Errata

The original manuscript contained some typographical errors. None affect the substance of the claims:

E.1 On lines 222, 223, and 235, the word "maximizer" should be replaced by "minimizer" instead.

E.2 On line 247, the signature "$Q_\psi : \mathcal{H} \times \mathcal{X} \to [-kT, kT]$" should read "$Q_\psi : \mathcal{H} \times \mathcal{X} \to \mathbb{R}$" instead.

E.3 On line 334, for LSTMs, "ELU activations" should be replaced by "tanh activations" instead.

E.4 Definition 2 and Proposition 2 erroneously omitted the constant Z_ϕ. See the correction in green:

> **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* $d_{\theta,\phi} : \mathcal{T} \to [0, 1]$:
>
> $$d_{\theta,\phi}(\tau) \coloneqq \frac{\frac{1}{Z_\phi}\tilde{p}_\phi(\tau)}{\frac{1}{Z_\phi}\tilde{p}_\phi(\tau) + p_\theta(\tau)} \tag{10}$$

> **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.

E.5 The conclusion of Proposition 5 is missing a constant factor of $-T$. See the correction in green:

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

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 $f_s : \mathcal{H} \times \mathcal{X} \to [-k, k]$ for some finite k. 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 $V^{\pi_\theta}_{s,t}(h) : \mathcal{H} \to [-kT, kT]$ and $Q^{\pi_\theta}_{s,t}(h, x) : \mathcal{H}\times\mathcal{X} \to [-kT, kT]$ be given as follows:

$$\begin{aligned} V^{\pi_\theta}_{s,t}(h) &\coloneqq \mathbb{E}_{\tau\sim p_\theta}[\Sigma^T_{u=t} f_s(h_u, x_u)|h_t = h] \\ Q^{\pi_\theta}_{s,t}(h, x) &\coloneqq \mathbb{E}_{\tau\sim p_\theta}[\Sigma^T_{u=t} f_s(h_u, x_u)|h_t = h, x_t = x] \end{aligned} \tag{13}$$

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,

$$\begin{aligned} \Delta\bar{F}_s(\theta) &= \mathbb{E}_{\tau\sim p_s} \textstyle\sum_t f_s(h_t, x_t) - \mathbb{E}_{\tau\sim p_\theta} \sum_t f_s(h_t, x_t) && (14)\\ &= \mathbb{E}_{\tau\sim p_s} \textstyle\sum_t (f_s(h_t, x_t) + V^{\pi_\theta}_{s,t}(h_t) - V^{\pi_\theta}_{s,t}(h_t)) - \mathbb{E}_{\tau\sim p_\theta} \sum_t f_s(h_t, x_t) && (15)\\ &= \mathbb{E}_{\tau\sim p_s} \textstyle\sum_t (f_s(h_t, x_t) + V^{\pi_\theta}_{s,t+1}(h_{t+1}) - V^{\pi_\theta}_{s,t}(h_t)) && (16)\\ &= \mathbb{E}_{\tau\sim p_s} \textstyle\sum_t (Q^{\pi_\theta}_{s,t}(h_t, x_t) - V^{\pi_\theta}_{s,t}(h_t)) && (17)\\ &= T\mathbb{E}_{h\sim\mu_s, x\sim\pi_s(\cdot|h)}(Q^{\pi_\theta}_{s,t}(h, x) - V^{\pi_\theta}_{s,t}(h)) && (18) \end{aligned}$$

where the third line telescopes terms and uses the fact $V^{\pi}_{s,T+1}(h) = 0$. This derivation can be interpreted as a non-stationary, non-Markovian analogue of the "performance difference" result in [99]. □

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

Proof. From Lemma 7,

$$\begin{aligned}
\Delta\bar{F}_s(\theta) &= T\mathbb{E}_{h\sim\mu_s, x\sim\pi_s(\cdot|h)} Q^{\pi_\theta}_{s,t}(h,x) - T\mathbb{E}_{h\sim\mu_s, x\sim\pi_\theta(\cdot|h)} Q^{\pi_\theta}_{s,t}(h) && (19)\\
&= T\mathbb{E}_{h\sim\mu_s}[\mathbb{E}_{x\sim\pi_s(\cdot|h)} Q^{\pi_\theta}_{s,t}(h,x) - \mathbb{E}_{x\sim\pi_\theta(\cdot|h)} Q^{\pi_\theta}_{s,t}(h,x)] && (20)\\
&\le \max_{f\in[-kT,kT]^{\mathcal{H}\times\mathcal{X}}} T\mathbb{E}_{h\sim\mu_s}[\mathbb{E}_{x\sim\pi_s(\cdot|h)} f(h,x) - \mathbb{E}_{x\sim\pi_\theta(\cdot|h)} f(h,x)] && (21)\\
&\le \max_{f\in\mathbb{R}^{\mathcal{H}\times\mathcal{X}}} T\mathbb{E}_{h\sim\mu_s}[\mathbb{E}_{x\sim\pi_s(\cdot|h)} Tf(h,x) - \mathbb{E}_{x\sim\pi_\theta(\cdot|h)} Tf(h,x)] && (22)\\
&\le T^2\epsilon && (23)
\end{aligned}$$

where the final inequality applies the assumption from the lemma. For a similar derivation, note that this can be interpreted as a non-Markovian version of the "off-policy upper bound" result in [100]. □

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

Proof. From Definition 1,

$$\begin{aligned}
\Delta\bar{F}_s(\theta) &= \mathbb{E}_{\tau\sim p_s} \textstyle\sum_t f_s(h_t,x_t) - \mathbb{E}_{\tau\sim p_\theta} \sum_t f_s(h_t,x_t) && (24)\\
&= T\mathbb{E}_{h\sim\mu_s,\pi_s(\cdot|h)} f_s(h,x) - T\mathbb{E}_{h\sim\mu_\theta,\pi_\theta(\cdot|h)} f_s(h,x) && (25)\\
&\le \max_{f\in[-k,k]^{\mathcal{H}\times\mathcal{X}}} (T\mathbb{E}_{h\sim\mu_s,\pi_s(\cdot|h)} f(h,x) - T\mathbb{E}_{h\sim\mu_\theta,\pi_\theta(\cdot|h)} f(h,x)) && (26)\\
&\le \max_{f\in\mathbb{R}^{\mathcal{H}\times\mathcal{X}}} (T\mathbb{E}_{h\sim\mu_s,\pi_s(\cdot|h)} f(h,x) - T\mathbb{E}_{h\sim\mu_\theta,\pi_\theta(\cdot|h)} f(h,x)) && (27)\\
&\le T\epsilon && (28)
\end{aligned}$$

where the final inequality applies the assumption from the lemma. Likewise for imitation learning, this derivation can be viewed as a non-Markovian analogue of the "reward upper bound" in [100]. □

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 [75] 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:

$$p_{\text{model}}(y;a,b) \coloneqq \tilde{p}_{\text{model}}(y;a)\exp(b) \tag{29}$$

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:

$$p_{\text{noise}}(y;c) \tag{30}$$

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:

$$d(y;a,b,c) \coloneqq \sigma(\log p_{\text{model}}(y;a,b) - \log p_{\text{noise}}(y;c)) \tag{31}$$

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}:

$$\mathcal{L}_{\text{class}}(a,b;c) \coloneqq -\mathbb{E}_{y\sim p_{\text{data}}} \log d(y;a,b,c) - \mathbb{E}_{y\sim p_{\text{noise}}} \log\left(1 - d(y;a,b,c)\right) \tag{32}$$

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 [50]. Now, let us consider the following correspondence:

$$\left(\mathcal{Y}, p_{\text{data}}, \tilde{p}_{\text{model}}(\cdot\,;a), b, p_{\text{noise}}(\cdot\,;c)\right) \coloneqq \left(\mathcal{T}, p_s, \tilde{p}_\phi, -\log Z_\phi, p_\theta\right) \tag{33}$$

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:

$$\begin{aligned} d_{\theta,\phi}(\cdot) &= d(\cdot\,; a, b, c) \\ \mathcal{L}_{\text{energy}}(\phi; \theta) &= \mathcal{L}_{\text{class}}(a, b; c) \end{aligned} \tag{34}$$

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}:

$$\mathcal{L}^M_{\text{class}}(a, b; c) \coloneqq -\tfrac{1}{M}\textstyle\sum_{m=1}^M \log d(y^{(m)}_{\text{data}}; a, b, c) - \tfrac{1}{M}\sum_{m=1}^M \log\big(1 - d(y^{(m)}_{\text{noise}}; a, b, c)\big) \tag{35}$$

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 [50]. But continuing the correspondence from before, it is easy to see that

$$\mathcal{L}^M_{\text{energy}}(\phi; \theta) = \mathcal{L}^M_{\text{class}}(a, b; c) \tag{36}$$

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 $y^{(m)}_s \sim p_s$ and $y^{(m)}_\theta \sim p_\theta$:

$$\mathcal{L}^M_{\text{energy}}(\phi; \theta) \coloneqq -\tfrac{1}{M}\textstyle\sum_{m=1}^M \log d_{\theta,\phi}(\tau^{(m)}_s) - \tfrac{1}{M}\sum_{m=1}^M \log\big(1 - d_{\theta,\phi}(\tau^{(m)}_\theta)\big) \tag{37}$$

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 be the associated minimizer for $\mathcal{L}_{\text{policy}}(\theta; \phi_k)$. Then $\nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta^*_k) = -\frac{T}{2} \nabla_\phi \mathcal{L}(\theta^*_k, \phi)$. That is, at θ^*_k the energy gradient (of Equation 11) recovers the original gradient (from Equation 7).

Proof. From Equation 11,

$$\nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta^*_k) = \nabla_\phi\big(-\mathbb{E}_{\tau\sim p_s} \log d_{\theta^*_k,\phi}(\tau) - \mathbb{E}_{\tau\sim p_{\theta^*_k}} \log\big(1 - d_{\theta^*_k,\phi}(\tau)\big)\big) \tag{38}$$

$$= \nabla_\phi\big(-\mathbb{E}_{\tau\sim p_s} \log \frac{p_\phi(\tau)}{p_\phi(\tau) + p_{\theta^*_k}(\tau)} - \mathbb{E}_{\tau\sim p_{\theta^*_k}} \log \frac{p_{\theta^*_k}(\tau)}{p_\phi(\tau) + p_{\theta^*_k}(\tau)}\big) \tag{39}$$

$$= -\mathbb{E}_{\tau\sim p_s} \nabla_\phi(\log p_\phi(\tau) - \log(p_\phi(\tau) + p_{\theta^*_k}(\tau))) + \mathbb{E}_{\tau\sim p_{\theta^*_k}} \nabla_\phi \log(p_\phi(\tau) + p_{\theta^*_k}(\tau)) \tag{40}$$

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

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

$$= -\mathbb{E}_{\tau\sim p_s}\big[\nabla_\phi \log p_\phi(\tau) - \tfrac{1}{2} \nabla_\phi \log p_\phi(\tau)\big] + \tfrac{1}{2} \mathbb{E}_{\tau\sim p_{\theta^*_k}} \nabla_\phi \log p_\phi(\tau) \tag{43}$$

$$= \tfrac{1}{2}\mathbb{E}_{\tau\sim p_{\theta_k^*}} \nabla_\phi \log p_\phi(\tau) - \tfrac{1}{2}\mathbb{E}_{\tau\sim p_s} \nabla_\phi \log p_\phi(\tau) \tag{44}$$

$$= \tfrac{1}{2}\mathbb{E}_{\tau\sim p_{\theta_k^*}} \nabla_\phi (\log \tilde{p}_\phi(\tau) - \log Z_\phi) - \tfrac{1}{2}\mathbb{E}_{\tau\sim p_s} \nabla_\phi (\log \tilde{p}_\phi(\tau) - \log Z_\phi) \tag{45}$$

$$= \tfrac{1}{2}\big(\mathbb{E}_{\tau\sim p_{\theta_k^*}} \nabla_\phi \textstyle\sum_t f_\phi(h_t, x_t) - \mathbb{E}_{\tau\sim p_s} \nabla_\phi \sum_t f_\phi(h_t, x_t)\big) \tag{46}$$

$$= \tfrac{1}{2}\big(T\mathbb{E}_{h\sim\mu_{\theta_k^*}, x\sim\pi_{\theta_k^*}(\cdot|h)} \nabla_\phi f_\phi(h, x) - T\mathbb{E}_{h\sim\mu_s, x\sim\pi_s(\cdot|h)} \nabla_\phi f_\phi(h, x)\big) \tag{47}$$

$$= -\tfrac{T}{2} \nabla_\phi \mathcal{L}(\theta_k^*, \phi) \tag{48}$$

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,

$$\nabla_\phi \mathcal{L}_{\text{energy}}(\phi; \theta_k^*) = -\mathbb{E}_{\tau\sim p_s}\big[\nabla_\phi \log p_\phi(\tau) - \tfrac{p_\phi(\tau)\nabla_\phi \log p_\phi(\tau)}{p_\phi(\tau) + p_{\theta_k^*}(\tau)}\big] + \mathbb{E}_{\tau\sim p_{\theta_k^*}} \tfrac{p_\phi(\tau)\nabla_\phi \log p_\phi(\tau)}{p_\phi(\tau) + p_{\theta_k^*}(\tau)} \tag{49}$$

$$= -\mathbb{E}_{\tau\sim p_s}\big[\nabla_\phi \log p_\phi(\tau) - \tfrac{K_\theta}{K_\theta+1} \nabla_\phi \log p_\phi(\tau)\big] + \tfrac{K_\theta}{K_\theta+1} \mathbb{E}_{\tau\sim p_{\theta_k^*}} \nabla_\phi \log p_\phi(\tau) \tag{50}$$

$$= \tfrac{K_\theta}{K_\theta+1} \mathbb{E}_{\tau\sim p_{\theta_k^*}} \nabla_\phi \log p_\phi(\tau) - \tfrac{1}{K_\theta+1} \mathbb{E}_{\tau\sim p_s} \nabla_\phi \log p_\phi(\tau) \tag{51}$$

$$= \tfrac{K_\theta}{K_\theta+1} \mathbb{E}_{\tau\sim p_{\theta_k^*}} \nabla_\phi (\log \tilde{p}_\phi(\tau) - \log Z_\phi) - \tfrac{1}{K_\theta+1} \mathbb{E}_{\tau\sim p_s} \nabla_\phi (\log \tilde{p}_\phi(\tau) - \log Z_\phi) \tag{52}$$

$$= \tfrac{K_\theta}{K_\theta+1} \mathbb{E}_{\tau\sim p_{\theta_k^*}} \nabla_\phi \textstyle\sum_t f_\phi(h_t, x_t) - \tfrac{1}{K_\theta+1} \mathbb{E}_{\tau\sim p_s} \nabla_\phi \sum_t f_\phi(h_t, x_t) \tag{53}$$

$$= \tfrac{TK_\theta}{K_\theta+1} \mathbb{E}_{h\sim\mu_{\theta_k^*}, x\sim\pi_{\theta_k^*}(\cdot|h)} \nabla_\phi f_\phi(h, x) - \tfrac{T}{K_\theta+1} \mathbb{E}_{h\sim\mu_s, x\sim\pi_s(\cdot|h)} \nabla_\phi f_\phi(h, x) \tag{54}$$

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). □

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):

$$\begin{aligned} V_\phi^{\pi_\theta}(h) &\coloneqq \mathbb{E}_{\tau\sim p_\theta}[\textstyle\sum_{u=t}^T f_\phi(h_u, x_u) + H(\pi_\theta(\cdot|h_u))|h_t = h] \\ Q_\phi^{\pi_\theta}(h, x) &\coloneqq f_\phi(h, x) + \mathbb{E}_{\tau\sim p_\theta}[\textstyle\sum_{u=t+1}^T f_\phi(h_u, x_u) + H(\pi_\theta(\cdot|h_u))|h_t = h, x_t = x] \end{aligned} \tag{55}$$

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):

$$\pi_{\theta^*}(x|h) = \frac{\exp(Q_\phi^{\pi_{\theta^*}}(h, x))}{\int_{\mathcal{X}} \exp(Q_\phi^{\pi_{\theta^*}}(h, x))dx} \tag{56}$$

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 $\mathbb{B}_\phi^{\pi_\theta} : \mathbb{R}^{\mathcal{H}\times\mathcal{X}} \to \mathbb{R}^{\mathcal{H}\times\mathcal{X}}$:

$$(\mathbb{B}_\phi^{\pi_\theta} Q)(h, x) \coloneqq f_\phi(h, x) + \mathbb{E}_{x'\sim\pi_\theta(\cdot|h')}[Q(h', x') - \log \pi_\theta(x'|h')] \tag{57}$$

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 $Q_\phi^{\pi_\theta}$):

$$\theta' \leftarrow \arg\min_\theta D_{\mathrm{KL}}\Big(\pi_{\theta'}(\cdot|h) \Big\| \frac{\exp(Q_\phi^{\pi_\theta}(h,\cdot))}{\int_{\mathcal{X}} \exp(Q_\phi^{\pi_\theta}(h,x))dx}\Big) \tag{58}$$

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 [53]. Specifically, the actor performs soft policy improvement steps as before, but now using Q_ψ:

$$\mathcal{L}_{\text{actor}}(\theta;\phi,\psi) \coloneqq \mathbb{E}_{h\sim\mathcal{B}}\, \mathbb{E}_{x\sim\pi_\theta(\cdot|h)}[\log \pi_\theta(x|h) - Q_\psi(h,x)] \tag{59}$$

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:

$$\mathcal{L}_{\text{critic}}(\psi;\phi) \coloneqq \mathbb{E}_{h,x\sim\mathcal{B}}(Q_\psi(h,x) - Q_\psi^{\text{target}}(h,x))^2 \tag{60}$$

with (bootstrapped) targets:

$$Q_\psi^{\text{target}}(h,x) \coloneqq f_\phi(h,x) + \mathbb{E}_{x'\sim\pi_\theta(\cdot|h')}[Q_\psi(h',x') - \log \pi_\theta(x'|h')] \tag{61}$$

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):

$$\mathcal{L}_{\text{energy}}(\phi;\theta) \coloneqq -\mathbb{E}_{\tau\sim p_s} \log d_{\theta,\phi}(\tau) - \mathbb{E}_{\tau\sim p_\theta} \log\big(1 - d_{\theta,\phi}(\tau)\big) \tag{62}$$

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. In particular, all components taking h_t as input require an encoder network to learn 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 and tanh activations to compute hidden states. These are not shared: separate components have their own encoder networks.

Task-Specific Networks For task-specific networks (i.e. 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 policy (as in T-Forcing and P-Forcing), and critic values (for TimeGCI). Note that TimeGAN and COT-GAN contain additional novelties (e.g. the embedding and recovery networks for generating/discriminating in latent space); these are kept as we use their original architectures.

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 62); 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 59); 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 60); 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 Throughout all experiments, we use the following hyperparameters for TimeGCI (and all benchmarks, wherever applicable). We use a replay buffer of size $|\mathcal{B}| = 10,000$ trajectories. The hidden dimension of both encoder and task-specific networks is set at 32. The entropy regularization (in the actor/policy loss) is set at $\alpha = 0.2$. 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 steps, if performance does not improve). The learning rates are set as follows: 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 similar 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`.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[100] Gokul Swamy, Sanjiban Choudhury, Zhiwei Steven Wu, and J Andrew Bagnell. Of moments and matching: Trade-offs and treatments in imitation learning. *International Conference on Machine Learning (ICML)*, 2021.