(Q_\phi^{\pi_\theta}(h,\cdot))}{\int_{\mathcal{X}} \exp(Q_\phi^{\pi_\theta}(h,x))dx}\Big) \tag{63}$$

for all $h \in \mathcal{H}$. In theory, then, finding the optimal policy can be approached by repeatedly applying the above policy evaluation and policy improvement steps starting from any initial policy π_θ; this is referred to as (soft) "policy iteration". However, in large continuous domains (such as $\mathcal{H} \times \mathcal{X}$) doing this exactly is impossible, so we need to rely on function approximation for representing value functions.

Practical Algorithm Precisely, the soft actor-critic approach is to introduce a function approximator to represent the value function (i.e. the "critic") parameterized by ψ, in addition to the policy itself (i.e. the "actor") parameterized by θ, and to alternate between optimizing both with stochastic gradient descent [57]. Specifically, the actor performs soft policy improvement steps as before, but now using Q_ψ:

$$\mathcal{L}_{\text{actor}}(\theta;\phi,\psi) \coloneqq \mathbb{E}_{h \sim \mathcal{B}}\, \mathbb{E}_{x \sim \pi_\theta(\cdot|h)}[\log \pi_\theta(x|h) - Q_\psi(h,x)] \tag{64}$$

where $\mathcal{B}$ is a replay buffer of samples generated by π_θ (that is, instead of enumerating all $h \in \mathcal{H}$, we are relying on $h \sim \mathcal{B}$). Note that the normalizing constant is dropped as it does not contribute to the gradient. The critic is trained to represent the value function by minimizing squared residual errors:

$$\mathcal{L}_{\text{critic}}(\psi;\phi) \coloneqq \mathbb{E}_{h,x \sim \mathcal{B}}(Q_\psi(h,x) - Q_\psi^{\text{target}}(h,x))^2 \tag{65}$$

with (bootstrapped) targets:

$$Q_\psi^{\text{target}}(h,x) \coloneqq f_\phi(h,x) + \mathbb{E}_{x' \sim \pi_\theta(\cdot|h')}[Q_\psi(h',x') - \log \pi_\theta(x'|h')] \tag{66}$$

Together, this provides a way to minimize the policy loss $\mathcal{L}_{\text{policy}}$ (Equation 12). The complete TimeGCI algorithm simply alternates between this and minimizing the energy loss $\mathcal{L}_{\text{energy}}$ (Equation 11):

$$\mathcal{L}_{\text{energy}}(\phi;\theta) \coloneqq -\mathbb{E}_{\tau \sim p_s} \log d_{\theta,\phi}(\tau) - \mathbb{E}_{\tau \sim p_\theta} \log\big(1 - d_{\theta,\phi}(\tau)\big) \tag{67}$$

Hence in Algorithm 1, gradient updates for the energy, policy, and critic are interleaved with policy rollouts. Note that several standard approximations are being used. First, the replay buffer provides samples $h \sim \mathcal{B}$ for optimizing the policy (in both actor and critic updates), instead of covering the entire space $\mathcal{H}$ (which is uncountable). Second, in the energy loss negative samples $\tau \sim \mathcal{B}$ are used in lieu of sampling fresh from p_θ at every iteration (this is known to give the benefit of providing more diverse negative samples). Finally, also per usual samples from the dataset $\tau \sim \mathcal{D}$ is used in lieu of p_s.

Practical Considerations First, in practice we must use a *vector representation* of histories $h \in \mathcal{H}$; here we use RNNs to encode histories into fixed-length vectors, which can then be treated as regular "states" in continuous space. Like our choice of policy optimization, this is also an arbitrary design choice—we could just as conceivably have used e.g. temporal convolutions, attention mechanisms, etc.

Second, *interleaving* multiple gradient updates of different networks requires some care: In soft actor-critic itself, policy updates have to be sufficiently small, and/or critic updates have to be sufficiently frequent, to prevent divergence. The situation is analogous when interleaving this with energy gradient updates as well: Both actor and energy updates have to be sufficiently small, and/or critic updates have to be sufficiently frequent. That said, the energy updates are indeed decoupled from the policy updates: Regardless of how quickly/slowly the policy is learning, the energy can learn on their negative samples. In practice, we perform multiple critic updates for every update of the policy and energy functions.

Finally, note that in large continuous domains such as $\mathcal{H} \times \mathcal{X}$ it is necessary to *pre-train* the networks beforehand such that optimization of the complete algorithm actually converges: On the one hand, the policy side requires a sufficiently good energy signal to actually make progress, and on the other hand, the energy side requires a sufficiently good policy providing challenging enough negative samples to actually make progress. Pre-training networks separately is standard in actor-critic methods (see for instance [35]); here we take a similar approach but with the addition of the energy update step as well:

1. Policy-only: π_θ is pre-trained using maximum likelihood;
2. Energy-only: f_ϕ is pre-trained using $\mathcal{L}_{\text{energy}}(\phi; \theta)$, holding π_θ fixed;
3. Critic-only: Q_ψ is pre-trained using $\mathcal{L}_{\text{critic}}(\psi; \phi)$, holding π_θ, f_ϕ fixed; and finally,
4. All: f_ϕ, π_θ, and Q_ψ are trained on $\mathcal{L}_{\text{energy}}(\phi; \theta)$, $\mathcal{L}_{\text{actor}}(\theta; \phi, \psi)$, and $\mathcal{L}_{\text{critic}}(\psi; \phi)$ (cf. Algorithm 1).

C Details on Experiments

Benchmark Algorithms Except where components are standardized (see below), we use the publicly available source code when constructing the benchmark algorithms; references are in the following:

- T-Forcing [5]: (straightforward MLE with ground-truth conditioning)
- P-Forcing [10]: `https://github.com/anirudh9119/LM_GANS`
- C-RNN-GAN [18]: `https://github.com/olofmogren/c-rnn-gan`
- COT-GAN [20]: `https://github.com/tianlinxu312/cot-gan`
- RC-GAN [21]: `https://github.com/ratschlab/RGAN`
- TimeGAN [12]: `https://github.com/jsyoon0823/TimeGAN`

Dataset Sources We use the original source code for preprocessing sines and UCI datasets from TimeGAN (`https://github.com/jsyoon0823/TimeGAN`). For MIMIC-III, we extract 52 clinical covariates including vital signs (e.g. respiratory rate, heart rate, O2 saturation) and lab tests (e.g. glucose, hemoglobin, white blood cell count) aggregated every hour during their ICU stay up to 24 hours.

- Sines [5]: `https://github.com/jsyoon0823/TimeGAN`
- Energy [10]: `archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction`
- Gas [18]: `archive.ics.uci.edu/ml/datasets/Gas+sensor+array+temperature+modulation`
- Metro [20]: `archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume`
- MIMIC-III [21]: `https://physionet.org/content/mimiciii/1.4/`

Encoder Networks For fair comparison, analogous network components across all benchmarks share the same architecture where possible. In particular, all components taking h_t as input require an encoder network to construct fixed-length vector representations of variable-length histories $(x_1, ..., x_t)$. To do so, these components use an LSTM network with a hidden layer of size 32 to compute hidden states for representing h. These are not shared: separate components have their own encoder networks.

Task-Specific Networks Then, for mapping from h_t and/or x_t to task-specific output variables, we use a fully-connected network with two hidden layers of size 32 and ELU activations. Where both h_t and x_t serve as inputs, their vectors are concatenated. For instance, the energy network for TimeGCI contains one such network for computing f_ϕ (as well as trainable parameter for Z_ϕ). The same applies to the mapping from h_t and/or x_t to implicit generator outputs (as in C-RNN-GAN and RC-GAN), black-box discriminator output (as in P-Forcing, C-RNN-GAN, and RC-GAN), transition policies (as in T-Forcing and P-Forcing), and critic values (for TimeGCI). Note that TimeGAN and COT-GAN were designed with additional unique components/losses that operate as a unit (e.g. the embedding/recovery networks for generating/discriminating within latent space); we use their original source code.

Replay Buffer The replay buffer $\mathcal{B}$ has a fixed size; once filled, new samples stored replace the oldest still in the buffer. Sampling from the buffer operates as follows: In updating the energy, we require $\tau \sim \mathcal{B}$ (that is, in lieu of p_θ, cf. Equation 67); this is done by randomly sampling a batch of trajectories from the replay buffer, without replacement. In policy the actor, we require $h \sim \mathcal{B}$ (cf. Equation 64); this is done by first randomly sampling a batch of trajectories from the replay buffer, and then randomly sampling a cutoff time t to obtain a batch of subsequences h_t. Finally, in updating the critic, we require $h, x \sim \mathcal{B}$ (cf. Equation 65); this is similarly done by first randomly sampling a batch of trajectories from the replay buffer, then randomly sampling a cutoff t to yield a batch of (h_t, x_t) pairs.

Hyperparameters In all experiments, we use the following hyperparameters for TimeGCI and for benchmarks (wherever applicable): The replay buffer is of size $|\mathcal{B}| = 10,000$. The hidden dimension of all encoder and task-specific networks is 32. The entropy regularization (in the actor/policy loss) is $\alpha = 0.2$. The policy network is pre-trained for 2,000 steps, energy for 4,000, and critic for 20,000. The complete algorithm is trained for up to 50,000 steps with checkpointing and early stopping (triggerable every 1,000 iterations, if performance does not improve). The Adam optimizer is used for all losses, and batch size $M = 64$. Learning rates: For energy networks $\lambda_{\text{energy}} = 0.0001$, for policy networks $\lambda_{\text{policy}} = 0.0001$ (same for implicit generator networks), for critic networks $\lambda_{\text{critic}} = 0.001$, and for black-box discriminator networks $\lambda_{\text{discrim}} = 0.001$. (Note that the critic/discriminator networks are

updated more greedily). Per usual in soft actor-critic algorithms, we also employ a lagged target critic network (i.e. used for bootstrapping); this is updated using polyak averaging at a rate of $\tau = 0.005$.

Performance Metrics We use the original source code for computing the TSTR metric (i.e. Predictive Score), publicly available at: `https://github.com/jsyoon0823/TimeGAN`; this is straightforwardly modified to compute TSTR scores for horizons of lengths three (+3 Steps Ahead) and five (+5 Steps Ahead). Likewise, we use the original source code for computing the cross-correlation score (x-Corr. Score), this is also publicly available at: `https://github.com/tianlinxu312/cot-gan`.

D Clarifying the Analogy

This section clarifies the analogy of "generation as imitation". While the point of our investigation is to explicitly invite an analogy between synthetic time-series generation and imitation learning, they are different problems with different considerations. In particular, TimeGCI is not suitable for imitation learning per se, and we make no claim that it is (Appendix D.1). Conversely, while existing adversarial imitation learning methods may be naively applied to time-series generation, their optimization objectives are different, and we empirically verify they do not perform well (Appendix D.2).

D.1 Can TimeGCI be used for Imitation Learning?

At its core, time-series generation is the problem of modeling a distribution of trajectories $p(\tau)$ faithfully, and is what TimeGCI does. In MDP parlance, in time-series generation the "environment dynamics" are by construction *known* and *deterministic*: $\omega(\cdot|z_t, u_t)$ is the Dirac delta centered at $z_{t+1} = (u_1, \ldots, u_t)$. We have $p(\tau) = \prod_t \pi(u_t|z_t)$. However, in imitation learning the dynamics are generally *unknown* and *stochastic*. We have $p(\tau) = \prod_t \pi(u_t|z_t)\omega(z_t|z_{t-1}, u_{t-1})$. This difference is crucial. Consider walking through the logic of Section 3, but applying it to imitation learning instead. Begin with the gradient update for learning the Gibbs parameters (Equation 7). To avoid costly inner-loop optimization of θ to completion, we again consider (1) importance sampling, as well as our preferred method of (2) contrastive estimation. We will see that in imitation learning neither of these work out of the box. For (1), the importance weights now become $\exp(F_\phi(\tau))/p_\theta(\tau)$, where $p_\theta(\tau) = \prod_t \pi_\theta(u_t|z_t)\omega(z_t|z_{t-1}, u_{t-1})$. But the problem is that we do not have access to ω. We can naively *estimate* it beforehand (clearly inadvisable). Or, we can perform a specific *approximation* in the energy model: $p_\phi(\tau) = \exp(F_\phi(\tau) - \log Z_\phi) \approx \frac{1}{Z_{\phi,\omega}} \exp(F_\phi(\tau)) \prod_t \omega(z_t|z_{t-1}, u_{t-1})$ such that ω cancels out from the expression. Instead of modeling the distribution of trajectories $p(\tau)$ exactly, this now assumes that transition randomness has a limited effect on behavior and that the partition function is constant for all random outcome samples (see e.g. [67]). The situation is similar for (2), where it is easy to see that in imitation learning scenarios Equation 10 is no longer accessible without performing the above approximation to cancel out ω. Three points bear emphasis. First, if we are willing to make this simplification, we are no longer speaking of TimeGCI anymore. Instead, it is easy to show that this effectively becomes a sort of trajectory-centric "maximum *causal* entropy" inverse optimal control (see e.g. [39]), which is no longer learning $p(\tau)$ with an exact objective. Second, in general we have little reason to believe that such a trajectory-centric approach would work well in imitation learning: The variance of sample-based estimates is now exacerbated by the unknown and stochastic environment dynamics in imitation learning, such that trajectory-based sampling is generally known to perform quite poorly (see e.g. [54]). In fact, for this reason modern adversarial imitation learning methods almost always take a transition-centric approach. Third, empirically a similar structured-classifier approach (with the above approximation) has indeed been found to be unworkable in practice due to high variance (see [54] Section 4, implementing [53]); moreover, an importance-sampling based approach (with the above approximation) has been made to work, but has required hand-crafted, domain-specific regularization to work, and the learned energy functions only explain the demonstrations locally [68]. For these reasons, we do not evaluate TimeGCI on imitation learning scenarios. It is simply not applicable without modifying it to become a different method altogether. Modern imitation learning methods are designed specifically to handle the unknown and stochastic nature of environment transitions. In particular, transition-centric approaches (i.e. scoring (z, u) pairs) have been found to be more effective empirically. What we are doing with TimeGCI, on the other hand, is to focus on an exact objective for learning time-series trajectories $p(\tau)$, which requires a trajectory-centric approach. This is because this allows us to equate TimeGCI with a special kind of noise-contrastive estimation, for which we can enjoy some theoretical guarantees. (Note that these properties are lost when performing the above approximation in an imitation learning setting).

D.2 Can AIL be used for Time-series Generation?

Table 4: *Performance Comparison of GAIL, AIRL, and TimeGCI*. Bold numbers indicate best-performing results.

Benchmark	*Metric*	Sines	Energy	Gas	Metro	MIMIC-III
GAIL-SAC	Predictive Score	0.447 ± 0.002	0.261 ± 0.001	0.022 ± 0.002	0.257 ± 0.001	0.017 ± 0.001
	+3 Steps Ahead	0.472 ± 0.003	0.262 ± 0.001	0.048 ± 0.001	0.250 ± 0.001	0.012 ± 0.001
	+5 Steps Ahead	0.542 ± 0.004	0.261 ± 0.001	0.074 ± 0.002	0.245 ± 0.003	0.011 ± 0.001
	x-Corr. Score	6.798 ± 0.014	142.457 ± 0.471	111.124 ± 0.232	1.044 ± 0.031	540.89 ± 0.159
AIRL-SAC	Predictive Score	0.223 ± 0.006	0.283 ± 0.002	0.0478 ± 0.0015	0.243 ± 0.001	0.018 ± 0.005
	+3 Steps Ahead	0.381 ± 0.002	0.295 ± 0.003	0.0883 ± 0.0029	0.250 ± 0.001	0.019 ± 0.003
	+5 Steps Ahead	0.349 ± 0.005	0.321 ± 0.001	0.1176 ± 0.0040	0.251 ± 0.002	0.019 ± 0.001
	x-Corr. Score	10.397 ± 0.005	202.61 ± 0.119	144.79 ± 0.3199	1.286 ± 0.098	2635.57 ± 0.050
TimeGCI	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	$\mathbf{1.195 \pm 0.011}$	$\mathbf{105.2 \pm 0.433}$	$\mathbf{47.91 \pm 0.811}$	$\mathbf{0.738 \pm 0.019}$	$\mathbf{194.3 \pm 0.180}$

Conversely to the preceding, we can also ask: Can adversarial imitation learning methods work for time-series generation (hence can we compare them against TimeGCI)? The answer is "yes", they can be *applied* to time-series generation, but "no", there is no reason to expect they would perform better. Modern adversarial imitation learning methods come in two broad flavors: (1) GAIL-based [107–110], which do not recover a reward function, and (2) AIRL-based [54, 69, 111], which do recover a reward function. Both are transition-centric (i.e. scoring (z, u) pairs). For (1): Here, optimization is of the familiar GAN-like objective: $\min_\theta \max_\phi \mathbb{E}_{z,u\sim\mu_s} \log d_\phi(z,u) + \mathbb{E}_{z,u\sim\mu_\theta} \log\big(1 - d_\phi(z,u)\big)$, which minimizes the JS-divergence between the *state-action* occupancy measures (i.e. distribution of transitions, in time-series generation terms) induced by the learned and source policies. Note that this is the same as the TimeGAN objective [12], modulo entropic/supervised regularization. As noted before, matching the transition marginals is indirectly performing the "global" sort of moment-matching. However, as pertains time-series generation, the key difference between GAIL and TimeGAN is that the former performs policy optimization for the generator, whereas the latter simply performs backpropagation. Otherwise, note that both methods are adversarial (viz. saddle-point optimization) and do not learn an explicit energy (viz. black-box discriminator). In light of this, we may consider GAIL for experimental comparison (to be consistent, we also use SAC as the policy optimization method); see Table 4 above. For (2): Here, the discriminator is first constructed to be *transition*-based: $d_{\theta,\phi}(z,u) = (\exp(g_\phi(z,u))/(\exp(g_\phi(z,u)) + \pi_\theta(u|z)))$. Note that this actually breaks the relationship with modeling the distribution of *trajectories* directly $p_\phi(\tau) = \exp(F_\phi(\tau) - \log Z_\phi)$. Importantly, if we apply AIRL to time-series generation, we lose the convergence property that comes from the relationship with noise-contrastive estimation (cf. Proposition 4), and the gradient of the discriminator is no longer related to the original trajectory-wise energy-based gradient (cf. Proposition 5). In fact, all we are left with is that the global optimum is a correct optimum—but this by itself is not helpful (e.g. the global optimum of naive MLE is also a correct optimum, but there is no guarantee that it can be found effectively). In fact, it has been formally shown that the original theoretical justifications for AIRL are incorrect (see e.g. [111] Sec. 2.4.2, or [112] Sec. B.2). (In stochastic domains in imitation learning, it is true that the empirical benefit of variance-reduction from sampling in this transition-based manner still dominates. In time-series generation, however, there is no stochasticity in the "environment", and doing it this way may unnecessarily bias our objective of learning $p(\tau)$). In light of this, we may consider AIRL for experimental comparison (to be consistent, we also use SAC for policy optimization); see Table 4.

D.3 Ablation Studies on Sequence Lengths

First, we compare the performance of T-Forcing to TimeGCI, using sequence lengths $T \in \{2, 8, 24\}$. We compute the TSTR predictive score, +3 steps ahead, +5 steps ahead, and x-Corr. score as usual for each of these settings. Note that for $T = 2$ we cannot ask the TSTR evaluation model to predict more than one step ahead, since there is no data for more than one step ahead; we indicate this with "N/A". The Gas dataset is used. For ease of interpretation, we consider T-Forcing as the "baseline", and also express the TimeGCI numbers as a fraction of the T-Forcing numbers. The results are consistent with what we would expect: The performance advantage enjoyed by TimeGCI over T-Forcing diminishes as the sequence lengths of the input dataset decreases, and increases as the sequence lengths of the input dataset increases. This is true regardless of what metric we are talking about (i.e. 1/3/5-step TSTR score, or feature correlation score). Moreover, at $T = 2$ there is almost no difference between

the TSTR scores of T-Forcing and TimeGCI. This shows that T-Forcing appears to preserve quite well the one-step ahead relationships of the original data (viz. $T = 2$). However, for longer sequences TimeGCI performs relatively better, which is consistent with the motivation behind using an objective that matches the distribution of trajectories, instead of simply matching transition conditionals:

Benchmark	*Metric*	$T = 2$	$T = 8$	$T = 24$
T-Forcing	Predictive Score	0.038 ± 0.001	0.036 ± 0.007	0.035 ± 0.003
	+3 Steps Ahead	N/A	0.076 ± 0.007	0.080 ± 0.001
	+5 Steps Ahead	N/A	0.114 ± 0.003	0.111 ± 0.001
	x-Corr. Score	160.3 ± 0.346	154.2 ± 0.219	150.8 ± 0.067
TimeGCI	Predictive Score	0.037 ± 0.001	0.022 ± 0.000	0.018 ± 0.000
	+3 Steps Ahead	N/A	0.048 ± 0.001	0.042 ± 0.001
	+5 Steps Ahead	N/A	0.071 ± 0.001	0.067 ± 0.001
	x-Corr. Score	140.4 ± 0.173	50.72 ± 0.816	47.91 ± 0.811
Ratio of Mean	Predictive Score	97.37%	61.11%	51.43%
	+3 Steps Ahead	N/A	63.16%	52.50%
	+5 Steps Ahead	N/A	62.28%	60.36%
	x-Corr. Score	87.59%	32.89%	31.77%

Second, we use a new synthetic simulation where what is being generated is a set of multi-dimensional sinusoidal waves which—if not perturbed by noise—correspond to different frequencies, phases, etc. This allows us to manually inject noise where we want, and also allows us to simulate "ground truth". First, we train both models (T-Forcing and TimeGCI) using the same training data generated by the simulator, and save these learned models. Then during validation, we obtain a sequence from the simulator, then add some additional noise. Specifically, we add independent Gaussian noise $\mathcal{N}(\mu, \sigma)$, using $\mu = 0$ and $\sigma = 0.1$, to each of the feature dimensions of the wave at time K. Then, our task is to predict t steps ahead with the learned models. To be clear, we ask the learned models to predict ahead using the information up to and including this (perturbed) K-th step; if the prediction is for $t > 1$, we perform open-loop sampling as usual. We can measure the model's prediction error (i.e. evaluating how well the model forecasts the future based on the provided histories as input, along with the perturbation). Note that this is entirely different from the TSTR predictive score hitherto used (i.e. evaluating how well the model's freely generated output dataset preserves the characteristics of the input dataset); this is only done as an additional sensitivity analysis: By analogy to imitation learning, we can interpret this experiment as a special case of "action-matching" (i.e. based on ground-truth state/history as input, we can measure the "prediction error" between the action chosen by the imitator policy and the actual action taken by the expert)—but specifically where the state/history has been perturbed with additional noise. We ask each model to perform t-steps ahead prediction, where $t \in \{1, 2, 3, 4, 5\}$ as a sensitivity. When we add noise, we use values $\{\sigma, 2\sigma, 3\sigma, 4\sigma, 5\sigma\}$ as a sensitivity. Given any sequence, the step K at which noise is artificially added (and at which point the models are asked to perform t-step ahead predictions) is uniformly randomly sampled from $K \in \{1, 2, ..., T-1\}$.

Noise at K	*Benchmark*	1-Ahead MSE	2-Ahead MSE	3-Ahead MSE	4-Ahead MSE	5-Ahead MSE
σ	T-Forcing	0.024 ± 0.000	0.029 ± 0.001	0.036 ± 0.001	0.041 ± 0.001	0.047 ± 0.001
	TimeGCI	0.024 ± 0.000	0.025 ± 0.001	0.026 ± 0.001	0.028 ± 0.001	0.031 ± 0.001
2σ	T-Forcing	0.055 ± 0.000	0.059 ± 0.001	0.065 ± 0.001	0.071 ± 0.002	0.076 ± 0.001
	TimeGCI	0.055 ± 0.001	0.055 ± 0.001	0.057 ± 0.001	0.059 ± 0.000	0.061 ± 0.001
3σ	T-Forcing	0.106 ± 0.001	0.109 ± 0.002	0.116 ± 0.001	0.121 ± 0.001	0.127 ± 0.002
	TimeGCI	0.106 ± 0.001	0.106 ± 0.001	0.107 ± 0.001	0.109 ± 0.001	0.111 ± 0.001
4σ	T-Forcing	0.176 ± 0.001	0.181 ± 0.002	0.185 ± 0.001	0.191 ± 0.002	0.197 ± 0.002
	TimeGCI	0.176 ± 0.002	0.176 ± 0.002	0.177 ± 0.001	0.179 ± 0.002	0.181 ± 0.001
5σ	T-Forcing	0.266 ± 0.003	0.271 ± 0.002	0.277 ± 0.002	0.280 ± 0.002	0.287 ± 0.002
	TimeGCI	0.266 ± 0.002	0.267 ± 0.003	0.267 ± 0.002	0.270 ± 0.003	0.272 ± 0.003

The results give some orthogonal intuition as to why the proposed method is better; specifically, is it because of better first-step prediction, or is it because of better robustness when having small errors in the previous steps? Observe that for *single*-step prediction, there is virtually no difference between the performance of T-Forcing and TimeGCI when evaluating the prediction MSE. The more noise added, the worse both models perform; but there is little difference between them, no matter the noise. On the other hand, for *multi*-step prediction, when predicting multiple steps ahead using open-loop sampling, T-Forcing performs worse than TimeGCI. In fact, the gap between their performances increases as t increases. So it appears that it is not the case that TimeGCI simply has better first-step prediction per se; also, it appears that TimeGCI does have better robustness when having errors in the previous steps. Because, while the 1-step performance of both T-Forcing and TimeGCI are impacted almost equally, TimeGCI appears to have an advantage in terms of "propagating" less of that error into later time steps.

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] James Jordon, Daniel Jarrett, Evgeny Saveliev, Jinsung Yoon, Paul Elbers, Patrick Thoral, Ari Ercole, Cheng Zhang, Danielle Belgrave, and Mihaela van der Schaar. Hide-and-seek privacy challenge: Synthetic data generation vs. patient re-identification. *NeurIPS 2020 Competition and Demonstration Track*, 2021.

[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] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised as supervised learning. *The Elements of Statistical Learning*, 2009.

[51] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[52] 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.

[53] 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.

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

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

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

[57] 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.

[58] 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.

[59] 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.

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

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

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

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

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

[65] Ioana Bica, Daniel Jarrett, Alihan Hüyük, and Mihaela van der Schaar. Learning what-if explanations for sequential decision-making. *International Conference on Learning Representations (ICLR)*, 2021.

[66] Daniel Jarrett, Alihan Hüyük, and Mihaela Van Der Schaar. Inverse decision modeling: Learning interpretable representations of behavior. *International Conference on Machine Learning (ICML)*, 2021.

[67] 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.

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

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

[70] 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.

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

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

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

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

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

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

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

[78] 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.

[79] 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.

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

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

[82] 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.

[83] 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.

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

[85] 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.

[86] 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.

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

[88] 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.

[89] 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.

[90] 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.

[91] 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.

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

[93] 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.

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

[95] 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.

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

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

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

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

[100] 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.

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

[102] 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.

[103] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. *International Conference on Machine Learning (ICML)*, 2002.

[104] Alekh Agarwal, Nan Jiang, and Sham M Kakade. Reinforcement learning: Theory and algorithms. 2019.

[105] Gokul Swamy, Sanjiban Choudhury, Steven Wu, and Andrew Bagnell. Of moments and matching tradeoffs and treatments in imitation learning. *International Conference on Machine Learning (ICML)*, 2021.

[106] Tian Xu, Ziniu Li, and Yang Yu. Error bounds of imitating policies and environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[107] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems (NeurIPS)*, 2016.

[108] Nir Baram, Oron Anschel, and Shie Mannor. Model-based adversarial imitation learning. *International Conference on Machine Learning (ICML)*, 2017.

[109] Wonseok Jeon, Seokin Seo, and Kee-Eung Kim. A bayesian approach to generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[110] Xin Zhang, Yanhua Li, Ziming Zhang, and Zhi-Li Zhang. f-gail: Learning f-divergence for generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[111] Oleg Arenz and Gerhard Neumann. Non-adversarial imitation learning and its connections to adversarial methods. *arXiv preprint*, 2020.

[112] Tianwei Ni, Harshit Sikchi, Yufei Wang, Tejus Gupta, Lisa Lee, and Benjamin Eysenbach. F-irl: Inverse reinforcement learning via state marginal matching. *Conference on Robot Learning (CoRL)*, 2020.