Can We Estimate The Entropy Of Arbitrary Distributions Known Up To A Normalization Constant? A Tale of Stein Variational Gradient Descent Scalability

Anonymous Author(s)

Abstract

Computing the differential entropy for distributions known up to a normalization constant is a challenging problem with significant theoretical and practical applications. Variational inference is widely used for scalable approximation of densities from samples, but is under-explored when *only unnormalized densities are available*. This setup is more challenging as it requires variational distributions that (1) leverage the unnormalized density, (2) are expressive enough to capture complex target distributions, (3) are computationally efficient, and (4) facilitate easy sampling. To address this, Messaoud et al. [2024] introduced **P-SVGD**, a particle-based variational method using Stein Variational Gradient Descent dynamics. However, we show that **P-SVGD** scales poorly to high-dimensional distributions. We propose **MET-SVGD**, an extension of **P-SVGD** that scales efficiently to high-dimensional settings with convergence guarantees. **MET-SVGD** incorporates (1) a sufficient condition for SVGD invertability, (1) optimized parameterizations of SVGD updates, (2) a Metropolis-Hastings acceptance step for asymptotic convergence guarantees and enhanced expressivity, and (3) a correction term for better scalability. Our method bridges the gap between Metropolis-Hastings, particle-based sampling and parametrized variational inference techniques, achieving SOTA results on scaling SVGD to high-dimensional spaces. We significantly outperform **P-SVGD** on entropy estimation, Maximum Entropy Reinforcement Learning, and image generation with Energy-Based Models benchmarks. Also, we will release an open-source **MET-SVGD** library (`https://tinyurl.com/2esyfx8j`).

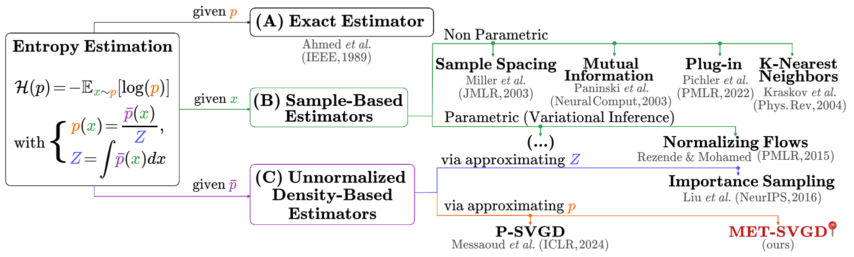
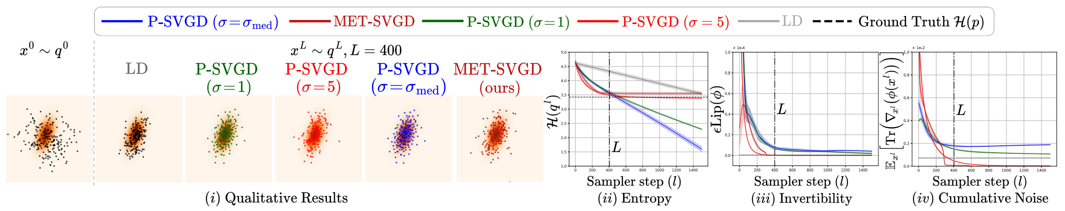

given p
(A) Exact Estimator
Ahmed *et al.* (IEEE, 1989)
Entropy Estimation
$\mathcal{H}(p) = -\mathbb{E}_{x\sim p}[\log(p)]$
with $\begin{cases} p(x) = \frac{\bar{p}(x)}{Z}, \\ Z = \int \bar{p}(x)dx \end{cases}$
given x
(B) Sample-Based Estimators
Non Parametric
Sample Spacing
Miller *et al.* (JMLR, 2003)
Mutual Information
Paninski *et al.* (Neural Comput, 2003)
Plug-in
Pichler *et al.* (PMLR, 2022)
K-Nearest Neighbors
Kraskov *et al.* (Phys. Rev, 2004)
Parametric (Variational Inference)
(...)
Normalizing Flows
Rezende & Mohamed (PMLR, 2015)
given $\bar{p}$
(C) Unnormalized Density-Based Estimators
via approximating Z
Importance Sampling
Liu *et al.* (NeurIPS, 2016)
via approximating p
P-SVGD
Messaoud *et al.* (ICLR, 2024)
MET-SVGD
(ours)

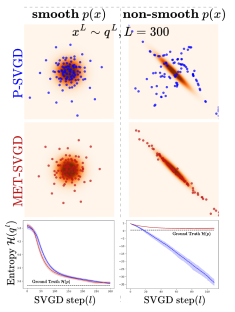
Figure 1: **MET-SVGD** is a new variational inference approach for entropy estimation of distributions known up to a normalization constant. It extends **P-SVGD** [Messaoud et al., 2024] to high-dimensional spaces by adressing its key limitations (see Fig. 2).

1 Introduction

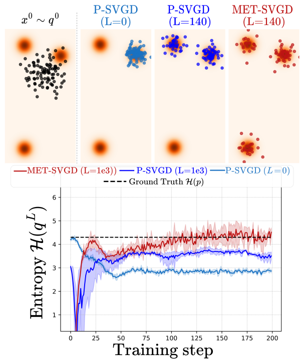
The differential entropy [Cover, 1999, Shannon, 2001] of a d-dimensional random variable X with a probability density function $p(x) = \bar{p}(x)/Z$ is $\mathcal{H}(p) = -\mathbb{E}_{x\sim p(x)}[\log p(x)] = -\int p(x)\log p(x)dx$, with $Z = \int \bar{p}(x)dx$ being the normalization constant. The differential entropy plays a central role in information theory, signal processing, and machine learning [Tarasenko, 1968, Learned-Miller and III, 2003, Wulfmeier et al., 2015, Liu et al., 2022a, Hino and Murata, 2010, Rubinstein and Kroese, 2004, Mannor et al., 2005]. However, estimating it is challenging as a closed-form expression is only available for a limited class of distributions (*e.g.*, Gaussians). In practice, only samples $x \sim p$ or the unormalized density $\bar{p}$ are given (Fig. 1). While significant progress has been made on estimating entropies from samples [Beirlant et al., 1997, Paninski, 2003], settings where only the unnormalized density is available remain largely under-explored. These settings arise in numerous



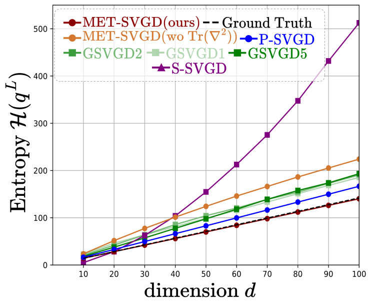
(A) **Limitation 1: P-SVGD** suffers from high sensitivity to the RBF kernel bandwidth σ. *(i)* Particles *qualitatively* converge under all sampling schemes, including Langevin Dynamics (LD). *(ii)* However, *quantitatively*, entropy only matches the ground-truth for specific σ values in **P-SVGD**. *(iii)* This is not due to violating the SVGD update invertibility as claimed by Messaoud et al. [2024]; our proposed sufficient condition (Prop. 3.2) is always satisfied. *(iv)* Instead, poor entropy convergence in LD and **P-SVGD** with $\sigma \in \{\sigma_{\text{med}}, 1\}$ arises from the cumulative trace term in Eq. 3, which stabilizes to small non-zero values, causing entropy divergence over time.



(B) **Limitation 2: P-SVGD** suffers from poor convergence to non-smooth targets.



(C) **Limitation 3: P-SVGD** suffers from mode collapse. L is the number of SVGD steps.



(D) **Limitation 4: P-SVGD** scales poorly to high dimensional spaces. Benchmark from Liu et al. [2022b]. MET-SVGD(wo. Tr(∇^2)) refers to the variant without the correction term.

Figure 2: **P-SVGD** limitations in **P-SVGD**: (A) high sensitivity to kernel variance (σ), (B) poor convergence to non-smooth targets, (C) sampling mode collapse and (D) limited scalability to high-dimensional spaces. **MET-SVGD** addresses these shortcomings and achieves improved accuracy and scalability in entropy estimation. Our contributions are illustrated in Fig. 3C.

machine learning application domains, including Energy-Based Models (EBMs) [LeCun et al., 2006], Maximum Entropy Reinforcement Learning (MaxEnt RL) [Haarnoja et al., 2018], and Bayesian inference [Harney, 2003][Hernandez-Lobato et al., 2014].

A common approach for entropy estimation in *unnormalized density setups* is to approximate the normalization constant Z using sampling-based techniques, *e.g.*, importance sampling [Cantwell, 2022]. Such estimates, however, suffer from high variance in high-dimensional spaces. Recently, Messaoud et al. [2024] introduced Parametrized Stein Variational Gradient Descent (**P-SVGD**), which leverages Stein Variational Gradient Descent (SVGD) sampler [Liu and Wang, 2016] to construct a variational distribution q^L from $\bar{p}$. SVGD updates a set of interacting particles $\{x_i\}_{i=1}^M$ using a deterministic velocity field $\phi(\cdot)$ that balances a gradient force and a repulsive one:

$$\phi(x_i^l) = \mathbb{E}_{x_j^l \sim q^l} \left[\kappa(x_i^l, x_j^l) \nabla_{x_j^l} \log p(x_j^l) + \nabla_{x_j^l} \kappa(x_i^l, x_j^l) \right], \tag{1}$$

following the update rule $x_i^{l+1} = x_i^l + \epsilon\phi(x_i^l)$. ϵ is the step-size, q^l is the particles distribution at step $l \in [1, L]$ and $\kappa(\cdot, \cdot)$ is typically an RBF kernel with variance σ^2, *i.e.*, $\kappa(x_i, x_j) = \exp(-||x_i - x_j||^2/2\sigma^2)$. **P-SVGD** derives a closed form expression of q^l at every step l including the last step L:

$$\log q^L(x_i^L) = \log q^0(x_i^0) - \epsilon \sum_{l=0}^{L-1} \sum_{i \neq j=0}^{M-1} \frac{\kappa(x_j^l, x_i^l)}{M\sigma^2} \left(d - \frac{\|x_i^l - x_j^l\|^2}{\sigma^2} - (x_i^l - x_j^l)^\top \nabla_{x_j^l} \log p(x_j^l) \right) + \mathcal{O}(\epsilon^2). \tag{2}$$

q^L can approximate a broad class of densities under mild assumptions [Villani et al., 2009], enabling accurate estimation of $\mathcal{H}(p)$ with compelling results in MaxEntr RL. Despite its promise, we show that **P-SVGD** has several limitations including sensitivity to SVGD hyperparameters, mode collapse, poor convergence to non-smooth targets and limited scalability in high-dimensional spaces (Fig. 2). To address these challenges, we introduce **MET-SVGD**, an extension of **P-SVGD** that scales to high-dimensional distributions with improved accuracy and convergence guarantees. Contrary to **P-SVGD**'s claim that sensitivity to the RBF kernel bandwidth is due to violating the invertibility assumption, we show that invertibility is actually satisfied, and that divergence stems from accumulating small noise over time; **MET-SVGD** mitigates this by learning step-wise kernel bandwidths and

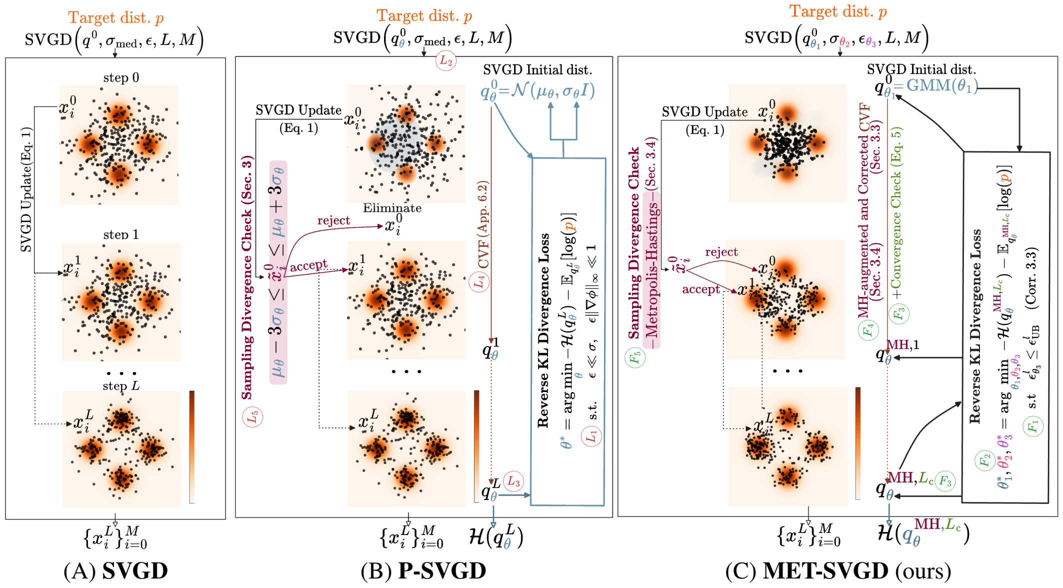

Target dist. p
SVGD$\left(q^0, \sigma_{\text{med}}, \epsilon, L, M\right)$
step 0
step 1
step L
SVGD Update(Eq. 1)
$\{x_i^L\}_{i=0}^M$
(A) **SVGD**
SVGD$\left(q_\theta^0, \sigma_{\text{med}}, \epsilon, L, M\right)$
SVGD Initial dist.
$q_\theta^0 = \mathcal{N}(\mu_\theta, \sigma_\theta I)$
SVGD Update (Eq. 1)
Eliminate
reject
accept
Sampling Divergence Check (Sec. 3)
$\mu_\theta - 3\sigma_\theta \leq x_i^0 \leq \mu_\theta + 3\sigma_\theta$
CVF (App. 6.2)
Reverse KL Divergence Loss
$\theta^* = \arg\min_\theta -\mathcal{H}(q_\theta^L) - \mathbb{E}_{q_\theta^L}[\log(p)]$
s.t. $\epsilon \ll \sigma, \quad \epsilon\|\nabla\phi\|_\infty \ll 1$
$\mathcal{H}(q_\theta^L)$
(B) **P-SVGD**
SVGD$\left(q_{\theta_1}^0, \sigma_{\theta_2}, \epsilon_{\theta_3}, L, M\right)$
$q_\theta^0 = \text{GMM}(\theta_1)$
Sampling Divergence Check –Metropolis-Hastings–(Sec. 3.4)
MH-augmented and Corrected CVF (Sec. 3.4) (Sec. 3.3)
+Convergence Check (Eq. 5)
$q_\theta^{\text{MH},1}$
q_θ^{MH,L_c}
$\theta_1^*, \theta_2^*, \theta_3^* = \arg\min_{\theta_1,\theta_2,\theta_3} -\mathcal{H}(q_\theta^{\text{MH},L_c}) - \mathbb{E}_{q_\theta^{\text{MH},L_c}}[\log(p)]$
s.t. $\epsilon_{\theta_3}^l \leq \epsilon_{\text{UB}}^l$ (Corr. 3.3)
$\mathcal{H}(q_\theta^{\text{MH},L_c})$
(C) **MET-SVGD** (ours)

Figure 3: Novelty over **P-SVGD**. The same indices are used for **P-SVGD** Limitations (L) and corresponding Fixes (F) in **MET-SVGD**. [L_1–F_1] A single sufficient condition for both global invertibility of the SVGD update and log-det approximation, replacing two informal independent ones. [L_2–F_2] End-to-end learning of kernel bandwidth and step-size via reverse KL minimization, mitigating hyperparameter sensitivity. [L_3–F_3] adaptive number of sampling steps L_c with Stein Identity-based convergence monitoring, enabling adaptability to complex distributions. [L_4–F_4] Efficient restoration of the missing trace-of-Hessian term in Eq. 14. [L_5–F_5] Replacement of the heuristic divergence control with an MH step for guaranteed asymptotic convergence.

step-sizes. Moreover, **P-SVGD**'s sufficient condition for invertibility, derived from the implicit function theorem, ensures only *local* invertibility, while *global* invertibility is required for the derivation; We propose a new condition, based on the Banach theorem (Theorem 6.3), for *global* invertibility. Also, while **P-SVGD** introduces two separate and informal conditions on the SVGD step-size for, respectively, invertibility and log-det approximation (Proposition 3.2 and Theorem 3.1 in Messaoud et al. [2024]), **MET-SVGD** consolidates both into one principled constraint. In addition, for computational efficiency, **P-SVGD** omits a trace of the Hessian term from Eq. 2, which we show is a major cause behind poor scalability as it's only valid asymptotically, *i.e.*, it's incorrect in the finite particle setup; **MET-SVGD** efficiently restores the missing term using Hutchinson's estimator. Besides, for sampling divergence control, rather than relying on **P-SVGD**'s particle truncation heuristic, which we show promotes mode collapse, **MET-SVGD** introduces a Metropolis–Hastings (MH) Robert et al. [2004] correction step that both improves the expressivity of the induced likelihood and guarantees asymptotic convergence. Finally, instead of fixing the number of sampling steps as in **P-SVGD**, **MET-SVGD** adaptively determines the number of steps using the Stein Identity Liu et al. [2016] as a convergence criterion, which is especially important in applications where adaptability to different complex distributions is required, *e.g.*, in RL, each state induces a different actions distribution.
MET-SVGD achieves SOTA performance on SVGD scalability benchmarks. Additionally, it significantly outperforms **P-SVGD** on image generation (20 vs 88 FID) and MaxEnr RL tasks (2.3% and 1.5% better final returns on Walker2d-v2 and Humanoid-v2 [Brockman et al., 2016], respectively). By bridging variational inference, SVGD, and MH methods, **MET-SVGD** sets a novel framework for entropy estimation from unnormalized densities and scalable sampling.

2 Related Work

We review Variational Inference (VI) followed by SVGD, P-SVGD and MH.
VI [Fox and Roberts, 2012] approximates the target p, via a simpler-to-sample-from distribution q^* from a predefined family $\boldsymbol{Q}$, by maximizing the reverse KL-divergence *i.e.*, $q^* = \arg\max_{q \in \boldsymbol{Q}} D_{\text{KL}}(q \| p)$. More expressive $\boldsymbol{Q}$ families yield better approximations.
SVGD [Liu and Wang, 2016] is a sampler with update rule in Eq. 1. Classically, the RBF kernel is used, with bandwidth set to $\sigma_{\text{med}} = \text{median}\{\|x_i^l - x_j^l\|\}_{i,j=1}^M / \log M$. Unlike classical VI, SVGD can sample from arbitrary complex distributions under mild conditions [Villani et al., 2009].
P-SVGD Messaoud et al. [2024] is a VI approach for entropy estimation from unnormalized densities.

It computes the density of the SVGD particles q^L by sequentially applying the Change of Variable formula (CVF) [Devore et al., 2012] over L steps under an invertibility condition derived via the implicit function theorem (App. 6.3), *i.e.*, $\log q^{l+1}(x^{l+1}) = \log q^l(x^l) - \log|\det(I + \epsilon\nabla_{x^l}\phi(x^l))|$. To avoid computing the full Jacobian, two approximations are used: (1) If $\epsilon\|\nabla_{x^l}\phi(x^l)\|_\infty \ll 1$, the Jacobian determinant is reduced to its trace following Jacobi's formula (App. 6.4) and leading to

$$\log q^L(x^L) = \log q^0(x^0) - \epsilon \sum_{l=0}^{L-1} \mathbb{E}_{x_j^l \sim q^l}\left[\mathrm{Tr}\left(\partial\bar{\phi}(x^l, x_j^l)/\partial x^l\right)\right] + \mathcal{O}(\epsilon^2), \tag{3}$$

where $\bar{\phi}(x^l, x_j^l)$ is the contribution of particle x_j^l to the update of particle x^l and the velocity $\phi(x^l) = \mathbb{E}_{x_j^l}[\bar{\phi}(x^l, x_j^l)]$ is defined in Eq. 1; (2) With an RBF kernel, the trace term is approximated using only first-order derivatives, resulting in Eq. 2. Note that, for efficiency, the authors omit a trace-of-Hessian term $\mathrm{Tr}(\nabla^2_{x^l}\log p(x^l))$ by sampling $x_j^l \neq x^l$ to approximate the expectation inside ϕ^l. Additionally, the authors show that *learning the initial distribution* $q_\theta^0 = \mathcal{N}(\mu_\theta, \sigma_\theta I)$ parameters via minimizing $D_{KL}(q_\theta^L||p)$ and *preventing samples divergence* by constraining the particles to be within few standard deviations of q_θ^0's mean are essential for scaling (Fig. 3B). In this work, we endow SVGD with convergence guarantees by augmenting the step-wise update (Eq. 1) with an MH step.

MH Robert et al. [2004] is an MCMC [Chib, 2001] sampling method involving two steps: (1) propose a new state $\tilde{x}^l$ from a proposal distribution $q^l(\tilde{x}^l|x^{l-1})$, (2) accept the proposal with probability

$$\alpha^l = \alpha(x^{l-1}, \tilde{x}^l) = \min\left[1, \left(p(\tilde{x}^l)/p(x^{l-1})\right) \cdot \left(q^l(x^{l-1}|\tilde{x}^l)/q^l(\tilde{x}^l|x^{l-1})\right)\right]. \tag{4}$$

If accepted, the proposal is set to $\tilde{x}^l$; $x^l = \tilde{x}^l$, else the current state is retained $\tilde{x}^l = x^{l-1}$. MH ensures asymptotic convergence to the target with sufficient steps [Roberts and Rosenthal, 2004]. We cover additional material on entropy, SVGD, MCMC and connection to residual flows in App. 7.

3 Approach

Similarly to **P-SVGD**, **MET-SVGD** is a VI-based method for computing the entropy of distributions p known up to a normalization constant, *i.e.*, it approximate p with a tractable, sample-efficient distribution and estimate $\mathcal{H}(p)$ using the entropy of this distribution. **MET-SVGD** introduces a series of optimizations to address **P-SVGD**'s key limitations as illustrated in Fig. 3C: [L_1-F_1] **P-SVGD** introduces two informal independant constraints on the step-size including a local invertibility one, although CVF requires global invertibility; **MET-SVGD** unifies these constraints into a single principled one satisfying global invertibility (Sec. 3.1). [L_2-F_2] **P-SVGD** exhibits high sensitivity to hyperparameters (Fig. 2A-*iii*) with no tuning guidelines; we show that this is due to the accumulation of noise in the trace term of Eq. 3, leading to entropy divergence and mitigate this by learning the SVGD hyperparameters end-to-end via reverse KL-divergence minimization (Sec. 3.2). [L_3-F_3] **P-SVGD** suffers from poor convergence to non-smooth targets and sampling mode collapse (Fig. 2B and Fig. 2C), due to its divergence control heuristic and the absence of convergence guarantees in the finite particle regime; **MET-SVGD** replaces this heuristic with a MH step, guaranteeing asymptotic convergence independently from the number of particles (Sec. 3.4). [L_4-F_4] P-SVGD's omission of the trace-of-Hessian correction term limits its scalability to high-dimensional spaces (Fig. 2D) which **MET-SVGD** efficiently restores as explained in Sec. 3.3. [L_5-F_5] P-SVGD uses a fixed number of SVGD steps L, which may be insufficient for convergence; **MET-SVGD** adaptively determines the number of steps L_c using the Stein Identity as a convergence criterion (Sec. 3.2).

3.1 Conditions On The SVGD Step-Size For Invertibility and log-det Approximation

In **P-SVGD**, Eq. 2 was derived by (1) leveraging the CVF (App. 6.2) assuming invertibility, and (2) approximating the log-det term in the CVF with an efficient trace one. Thes steps introduce two conditions on the SVGD step-size: (1) $\epsilon \ll \sigma$ and (2) $\epsilon||\nabla_{x^l}\phi(x^l)||_\infty \ll 1$. However, these conditions present two major issues: (1) Both are informal (use of $\ll$); in practice, ϵ is simply set to an arbitrarily small value, hoping that both constraints hold, which may not be true and often results in more steps than necessary. (2) The step-size condition only guarantees *local* invertibility, whereas the CVF requires *global* invertiblity. To address this, we extend a sufficient condition for invertible residual networks (Behrmann et al. [2019]) to SVGD. We also derive a precise condition on ϵ for the log-det approximation (Proposition 3.2) and unify both into a single efficient bound (Corollary 3.3).

Proposition 3.1 (Sufficient condition for global SVGD invertibility)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}^d$ with $f = (f^1 \circ \cdots \circ f^L)$ denote a sequence of SVGD updates with $f^l = I + \epsilon\phi^l$. We denote by Lip(ϕ^l) the Lipschitz constant of the velocity ϕ^l at step l. f is invertible if: $\epsilon\,$Lip$(\phi^l) < 1$, for all $l \in [0, L-1]$.*

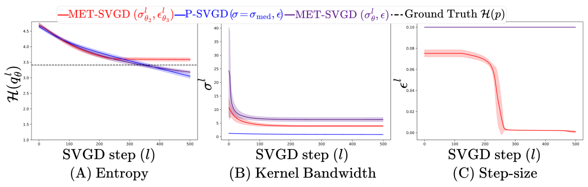


Figure 4: (A) Entropy, (B) RBF kernel bandwidth, and (C) step-size across SVGD steps. Target is the Gaussian target from Fig. 2A.

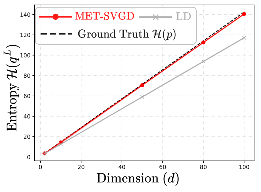


Figure 5: **MET-SVGD** is less sensitive to the *Tr* approximation than LD. Target is a slanted Gaussian (details in App. 6).

The proof is in App. 8.1. Using the mean value theorem (App.6.6), we estimate this Lipschitz constant as $\|\nabla\phi^l\|_2 = \max_{x^l} \|\nabla_{x^l}\phi(x^l)\|_2$ with $\|\cdot\|_2$ being the spectral norm (largest singular value).

Proposition 3.2 (Sufficient condition for log-det Approximation). *Let $\phi^l : \mathbb{R}^d \to \mathbb{R}^d$, $\log|\det(I + \epsilon\nabla\phi^l)| = \epsilon Tr(\nabla\phi^l)$ if $\epsilon\,|\lambda_{max}(\nabla\phi^l)| < 1$ for all $l \in [0, L-1]$, with λ_{max} being the largest eigenvalue value and ∇ the gradient operator w.r.t the input.*

Corollary 3.3. *The distribution induced by the SVGD update (Eq. 1) using an RBF kernel is given by Eq. 2 if $\epsilon < \epsilon^l_{UB} = 1/\sup_x \sqrt{Tr(\nabla\phi^l(x)\nabla\phi^{l,T}(x)}$ for all $l \in [0, L-1]$.*

Proof Sketch: Given $A \in \mathbb{R}^{d\times d}$, the following always holds: $|\lambda_{max}(A)| \le \|A\|_2 \le \sqrt{Tr(AA^T)}$. Proof is in App. 8.3, where we also show that $Tr(AA^T)$ can be efficiently computed using only first-order derivatives and vector dot products, making this condition practical.

3.2 Optimized SVGD Parameters

A major drawback of **P-SVGD** is its high sensitivity to the RBF kernel bandwidth σ, which Messaoud et al. [2024] attribute to violation of the invertibility of the SVGD update rule (Eq. 1): In a 2-d Gaussian target setup (reproduced in Fig. 2A), they show that, paradoxically, although SVGD and Langevin Dynamics (LD) (update rule in App. 6.1) *qualitatively* converge to the target, *i.e.*, the particles reach high-density regions (Fig. 2A-i), the entropy estimate only converges for specific σ values, *e.g.*, $\sigma = 5$ (Fig. 2A-ii). The authors hypothesise that this is due to LD being inherently non-invertible and SVGD being invertible only for certain σ values. This is incorrect: in Fig. 2A-iii, we show that the step-size condition from Corollary 3.3 is always satisfied. Instead we show that the poor quantitative convergence of $\mathcal{H}(q^L)$ to $\mathcal{H}(p)$ arises from the cumulative residual noise in the trace term of Eq. 3, *i.e.*, $\mathbb{E}_{x^l\sim q^l}[\text{Tr}(\nabla_{x^l}\phi(x^l))] \nrightarrow 0$ as $l \to \infty$ (Fig. 2A-iv), resulting in a quasi-linear growth in the entropy with the number of steps (Fig. 2A-ii). To address this, we propose a principled procedure for SVGD hyperparameter selection: leveraging the closed-form expression of $q^{L_c}_\theta$, **MET-SVGD** learns a step-wise kernel bandwidth $\sigma^l_{\theta_2}$ and step-size $\epsilon^l_{\theta_3}$ alongside the initial distribution $q^0_{\theta_1}$ by minimizing the reverse KL-divergence:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{x^{L_c}\sim q^{L_c}_\theta}[\log q^{L_c}_\theta(x^{L_c}) - \log p(x^{L_c})], \quad \text{s.t.} \quad \epsilon^l_{\theta_3} \le \epsilon^l_{\text{UB}} \quad \forall l \in [0, L_c - 1],$$

with ϵ^l_{UB} being the upper-bound from Corollary. 3.3 and $\theta = \{\theta_i\}^3_{i=1}$. Besides, we propose an efficient convergence check enabling an adaptive number of steps L_c, rather than fixing it a priori. Beyond improving stability and accuracy, **MET-SVG** provides a principled general framework for systematically selecting samplers hyperparameter, addressing a broader gap in the literature.

Kernel and Parameters. We observe that learning a step-wise RBF kernel bandwidth σ_{θ_2} led to significantly better convergence to the target density compared to the median heuristic, as demonstrated by the difference in trends in Fig. 4A. We also explored the Bilinear kernel and Deep Kernel Embedding Functions (DKEF) (details in App. 8.6 and App. 8.7), but found that the RBF kernel strikes a more favorable balance between flexibility and efficiency.

Step-Size. Learning the kernel bandwidth alone is insufficient to ensure convergence to ground-truth entropy, *i.e.*, the cumulative trace term $\mathbb{E}_{x^l\sim q^l}[\epsilon Tr(\nabla_{x^l}\phi(x^l))]$ does not necessarily vanish as $l \to \infty$. In App. 10, we show, via Taylor expansion, that this term corresponds to a 4th-degree polynomial which convergence to zero requires the existence of at least one real root. This is a non-trivial and fragile condition, as the coefficients of this polynomial depend on the particle positions during training. To address this, we propose learning a step-wise step-size $\epsilon^l_{\theta_3}$ which can be flexibly modulated by the model for convergence. In the setup of Fig. 4C, the step-size eventually becomes 0.

Number of Steps, L, in **P-SVGD**, is fixed, which does not guarantee convergence to the target. To address this, **MET-SVGD** employs an adaptive number of steps L_c determined dynamically using the Stein Identity (SI) as a step-wise convergence check (Proposition 3.4). This is a big advantage over

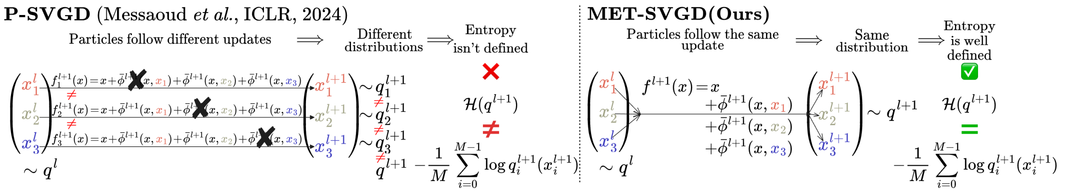

P-SVGD (Messaoud et al., ICLR, 2024)
Particles follow different updates
Different distributions
Entropy isn't defined
MET-SVGD(Ours)
Particles follow the same update
Same distribution
Entropy is well defined

Figure 6: Correction Term. In **P-SVGD**, excluding the updated particle (crossed) when approximating the expectation in Eq. 2 is incorrect: particles undergo different updates leading them to follow different distributions which makes the estimation of the target's entropy incorrect. **MET-SVGD** incorporates the missing term in the entropy estimate using Hutchinson's estimator (Sec. 3.3).

MCMC samplers, which typically rely on approximate chain statistics that are more costly and less reliable to detect convergence [Robert, 1999]. We derive a practical form of the SI that depends on $Tr(\nabla_{x^l}\phi(x^l))$, hence only on 1st-order derivatives and vector dot products, as we show in App. 9.3.

Proposition 3.4. *The Stein identity $SI(q^l, p)$ at every step l of the SVGD update is computed as:*

$$SI(q^l_\theta, p) = \sqrt{\mathbb{E}_{x^l}\left[\phi_\theta(x^l)^T\nabla_{x^l}\log p(x^l) + Tr(\nabla_{x^l}\phi_\theta(x^l))\right]} \tag{5}$$

Complexity. Since **P-SVGD** already learns $q^0_{\theta_1}$, additionally learning $\sigma^l_{\theta_2}$ and $\epsilon^l_{\theta_3}$ using lightweight deepnets introduces little overhead during training. Inference is reduced to only a forward pass through the deepnets. In contrast, grid search is significantly more expensive and suboptimal as it relies on a finite set of candidate values evaluation by repeated experiments. σ_{med} also scales quadratically with the number of particles, as it requires computing all pairwise distances. As for L_c, backpropagating through every SVGD update can cause memory issues and the computational graph grows with every step. To mitigate this, in large-scale experiments, we attach updates to the graph only every k steps, assuming small difference in particles positions between updates.

3.3 Corrected Derivation of q^l_θ

As explained in Sec. 3, **P-SVGD** approximate the expectation over particles in Eq. 3 by excluding the updated particle itself ($x^l \neq x^l_j$), so that the trace term can be estimated using only first-order derivatives and thereby avoiding explicit Hessian computation. While this approximation is valid asymptotically, it breaks in the finite-particle regime. As illustrated in Fig. 6, this translates into inconsistent updates across particles, as the crossed term is missing, inducing a different distribution for every particle, and making the entropy ill-defined. Additionally, in **P-SVGD**, the expectation over particles is handled inconsistently: the density estimation in Eq. 2 excludes the updated particle, while the update rule in Eq. 1 includes all particles resulting in a mismatch between the sampling process and its corresponding density computation. This inconsistency is a key source of **P-SVGD**'s poor scalability as shown in Fig. 2D (orange vs brown). To address this, **MET-SVGD** corrects the approximation by adding the missing term to the entropy using (1) the Hutchinson estimator [Hutchinson, 1989] and (2) the double differentiation trick [Song et al., 2020]: $Tr\left(\nabla^2_{x^l_i}\log p(x^l_i)\right) \overset{(1)}{=} \mathbb{E}_{v\sim p_v}\left[v^T\nabla^2_{x^l_i}\log p(x^l_i)v\right] \overset{(2)}{=} \mathbb{E}_{v\sim p_v}\left[\nabla_{x^l_i}\left(v^T\nabla_{x^l_i}\log p(x^l_i)\right)v\right]$, where p_v is chosen such that $\mathbb{E}[v]=0$ and $\mathbb{E}[vv^T] = I$ (*e.g.*, p_v is the Radamacher distr). Importantly, SVGD is less sensitive to trace approximation errors compared to other MCMC methods (*e.g.*, LD) as shown in Fig. 5. Notably, the trace term in SVGD is scaled by the number of particles M:

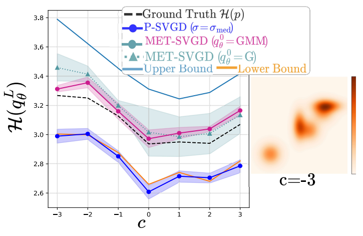
$$\log q^L_\theta(x^L) = \log q^0_{\theta_1}(x^0) + \epsilon^l_{\theta_3}\sum_{l=0}^{L-1}\sum_{x^l_j\neq x^l} Tr\left(\frac{\partial\bar{\phi}_\theta(x^l, x^l_j)}{\partial x^l}\right) + \frac{\epsilon^l_{\theta_3}}{MV}\sum_{v=0}^{V-1}\nabla_{x^l}\left(v^T\nabla_{x^l}\log p(x^l)\right)v,$$

unlike in LD: $\log q^L_\theta(x^L) = \log q^0_{\theta_1}(x^0) + (\epsilon^l_{\theta_3}/V)\sum_{l=0}^{L-1}\sum_{v=0}^{V-1}\nabla_{x^l}(v^T\nabla_{x^l}\log p(x^l))v$, (proof in App. 8.8). Hence, by incorporating this correction, **MET-SVGD** improves scalability and ensures consistency between the sampling dynamics and the associated density derivation.

Complexity. The estimator requires only one additional first-order derivative and two vector dot products per sample. In practice, we find that a single sample v is typically sufficient.

3.4 Divergence Control via Metropolis Hastings

In many applications, $\bar{p}$ is modeled as a deepnet and learnt end-to-end, which often results in non-smooth regions with abrupt gradients, causing samples divergence. To prevent this, **P-SVGD** introduces a heuristic that removes particles deviating beyond a fixed number of standard deviations from the mean of the initial Gaussian distribution $q^0_{\theta_1}$ (Fig. 3B). Intuitively, the initial distribution



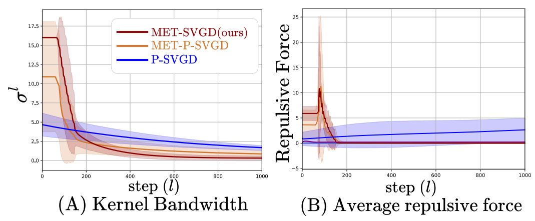


Figure 7: **MET-SVGD** outperforms **P-SVGD** on GMM entropy estimation (details in App. 10.2).

Figure 8: Scalability Results. (A) kernel bandwidth and (B) repulsive force across SVGD steps for the target in Fig. 2D.

approximates the support of the target, and particles that stray too far are likely to be out-of-distribution. Yet, this heuristic has many limitations: it (1) exacerbates mode collapse by discouraging exploration of distant modes (Fig. 2C), (2) is inefficient as replacing rejected particles requires restarting the sampling chain, and (3) prevents divergence but does not improve convergence to non-smooth targets (Fig. 2B). To overcome this, we propose a more principled divergence control mechanism based on MH Robert et al. [2004]. After every update, the proposed position $\tilde{x}^l = x^{l-1} + \epsilon_{\theta_3}\phi_\theta(x^{l-1})$ is accepted with probability α^l_θ, *i.e.*, $x^l = \tilde{x}^l$ or the old position is retained, *i.e.*, $x^l = x^{l-1}$ with probability $1-\alpha^l_\theta$. We compute α^l_θ efficiently by leveraging $Tr(\nabla_{x^l}\phi_\theta(x^l))$.

Proposition 3.5. *Given a target $p = \bar{p}/Z$, the log-likelihood of the MH acceptance probability for an SVGD update of a particle x^{l-1} is:* $\log \alpha^l_\theta = \min[0, \log \bar{p}(\tilde{x}^l) - \log \bar{p}(x^{l-1}) + \epsilon_{\theta_3}\text{Tr}(\nabla_{x^l}\phi_\theta(x^l))]$.

The proof is provided in App. 9. **MET-SVGD** is an MH with an efficient SVGD-based proposal distribution. It therefore inherits asymptotic **convergence guarantees** from MH, *i.e.*, for $L \to \infty$, $q^{\text{MH},L}_\theta$ converges to the target p [Mengersen and Tweedie, 1996]. Unlike SVGD, which requires $L, M \to \infty$ for convergence [Sun et al., 2023], **MET-SVGD** guarantees asymptotic convergence for any number of particles M. The **MH-augmented density** over particles after incorporating MH correction evolves as follows: $q^{\text{MH},l}_\theta(x^l) = \alpha^l_\theta q^{\text{MH},l-1}_\theta(x^{l-1})|\det \nabla_{x^l}\phi_\theta(x^l)|^{-1} + (1-\alpha^l_\theta)q^{\text{MH},l-1}_\theta(x^{l-1})$, with $q^{\text{MH},0}_{\theta_1} = q^0_{\theta_1}$. If all steps are accepted, $q^{\text{MH},L}_\theta$ reduces to q^L_θ (Eq. 2). Thus, $q^{\text{MH},L}_\theta$ is a mixture of SVGD-based distributions over different paths, significantly enriching the expressiveness of the variational family. Moreover, in setups where $\bar{p}$ is learned end-to-end, α^l_θ naturally down-weight poor updates in non-smooth regions. This introduces a self-regularizing effect that not only prevents divergence but also improves learning stability by leveraging information from rejected samples.

Complexity. The MH step introduces negligible computational overhead. The rejection probability depends only on the unnormalized target and $Tr(\nabla_{x^l}\phi(\tilde{x}^l))$, which is already computed for the entropy estimate. Also, there is no added cost for computing the MH-augmented density.

4 Experiment

We report results on (1) entropy estimation for distributions with ground-truth (GT) entropies or bounds, (2) image generation with EBMs and (3) MaxEntr RL.

4.1 Entropy Estimation on Gaussian and GMM Targets

MET-SVGD consistently outperforms **P-SVGD** in entropy estimation across Gaussian (Fig.2A, Fig.2D, Fig.17) and GMM (Fig.7, Fig.23) setups. Notably, Fig.2D and Fig.23 show that, while **P-SVGD** and projection-based baselines such as S-SVGD [Gong et al., 2021] and GSVGD [Liu et al., 2022b] struggle to scale beyond 20 dimensions, **MET-SVGD** achieves high accuracy in up to 100 dimensions. Notice that **MET-SVGD** mitigates the vanishing repulsive force (Fig.8B) previously identified as the root cause of SVGD's poor scalability Ba et al. [2022]. These gains are enabled by learning $\sigma^l_{\theta_2}$ (Fig. 8A), as indicated by the trend difference compared to σ_{med} (Fig. 8A), and incorporating the correction term (Fig. 2D).

4.2 Learning Energy-Based Models

Training EBMs $p_\phi(x) = \bar{p}_\phi(x)/Z$ via maximum likelihood is intractable due to the partition function Z. When the sampler has a tractable distribution q_θ, a tight lower bound can be computed: $\mathcal{L}_{\text{ELBO}}(\phi,\theta) = \mathbb{E}_{x\sim q_\theta}[\log \bar{p}_\phi(x)] - \mathbb{E}_{x\sim p_d}[\log \bar{p}_\phi(x)] + \mathcal{H}(q_\theta)$, as detailed in App. 11. The entropy is often omitted due to its computational complexity, yielding the commonly used contrastive divergence loss $\mathcal{L}_{\text{CD}}(\phi)$. We optimize $\mathcal{L}_{\text{ELBO}}(\phi,\theta)$ using both **P-SVGD** and **MET-SVGD**, and train with $\mathcal{L}_{\text{CD}}(\phi)$ using LD. Experiments are conducted on the **Moon dataset** [Rezende and Mohamed, 2015b] (Fig. 24) and CIFAR10 (Fig. 9). For CIFAR10, we report the Frechet Inception Distance

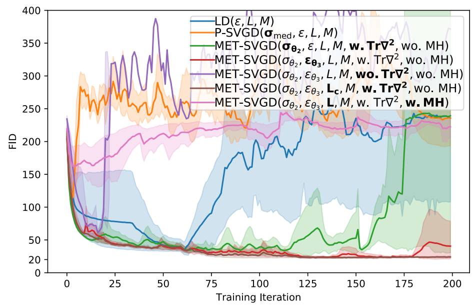
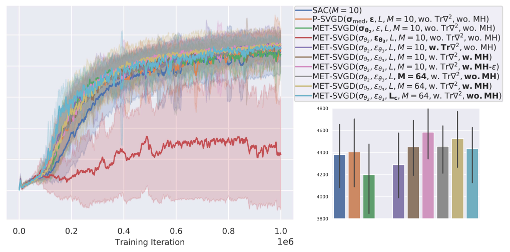


Figure 9: FID on CIFAR10. Modification between consecutive configs is **bolded**. **MET-SVGD** improves training stability.



(A) Return IQM (B) Final Return IQM

Figure 10: Return IQM for Walker2d-v2. MH variants yield the best returns.

(FID) over 5 seeds. In Fig. 9, we show that not including the trace of Hessian in **MET-SVGD** (purple) leads to divergence. Using an adaptive number of steps L_c stabilizes the training (green). Replacing σ_{med} with the learnable one (red) improves stability and yields significantly better FID scores relative to **P-SVGD** (orange). Additionally, learning the step-size (brown) enables faster convergence to the target ($\epsilon^l_{\theta_3} \gg \epsilon$ in Fig. 28) and results in smoother landscapes (Fig. 27). Yet, experiments with MH diverge. In App. 11, we show that MH-augmented updates lead to a high rejection rate due to landscape complexity. This results in poor sampling and eventually divergence. To mitigate this, in future work, we plan to explore controlling the Lipschitz constant of the target. Also, learning only the kernel bandwidth does improve over **P-SVGD**. We attribute this to vanishing gradients in high-dimensions, *i.e.*, the kernel collapses to zero. To address this, we plan to explore dimension-wise decomposable kernels. Qualitative results and implementation details are in App. 11.

4.3 Max-Entropy Reinforcement Learning

Unlike classical RL, which learns a deterministic policy [Sutton et al., 1999], MaxEnt RL [Ziebart, 2010] learns a stochastic policy π_θ by maximizing the sum of expected rewards and entropies: $\pi^*_\theta = \arg\max_{\pi_\theta} \sum_t \mathbb{E}_{(s_t, a_t)}\big[r(s_t, a_t) + \alpha\mathcal{H}(\pi_\theta(\cdot|s_t))\big]$. Following S^{2}AC Messaoud et al. [2024] (**P-SVGD**), we model the policy as an SVGD sampler and estimate the entropy using **MET-SVGD** on Walker2d-v2 and Humanoid-v2 environments[Brockman et al., 2016]. We compare to SAC [Haarnoja et al., 2018], which models the policy as a Gaussian. We train 5 instances of each algorithm with different random seeds and report the average return on 10 rollouts every 1000 steps in Fig. 10 for Walker2d-v2 and Fig. 29A for Humanoid-v2. In both environments, including the missing Hessian trace in **MET-SVGD** improves performance by smoothing the landscape for better sampling (Fig. 36). While MH with few particles hinders early exploration, using an epsilon-greedy MH step and increasing particle count mitigates this and improves performance. Learning $\sigma^l_{\theta_3}$ alone yields results comparable to **P-SVGD** (Fig. 35). Using an adaptive number of steps is more sample efficient as it helps adapt the sampling to more complex distributions. MH variants yield the best results. In particular, MH-ϵ corresponds to the version with an epsilon-greedy strategy for applying the MH step. Intuitively, it allows more exploration initially. In the future, we will explore using importance sampling to directly optimize the KL divergence for further improving the exploration (Eq. 3.3). Additional results are in App. 12.

5 Conclusion

We introduced **MET-SVGD**, a novel VI approach for entropy estimation in setups where only the unnormalized density is given. To the VI community, **MET-SVGD** is a new method that bridges the gap between VI, particle-based inference and MCMC. To the SVGD community, it sets a new SOTA for scaling SVGD sampling to high-dimensional and non-smooth densities. For the broader Bayesian community, it introduces a framework for learning sampler parameters end-to-end, addressing the long standing challenge of how to select sampler parameters. To the generative models community, **MET-SVGD** is a new residual flow model with full rank Jacoian and adaptive number of layers.

References

John Wiley Sons, Ltd, 1992.

I. Ahmad and P.-E. Lin. A nonparametric estimation of the entropy for absolutely continuous distributions. *IEEE Trans. Inf. Theory*, 1976.

et al. Ahmed, Zafarali. Understanding the impact of entropy on policy optimization. *ICML*, 2019.

et al. Alemi, A. A. Deep variational information bottleneck. *ICLR*, 2016.

Gil Ariel and Yoram Louzoun. Estimating differential entropy using recursive copula splitting. *Entropy*, 2020.

Jimmy Ba, Murat A. Erdogdu, Marzyeh Ghassemi, Shengyang Sun, Taiji Suzuki, Denny Wu, and Tianzong Zhang. Understanding the variance collapse of svgd in high dimensions. In *ICML*, 2022.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *ICML*, 2019.

Jan Beirlant, Edward J Dudewicz, László Györfi, Edward C Van der Meulen, et al. Nonparametric entropy estimation: An overview. *Int. J. Math. Stat. Sci.*, 1997.

et al. Belghazi, M. I. Mutual information neural estimation. *ICML*, 2018.

Jose M Bernardo. Reference posterior distributions for bayesian inference. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 1979.

Vladimir Igorevich Bogachev, Aleksandr Viktorovich Kolesnikov, and Kirill Vladimirovich Medvedev. Triangular transformations of measures. *Sb. Math.*, 2005.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

George T Cantwell. Approximate sampling and estimation of partition functions using neural networks. *arXiv preprint arXiv:2209.10423*, 2022.

Markov Chain Monte Carlo. Honest exploration of intractable probability distributions via. *Stat. Sci.*, 2001.

Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. Hamiltonian variational auto-encoder. *NeurIPS*, 2018.

Yogendra P. Chaubey and Pranab K. Sen. On nonparametric estimation of the density of a non-negative function of observations. *Calcutta Stat. Assoc. Bull.*, 2013.

Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *NeurIPS*, 2019.

Wei-Chia Chen, Ammar Tareen, and Justin B Kinney. Density estimation on small data sets. *Phys. Rev. Lett.*, 2018.

Siddhartha Chib. Markov chain monte carlo methods: computation and inference. *Handbook of econometrics*, 2001.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Bo Dai, Hanjun Dai, Arthur Gretton, Le Song, Dale Schuurmans, and Niao He. Kernel exponential family estimation via doubly dual embedding. In *AISTATS*, 2019a.

Bo Dai, Zhen Liu, Hanjun Dai, Niao He, Arthur Gretton, Le Song, and Dale Schuurmans. Exponential family estimation via adversarial dynamics embedding. *NeurIPS*, 32, 2019b.

Jay L Devore, Kenneth N Berk, Matthew A Carlton, et al. *Modern mathematical statistics with applications*. Springer, 2012.

Yu G. Dmitriev and F. P. Tarasenko. On estimation of functionals of the probability density function and its derivatives. *Theory Probab. Its Appl.*, 1973.

M. D. Donsker and S. R. S. Varadhan. Asymptotic evaluation of certain markov process expectations for large time—ii. *Commun. Pure Appl. Math.*, 1975.

Andrew Duncan, Nikolas Nüsken, and Lukasz Szpruch. On the geometry of stein variational gradient descent. *J. Mach. Learn. Res.*, 2023.

Charles W Fox and Stephen J Roberts. A tutorial on variational bayesian inference. *Artif. Intell. Rev.*, 2012.

Tomas Geffner and Justin Domke. Langevin diffusion variational inference. In *AISTATS*, 2023.

Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *NeurIPS*, 2017.

Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. Sliced kernelized stein discrepancy. In *ICLR*, 2021.

László Györfi and Edward C Van der Meulen. Density-free convergence properties of various estimators of entropy. *Comput. Stat. Data Anal.*, 1987.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Eldad Haber, Lars Ruthotto, Elliot Holtham, and Seong-Hwan Jun. Learning across scales - a multiscale method for convolution neural networks. 2017.

Peter Hall and Sally C Morton. On the estimation of entropy. *Ann. Inst. Stat. Math.*, 1993.

Hanns-Ludwig Harney. Bayesian inference. *Advanced Texts in Physics. Springer*, 2003.

Kakade S. M. Singh K. Hazan, E. and A. Van Soest. Provably efficient maximum entropy exploration. *NeurIPS*, 2019.

Jose Miguel Hernandez-Lobato, Neil Houlsby, and Zoubin Ghahramani. Probabilistic matrix factorization with non-random missing data. *ICML*, 2014.

Hideitsu Hino and Noboru Murata. A conditional entropy minimization criterion for dimensionality reduction and multiple kernel learning. *Neural Comput.*, 2010.

Matthew D Hoffman. Learning deep latent gaussian models with markov chain monte carlo. In *ICML*, 2017.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Commun. Stat. Simul. Comput.*, 1989.

Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *ICLR*, 2018.

Harry Joe. Estimation of entropy and other functionals of a multivariate density. *Ann. Inst. Stat. Math.*, 1989.

Galin L Jones and James P Hobert. Sufficient burn-in for gibbs samplers for a hierarchical random effects model. *Ann. Stat.*, 2004.

et al. Kandasamy, Kirthevasan. Nonparametric von mises estimators for entropies, divergences and mutual informations. *NeurIPS*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2013a.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2013b.

Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

Anna Korba, Adil Salim, Michael Arbel, Giulia Luise, and Arthur Gretton. A non-asymptotic analysis for stein variational gradient descent. *NeurIPS*, 2020.

A. Kraskov. Estimating mutual information. *Phys. Rev. E*, 2004.

Ullrich Köthe. A review of change of variable formulas for generative modeling. 2023.

Erik G Learned-Miller and John W Fisher III. Ica using spacings estimates of entropy. *J. Mach. Learn. Res.*, 2003.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Pred. Struct. Data*, 2006.

Haozhe Liu, Bing Li, Haoqian Wu, Hanbang Liang, Yawen Huang, Yuexiang Li, Bernard Ghanem, and Yefeng Zheng. Combating mode collapse in gans via manifold entropy estimation. *AAAI*, 2022a.

Qiang Liu. A short introduction to kernelized stein discrepancy. 2016. URL `https://api.semanticscholar.org/CorpusID:16209224`.

Qiang Liu. Stein variational gradient descent as gradient flow. *NeurIPS*, 2017.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *NeurIPS*, 2016.

Qiang Liu, Jason D. Lee, and Michael I. Jordan. A kernelized stein discrepancy for goodness-of-fit tests and model evaluation. *ICML*, 2016.

Tianle Liu, Promit Ghosal, Krishnakumar Balasubramanian, and Natesh Pillai. Towards understanding the dynamics of gaussian-stein variational gradient descent. *NeurIPS*, 2024.

Xing Liu, Harrison Zhu, Jean-François Ton, George Wynne, and Andrew Duncan. Grassmann stein variational gradient descent. *AISTATS*, 2022b.

Shie Mannor, Dori Peleg, and Reuven Rubinstein. The cross entropy method for classification. In *ICML*, 2005.

Kerrie L Mengersen and Richard L Tweedie. Rates of convergence of the hastings and metropolis algorithms. *Ann. Stat.*, 1996.

Safa Messaoud, Billel Mokeddem, Zhenghai Xue, Linsey Pang, Bo An, Haipeng Chen, and Sanjay Chawla. SΘ2 ac: Energy-based reinforcement learning with stein soft actor critic. *ICLR*, 2024.

Kevin R Moon, Kumar Sricharan, Kristjan Greenewald, and Alfred O Hero III. Ensemble estimation of information divergence. *Entropy*, 2018.

Liam Paninski. Estimation of entropy and mutual information. *Neural Comput.*, 2003.

E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Stat.*, 1962.

Colombo P. Boudiaf Pichler, G. Knife: Kernelized-neural differential entropy estimation. *ICLR*, 2022.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *ICML*, 2015a.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015b.

Christian P Robert, George Casella, Christian P Robert, and George Casella. The metropolis—hastings algorithm. *Monte Carlo statistical methods*, 2004.

CP Robert. Monte carlo statistical methods, 1999.

Gareth O Roberts and Jeffrey S Rosenthal. General state space markov chains and mcmc algorithms. *Probab. Surv.*, 2004.

M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Ann. Math. Statist.*, 1956.

Jeffrey S Rosenthal. Minorization conditions and convergence rates for markov chain monte carlo. *J. Am. Stat. Assoc.*, 1995.

Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics)*. Springer-Verlag, 2004.

Adil Salim, Lukang Sun, and Peter Richtarik. A convergence theory for svgd in the population limit under talagrand's inequality t1. In *ICML*. PMLR, 2022.

Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *ICML*. PMLR, 2015.

N. N. Schraudolph. Gradient-based manipulation of nonparametric entropy estimates. *IEEE Trans. Neural Netw. Learn. Syst.*, 2004.

Claude Elwood Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 1948.

Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 2001.

Jiaxin Shi and Lester Mackey. A finite-particle convergence rate for stein variational gradient descent. *NeurIPS*, 2024.

P Shyam. Model-based active exploration. *ICLR*, 2019.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Trans. Signal Process*, 2017.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *UAI*, 2020.

Kumar Sricharan, Dennis Wei, and Alfred O Hero. Ensemble estimators for multivariate entropy estimation. *IEEE Trans. Inf. Theory*, 2013.

Lukang Sun, Avetik Karagulyan, and Peter Richtarik. Convergence of stein variational gradient descent under a weaker smoothness condition. In *AISTATS*. PMLR, 2023.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *NeurIPS*, 1999.

F.P. Tarasenko. On the evaluation of an unknown probability density function, the direct estimation of the entropy from independent observations of a continuous random variable, and the distribution-free entropy test of goodness-of-fit. *Proc. IEEE*, 1968.

Achille Thin, Nikita Kotelevskii, Jean-Stanislas Denain, Leo Grinsztajn, Alain Durmus, Maxim Panov, and Eric Moulines. Metflow: a new efficient method for bridging the gap between markov chain monte carlo and variational inference. *arXiv preprint arXiv:2002.12253*, 2020.

Luke Tierney. Markov chains for exploring posterior distributions. *Ann. Stat.*, 1994.

O. Vasicek. A test for normality based on sample entropy. *J R Stat Soc Series B Stat Methodol*, 1976.

Cédric Villani et al. *Optimal transport: old and new*. Springer, 2009.

Nicol Schraudolph Viola, Paul and Terrence J. Sejnowski. Empirical entropy manipulation for real-world problems. *NeurIPS*, 1995.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

Christopher S. Withers and Saralees Nadarajah. log det a = tr log a. *Int. J. Math. Educ. Sci. Technol.*, 2010.

Henry Wolkowicz and George PH Styan. Bounds for eigenvalues using traces. *Linear Algebra Appl.*, 1980.

Jiaxi Wu, Jiaxin Chen, and Di Huang. Entropy-based active learning for object detection with progressive diversity constraint. In *CVPR*, 2022.

Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

A. Zellner. An introduction to bayesian inference in econometrics. 1971.

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

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately describe **MET-SVGD** contributions. Empirical results (GMM entropy estimation, EBMs, MaxEnt RL tasks) align closely with claims made.

   Guidelines:

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations at the end of Sec. 4.2 and Sec. 4.3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We add assumptions and proofs in supplementary.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide experimental details in Appendix and share a link to the code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: provide code for reproducing all the figures in the paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We share important details in the paper and the rest in supplementary.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, across all our experiments, we run for several seeds (exact number depends on the experiment and is mentioned in the paper). We report mean and variance. For the RL experiments, we report the IQM (inter-quantile mean).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the required details in supplementary.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We took the required measures to ensure that our submission is anonymous.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our paper proposes a new sampling technique that can be leveraged in standard machine learning applications.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: we propose a new algorithm for sampling.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: We are the original owners.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We share the code

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:[NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA] .

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

Supplementary Material

MET-SVGD is a novel variational inference approach for entropy estimation that overcomes key limitations of **P-SVGD** Messaoud et al. [2024], particularly poor convergence and scalability in high-dimensional spaces (Fig. 2). To achieve this, it introduces: (1) Sufficient condition for global invertibility. (2) Optimized parameter search for improved stability (Sec. 3.2). (3) Metropolis-Hastings augmented SVGD updates to ensure asymptotic convergence (Sec. 3.4). (4) A correction term to the density estimation in **P-SVGD** (Sec. 3.3). **MET-SVGD** maintains computational efficiency, requiring no significant additional memory or runtime overhead. Its full workflow is illustrated in Algorithm 1. Beyond entropy estimation, **MET-SVGD** can be valuable to different research communities:

- **MET-SVGD** bridges the gap between Metropolis-Hastings algorithms (MH) Robert et al. [2004], particle-based sampling techniques (SVGD) Liu and Wang [2016], and parametrized variational inference (P-VI) Fox and Roberts [2012], leveraging the strengths of each (Tab. 1): (1) scalability from P-VI, (2) expressivity, convergence detection, and particle efficiency from SVGD, as well as (3) convergence guarantees from MH. See Fig. 9
- **MET-SVGD** is a new approach for unprecedentedly scaling SVGD to high-dimensional spaces while being computationally more efficient than all proposed approaches in the literature Gong et al. [2021], Liu et al. [2022b]
- **MET-SVGD** is a new approach for end-to-end learning of sampler parameters. It enables training samplers via KL-divergence minimization, achieving compelling results for both LD (Fig. 12B) and SVGD (Fig. 12A).
- **MET-SVGD** is a new normalizing flow model with (1) an adaptive number of updates controlled by a convergence check and (2) a full-rank Jacobian for improved flexibility and expressivity (Fig. 13). We plan to extend **MET-SVGD** to image generation using flow-matching in future work.

The detailed algorithm is in Alg.1. We build a library for **MET-SVGD**. Our code is available at: `https://anonymous.4open.science/r/Variational-Inference-with-SVGD--3F81/README.md`.

Criterion	**P-VI**	**MCMC**	**SVGD**	**P-SVGD**	**MET-SVGD**
Expressivity	✗	✓	✓	✓	✓✓
Convergence Detection	✓	✗	✓	✓	✓
Convergence Guarantees	✗	✓	✗	✗	✓
Sampling Efficiency	✓	✗	✓	✓	✓
Tractable Entropy	✓	✗	✗	✓	✓
Parameter Efficiency	✓	-	-	✓✓	✓✓

Table 1: **MET-SVGD** inherits advantages of different approximate inference methods: VI, SVGD, and MCMC.

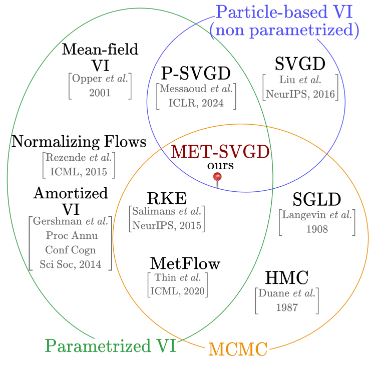

Particle-based VI (non parametrized)
Mean-field VI
[Opper et al. 2001]
P-SVGD
[Messaoud et al. ICLR, 2024]
SVGD
[Liu et al. NeurIPS, 2016]
Normalizing Flows
[Rezende et al. ICML, 2015]
MET-SVGD
ours
Amortized VI
[Gershman et al. Proc Annu Conf Cogn Sci Soc, 2014]
RKE
[Salimans et al. NeurIPS, 2015]
SGLD
[Langevin et al. 1908]
MetFlow
[Thin et al. ICML, 2020]
HMC
[Duane et al. 1987]
Parametrized VI
MCMC

Figure 11: Bridges the gap ...

Table 2: **P-SVGD** vs **MET-SVGD**

Category	**P-SVGD**	**MET-SVGD**
Invertibility Condition	Local (Implicit Function Theoremm); imprecise: $\epsilon \ll \sigma$ (Proposition 3.2, P-SVGD paper)	Global (Banach Theorem); precise: $\sqrt{\mathrm{Tr}(\nabla\phi^l \nabla\phi^{l,\top})}$ (Corollary 3.3)
Entropy Trace Approximation	Imprecise: $\epsilon\|\|\nabla\phi^l\|\|_\infty \ll 1$ (Theorem 3.1, P-SVGD paper)	Automatically implied by invertibility condition (Corollary 3.3)
Divergence Control	Heuristic: particles truncation beyond 3 std from $q^{(0)}_{\theta_1}$ mean (Eq. 9, P-SVGD paper)	Metropolis-Hastings correction (Section 3.4)
Tr(Hessian) in Entropy	Omitted; invalid for finite particles (Theorem 3.3, P-SVGD paper)	Restored via Hutchinson estimator (Section 3.3)
Kernel Bandwidth σ	Median heuristic: $O(M^2)$	Learned via lightweight GNN (Section 3.2)
Step Size ϵ	Fixed	Learned via lightweight GNN (Section 3.2)
Number of Steps L	Fixed	Adaptive via Stein Identity (Section 3.2)
Computation	Grid search for ϵ, median heuristic for σ^2 ($O(M^2)$)	Efficient reuse of $\mathrm{Tr}(\nabla\phi^l)$ for the invertibility bound (Corollary 3.3), MH correction (Proposition 3.4), and convergence check ; GNN inference adds minor overhead (Section 3.2)
Memory	-	Two small GNNs for σ, ϵ (Section 3.2)
Convergence Guarantee	$L, M \to \infty$	$L \to \infty$
Empirical Performance	Sensitive to hyperparameters (Fig. 2A); mode collapse (Fig. 2C); poor scalability to non-smooth and high-dimensional targets (Fig. 2B and Fig. 2D)	SoTA entropy on G/GMM (Fig. 4 and Fig. 7); better FID, stability in EBMs for image generation (Fig. 9); improved MaxEnt RL returns (Fig. **??**)

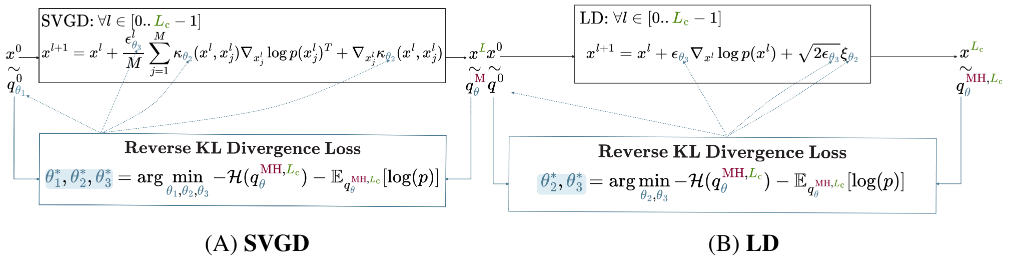

SVGD: $\forall l \in [0..L_c - 1]$
$x^{l+1} = x^l + \frac{\epsilon^l_{\theta_3}}{M}\sum_{j=1}^{M}\kappa_{\theta_2}(x^l, x^l_j)\nabla_{x^l_j}\log p(x^l_j)^T + \nabla_{x^l_j}\kappa_{\theta_2}(x^l, x^l_j)$
$x^0 \sim q^0_{\theta_1}$
$x^L \sim q^M_\theta$
Reverse KL Divergence Loss
$\theta_1^*, \theta_2^*, \theta_3^* = \arg\min_{\theta_1,\theta_2,\theta_3} -\mathcal{H}(q_\theta^{\mathrm{MH},L_c}) - \mathbb{E}_{q_\theta^{\mathrm{MH},L_c}}[\log(p)]$
LD: $\forall l \in [0..L_c - 1]$
$x^{l+1} = x^l + \epsilon_{\theta_3}\nabla_{x^l}\log p(x^l) + \sqrt{2\epsilon_{\theta_3}}\xi_{\theta_2}$
$x^0 \sim q^0$
$x^{L_c} \sim q_\theta^{\mathrm{MH},L_c}$
Reverse KL Divergence Loss
$\theta_2^*, \theta_3^* = \arg\min_{\theta_2,\theta_3} -\mathcal{H}(q_\theta^{\mathrm{MH},L_c}) - \mathbb{E}_{q_\theta^{\mathrm{MH},L_c}}[\log(p)]$
(A) **SVGD**
(B) **LD**

Figure 12: **MET-SVGD provides a principled approach to learn sampler parameters** via first computing the particles induced density, then Learning the parameters through KLD minimization.

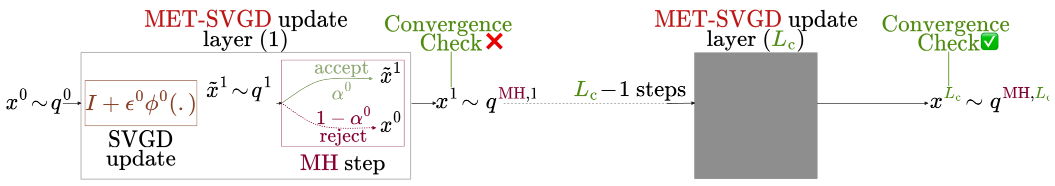


Figure 13: **MET-SVGD** is a normalizing flow model with a full rank Jacobian and an adaptive number of layers.

The rest of the appendix is organized as follows:

- Appendix 6: **Preliminaries**, including the Change of Variable formula for probability densities, Jacobi's formula corollary, the Stein Identity, the Banach Theorem and the implicit function theorem.

- Appendix 7: **Additional related work and Background** on entropy estimation, sampling-based variational inference, Normalizing Flows, Metropolis-Hastings, SVGD and the Stein Identity.
- Appendix 8: **Derivation of closed-form density expressions for LD and SVGD samplers** using RBF, Bilinear, and DKEF kernels. This section also includes derivation of the sufficient condition on the step-size.
- Appendix 9 **Derivation of the Metropolis-Hastings augmented entropy**
- Appendix 10: **Additional results on entropy estimation**
- Appendix 11: **Additional results on learning EBMs for image generation**
- Appendix 12: **Additional results on MaxEntr RL**

Algorithm 1: MET-SVGD (Training)

```
input : Unormalized density $\bar{p}$. SVGD parameters: (i) initial distr. $q^0_{\theta_1}$, (ii) number of particles $M$,
    (iii) maximum number of steps $L$, (vi) RBF kernel variance deepnet $\sigma_{\theta_2}$ and (v) learning rate
    deepnet $\epsilon_{\theta_3}$.
output : $\theta^* = \{\theta^*_1, \theta^*_2, \theta^*_3\}$.
1:  for Each training iteration do
2:      $l = 0$ % initialize the number of SVGD steps
3:      $\{x^0_i\}_{i=0}^{M-1} \sim q^0_{\theta_1}$ % sample initial particles from $q^0_{\theta_1}$
4:      $q^0_{\text{MH}} = q^0_{\theta_1}$ % Initialize $q^0_{\text{MH}}$
5:      % Run SVGD chain to convergence of si(Eq. 3.3)
6:      while $\Big(l \leq L\Big)$ and $\Big(\ \Delta\text{SI}(q^{\text{MH},l}_{\theta}, p) \leq 0\Big)$ do
7:          $\epsilon^l_{\theta_3} = \text{GNN}(\{x^l_i\}_{i=0}^{M-1}; \theta_3)$ % Compute learning rate
8:          $\epsilon^l_{\theta_3} = \min(\epsilon^l_{\theta_3}, \epsilon^l_{\text{UB}})$ % Learning rate truncation (Corr.3.3)
9:          $\sigma^l_{\theta_2} = \text{GNN}(\{x^l_i\}_{i=0}^{M-1}; \theta_2)$ % Compute kernel variance
10:         $\tilde{x}^{l+1}_i \leftarrow x^l_i + \epsilon\phi(x^l_i), \forall i \in [0, M-1]$ % SVGD update (Eq .1)
11:         % Metropolis Hastings Step (Sec. 3.4)
12:         $\alpha^l_{i,\theta_{2,3}} = (\alpha^l_{i,\theta_{2,3}})^{a_l}, \quad a_l \in \{0,1\}, \forall i \in [0, M-1]$ % MH acceptance probability
13:         $u^l_i \sim \mathcal{N}(0, I)$ % Generate uniform random number
14:         $x^{l+1}_i = \tilde{x}^{l+1}_i$ If $u^l_i \geq \alpha_i$, Else $x^{l+1}_i = \tilde{x}^l_i, \forall i \in [0, M-1]$ % Update
15:         % Update $q^{\text{MH},l}_{\theta}$ (Eq. 3.4)
16:         $\log q^{\text{MH},l}_{\theta}(x^l) = \log q^{\text{MH},l-1}_{\theta}(x^{l-1}) + \log\Bigg[\exp\Big(\log(\alpha^l_{\theta_{2,3}}) - \epsilon_{\theta_3}\text{Tr}(\nabla_{x^l}\phi(x^l))\Big) + \exp\Big(\log(1-\alpha^l_{\theta_{2,3}})\Big)\Bigg]$
17:         $l \leftarrow l + 1$ % Update number of steps
18:     end while
19:     $L_c \leftarrow l$
20:     $\mathcal{H}(q^{L_c}_{\theta}) = \frac{-1}{M}\sum_{i=0}^{M-1} \log q^{\text{MH},L_c}(x^{L_c}_i)$ % Compute entropy
21:     % Update $\theta$
22:     $\{\theta^*_1, \theta^*_2, \theta^*_3\} = \arg\max_{\theta_1,\theta_2,\theta_3} \mathbb{E}_{x^{L_c} \sim q^{\text{MH},L_c}_{\theta}}[\log p(x)] + \mathcal{H}(q^{\text{MH},L_c}_{\theta})$
23: end for
24: Return $\theta^* = \{\theta^*_1, \theta^*_2, \theta^*_3\}$
```

6 Preliminaries

In the following, we review preliminaries about Langevin Dynamics, the Change of Variable formula for pdfs, the corollary of the Jacobi formula, the Banach theorem, the Mean Value theorem, a Sufficient Condition for residual flows invertibility and the Stein Identity.

6.1 Langevin Dynamics

SGLD [Welling and Teh, 2011] is a popular Markov chain Monte Carlo (MCMC) method for sampling from a distribution. It first initializes a sample x^0 from a random initial distribution. Then at every step, it adds the gradient of the current proposal distribution $p(x)$ to the previous sample x^l, together with a Brownian motion $\xi \sim \mathcal{N}(0, I)$. We denote with ϵ the step size. The iterative update for SGLD is:

$$x^{l+1} = x^l + \epsilon \nabla_{x^l} \log p(x^l) + \sqrt{2\epsilon}\xi. \tag{6}$$

6.2 Change of Variable Formula (CVF)

We first introduce the concept of an Invertible Function.

According to [Köthe, 2023], the following holds: if $F : Z \to X$ is an invertible function then:

$$p_X(x) = p_Z(z)\Big|\det \frac{\partial F^{-1}(x)}{\partial x}\Big| = p_Z(z)\Big|\det \frac{\partial F(z)}{\partial z}\Big|^{-1}$$

6.3 Implicit Function Theorem

Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be continuously differentiable on some open set containing a, and suppose $\det(\nabla_x f(x)) \neq 0$. Then, there is some open set V containing x and an open W containing $f(x)$ such that $f : V \to W$ has a continuous inverse $f^{-1} : W \to V$ which is differentiable $\forall y \in W$.

6.4 Corollary of Jacobi's Formula

Given an invertible matrix A, the following equality holds:

$$\log(\det A) = \mathit{Tr}(\log A) = \mathit{Tr}\Big(\sum_{k=1}^{\infty} (-1)^{k+1} \frac{(A - I)^k}{k}\Big). \tag{7}$$

The second equation is obtained by taking the power series of $\log A$. Hence, under the assumption $\|A - I\|_\infty \ll 1$, we obtain: $\log(\det A) \approx \text{tr}(A - I)$, where $\|\cdot\|_\infty$ is the infinity norm.

6.5 Banach Theorem

We begin by introducing the concepts of a cauchy sequence and a contractive mapping. Next, we discuss the Banach Fixed Point theorem.

Theorem 6.1 (Cauchy Sequence). *If a sequence $\{x_n\}_{n\in\mathbb{N}}$ satisfy **either** of the following conditions:*

1. $|x_{n+1} - x_n| \leq \alpha^n, \quad \forall n \in \mathbb{N}$
2. $|x_{n+2} - x_{n+1}| \leq \alpha |x_{n+1} - x_n|, \quad \forall n \in \mathbb{N},$

where $0 < \alpha < 1$, then $\{x_n\}$ is a Cauchy sequence.

Theorem 6.2 (Contractive Mapping). *Let $(\mathcal{X}, d)$ be a metric space with d a distance function and let $\phi : \mathcal{X} \to \mathcal{X}$ be a mapping on $\mathcal{X}$. ϕ is called a contraction if and only if:*

$$\exists K \in [0, 1[\quad \textit{s.t.} \quad d(\phi(x), \phi(\tilde{x})) \leq K d(x, \tilde{x}), \quad \forall x, \tilde{x} \in \mathcal{X} \tag{8}$$

Theorem 6.3 (Banach Fixed Point). *Let (X, d) be a complete metric space (*i.e.*, all **Cauchy Sequences** are convergent) with d a distance function. If ϕ is a contraction, then it has a **unique fixed point** $x^* \in X$,* i.e.*, $\phi(x^*) = x^*$ and*

$$\forall x_0 \in \boldsymbol{X}, \quad \lim_{n\to\infty} \phi^n(x_0) = x^*, \quad \textit{with} \quad \phi^n(x_0) = \underbrace{\phi \circ \phi \circ \cdots \circ}_{n\ times} \phi(x_0) = x_n.$$

Proof. The proof is structured in two main parts: we first establish the *existence* of a fixed point by showing that $(x_n)_{n\in\mathbb{N}}$ is a Cauchy sequence. Then prove *uniqueness* of the fixed point using a proof by contradiction.

Step 1: Existence of a fixed point. $(x_n)_{n\in\mathbb{N}}$ is a Cauchy sequence, we distinguish two cases: consecutive samples and non-consecutive samples.

- *consecutive samples:*

$$d(x_{n+1}, x_n) = d(\phi(x_n), \phi(x_{n-1})) \leq K\, d(x_n, x_{n-1}) \leq K^2\, d(x_{n-1}, x_{n-2}) \leq \cdots \leq K^n\, d(x_1, x_0)$$

- *non-consecutive samples x_n and x_m with $n < m$*

$$\begin{aligned}
d(x_n, x_m) &\leq d(x_n, x_{n-1}) + d(x_{n-1}, x_{n-2}) + \cdots + d(x_{m+1}, x_m) \\
&\leq (K^{n-1} + K^{n-2} + \cdots + K^m)\, d(x_1, x_0) \\
&\leq K^m \underbrace{\sum_{k=0}^{n-1-m}}_{\leq \sum_{k=0}^{\infty} K^k} K^k\, d(x_1, x_0) \\
&\leq K^m \left(\sum_{k=0}^{\infty} K^k \right) d(x_1, x_0) = \frac{K^m}{1-q}\, d(x_1, x_0)
\end{aligned}$$

It follows that $\{x_n\}_{n\in\mathbb{N}}$ is a Cauchy sequence since $d(x_n, x_m) \to 0$ as $n, m \to \infty$. Because the metric space is complete, this implies convergence to a limit $x^* \in \mathcal{X}$: *i.e.*, , $x^* = \lim_{n\to\infty} x_n$. Additionally, since ϕ is continuous,

$$\phi(x^*) = \phi\left(\lim_{n\to\infty} x_n\right) = \lim_{n\to\infty} \phi(x_n) = \lim_{n\to\infty} x_{n+1} = x^*.$$

Hence, x^* is a fixed point of ϕ.

Step 2: Uniqueness of the fixed point. Assume that there exist two distinct fixed points x^* and $\hat{x}$ such that $\phi(x^*) = x^*$ and $\phi(\hat{x}) = \hat{x}$. Then, If $x^* \neq \hat{x} \quad \Rightarrow \quad d(x^*, \hat{x}) = d(\phi(x^*), \phi(\hat{x})) \leq K\, d(x^*, \hat{x})$ Which implies $\Rightarrow \frac{d(x^*, \hat{x})}{d(x^*, \hat{x})} \leq K \Rightarrow 1 \leq K$. which contradicts the assumption that $K < 1$. Hence, the fixed point exists and is unique. We can compute it using the following algorithm:

Algorithm 2: Inverse of $g(x)$ via fixed point iteration

```
input  $y^0 = g(x)$, number of fixed-point iterations $n$
1: for $i = 0 \cdots n-1$ do
2:     $y^{i+1} = y^0 - g(y^i)$
3: end for
4: Return $y^n = g(x)^{-1}$
```

□

6.6 The Mean Value Theorem

Theorem 6.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable on $\mathbb{R}^n$ with a Lipschitz continuous gradient ∇f. Then for given x and $\bar{x}$ in $\mathbb{R}^n$, there is $y = x + t(x - \bar{x})$ with $t \in [0, 1]$, such that*

$$f(x) - f(\bar{x}) = \nabla f(y) \cdot (x - \bar{x}).$$

6.7 Stein Identity ([Liu, 2016])

Let $p(x)$ be a continuously differentiable density supported on $\mathcal{X} \subseteq \mathbb{R}^d$, and let $\phi(x) = [\phi_1(x), \cdots, \phi_d(x)]^T$ be a vector-valued function. **Stein's identity** states that for sufficiently regular ϕ, we have:

$$\mathbb{E}_{x\sim p}[\mathcal{A}_p \phi(x)] = 0,$$

where the Stein operator $\mathcal{A}_p$ is defined as: $\mathcal{A}_p\phi(x) = \phi(x)\nabla_x \log p(x) + \nabla_x \phi(x)$.

Proof. We can verify this identity using integration by parts under mild boundary assumptions: either $p(x)\phi(x) = 0, \quad \forall x \in \partial\mathcal{X}$ when $\mathcal{X}$ is compact, or $\lim_{\|x\|\to\infty} \phi(x)p(x) = 0$ when $\mathcal{X} = \mathbb{R}^d$.

In the following we assume $\mathcal{X} = [a, b]$:

$$\begin{aligned}\mathbb{E}_{x\sim p}[\mathcal{A}_p\phi(x)] &= \int_a^b p(x)\phi(x)\nabla_x \log p(x) + p(x)\nabla_x\phi(x)\, dx \\ &\overset{(i)}{=} \int_a^b \phi(x)\nabla_x p(x) + p(x)\nabla_x \phi(x)\, dx \overset{(ii)}{=} [\phi(x)p(x)]_a^b \overset{(iii)}{=} 0\end{aligned}$$

(i) Uses the identity $\nabla_x \log p(x) = \frac{\nabla_x p(x)}{p(x)}$.

(ii) Applies integration by parts: $\int_a^b f(x)g'(x) + f'(x)g(x)\, dx = [f(x)g(x)]_a^b$.

(iii) Boundary term vanishes under the stated assumptions.

□

7 Additional Related Work

In the following, we review additional work on the differential entropy, sampling-based variational inference, Normalizing Flows, Stein Variational Gradient Descent (SVGD) and Metropolis Hastings (MH) convergence.

7.1 Differential Entropy

Differential entropy, first introduced by Shannon in his foundational work on information theory Shannon [1948], has been widely studied in statistics Box [1992], Zellner [1971], Bernardo [1979]. For a continuous random variable z with density $p(z)$, the entropy is defined as:

$$\mathcal{H}(x) = -\int_{-\infty}^{\infty} p(x) \log \big(p(x)\big)\, dx.$$

Applications of Entropy: Entropy plays a crucial role in machine learning, Bayesian inference (BI), reinforcement learning (RL), and variational inference (VI): (i) In classification & calibration, the entropy measures model confidence Shyam [2019], used in active learning Wu et al. [2022]. (ii) In Bayesian Inference, the Maximum Entropy principle ensures the least informative prior Bernardo [1979]. (iii) In Reinforcement learning, it prevents overly deterministic policies by incorporating entropy into the reward function Hazan and Van Soest [2019], Ahmed [2019]. (iv) Variational inference & generative Models: The entropy appears in ELBO Kingma and Welling [2013b] for posterior approximation and mitigates mode collapse in GANs and VAEs Alemi [2016], Belghazi [2018].

Challenges in Entropy Estimation: Despite its simple definition, entropy is analytically tractable only for limited distributions. For instance, for a uniform $p(x) = \frac{1}{b-a}$ for $x \in [a, b]$ and $p(x) = 0$ for $x \notin [a, b]$ the entropy is $\mathcal{H}(p) = \frac{1}{2}[1 + \log(2\pi\sigma^2)]$. for a Gaussian $p(x) = \mathcal{N}(\mu, \sigma^2)$, the entropy is $\mathcal{H}(p(y|\mu, \sigma^2) = \frac{1}{2}\left(1 + \log(2\pi\sigma^2)\right)$. For general distributions, numerical integration (*e.g.*, Monte Carlo) is required as direct computation is often infeasible. Different methods have been developed for entropy estimation from samples.

Entropy estimation methods from samples can be classified into:

- *Plug-in Estimators:* Estimate density from data, then apply entropy formula. Given a sample $x = \{x_i\}_{i=1}^{M}$, the plug-in method estimates the pdf $\hat{p}(x)$ from the data and then substitutes this estimate into the entropy formula: $\mathcal{H}^{\text{PLUGIN}}(p) \approx -\frac{1}{M}\sum_{i=1}^{M} \log \hat{p}(x_i)$. This approach was first proposed by Dmitriev et al. Dmitriev and Tarasenko [1973] and later investigated by others using kernel density estimator Joe [1989], Hall and Morton [1993], Moon et al. [2018], Pichler [2022], histogram estimator Györfi and Van der Meulen [1987], Hall and Morton [1993] and field-theoretic approaches Chen et al. [2018]. Early approaches leverage kernels that capture pairwise distances between the particles. For instance, Parzen-Rosenblatt estimator Rosenblatt [1956], Parzen [1962]: $\hat{p}(x) = \frac{1}{w^p n}\sum_{i=1}^{M} \kappa\Big(\frac{x - x_i}{w}\Big)$, where w denotes the bandwidth and κ is a kernel density. The resulting entropy estimator was analyzed by Ahmad and Lin [1976]. Schraudolph [2004] extended this approach using a kernel estimator: $\hat{p}(x) = \frac{1}{M}\sum_{i=1}^{M} \kappa_{\Sigma_i}(x - x_i)$, where $\Sigma = (\Sigma_1, \cdots, \Sigma_n)$ are distinct diagonal covariance matrices and $\kappa_\Sigma(x) \sim \mathcal{N}(0, \Sigma)$ is a centered Gaussian density with covariance matrix Σ. Pichler [2022] introduced KNIFE, a kernel-based estimator for density estimation (DE) defined as: $\hat{p}^{\text{KNIFE}}(x; \theta) = \sum_{i=1}^{M} \mu_i \kappa_{\Sigma_i}(x - b_i)$, where $\Sigma = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n)$, and $\theta = (\Sigma, b, \mu)$, with the constraints $\sum_{i=1}^{M} \mu_i = 1$. The covariance matrices Σ_i are symmetric and positive definite but not necessarily diagonal. Despite its advantages, the method has a significant limitation in its simple structure, being restricted to either individual Gaussian kernels or Gaussian Mixture Models (GMMs) with a fixed number of components n. This can limit its flexibility in modeling complex data distributions. Traditional off-the-shelf density estimators often suffer from key drawbacks, such as non-differentiability, computational intractability, or an inability to adapt to changes in the underlying data distribution. These limitations make them unsuitable for applications requiring integration into neural network training pipelines as regularizers. To improve density estimation for non-negative random variables, recent studies have suggested replacing Gaussian kernels with Poisson weight-based estimators to fit counts or rate-based data Chaubey and Sen [2013] defined as: $\hat{p}^{\text{POIS}}(x) = k \sum_{i=0}^{\infty} \Big(F_n(\frac{i+1}{k}) - F_n(\frac{i}{k})\Big) e^{-kx} \frac{(kx)^i}{i!}$, where $F_n(.)$ is the empirical distribution function, and k is a smoothing parameter. Additionally, the concept of learning kernel parameters end-to-end has been explored, providing a foundation for modern differentiable approaches. The idea of learning kernel parameters end-to-end has also been explored previously Viola and Sejnowski [1995], Schraudolph [2004], providing a foundation for modern differentiable approaches.

- *Sample-spacing Estimates* use distances between ordered samples (*e.g.*, Vasicek estimator Vasicek [1976]). Sample spacing methods rely on the spacing of sorted samples and was initiated by Vasicek Vasicek [1976]: $H^{Vasicek}(p) \approx -\frac{1}{M} \sum_{i=1}^{M} \log \Big(\frac{n}{2m}\Big(x_{i+1} - x_i\Big)\Big)$, where x_i are the order statistics and m is a positive integer smaller than $\frac{n}{2}$. One of the greatest weakness of sample-spacing-based estimator is the choice of spacing parameter m, which does not have the optimal form.

- *Nearest-Neighbor Methods*: leverage distances to k-th nearest neighbor Kraskov [2004]. This method estimates entropy using distances to the k-th nearest neighbor in the sample space Kraskov [2004], *i.e.*, $\mathcal{H}(p) \approx \psi(n) - \psi(k) + \log(c_d) + \frac{d}{n} \sum_{i=1}^{M} \log \epsilon_i$, where ψ is the digamma function defined as the logarithmic derivative of the gamma function $\frac{d}{dx} \ln(\Gamma(x))$, c_d is the volume of the unit d-dimensional ball, and ϵ_i is the distance to the k-th nearest neighbor.

- *Variational Inference*: Optimizes a surrogate distribution $q(x)$ to approximate $p(x)$ Kingma and Welling [2013a]. The entropy is computed as Kingma and Welling [2013a]: $\mathcal{H}(p) \approx -\mathbb{E}_{q(x)}[\log q(x)]$, where $q(y)$ is optimized to approximate $p(x)$. q is chosen to be easy to sample from, *e.g.*, Gaussians, GMMs and Normalizing Flows Rezende and Mohamed [2015a].

- *Mutual Information (MI) Estimators*: Approximate entropy indirectly via MI relationships, *i.e.*, $I(x, y) = \mathcal{H}(p_x) + \mathcal{H}(p_y) - \mathcal{H}(p_{x,y})$, Belghazi [2018], where $p_{x,y}$ is the joint distribution and $p_x \cdot p_y$ is the product of the marginal distributions p_x and p_y. Neural networks were used to approximate the mutual information between two variables using the Donsker-Varadhan representation of the KL-Divergence Donsker and Varadhan [1975]: $D_{\text{KL}}(p\|q) = \sup_{T \in \mathcal{T}} \left(\mathbb{E}_p[T(x)] - \log \mathbb{E}_q[e^{T(x)}]\right)$, where $\mathcal{T}$ is a class of functions where $P_{x,y}$ is the joint distribution and $p_x \cdot p_y$ is the product of the marginal distributions. The MI lower bound is expressed as: $I_\theta(x; y) = \sup_\theta \left(\mathbb{E}_{p_{x,y}}[T_\theta(x, y)] - \log \mathbb{E}_{p_x \cdot p_y}[e^{T_\theta(x,y)}]\right)$, where: $T_\theta(x, y)$ is the output of a neural network parameterized by θ, $\mathbb{E}_{p_{x,y}}[T_\theta(x, y)]$ is the expectation over samples from the joint distribution $p_{x,y}$, and $\mathbb{E}_{p_x \cdot p_y}[e^{T_\theta(x,y)}]$ is the expectation over samples from the product of the marginals. The neural network is trained to maximize this bound, providing an approximation of $I(x; y)$. If two of these three entropies $\mathcal{H}(p_x)$, $\mathcal{H}(p_y)$ or $\mathcal{H}(p_{x,y})$ are available, the third one can be computed.

- *Ensemble Methods:* Weight different entropy estimators adaptively Sricharan et al. [2013]. The estimators in the ensemble are assigned different weights, and the overall entropy estimate is calculated as a weighted combination of the individual estimators where optimal weights are determined by solving a convex optimization problem. Ariel and Louzoun [2020] proposed an innovative approach to estimating the entropy of high-dimensional data by decomposing the target entropy into two components: $\mathcal{H}^{\text{CADEE}}(x) = \sum_{i=1}^{d} x_i + \mathcal{H}_{\text{copula}}$, where $\mathcal{H}(y)$ is the total entropy of the multivariate distribution, $\mathcal{H}(x_i)$ is the marginal entropy of each variable, and H_{copula} represents the entropy of the copula, capturing the dependencies between variables. The idea comes from the fact that any density distribution $p(x)$ can be decomposed as the following: $p(x) = p_1(x_1) \ldots p_d(x_d) c\Big(F_1(x_1), \ldots, F_d(x_d)\Big)$, where $c(u_1, \ldots, u_d)$ is the density of copula. The copula entropy is estimated *recursively* by splitting the data into subgroups based on statistically dependent dimensions. This recursive process (1) identifies pairs or groups of dimensions with high statistical dependence, (2) splits the data along these dimensions and (3) repeats the process within each subgroup until the dependencies are resolved. [Kandasamy, 2015] proposed a leave-one-out technique to improve the robustness of entropy estimation using the von Mises expansion-based estimator. The key idea is to iteratively remove one data point from the sample and compute the entropy estimate using the remaining data points. This procedure helps reduce bias and ensures that the estimator is not overly influenced by any single data point. The leave-one-out entropy is given by: $\mathcal{H}^{\text{LOO}}(x) = \frac{1}{M}\sum_{i=1}^{M}\mathcal{H}(x_{-i})$ where $\mathcal{H}(x_{-i})$, is calculated for $x_{-i} = \{x_1, ..., x_{i-1}, x_{i+1}, ..., x_n\}$. This approach provides a more robust estimate of the entropy by mitigating the influence of outliers or anomalous data points.

A summary of these methods is provided in Tab. 3.

Method	**Formula**	**Key Idea**
Analytical	$\mathcal{H}(x)$	Closed-form expressions
Plugin	$-\frac{1}{M}\sum_{i=1}^{M}\log \hat{p}(x_i)$	Sampling-based estimation
KDE	$-\frac{1}{M}\sum_{i=1}^{M}\log\left(\frac{1}{nh}\sum_{j=1}^{M}\kappa\left(\frac{x_i-x_j}{h}\right)\right)$	Density smoothing
KNIFE	$\sum_{i=1}^{M}\mu_i\kappa_{\Sigma_i}(x-b_i)$	Kernel-based estimator
Nearest-Neighbor	$\psi(n)-\psi(k)+\log(c_d)+\frac{d}{n}\sum_{i=1}^{M}\log\epsilon_i$	Distance-based estimation
Vasicek	$-\frac{1}{M}\sum_{i=1}^{M}\log\left\{\frac{n}{2m}\left(x_{i+1}-x_i\right)\right\}$	Sorted sample spacing
Variational Inference	$-\mathbb{E}_{q(x)}[\log q(x)]$	Surrogate distribution
MINE	$\sup_\theta\left(\mathbb{E}_{p_{x,y}}[T_\theta(x,y)]-\log\mathbb{E}_{p_x\cdot p_y}[e^{T_\theta(x,y)}]\right)$	Calculate it via Informtion
CADEE	$=\sum_{i=1}^{d}\mathcal{H}(y_i)+\mathcal{H}_{\text{copula}}$	Marginal via copula
LOO	$\frac{1}{M}\sum_{i=1}^{M}\mathcal{H}(x_{-i})$	Data driven approach

Table 3: Summary of Differential Entropy Approximations

7.2 Sampling-based Variational Inference.

Bridging the gap between parametric variational inference (VI) and Markov Chain Monte Carlo (MCMC) has been a key research focus to achieve both expressivity and scalability in inference. A central challenge is deriving an analytical expression for the marginal distribution of the last sample in an MCMC chain, which is often intractable. To address this, prior work [Salimans et al., 2015, Geffner and Domke, 2023] introduced auxiliary variables to construct augmented variational distributions that include all samples from the chain. However, this approach requires optimizing a looser ELBO and estimating the reverse Markov kernel, which introduces additional parameters and complex design choices. Several extensions have been proposed to avoid estimating the reverse kernel: (i) Hoffman [2017] optimize ELBO with respect to the initial distribution and only uses the MCMC steps to produce "better" samples to the target distribution. However, this method lacks direct feedback between the final marginal distribution and variational parameters, limiting full unification of VI and MCMC, (ii) Caterini et al. [2018] propose a deterministic Hamiltonian MCMC by removing resampling and the accept-reject step. However, this sacrifices MCMC guarantees, (iii) Thin et al. [2020] introduce MetFlow, a Metropolis-Hastings method that models the proposal distribution as a normalizing flow, removing the need for inverse kernel estimation. **MET-SVGD** has several advantages compared with the aforementioned approaches: It computes the exact loglikelihood, *i.e.*, via using the change of variable formula (Sec. 6.2). Hence, there is no need in the variational approximation on the joint distribution of the samples of the Markov chain, to estimate the reverse dynamics. Besides, it leverages knowledge of the unormalized density unlike classical flow models. This makes our approach very easy to integrate in modern day deep learning pipelines. The idea of approximating log-likelihoods for distributions known up to a normalization constant using MCMC and the change-of-variable formula was first explored by [Dai et al., 2019b], applying it to Hamiltonian Monte Carlo (HMC) and Langevin Dynamics (LD). Since, they augment the input with noise or velocity variable for LD and HMC, respectively, the derived log-likelihood of the sampling distribution turns out to be –counter-intuitively– independent of the sampler's dynamics and equal to the initial distribution, which is then parameterized using a normalizing flow model [Kobyzev et al., 2020]. Our derived log-likelihood is more intuitive as it depends on the SVGD dynamics.

7.3 Normalizing Flows, Residual Flows and Neural ODEs

We review Normalizing Flows in general and focus on residual flows as **MET-SVGD** is one. We also draw the connection to neural ODEs. **Normalizing Flows** are generative models that produce tractable distributions where both sampling and density evaluation can be efficient and exact. This is achieved by transforming a simple probability distribution (*e.g.*, a standard normal) into a more complex distribution by a sequence of invertible and differentiable mappings. The density of a sample can be evaluated by transforming it back to the original simple distribution and then computing the product of the density of the inverse-transformed sample under this distribution and the associated change in volume induced by the sequence of inverse transformations. The change in volume is the product of the absolute values of the determinants of the Jacobians for each transformation, as required by the change of variables formula (See App.6.2). Formally, Let $x = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$ be a random variable with a known and tractable probability density function $p_x : \mathbb{R}^d \to \mathbb{R}$. Let g be an invertible function and $x = F(z)$. Then using the change of variables formula, one can compute the probability density function of the random variable y:

$$p_x(x) = p_z(F^{-1}(x))\Big|\det \nabla_x g^{-1}(x)\Big| \tag{9}$$

Intuitively, if the transformation F can be arbitrarily complex, one can generate any distribution p_x from any base distribution p_z under reasonable assumptions on the two distributions. This has been formally proven [Bogachev et al., 2005]. However, constructing arbitrarily complicated non-linear invertible functions can be difficult. Additionally, F should be sufficiently expressive to model the distribution of interest and computationally efficient, both in terms of computing F, its inverse and the determinant of the Jacobian $\nabla_x F^{-1}(x)$.

Different types of flows have been constructed: (1) **Elementwise Flows**, (2) **Linear Flows**, (3) **Planar Flows**, (4) **Radial Flows**, (5) **Coupling Flows**, (6) **Autoregressive Flows**, and (7) **Residual Flows**, which we focus on due to relevance to **MET-SVGD**.

Residual Flows are compositions of the function of the form $g(x) = x + \phi(x)$. The first attempts to build a reversible network architecture based on residual connections was motivated by saving memory (each layer activation can be reconstructed from the previous layer) Gomez et al. [2017], Jacobsen et al. [2018] and was achieved via partitioning units in each layer into two groups and defining coupling functions as:

$$y^A = x^A + F(x^B)y^B = x^B + G(y^A), \tag{10}$$

where $x = (x^A, x^B)$ and $y = (y^A, y^B)$ are respectively the input and output activations, $F : \mathbb{R}^{D-d} \to \mathbb{R}^d$ and $G : \mathbb{R}^d \to \mathbb{R}^{D-d}$ are residual blocks. The Jacobian of such a transformation is, however inefficient to compute and constrains the architecture. To address this, to enable unconstrained architectures for each residual block, Behrmann et al. [2019] proved the following statement:

Proposition 7.1. *A residual connection is invertible if the Lipschitz constant of the residual block is Lip*$(\phi) < 1$, *where Lip*$(\phi) = \sup_{x\neq y} \frac{|\phi(x)-\phi(y)|}{|x-y|}$. *By the mean value theorem 6.6, if* ϕ *is differentiable* $\forall x$, *then Lip*$(\phi) = \sup_x \|\nabla_x \phi(x)\|_2$ *with* $\|\cdot\|$ *being the spectral norm.*

The detailed proof is in (App. 8.1) . Controlling the Lipschitz constant of a neural network is not trivial. Note, that regularizing the spectral norm of the Jacobian of ϕ Sokolić et al. [2017] only reduces it locally and does not guarantee the above condition. Instead, Jacobsen et al. [2018] proposes constraining the spectral radius of each convolutional layer in this network to be less than one.

In residual flows, the density is also derived using the change of variable formula (App. 6.2). A different approach is proposed to approximate the log-det term:

$$\log|\det(I + \nabla_x\phi(x)| \overset{(i)}{=} \mathrm{Tr}(\log(I + \nabla_x\phi(x))) \overset{(ii)}{=} \sum_{k=1}^{\inf}(-1)^{k+1}\frac{\mathrm{Tr}(\nabla_x\phi(x))^k}{k}$$

Where (i) is obtained using the matrix identity result $\log\det(A) = \mathrm{Tr}(\log(A))$ for non-singular $A \in \mathbb{R}^{d\times d}$ Withers and Nadarajah [2010] and (ii) follows from replacing the trace of the matrix by its power series. By truncating this series one can calculate an approximation to the log Jacobian determinant. To efficiently compute each member of the truncated series, the Hutchinson trick is used.

However, this resulted in a biased estimate of the log Jacobian determinant. An unbiased stochastic estimator was proposed by [Chen et al., 2019]. In a model they called a Residual flow [Chen et al., 2019], the authors used a Russian roulette estimator instead of truncation. Informally, the next term is added to the partial sum while calculating the series, one flips a coin to decide if the calculation should be continued or stopped.

Neural ODEs. Due to the similarity of ResNets and Euler discretizations, there are many connections between the i-ResNet and ODEs. Residual connections can be viewed as discretizations of a first order ordinary differential equation (ODE) [Haber et al., 2017]:

$$\frac{d}{dt}\mathbf{x}(t) = F(\mathbf{x}(t), \theta(t)), \tag{11}$$

where $F : \mathbb{R}^D \times \Theta \to \mathbb{R}^D$ is a function which determines the dynamic (the *evolution function*), Θ is a set of parameters and $\theta : \mathbb{R} \to \Theta$ is a parameterization. The discretization of this equation (Euler's method) is

$$\mathbf{x}_{n+1} - \mathbf{x}_n = \varepsilon F(\mathbf{x}_n, \theta_n), \tag{12}$$

and this is equivalent to a residual connection with a residual block $\varepsilon F(\cdot, \theta_n)$.

7.4 Stein Variational Gradient Descent

In the following, we provide an explanation of the RBF kernel variance and its effect on the SVGD dynamics, followed by the formal derivation of SVGD and related work on its convergence rate.

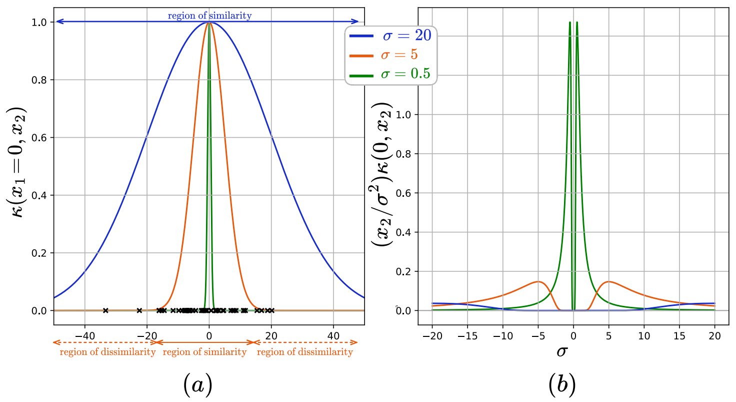


Figure 14: (a) Regions of similarity/dissimilarity for an RBF $\kappa(x_1, x_2)$ evaluated at $x_1 = 0$. (b) Repulsion term in the SVGD update as a function of σ.

RBF Kernel Variance Interpretation. RBF kernels are the most generalized form of kernelization and is one of the most widely used kernels due to its similarity to the Gaussian distribution. The RBF kernel function for two points x_1 and x_2 computes the similarity or how close they are to each other. This kernel can be mathematically represented as follows: $\kappa(x_1, x_2) = \exp(-\frac{\|x_1 - x_2\|^2}{\sigma^2})$, where σ is the kernel variance and $\|x_1 - x_2\|$ is the L_2 distance between x_1 and x_2. The maximum value that the RBF kernel can reach is 1 when $x_1 = x_2$. When a large distance separates the points, the kernel value is less than 1 and close to 0 indicating dissimilarity between x_1 and x_2. This also means that the particles are independent, *i.e.*, : they follow their own gradients (the expectation in the SVGD update is reduced to one term corresponding to $x_i = x_j$). The width of the region of similarity is controlled by σ, *i.e.*, a larger sigma results in a larger region of similarity with $\kappa(x_1, x_2) \neq 0$ (Fig. 14 which also means that the particle update is impacted by its neighbors' gradients (b)).

Setting σ in the SVGD update rule;

$$x^{l+1} = x^l + \epsilon \mathbb{E}_{x_j^l}\Big[\underbrace{\kappa(x^l, x_j^l)\nabla_{x_j^l}\log p(x_j^l)}_{\text{drift term}} + \underbrace{\frac{(x^l - x_j^l)}{\sigma^2}\kappa(x^l, x_j^l)}_{\text{repulsion term}}\Big], \tag{13}$$

is not obvious: σ in the drift term determines the neighboring samples x_j^l that will contribute with their scores to the update. A larger σ implies, more influence from the neighbors. For the repulsion term, both a very small or a very large σ value can result in setting the repulsion term to 0 as shown in Fig. 14 (a). Classically, the median trick is used to set the σ, *i.e.*, $\sigma_{\text{med}} = \text{median}\{\|x_i^l - x_j^l\|\}_{i,j=1}^M / \log M$ with M being the number of particles. In our experiments, we show that this is suboptimal and that that a more optimal σ can be learnt end-to-end via minimizing the KL-divergence (Eq. 3.2).

SVGD Derivation. Liu and Wang [2016] The goal is to approximate a target via a variational distribution $q \in \mathcal{Q}$ *i.e.*, :

$$q^* = \arg\min_{q \in \mathcal{Q}} D_{KL}(q||p).$$

$\mathcal{Q}$ is obtained by transforming a reference density q^0 via an invertible map $F : \mathcal{X} \to \mathcal{X}$, where for any particle $x \sim q^0$, we define $y = F(x)$. The distributions of y and x are related by CVF (App. 6.2):

$$q_{[F]}(y) = q(F^{-1}(y)) \cdot |\det(\nabla_y F^{-1}(y))|$$

In this setup, $F(x)$ is chosen to have a specific form: $F(x) = x + \epsilon\phi(x)$, where ϵ is a stepsize and ϕ is a perturbation direction chosen to maximally decrease the KL divergence:

$$\phi^* = \arg\max_{\phi}\{D_{KL}(q||p) - D_{KL}(q_{[F]}||p)\}\} = \arg\max_{\phi\in\mathcal{F}} \nabla_\epsilon D_{KL}(q_{[F]}||p)$$

This maximization has a closed form expression if we constrain the space of perturbations $\mathcal{F}$ to be a reproducing kernel Hilbert space (RKHS) with a positive kernel $\kappa(\cdot,\cdot)$, and $\|\phi\|_{\mathcal{F}} \leq 1$. In this case $\arg\max_{\phi\in\mathcal{F}} \nabla_\epsilon D_{KL}(q_{[F]}||p) = \mathbb{E}_q[\text{Tr}(\mathcal{A}_p\phi)]$. The optimal perturbation direction ϕ^* is, hence, the one that maximizes the Stein Discrepancy [Liu et al., 2016]:

$$\mathbb{S}(q,p) = \max_{\phi\in\mathcal{F}}\{\mathbb{E}_q[\text{Tr}(\mathcal{A}_p\phi)] \quad s.t \quad \|\phi\|_{\mathcal{F}} \leq 1\}$$

and given by:

$$\phi^*_{p,q}(.) = \mathbb{E}_q\Big[\kappa(x,.)\nabla_x \log p + \nabla_x\kappa(x,.)\Big].$$

SVGD Convergence Rate. SVGD is difficult to analyze theoretically because it involves a system of particles that interact with each other in a complex way. In the infinite particles case, [Liu, 2017] proved that SVGD converges (weakly) to p in KSD. [Korba et al., 2020, Salim et al., 2022, Sun et al., 2023] refined these results with path-independent constants, weaker smoothness conditions, and explicit rates of convergence. [Duncan et al., 2023] provides conditions for exponential convergence. For the finite particles case, [Liu, 2017] shows that finite particles SVGD converges to infinite particles SVGD in bounded-Lipschitz distance but only under boundedness assumptions violated by most applications of SVGD. [Korba et al., 2020] explicitly bounded the expected squared Wasserstein distance between n-particle and continuous SVGD but only under the assumption of bounded $\log p$. Also they do not provide convergence rates.[Liu et al., 2024] show that SVGD with finite particles achieves linear convergence in KL divergence under a very limited setting where the target distribution is Gaussian. [Shi and Mackey, 2024] shows that SVGD convergence rate is $\mathcal{O}(1/\sqrt{\log\log n})$ under the assumption that the target is sub-Gaussian with a Lipschitz score.

7.5 Metropolis–Hastings

The Metropolis–Hastings algorithm's goal is to generate a Markov Chain $\{x^{(l)}\}_{l=0}^{\infty}$ that simulates samples from a given probability distribution p. The chain starts with samples from an initial distribution $q^{(0)}$ and updates its state by leveraging a proposal distribution $q(\tilde{x}|x^{(l)})$ as

$$x^{(l+1)}|x^{(l)} = \begin{cases} \tilde{x}, & \text{if } \alpha^l \leq \frac{p(\tilde{x})q(x^{(l)}|\tilde{x})}{p(x^{(l)})q(\tilde{x}|x^{(l)})} \\ x^{(l)}, & \text{otherwise} \end{cases}$$

where $\alpha^{(l)} \sim \mathcal{U}(0,1)$.

Importantly, because the update only involves ratios of p, its normalization constant is not required. Furthermore, by construction, a chain that is constructed using the Metropolis-Hastings algorithm is reversible [Tierney, 1994], which means that if $x^{(0)} \sim p$, then $x^l \sim p$ for all iterations l.

As an example, the Metropolis-Adjusted Langevin Algorithm employs the following proposal distribution

$$q(\tilde{x}^{(l+1)}|x^{(l)}) = \mathcal{N}_d\left(x^{(l)} + \epsilon\nabla\log p(x^{(l)}), 2\epsilon I_d\right)$$

7.6 Convergence of Metropolis Hastings

Under relatively weak conditions, generating samples from an MCMC algorithm such as Metropolis-Hastings asymptotically draws samples from the target distribution [Robert, 1999]. The finite number of steps required for the marginal distribution of the Markov chain to reach the target under a discrepancy measure, has been heavily studied for both the total variation and Wasserstein distances [Carlo, 2001, Jones and Hobert, 2004, Rosenthal, 1995, Villani et al., 2009]. A popular approach is to show geometric ergodicity and provide an exponential convergence rate to the target distribution from any point of initialization in total variation. Explicit convergence rates have been rare with the exception of some Metropolis-Hastings independence samplers [Tierney, 1994]. To quantify said convergence, discrepancy measures are used. Notably, the total variation distance between two densities p and q defined as: $d_{\text{TV}}(p,q) = \frac{1}{2}\int_{\mathcal{X}} |p(x) - q(x)| dx$.

An upper bound on the convergence rate can be computed as:

$$d_{\text{TV}}(q^L, p) \leq \left(1 - \frac{1}{\beta}\right)^L \quad \text{with } \beta = \sup_{x \in \mathcal{X}} \frac{p(x)}{q(x)}$$

A lower bound can be computed as:

$$d_{\text{TV}}(q^l, p) \geq (1 - \alpha(x))^l \quad \text{with } \alpha(x) = \mathbb{E}\left[\min\left(\frac{p(\tilde{x})q(x \mid \tilde{x})}{p(x)q(\tilde{x} \mid x)}, 1\right)\right]$$

In our case computing the lower bounds for **MET-SVGD** is possible as we have a closed-form expression for the acceptance probability.

Notation: We start by introducing the notation for this section. We compute the first and second order derivatives of the kernel as follows:

$$\forall i,j \in \{1..M\}^2 \quad \gamma = \frac{1}{2\sigma^2} \quad \text{and} \quad \delta_{i,j} = (x_i^l - x_j^l) \quad \text{hence we express } \kappa, \nabla_{x_i}\kappa, \nabla_{x_i}\nabla_{x_j}\kappa \text{ as follows:}$$

$$\begin{aligned}
\kappa(x_i^l, x_j^l) &= \exp(-\gamma\|x_i^l - x_j^l\|^2),\\
\nabla_{x_j^l}\kappa(x_i^l, x_j^l) &= 2\gamma\delta_{i,j}\kappa(x_i^l, x_j^l)\\
\nabla_{x_i^l}\kappa(x_i^l, x_j^l) &= -2\gamma\delta_{i,j}\kappa(x_i^l, x_j^l) = -\nabla_{x_j^l}\kappa(x_i^l, x_j^l)\\
\nabla_{x_i^l}\nabla_{x_j^l}\kappa(x_i^l, x_j^l) &= \nabla_{x_i^l}\left(2\gamma\delta_{i,j}\kappa(x_i^l, x_j^l)\right) = 2\gamma\left(I - 2\gamma\delta_{i,j}\delta_{i,j}^T\right)\kappa(x_i^l, x_j^l)
\end{aligned}$$

8 SVGD Density Derivation

Theorem 8.1. *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be an invertible transformation of the form $F(x) = x + \epsilon\phi(x)$. We denote by $q^L(x^L)$ the distribution obtained from repeatedly (L times) applying F to a set of action samples (called "particles") $\{x^0\}_{i=1}^M$ from an initial distribution $q^0(x^0)$, i.e., $x^L = F \circ F \circ \cdots \circ F(x^0)$. Under the condition $\epsilon < \epsilon_{UB}^l = 1/\sup_x \sqrt{\mathrm{Tr}(\nabla\phi^l(x)\nabla\phi^{l,T}(x)}$,, $\forall l \in [0..L]$, the closed-form expression of $\log q^L(x^L)$ is:*

$$\log q^L(x^L) = \log q^0(x^0) - \epsilon \sum_{l=0}^{L-1} \mathrm{Tr}(\nabla_{x^l}\phi(x^l)) + \mathcal{O}(\epsilon^2) \tag{14}$$

Proof. Based on the change of variable formula (6.2), when for every iteration $l \in [1, L]$, the transformation $x^l = F(x^{l-1})$ is invertible and we have:

$$q^l(x^l) = q^{l-1}(x^{l-1}) \left|\det \nabla_{x^l}\phi(x^l)\right|^{-1}, \forall l \in [1, L].$$

By induction, we derive the probability distribution of sample x^L:

$$q^L(x^L) = q^0(x^0) \prod_{l=0}^{L-1} \left|\det\left(I + \epsilon\nabla_{x^l}\phi(x^l)\right)\right|^{-1}$$

By taking the log for both sides, we obtain:

$$\log q^L(x^L) = \log q^0(x^0) - \sum_{l=0}^{L-1} \log\left|\det\left(I + \epsilon\nabla_{x^l}\phi(x^l)\right)\right|.$$

This, however, requires computing the Jacobian $\nabla_{x^l}\phi(x^l)$. Next, we show that $\log\Big|\det\big(I + \epsilon\nabla_{x^l}\phi(x^l)\big)\Big|$ can be approximated efficiently via $\epsilon\mathrm{Tr}(\nabla_{x^l}\phi(x^l)) + O(\epsilon^2)$ under an assumption on the learning rate in section 8.3, that's satisfied by the invertibility assumption (Sec.8.1) and derive the expression of $Tr(\nabla\phi)$ for the RBF, Bilinear and DKEF Kernels (Sec 8.5). □

8.1 Sufficient Condition For $x + \epsilon\phi(x)$ Invertibility (Prop. 3.1)

Proposition 3.1 (Sufficient condition for invertible SVGD).

Let $f : \mathbb{R}^d \to \mathbb{R}^d$ with $f = (f^1 \circ \cdots \circ f^L)$ denote a sequence of SVGD updates with $f^l = I + \epsilon\phi^l$. We denote by Lip$\big(\phi^l\big)$ the Lipschitz constant of the velocity ϕ^l at step l. f is invertible if ϵ Lip$\big(\phi^l\big) < 1$, for all $l \in [0, L-1]$.

Proof. Given x^{l+1}, the goal is to find x^l. We denote by c x^{l+1} resulting in $x^{l+1} = c - \epsilon\phi(x^l)$. Hence, we are interested in the invertibility of the function $g(x) = c - \epsilon\phi(x)$. for this, we show that g is a contractive mapping:

$$\begin{aligned} d(g(x), g(\tilde{x})) &= d(c - \epsilon\phi(x), c - \epsilon\phi(\tilde{x})) \\ &\overset{(i)}{=} d(-\epsilon\phi(x), -\epsilon\phi(\tilde{x})) \\ &\overset{(ii)}{=} |\epsilon| d(\phi(x), \phi(\tilde{x})) \\ &\overset{(iii)}{\leq} |\epsilon| K \cdot d(x, \tilde{x}), \quad \text{with } |\epsilon| K < 1 \end{aligned}$$

(i) The distance is translation invariant.

(ii) The distance is absolutely homogeneous.

(iii) $\epsilon\phi$ is a contractive mapping, *i.e.*, , $d(\epsilon\phi(x), \epsilon\phi(\tilde{x})) \leq \epsilon K d(x, \tilde{x})$ with $\epsilon K \leq 1$. Note that $\text{Lip}(\epsilon\phi) = \sup_x \frac{d(\epsilon\phi(x), \epsilon\phi(\tilde{x}))}{d(x,\tilde{x})} = \epsilon K$

Therefore, $g(x)$ is a contractive mapping, and by the **Banach fixed point theorem** 2, it has a unique fixed point. This implies that the inverse of the mapping $x^{l+1} = x^l + \epsilon\phi(x^l)$ exists and is unique.

Hence, we demonstrate that $f^l = I + \epsilon\phi^l$ is invertible if $\epsilon\,\text{Lip}\big(\phi^l\big) < 1$. Since f is a composition of f^l ($l \in [1 \cdots L[$), we conclude that f is invertible. □

8.2 A sufficient condition for invertibility check - an upper bound (Corr. 3.3)

Corollary 3.3. *The distribution induced by the SVGD update (Eq. 1) using an RBF kernel is given by Eq. 2 if* $\epsilon < \epsilon^l_{UB} = 1/\sup_x \sqrt{\text{Tr}(\nabla\phi^l(x)\nabla\phi^{l,T}(x))} \quad \forall l \in [0, L-1]$

Proof. $x^{l+1} = x^l + \epsilon\phi(x^l)$ is invertible if $\text{Lip}\,(\phi(x)) < 1$ as we demonstrate in (App. 8.1)

We compute the Lipschitz constant:

$$\text{Lip}(\phi) = \sup_x \frac{\|\phi(x) - \phi(y)\|}{\|x - y\|} \overset{(i)}{=} \sup_x \|\nabla_x\phi(x)\|_2 \overset{(ii)}{=} \sup_x \sigma_{\max}\{\nabla_x\phi(x)\}$$

(i) We consider the ℓ_2 norm in computing the operator norm.

(ii) Using the definition of the Lipschitz constant via Jacobian norm: $\|\phi(x) - \phi(y)\| \leq \sup_x \|\nabla_x\phi(x)\| \cdot \|x - y\|$.

The following always holds: $\lambda_{\max}\{\nabla\phi\}$ is upper bounded by $\|\nabla\phi\|_2 \leq \sqrt{\text{Tr}(\nabla\phi\nabla\phi^T)}$. □

8.3 A sufficient condition for log-det approximation (Prop. 3.2)

Proposition 3.2(Condition for log-det Approximation) *Let* $\phi^l : \mathbb{R}^d \to \mathbb{R}^d$, $\log|\det(I + \epsilon\nabla\phi^l)| = \epsilon\text{Tr}(\nabla\phi^l)$ *if* $\epsilon\,|\lambda_{max}\big(\nabla\phi^l\big)| < 1$ *for all* $l \in [0, L-1]$*, with* λ_{max} *being the largest eigenvalue value and* ∇ *is the gradient operator w.r.t the input.*

Proof. We discuss two approaches leveraging the corollary of Jacobi's formula and the bounds on the eigenvalues of $\nabla_{x^l}\phi(x^l)$:

Method 1 (P-SVGD): Leveraging the Corollary of the Jacobi's formula. Let $A = I + \epsilon\nabla_{x^l}\phi(x^l)$, under the assumption $\epsilon||\nabla_{x_i}\phi(x_i)||_\infty \ll 1$, *i.e.*, $||A - I||_\infty \ll 1$, we apply the collorary of Jacobi's formula (App. 6.4) and get

$$\begin{aligned} \log q^L(x^L) &= \log q^0(x^0) - \sum_{l=0}^{L-1} \text{Tr}\big(\log(I + \epsilon\nabla_{x^l}\phi(x^l))\big) + \mathcal{O}(\epsilon^2) \\ &= \log q^0(x^0) - \epsilon\sum_{l=0}^{L-1} \text{Tr}\big((I + \epsilon\nabla_{x^l}\phi(x^l) - I)\big) + \mathcal{O}(\epsilon^2) \end{aligned}$$

$$= \log q^0(x^0) - \epsilon \sum_{l=0}^{L-1} \operatorname{Tr}\left(\nabla_{x^l}\phi(x^l)\right) + \mathcal{O}(\epsilon^2)$$

In practice, since this bound is informal, [Messaoud et al., 2024] recommend choosing a small enough learning rate.

Method 2 (MET-SVGD): Leveraging bounds on the eigenvalues of $\nabla_{x^l}\phi(x^l)$. In the following we denote by $\lambda_i\{A\}$ the eigenvalue of matrix A

$$\begin{aligned}
\left|\det\left(I + \epsilon\nabla_{x^l}\phi(x^l)\right)\right| &\overset{(i)}{=} \left|\prod_{i=1}^{d} \lambda_i\{I + \epsilon\nabla_{x^l}\phi(x^l)\}\right| = \prod_{i=1}^{d} \left|\lambda_i\{I + \epsilon\nabla_{x^l}\phi(x^l)\}\right| \\
&\overset{(ii)}{=} \prod_{j=1}^{d} \left|1 + \epsilon\lambda_j\{\nabla_{x^l}\phi(x^l)\}\right| = \exp\Big(\sum_{j=1}^{d} \ln\left|1 + \epsilon\lambda_j\{\nabla_{x^l}\phi(x^l)\}\right|\Big) \\
&= \exp\Big(\sum_{j=1}^{d} \ln\left(1 + \epsilon\lambda_j\{\nabla_{x^l}\phi(x^l)\}\right)\Big) \quad \text{if } \lambda_j\{\nabla_{x^l}\phi(x^l)\} > \frac{-1}{\epsilon} \\
&\overset{(iii)}{=} \exp\Big(\sum_{j=1}^{d} \epsilon\lambda_j\{\nabla_{x^l}\phi(x^l)\} + \mathcal{O}(\epsilon^2)\Big) \\
&= \exp\Big(\epsilon\operatorname{Tr}(\nabla_{x^l}\phi(x^l)) + \mathcal{O}(\epsilon^2)\Big)
\end{aligned}$$

(i) By definition of the determinant.

(ii) Let λ_i be the eigenvalue of $\{I + \epsilon\nabla_{x^l}\phi(x^l)\}$ associated with the eigenvector v_i. We show that $\lambda_i - 1$ is the eigenvalue associated with $\epsilon\nabla_{x^l}\phi(x^l)$:

$$\begin{aligned}
&\Leftrightarrow\ (I + \epsilon\nabla_{x^l}\phi(x^l))v_i = \lambda_i v_i \\
&\quad\Rightarrow\ \epsilon\nabla_{x^l}\phi(x^l)v_i = (\lambda_i - 1)v_i \\
&\qquad\qquad\Rightarrow\ \lambda_j = (\lambda_i - 1) \text{ is an eigenvalue of } \epsilon\nabla_{x^l}\phi(x^l)
\end{aligned}$$

(iii) We use Taylor expansion of $\ln(1+\epsilon a) = \sum_i \frac{(-1)^{i-1}(\epsilon a)^i}{i} = \epsilon a + \mathcal{O}(\epsilon^2)$ around $\epsilon a \to 0$.

Hence, under the condition $\lambda_i\{\nabla_{x^l}\phi(x^l)\} > \frac{-1}{\epsilon}$, the approximation $\log\left|\det\left(I + \epsilon\nabla_{x^l}\phi(x^l)\right)\right| = \epsilon\operatorname{Tr}(\nabla_{x^l}\phi(x^l))$ holds exactly.

$$\begin{aligned}
&\lambda_i\{\nabla_{x^l}\phi(x^l)\} > \frac{-1}{\epsilon} \quad \forall i \in [1..d] \\
&\qquad\Leftrightarrow\ \left|\lambda_i\epsilon\right| < 1 \\
&\qquad\Leftrightarrow\ \left|\lambda_i\epsilon\right| < \left|\lambda_{\max}\epsilon\right| < 1 \quad \text{s.t} \quad \forall i \quad \lambda_i < \lambda_{\max} \\
&\qquad\quad\Leftrightarrow\ \epsilon < \underbrace{\frac{1}{|\lambda_{\max}|}}_{\epsilon_{\text{UB}}}
\end{aligned}$$

Even though the condition $\epsilon < \frac{\alpha}{|\lambda_{\max}|}$ is more exact than the one derived by [Messaoud et al., 2024], it's still impractical as it requires computing the Jaccobian.

8.4 Unifying the sufficient conditions for invertibility and $\log|\det(I + \epsilon A) = \epsilon\operatorname{Tr}(A) + \mathcal{O}(\epsilon^2)$

Corollary 8.2. *Following [Wolkowicz and Styan, 1980], Let A be an $d \times d$ complex matrix, and let A^* be the Hermitian of A:*

$$|\lambda_i| \le \sigma_i \le (\operatorname{Tr}(A^*A))^{1/2} \quad \forall i \in [1..d]$$

Where σ_i is the i-th singular value of A.

Proof. In our setup $A = \nabla_{x_i^l}\phi(x_i^l)$, which we can easily compute as illustrated in the following:

$$\nabla_{x_i^l}\phi(x_i^l) = \underbrace{\frac{1}{M}\sum_{j=1, j\neq i}^{M} \nabla_{x_i^l}\kappa(x_i^l, x_j^l)s_p(x_j^l)^T + \nabla_{x_i^l}\nabla_{x_j^l}\kappa(x_i^l, x_j^l)}_{A_i} + \underbrace{\frac{1}{M}\underbrace{\kappa(x_i^l, x_i^l)}_{=1}\nabla_{x_i^l}s_p(x_i^l)^T}_{B_i}$$

Next we compute $\mathrm{Tr}(\nabla_{x_i^l}\phi(x_i^l))$. We denote by A_i and B_i the two terms of $\nabla_{x_i^l}\phi(x_i^l)$:

$$\begin{aligned}
\mathrm{Tr}\big((\nabla_{x_i^l}\phi(x_i^l))^T\nabla_{x_i^l}\phi(x_i^l)\big) &= \mathrm{Tr}(A^TA) = \mathrm{Tr}\big((A_i+B_i)^T(A_i+B_i)\big)\\
&= \mathrm{Tr}\big(A_i^TA_i + B_i^TB_i + A_i^TB_i + A_iB_i^T\big)\\
&= \mathrm{Tr}(A_i^TA_i) + \mathrm{Tr}(B_i^TB_i) + 2\mathrm{Tr}(A_iB_i^T)\\
&= \underbrace{\mathrm{Tr}(A_i^TA_i)}_{(1)} + \underbrace{\mathrm{Tr}(B_i^TB_i)}_{(2)} + \underbrace{2\mathrm{Tr}(B_i^TA_i)}_{(3)}
\end{aligned}$$

For a term by term breakdown:

$$\begin{aligned}
\text{Term (1)} &= \mathrm{Tr}(A_i^TA_i)\\
&= \mathrm{Tr}\Bigg(\Big(\frac{1}{M}\sum_{\substack{j=1\\ j\neq i}}^{M}\overbrace{\nabla_{x_i^l}\kappa(x_i^l, x_j^l)s_p(x_j^l)^T}^{C_{i,j}} + \overbrace{\nabla_{x_i^l}\nabla_{x_j^l}\kappa(x_i^l, x_j^l)}^{D_{i,j}}\Big)^T\Big(\frac{1}{M}\sum_{\substack{r=1\\ r\neq i}}^{M}\underbrace{\nabla_{x_i^l}\kappa(x_i^l, x_r^l)s_p(x_r^l)^T}_{C_{i,r}} + \underbrace{\nabla_{x_i^l}\nabla_{x_r^l}\kappa(x_i^l, x_r^l)}_{D_{i,r}}\Big)\Bigg)\\
&= \mathrm{Tr}\Bigg(\frac{1}{M^2}\sum_{\substack{j=1\\ j\neq i}}^{M}\sum_{\substack{r=1\\ r\neq i}}^{M}\Big({C_{i,j}}^T + {D_{i,j}}^T\Big)\Big(C_{i,r} + D_{i,r}\Big)\Bigg)\\
&= \frac{1}{M^2}\sum_{\substack{j=1\\ j\neq i}}^{M}\sum_{\substack{r=1\\ r\neq i}}^{M}\underbrace{\mathrm{Tr}(C_{i,j}^TC_{i,r} + D_{i,j}^TC_{i,r})}_{(1a)} + \underbrace{\mathrm{Tr}(D_{i,j}^TD_{i,r} + C_{i,j}^TD_{i,r})}_{(1b)}
\end{aligned}$$

$$\begin{aligned}
\text{Term (1a)} &= \mathrm{Tr}(C_{i,r}^TC_{i,j} + D_{i,r}^TC_{i,j})\\
&= \mathrm{Tr}\Big((\nabla_{x_i^l}\kappa(x_i^l, x_r^l)s_p(x_r^l)^T)^T(\nabla_{x_i^l}\kappa(x_i^l, x_j^l)s_p(x_j^l)^T) + (\nabla_{x_i^l}\nabla_{x_r^l}\kappa(x_i^l, x_r^l))^T(\nabla_{x_i^l}\kappa(x_i^l, x_j^l)s_p(x_j^l)^T)\Big)\\
&= \mathrm{Tr}\Big(s_p(x_r^l)\nabla_{x_i^l}\kappa(x_i^l, x_r^l)^T\nabla_{x_i^l}\kappa(x_i^l, x_j^l)s_p(x_j^l)^T + (\nabla_{x_i^l}\nabla_{x_r^l}\kappa(x_i^l, x_r^l))^T(\nabla_{x_i^l}\kappa(x_i^l, x_j^l)s_p(x_j^l)^T)\Big)\\
&= \mathrm{Tr}\Big(4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)s_p(x_r^l)\delta_{i,r}^T\delta_{i,j}s_p(x_j^l)^T - 4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)\big(I - 2\gamma\delta_{i,r}\delta_{i,r}^T\big)\delta_{i,j}s_p(x_j^l)^T\Big)\\
&= \mathrm{Tr}\Big(4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)\big(s_p(x_r^l)\delta_{i,r}^T - I + 2\gamma\delta_{i,r}\delta_{i,r}^T\big)\delta_{i,j}s_p(x_j^l)^T\Big)\\
&= 4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)\big(\delta_{i,r}^Ts_p(x_r^l) - d + 2\gamma\|\delta_{i,r}\|^2\big)s_p(x_j^l)^T\delta_{i,j}
\end{aligned}$$

$$\begin{aligned}
\text{Term (1b)} &= \mathrm{Tr}(D_{i,r}^TD_{i,j} + C_{i,r}^TD_{i,j})\\
&= \mathrm{Tr}\Big((\nabla_{x_i^l}\nabla_{x_r^l}\kappa(x_i^l, x_r^l))^T(\nabla_{x_i^l}\nabla_{x_j^l}\kappa(x_i^l, x_j^l)) + (\nabla_{x_i^l}\kappa(x_i^l, x_r^l)s_p(x_r^l)^T)^T(\nabla_{x_i^l}\nabla_{x_j^l}\kappa(x_i^l, x_j^l))\Big)\\
&= \mathrm{Tr}\Bigg(4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)\Big(I - 2\gamma\delta_{i,r}\delta_{i,r}^T\Big)\Big(I - 2\gamma\delta_{i,j}\delta_{i,j}^T\Big) - 4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)s_p(x_r^l)\delta_{i,r}^T\Big(I - 2\gamma\delta_{i,j}\delta_{i,j}^T\Big)\Bigg)\\
&= \mathrm{Tr}\Bigg(4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)\Big(I - 2\gamma\delta_{i,r}\delta_{i,r}^T - s_p(x_r^l)\delta_{i,r}^T\Big)\Big(I - 2\gamma\delta_{i,j}\delta_{i,j}^T\Big)\Bigg)
\end{aligned}$$

$$= 4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)\Big(d - \delta_{i,r}^T s_p(x_r^l) - 2\gamma|\delta_{i,r}|^2\Big)\Big(d - 2\gamma|\delta_{i,j}|^2\Big)$$

Adding these sub-terms together

$$\begin{aligned}
\textbf{Term ①} &= \frac{1}{M^2}\sum_{\substack{j=1\\j\neq i}}^{M}\sum_{\substack{r=1\\r\neq i}}^{M} 4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)\Big(\delta_{i,r}^T s_p(x_r^l) - d + 2\gamma\|\delta_{i,r}\|^2\Big)s_p(x_j^l)^T\delta_{i,j} \\
&\quad + 4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)\Big(d - \delta_{i,r}^T s_p(x_r^l) - 2\gamma\|\delta_{i,r}\|^2\Big)\Big(d - 2\gamma\|\delta_{i,j}\|^2\Big) \\
&= \frac{1}{M^2}\sum_{\substack{j=1\\j\neq i}}^{M}\sum_{\substack{r=1\\r\neq i}}^{M} 4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)\Big(\delta_{i,r}^T s_p(x_r^l) - d + 2\gamma\|\delta_{i,r}\|^2\Big)\Big(s_p(x_j^l)^T\delta_{i,j} - d + 2\gamma\|\delta_{i,j}\|^2\Big)
\end{aligned}$$

$$\begin{aligned}
\textbf{Term ②} &= \mathrm{Tr}(B_i^T B_i) \\
&= \mathrm{Tr}\left(\frac{1}{M^2}\nabla_{x_i^l} s_p(x_i^l)\Big(\nabla_{x_i^l} s_p(x_i^l)\Big)^T\right) \\
&= \frac{1}{VM^2}\sum_{t=1}^{V} v_t^T\nabla_{x_i^l} s_p(x_i^l)\Big(\nabla_{x_i^l} s_p(x_i^l)\Big)^T v_t \\
&= \frac{1}{VM^2}\sum_{t=1}^{V}\Big|\Big|\nabla_{x_i^l}\Big(v_t^T s_p(x_i^l)\Big)\Big|\Big|^2
\end{aligned}$$

$$\begin{aligned}
\textbf{Term ③} &= \mathrm{Tr}(B_i^T A_i) \\
&\approx \frac{1}{V}\sum_{t=1}^{V} v_t^T\left(\frac{1}{M^2}\sum_{\substack{j=1\\j\neq i}}^{M}\Big[\underbrace{-2\gamma\nabla_{x_i^l} s_p(x_i^l)\delta_{i,j}s_p(x_j^l)^T}_{E_{i,j}} + \underbrace{2\gamma\nabla_{x_i^l} s_p(x_i^l)(I - 2\gamma\delta_{i,j}\delta_{i,j}^T)}_{F_{i,j}}\Big]\kappa(x_i^l, x_j^l)\right)v_t
\end{aligned}$$

Using Hutchinson Trace Estimation Hutchinson [1989]

$$\begin{aligned}
&\approx \frac{1}{V}\sum_{t=1}^{V}\frac{1}{M^2}\sum_{\substack{j=1\\j\neq i}}^{M}\kappa(x_i^l, x_j^l)\Big[v_t^T E_{i,j}v_t + v_t^T F_{i,j}v_t\Big] \\
&\approx \frac{1}{VM^2}\sum_{t=1}^{V}\sum_{\substack{j=1\\j\neq i}}^{M}\kappa(x_i^l, x_j^l)\Big[-2\gamma(v_t^T\nabla_{x_i^l} s_p(x_i^l))(\delta_{i,j}^T v_t)s_p(x_j^l)^T + 2\gamma(v_t^T\nabla_{x_i^l} s_p(x_i^l))(v_t - 2\gamma(\delta_{i,j}^T v_t)\delta_{i,j})\Big]
\end{aligned}$$

By combining **Terms ①**, **②** and **③**, we obtain:

$$\begin{aligned}
(1) + (2) + (3) &= \frac{1}{M^2}\sum_{j=1\ j\neq i}^{M}\sum_{r=1\ r\neq i}^{M} 4\gamma^2\kappa(x_i^l, x_r^l)\kappa(x_i^l, x_j^l)\Big(\delta_{i,r}^T s_p(x_r^l) - d + 2\gamma|\delta_{i,r}|^2\Big)\Big(s_p(x_j^l)^T\delta_{i,j} - d + 2\gamma|\delta_{i,j}|^2\Big) \\
&\quad + \frac{2}{M^2}|\nabla_{x_i^l} s_p(x_i^l)|^2 \\
&\quad + \frac{1}{VM^2}\sum_{t=1}^{V}\sum_{j=1\ j\neq i}^{M}\kappa(x_i^l, x_j^l)\Big[-2\gamma(v_t^T\nabla_{x_i^l} s_p(x_i^l))(\delta_{i,j}^T v_t)s_p(x_j^l)^T + 2\gamma(v_t^T\nabla_{x_i^l} s_p(x_i^l))(v_t - 2\gamma(\delta_{i,j}^T v_t)\delta_{i,j})\Big]
\end{aligned}$$

□

8.5 Computing $\mathrm{Tr}(\nabla_{x^l}\phi(x^l))$ with RBF kernel

We show that the closed-form estimate of the log-likelihood $\log q^L(x^L)$ for the SVGD-based sampler with an RBF kernel $\kappa(\cdot,\cdot)$ is

$$\log q^L(x^L) \approx \log q^0(x^0) - \frac{\epsilon}{M\sigma^2}\sum_{l=0}^{L-1}\sum_{\substack{j=1\\ x^l\neq x_j^l}}^{M}\Big(\kappa(x_j^l,x^l)\Big(-(x^l-x_j^l)^\top\nabla_{x_j^l}s_p(x_j^l)-\frac{\alpha}{\sigma^2}\|x^l-x_j^l\|^2+d\alpha\Big)\Big)-\frac{\epsilon}{M}\mathrm{Tr}\Big(\nabla^2_{x_i^l}\log p(x_i^l)\Big)$$

Proof. We explicitly compute $\mathrm{Tr}(\nabla_{x^l}\phi(x^l))$ as follows:

$$\log q^L(a_i^L) = \log q^0(x_i^0) - \frac{\epsilon}{m}\sum_{l=0}^{L-1}\Bigg[\sum_{\substack{j=1\\ x_i^l\neq x_j^l}}^{m-1}\Big[\underbrace{\mathrm{Tr}\Big(\nabla_{x_i^l}(\kappa(x_i^l,x_j^l)\nabla_{x_j^l}\log p(x_j^l))\Big)}_{①}+\underbrace{\mathrm{Tr}\Big(\nabla_{x_i^l}\nabla_{x_j^l}\kappa(x_i^l,x_j^l)\Big)}_{②}\Big] + \underbrace{\mathrm{Tr}\Big(\nabla^2_{x_i^l}\log p(x_i^l)\Big)}_{③}\Bigg]$$

Next we compute simplifications for all subterms ① and ② respectively. In the following, we denote by $()^{(k)}$ the k-th dimension of the vector.

Term ①:

$$\begin{aligned}\mathrm{Tr}\Big(\nabla_{x_i^l}(\kappa(x_j^l,x_j^l)\nabla_{x_j^l}s_p(x_j^l)^T)\Big) &= \mathrm{Tr}\Big(\nabla_{x_i^l}\kappa(x_j^l,x_j^l)(\nabla_{x_j^l}s_p(x_j^l))^\top+\kappa(x_j^l,x_j^l)\nabla_{x_i^l}\nabla_{x_j^l}s_p(x_j^l))\Big)\\ &= \sum_{t=1}^{d}\frac{\partial\kappa(x_j^l,x_j^l)}{\partial(x_i^l)^{(t)}}\frac{\partial s_p(x_j^l)}{\partial(x_i^l)^{(t)}}+0\\ &= (\nabla_{x_i^l}\kappa(x_j^l,x_j^l))^\top\nabla_{x_j^l}s_p(x_j^l)\\ &= -\frac{1}{2\sigma^2}\kappa(x_j^l,x_j^l)(x_i^l-x_j^l)^\top\nabla_{x_j^l}s_p(x_j^l)\end{aligned}$$

Term ②:

$$\begin{aligned}\mathrm{Tr}\Big(\nabla_{x_i^l}\nabla_{x_j^l}\kappa(x_i^l,x_j^l)\Big) &= \mathrm{Tr}\Big(\nabla_{x_i^l}\Big(\frac{1}{\sigma^2}\kappa(x_i^l,x_j^l)(x_i^l-x_j^l)\Big)\Big)\\ &= \frac{1}{\sigma^2}\sum_{k=1}^{d}\left(\frac{\partial\kappa(x_i^l,x_j^l)}{\partial(x_i^l)^{(k)}}(x_i^l-x_j^l)^{(k)}+\kappa(x_i^l,x_j^l)\right)\\ &= \frac{1}{\sigma^2}\Big(\nabla_{x_i^l}\kappa(x_i^l,x_j^l)^\top(x_i^l-x_j^l)+d\times\kappa(x_i^l,x_j^l)\Big)\\ &= \frac{1}{\sigma^2}\Big(\nabla_{x_i^l}\kappa(x_i^l,x_j^l)^\top(x_i^l-x_j^l)+d\times\kappa(x_i^l,x_j^l)\Big)\\ &= -\frac{1}{2\sigma^4}\times\kappa(x_i^l,x_j^l)\|x_i^l-x_j^l\|^2+\frac{1}{2\sigma^2}\times d\times\kappa(x_i^l,x_j^l)\\ &= \kappa(x_i^l,x_j^l)\left(-\frac{1}{2\sigma^4}\|x_i^l-x_j^l\|^2+\frac{d}{2\sigma^2}\right)\end{aligned}$$

Term ③: Using Hutchinson Trace Estimation Hutchinson [1989]

$$\mathrm{Tr}\Big(\nabla^2_{x_i^l}\log p(x_i^l)\Big) \approx \frac{1}{V}\sum_{t=1}^{V}v_t^T\nabla^2_{x_i^l}\log p(x_i^l)v_t$$

By combining **Terms** ①, ② and ③, we obtain:

$$\log q^L(x_i^L) = \log q^0(x_i^0) - \frac{\epsilon}{M\sigma^2}\sum_{l=0}^{L-1}\sum_{j=1}^{M}\kappa(x_j^l, x_j^l)\left(-(x_i^l - x_j^l)^\top \nabla_{x_j^l} s_p(x_j^l) - \frac{\alpha}{\sigma^2}\|x_i^l - x_j^l\|^2 + d\alpha\right)$$
$$-\frac{\epsilon}{MV}\sum_{t=1}^{V} v_t^T \nabla^2_{x_i^l}\log p(x_i^l) v_t$$

Proof done if we take a generic action particle x_i in place of x.

□

8.6 Computing $\mathrm{Tr}(\nabla_{x^l}\phi(x^l))$ with Bilinear kernel

[Liu et al., 2024] show that, for a Gaussian initial distribution, $q^0(x) = \mathcal{N}(\mu_0, \Sigma_0)$, and a target distribution $p(x) = \mathcal{N}(b, Q)$ such that $Q \in \mathbb{R}^{d\times d}$. Applying SVGD with a Bilinear kernel $\kappa(x_i, x_j) = \frac{x_j^T x_i}{C} + 1$ and explicitly showing $\log p(x^l) = -V(x^l) = -\frac{1}{2}(x^l - b)^T Q^{-1}(x^l - b)$ produces a Gaussian density q^l at every step with mean μ^l and covariance matrix Σ^l, satisfying the following system of equations:

$$\begin{cases} \mu^{l+1} = \mu^l + \epsilon^l\Big[\big(I - (\Sigma^l + \mu^l {\mu^l}^T)Q^{-1} + \mu^l b^T Q^{-1}\big)\frac{\mu^l}{C} + (b - \mu^l)^T Q^{-1}\Big], \\ \Sigma^{l+1} = \Sigma^l + \epsilon^l\Big[2\frac{\Sigma^l}{C} - \frac{\Sigma^l}{C}Q^{-1}(\Sigma^l + (\mu^l - b){\mu^l}^T) - (\Sigma^l + \mu^l(\mu^l - b)^T)Q^{-1}\frac{\Sigma^l}{C}\Big]. \end{cases} \tag{15}$$

We use this property to verify that the intermediate distributions q^l with the bilinear kernel are also Gaussian. We make use of the bilinear kernel's expression in the proof

Proof. For $\kappa(x_i^l, x_j^l) = \frac{{x_j^l}^T x_i^l}{C} + 1, \quad \nabla_{x_j^l}\kappa(x_i^l, x_j^l) = \frac{x_i^l}{C}, \quad$ and $\quad \nabla_{x_i^l}\nabla_{x_j^l}\kappa(x_i^l, x_j^l) = \frac{I}{C}$ we have the following SVGD dynamics at every step:

$$\left(x_i^{l+1} - x_i^l\right)/\epsilon^l = \frac{1}{M}\sum_{j=1}^{M}\nabla_{x_j}\kappa(x_i^l, x_j^l) + \frac{1}{M}\sum_{j=1}^{M}\kappa(x_i^l, x_j^l)\nabla_{x^l}\log p(x^l).$$

Hence, substituting these into the dynamics we obtain:

$$\begin{aligned}\left(x_i^{l+1} - x_i^l\right)/\epsilon^l &= \frac{1}{M}\sum_{j=1}^{M}\frac{x_i^l}{C} - \frac{1}{M}\sum_{j=1}^{M}\left(\frac{{x_j^l}^T x_i^l}{C} + 1\right)\nabla V(x_j^l) \\ &= \frac{x_i^l}{C} - \frac{1}{M}\sum_{j=1}^{M}Q^{-1}(x_j^l - b)\left(\frac{{x_j^l}^T x_i^l}{C} + 1\right) \\ &= \frac{x_i^l}{C} - \frac{1}{M}\sum_{j=1}^{M}Q^{-1}(x_j^l - b) - \frac{1}{M}\sum_{j=1}^{M}Q^{-1}(x_j^l - b)\frac{{x_j^l}^T}{C}x_i^l \\ &= \frac{x_i^l}{C} - \frac{Q^{-1}}{M}\sum_{j=1}^{M}(x_j^l - b) - \frac{Q^{-1}}{M}\sum_{j=1}^{M}x_j^l\frac{{x_j^l}^T}{C}x_i^l + \frac{Q^{-1}b}{M}\sum_{j=1}^{M}\frac{{x_j^l}^T}{C}x_i^l \\ &= \frac{x_i^l}{C} - Q^{-1}(\mu^l - b) - \frac{Q^{-1}}{C}\Big(\Sigma^l + \mu^l {\mu^l}^T\Big)x_i^l + \frac{Q^{-1}}{C}b{\mu^l}^T x_i^l \\ &= \left(\frac{I}{C} - \frac{Q^{-1}}{C}\Big(\Sigma^l + \mu^l {\mu^l}^T\Big) + \frac{Q^{-1}}{C}b{\mu^l}^T\right)x_i^l - Q^{-1}(\mu^l - b) \qquad \text{(i)}\end{aligned}$$

We substitute $\mu^l = \frac{1}{M}\sum_{j=1}^{M}x_j^l$ and that $\Sigma^l + \mu^l {\mu^l}^T = \frac{1}{M}\sum_{j=1}^{M}x_j^l x_j^{lT}, \quad$ and obtain

$$
\begin{aligned}
\frac{1}{M}\sum_{i=1}^{M}\left(x_i^{l+1}-x_i^l\right)/\epsilon^l &= \left(\frac{I}{C}-\frac{Q^{-1}}{C}\left(\Sigma^l+\mu^l{\mu^l}^T\right)+\frac{Q^{-1}}{C}b{\mu^l}^T\right)\mu^l - Q^{-1}(\mu^l-b)\\
&= \left(\frac{I}{C}-\frac{Q^{-1}}{C}\Sigma^l-\frac{Q^{-1}}{C}\left(\mu^l-b\right){\mu^l}^T\right)\mu^l - Q^{-1}(\mu^l-b)\\
&= \left(I-Q^{-1}\Sigma^l\right)\frac{\mu^l}{C}-\frac{Q^{-1}}{C}\left(\mu^l-b\right){\mu^l}^T\mu^l - Q^{-1}(\mu^l-b)\\
&= \left(I-Q^{-1}\Sigma^l\right)\frac{\mu^l}{C}-Q^{-1}\left(\mu^l-b\right)\left(\frac{{\mu^l}^T\mu^l}{C}+1\right)\\
\text{Hence } (\mu^{l+1}-\mu^l)/\epsilon^l &= \left(I-Q^{-1}\Sigma^l\right)\frac{\mu^l}{C}-Q^{-1}\left(\mu^l-b\right)\left(\frac{{\mu^l}^T\mu^l}{C}+1\right) \qquad \text{(ii)}
\end{aligned}
$$

Knowing that

$$
\begin{aligned}
\left(\Sigma^{l+1}-\Sigma^l\right)/\epsilon^l &= \frac{\partial \Sigma^l}{\partial l}\\
&= \frac{\partial}{\partial l}\frac{1}{M}\sum_{j=1}^{M} x_j^l x_j^{lT} - \mu^l{\mu^l}^T
\end{aligned}
$$

Taking into consideration (i) and (ii)

$$
\left(\Sigma^{l+1}-\Sigma^l\right)/\epsilon^l = 2\frac{\Sigma^l}{C}-\frac{\Sigma^l}{C}Q^{-1}\left(\Sigma^l+(\mu^l-b)\mu^l\right)-\left(\Sigma^l+\mu^l(\mu^l-b)^T\right)Q^{-1}\frac{\Sigma^l}{C} \qquad \text{(iii)}
$$

□

In Fig. 15, we empirically verify that SVGD intermediate distributions coincide with the derived Gaussians.

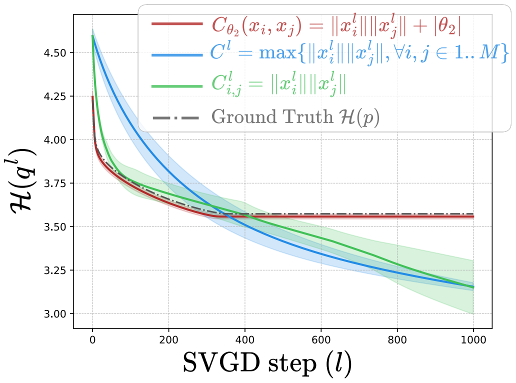


Figure 15: Bilinear kernel. q_θ^l coincides with theoretically derived intermediate distributions by Liu et al. [2024].

8.7 Computing $\mathrm{Tr}(\nabla_{x^l}\phi(x^l))$ with DKEF kernel

In the following, we show that using the deep exponential kernel (DKEF) $\kappa(x_i^l, x_j^l) = \exp\left(-\|\psi(x_i^l)-\psi(x_j^l)\|^2\right)$ where we denote by $\psi(x) \in \mathbb{R}^m$ an m dimensional vector, is computationally inefficient due to the requirement of computing $\nabla_x\psi(x)$ in our entropy derivation, *i.e.*, terms 1 and 2 in the density below:

$$\log q^L(a_i^L) = \log q^0(x_i^0) - \frac{\epsilon}{m}\sum_{l=0}^{L-1}\Bigg[\sum_{\substack{j=1\\ x_i^l\neq x_j^l}}^{m-1}\Big[\underbrace{\mathrm{Tr}\Big(\nabla_{x_i^l}(\kappa(x_i^l,x_j^l)\nabla_{x_j^l}\log p(x_j^l))\Big)}_{①} + \underbrace{\mathrm{Tr}\Big(\nabla_{x_i^l}\nabla_{x_j^l}\kappa(x_i^l,x_j^l)\Big)}_{②}\Big] + \underbrace{\mathrm{Tr}\Big(\nabla^2_{x_i^l}\log p(x_i^l)\Big)}_{③}\Bigg]$$

Proof. **Term ①:**

$$\begin{aligned}\mathrm{Tr}\left(\nabla_{x_i^l}(\kappa(x_j^l,x_j^l)\nabla_{x_j^l}\log p(x_j^l))\right) &= \mathrm{Tr}\left(\nabla_{x_i^l}\kappa(x_j^l,x_j^l)(\nabla_{x_j^l}\log p(x_j^l))^\top + \kappa(x_j^l,x_j^l)\nabla_{x_i^l}\nabla_{x_j^l}\log p(x_j^l))\right)\\ &= \sum_{t=1}^{d}\frac{\partial\kappa(x_j^l,x_j^l)}{\partial(x_i^l)^{(t)}}\frac{\partial\log p(x_j^l)}{\partial(x_i^l)^{(t)}} + 0\\ &= (\nabla_{x_i^l}\kappa(x_j^l,x_j^l))^\top\nabla_{x_j^l}\log p(x_j^l)\\ &= -\frac{1}{\sigma^2}\kappa(x_j^l,x_j^l)\nabla_{x_i}\psi(x_i)(\psi(x_i^l)-\psi(x_j^l))^\top\nabla_{x_j^l}\log p(x_j^l)\end{aligned}$$

Term ②:

$$\begin{aligned}\mathrm{Tr}\left(\nabla_{x_i^l}\nabla_{x_j^l}\kappa(x_i^l,x_j^l)\right) &= \mathrm{Tr}\left(\nabla_{x_i^l}\left(\frac{1}{\sigma^2}\kappa(x_i^l,x_j^l)(\psi(x_i^l)-\psi(x_j^l))\right)\right)\\ &= \frac{1}{\sigma^2}\mathrm{Tr}\left(\nabla_{x_i^l}\kappa(x_i^l,x_j^l)(\psi(x_i^l)-\psi(x_j^l))^\top + \kappa(x_i^l,x_j^l)\cdot I\right)\\ &= \frac{1}{\sigma^2}\mathrm{Tr}\left(-\frac{1}{\sigma^2}\kappa(x_i^l,x_j^l)\nabla_{x_i^l}\psi(x_i^l)(\psi(x_i^l)-\psi(x_j^l))(\psi(x_i^l)-\psi(x_j^l))^\top + \kappa(x_i^l,x_j^l)\cdot I\right)\end{aligned}$$

Term ③:

$$\mathrm{Tr}\Big(\nabla^2_{x_i^l}\log p(x_i^l)\Big) \approx \frac{1}{V}\sum_{t=1}^{V}v_t^T\nabla^2_{x_i^l}\log p(x_i^l)v_t$$

$\nabla_{x_i^l}\psi(x_i^l)\in\mathbb{R}^{m\times d}$ is required for both **Term ①** and **Term ②**, which introduces significant computational overhead and renders the density intractable. □

8.8 Derivation of the LD density

Similarly to the derivation above the LD induced density can be derived as:

Proof.

$$\begin{aligned}\log q^{l+1}(x^{l+1}) &= \log q^l(x^l) + \log\left|\det\nabla_{x^l}\phi(x^l)\right|^{-1} \quad \text{where } \phi(x^l) = x^l - \epsilon\nabla_{x^l}\log p(x^l) + \sqrt{2\epsilon}\xi\\ &\overset{(i)}{=} \log q^l(x^l) - \epsilon\mathrm{Tr}\Big(\nabla_{x^l}\phi(x^l)\Big), \quad \text{if } \lambda_i\{\nabla_{x^l}\phi(x^l)\} > -\frac{1}{\epsilon} \quad \forall i\\ &= \log q^l(x^l) - \epsilon\mathrm{Tr}\Big(\nabla^2_{x^l}\log p(x^l)\Big)\\ &\overset{(ii)}{=} \log q^l(x^l) - \frac{\epsilon}{V}\sum_{t=1}^{V}v_t^T\nabla^2_{x_i^l}\log p(x_i^l)v_t\end{aligned}$$

(i) Using CVF (Eq 6.2)

(ii) Using the Hutchinson Estimator, where p_v is chosen such that $\mathbb{E}[vv^T] = I$(eg. p_v is Radamacher distribution.)

In conclusion:

$$\log q^L(x^L) = \log q^0(x^0) - \frac{\epsilon}{V} \sum_{l=0}^{L} \sum_{t=1}^{V} v_t^T \nabla^2_{x_i^l} \log p(x_i^l) v_t$$

$\square$

9 Metropolis Hastings augmented entropy

In the following, we provide derivations for (1) the acceptance probability (Prop.3.5), (2) convergence check (Eq. 5) and (3) MH-augmented SVGD density (Eq. 5).

9.1 Acceptance Probability (Prop.3.5)

Proposition 3.4 Given a target $p = \bar{p}/Z$, the log-likelihood of the MH acceptance probability for an SVGD update of a particle x^{l-1} at step l is

$$\log \alpha(x^{l-1}, \tilde{x}^l) = \min\Big[0, \log \bar{p}(\tilde{x}^l) - \log \bar{p}(x^{l-1}) + \epsilon \mathrm{Tr}(\nabla_{x^l} \phi(x^l))\Big].$$

Proof. By leveraging Bayes' rule, we compute:

$$\begin{aligned}
\frac{q^l(x^{l-1} \mid \tilde{x}^l)}{q^l(\tilde{x}^l \mid x^{l-1})} &= \frac{q^l(x^{l-1}, \tilde{x}^l)}{q^l(\tilde{x}^l, x^{l-1})} \cdot \frac{q(x^{l-1})}{q(\tilde{x}^l)} \\
&= \frac{q(x^{l-1})}{q(\tilde{x}^l)} \\
&= \frac{q(x^{l-1})}{q(x^{l-1}) \,|\det(I + \epsilon \nabla_{x^{l-1}} \phi(x^{l-1}))|^{-1}} \\
&= |\det(I + \epsilon \nabla_{x^{l-1}} \phi(x^{l-1}))|
\end{aligned}$$

Thus, the Metropolis-Hastings ratio becomes:

$$\frac{p(x^l)}{p(x^{l-1})} \cdot \frac{q^l(x^{l-1} \mid \tilde{x}^l)}{q^l(\tilde{x}^l \mid x^{l-1})} = \frac{p(x^l)}{p(x^{l-1})} \cdot |\det(I + \epsilon \nabla_{x^{l-1}} \phi(x^{l-1}))|$$

Taking logs:

$$\log \left(\frac{p(x^l)}{p(x^{l-1})} \cdot \frac{q^l(x^{l-1} \mid \tilde{x}^l)}{q^l(\tilde{x}^l \mid x^{l-1})} \right) = \log \left(\frac{p(x^l)}{p(x^{l-1})} \right) + \log \left|\det(I + \epsilon \nabla_{x^{l-1}} \phi(x^{l-1}))\right|$$

Finally, using the first-order approximation $\log|\det(I + A)| \approx \mathrm{Tr}(A)$ for small ϵ, we obtain:

$$= \log \left(\frac{p(x^l)}{p(x^{l-1})} \right) + \epsilon \mathrm{Tr}(\nabla_{x^{l-1}} \phi(x^{l-1}))$$

□

9.2 Motivation: learning the SVGD learning rate (Sec. 3.2-Step-Sise)

Learning the kernel bandwidth alone is generally insufficient to ensure convergence of the entropy term. Specifically, the expectation $\mathbb{E}_{x^l \sim q^l}[\epsilon Tr(\nabla_{x^l} \phi(x^l))]$ does not necessarily vanish as $l \to \infty$. We show, via a Taylor expansion around 0, that this cumulative trace term corresponds to a 4th-degree polynomial in whose convergence to zero requires the existence of at least one real root. However, the coefficients of this polynomial depend on the particle positions and are not guaranteed to yield a real root during training, making this condition both non-trivial and fragile.

Proof.

$$\mathrm{Tr}\left(\nabla_{x^{L_c}}\phi(x^{L_c})\right) = \frac{\epsilon}{M\sigma^2}\sum_{\substack{j=1\\ x^{L_c}\neq x_j^{L_c}}}^{M}\kappa\left(x_j^{L_c}, x_j^{L_c}\right)\left((x^{L_c}-x_j^{L_c})^T\nabla_{x_j^{L_c}}\log p(x_j^{L_c}) + \frac{1}{\sigma^2}\|x^{L_c}-x_j^{L_c}\| - d\right)$$
$$+\frac{\epsilon}{M}\mathrm{Tr}\left(\nabla^2_{x_j^{L_c}}\log p(x_j^{L_c})\right)$$

We approximate the RBF kernel using a Taylor expansion:

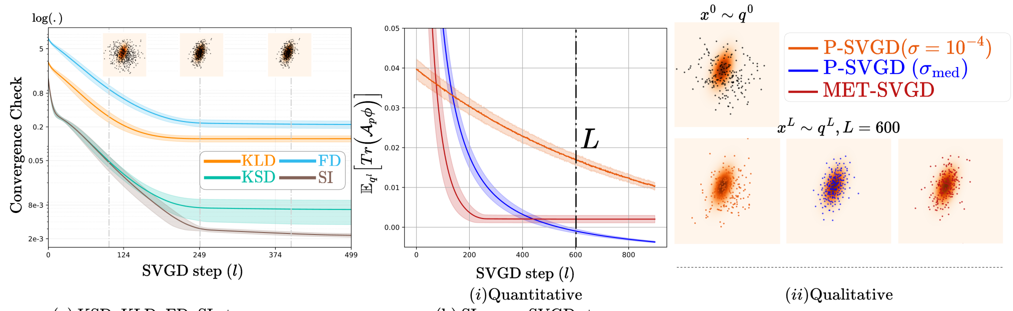
$$\exp\left(-\frac{\|x^{L_c}-x_j^{L_c}\|^2}{\sigma^2}\right)\approx 1-\frac{\|x^{L_c}-x_j^{L_c}\|^2}{\sigma^2}+\frac{\|x^{L_c}-x_j^{L_c}\|^4}{2\sigma^4}$$

We substitute in the formula above and obtain:

$$\mathrm{Tr}\left(\nabla_{x^{L_c}}\phi(x^{L_c})\right) = \frac{\epsilon}{M\sigma^2}\sum_{\substack{j=1\\ x^{L_c}\neq x_j^{L_c}}}^{M}\left(1-\frac{\|x^{L_c}-x_j^{L_c}\|^2}{\sigma^2}+\frac{\|x^{L_c}-x_j^{L_c}\|^4}{2\sigma^4}\right)\times\left((x^{L_c}-x_j^{L_c})^T\nabla_{x_j^{L_c}}\log p(x_j^{L_c})\right.$$
$$\left.+\frac{1}{\sigma^2}\|x^{L_c}-x_j^{L_c}\|-d+\frac{\epsilon}{M}\mathrm{Tr}\left(\nabla^2_{x_j^{L_c}}\log p(x_j^{L_c})\right)\right.$$
$$=\frac{\epsilon\sigma^8}{M\sigma^2}\sum_{\substack{j=1\\ x^{L_c}\neq x_j^{L_c}}}^{M}\left(\sigma^4-\sigma^2\|x^{L_c}-x_j^{L_c}\|^2+\frac{\|x^{L_c}-x_j^{L_c}\|^4}{2}\right)\times\left(\sigma^2(x^{L_c}-x_j^{L_c})^T\nabla_{x_j^{L_c}}\log p(x_j^{L_c})\right.$$
$$\left.+\|x^{L_c}-x_j^{L_c}\|-d\sigma^2+\frac{\epsilon\sigma^8}{M}\mathrm{Tr}\left(\nabla^2_{x_j^{L_c}}\log p(x_j^{L_c})\right)\right)$$

Since we have a polynomial of degree 8, we have 8 roots, not all of which are guaranteed to be real for σ. In fact, we need the number of real roots to be computed as the signature of the Hermitian matrix. ie the number of real roots is equal to the number of positive values. In Fig. 16, we show that SI converges to 0 under different sampler configs of the experiments.

9.3 Convergence Check (Eq. 5)



(a) KSD, KLD, FD, SI at convergence (b) SI across SVGD steps

Figure 16: (a) SI shows the same convergence trend as other convergence metrics such as Fisher Divergence (FD) and Kernelized Stein Discrepancy (KSD). With SI being the tractable and computationally efficient metric. (b) SI can be used to check SVGD convergence across steps, for **MET-SVGD** and **P-SVGD**(σ_{med}, $\sigma=10^{-4}$).

9.3.1 Derivation of the Stein Identity

The following is a proof of proposition 3.4:

Proof. The Stein Identity is the square of the Kernelized Stein Discrepancy [Liu et al., 2016]:

$$\mathbb{S}(q^l, p) = \max_{\phi \in \mathcal{H}^d} \Big[\mathbb{E}_{x^l}\left[\mathrm{Tr}(\mathcal{A}_p \phi(x^l))\right]^2, \quad \text{s.t } \|\phi\|_{\mathcal{H}^d} \leq 1\Big]$$

The optimal perturbation ϕ is given by:

$$\phi(x^l) = \frac{\phi^*_{q,p}(x^l)}{\|\phi^*_{q,p}\|_{\mathcal{H}^d}}, \quad \text{with} \quad \phi^*_{q,p}(\cdot) = \mathbb{E}_{x^l}\left[\mathcal{A}_p k(x^l, \cdot)\right] \quad \text{and} \quad \mathbb{S}(q^l, p) = \|\phi^*_{q,p}\|_{\mathcal{H}^d}$$

We compute:

$$\begin{aligned}\|\phi^*_{q,p}\|^2_{\mathcal{H}^d} &= \langle \phi^*_{q,p}, \phi^*_{q,p}\rangle_{\mathcal{H}_d} \\ &= \mathbb{E}_{x_i^l}\mathbb{E}_{x_j^l}\Big[\big(k(x_i^l, x_j^l)\nabla_{x_j^l}\log p(x_j^l) + \nabla_{x_j^l} k(x_i^l, x_j^l)\big)\cdot\big(k(x_j^l, x_i^l)\nabla_{x_i^l}\log p(x_i^l) + \nabla_{x_i^l} k(x_j^l, x_i^l)\big)\Big]\end{aligned}$$

And obtain

$$\mathbb{S}(q^l, p) = \mathbb{E}_{x^l\sim q^l}\left[\mathrm{Tr}\left(\frac{\phi^*(x^l)}{\|\phi^*\|}\nabla_{x^l}\log p(x^l)^T + \nabla_{x^l}\frac{\phi^*(x^l)}{\|\phi^*\|}\right)\right] = \mathbb{E}_{x^l\sim q^l}\left[\frac{\phi(x^l)^T\nabla_{x^l}\log p(x^l) + \mathrm{Tr}(\nabla_{x^l}\phi(x^l))}{\|\phi^*\|}\right]$$

□

9.3.2 Intractability of KSD, FD

Even though Fisher Divergence $\mathbb{F}(q^l, p)$ and Kernelized Stein Discrepancy $\mathbb{S}(q^l, p)$ follow the same trend with the Stein Identity, they cannot be used as convergence metrics as they require computing $\nabla_{x^l}\log q^l(x^l)$ which will require computing a jacobian on $\nabla_{x^{l-1}}\phi(x^{l-1})$.

Proof. Note that $\forall x, x'$ being i.i.d. draws from q^l, $\mathbb{F}(q^l, p) = \mathbb{E}_{x^l}[\nabla_x \log q^l(x) - \nabla_x \log p(x)]$ and $\mathbb{S}(q^l, p) = \mathbb{E}_{x,x'\sim q^l}\Big[\big(\nabla_x \log q^l(x) - \nabla_x \log p(x)\big)^T k(x, x')\big(\nabla_{x'}\log q^l(x') - \nabla_{x'}\log p(x')\big)\Big]$. Computing either is possible thanks to the closed form expression of the log density 14, which circumvents the intractability issue encountered due to the score of $q^l, \forall l \in [0 \cdots L[$:

$$\nabla_{x^l}\log q^l(x^l) = \frac{\partial \log q^l(x^l)}{\partial x^l} = \left(\frac{\partial x^l}{\partial x^{l-1}}\right)^{-1}\cdot\frac{\partial \log q^l(x^l)}{\partial x^{l-1}} = \Big(\nabla_{x^{l-1}}\phi(x^{l-1})\Big)^{-1}\cdot\nabla_{x^{l-1}}\log q^l(x^l)$$

$$\text{Such that} \quad \log q^l(x^l) = \log q^{l-1}(x^{l-1}) - \epsilon \log\Big|\det \nabla_{x^{l-1}}\phi(x^{l-1})\Big|$$

$$\text{So} \quad \nabla_{x^l}\log q^l(x^l) = \underbrace{\Big(\nabla_{x^{l-1}}\phi(x^{l-1})\Big)^{-1}}_{\text{Intractable}}\cdot\Big(\nabla_{x^{l-1}}\log q^{l-1}(x^{l-1}) - \epsilon\nabla_{x^{l-1}}\mathrm{Tr}(\nabla_{x^{l-1}}\phi(x^{l-1}))\Big)$$

□

9.4 MH-augmented SVGD Density (Eq 5)

Proposition 9.1. *The **MH-augmented density** over particles after incorporating MH correction is:*

$$q_\theta^{MH,l}(x^l) = \alpha^l_{\theta_{2,3}} q_\theta^{MH,l-1}(x^{l-1})|\det\nabla_{x^l}\phi_{\theta_2}(x^l)|^{-1} + (1-\alpha^l_{\theta_{2,3}})q_\theta^{MH,l-1}(x^{l-1}), \quad \textit{with } q_{\theta_1}^{MH,0} = q^0_{\theta_1}$$

Proof. We prove the statement above by induction. We leverage:

$$q_\theta^{\mathrm{MH},l} = \alpha^{l-1}_{\theta_{2,3}} q_\theta^{\mathrm{MH},l-1}(x^{l-1})\Big|\det(I + \epsilon\nabla_{x^{l-1}}\phi(x^{l-1}))\Big|^{-1} + (1-\alpha^{l-1}_{\theta_{2,3}})q_\theta^{\mathrm{MH},l-1} \qquad (16)$$

$$\begin{cases} \text{Case } l=0: \quad q_\theta^{\text{MH},0} = q_\theta^0 \\ \\ \text{Case } l=1: \quad q_\theta^{\text{MH},1}(x^1) = \alpha^0_{\theta_{2,3}} q_\theta^0(x^0) \Big| \det(I + \epsilon \nabla_{x^0} \phi(x^0)) \Big| + (1-\alpha^0_{\theta_{2,3}}) q_\theta^0(x^0) \end{cases} \quad \text{,with } a_{1;L} = a_1$$

On the other hand, evaluating $q_\theta^{\text{MH},1}(x^1, a_1)$ by marginalizing over $a_1 \in \{0,1\}$:

$$\begin{cases} q_\theta^{\text{MH},1}(x^1, a_1 = 0) = q_\theta^0(x_0)(1-\alpha^1_{\theta_{2,3}}) \\ \\ q_\theta^{\text{MH},1}(x^1, a_1 = 1) = q_\theta^0(x_0)(\alpha^1_{\theta_{2,3}}) \Big| \det(I + \epsilon \nabla_{x_0} \phi(x_0)) \Big| \end{cases}$$

Therefore:

$$q_\theta^{\text{MH},1}(x^1) = q_\theta^0(x_0)\alpha^1_{\theta_{2,3}} \Big| \det(I + \epsilon \nabla_{x_0} \phi(x_0)) \Big| + q_\theta^0(x_0)(1-\alpha^1_{\theta_{2,3}})$$

Case $l > 1$: We assume

$$q_\theta^{\text{MH},l}(x^l) = q_\theta^l(x^l) \prod_{l=1}^{l} \alpha^l_{\theta_{2,3}} + \sum_{a_{1:l} \neq 1} q_\theta^{\text{MH},l}(x^l, a_{1:l}), \quad \forall l \in]1, L_c] \tag{17}$$

and show that it holds for $q_\theta^{\text{MH},l+1}$.

We know that:

$$q_\theta^{\text{MH},L_c+1}(x^{L_c+1}) = \alpha^{L_c+1}_{\theta_{2,3}} q_\theta^{\text{MH},L_c}(x^{L_c}) \Big| \det(I + \epsilon \nabla_{x^{L_c}} \phi(x^{L_c})) \Big| + (1-\alpha^{L_c}) q_\theta^{\text{MH},L_c}(x^{L_c}) \tag{18}$$

We plug Eq. 17 into Eq. 18:

$$\begin{aligned} q_\theta^{\text{MH},L_c+1}(x^{L_c+1}) &= q_\theta^{\text{MH},L_c}(x^{L_c}) \times \Big[\alpha^{L_c+1}_{\theta_{2,3}} \Big| \det(I + \epsilon \nabla_{x^{L_c}} \phi(x^{L_c})) \Big| + (1-\alpha^{L_c})\Big] \\ &= \left[q_\theta^{L_c}(x^{L_c}) \prod_{l=1}^{L_c} \alpha^l_{\theta_{2,3}} + \sum_{a_{1..L_c} \neq 1} q_\theta^{\text{MH},L_c}(x^{L_c}, a_{1..L_c}) \right] \times \Big[\alpha^{L_c+1}_{\theta_{2,3}} \Big| \det(I + \epsilon \nabla_{x^{L_c}} \phi(x^{L_c})) \Big| + (1-\alpha^{L_c})\Big] \\ &= \underbrace{q_\theta^{L_c}(x^{L_c}) \prod_{l=1}^{L_c+1} \alpha^l_{\theta_{2,3}} \Big| \det(I + \epsilon \nabla_{x^{L_c}} \phi(x^{L_c})) \Big|}_{q_\theta^{L_c+1}(x^{L_c+1}) \prod_{l=1}^{L_c+1} \alpha^l_{\theta_{2,3}}} + \underbrace{q_\theta^{L_c}(x^{L_c}) \prod_{l=1}^{L_c} \alpha^l_{\theta_{2,3}} (1-\alpha^{L_c})}_{q_\theta^{\text{MH},L_c+1}(x^{L_c+1}, a_{1..L_c}=1, a^{L_c+1}=1)} \\ &\quad + \underbrace{\sum_{a_{1..L_c} \neq 1} q_\theta^{\text{MH},L_c}(x^{L_c}, a_{1..L_c}) \alpha^{L_c+1}_{\theta_{2,3}} \Big| \det(I + \epsilon \nabla_{x^{L_c}} \phi(x^{L_c})) \Big|}_{q_\theta^{\text{MH},L_c+1}(x^{L_c+1}, a_{1..L_c} \neq 0, a^{L_c+1}=1)} + \underbrace{\sum_{a_{1..L_c} \neq 1} q_\theta^{\text{MH},L_c}(x^{L_c}, a_{1..L_c})(1-\alpha^{L_c})}_{q_\theta^{\text{MH},L_c+1}(x^{L_c+1}, a_{1..L_c} \neq 0, a^{L_c+1}=0)} \end{aligned}$$

□

10 Additional Results on Entropy Estimation

In the following, we provide implementation details and additional experiments for the toy experiments.

10.1 Gaussian Targets

Implementation Details for Fig. 2A are reported in Tab. 4.

Algorithm	**Parameter**	**Value**
LD P-SVGD MET-SVGD	Target p Number of Particles M Number of Steps L Initial Distribution q^0	$p = \mathcal{N}([-0.69, 0.8], [[1.13, 0.82], [0.82, 3.39]])$ $M = 200$ $L = 1500$ $\mathcal{N}(0, 6I)$
LD P-SVGD	Learning Rate ϵ	$\epsilon = 0.1$
P-SVGD	Kernel Bandwidth σ	$\sigma \in \{1, 5, \sigma_{\text{med}}\}$
	Kernel Architecture	
	Architecture	$\sigma_{\theta_2} = \text{GNN}(\{x_i^l\}_{i=1}^M; \theta_2)$
	# Layers	3
	Activation	{ReLU, Exponential, Truncate}
MET-SVGD	**Learning Rate Architecture ??**	
	Architecture	$\epsilon_{\theta_3}^l = \min\left(\epsilon_{\theta_3}^0, \epsilon_{\theta_3}^0 d^{l/s_{\theta_3}}\right)$
	Initial Learning Rate $\epsilon_{\theta_3}^0$	0.1
	Decay Factor d_{θ_3}	5×10^{-3}
	Decay Scale s_{θ_3}	L
	Training Parameters	
	Optimizer	Adam
	Learning Rate	$5 \cdot 10^{-3}$
	Epochs	300
	Loss	{KL Divergence}
Resources	GPU RAM Per-epoch runtime	Tesla V100-SXM2-32GB 2 GB 2.6 seconds

Table 4: Experimental setup for Fig. 2A

Implementation Details (Fig. 15) & (Fig. 16)

Parameter	**Figure 15**	**Figure 16**
Target p	$p = \mathcal{N}([-0.69, 0.8], [[1.13, 0.82], [0.82, 3.39]])$	
Number of Particles M	$M = 500$	
Number of Steps L	$L = 1000$	$L = 500$
Initial Distribution q^0	$\mathcal{N}(0, 6I)$	
Kernel Architecture		
Architecture	$C_{\theta_2} = \text{GNN}(\{x_i^l\}_{i=1}^M; \theta_2)$	$\sigma_{\theta_2} = \text{GNN}(\{x_i^l\}_{i=1}^M; \theta_2)$
Kernel Type	Bilinear: $\frac{x_i^T x_j}{C_{\theta_2}^2} + 1$	RBF: $\exp(-\|x_i - x_j\|^2 / 2\sigma^2)$
Learning Rate Architecture ??		
Architecture	$\epsilon_{\theta_3}^l = \min\left(\epsilon_{\theta_3}^0, \epsilon_{\theta_3}^0 d^{l/s_{\theta_3}}\right)$	
Initial Learning Rate $\epsilon_{\theta_3}^0$	$\epsilon_{\theta_3}^0 = 0.1$	
Decay Factor d_{θ_3}	$d_{\theta_3} = 5 \times 10^{-3}$	
Decay Scale s_{θ_3}	$s_{\theta_3} = L$	
Training Parameters		
Optimizer	Adam	
Learning Rate	$5 \cdot 10^{-3}$	
Epochs	300	

Table 5: Experimental setup for Figs. 15 and 16.

Implementation Details (Fig. 5) **Langevin Dynamics.** In Fig. 5, we show that we can learn the Langevin dynamics' parameters $\theta = \{\epsilon_{\theta_2}, \mu_{\theta_3}\}$ such that $x^{l+1} = x^l + \epsilon_{\theta_2} \nabla_{x^l} \log p(x^l) + \sqrt{2\epsilon_{\theta_2}} \xi_{\theta_3}$ end-to-end by minimizing the reverse KL-Divergence:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{x^L \sim q_\theta^L} [\log q_\theta^L(x^L) - \log p(x^L)]$$

subject to $\epsilon_{\theta_3}^l \leq \epsilon_{\text{UB}}^l$ for all $l \in [0, L-1]$.

Here $q^L = q^0 + \epsilon \sum_{l=0}^{L-1} \text{Tr}(\nabla_{x^l}^2 p(x^l))$, where the trace is approximated via the Hutchinson estimator (Eq. 8.5). We show SVGD is less sensitive to this approximation than LD in high dimensions.

Parameter	**LD**	**MET-SVGD**
Number of particles	$M = 100$	
Number of iterations	$L = 1000$	
Target distribution	$p = \mathcal{N}(0, I_d), \quad d \in \{2, 10, 50, 80, 100\}$	
Initial distribution	$\mathcal{N}(0, 6I_d)$ (augmented)	
Algorithm-Specific Learned Parameters		
Learned parameter	Gaussian noise	Kernel variance σ
Learned LR	$\epsilon_{\theta_3}^l = \min\left(\epsilon_{\theta_3}^0, \epsilon_{\theta_3}^0 d^{l/s_{\theta_3}}\right)$	
Training Parameters		
Optimizer	Adam	
Learning rate	$5 \cdot 10^{-3}$	
Epochs	300	

Table 6: Experimental setup for Fig. 5

Scalability. (a) Multivariate Gaussian. In Fig. 17, we visualize the learnt SVGD learning rate for the setup in Fig. 8 in the main paper. The target is a d-dimensional multivariate Gaussian $p(x) = \mathcal{N}(x; 0, I_d)$. For each method, 100 particles are initialized from $\mathcal{N}(x; 2\mathbb{1}, 2I_d)$, where $\mathbb{1} \in \mathbb{R}^d$ denotes the vector of ones. This is a standard benchmark illustrating the diminishing variance issue of SVGD. We show that MET-SVGD outperforms other baselines in high dimensional spaces as measured by the entropy and the variance across dimensions (Fig. 17-a,b). The SVGD repulsive force is large during the first updates, preventing the particles from collapsing, unlike other baselines (Fig. 17-c). P-SVGD is not scalable due to a missing term in the entropy (see Sec. 8.3). This is subsequently fixed in the MET-SVGD update.

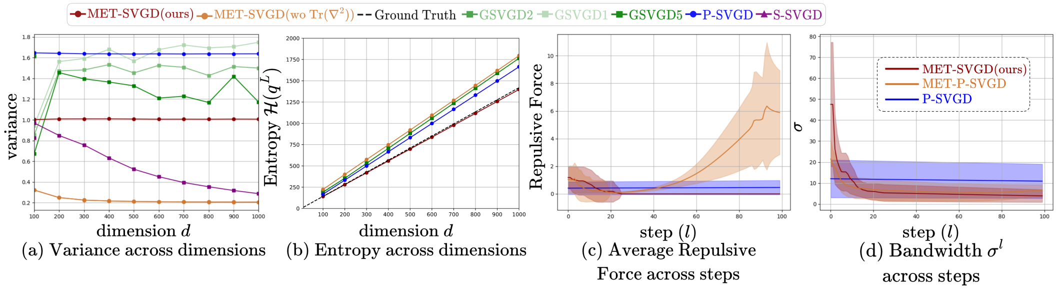


Figure 17: Scalability Results: **MET-SVGD** achieves higher accuracy on both (a) Entropy Estimation and (b) Variance across dimensions

Accelerating Convergence. In Fig. 18, we show that, for the setup of Fig. 16, convergence can be accelerated either by (1) adding a regularization on the decay rate in the optimization objective (see Sec.3.2) or (2) randomizing the maximum number of steps during training (Eq. 5). We observe a quicker drop in the kernel variance (particles are less correlated initially).

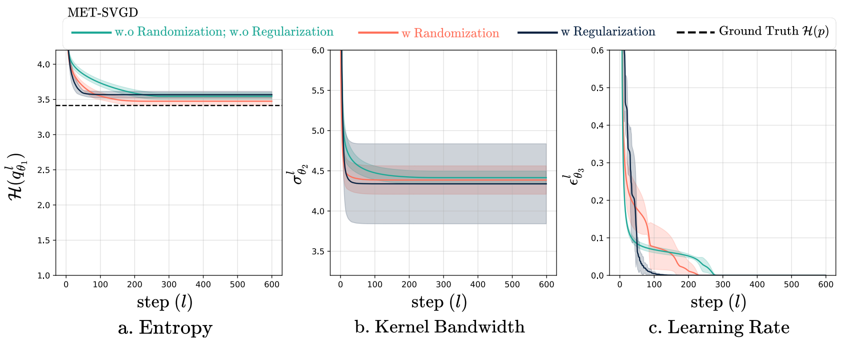


Figure 18: Accelerated convergence via regularization and number of SVGD steps randomization for the same MET-SVGD setup in Tab.6

10.2 Gaussian Mixture Model

10.2.1 Experiment on 2D GMM with moving component

In the following, we provide implementation details and additional experiments for the toy experiments where the target is a 2D GMM.

Implementation Details for (Fig. 7) are reported in Tab.7.

Parameter	**P-SVGD**	**MET-SVGD**
Distribution type	GMM with 5 components	
Fixed components	$\mu_1 = (0.0, 0.0), \Sigma_1 = 0.16 I_2$	
	$\mu_2 = (3.0, 2.0), \Sigma_2 = I_2$	
	$\mu_3 = (1.0, -0.5), \Sigma_3 = 0.5 I_2$	
	$\mu_4 = (2.5, 1.5), \Sigma_4 = 0.5 I_2$	
Mobile component	$\mu_5 = (c, c), \Sigma_5 = 0.5 I_2$ where $c \in [-3, 3]$	
Dimension	$d = 2$	
Number of particles	$M \in \{50, 100, 500\}$	
Number of iterations	$L = 1500$	
Initial Distribution Settings		
Setting 1	$\mathcal{N}(0, I_2)$	
Setting 2	GMM$(K = 10)$ with $\mu_k \overset{\text{i.i.d.}}{\sim} \mathcal{U}([-4, 4]^2)$ and $\Sigma_k = I_2 \; \forall k$	
Algorithm-Specific Parameters		
Kernel variance	$\sigma \in \{1, 5, \text{median}\}$	$\sigma = \text{GNN}(\{x_i^l\}_{i=1}^M; \theta_2)$
Learning rate	$\epsilon = 0.1$	$\epsilon_{\theta_3}^l = \min\left(\epsilon_{\theta_3}^0, \epsilon_{\theta_3}^0 d^{l/s_{\theta_3}}\right)$
Base learning rate	-	$\epsilon_{\theta_3}^0 = 0.1$
Decay factor	-	$d_{\theta_3} = 5 \times 10^{-3}$
Decay scale	-	$s_{\theta_3} = L$
Training Parameters		
Optimizer	Adam	
Learning rate	5.10^{-3}	
Epochs	300	

Table 7: Experimental setup for Fig. 7

Qualitative results for different c values, as well as the KL-divergence, entropy, kernel Bandwidth and step-size are reported in Fig. 19. We observe that **P-SVGD** with σ_{med} has poor convergence. In facr, σ_{θ_2} shows a different trend entirely. Whereas ϵ_{θ_3} converges consistently across all configurations.

Additionally, we show in Fig. 20 that the number of particles is key to an accurate, low variance estimation. Adding more particles significantly helps improve the accuracy. **P-SVGD** performs poorly. In fact, the estimation is worse than a trivial lower bound that we derive as follows:

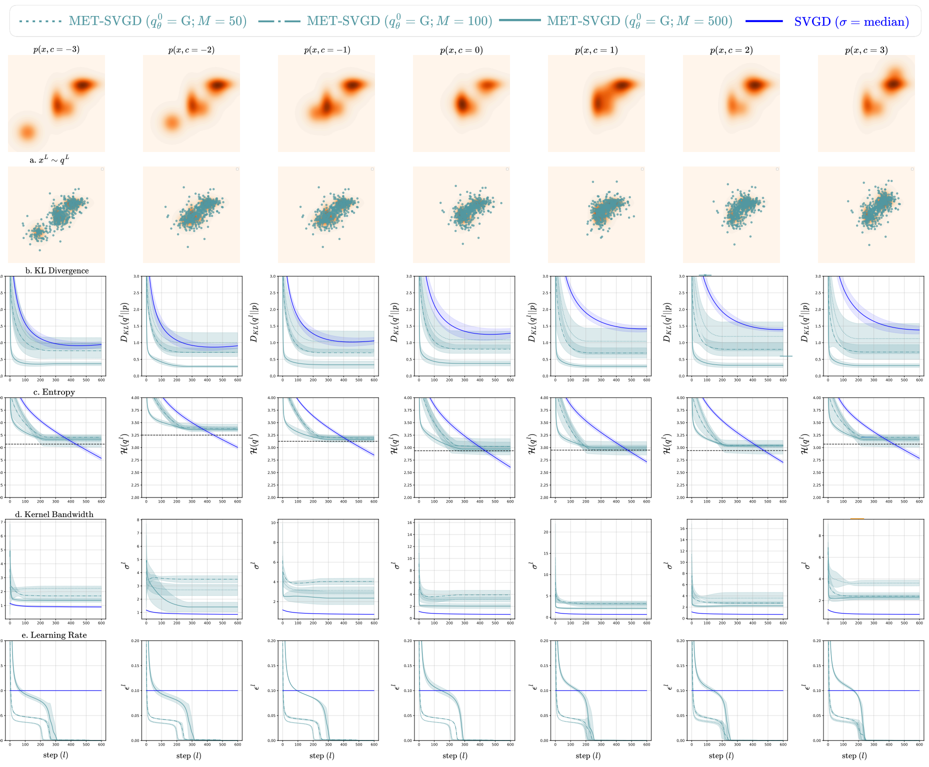


Figure 19: Results on entropy estimation for different c configurations. (a) KLD, (b) Entropy, (c) Learnt kernel bandwidth and (d) SVGD step-size.

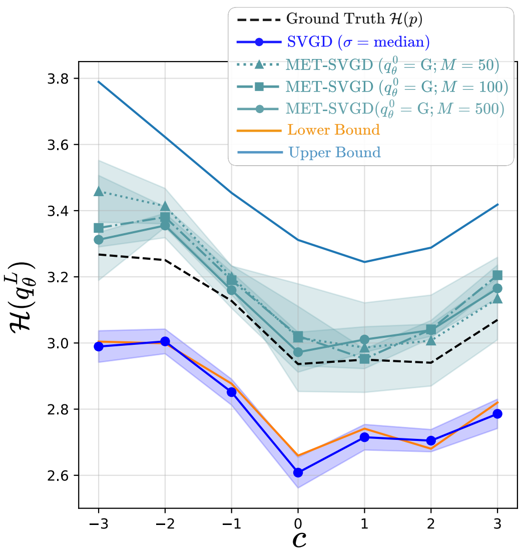


Figure 20: Entropy results on a 2D GMM with a moving component(Fig. 19). **MET-SVGD** significantly outperforms **P-SVGD**. Increasing the number of particles reduces the variances. Results are reported on 5 different seeds.

Next we explain the derivation of the Upper & Lower Bounds for a GMM target as seen in Fig. 20 as follows:

The pdf of A GMM with K components is given by the formula:

$$p(x) = \sum_{i=1}^{K} \omega_i \mathcal{N}(x; \mu_i, C_i),$$

where $\{\omega_i\}_{i=1}^{K}$ are non-negative weighting coefficients such that $\sum_i \omega_i = 1$ and $\mathcal{N}(x; \mu_i, C_i)$ is a gaussian density with mean μ_i and covariance C_i. Note that the entropy generally cannot be calculated in closed form for GMM due to the logarithm of a sum of exponential functions (except for the special case of a single Gaussian density). We derive two trivial bounds to assess the performance of the different baselines. We start by deriving the lower bound $\text{LB}(x) = -\sum_{i=1}^{K} \omega_i \log(\sum_{j=1}^{K} \omega_j \mathcal{N}(\mu_i; \mu_j, C_i + C_j))$.

Proof.

$$\begin{aligned}
\mathcal{H}(x) &= -\sum_{i=1}^{K} \omega_i \int_{\mathbb{R}^N} \mathcal{N}(x; \mu_i, C_i) \log p(x) dx \\
&\geq -\sum_{i=1}^{K} \omega_i \log \Big[\int_{\mathbb{R}^N} \mathcal{N}(x; \mu_i, C_i) p(x) dx \Big] \quad \text{by Jensen inequality} \\
&\geq -\sum_{i=1}^{K} \omega_i \log \Big[\int_{\mathbb{R}^N} \mathcal{N}(x; \mu_i, C_i) \sum_{j=1}^{K} \omega_j \mathcal{N}(x; \mu_j, C_j) dx \Big] \geq \underbrace{-\sum_{i=1}^{K} \omega_i \log \Big[\sum_{j=1}^{K} \omega_j \mathcal{N}(\mu_i; \mu_j, C_i + C_j) \Big]}_{\text{LB}(x)}
\end{aligned}$$

Upper-bound UB: We prove that $\mathcal{H}(p(x)) \leq \text{UB}$ by deriving the entropy of the Gaussian enveloping the target:

$$\text{UB} = \mathcal{H}(\mathcal{N}(\mu_\text{G}, C_\text{G})), \quad \text{with} \begin{cases} \mu_\text{G} = \sum_{i=1}^{L} \omega_i \mu_i \\ C_\text{G} = \sum_{i=1}^{L} \omega_i (C_i + \mu_i \mu_i^T) - \sum_{i=1}^{L} \sum_{j=1}^{L} \omega_i \omega_j \mu_i \mu_j^T \end{cases}$$

Proof. Consider a Gaussian mixture model $p(x) = \sum_{i=1}^{L} \omega_i \mathcal{N}(x; \mu_i, C_i)$. The mean is $\mu = \mathbb{E}[p(x)] = \sum_{i=1}^{L} \omega_i \mu_i$. The covariance can be computed as:

$$\begin{aligned}
C = \mathbb{E}[(x-\mu)(x-\mu)^T] = \mathbb{E}[xx^T] - \mu\mu^T &= \int_x \left(\sum_{i=1}^{L} \omega_i \mathcal{N}(x; \mu_i, C_i) \right) xx^T dx - \mu\mu^T \\
&= \sum_{i=1}^{L} \omega_i \int_x xx^T \mathcal{N}(x; \mu_i, C_i) dx - \mu\mu^T
\end{aligned}$$

For a single Gaussian component, $C_i = \mathbb{E}[(x-\mu_i)(x-\mu_i)^T] = \mathbb{E}[xx^T] - \mu_i\mu_i^T$, which means $\mathbb{E}_{x \sim \mathcal{N}(x;\mu_i,C_i)}[xx^T] = C_i + \mu_i\mu_i^T$. Substituting this in the above:

$$\begin{aligned}
C &= \sum_{i=1}^{L} \omega_i (C_i + \mu_i \mu_i^T) - \mu\mu^T \\
&= \sum_{i=1}^{L} \omega_i (C_i + \mu_i \mu_i^T) - \left(\sum_{i=1}^{L} \omega_i \mu_i \right) \left(\sum_{j=1}^{L} \omega_j \mu_j^T \right) = \sum_{i=1}^{L} \omega_i (C_i + \mu_i \mu_i^T) - \sum_{i,j=1}^{L} \omega_i \omega_j \mu_i \mu_j^T = C_G
\end{aligned}$$

Since a Gaussian distribution maximizes entropy among all distributions with the same mean and covariance, we have $\mathcal{H}(p(x)) \leq \mathcal{H}(\mathcal{N}(\mu_\text{G}, C_\text{G})) = \text{UB}(x)$.

10.2.2 Targets with distant modes

Implementation Details (Fig. 21) are reported in Tab.8. We show that the divergence control heuristic based on eliminating particles further than 3 standard deviations of the initial distribution mean exacerbates mode collapse is already an issue when using the reverse KL-divergence.

Qualitative results particles from the initial distribution are visualized in (i) a. We apply the truncation heuristic to different setups (**P-SVGD** with 0 steps, ie with only a learnable initial distribution and **P-SVGD** with $L = 140$ steps).

Algorithm	**Parameter**	**Value**
No SVGD P-SVGD MET-SVGD	Target p	GMM with 3 components $\mu_1 = (-4.0, 2.0), \Sigma_1 = I_2$ $\mu_2 = (4.0, 2.0), \Sigma_2 = I_2$ $\mu_3 = (0.0, -12.0), \Sigma_3 = 2I_2$
	Number of Particles M	$M = 200$
	Number of Steps L	$L = 1000$
	Initial Distribution q^0	$\mathcal{N}(0, 6I)$
P-SVGD	Learning Rate ϵ	$\epsilon = 0.1$
	Kernel Bandwidth σ	$\sigma \in \{1, 5, \text{median}\}$
MET-SVGD	**Kernel Architecture**	
	Architecture	$\sigma_{\theta_2} = \text{GNN}(\{x_i^l\}_{i=1}^M; \theta_2)$
	Learning Rate Architecture	
	Architecture	$\epsilon_{\theta_3}^l = \min\left(\epsilon_{\theta_3}^0, \epsilon_{\theta_3}^0 d^{l/s_{\theta_3}}\right)$
	Initial Learning Rate $\epsilon_{\theta_3}^0$	$\epsilon_{\theta_3}^0 = 0.1$
	Decay Factor d_{θ_3}	$d_{\theta_3} = 5 \times 10^{-3}$
	Decay Scale s_{θ_3}	$s_{\theta_3} = L$
	Training Parameters	
	Optimizer	Adam
	Learning Rate	$5 \cdot 10^{-3}$
	Epochs	300
	Activation Functions	{Exponential(), Truncate(min, max)}

Table 8: Experimental setup for Fig. 21

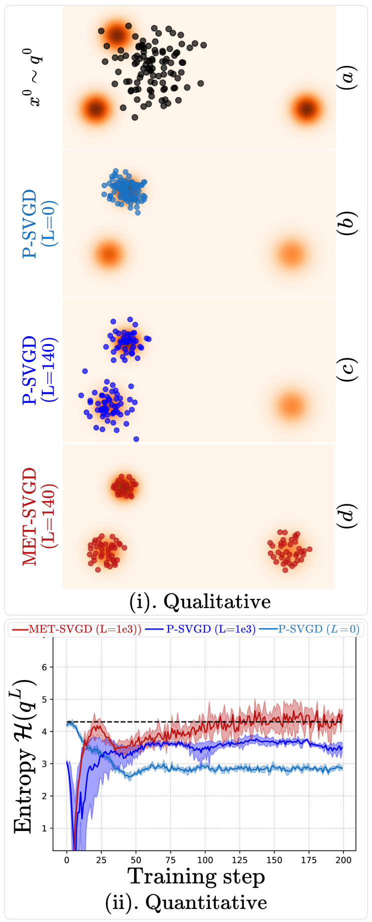
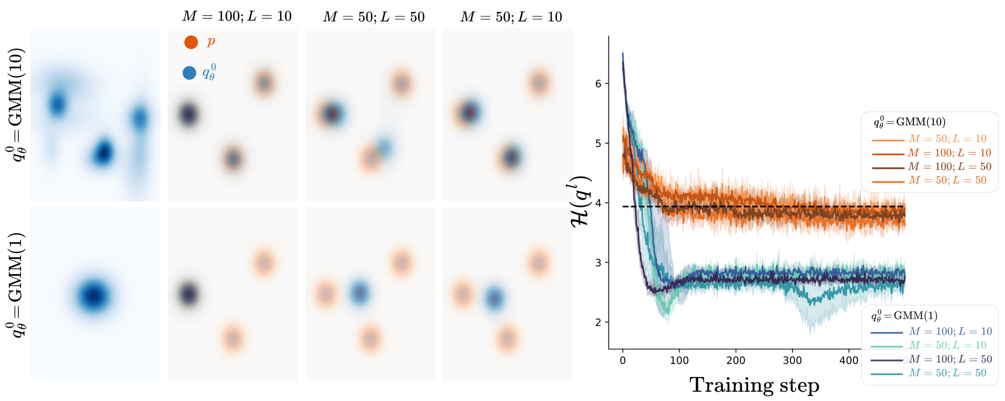


Figure 21: P-SVGD struggles with distributions with distant modes.

Effect of the Initial Distribution. In Fig. 22, we compare the results of using a Gaussian and a GMM with 10 components. We show that when using a GMM, less steps and particles are needed to learn the target. All experiments with a Gaussian initial distribution resulted in mode collapse. Using a GMM mitigates the mode collapse issue when learning the parameters using the reverse KL-divergence. In the future, we will explore training with the forward KLD while leveraging importance sampling.

Implementation details are in Tab.9.



a. Learnt initial Distribution

b. Entropy across training steps

Figure 22: Effect of the initial distribution on a GMM setup.

Algorithm	Parameter	Value
MET-SVGD	Target p	GMM with 3 components (orange spots in figure)
	Number of Particles M	$M \in \{50, 100\}$
	Number of Steps L	$L \in \{10, 50\}$

	Initial Distribution q_θ^0	Two settings:
		GMM(1): Single component (left column)
		GMM(10): 10 components (right column)

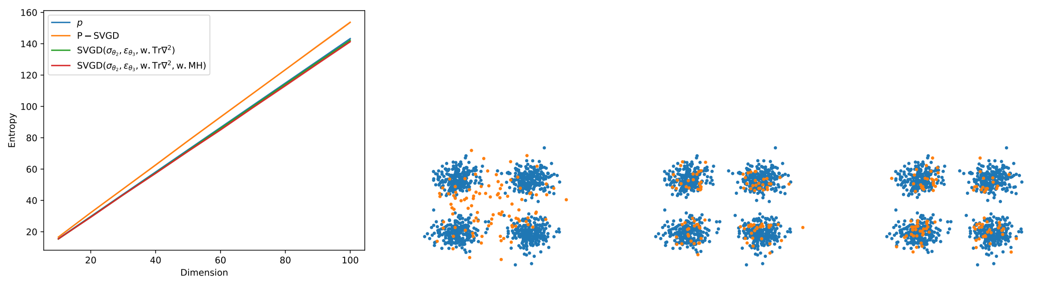
	Kernel Architecture	
	Architecture	$\sigma_{\theta_2} = \text{GNN}(\{x_i^l\}_{i=1}^M; \theta_2)$
	Learning Rate Architecture	
	Architecture	$\epsilon_{\theta_3}^l = \min\left(\epsilon_{\theta_3}^0, \epsilon_{\theta_3}^0 d^{l/s_{\theta_3}}\right)$
	Initial Learning Rate $\epsilon_{\theta_3}^0$	$\epsilon_{\theta_3}^0 = 0.1$
	Decay Factor d_{θ_3}	$d_{\theta_3} = 5 \times 10^{-3}$
	Decay Scale s_{θ_3}	$s_{\theta_3} = L$
	Training Parameters	
	Optimizer	Adam
	Learning Rate	$5 \cdot 10^{-3}$
	Epochs	300
	Activation Functions	{Exponential(), Truncate(min, max)}

Table 9: Experimental setup for Fig. 22

10.3 High Dimensional GMMs

The goal of this experiment is to further assess the scalability of **MET-SVGD**.

Implementation Details The target distribution is a mixture of 4 d-dimensional Gaussian distributions $p(x) = \sum_{k=1}^4 0.25\mathcal{N}(x, \mu_k, I_d)$ with uniform mixture ratios. The first two coordinates of the mean vectors are equally spaced on a circle, while the other coordinates are set to 0 (Fig. 23.A-D). Particles are initialized from $cN(0, I_d)$ and only the first two dimensions need to be learned. In Fig. 23A, we show that **MET-SVGD** efficiently recovers the low-dimensional structure.



(A) Estimated entropy across dimensions. (B) **P-SVGD** (C) **MET-SVGD** (w. MH) (D) **MET-SVGD** (wo. MH)

Figure 23: Scalability results. Target is a high-dimensional GMM. **MET-SVGD** successfully recovers the entropy of the low-dimensional GMM.

11 Additional Results: Energy Based Models

Proposition 11.1 (Sec.4.2)**.** *Training EBMs* $p_\theta(x) = \bar{p}_\phi(x)/Z$ *via maximum likelihood* ($\mathcal{L}_{ebm}(\phi, \theta) = -\mathbb{E}_{x\sim p_d}[\log p_\theta(x)]$ *)is intractable due to the partition function* Z*. When the sampler has a tractable distribution* q_ϕ*, a tight lower bound can be computed in return:* $\mathcal{L}_{ELBO}(\phi) = \mathbb{E}_{x\sim q}[\log \bar{p}_\phi(x)] - \mathbb{E}_{x\sim p_d}[\log \bar{p}_\phi(x)] + \mathcal{H}(q_\phi)$ *with* p_d *being the data distribution,*

Proof. Given:

$$\mathcal{L}_{\text{ebm}}(\phi, \theta) = -\mathbb{E}_{x\sim p_d}[\log p_\theta(x)] = -\mathbb{E}_{x\sim p_d}[\log \bar{p}_\phi(x)] + \log Z(\theta).$$

We bound the partition function using the KL-divergence:

$$\begin{aligned}
\log Z(\theta) &\geq \log Z(\theta) - D_{\text{KL}}(q_\phi(x) \| p_\theta(x)) \\
&\geq \log Z(\theta) + \int_x q_\phi(x) \log \frac{p_\theta(x)}{q_\phi(x)} dx \\
&\geq \log Z(\theta) + \int_x q_\phi(x) \log \frac{\frac{\bar{p}_\phi(x)}{Z(\theta)}}{q_\phi(x)} dx \\
&\geq \log Z(\theta) + \int_x q_\phi(x) \log \bar{p}_\phi(x) dx - \int_x q_\phi(x) \log Z(\theta) dx - \int_x q_\phi(x) \log q_\phi(x) dx \\
&\geq \log Z(\theta) + \mathbb{E}_{x\sim q_\phi}[\log \bar{p}_\phi(x)] - \log Z(\theta) + \mathcal{H}(q_\phi) \\
&\geq \mathbb{E}_{x\sim q_\phi}[\log \bar{p}_\phi(x)] + \mathcal{H}(q_\phi).
\end{aligned}$$

Substituting back into the MLE objective:

$$\begin{aligned}
\mathcal{L}_{\text{ebm}}(\phi, \theta) &= -\mathbb{E}_{x\sim p_d}[\log \bar{p}_\phi(x)] + \log Z(\theta) \\
&\geq -\mathbb{E}_{x\sim p_d}[\log \bar{p}_\phi(x)] + \mathbb{E}_{x\sim q_\phi}[\log \bar{p}_\phi(x)] + \mathcal{H}(q_\phi) \\
&\geq \mathcal{L}_{\text{ELBO}}(\phi).
\end{aligned}$$

□

11.1 Synthetic Experiment: Moon Distribution

We evaluate on the Moon dataset [Rezende and Mohamed, 2015b] with varying smoothness.

Implementation Details (Fig. 24) are described in Tab. 10.

	Parameter	**Value**
	Target distribution Initial distribution	$p_\theta(x) = \frac{\exp f_\theta(x)}{Z_\theta}$ $f_\theta(x) = \text{MLP}_\theta(128, \text{Swish}, 128, \text{Swish}, 128, \text{Swish}, 1)$ $q^0 = \mathcal{N}([0,0], 7I)$
Default SVGD parameters	Learning rate Number of steps Number of particles Kernel variance	$\epsilon = \epsilon^l_{\theta_2}$ (l free learnable parameters) $L = 100$ $m = 129$ $\sigma = \sigma^l_{\theta_2}$ (l free learnable parameters)
Training	Optimizer θ Learning rate ϕ Learning rate Epochs	Adam 10^{-3} 10^{-2} 1250
Resources	GPU RAM Per-epoch runtime	Tesla V100-SXM2-32GB 2 GB 2.6 seconds

Table 10: Experimental configuration for EBM results.

Performance: As smoothness decreases, **MET-SVGD** consistently outperforms all baselines in terms of the MMD score [Dai et al., 2019a], where $\text{MMD}(p,q) = \mathbb{E}_{x,x'\sim p}\left[\kappa(x,x')\right] + \mathbb{E}_{y,y'\sim q}\left[\kappa(y,y')\right] - 2\mathbb{E}_{x\sim p, y\sim q}\left[\kappa(x,y)\right], \quad \text{s.t} \quad \kappa(x,y) = \exp\left(-\|x-y\|^2/2\sigma^2\right)$

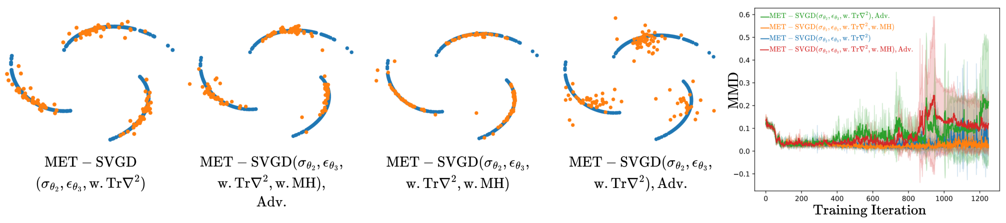


Figure 24: EBM Results. **MET-SVGD** outperforms **P-SVGD** and **LD** on learning EBMs to fit non-smooth data distributions.

11.2 Image generation

Implementation Details (Fig. 26) are reported in Tab. 11. All experiments were conducted on a single NVIDIA A100 80GB with 8GB allocated memory; average runtime was around 6 seconds per iteration (processing one batch of data composed of 64 particles).

Qualitative Results. In Fig. 26, we visualize generated images sampled from the different models.

FID and Inception Score In Fig. 26, we report the FID and IS scores for baselines: (1) LD trained with contrastive divergence:

$$\min_\theta \mathcal{L}_{\text{CD}}(\theta) = \min_\theta -\mathbb{E}_{x\sim p_d}\left[f_\theta(x)\right] + \mathbb{E}_{x\sim q}\left[f_\theta(x)\right], \tag{19}$$

with q being the empirical distribution induced by the LD particles; (2) **P-SVGD** and **MET-SVGD** variants trained adversarially by alternating between learning the sampler parameters

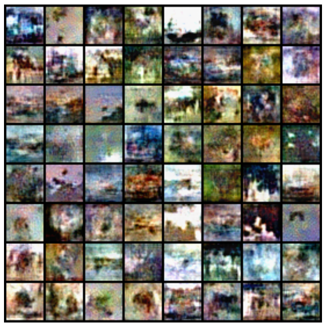
$$\min_\phi -\mathcal{L}_{\text{ELBO}}(\phi) = \max_\phi -\mathbb{E}_{x\sim p_d}\left[f_\theta(x)\right] + \mathbb{E}_{x\sim q}\left[f_\theta(x)\right] + \mathcal{H}(q_\phi), \tag{20}$$

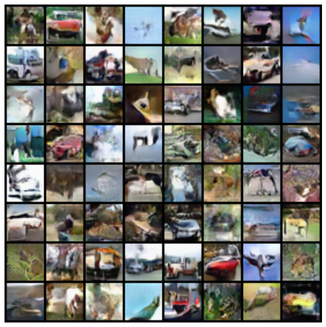
and minimizing the contrastive divergence

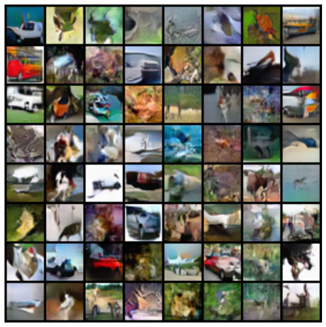
$$\min_\theta \mathcal{L}_{\text{CD}}(\theta) = \min_\theta \mathbb{E}_{x\sim q}[\log \bar{p}_\phi(x)] - \mathbb{E}_{x\sim p_d}[\log \bar{p}_\phi(x)] \tag{21}$$

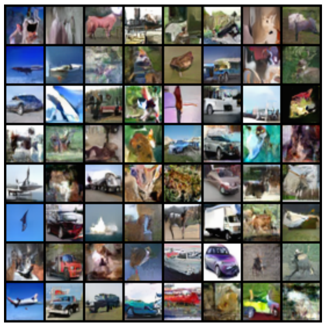
in Fig. 28A and Fig. 26C. (3) **MET-SVGD** variants trained adversarially using the ELBO loss for both learning the sampler and the energy:

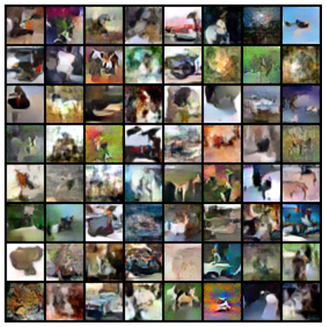
$$\min_\theta \max_\phi \mathcal{L}_{\text{ELBO}}(\theta,\phi) = \min_\theta \max_\phi \mathbb{E}_{x\sim q}[\log \bar{p}_\phi(x)] - \mathbb{E}_{x\sim p_d}[\log \bar{p}_\phi(x)] + \mathcal{H}(q_\phi), \tag{22}$$

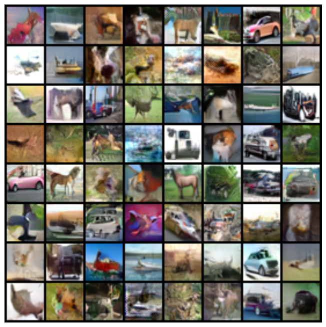

(A) MET-SVGD($\sigma_{\theta_2}, \epsilon_{\theta_3}$, w. Tr$\nabla^2$, w. MH)


(B) MET-SVGD($\sigma_{\theta_2}, \epsilon_{\theta_3}$, wo. Tr$\nabla^2$)

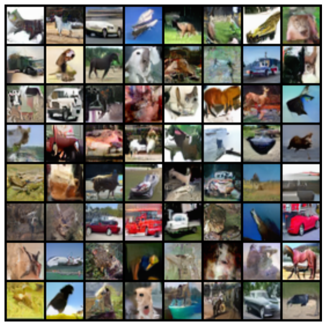

(C) MET-SVGD($\sigma_{\theta_2}, \epsilon_{\theta_3}$, w. Tr$\nabla^2$, $\mathrm{Adv.}$)

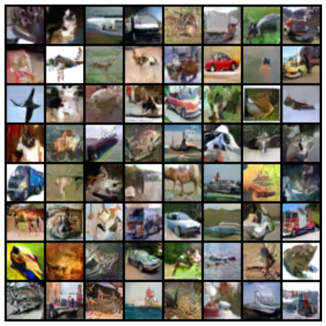

(D) MET-SVGD($\sigma_{\theta_2}, \epsilon_{\theta_3}, L_c$, w. Tr$\nabla^2$)


(E) P-SVGD


(F) LD

(G) MET-SVGD(σ_{θ_2}, w. Tr∇^2)


(H) MET-SVGD($\sigma_{\theta_2}, \epsilon_{\theta_3}$, w. Tr$\nabla^2$)


(I) MET-SVGD($\sigma_{\theta_2}, \epsilon_{\theta_3}$, w. Tr$\nabla^2$, q_0=RB-PCA, Adv.)

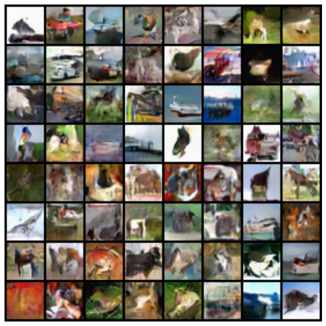

(J) MET-SVGD($\sigma_{\theta_2}, \epsilon_{\theta_3}$, w. Tr$\nabla^2$, w. MH, q_0=RB-PCA)

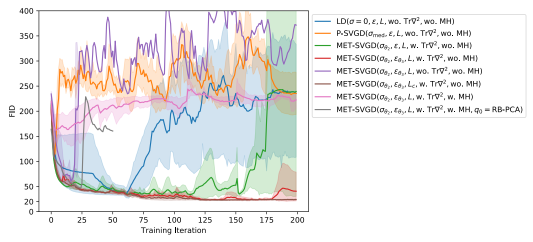
Figure 25: Image generation using EBMs across different configurations.

	Parameter	Value
	Target distribution	$p_\theta(x) = \frac{\exp f_\theta(x)}{Z_\theta}$ $f_\theta(x)$ is a WideResnet(28,10) network
	Initial distribution	q^0 is a replay buffer initialized using a GMM whose modes are based on the class-conditional means and co-variances
Default SVGD parameters	Learning rate	$\epsilon_{\mathrm{LD}} = \epsilon_{\mathrm{P-SVGD}} = 64$ (this number is divided by m in the SVGD update formula) $\epsilon_{\mathrm{MET-SVGD}} = \mathrm{GNN}(\{x_i^l\}, \{\nabla_{x_i^l} \log p_\theta(x_i^l)\}; \theta_3)$
	Number of steps	$L = 5$ $L_c = 10$
	Number of particles	$M = 64$
	Kernel variance	$\sigma_{\mathrm{LD}} = 0$ $\sigma_{\mathrm{P-SVGD}} = \sqrt{\frac{\mathrm{med}(\lVert x_i - x_j \rVert^2)}{2 \ln M}}$ $\sigma_{\mathrm{MET-SVGD}} = \mathrm{GNN}(\{x_i^l\}; \theta_2)$
Training	Optimizer	SGD
	θ Learning rate	10^{-1} with 1000 iterations warm-up and decay at epochs 60, 120, and 180.
	ϕ Learning rate	10^{-4}
	Epochs	200
Resources	GPU	NVIDIA A100 80GB
	RAM	8 GB
	Per-iteration runtime	6 seconds

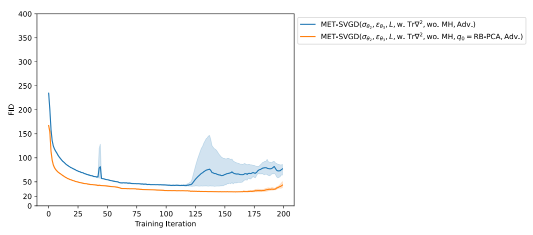
Table 11: Experimental configuration for EBM results.

in Fig. 28B and Fig. 26D. The second setup (Fig. 28A, Fig. 26C) led to the best result.

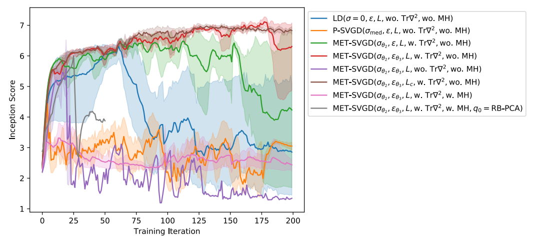
Performance: The best FID (lowest) was obtained when both the kernel bandwidth and step-size are learnt. Comparative results with better scalability are obtained with an adaptive number of steps L_c. Removing the trace led to early divergence showcasing the importance of this correction. Both LD and **P-SVGD** resulted in early divergence. This was also the case for the setup with MH due to high rejection rates and hence limited number of steps (constrained by the GPU memory) leading to poor convergence to the target. In the future, we will explore optimizing the memory usage to afford more steps. Also, we find that the performance improvement obtained from learning the step-size on top of the kernel bandwidth, *i.e.*, **MET-SVGD**($\sigma_{\theta_2}, \epsilon_{\theta_3}$) vs. **MET-SVGD**($\sigma_{\theta_2}$), is due to the learning the step-size resulting in smoother energy landscapes (Fig. 27). The third setup (Fig. 28B, Fig. 26D), where both the sampler and the energy are learnt using the ELBO (with the entropy term in learning the energy) didn't work as well. In Fig. 27, we show that this is due to the score exploding frequently leading to almost zeo learning rates. We plan to address this by contraining the Lipschitz constant of the deepnet in the future.



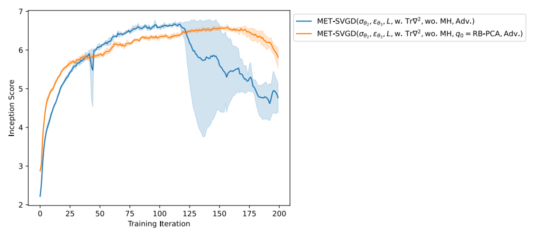
(A) FID score $\left(\mathcal{L}_{\text{ELBO}}(\phi), \mathcal{L}_{\text{CD}}(\theta)\right)$



(B) FID score $\left(\mathcal{L}_{\text{ELBO}}(\phi, \theta)\right)$



(C) IS score $\left(\mathcal{L}_{\text{ELBO}}(\phi), \mathcal{L}_{\text{CD}}(\theta)\right)$



(D) IS score $\left(\mathcal{L}_{\text{ELBO}}(\phi, \theta)\right)$

Figure 26: EBM Results. We report the FID and IS scores across training iterations for both (A-B) the set-up where the sampler (ϕ) is trained using $\mathcal{L}_{\text{ELBO}}(\phi)$ and energy (θ) using $\mathcal{L}_{\text{CD}}(\theta)$, and (B-C) the setup where the sampler and the energy are learnt adversarially using $\mathcal{L}_{\text{ELBO}}(\theta, \phi)$. *Tr*$\nabla^2$ stands for including the trace of Hessian term and q_0 = RB-PCA for initializing the replay buffer with samples obtained via a linear combination of the principal components of the data samples.

Smoothness. In Fig. 27, we visualize the scores of the learn distribution $\nabla_x f_\theta(x)$ across training iterations. The setups with the lowest FID and highest IS score are associated with the smoothest landscapes, *i.e.*, lowest scores. This is the case of **MET-SVGD**($\sigma_{\theta_2}, \epsilon_{\theta_3}$, w. *Tr*$\nabla^2$) and **MET-SVGD**($\sigma_{\theta_2}, \epsilon_{\theta_3}$, L_c, w. *Tr*∇^2). The diverging setups are associated with frequently exploding scores.

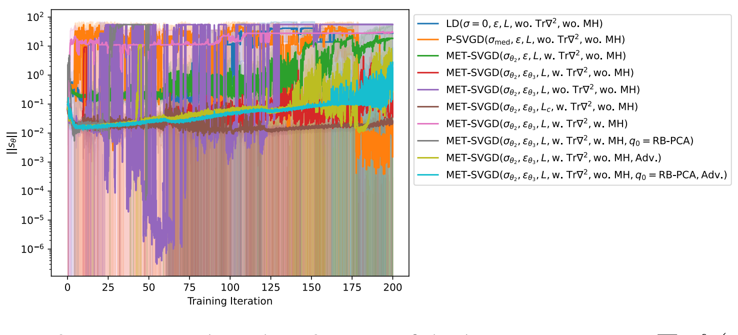
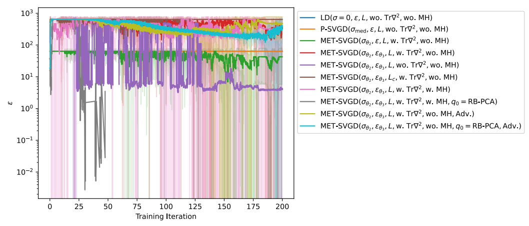
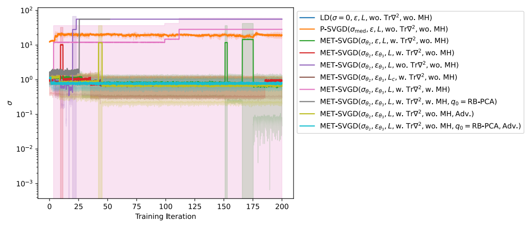


Figure 27: EBM results. The L2-norm of the learnt EBM score $\nabla_x f_\theta(x)$.

Trainable SVGD Hyperparameters. In Fig. 28, we visualize the SVGD step-size and kernel-bandwidth across training iterations. We observe that σ_{med} is frequently higher that the learned ones. The learned step-size is also higher than the **P-SVGD** one in setups that led to the best FID. In setups with high FID, we observe that ϵ_{θ_3} is associated with high variance: it frequently becomes very small. This is mostly driven by the score of the energy.



(A) Average SVGD step-size $\epsilon^l_{\theta_3}$ across training iterations.



(B) SVGD kernel bandwidth $\sigma^l_{\theta_2}$ across training iterations.

Figure 28: EBM Results. Visualization of the learnt kernel bandwidth and step-size across training iterations.

12 Additional Results: MaxEntr RL

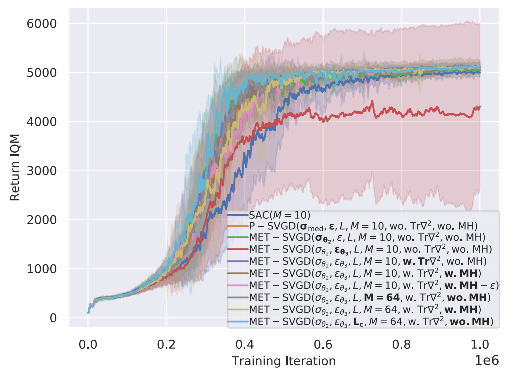
Implementation Details for (Fig. 10) are reported in Tab. 12

Table 12: Hyperparameters

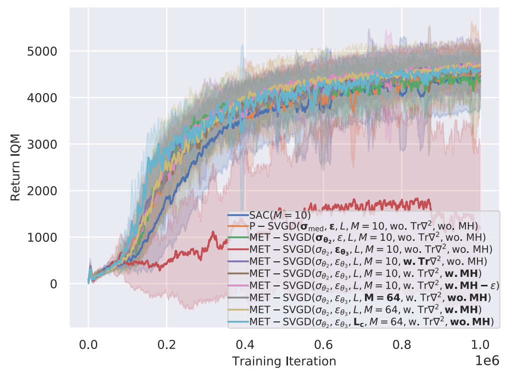
	Hyperparameter	Value
Training	Optimizer	Adam
	Actor and Critic Learning rate	10^{-4} for Humanoid and 10^{-3} for all other environments
	Batch size	100
Deepnet	Number of hidden layers	2 Critic and 3 Actor
	Number of hidden units per layer	256
	Nonlinearity	ELU
RL	Target smoothing coefficient	0.005
	Discount γ	0.99
	Target update interval	1
	Entropy weight α	0.2
	Replay buffer size $\|\mathcal{D}\|$	10^6
SVGD	Initial distribution	$q_0 = \mathcal{N}(\mu_\theta, \mathrm{diag}(\sigma_\theta))$
	Number of steps	$L = 3$
	Number of particles	$M = 10$
	Kernel variance	$\sigma^2 = \frac{\sum_{i,j} \|a_i - a_j\|^2}{4(2\log m+1)}$
		$\sigma = \mathrm{GNN}(s_t, \{x_i^l\}, \{\nabla_{x_i^l} \log p(x_i^l)\}; \theta_2)$
	Learning rate	$\epsilon = 0.1$
		$\epsilon = \mathrm{GNN}(s_t, \{x_i^l\}, \{\nabla_{x_i^l} \log p(x_i^l)\}; \theta_3)$

Performance. In Fig. 29A and Fig. 29B, we report the Inter Quantile Mean (IQM) return values averaged over 5 runs, where every run is the average of 10 evaluations of the policy. We refer to the approach leveraging **P-SVGD** as proposed by Messaoud et al. [2024] as S^{2}AC, and our approach as S^{2}AC$^+$. We follow the same convention form EBM experiments for naming the different methods: for the different S^{2}AC$^+$ variants, we only include arguments that are different from the S^{2}AC setup. The default parameters for S^{2}AC are ($q^0_{\theta_1}$, $\epsilon = 1e^{-4}$, $M = 10$, $L = 3$), divergence control w. particles truncation, wo. *Tr*∇^2). In the Humanoid environment (Fig. 29A), adding the missing trace of Hessian term resulted in faster convergence. Increasing the number of particles and incorporating MH for divergence control led to improved performance: better exploration through more particles and exploitation through the MH step.

In Walker environment (Fig. 29B), MH helped achieve higher return. We also see that S^{2}AC($\sigma_{\theta_2}, \epsilon_{\theta_3}$) failed because the learning rate became very small for many iterations (Fig. 35A), which is visible in the high variance of the score norm (Fig. 36). This is the same phenomenon observed in the other EBM experiments.

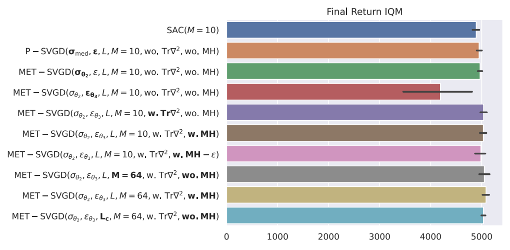


(A) Humanoid's IQM Return.

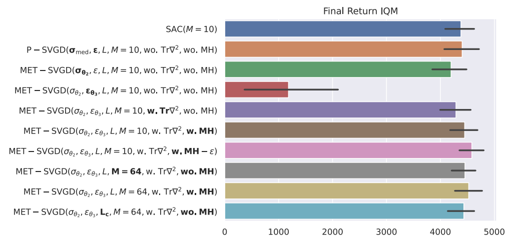


(B) Walker's IQM Return.

Figure 29: IQM return scores across environments.

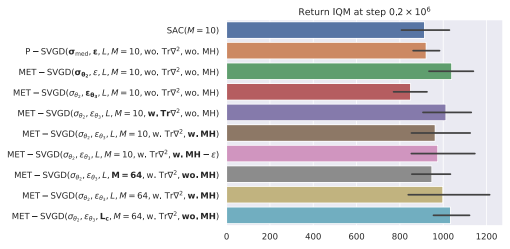


(A) Humanoid's Final IQM Return.

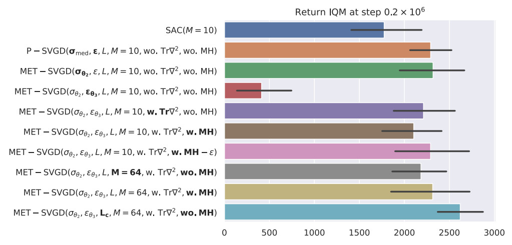


(B) Walker's Final IQM Return.

Figure 30: Final IQM return scores across environments.



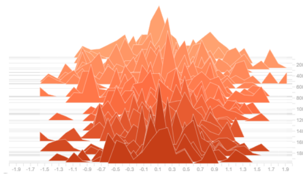
(A) Humanoid's IQM Return at step 0.2×10^6.

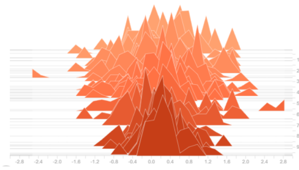


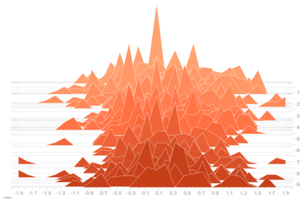
(B) Walker's IQM Return at step 0.2×10^6.

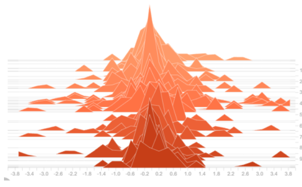
Figure 31: IQM return scores at step 0.2×10^6 across environments.

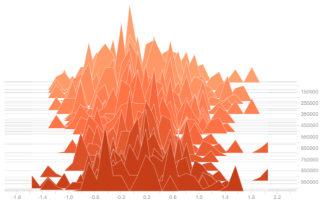
Trainable SVGD Parameters (Humanoid Env) In Fig. 32, we visualize a histogram of the mean of **initial distribution** $q^0_{\theta_1}$ across training iterations. We observe that the mean has several components outside the [-1,1] range of valid actions. While the actions are truncated to satisfy the constraints, this still limits the exploration as many particles would end-up having -1/1 as values. This trend is exacerbated across S^{2}AC$^+$ variants, especially for the cases of learnable step-size ϵ_{θ_3}, adaptive number of steps L_c and larger number of particles $M = 64$. In the future, we will explore mechanisms for constraining the support of the policy distribution to the valid range. This is not a trivial problem as the obvious solution of truncating the mean leads to vanishing gradients and modeling $q^0_{\theta_1}$ as a distribution with a limited support (*e.g.*, beta distribution) is not obvious as such a distribution is highly sensitive to parameters with big ranges. Also, enforcing the constraints in the Q-value through reward is not trivial as it can lead to non-smooth hard-to-learn landscapes. This poor exploration limits the effect of our contribution, as our approach helps explore better in the local neighborhood of the modes identified through exploration.

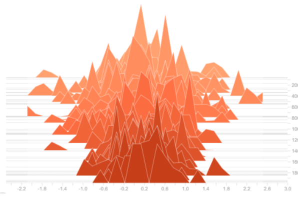

(A) SAC

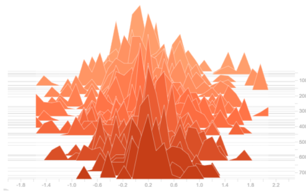

(B) $\mathrm{S}^2\mathrm{AC}$

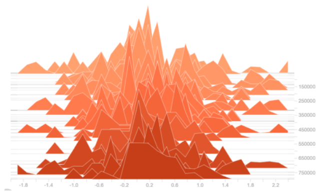

(C) $\mathrm{S}^2\mathrm{AC}^+(\sigma_{\theta_2})$


(D) $\mathrm{S}^2\mathrm{AC}^+(\sigma_{\theta_2}, \epsilon_{\theta_3})$


(E) $\mathrm{S}^2\mathrm{AC}^+(\sigma_{\theta_2}, \epsilon_{\theta_3}, Tr\nabla^2)$


(F) $\mathrm{S}^2\mathrm{AC}^+(\sigma_{\theta_2}, \epsilon_{\theta_3}, Tr\nabla^2, M = 64)$


(G) $\mathrm{S}^2\mathrm{AC}^+(\sigma_{\theta_2}, \epsilon_{\theta_3}, Tr\nabla^2, M = 64, L_c)$


(H) $\mathrm{S}^2\mathrm{AC}^+(\sigma_{\theta_2},\epsilon_{\theta_3}, Tr\nabla^2, M = 64, L_c$, w. MH)

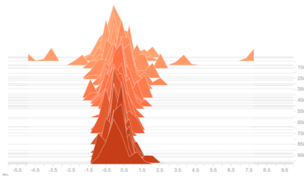

(I) $\mathrm{S}^2\mathrm{AC}^+(\sigma_{\theta_2},\epsilon_{\theta_3}, Tr\nabla^2, L_c$, w. MH)

Figure 32: Histogram of the mean of q^0 across training iterations (Humanoid env).

In Fig. 35, we present a histogram of the learned **kernel bandwidth** across training iterations. Note that in these experiments $\sigma_{\theta_2} \in \mathbb{R}^d$. We observe that for certain dimensions the bandwidth was small indicating independant particles while for other states the particles were more interdependent (large σ_{θ_2} values). Also, note that the kernel bandwidth values for S2AC are consistently large.

The **SVGD step-size** $\epsilon^l_{\theta_3}$ is visualized in Fig. 34.

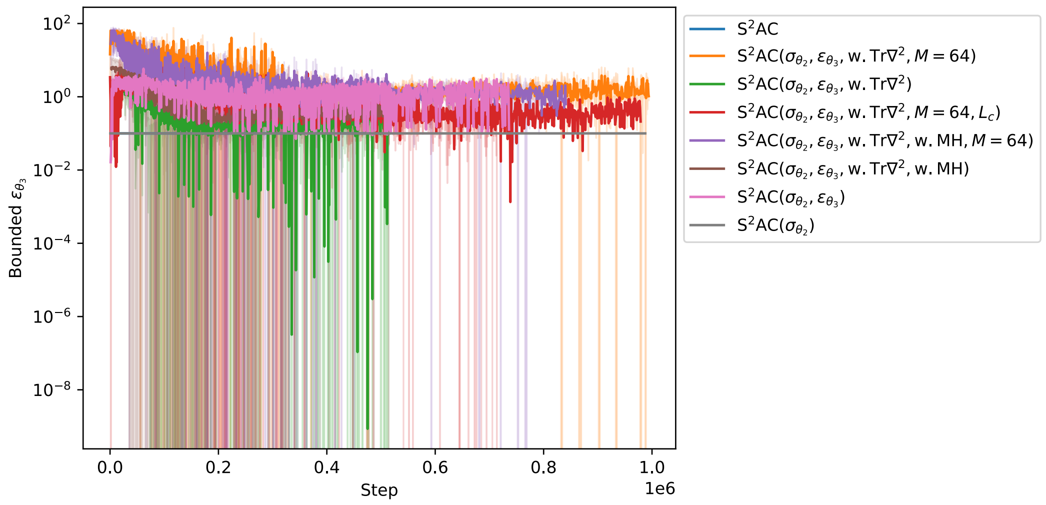
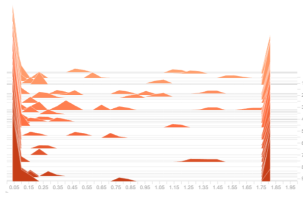
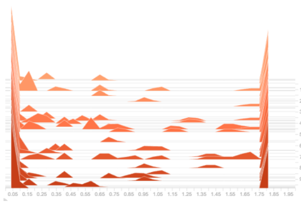
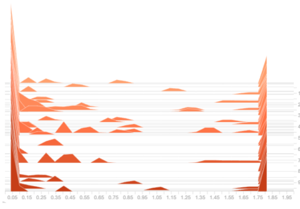
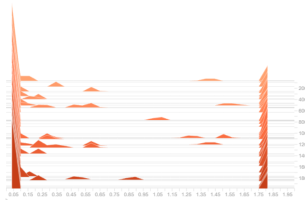


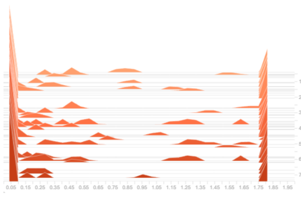
Figure 34: Learned step-size $\epsilon^l_{\theta_3}$ for Humanoid env.

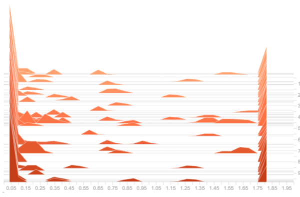

(A) S^{2}AC$^+$(σ_{θ_2})

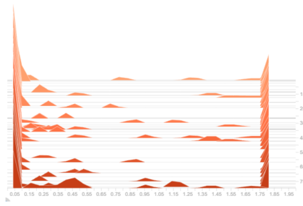

(B) S^{2}AC$^+$($\sigma_{\theta_2}, \epsilon_{\theta_3}$)

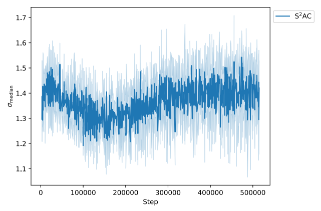

(C) S^{2}AC$^+$($\sigma_{\theta_2}, \epsilon_{\theta_3}$,$Tr\nabla^2$)


(D) S^{2}AC$^+$($\sigma_{\theta_2}, \epsilon_{\theta_3}$,$Tr\nabla^2$, $M =$ 64)


(E) S^{2}AC$^+$($\sigma_{\theta_2}, \epsilon_{\theta_3}$,$Tr\nabla^2$, $M =$ 64, L_c)

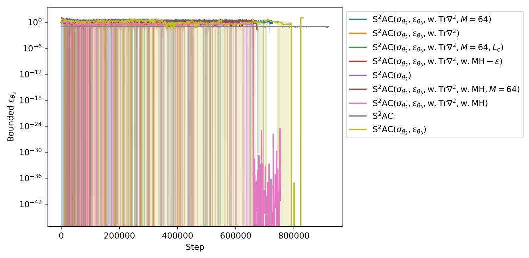

(F) S^{2}AC$^+$(σ_{θ_2}, ϵ_{θ_3},$Tr\nabla^2$, w. MH)


(G) S^{2}AC$^+$($\sigma_{\theta_2}, \epsilon_{\theta_3}$,$Tr\nabla^2$, $M =$ 64, w. MH)

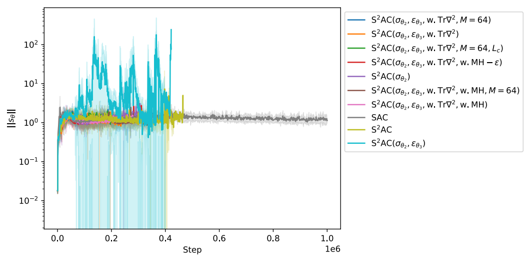


(H) S^{2}AC

Figure 33: Histogram of the kernel bandwidth $\sigma^l_{\theta_2}$



(A) Learnable step-size ϵ^l_3 (Walker env.)



(B) Scores (Walker env.)

Figure 35: Learned step-size and scores in Walker env.

Trainable SVGD Parameters (Walker Env). We visualize the scores and step-size in Fig. 35A and Fig. 36.

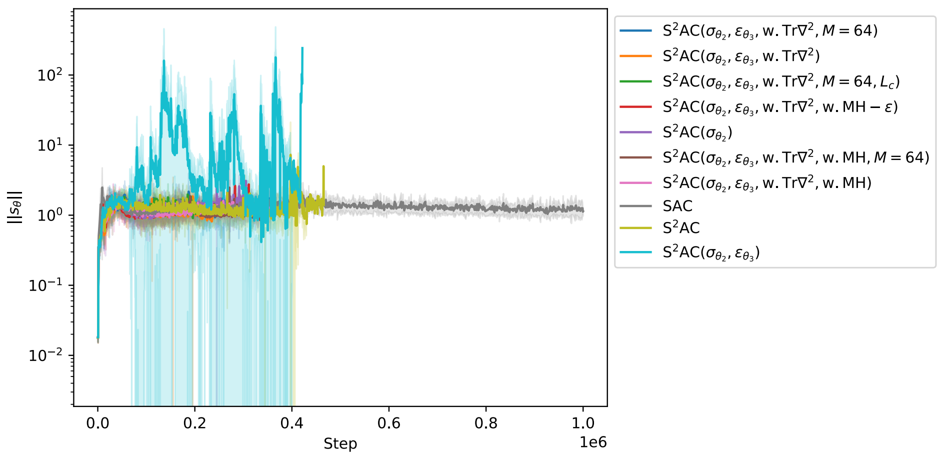


Figure 36: Scores (Walker env.)