ADAPTIVE SYMMETRIZATION OF THE KL DIVERGENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Many tasks in machine learning can be described as or reduced to learning a probability distribution given a finite set of samples. A common approach is to minimize a statistical divergence between the (empirical) data distribution and a parameterized distribution, e.g., a normalizing flow (NF) or an energy-based model (EBM). In this context, the forward KL divergence is a ubiquitous due to its tractability, though its asymmetry may prevent capturing some properties of the target distribution. Symmetric alternatives involve brittle min-max formulations and adversarial training (e.g., generative adversarial networks) or evaluating the reverse KL divergence, as is the case for the symmetric Jeffreys divergence, which is challenging to compute from samples. This work sets out to develop a new approach to minimize the Jeffreys divergence. To do so, it uses a proxy model whose goal is not only to fit the data, but also to assist in optimizing the Jeffreys divergence of the main model. This joint training task is formulated as a constrained optimization problem to obtain a practical algorithm that adapts the models priorities throughout training. We illustrate how this framework can be used to combine the advantages of NFs and EBMs in tasks such as density estimation, image generation, and simulation-based inference.

1 INTRODUCTION

An established approach to learn from observations is to assume they are drawn from a probability distribution (Murphy, 2012) and find the best distribution that explains the data. By learning the underlying distribution we can solve tasks such as classification, regression, and sampling (Kingma & Welling, 2022; LeCun et al., 2015; Grathwohl et al., 2020). Parameterized functions in the form of neural networks are a pivotal instrument to bring this theory into practice, as they can model complex distributions (Hornik, 1991). Training a network to capture a distribution typically involves minimizing the difference between the ground truth distribution, accessible only through a finite number of samples, to the distribution represented by the network. For instance, parameterized distributions such as normalizing flows (NFs) (Tabak & Vanden-Eijnden, 2010; Tabak & Turner, 2013; Papamakarios et al., 2021) or energy-based models (EBMs) (Teh et al., 2003; Du & Mordatch, 2019) are generally trained minimizing the *forward* Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(\pi \parallel p)$ between the true data distribution π and the model distribution p. However, because of the asymmetry of the KL divergence, i.e., $D_{\mathrm{KL}}(\pi \parallel p) \neq D_{\mathrm{KL}}(p \parallel \pi)$, training on discrete samples may lead to a mismatch between the modeled distribution and the data distribution (illustrated in Figure 1a). Ordinarily, minimizing a symmetric divergence would alleviate this issue, but it demands knowing the data distribution directly. Generative adversarial networks (GANs) (Goodfellow et al., 2014; Arjovsky et al., 2017) overcome this obstacle by representing a symmetric divergence as a variational problem, where two models are trained against each other. This formulation provides high-quality results, but is very sensitive to hyperparameters and prone to diverging (illustrated in Figure 1b).

In this paper we minimize the sum of the forward and reverse KL divergence, also known as the (symmetric) Jeffreys divergence (Jeffreys, 1998). To accomplish this, we approximate the reverse KL divergence using a proxy model that simulates the true data distribution. We then use constrained optimization to formulate the joint problem of minimizing the Jeffreys divergence and fitting the proxy model (illustrated in Figure 1c). However, errors in the proxy model could hinder the approximation of the Jeffreys divergence and in extreme cases, render the constrained problem infeasible. We overcome this issue by using an *adaptive symmetrization* that dynamically adjusts the priority

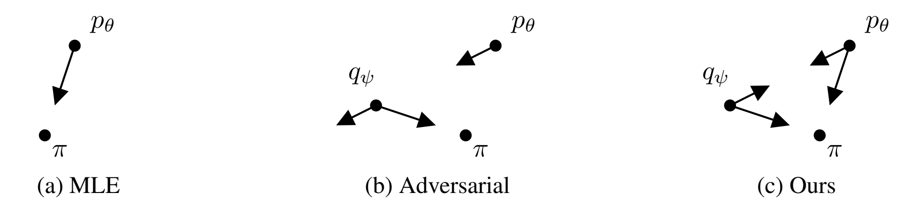

Figure 1: Illustration of different training dynamics, where the goal is to train q_ψ and p_θ to match the data distribution π. Each point represents a probability distribution in conceptual distribution space, and the arrows indicate the direction of gradient descent (GD). (a) In maximum likelihood estimation (MLE), GD minimizes the divergence $D_{\mathrm{KL}}\left(\pi \parallel p_\theta\right)$. In practice, GD minimizes a finite sample estimation of the KL, rather than the true KL. (b) GD of generative adversarial networks (GAN) pulls the generator p_θ towards the discriminator q_ψ and the discriminator towards π, but it also pushes q_ψ away from p_θ. This repelling dynamic makes GAN training unstable. (c) In our framework, GD pulls p_θ and q_ψ towards each other, while also pulling both towards π. The mutual attraction stabilizes training.

between optimizing the forward or reverse KL divergence, as well as fitting the proxy model. We develop a practical algorithm based on (non-convex) duality theory and the synergetic combination of NFs and EBMs. We compare our method with NF and Wasserstein GAN and find that our method is stable, robust and more accurate on a variety of problems, such as density estimation, image generation, and simulation-based inference (SBI).

To summarize, our contributions are: (1) We develop a practical method for optimizing the Jeffreys divergence, a symmetric version of the KL divergence, using constrained optimization and a proxy model; (2) We propose an adaptively symmetrized alternative to the Jeffreys divergence that enables different balances between the forward and reverse KL divergences. These balances are set automatically to trade off between optimizing each of these metrics and fitting the proxy model; (3) We develop a gradient descent-ascent algorithm to tackle this problem and simultaneously train an NF and an EBM. This joint training leverages the complementary strength of these models, mitigating their weaknesses,

2 Problem Formulation

This paper seeks to determine the distribution $p_\theta \in \mathcal{H}$, parameterized by $\theta \in \mathbb{R}^k$, that is *closest* to an unknown probability distribution π over $\mathbb{R}^m$. We assess this closeness using a set of N independently and identically distributed (i.i.d.) observations x_i from π. We assume all distributions are absolutely continuous with respect to Lebesgue measure and use the same for a distribution p_θ and its density $p_\theta(x)$. To formalize this problem, we next define (i) the hypothesis class $\mathcal{H}$ and (ii) a notion of distance.

Hypothesis classes There are as many ways to define $\mathcal{H}$ as there are ways of defining a probability distribution. A traditional choice from variational Bayesian inference considers a set of parametrized distributions, e.g., a Gaussian mixture model (GMM) (McLachlan & Peel, 2000). A more flexible approach used by *normalizing flows* (NFs) consists of letting $p_\theta \in \mathcal{H}$ be the pushforward of a tractable distribution p_0 by an invertible function $g_\theta : \mathbb{R}^m \to \mathbb{R}^m$, namely,

$$p_\theta(x) = p_0\big(g_\theta^{-1}(x)\big)\left|\det Dg_\theta^{-1}(x)\right| \tag{1}$$

where Dg_θ^{-1} denotes the Jacobian of the inverse of g_θ. While g_θ is often a neural network (NN), its architecture has to be carefully designed to ensure that the its inverse and Jacobian are easy to compute (Dinh et al., 2017). This restriction reduces the flexibility of $\mathcal{H}$, but has the advantage of allowing NFs to efficiently sample from the distribution p_θ *as well as* evaluate its density.

A less restricted class is that of *energy-based models* (EBMs) that are based on unnormalized representations of distributions. Explicitly, EBMs are parametrized by an arbitrary function $f_\theta : \mathbb{R}^m \to \mathbb{R}$ that imply the density

$$p_\theta(x) = \frac{e^{f_\theta(x)}}{\zeta_\theta}, \quad \text{with } \zeta_\theta = \int e^{f_\theta(y)}\mathrm{d}y. \tag{2}$$

Since there are no restrictions on f_θ, EBMs offer more powerful distribution models, though it is typically impossible to evaluate their density p_θ since estimating the partition function ζ_θ in (2) is difficult even in moderate dimensions (Kloeckner, 2012). Nevertheless, it is possible to obtain samples directly from f_θ using Monte Carlo Markov Chain (MCMC) methods such as the Langevin Monte Carlo (LMC) algorithm (Parisi, 1981; Grenander & Miller, 1994). Further details about NF and EBM are provided in Appendices A.1 and A.2.

Statistical distances There is a plethora of ways to measure the distance between p_θ and π, from integral probability metrics (e.g., the Wasserstein distance) to f-divergences (Rényi, 1961; Ali & Silvey, 1966; Csiszár, 1967). While they all find particular use cases, the Kullback-Leibler (KL) divergence D_{KL} (Kullback & Leibler, 1951) remains a popular choice as it can be optimized as

$$\underset{p\in\mathcal{H}}{\operatorname{minimize}}\ D_{\mathrm{KL}}(\pi \parallel p) \triangleq \mathbb{E}_{x\sim\pi}\left[\log\left(\frac{\pi(x)}{p(x)}\right)\right] = -H(\pi) - \mathbb{E}_{x\sim\pi}[\log p(x)]. \tag{P-KL}$$

Note that the entropy $H(\pi) = \mathbb{E}_{x\sim\pi}[-\log \pi(x)]$, while unknown, is independent of p. Hence, (P-KL) is equivalent to a statistical learning problem whose solution can be approximated combining empirical risk minimization (ERM) and the parameterizations in (1) or (2). Explicitly,

$$\underset{\theta\in\mathbb{R}^k}{\operatorname{minimize}}\,\mathrm{NLL}\,(p_\theta) \triangleq -\frac{1}{N}\sum_{i=1}^{N}\log p_\theta(x_i). \tag{P-NLL}$$

Hence, (P-KL) is equivalent to minimizing the negative log-likelihood (NLL), i.e., solving a maximum likelihood estimation (MLE) problem (Huber et al., 1967).

Though practical, the KL divergence is not symmetric. Indeed, minimizing the *forward* KL divergence $D_{\mathrm{KL}}(\pi \parallel p_\theta)$ as in (P-KL), leads to solutions that cover the support of π at the cost of overlooking sharp modes. In contrast, the *reverse* KL divergence $D_{\mathrm{KL}}(p_\theta \parallel \pi)$ promotes a mode-seeking behavior that may ignore regions of π with significant mass (Murphy, 2012). One way to overcome this asymmetry is combining forward and reverse divergences as in

$$\underset{\theta\in\mathbb{R}^p}{\operatorname{minimize}}\ D_{\mathrm{J}}(p,\pi) \triangleq D_{\mathrm{KL}}(\pi \parallel p_\theta) + D_{\mathrm{KL}}(p_\theta \parallel \pi). \tag{P-JD}$$

The metric D_{J} is referred to as the *Jeffreys divergence* (JD) (Jeffreys, 1998; Kullback & Leibler, 1951). Yet, in contrast to its forward counterpart, the reverse KL divergence $D_{\mathrm{KL}}(p_\theta \parallel \pi)$ in (P-JD) is difficult to estimate or optimize based only on samples, as it requires access to the unknown density π.

Generative adversarial networks (GANs) (Goodfellow et al., 2014; Arjovsky et al., 2017) avoid this challenge by representing a symmetric divergence, such as the Jensen-Shannon divergence, as a variational problem (see Appendix A.3). This formulation results in a brittle min-max problem susceptible to diverging and an extreme form of mode-seeking known as *mode collapse* (Arjovsky & Bottou, 2017; Farnia & Ozdaglar, 2020). Instead, we next tackle the issues facing (P-JD) using (i) a proxy model and (ii) constrained optimization.

3 Minimizing the Jeffreys Divergence

In this section, we tackle the symmetrized (P-JD) by approximating the reverse KL using a proxy model and constrained optimization (Section 3.1). We then discuss the practical benefits of adapting the level of symmetrization throughout training, which we enable by rewriting (P-JD) as a feasibility problem with optimizable constraint specifications (Section 3.2).

3.1 Approximating the reverse KL

A key roadblock in solving (P-JD) is the fact that the reverse KL divergence $D_{\mathrm{KL}}(p_\theta \parallel \pi)$ directly depends on the unknown data density π. Hence, the reverse KL can only be computed after estimating π from its samples using, e.g., kernel density estimation or k-nearest neighbors (Silverman, 2018). Though effective for estimation, these methods come with bias-variance trade-offs that are difficult to balance, especially as p_θ evolves during training.

To overcome this limitation, we incorporate this data density estimation step directly in (P-JD). Explicitly, consider a proxy distribution model $q_\psi \in \mathcal{H}_\psi$ parametrized by $\psi \in \mathbb{R}^k$ (we assume that θ and ψ have the same dimensions without loss of generality). As long as $q_\psi \approx \pi$, we can use it to train p_θ by computing the reverse KL divergence in the objective of (P-JD) as $D_{\text{KL}}(p_\theta \parallel q_\psi)$. Using constrained optimization, we can simultaneously minimize the Jeffreys divergence, ensure that q_ψ fits the data distribution. Beyond that, we can impose additional constraints, e.g., to regularize the models. Explicitly, we replace (P-JD) by

$$\begin{aligned} \underset{\theta,\psi\in\mathbb{R}^k}{\text{minimize}} \quad & D_{\text{KL}}(\pi \parallel p_\theta) + D_{\text{KL}}(p_\theta \parallel q_\psi) \\ \text{subject to} \quad & D_{\text{KL}}(\pi \parallel q_\psi) \leq \epsilon, \quad h(p_\theta, q_\psi) \leq c, \end{aligned} \tag{PI}$$

where $\epsilon \geq 0$ determines the accuracy of the reverse KL approximation and h, c define (optional) task-dependent constraints. Notice that the proxy model q_ψ is simultaneously trained to fit π (constraint) as well as used to evaluate the reverse KL of p_θ (objective). In general, $\mathcal{H}_\psi$ need not be the same as $\mathcal{H}$ and it might in fact be beneficial to choose different model types (see Section 4.2).

3.2 Adaptive symmetrization

Depending on the expressiveness of the proxy model q_ψ, it may be impossible to choose a small enough value for ϵ to ensure that it is close enough to π. Even when it is possible, unless we ensure that q_ψ is close enough to π throughout training, there could be phases during which the reverse KL approximation is inaccurate. In such situations, we may want to put more emphasis on the forward divergence in the objective of (PI), since it is computed directly from data. In other words, we may want to (temporarily) break the symmetry of the Jeffreys divergence while training q_ψ to fit π. As q_ψ starts to better reflect the data distribution, we can go back to balancing the symmetrization. Dynamically changing the focus between forward and reverse KL divergences steers training between mode-covering and mode-seeking behaviors.

Naturally, it would be impractical to manually adjust these trade-offs throughout training, especially since they depend on the model classes $\mathcal{H}, \mathcal{H}_\psi$ and the data. We therefore rely on the resilient optimization framework proposed in Chamon et al. (2020a;b); Hounie et al. (2023). This approach adjusts constraint specifications such as ϵ in (PI) depending on how difficult it is to fulfill. Accordingly, a constraint that is difficult (easy) to satisfy will have its specification increased (decreased). In its simplest form, this behavior can be achieved by writing (PI) as a feasibility problem and incorporating the constraint specifications in the objective as in

$$\begin{aligned} \underset{\substack{\theta,\psi\in\mathbb{R}^k \\ \epsilon_{\text{fw}},\epsilon_{\text{rv}},\epsilon_{\text{prx}}\geq 0}}{\text{minimize}} \quad & \epsilon_{\text{fw}}^2 + \epsilon_{\text{rv}}^2 + \epsilon_{\text{prx}}^2 \\ \text{subject to} \quad & D_{\text{KL}}(\pi \parallel p_\theta) \leq \epsilon_{\text{fw}}, \quad D_{\text{KL}}(p_\theta \parallel q_\psi) \leq \epsilon_{\text{rv}} \\ & D_{\text{KL}}(\pi \parallel q_\psi) \leq \epsilon_{\text{prx}}, \quad h(p_\theta, q_\psi) \leq c. \end{aligned} \tag{P-DYN}$$

For simplicity, we keep the additional requirements specification c fixed. This formulation is equivalent, under mild conditions, to equalizing the relative sensitivity of the constraints (Chamon et al., 2020a;b; Hounie et al., 2023). In the next section, we develop a practical algorithm to tackle (P-DYN) using NFs and EBMs.

4 Algorithm development

Solving (P-DYN) is challenging due to its dependence on the unknown data distribution π. Even if we used the empirical approximation in (P-NLL), it would remain a non-convex constrained optimization problem for typical choices of p_θ, q_ψ, including NFs and EBMs. It therefore lacks *strong duality* (Bertsekas, 2009; Bonnans & Shapiro, 2013). Although it might be appealing to use the penalty formulation, where each term is weighted by a fixed weight w as

$$\underset{\theta,\psi\in\mathbb{R}^k}{\text{minimize}} \; w_{\text{fw}} D_{\text{KL}}(\pi \parallel p_\theta) + w_{\text{rv}} D_{\text{KL}}(p_\theta \parallel q_\psi) + w_{\text{prx}} D_{\text{KL}}(\pi \parallel q_\psi) \tag{P-W}$$

for weights $w_{\text{fw}}, w_{\text{rv}}, w_{\text{prx}} > 0$, it is not equivalent to solving (P-DYN). In fact, (P-W) leads to worse results in practice (see Section 5.1). To overcome this challenge, we use non-convex duality theory in Section 4.1. We then derive a practical algorithm by using NFs and EBMs (Section 4.2).

4.1 The empirical dual problem

We formulate the empirical version of (P-DYN) with the empirical mean over the data samples and incorporate the constant entropy of the data distributions in the constraint specifications $\boldsymbol{\epsilon}$. Explicitly,

$$\begin{aligned} P^\star \triangleq \min_{\substack{\theta,\psi \in \mathbb{R}^k \\ \boldsymbol{\epsilon} \geq 0}} \quad & \epsilon_{\text{fw}}^2 + \epsilon_{\text{rv}}^2 + \epsilon_{\text{prx}}^2 \\ \text{subject to} \quad & -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(x_i) \leq \epsilon_{\text{fw}}, \quad -\frac{1}{N}\sum_{i=1}^{N} \log q_\psi(x_i) \leq \epsilon_{\text{prx}} \\ & D_{\text{KL}}(p_\theta \parallel q_\psi) \leq \epsilon_{\text{rv}}, \qquad h(p_\theta, q_\psi) \leq c, \end{aligned} \tag{$\hat{\text{P}}$-DYN}$$

where we collect $\epsilon_{\text{fw}}, \epsilon_{\text{rv}}, \epsilon_{\text{prx}}$ in the vector $\boldsymbol{\epsilon}$ and write $\boldsymbol{\epsilon} \geq 0$ to denote that it belongs to the non-negative orthant. We then define the dual problem of ($\hat{\text{P}}$-DYN) as

$$D^\star = \max_{\boldsymbol{\lambda} \geq 0} \min_{\theta,\psi \in \mathbb{R}^k, \boldsymbol{\epsilon} \geq 0} \mathcal{L}(\theta, \psi, \boldsymbol{\epsilon}, \boldsymbol{\lambda}) \tag{$\hat{\text{D}}$-DYN}$$

for the Lagrangian

$$\begin{aligned} \mathcal{L}(\theta, \psi, \boldsymbol{\epsilon}, \boldsymbol{\lambda}) = {} & \epsilon_{\text{fw}}^2 + \epsilon_{\text{rv}}^2 + \epsilon_{\text{prx}}^2 + \lambda_{\text{fw}}\left[-\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(x_i) - \epsilon_{\text{fw}}\right] \\ & + \lambda_{\text{rv}}\left[D_{\text{KL}}(p_\theta \parallel q_\psi) - \epsilon_{\text{rv}}\right] + \lambda_{\text{prx}}\left[-\frac{1}{N}\sum_{i=1}^{N} \log q_\psi(x_i) - \epsilon_{\text{prx}}\right] + \lambda_h\left[h(p_\theta, q_\psi) - c\right], \end{aligned} \tag{3}$$

where the dual variables $\lambda_{\text{fw}}, \lambda_{\text{rv}}, \lambda_{\text{prx}}, \lambda_h$ are collected in the vector $\boldsymbol{\lambda}$. Observe from (3) that the objective of the dual problem ($\hat{\text{D}}$-DYN) is reminiscent of the penalty formulation in (P-W). In fact, they are essentially equivalent for $\boldsymbol{\epsilon} = 0$ and $\lambda_h = 0$. The crucial distinction is that the dual variables $\boldsymbol{\lambda}$ are optimization parameters of ($\hat{\text{D}}$-DYN), whereas the weights w in (P-W) are fixed. This distinction allows us to show that ($\hat{\text{D}}$-DYN) approximates ($\hat{\text{P}}$-DYN) despite non-convexity:

Theorem 1. *Let $p_\theta, q_\psi > 0$ and h be a convex and Lipschitz continuous function. Suppose there exists $\nu \geq 0$ such that for each $p \in \bar{\mathcal{H}}$, the closed convex hull of $\mathcal{H}$, there exists $p_\theta \in \mathcal{H}$ such that $\|p_\theta - p\|_{TV} \leq \nu$. Suppose the same holds for $\mathcal{H}_\psi$. Then, there exists a finite constant B such that*

$$0 \leq P^\star - D^\star \leq B\nu. \tag{4}$$

Proof. The result follows from Chamon et al. (2023, Prop. III.3, Lemma III.1) by noticing that (i) for $p_\theta, q_\psi > 0$, the losses in ($\hat{\text{P}}$-DYN) are Lipschitz continuous (Assumption 1 in Prop. III.3), (ii) $\mathcal{H}$ and $\mathcal{H}_\psi$ satisfy Assumption 3 by hypothesis, and (iii) Assumption 4 is satisfied trivially since $\boldsymbol{\epsilon}$ is an optimization variable. Assumption 2 is not needed for either result. □

Algorithm 1 Gradient descent-ascent

```
Require: Learning rates $\alpha_\psi, \alpha_\theta, \alpha_\lambda, \alpha_\epsilon$
1: $\psi \leftarrow \psi - \alpha_\psi \nabla_\psi \mathcal{L}$
2: $\theta \leftarrow \theta - \alpha_\theta \nabla_\theta \mathcal{L}$
3: $\boldsymbol{\epsilon} \leftarrow \max\{\mathbf{0}, \boldsymbol{\epsilon} - \alpha_\epsilon \nabla_{\boldsymbol{\epsilon}} \mathcal{L}\}$
4: $\boldsymbol{\lambda} \leftarrow \max\{\mathbf{0}, \boldsymbol{\lambda} + \alpha_\lambda \nabla_{\boldsymbol{\lambda}} \mathcal{L}\}$
```

Theorem 1 enables us to (approximately) solve the constrained problem ($\hat{\text{P}}$-DYN) by means of its unconstrained dual problem ($\hat{\text{D}}$-DYN). The latter can be tackled using gradient descent-ascent (GDA) techniques common in primal-dual methods (Boyd & Vandenberghe, 2004; Chamon & Ribeiro, 2020; Chamon et al., 2023; Elenter et al., 2024). They involve taking (projected) gradient descent steps for the primal optimization variables $\theta, \psi, \boldsymbol{\epsilon}$ and (projected) gradient ascent steps for the dual variables $\boldsymbol{\lambda}$ (see Algorithm 1). Theorem 1 also shows that the approximation error can be made arbitrarily small by sufficiently rich models for p_θ and q_ψ. Their choice also affects the implementation of Algorithm 1, as we show next.

4.2 Choosing the main and proxy models

The dual problem ($\hat{\text{D}}$-DYN) jointly optimizes two models, similarly to GANs. Yet the GAN objective (Appendix A.3) permits models without explicit densities, while ($\hat{\text{D}}$-DYN) enforces parameterized distributions. Therefore, natural choices for p_θ and q_ψ are either NFs or EBMs. NFs directly

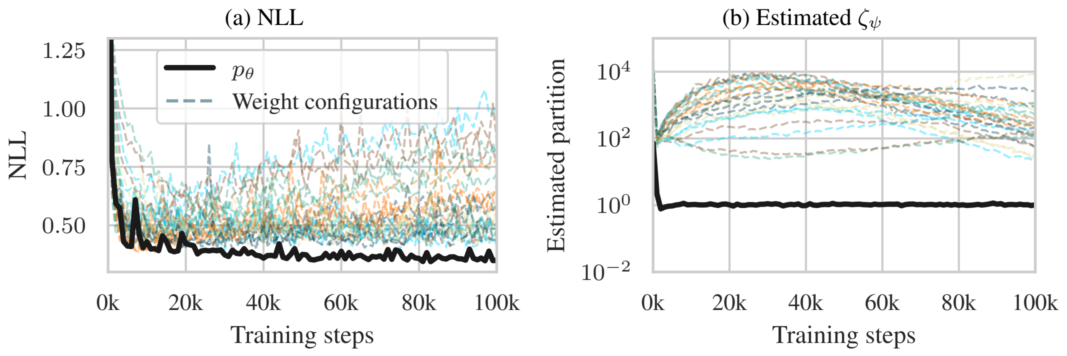

Figure 2: Solving the dual problem (solid black line) achieves better results than 25 weight configurations of the weighted problem Equation (P-W) (dashed colored lines) over a synthetic Gaussian mixture dataset. (a) The p_θ of dual problem has better and more stable NLL than any configuration of the weighted problem. (b) The partition function of the EBM of the dual problem is stable at unity, while the weighted problem may reach large unstable values.

parametrize a probability distribution (Equation (1)) and are easy to evaluate and optimize, making them the ideal choice. However, NF's restricted architecture may prevent them from perfectly capturing π, introducing an irreducible error to the reverse KL approximation and empirical dual problem ($\hat{\text{D}}$-DYN). This hindrance suggests that EBMs are a better choice, as they have virtually unlimited expressivity.

We propose to use NF for p_θ and EBM for q_ψ to exploit their complementary nature to overcome their limitations. The NF makes the log-likelihood of p_θ straightforward to evaluate, while the EBM's versatility reduces the errors incurred from using a proxy model. This symbiotic relationship between NFs and EBMs has been observed in several prior works (see Appendix A.5). At the same time, since EBMs are unnormalized (see Equation (2)), evaluating and optimizing log-likelihoods and KL divergences involving q_ψ require obtaining (a potentially large number of) samples from its distribution, which calls for time-consuming MCMC methods. However, it is straightforward to both obtain samples from the NF p_θ as well as evaluate their density. The partition function of the EBM q_ψ can therefore be estimated using importance sampling as in

$$\zeta_\psi = \int e^{f_\psi(y)} \frac{p_\theta(y)}{p_\theta(y)} dy = \mathbb{E}_{y \sim p_\theta}\left[e^{f_\psi(y) - \log p_\theta(y)}\right]. \tag{5}$$

This replaces the slow EBM sampling by the faster NF. Furthermore, notice that ϵ_{rv} in ($\hat{\text{P}}$-DYN) has the effect of keeping p_θ and q_ψ close to each other, reducing the variance of the importance sampling estimate in (5) (Kloek & van Dijk, 1978; Chatterjee & Diaconis, 2018; Sanz-Alonso, 2018). Still, we can improve numerical stability by ensuring that ζ_ψ remains close to one, i.e., that q_ψ is almost normalized. This can be done in ($\hat{\text{P}}$-DYN) by using $h(q_\psi) = [\zeta_\psi - 1,\ 1 - \zeta_\psi]^\top$.

Using these observations, we can obtain a practical version of Algorithm 1 to tackle ($\hat{\text{D}}$-DYN) and, consequently, the adaptively symmetrized KL divergence optimization problem ($\hat{\text{P}}$-DYN) (see Theorem 1). We present the full algorithm in Appendix B.

5 Experimental Results

5.1 Validation and comparison

We begin by illustrating the advantages of our method by comparing our approach to other baselines. First, we compare our method to the penalty approach (P-W) with fixed weights. To that end, we train both formulations on a synthetic 2D dataset of 800 points drawn from a GMM with 40 components (Midgley et al., 2023) (see Appendix D.1 for the technical details of this experiment). We measure the NLL and partition function over a test dataset of 10k samples. Since we approximate the partition function of q_ψ, the exact NLL is unknown, and the values we report are estimations. To

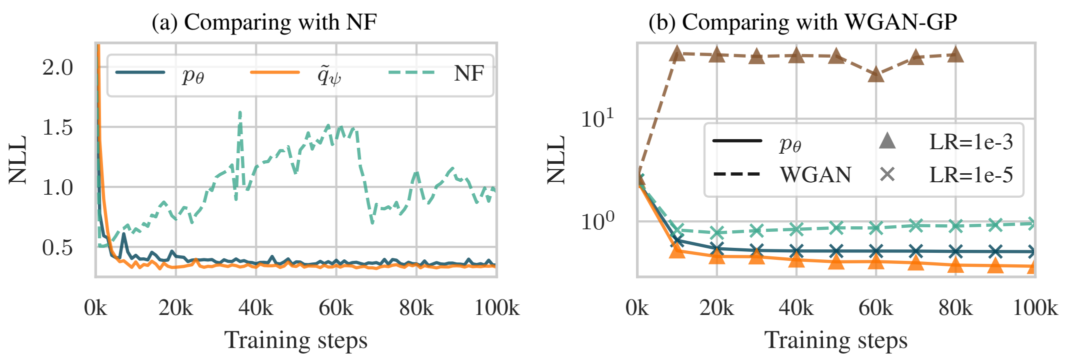

Figure 3: Our dual is more stable than NF and WGAN and outperforms them on a synthetic 2D GMM dataset. (a) The dual problem (solid lines) has a stable KL divergence lower than NF (dashed line). NF's KL is increasing since it overfits to a finite distribution rather than the test GMM distribution. (b) WGANs (dashed lines) is unstable when increasing the learning rate, while our method (solid lines) is stable. Triangles represent high learning rate, and X represents low learning rate.

make this fact explicit, we denote the quasi-normalized density of q_ψ as $\tilde{q}_\psi$. We compare 25 different weight configurations for (P-W) against a single instance of ($\hat{\text{D}}$-DYN) in Figure 2a. The adaptive weights of the dual problem attain lower and more stable NLL than any of the weight combinations we tried. Additionally, Figure 2b shows the numerical stability of ($\hat{\text{D}}$-DYN) compared to (P-W) due to its constraint on the partition function ζ_ψ of the EBM. This constraint effectively controls the value of ζ_ψ, which in contrast can become large when solving (P-W). This instability issue can also be addressed through other means, such as regularization or projected gradient methods.

We also compare our adaptive symmetrization approach with the traditional forward KL divergence used to train NFs. To do so, we use the same dataset, architecture and learning rate as p_θ. While the pure NF attains low NLL initially, the asymmetry of the forward KL divergence overfits the training set and leads to higher NLL on the actual test distribution π. However, both p_θ and q_ψ remain stable at a lower NLL, confirming the advantage of the symmetrized divergence.

Finally, we compare with Wasserstein GAN with gradient penalty (WGAN-GP) (Gulrajani et al., 2017), which minimizes the symmetric Wasserstein distance. To compare with our formulation, we use the same architecture and learning rate (LR) for p_θ and the WGAN's generator and the same architecture and LR for q_ψ and the WGAN's critic. This is a fair comparison since the critic (or discriminator) in GANs are implicit EBMs (Che et al., 2020; Ben-Dov et al., 2024). We then tested two learning rates for both p_θ and the generator and present the results in Figure 3b. Note that the LR of q_ψ was reduced by an order of magnitude from the previous experiments to keep the WGAN training stable. As expected, WGAN is stable only at low learning rates due to the opposing objectives of the generator and the critic. Since our approach is collaborative rather than adversarial, it remains stable over a wider range of LRs and reaches lower NLL values.

5.2 Density estimation

We test the density estimation capabilities of our method on five common 2D datasets: Gaussian mixture ring, Gaussian mixture grid, two moons, spiral, and concentric circles. Some of these distribution have low entropy and result in negative NLL, which is not considered by the resilient approach in (P-DYN). To compensate for that, we use a modified formulation that accommodates negative constraints (see Appendix B.7). Figure 4 presents two of these datasets, while Appendix D.2 presents the rest and details the training configurations. In all the datasets, our method learn a density with lower and more consistent NLL than a pure NF (Figures 4a and 4c). Moreover, the qualitative density maps in Figures 4b and 4d show that both models perfectly capture the shape and all modes.

5.3 Sampling

Next, we illustrate the sampling quality of p_θ by learning to sample from the image dataset CelebA (Liu et al., 2015). The main problem when training on a high-dimensional space is that

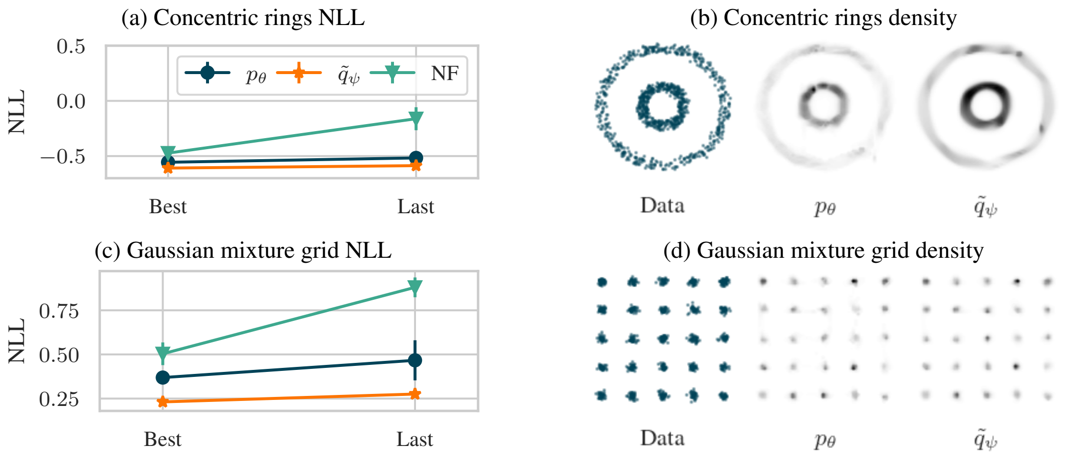

Figure 4: Our framework is able to accurately learn the density of various 2D datasets. The left column (a,c) present the lowest NLL each model achieved during training ("Best") and the NLL at the end of training ("Last"). The error bars represent the standard deviation over 5 seeded runs, demonstrating that our method outperforms NF in terms of values and consistency. The right column (b,d) qualitatively displays the finite dataset (left) and the learned density values from of p_θ and the quasi-normalized $\tilde{p}_\psi$ (middle and right respectively). The qualitative density maps show that both models perfectly capture the shape and all modes.

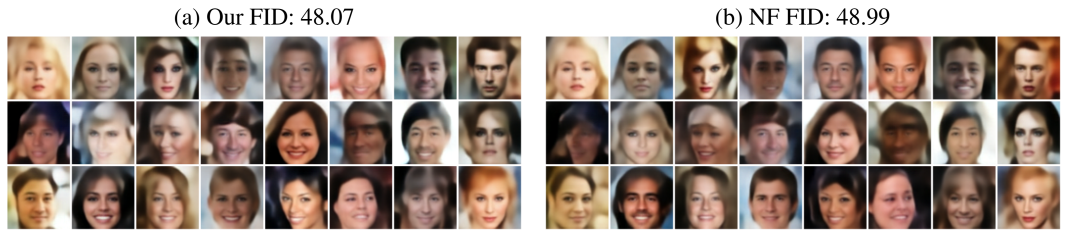

Figure 5: Our p_θ learns a comparable 100D encoded space of CelebA as NF. (a) reports the FID and qualitative samples generated from our p_θ, and (b) reports the same for NF. We used the same seed to generate the sample, which explains the similarity between the images.

the gradient of ψ may explode due to the exponential in $\nabla_\psi \mathcal{L}$ (see Appendix B). We mitigate this issue by taking three steps. First, we train an autoencoder on the training dataset and learn a 100D encodings that serve as the training data for our models. Second, we add a temperature parameter to the EBM that scales the density values to the correct order of magnitude. Third, before training we warm start the EBM by training it to have a similar density as the NF. Appendix B.8 details the implementation of these steps and Appendix D.3 the training details. To compare between our method and a pure NF, we use the Fréchet inception distance (FID) (Heusel et al., 2017), a common metric to measure the generation quality of a model. Figure 5 displays the decoded samples from p_θ and an NF and their FID values. Our model gets comparable FID to NF.

5.4 Simulation based inference

Here we utilize our dual formulation to to learn conditional distributions for simulations-based inference (SBI). Many scientific fields attempt to determine the parameters of a mechanism β given observations of its outcome u. However, even with perfect knowledge of the mechanism and the ability to predict $\pi(u|\beta)$, the inverse — inferring the parameters given the outcome $\pi(\beta|u)$ — is challenging when the mechanism is stochastic. For example, despite having a perfect model to predict gravitational waves from a known source, it is harder to infer the sources given a wave (Dax et al., 2021; Wildberger et al., 2023). SBI provides solutions to amortized inference by training a model on simulated instances (Cranmer et al., 2020). A simple method to perform this inference is neural posterior estimation (NPE) (Papamakarios & Murray, 2016; Lueckmann et al., 2017; Radev et al., 2022). This requires to define a prior distribution p_β over the parameters, sample N instance

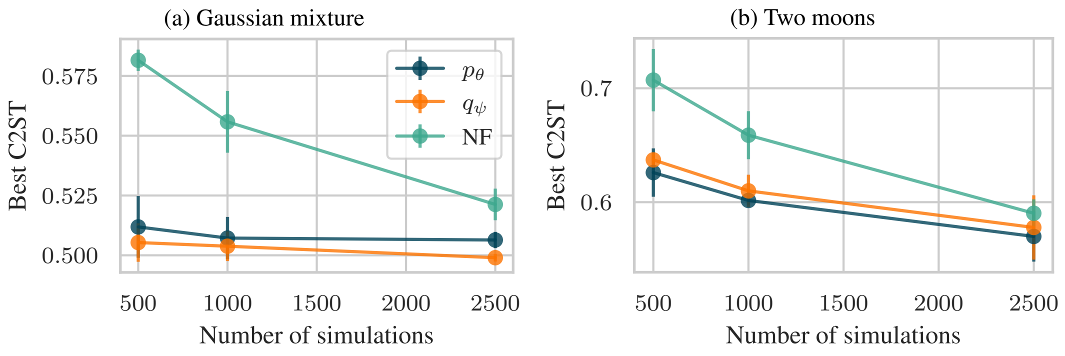

Figure 6: Our method requires less simulations than NF to achieve C2ST scores closer to 0.5 for common SBI benchmarks. For each method, over 5 different seeds, we compute the best C2ST over the run and report the mean and standard deviation as the error bars.

of β, and simulate N outcomes $\pi\left(u|\beta\right)$. The conditional inference network then trains on the pairs (β, u) to learn the posterior $\pi\left(\beta|u\right)$. Prior work shows that maximizing the entropy of the learned distribution improves the chances of recovering the underlying distribution (Vetter et al., 2024). While prior works add an entropy regularization term, note that the Lagrangian in Equation (3) naturally contains the entropy of p_θ due to the reverse KL.

In our experiments, we use the conditional version of ($\hat{\text{P}}$-DYN), which we develop in Appendix B.6. We trained our method on the common SBI benchmarks two moons (Greenberg et al., 2019), and Gaussian mixture (Sisson et al., 2007). As common in the SBI literature (Ramesh et al., 2022; Vetter et al., 2024), we measure the effect of the size of the dataset (number of simulations) on the quality of the learned posterior. See Appendix D.4 for the full technical details. For evaluation, we used the classifier-two-sample testing (C2ST). The C2ST score is defined as the accuracy of a classifier trained to distinguish between samples from the ground truth posterior and samples from the learned posterior. The optimal value is 0.5, indicating that the generated posterior is indistinguishable from the ground truth posteriors. To train the classifier, we created a data set by sampling 5000 points from both the ground truth posterior and our models, conditioned on the same pre-determined u_0 and labeled them accordingly. We run the experiment 5 times with different seeds and report the results in Figure 6. Our method generates higher quality posterior samples than a pure NF when trained on the same number of samples.

6 Conclusion

Learning a distribution from finite samples commonly requires minimizing the asymmetric KL divergence or a symmetric divergence via an unstable min-max objective. In this paper we minimize the symmetric Jeffreys divergence by having two models collaborate rather than being adverse. The additional model's task is to mimic the data distribution, which we use to estimate the reverse KL divergence. We use a constrained optimization problem to jointly train the two models by solving the unconstrained dual problem. We demonstrate on various tasks that our method is stable, and requires less data points to arrive at the same accuracy as other methods. An additional benefit of our method is that while proxy model (EBM) is used to assist the training of the primary model (NF), it is also itself trained on the data and can be used to potentially obtain higher quality samples.

We find this formulation to have exciting future research directions. For example, it would be interesting to extend this method to work in high-dimensional settings by estimating the probability of generated samples (Ben-Dov et al., 2024) or by using rectangular flows (Caterini et al., 2021). Also, while estimating the partition ζ_ψ using samples from p_θ is relatively efficient (Appendix E), high dimensions may require streamlining this process by adaptively changing the number of samples and the frequency of estimation. A theoretical direction involves using other models and divergences in (P-DYN), such as score-based models with the Fisher divergence. One form of the Fisher divergence avoids the score of π by computing the Hessian (Hyvärinen, 2005). With our formulation, rather than the expensive computations of the Hessian, we can use the score of a tractable proxy model. Future work also includes training the models with additional constraints for specific tasks, such as fairness or privacy.

Reproducibility statement

We provide the technical details to reproduce our experiments in the appendix. In Appendix B we detail our algorithm and models and in Appendix D we specify the specific of each experiment.

References

Syed Mumtaz Ali and Samuel D. Silvey. A General Class of Coefficients of Divergence of One Distribution from Another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966.

Martin Arjovsky and Leon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 214–223. PMLR, 2017.

Omri Ben-Dov, Pravir Singh Gupta, Victoria Abrevaya, Michael J. Black, and Partha Ghosh. Adversarial Likelihood Estimation With One-Way Flows. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3779–3788, 2024.

Dimitri Bertsekas. *Convex Optimization Theory*, volume 1. Athena Scientific, 2009.

J Frédéric Bonnans and Alexander Shapiro. *Perturbation Analysis of Optimization Problems*. Springer Science & Business Media, 2013.

Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

Anthony L Caterini, Gabriel Loaiza-Ganem, Geoff Pleiss, and John P Cunningham. Rectangular flows for manifold learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 30228–30241. Curran Associates, Inc., 2021.

Luiz Chamon and Alejandro Ribeiro. Probably approximately correct constrained learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 16722–16735. Curran Associates, Inc., 2020.

Luiz F. O. Chamon, Alexandre Amice, Santiago Paternain, and Alejandro Ribeiro. Resilient control: Compromising to adapt. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 5703–5710. IEEE Press, 2020a.

Luiz F. O. Chamon, Santiago Paternain, and Alejandro Ribeiro. Counterfactual programming for optimal control. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120, pp. 235–244. PMLR, 2020b.

Luiz F. O. Chamon, Santiago Paternain, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained Learning With Non-Convex Losses. *IEEE Transactions on Information Theory*, 69(3):1739–1760, 2023.

Sourav Chatterjee and Persi Diaconis. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is secretly an energy-based model and you should use discriminator driven latent sampling. In *Advances in Neural Information Processing Systems*, volume 33, pp. 12275–12287. Curran Associates, Inc., 2020.

Andrzej Cichocki and Shun-ichi Amari. Families of Alpha- Beta- and Gamma- Divergences: Flexible and Robust Measures of Similarities. *Entropy*, 12(6):1532–1568, 2010.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

Imre Csiszár. On information-type measure of difference of probability distributions and indirect observations. *Studia Sci. Math. Hungar.*, 2:299–318, 1967.

Danilo Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 1530–1538. PMLR, 2015.

Maximilian Dax, Stephen R. Green, Jonathan Gair, Jakob H. Macke, Alessandra Buonanno, and Bernhard Schölkopf. Real-time gravitational wave science with neural posterior estimation. *Physical Review Letters*, 127(24):241103, 2021.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations*, 2017.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Juan Elenter, Luiz F. O. Chamon, and Alejandro Ribeiro. Near-optimal solutions of constrained learning problems. In *The Twelfth International Conference on Learning Representations*, 2024.

Farzan Farnia and Asuman Ozdaglar. Do GANs always have Nash equilibria? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 3029–3039. PMLR, 2020.

Ruiqi Gao, Erik Nijkamp, Diederik P. Kingma, Zhen Xu, Andrew M. Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Ian J. 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*, volume 27. Curran Associates, Inc., 2014.

Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020.

David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2404–2414. PMLR, 2019.

Ulf Grenander and Michael I. Miller. Representations of Knowledge in Complex Systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-GAN: Combining maximum likelihood and adversarial learning in generative models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2722–2730. PMLR, 2019.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4 (2):251–257, 1991.

Ignacio Hounie, Alejandro Ribeiro, and Luiz F. O. Chamon. Resilient constrained learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 71767–71798. Curran Associates, Inc., 2023.

Peter J Huber et al. The behavior of maximum likelihood estimates under nonstandard conditions. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pp. 221–233. Berkeley, CA: University of California Press, 1967.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

Harold Jeffreys. *Theory of Probability*. Oxford University Press, 1998.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A Theory of Generative ConvNet. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pp. 2635–2644. PMLR, 2016.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, 2022.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Benoît Kloeckner. Approximation by finitely supported measures. *ESAIM: COCV*, 18(2):343–359, 2012.

Teun Kloek and Herman K van Dijk. Bayesian Estimates of Equation System Parameters: An Application of Integration by Monte Carlo. *Econometrica*, 46(1):1–19, 1978.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Geoffrey McLachlan and Davis Peel. Finite mixture models. *A wiley-interscience publication*, 2000.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 3481–3490. PMLR, 2018.

Laurence Illing Midgley, Vincent Stimper, Gregor N. C. Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow Annealed Importance Sampling Bootstrap. In *The Eleventh International Conference on Learning Representations*, 2023.

Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. F-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with bayesian conditional density estimation. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Giorgio Parisi. Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384, 1981.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

Stefan T Radev, Ulf K Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Kothe. BayesFlow: Learning Complex Stochastic Models With Invertible Neural Networks. *IEEE Trans Neural Netw Learn Syst*, 33(4):1452–1466, 2022.

Poornima Ramesh, Jan-Matthis Lueckmann, Jan Boelts, Álvaro Tejero-Cantero, David S. Greenberg, Pedro J. Goncalves, and Jakob H. Macke. GATSBI: Generative Adversarial Training for Simulation-Based Inference. In *International Conference on Learning Representations*, 2022.

Alfréd Rényi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, volume 4, pp. 547–562. University of California Press, 1961.

Daniel Sanz-Alonso. Importance sampling and necessary sample size: An information theory approach. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):867–879, 2018.

Bernard W Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, 2018.

S. A. Sisson, Y. Fan, and Mark M. Tanaka. Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760–1765, 2007.

Yang Song and Diederik P. Kingma. How to Train Your Energy-Based Models, 2021.

Esteban G. Tabak and Cristina V. Turner. A Family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003.

Benigno Uria, Iain Murray, and Hugo Larochelle. RNADE: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Julius Vetter, Guy Moss, Cornelius Schröder, Richard Gao, and Jakob H. Macke. Sourcerer: Sample-based maximum entropy source distribution estimation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 88772–88806. Curran Associates, Inc., 2024.

Jonas Wildberger, Maximilian Dax, Stephen R. Green, Jonathan Gair, Michael Pürrer, Jakob H. Macke, Alessandra Buonanno, and Bernhard Schölkopf. Adapting to noise distribution shifts in flow-based gravitational-wave inference. *Physical Review D: Particles and Fields*, 107(8):084046, 2023.

Jianwen Xie, Yaxuan Zhu, Jun Li, and Ping Li. A tale of two flows: Cooperative learning of langevin flow and normalizing flow toward energy-based model. In *International Conference on Learning Representations*, 2022.

Miaoyun Zhao, Yulai Cong, Shuyang Dai, and Lawrence Carin. Bridging Maximum Likelihood and Adversarial Learning via α-Divergence. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6901–6908, 2020.

7 Appendix

A Related Work

A.1 Energy-based models

Any function $f_\psi : \mathbb{R}^m \to \mathbb{R}$ has a corresponding probability distribution

$$q_\psi(x) = \frac{e^{f_\psi(x)}}{\zeta_\psi}, \text{ with } \quad \zeta_\psi = \int e^{f_\psi(y)} \mathrm{d}y. \tag{6}$$

The exponent keeps the output non-negative and the partition function ζ_ψ ensures that the distribution integrates to one. Essentially, Equation (2) is the definition of a Boltzmann distribution with an energy function $-f_\psi$, consequently naming this class of functions as energy-based models (Teh et al., 2003; Jianwen Xie et al., 2016; Du & Mordatch, 2019).

The gradient of the NLL objective in Equation (P-KL) with q_ψ from Equation (2) is

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{x\sim\pi}\left[-\log q_\psi(x)\right] &= \mathbb{E}_{x\sim\pi}\left[-\nabla_\theta f_\psi(x) + \nabla_\theta \log \zeta_\psi\right] \\
&= \mathbb{E}_{x\sim\pi}\left[-\nabla_\theta f_\psi(x)\right] + \nabla_\theta \log \zeta_\psi \\
&= \mathbb{E}_{x\sim\pi}\left[-\nabla_\theta f_\psi(x)\right] + \frac{1}{\zeta_\psi}\nabla_\theta \int e^{f_\psi(y)} \mathrm{d}y \\
&= \mathbb{E}_{x\sim\pi}\left[-\nabla_\theta f_\psi(x)\right] + \int \frac{1}{\zeta_\psi} e^{f_\psi(y)} \nabla_\theta f_\psi(y)\, \mathrm{d}y \\
&= \mathbb{E}_{x\sim\pi}\left[-\nabla_\theta f_\psi(x)\right] + \int q_\psi(y)\, \nabla_\theta f_\psi(y)\, \mathrm{d}y \\
&= \mathbb{E}_{x\sim\pi}\left[-\nabla_\theta f_\psi(x)\right] + \mathbb{E}_{y\sim q_\psi}\left[\nabla_\theta f_\psi(y)\right]
\end{aligned} \tag{7}$$

The second term of the gradient requires sampling from the EBM itself (Du & Mordatch, 2019; Song & Kingma, 2021). Sampling from q_ψ is not straightforward and typically involves running a Markov Chain Monte Carlo (MCMC) method, such as Langevin dynamics (Parisi, 1981; Grenander & Miller, 1994). Starting with x_0 from a simple prior distribution, the k-th Langevin step is

$$x_k = x_{k-1} + s\nabla_x \log q_\psi(x) + \sqrt{2s}\boldsymbol{z}_{k-1}, \tag{8}$$

with s being the step-size and $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, I)$. EBMs are a powerful class of functions due to their unrestricted architecture, but the Langevin sampling makes their training slow and requires more hyperparameter tuning.

A.2 Normalizing flows

Unlike EBMs, which directly define the unnormalized density, NFs define an invertible mapping $f_\theta : \mathbb{R}^m \to \mathbb{R}^m$, parameterized by θ, from a simple base distribution p_0 to the data distribution (Tabak & Vanden-Eijnden, 2010; Tabak & Turner, 2013; Danilo Rezende & Shakir Mohamed, 2015; Papamakarios et al., 2021). Training NF involves minimizing the NLL from Equation (P-KL) through the change-of-variables law of probabilities

$$\log p_\theta(x) = \log p_0\left(f_\theta^{-1}(x)\right) + \log\left|\det \frac{\mathrm{d}f_\theta^{-1}(x)}{\mathrm{d}x}\right|, \tag{9}$$

where $\frac{\mathrm{d}f_\theta^{-1}(x)}{\mathrm{d}x}$ is the Jacobian of f_θ^{-1} at x. While computing the Jacobian of a multivariate function is computationally expensive, practical NF relies on architectures with tractable Jacobians (Dinh et al., 2017; Kingma & Dhariwal, 2018; Durkan et al., 2019). These Jacobians simplify the training of NF, and sampling is quick compared to EBMs. Yet, since the forward KL is minimized from a finite dataset, NF tend to overfit to the empirical distribution rather than the underlying distribution. This problem is usually addressed by adding random noise to the data (Uria et al., 2013; Ho et al., 2019). Sampling from NF is quicker than sampling from an EBM as it requires only sampling from the base distribution and applying the invertible mapping.

A.3 Generative adversarial networks

When minimizing only the empirical forward D_{KL}, the model learns to give high probability to the training samples, but may assign high probability to regions where π is low, i.e., have low precision. This is due to the asymmetry of the KL divergence $D_{\mathrm{KL}}\left(p \parallel q\right) \neq D_{\mathrm{KL}}\left(q \parallel p\right)$, which relies on sampling from only one of the distributions. Minimizing a symmetric divergence may avoid this pitfall, but generally requires access to the unknown π. However, some symmetric divergences can be expressed as a variational representation without direct access to π. For example, minimizing any f-divergence, a divergence D_f defined by a function f , can be formulated as the adversarial min-max objective (Nguyen et al., 2010; Nowozin et al., 2016)

$$\min_{\theta} D_f\left(\pi \parallel p_\theta\right) = \min_{\theta} \max_{\psi} \left(\mathbb{E}_{x\sim\pi}\left[g_\psi\left(x\right)\right] - \mathbb{E}_{x\sim p_\theta}\left[f^*\left(g_\psi\left(x\right)\right)\right]\right), \tag{10}$$

where g_ψ is a parameterized function and f^* is the convex conjugate of f. The most common applications of this formulation is the generative adversarial networks (GAN) (Goodfellow et al., 2014) minimizing the Jensen-Shannon divergence, and WGAN (Arjovsky et al., 2017) which minimizes the Wasserstein distance[1]. The advantages of adversarial training is the efficient sampling from the generator p_θ, and that the discriminator g_ψ can be used as an EBM (Che et al., 2020; Ben-Dov et al., 2024). Yet, GANs' adversarial minimax game between the two models may not always converge to a stable solution, may have vanishing gradients or suffer mode collapse (Arjovsky & Bottou, 2017; Mescheder et al., 2018; Farnia & Ozdaglar, 2020). In this paper we propose a non-adversarial objective to train a symmetrized divergence.

A.4 Minimizing the forward and reverse KL

Prior attempts to explicitly minimize both the forward and reverse KL divergences include the α-bridge (Zhao et al., 2020). This method leverages the definition of an α-divergence (Cichocki & Amari, 2010) which converges to the forward and reverse KL in the limits $\alpha \to 0^+$ and $\alpha \to 1^-$ by training model in three steps. First, the algorithm minimizes the forward KL by maximizing the evidence lower bound (ELBO), similarly to variational autoencoders (VAE). Second, the algorithm increases α according to a fixed schedule and minimizes the corresponding α-divergence. Third, the algorithm minimizes the reverse KL through adversarial training (as the f-divergence in Appendix A.3). Our method (Section 4) avoids the ELBO approximation and adversarial training by jointly optimizing both divergences in a single objective. Additionally, rather than a fixed schedule to balance between the two divergence, our method dynamically adjusts the weight of each direction during optimization.

A.5 Jointly training EBM and NF

NF are fast samplers but have low expressiveness compared to EBM which are flexible, but requires a slow MCMC sampling process. This contrast encouraged research into jointly training these models so each model's strength covers for the other's weakness. Xie et al. (2022) propose to fit an NF to an EBM, which will then generate an initial starting point for the Langevin dynamics in the training of the EBM. This approach accelerates the training of the EBM but keeps the MCMC process, albeit shortening the mixing time. Other approaches remove the MCMC by installing the EBM and NF in a GAN setting (Grover et al., 2018; Gao et al., 2020; Ben-Dov et al., 2024). While these works avoid the slow EBM sampling, they inherit the instability of adversarial training.

[1]Wasserstein distance is not an f-divergence, but can also be expressed as a variational problem with a 1-Lipschitz constraint over g_ψ.

B Algorithm

B.1 Practical problem formulation

In practice, we formulate Equation ($\hat{\text{P}}$-DYN) in our experiments by adding an additional constraint over the normalization of the EBM as

$$\begin{aligned}
\underset{\psi,\theta,\boldsymbol{\epsilon}\geq 0}{\text{minimize}} \quad & \epsilon_{\text{fw}}^2 + \epsilon_{\text{rv}}^2 + \epsilon_{\text{EBM}}^2 \\
\text{subject to} \quad & -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta\left(x_i\right) \leq \epsilon_{\text{fw}} \\
& \frac{1}{N}\sum_{i=1}^{N} \left[\log p_\theta\left(y_i\right) - \log f_\psi\left(y_i\right)\right] + \log \zeta_\psi \leq \epsilon_{\text{rv}} \\
& -\frac{1}{N}\sum_{i=1}^{N} \left[\log f_\psi\left(x_i\right)\right] + \log \zeta_\psi \leq \epsilon_{\text{EBM}} \\
& 1 \leq \zeta_\psi \leq 1 + \epsilon_\zeta,
\end{aligned} \tag{PII}$$

where x_i are the data points and y_i are samples from p_θ. The corresponding Lagrangian is

$$\begin{aligned}
\mathcal{L}\left(\psi,\theta,\boldsymbol{\epsilon};\boldsymbol{\lambda}\right) = \frac{1}{N}\sum_{i=1}^{N} & \left[-\lambda_{\text{fw}} \log p_\theta\left(x_i\right) - \lambda_{\text{prx}} \log f_\psi\left(x_i\right) + \lambda_{\text{rv}} \log p_\theta\left(y_i\right) - \lambda_{\text{rv}} \log f_\psi\left(y_i\right)\right] \\
& + \epsilon_{\text{fw}}^2 - \lambda_{\text{fw}}\epsilon_{\text{fw}} + \epsilon_{\text{rv}}^2 - \lambda_{\text{rv}}\epsilon_{\text{rv}} + \epsilon_{\text{EBM}}^2 - \lambda_{\text{prx}}\epsilon_{\text{EBM}} \\
& + \lambda_{\text{u}}\left(-1-\epsilon_\zeta\right) + \lambda_\ell + \left(\lambda_{\text{u}} - \lambda_\ell\right)\zeta_\psi + \left(\lambda_{\text{prx}} + \lambda_{\text{rv}}\right)\log\zeta_\psi.
\end{aligned} \tag{11}$$

The partition function is empirically estimated as

$$\zeta_\psi \approx \frac{1}{M}\sum_{i=1}^{M}\left[e^{f_\psi(y_i) - \log p_\theta(y_i)}\right], \quad \text{with } y_i \sim p_\theta. \tag{12}$$

B.2 Closed form solution for $\boldsymbol{\epsilon}$

Given $\boldsymbol{\lambda}$, equating the derivative $\nabla_{\boldsymbol{\epsilon}}\mathcal{L}$ to zero, results in a closed form solution

$$\begin{aligned}
0 &= \nabla_{\boldsymbol{\epsilon}}\mathcal{L} \\
&= \nabla_{\boldsymbol{\epsilon}}\left(\boldsymbol{\epsilon}^2 - \boldsymbol{\lambda}\boldsymbol{\epsilon}\right) \\
&= 2\boldsymbol{\epsilon} - \boldsymbol{\lambda} \\
\boldsymbol{\epsilon} &= \frac{1}{2}\boldsymbol{\lambda}
\end{aligned} \tag{13}$$

B.3 Gradient for ψ

First we derive the gradient of ζ_ψ

$$\begin{aligned}
\nabla_\psi \zeta_\psi &= \int \frac{p_\theta\left(y\right)}{p_\theta\left(y\right)} \nabla_\psi e^{f_\psi(y)} \mathrm{d}y \\
&= \int p_\theta\left(y\right) e^{f_\psi(y) - \log p_\theta(y)} \nabla_\psi f_\psi\left(y\right) \mathrm{d}y \\
&= \mathbb{E}_{y\sim p_\theta}\left[e^{f_\psi(y) - \log p_\theta(y)} \nabla_\psi f_\psi\left(y\right)\right] \\
&\approx \frac{1}{M}\sum_{i=1}^{M}\left[e^{f_\psi(y_i) - \log p_\theta(y_i)} \nabla_\psi f_\psi\left(y_i\right)\right].
\end{aligned} \tag{14}$$

We use this for deriving the gradient of the Lagrangian

$$\begin{aligned}
\nabla_\psi \mathcal{L} &= \lambda_{\text{prx}} \frac{1}{N} \sum_{i=1}^{N} \left[-\nabla_\psi f_\psi \left(x_i \right) \right] + \lambda_{\text{rv}} \frac{1}{N} \sum_{i=1}^{N} \left[-\nabla_\psi f_\psi \left(y_i \right) \right] \\
&\quad + \left(\frac{\lambda_{\text{rv}} + \lambda_{\text{prx}}}{\zeta_\psi} + \lambda_{\zeta,u} - \lambda_{\zeta,l} \right) \nabla_\psi \zeta_\psi \\
&= \frac{1}{N} \sum_{i=1}^{N} \left[-\lambda_{\text{prx}} \nabla_\psi f_\psi \left(x_i \right) \right] + \frac{1}{N} \sum_{i=1}^{N} \left[-\lambda_{\text{rv}} \nabla_\psi f_\psi \left(y_i \right) \right] \\
&\quad + \left(\frac{\lambda_{\text{rv}} + \lambda_{\text{prx}}}{\zeta_\psi} + \lambda_{\zeta,u} - \lambda_{\zeta,l} \right) \frac{1}{N} \sum_{i=1}^{N} \left[e^{f_\psi(y_i) - \log p_\theta(y_i)} \nabla_\psi f_\psi \left(y_i \right) \right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \left[-\lambda_{\text{prx}} \nabla_\psi f_\psi \left(x_i \right) \right] \\
&\quad + \frac{1}{N} \sum_{i=1}^{N} \left[\left(\left(\frac{\lambda_{\text{rv}} + \lambda_{\text{prx}}}{\zeta_\psi} + \lambda_{\zeta,u} - \lambda_{\zeta,l} \right) e^{f_\psi(y_i) - \log p_\theta(y_i)} - \lambda_{\text{rv}} \right) \nabla_\psi f_\psi \left(y_i \right) \right]
\end{aligned} \tag{15}$$

where ζ_ψ is estimated as in Equation (12).

B.4 Gradient for θ

We first derive the helper gradients

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{y \sim p_\theta} \left[f_\psi \left(y \right) \right] &= \nabla_\theta \int f_\psi \left(y \right) p_\theta \left(y \right) \mathrm{d}y \\
&= \int f_\psi \left(y \right) p_\theta \left(y \right) \nabla_\theta \log p_\theta \left(y \right) \mathrm{d}y \\
&= \mathbb{E}_{y \sim p_\theta} \left[f_\psi \left(y \right) \nabla_\theta \log p_\theta \left(y \right) \right] \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \left[f_\psi \left(y_i \right) \nabla_\theta \log p_\theta \left(y_i \right) \right]
\end{aligned} \tag{16}$$

and

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{y \sim p_\theta} \left[\log p_\theta \left(y \right) \right] &= \nabla_\theta \int p_\theta \left(y \right) \log p_\theta \left(y \right) \mathrm{d}y \\
&= \int \log p_\theta \left(y \right) \nabla_\theta p_\theta \left(y \right) + p_\theta \left(y \right) \nabla_\theta \log p_\theta \left(y \right) \mathrm{d}y \\
&= \int p_\theta \left(y \right) \cdot \log p_\theta \left(y \right) \nabla_\theta \log p_\theta \left(y \right) \mathrm{d}y + \int \nabla_\theta p_\theta \left(y \right) \mathrm{d}y \\
&= \mathbb{E}_{y \sim p_\theta} \left[\log p_\theta \left(y \right) \nabla_\theta \log p_\theta \left(y \right) \right] + \cancel{\nabla_\theta \int p_\theta \left(y \right) \mathrm{d}y} \\
&= \mathbb{E}_{y \sim p_\theta} \left[\log p_\theta \left(y \right) \nabla_\theta \log p_\theta \left(y \right) \right] \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \left[\log p_\theta \left(y_i \right) \nabla_\theta \log p_\theta \left(y_i \right) \right],
\end{aligned} \tag{17}$$

where the last term is removed because the integral over a probability distribution is unity, and the gradient of a constant is zero. We use these for gradient of the Lagrangian

$$\begin{aligned}
\nabla_\theta \mathcal{L} &= \nabla_\theta \left(\frac{1}{N} \sum_{i=1}^{N} \left[-\lambda_{\text{fw}} \log p_\theta(x_i) \right] + \lambda_{\text{rv}} \frac{1}{N} \sum_{i=1}^{N} \left[\log p_\theta(y_i) - f_\psi(y_i) \right] \right) \\
&= \frac{1}{N} \sum_{i=1}^{N} \left[-\lambda_{\text{fw}} \nabla_\theta \log p_\theta(x_i) \right] + \lambda_{\text{rv}} \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \left[\log p_\theta(y_i) \right] - \lambda_{\text{rv}} \frac{1}{N} \sum_{i=1}^{N} \left[f_\psi(y_i) \right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \left[-\lambda_{\text{fw}} \nabla_\theta \log p_\theta(x_i) \right] + \lambda_{\text{rv}} \frac{1}{N} \sum_{i=1}^{N} \left[\log p_\theta(y_i) \nabla_\theta \log p_\theta(y_i) \right] \\
&\quad - \lambda_{\text{rv}} \frac{1}{N} \sum_{i=1}^{N} \left[f_\psi(y_i) \nabla_\theta \log p_\theta(y_i) \right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \left[-\lambda_{\text{fw}} \nabla_\theta \log p_\theta(x_i) \right] + \frac{1}{N} \sum_{i=1}^{N} \left[\lambda_{\text{rv}} \left(\log p_\theta(y_i) - f_\psi(y_i) \right) \nabla_\theta \log p_\theta(y_i) \right]
\end{aligned} \tag{18}$$

B.5 Lagrange multipliers gradients

The gradients of the Lagrange multipliers $\boldsymbol{\lambda}$ are relatively straightforward:

$$\begin{aligned}
\nabla_{\lambda_{\text{fw}}} \mathcal{L} &= \frac{1}{N} \sum_{i=1}^{N} \left[-\log p_\theta(x_i) \right] - \epsilon_{\text{fw}} \\
\nabla_{\lambda_{\text{rv}}} \mathcal{L} &= \frac{1}{N} \sum_{i=1}^{N} \left[\log p_\theta(y_i) - f_\psi(y_i) + \log \zeta_\psi \right] - \epsilon_{\text{rv}} \\
\nabla_{\lambda_{\text{prx}}} \mathcal{L} &= \frac{1}{N} \sum_{i=1}^{N} \left[-f_\psi(y_i) + \log \zeta_\psi \right] - \epsilon_{\text{prx}} \\
\nabla_{\lambda_{\text{u}}} \mathcal{L} &= \frac{1}{M} \sum_{i=1}^{M} \left[e^{f_\psi(y_i) - \log p_\theta(y_i)} \right] - 1 - \epsilon_\zeta \\
\nabla_{\lambda_\ell} \mathcal{L} &= 1 - \frac{1}{M} \sum_{i=1}^{M} \left[e^{f_\psi(y_i) - \log p_\theta(y_i)} \right]
\end{aligned} \tag{19}$$

Algorithm 2 Estimation of $\log \zeta_\psi$ according to Equation (5)

```
Require: An EBM $f_\psi$, a NF $p_\theta$, number of samples $m$
  Sample $\{x_i\}_1^m$ from $p_\theta$
  Compute the log probabilities $\{\log p_\theta(x_i)\}_1^m$
  Compute the EBM outputs $\{f_\psi(x_i)\}_1^m$
  $\log \zeta_\psi \leftarrow \text{LogSumExp}\left(\{f_\psi(x_i) - \log p_\theta(x_i)\}_1^m\right) - \log m$
  Return $\log \zeta_\psi$
```

Algorithm 3 Optimization algorithm for Equation ($\hat{\text{D}}$-DYN)

```
Require: Datasets $\mathcal{D}_{\text{data}}$, learning rates $\alpha$, batch size $B$, number of epochs $E$
  for $e = 1$ to $E$ do
      for each minibatch $\mathcal{B}_{\text{data}} \subseteq \mathcal{D}_{\text{data}}$ do
          StopGradient: Sample $|\mathcal{B}_{\text{data}}|$ samples $x$ from $p_\theta$
          StopGradient: Estimate $\log \zeta_\psi$ from Algorithm 2

          Compute the log probabilities $\log p_\theta(x)$
          Compute the output for the data samples $f_\psi(\mathcal{B}_{\text{data}})$ and $p_\theta$ samples $f_\psi(x)$
          Compute the log probability of the data under $p_\theta$ $\log p_\theta(\mathcal{B}_{\text{data}})$

          Solve $\boldsymbol{u}$ according to Equation (13)
          Compute $\nabla_\psi \mathcal{L}$ from Equation (15)
          Compute $\nabla_\theta \mathcal{L}$ from Equation (18)
          Compute $\nabla_{\boldsymbol{\lambda}} \mathcal{L}$ from Equation (19)
          Run optimizer step and update $\psi$, $\theta$ with gradient descent, and $\boldsymbol{\lambda}$ with gradient ascent
          StopGradient: Clamp $\boldsymbol{\lambda}$ to be non-negative $\boldsymbol{\lambda} \leftarrow \max\{\mathbf{0}, \boldsymbol{\lambda}\}$
      end for
  end for
  return $\psi^*$, $\theta^*$
```

B.6 Conditional formulation

We use the following formulation for conditional probabilities

$$\begin{aligned} \underset{\psi,\theta,\boldsymbol{\epsilon}}{\text{minimize}} \quad & \epsilon_{\text{fw}}^2 + \epsilon_{\text{rv}}^2 + \epsilon_{\text{EBM}}^2 \\ \text{subject to} \quad & -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\beta_i|u_i) \leq \epsilon_{\text{fw}} \\ & \frac{1}{N}\sum_{i=1}^{N} D_{\text{KL}}\left(p_\theta(\cdot|u_i) \parallel q_\psi(\cdot|u_i)\right) \leq \epsilon_{\text{rv}} \\ & -\frac{1}{N}\sum_{i=1}^{N}\left[\log f_\psi(\beta_i|u_i) - \log \zeta_\psi^{(\beta_i)}\right] \leq \epsilon_{\text{EBM}} \\ & 1 \leq \frac{1}{N}\sum_{i=1}^{N} \zeta_\psi^{(\beta_i)} \leq 1 + \epsilon_\zeta, \end{aligned} \tag{PIII}$$

where γ_i are generated by p_θ. Since each condition has a different partition function, we choose to take the empirical mean of partitions over the given conditions.

B.7 Negative NLL

Note that for the simplified problem

$$\begin{aligned} \underset{\psi,\epsilon}{\text{minimize}} \quad & \epsilon^2 \\ \text{subject to} \quad & f(\psi) \leq \epsilon, \end{aligned} \tag{PIV}$$

the NLL can never have a negative upper bound due to the update rule of u (Equation (13)) and the fact that $\boldsymbol{\lambda}$ are always non-negative. But in certain cases, a model can attain a negative NLL. In these cases we use a more generalized form of resilient learning as

$$\begin{aligned} \underset{\psi,\epsilon,\delta}{\text{minimize}} \quad & \epsilon^2 - \delta \\ \text{subject to} \quad & f(\psi) \leq \epsilon - \delta^2. \end{aligned} \tag{PV}$$

This revised problem now allows the upper bound of the NLL to be negative, as v is maximized which can make the upper bound in the constraint negative. In this case, all gradients are unchanged and u retains its closed form solution (Equation (13)). Solving δ also has a closed form solution

$$\begin{aligned} 0 &= \nabla_\delta \mathcal{L} \\ &= \nabla_\delta \left[\epsilon^2 - \delta + \lambda \left(f\left(\psi\right) - \epsilon + \delta^2\right)\right] \\ &= -1 + 2\lambda\delta \\ \delta &= \frac{1}{2\lambda}. \end{aligned} \tag{20}$$

Note that for the NLL and KL, the constraint is always active, preventing λ from being zero and v from being undefined. When using this form, we also initialize $\boldsymbol{\lambda}$ to be non-zero. The full revised problem is then

$$\begin{aligned} \min_{\psi,\theta,\boldsymbol{\epsilon},\boldsymbol{\delta}} \quad & \epsilon_{\text{fw}}^2 - \delta_{\text{fw}} + \epsilon_{\text{rv}}^2 + \epsilon_{\text{EBM}}^2 - \delta_{\text{EBM}} \\ \text{subject to} \quad & \text{NLL}\left(p_\theta\right) \leq \epsilon_{\text{fw}} - \delta_{\text{fw}}^2 \\ & D_{\text{KL}}\left(p_\theta \parallel q_\psi\right) \leq \epsilon_{\text{rv}} \\ & \text{NLL}\left(q_\psi\right) \leq \epsilon_{\text{EBM}} - \delta_{\text{EBM}}^2 \\ & 1 \leq \zeta_\psi \leq 1 + \epsilon_\zeta. \end{aligned} \tag{PVI}$$

B.8 Training in high dimensions

In high dimensions, the initial values of p_θ and f_ψ can be in different orders of magnitude. The initial outputs of the MLP f_ψ are usually small around zero while the log probabilities of p_θ are larger negative values the higher the dimension. These differences can lead to exploding gradients due to the exponent term in Equation (15). To mitigate this difference in order of magnitude, we add two mechanisms. First, we add a temperature T to the EBM, which makes it easier for the EBM to output high negative values

$$q_\psi\left(x\right) = \frac{1}{\zeta_\psi} e^{\frac{1}{T} f_\psi(x)}. \tag{21}$$

Then, before starting training, we warm start f_ψ to produce values similar to the log probabilities of p_θ in an iterative process. In each iteration, we sample from p_θ with the log probabilities, compute the f_ψ values on the samples and minimize the mean squared error between the the two sets of corresponding values. With these two additions, training is starting in a stable manner with less risk of exploding gradients.

We also replace the KL term between p_θ and q_ψ with a weighted KL using the same temperature T as

$$\begin{aligned} T \cdot D_{\text{KL}}\left(p_\theta \parallel q_\psi\right) &= T \mathbb{E}_{x\sim p_\theta}\left[\log p_\theta\left(x\right) - \frac{1}{T} f_\psi\left(x\right) + \log \zeta_\psi\right] \\ &= \mathbb{E}_{x\sim p_\theta}\left[T \log p_\theta\left(x\right) - f_\psi\left(x\right) + T \log \zeta_\psi\right], \end{aligned} \tag{22}$$

which avoids the large values of p_θ.

C Implementation Details

C.1 Model architectures

For NF and WGAN's generators, we use a neural spline flow (Durkan et al., 2019), implemented with the Zuko library[2]. For the EBMs and WGAN's critic we used a residual MLP comprising several residual blocks as described in Algorithm 4. Each residual block consists of several simple layers, as detailed in Algorithm 5. For the conditional case, we use the same architecture as the residual MLP, but replace the block with residual FiLM blocks (Perez et al., 2018), as detailed in Algorithm 6.

[2]https://zuko.readthedocs.io/stable/

Algorithm 4 Residual MLP

```
Require: Input dimension $d$, hidden dimension $h$, number of blocks $n$
Require: Input vector $x \in \mathbb{R}^d$
  $x' \leftarrow$ Linear $(x)$        ▷ Linear layer with output dimension $h$ with spectral normalization
  $x' \leftarrow$ SiLU $(x')$        ▷ Sigmoid Linear Unit activation
  for $i = 1$ to $n$ do
    $x' \leftarrow$ ResidualBlock $(x')$        ▷ Residual block as in Algorithm 5 with dimension $h$
  end for
  $x' \leftarrow$ Linear $(x')$        ▷ Linear layer with input dimension $h$ and output dimension 1
  Return $x'$
```

Algorithm 5 Residual block

```
Require: Dimension $d$
Require: Input vector $x \in \mathbb{R}^d$
  $x' \leftarrow$ Layer Normalization $(x)$
  $x' \leftarrow$ SiLU $(x')$        ▷ Sigmoid Linear Unit activation
  $x' \leftarrow$ Linear $(x)$        ▷ Linear layer with input and output dimension $d$ and spectral norm
  $x' \leftarrow$ Layer Normalization $(x)$
  $x' \leftarrow$ SiLU $(x')$        ▷ Sigmoid Linear Unit activation
  $x' \leftarrow$ Linear $(x)$        ▷ Linear layer with input and output dimension $d$ and spectral norm
  $x' \leftarrow \alpha x + x'$        ▷ Residual connection with trainable parameter $\alpha$
  $x' \leftarrow$ SiLU $(x')$
  Return $x'$
```

Algorithm 6 Conditional residual block

```
Require: Dimension $d$, number of FiLM layers $m$
Require: Input vector $x \in \mathbb{R}^d$, condition vector $c \in \mathbb{R}^k$
  $\gamma \leftarrow$ MLP $(c)$
        ▷ MLP with input dimension $k$ and output dimension $d$, with $m$ layers, SiLU activation

  $\beta \leftarrow$ MLP $(c)$
        ▷ MLP with input dimension $k$ and output dimension $d$, with $m$ layers, SiLU activation

  $x' \leftarrow$ MLP $(x)$
        ▷ MLP with input dimension $d$ and output dimension $d$, with 2 layers, SiLU activation

  $x' \leftarrow \gamma \odot x' + \beta$        ▷ $\odot$ represents element-wise multiplication
  $x' \leftarrow$ SiLU $(\alpha x + x')$        ▷ Residual connection with trainable parameter $\alpha$
  Return $x'$
```

C.2 Autoencoder

To reduce the dimensionality of CelebA, we use a autoencoder with a 100D latent space. We use the 64X64 version of the dataset, and scale the pixel values to be between -1 and 1. We used Algorithm 7 for the encoder and Algorithm 8 for the decoder, and train for 20 epochs with a batch size of 128 and a learning rate of 10^{-3} using Adam. The objective for the autoencoder was the mean square error (MSE) loss between the input and reconstructed output

Algorithm 7 Encoder

```
Require: Latent dimension $d$
Require: Input vector $x \in \mathbb{R}^D$
  $x' \leftarrow$ Conv2D$(x)$      ▷ Convolution layer with 32 channels, kernel size 4, stride 2 and padding 1
  $x' \leftarrow$ ReLU$(x')$
  $x' \leftarrow$ Conv2D$(x')$     ▷ Convolution layer with 64 channels, kernel size 4, stride 2 and padding 1
  $x' \leftarrow$ ReLU$(x')$
  $x' \leftarrow$ Conv2D$(x')$     ▷ Convolution layer with 128 channels, kernel size 4, stride 2 and padding 1
  $x' \leftarrow$ ReLU$(x')$
  $x' \leftarrow$ Conv2D$(x')$     ▷ Convolution layer with 256 channels, kernel size 4, stride 2 and padding 1
  $x' \leftarrow$ ReLU$(x')$
  $x' \leftarrow$ MLP$(x')$        ▷ MLP with output dimension $d$
  Return $x'$
```

Algorithm 8 Decoder

```
Require: Input vector $x \in \mathbb{R}^d$
  $x' \leftarrow$ MLP$(x)$          ▷ MLP with output dimension $256 \times 4 \times 4$
  $x' \leftarrow$ ConvT2D$(x')$     ▷ Transposed convolution: 64 channels, kernel size 4, stride 2, padding 1
  $x' \leftarrow$ ReLU$(x')$
  $x' \leftarrow$ ConvT2D$(x')$     ▷ Transposed convolution: 32 channels, kernel size 4, stride 2, padding 1
  $x' \leftarrow$ ReLU$(x')$
  $x' \leftarrow$ ConvT2D$(x')$     ▷ Transposed convolution: 3 channels, kernel size 4, stride 2, padding 1
  Return $x'$
```

C.3 Optimization and evaluation

All our code is written in Python using PyTorch[3] and Lightning[4]. For optimization we use Adam with parameters $(\beta_1, \beta_2) = (0.0, 0.9)$ and change the learning rate and batch size per experiment.

D Experimental details and results

D.1 Comparison on 40-component Gaussian mixture

In Section 4 we compare our method with the weighted sum formulation, NF and WGAN. The 2D distribution we used for the comparison, inspired by Midgley et al. (2023), consisted of a Gaussian mixture with 40 components. The center of each component is drawn uniformly between 0 and 1, and each component has a standard deviation of 0.05. For the train set we used 800 samples, and for the test set 10,000 samples. We trained all models for 100,000 iterations with full batches of the whole train set. For the Lagrange multipliers we used a learning rate of 10^{-4}, while the lagrange multipliers corresponding to the ζ_ψ normalization constraints were trained with a learning rate of 10^{-3} where $\epsilon_\zeta = 0.1$. Since some of the shapes have low entropies, we used the formulation in Appendix B.7 in all datasets.

For NF and our p_θ we used a neural spline flow (Appendix C.1) with 5 transforms and 8 bins. For our q_ψ and WGAN's critic we used a residual MLP (Algorithm 4) with 6 residual blocks and a hidden dimension of 64.

[3]https://pytorch.org/

[4]https://lightning.ai/docs/pytorch/stable/

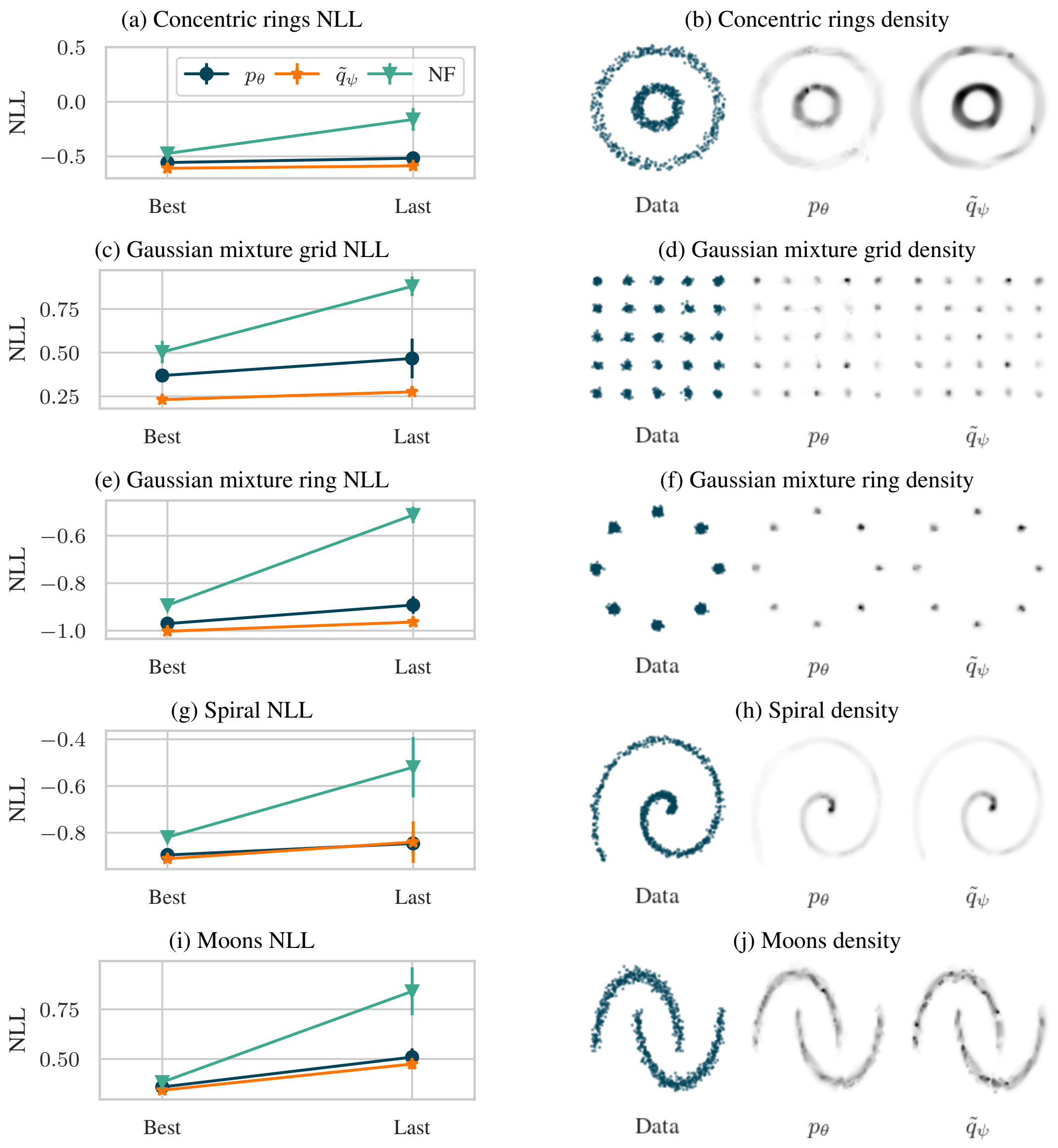

Figure 7: Our framework is able to accurately learn the density of various 2D datasets. The left column presents the lowest NLL each model achieved during training and the NLL at the end of training, demonstrating that our method outperforms NF in terms of values and consistency. The right column qualitatively displays the finite dataset (left) and the learned density values from of p_θ and the quasi-normalized $\tilde{p}_\psi$ (middle and right respectively). The qualitative density maps show that both models perfectly capture the shape and all modes.

In the comparison with the weighted sum formulation (Equation (P-W)) we used 25 different weight configurations, where each weight $w_{\text{back}}, w_{\text{for}} \in \{0.2, 0.5, 1.0, 2.0, 5.0\}$. For both formulations, we used a learning rate (LR) of 10^{-3} for p_θ and 10^{-4} for q_ψ, and we estimated ζ_ψ with $M = 1000$ samples from p_θ. We trained the NF comparison also with a LR of 10^{-3}.

When comparing with WGAN, we used a reduced LR for q_ψ and the critic of 10^{-5} to stabilize training, and trained p_θ and the generator with a LR of either 10^{-3} or 10^{-5}. We trained the WGAN with gradient penalty (Gulrajani et al., 2017) with a coefficient of 0.1.

D.2 Density estimation

In the density estimation experiments in Section 5.2 we tested 5 different distributions, which can be seen in Figure 7. For each dataset we used a train set of 1000 samples and a test set of 10000 samples and trained for 25000 full-batch iterations. We used a LR of 10^{-3} for p_θ and NF, and LR of 10^{-4} for q_ψ. We estimated ζ_ψ with $M = 1000$ samples from p_θ. For the Lagrange multipliers we used a learning rate of 10^{-4}, while the lagrange multipliers corresponding to the ζ_ψ normalization constraints were trained with a learning rate of 10^{-3}, where $\epsilon_\zeta = 0.1$. The initial values for the KL Lagrange multipliers was set to 0.01

To create the concentric rings dataset, we sample N points from a standard normal distribution, normalize them to have unit norm, and scale them by a either $r_1 = 0.2$ or $r_2 = 0.6$ with 0.5 probability. We then added uniform random noise $\mathcal{U}(0, 0.1)$ to each point. To create the Gaussian mixture grid, we created a Gaussian mixture distribution of a 5x5 grid of centers between -1 and 1 which we then scaled to have zero mean and unit variance, each with a standard deviation of 0.05. To create the Gaussian mixture ring, we created a Gaussian mixture distribution of a 8 components equidistant on a unit circle which we then scaled to have zero mean and unit variance. Each component has a standard deviation of 0.05. To create the spiral distribution, we sampled N points uniformly $t \sim \mathcal{U}(0, 10)$, and created points (x, y) with $x = t\cos(t)$ and $y = t\sin(t)$. We then added Gaussian noise with standard deviation of 0.02 to each point. For the moons dataset we used the "make moons" function from the scikit-learn package[5].

To draw the qualitative results, we divided the visualized region into 64×64 pixels and computed the density value of each pixel as $v = p(x) \cdot A$, with A being the area of each pixel. We then plot the filled contours of the density values.

D.3 Image sampling

We train our models on the encoded 100D space of CelebA (Liu et al., 2015). We use an autoencoder to reduce dimensionality as described in Appendix C.2. For p_θ and NF we used a neural spline flow (Appendix C.1) with 6 transforms and 16 bins. For q_ψ we used a residual MLP (Algorithm 4) with 5 residual blocks and a hidden dimension of 1024. For training our models, we used a temperature 0.01, as described in Appendix B.8, and used 5000 samples to estimate ζ_ψ. To save time, we sampled the 5000 samples only every 100 iterations. We trained all models for 40 epochs with batch size 128. The LR for p_θ and NF was 10^{-3}, and for q_ψ it was 10^{-6}. After every epoch, we evaluated the FID[6] with 50,000 samples and report the best FID that p_θ and NF achieved during training. For the Lagrange multipliers we used a learning rate of 10^{-6}, while the lagrange multipliers corresponding to the ζ_ψ normalization constraints were trained with a learning rate of 10^{-4}, where $\epsilon_\zeta = 100$.

D.4 Simulation-based inference

For SBI, we used two benchmark datasets: two moons (Greenberg et al., 2019), Gaussian mixture (Sisson et al., 2007). Each benchmark defines a prior and a simulator.

Gaussian mixture The parameter dimension is $\beta \in \mathbb{R}^2$ and the outcome dimension is $u \in \mathbb{R}^2$. The prior is a uniform distribution $p_\beta = \mathcal{U}\left([0.5, 1]^2\right)$ and the simulator is a Gaussian mixture of 2 components $p(x|\beta) = 0.5\mathcal{N}(\beta, I) + 0.5\mathcal{N}(\beta, 0.01I)$, where I is the identity matrix.

Two moons The parameter dimension is $\beta \in \mathbb{R}^2$ and the outcome dimension is $u \in \mathbb{R}^2$. The prior is a uniform distribution $p_\beta = \mathcal{U}\left([-1, 1]^2\right)$. The simulator is

$$u|\beta = (r\cos(\alpha) + 0.25, r\sin(\alpha)) + \left(-|\beta_1 + \beta_2| / \sqrt{2}, (-\beta_1 + \beta_2) / \sqrt{2}\right),$$

where $\alpha \sim \mathcal{U}\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ and $r \sim \mathcal{N}\left(0.1, 0.01^2\right)$.

[5]https://scikit-learn.org/

[6]We adapted the code from https://github.com/mseitzer/pytorch-fid.

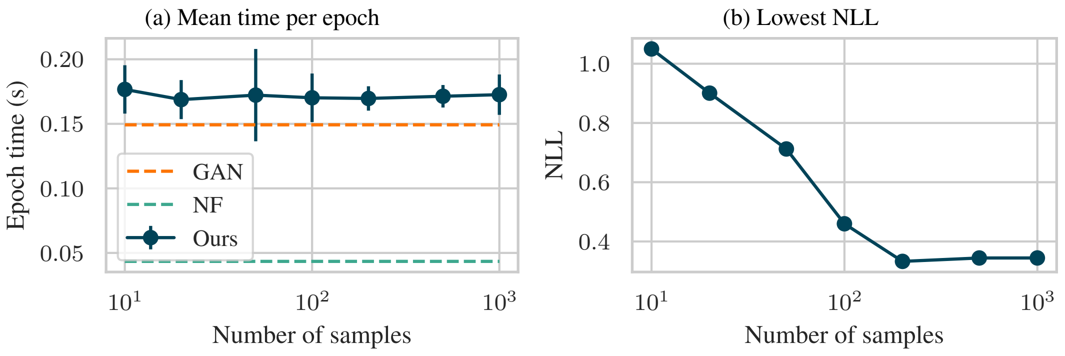

Figure 8: How the number of samples for the estimation of ζ_ψ with importance sampling (Equation (5)) affect training. (a) The computation time for our method is slower than NF and GANs, but does not scale much with more samples. (b) Using more samples improves the NLL.

To evaluate the C2ST, we generate 5000 samples from the posterior using rejection sampling. For each dataset we defined a test condition u_0 and generated samples by sampling from the prior, running the simulator and keep the first 5000 samples that were close to u_0. For the Gaussian mixture we set $u_0 = (0.75, 0.75)$ and for the two moons $u_0 = (0, 0)$. We also generate 5000 samples from the trained models by conditioning on u_0. We split the total of 10000 to a train and test set with a 0.3 split and train a classifier[7] with positive labels for the ground truth points and negative labels for the model points, and use its accuracy on the test set as the C2ST metric.

To generate from q_ψ, we followed the MCMC sampling in Equation (8). We used 2000 steps with a step size of 10^{-4}. The initial points were sampled from p_θ.

For p_θ and NF we used with 4 transforms and 8 bins. For q_ψ we used a different architecture for each dataset. For the Gaussian mixture, we used a conditional residual MLP (Algorithm 6) with 6 residual blocks and a hidden dimension of 64 and 2 FiLM layers. For the two moons, we used 3 residual blocks, each with 128 channels and 2 FiLM layers. We trained with different number of simulations (data points), but used a batch size of 1000 for all experiments. We used the negative NLL formulation (Appendix B.7) for both datasets, with LR of 10^{-4} for p_θ and NF, and LR of 10^{-5} for q_ψ. The Lagrange multipliers LR was set to 10^{-6}, while the lagrange multipliers corresponding to the ζ_ψ normalization constraints were trained with a learning rate of 10^{-2}, where $\epsilon_\zeta = 30$. The initial values for the KL Lagrange multipliers was set to 0.01.

E Compute Time

Our method requires generating M samples to estimate the partition function ζ_ψ with importance sampling (Equation (5)). Here we study how the number of samples M affects the training time and performance. We used our training data from the 40 components Gaussian mixture experiment from Section 5.1. We trained all models on an NVIDIA A100 GPU and averaged the time per epoch over 100,000 epochs.

Figure 8a shows that our method is indeed slower than NF, but not much slower than GANs. It is also interesting to observe that the compute time barely increases with more samples. This is because the importance sampling does not require gradients, and with the stop gradient (Algorithm 3), the computation is efficient. For a much larger number of samples, or higher dimensions, we expect the time per epoch to increase.

We also examine how the number of samples affects the quality of learning. In Figure 8b we plot the best NLL achieved during training per number of samples. It is clear that with too few samples, the inaccuracy in the estimation harms learning.

[7]HistGradientBoostingClassifier from https://scikit-learn.org