Can We Estimate The Entropy Of Arbitrary Distributions Known Up To A Normalization Constant?

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*. Messaoud et al. [54] introduced **P-SVGD**, a particle-based variational method using Stein Variational Gradient Descent. However, we show that **P-SVGD** scales poorly to high-dimensional spaces. We propose **MET-SVGD**, an extension of **P-SVGD** that scales efficiently with convergence guarantees. **MET-SVGD** achieves SOTA results on scaling SVGD. We significantly outperform **P-SVGD** on entropy estimation, Maximum Entropy Reinforcement Learning, and image generation with Energy-Based Models benchmarks. Code: https://tinyurl.com/2esyfx8j.

1. Introduction

The differential entropy [19, 71] 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 entropy plays a central role in machine learning [66]. While significant progress has been made on estimating entropies from samples [8, 56], settings where only the unnormalized density is available remain under-explored. Recently, Messaoud et al. [54] introduced Parametrized Stein Variational Gradient Descent (**P-SVGD**), which leverages Stein Variational Gradient Descent (SVGD) sampler [49] 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:

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

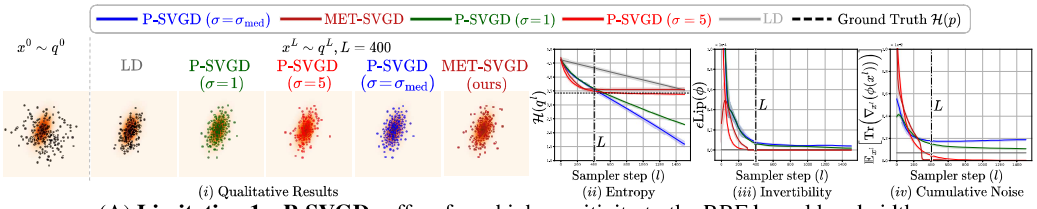
following the update rule $x_i^{l+1} = x_i^l + \epsilon\phi(x_i^l)$. ϵ is the step-size, q^l is the particles distribution at step $l \in [1, L]$ and $\kappa(\cdot,\cdot)$ is typically an RBF kernel, *i.e.*, $\kappa(x_i, x_j) = \exp(-||x_i - x_j||^2/2\sigma^2)$, with bandwidth σ^2 set heuristically as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

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

where $\bar{\phi}(x^l, x_j^l)$ is the contribution of x_j^l to the update of particle x^l and $\phi(x^l) = \mathbb{E}_{x_j^l}[\bar{\phi}(x^l, x_j^l)]$ is defined in Eq.1. The trace term is approximated using only first-order derivatives and, for efficiency, the authors omit a trace-of-Hessian term $\mathrm{Tr}(\nabla^2_{x^l}\log p(x^l))$ by sampling $x_j^l \neq x^l$:

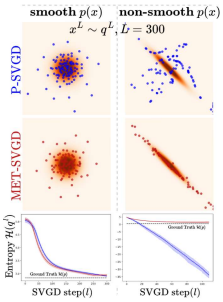
$$\log q^L(x_i^L) = \log q^0(x_i^0) - \epsilon\sum_{l=0}^{L-1}\sum_{i\neq j=0}^{M-1}\frac{\kappa(x_j^l, x_i^l)}{M\sigma^2}\left(d - \frac{\|x_i^l - x_j^l\|^2}{\sigma^2} - (x_i^l - x_j^l)^\top\nabla_{x_j^l}\log p(x_j^l)\right) + \mathcal{O}(\epsilon^2). \tag{3}$$

q^L can approximate a broad class of densities under mild assumptions [82], 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. 1). 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. **MET-SVGD** learns step-wise kernel bandwidths and step-sizes, replacing the non-robust heuristics of **P-SVGD**; upgrades its local invertibility condition to a principled global one that satisfies the assumptions in the entropy derivation; consolidates two independent, informal step-size constraints into a unified,

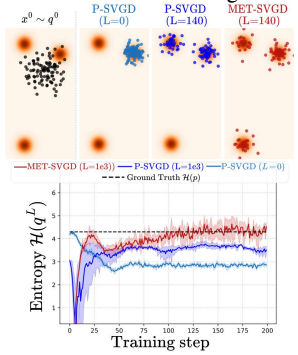


(*i*) Qualitative Results (*ii*) Entropy (*iii*) Invertibility (*iv*) Cumulative Noise

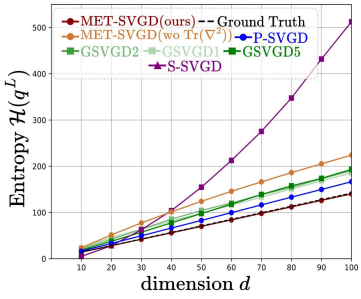
(A) **Limitation 1: P-SVGD** suffers from high sensitivity to the RBF kernel bandwidth σ



(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. [2022]. MET-SVGD(wo. Tr(∇^2)) refers to the variant without the correction term.

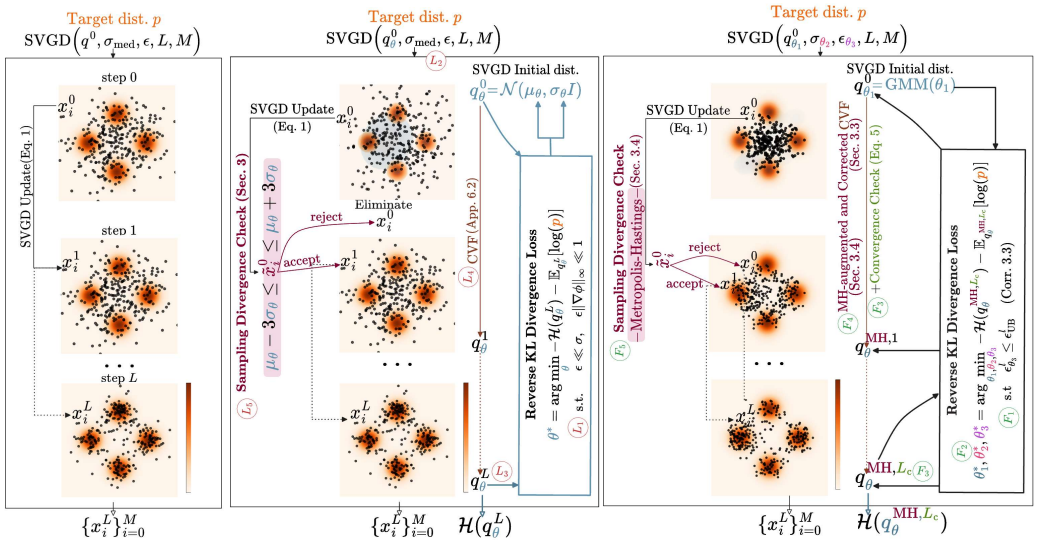
Figure 1: **P-SVGD** limitations.

principled rule; efficiently restores the missing Hessian term via Hutchinson's estimator; incorporates a Metropolis–Hastings correction step to ensure asymptotic convergence; and adaptively tunes the number of sampling steps to better handle complex, high-dimensional distributions.

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 [12], 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. 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 approximates p with a tractable, sample-efficient distribution and estimates $\mathcal{H}(p)$ using the entropy of this distribution. **MET-SVGD** introduces a series of optimizations to address **P-SVGD**'s key limitations as illustrated in Fig. 2B: [L_1-F_1] **P-SVGD** introduces two informal independent 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. 2.1). [L_2-F_2] **P-SVGD** exhibits high sensitivity to hyperparameters (Fig. 1A-*iii*) with no tuning guidelines; we show that this is due to the accumulation of noise in the trace term of Eq. 2, leading to entropy divergence and mitigates this by learning the SVGD hyperparameters via reverse KL minimization (Sec. 2.2). [L_3-F_3] **P-SVGD** suffers from poor convergence to non-smooth targets and sampling mode collapse (Fig. 1B and Fig. 1C), 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. 2.4). [L_4-F_4] P-SVGD's omission of the trace-of-Hessian term limits its scalability to high-dimensions (Fig. 1D) which **MET-SVGD** efficiently



(A) **SVGD** (B) **P-SVGD** (C) **MET-SVGD** (ours)

Figure 2: Novelty over **P-SVGD**. Limitations (L) and corresponding Fixes (F).

restores as explained in Sec. 2.3. [L_5-F_5] P-SVGD uses a fixed number of steps (L), which may be insufficient for convergence; **MET-SVGD** determines L_c using the Stein Identity (Sec. 2.2).

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

In **P-SVGD**, Eq. 3 was derived by (1) leveraging the CVF (App. 5.2) 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. [7]) to SVGD. We also derive a precise condition on ϵ for the log-det approximation (Proposition 2) and unify both into a single efficient bound (Corollary 3).

Proposition 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 $Lip(\phi^l)$ the Lipschitz constant of the velocity ϕ^l at step l. f is invertible if: $\epsilon\, Lip(\phi^l) < 1$, for all $l \in [0, L-1]$.*

The proof is in App. 7.1. Using the mean value theorem (App.5.6), 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 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 *The distribution induced by the SVGD update (Eq. 1) using an RBF kernel is given by Eq. 3 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. 7.3, 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.

2.2. Optimized SVGD Parameters

A major drawback of **P-SVGD** is its high sensitivity to the RBF kernel bandwidth σ, which [54] attribute to violation of the invertibility of the SVGD update rule (Eq. 1): In a 2-d Gaussian target setup (reproduced in Fig. 1A), they show that, paradoxically, although SVGD and Langevin Dynamics (LD) (update rule in App. 5.1) *qualitatively* converge to the target, *i.e.*, the particles reach high-density regions (Fig. 1A-i), the entropy estimate only converges for specific σ values, *e.g.*, $\sigma=5$ (Fig. 1A-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. 1A-iii, we show that the step-size condition from Corollary 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. 2, *i.e.*, $\mathbb{E}_{x^l\sim q^l}[Tr(\nabla_{x^l}\phi(x^l))] \nrightarrow 0$ as $l\to\infty$ (Fig. 1A-iv), resulting in a quasi-linear growth in the entropy with the number of steps (Fig. 1A-ii). To address this, we propose leveraging the closed-form expression of $q_\theta^{L_c}$ to learn a step-wise kernel bandwidth $\sigma_{\theta_2}^l$ and step-size $\epsilon_{\theta_3}^l$ alongside the initial distribution $q_{\theta_1}^0$ by minimizing the reverse KL-divergence:

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

with ϵ_{UB}^l being the upper-bound from Corollary. 3 and $\theta=\{\theta_i\}_{i=1}^3$. Besides, we propose an efficient convergence check enabling an adaptive number of steps L_c, rather than fixing it a priori. 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. 13A. Learning the kernel bandwidth alone is insufficient to ensure convergence, *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 $l\to\infty$. We provide a proof in App. 9. We propose learning a step-wise **step-size** $\epsilon_{\theta_3}^l$ to be flexibly modulated for convergence. In Fig. 13C, $\epsilon_{\theta_3}^l$ eventually becomes 0. Besides, the **Number of Steps**, L, in **P-SVGD**, is fixed, which does not guarantee convergence to the target. To address this, **MET-SVGD** uses an adaptive number of steps L_c determined dynamically using the Stein Identity (SI) as a convergence check (Proposition 4). We derive a practical SI form that depends on $Tr(\nabla_{x^l}\phi(x^l))$ (App. 8.3).

Proposition 4 *The Stein identity $SI(q^l,p)$ at every step l of the SVGD update is computed as:*

$$SI(q_\theta^l, p) = \sqrt{\mathbb{E}_{x^l}\left[\phi_\theta(x^l)^T\nabla_{x^l}\log p(x^l) + Tr(\nabla_{x^l}\phi_\theta(x^l))\right]} \tag{4}$$

2.3. Corrected Derivation of q_θ^l

As explained in Sec. 2, **P-SVGD** approximate the expectation over particles in Eq. 2 by excluding the updated particle itself ($x^l \neq x_j^l$), 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 and translates into inconsistent updates across particles ($i=j$ term is missing), leading to particles following different distributions, and making the entropy ill-defined. This is a key source of **P-SVGD**'s poor scalability as shown in Fig. 1D (orange vs brown). To address this, **MET-SVGD** adds the missing term to the entropy using (1) the Hutchinson estimator [35] and (2) the double differentiation trick [76]: $Tr(\nabla^2_{x_i^l}\log p(x_i^l)) \overset{(1)}{=} \mathbb{E}_{v\sim p_v}[v^T\nabla^2_{x_i^l}\log p(x_i^l)v] \overset{(2)}{=} \mathbb{E}_{v\sim p_v}[\nabla_{x_i^l}(v^T\nabla_{x_i^l}\log p(x_i^l))v]$, 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. 14. Notably, the trace term in SVGD is scaled by the number of particles M:

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

unlike LD: $\log q_\theta^L(x^L) = \log q_{\theta_1}^0(x^0) + (\epsilon_{\theta_3}^l/V)\sum_{l=0}^{L-1}\sum_{v=0}^{V-1}\nabla_{x^l}(v^T\nabla_{x^l}\log p(x^l))v$ (App. 7.8).

2.4. Divergence Control via Metropolis Hastings

To prevent divergence during sampling due to steepness in the target, **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. 2B). This heuristic, however, exacerbates mode collapse by discouraging exploration of distant modes (Fig. 1C). Instead, we propose a more principled MH-based divergence control [62]. After each 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.*, $x^l = \tilde{x}^l$), otherwise the old position is retained (*i.e.*, $x^l = x^{l-1}$). We compute α^l_θ efficiently by leveraging $Tr(\nabla_{x^l}\phi_\theta(x^l))$: $\log \alpha^l_\theta = \min[0, \log \bar{p}(\tilde{x}^l) - \log \bar{p}(x^{l-1}) + \epsilon_{\theta_3} Tr(\nabla_{x^l}\phi_\theta(x^l))]$ (proof in App. 8). **MET-SVGD** is an MH with an efficient SVGD-based proposal distribution. It therefore inherits asymptotic **convergence guarantees** from MH (details in App. 6.8).

3. Experiment

Entropy Estimation on Gaussian (Fig.1A, Fig.1D, Fig.15) **and GMM** (Fig.17, Fig.26) **Targets.** **MET-SVGD** consistently outperforms **P-SVGD**, notably, Fig.1D and Fig.26 show that, while **P-SVGD** and projection-based baselines, *e.g.*, S-SVGD [29] struggle to scale beyond 20-d spaces, **MET-SVGD** achieves high accuracy in up to 100-d. Note that **MET-SVGD** mitigates the vanishing repulsive force (Fig.15c) identified as the root cause of SVGD's poor scalability in [6]. **Learning EBMs.** 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. 10. 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** [61] (Fig. 27) and CIFAR10 (Fig. 3). For CIFAR10, we report the Frechet Inception Distance (FID) over 5 seeds. In Fig. 3, 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. 47) and results in smoother landscapes (Fig. 44). Yet, experiments with MH diverge. In App. 10, 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. 10. **Max-Entropy RL** qualitative results and implementation details are in App. 11.

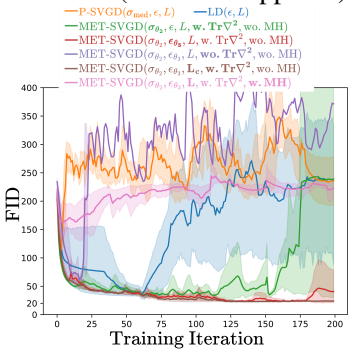


Figure 3: FID on CIFAR10. The modification between two consecutive configs is **bolded**.

4. Conclusion

MET-SVGD is a novel VI approach for entropy estimation from unnormalized density. It bridges the gap between VI, particle-based inference and MCMC; sets a new SOTA for scaling SVGD sampling to high-dimensional and non-smooth densities; and introduces a framework for learning sampler parameters end-to-end and is also a new residual flow model with full rank Jacoian and adaptive number of layers.

References

[1] John Wiley Sons, Ltd, 1992.

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

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

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

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

[6] 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.

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

[8] 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.

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

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

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

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

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

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

[15] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[40] Sudheesh Kumar Kattumannil. On stein's identity and its applications. *Stat. Probab. Lett.*, 2009.

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

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

[43] 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.

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

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

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

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

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

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

[50] Qiang Liu and Dilin Wang. Stein variational gradient descent as moment matching. *NeurIPS*, 2018.

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

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

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

[54] Safa Messaoud, Billel Mokeddem, Zhenghai Xue, Linsey Pang, Bo An, Haipeng Chen, and Sanjay Chawla. S2 ac: Energy-based reinforcement learning with stein soft actor critic. *ICLR*, 2024.

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

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

[57] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *JMLR*, 2021.

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

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

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

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

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

[63] CP Robert. Monte carlo statistical methods, 1999.

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

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

[66] 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.

[67] 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.

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

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

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

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

[72] Jiaxin Shi and Lester Mackey. A finite-particle convergence rate for stein variational gradient descent. *NeurIPS*, 2022.

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

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

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

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

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

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

[79] 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.

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

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

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

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

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

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

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

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

[88] A. Zellner. An introduction to bayesian inference in econometrics. 1971.

[89] Jiankui Zhou and Yue Qiu. Augmented message passing stein variational gradient descent. *arXiv preprint arXiv:2305.10636*, 2023.

[90] Jingwei Zhuo, Chang Liu, Jiaxin Shi, Jun Zhu, Ning Chen, and Bo Zhang. Message passing stein variational gradient descent. In *ICML*. PMLR, 2018.

Supplementary Material

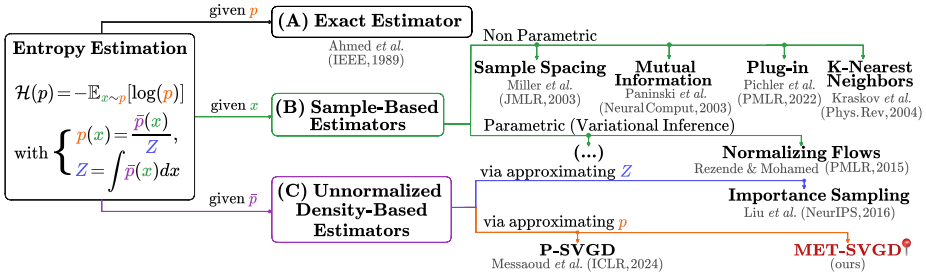


Figure 4: **MET-SVGD** is a new variational inference approach for entropy estimation of distributions known up to a normalization constant. It extends **P-SVGD** [54] to high-dimensional spaces by adressing its key limitations (see Fig. 1).

MET-SVGD is a novel variational inference approach for entropy estimation that overcomes key limitations of **P-SVGD** [54], particularly poor convergence and scalability in high-dimensional spaces (Fig. 1). To achieve this, it introduces: (1) Sufficient condition for global invertibility. (2) Optimized parameter search for improved stability (Sec. 2.2). (3) Metropolis-Hastings augmented SVGD updates to ensure asymptotic convergence (Sec. 2.4). (4) A correction term to the density estimation in **P-SVGD** (Sec. 2.3). **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) [62], particle-based sampling techniques (SVGD) [49], and parametrized variational inference (P-VI) [26], 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. 3
- **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 [29, 53]
- **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. 7) and SVGD (Fig. 6).
- **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. **??**). 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.

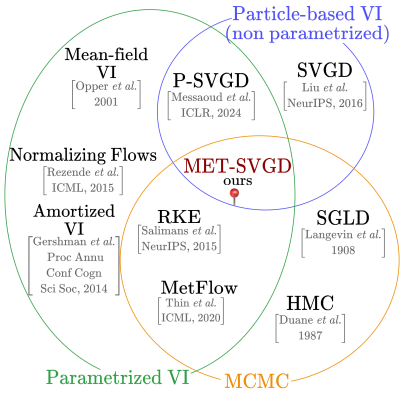


Figure 5: Bridges the gap ...

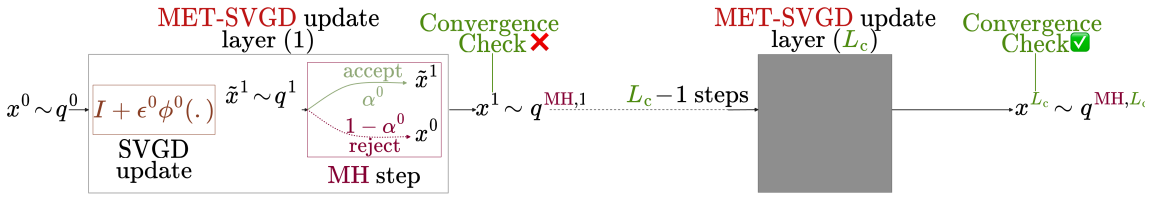


Figure 9: **MET-SVGD** is a normalizing flow model with a full rank Jacobian and an adaptive number of layers.

The rest of the appendix is organized as follows:

- Appendix 5: **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 6: **Related work and Background** on entropy estimation, variational inference, sampling-based variational inference, Normalizing Flows, Metropolis-Hastings, SVGD and the Stein Identity.
- Appendix 7: **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 8 **Derivation of the Metropolis-Hastings augmented entropy**
- Appendix 9: **Additional results on entropy estimation**
- Appendix 10: **Additional results on learning EBMs for image generation**
- Appendix 11: **Additional results on MaxEntr RL**

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: $\epsilon < \sqrt{\mathrm{Tr}(\nabla\phi^l \nabla\phi^{l,\top})}$ (Corollary 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)
Divergence Control	Heuristic: particles truncation beyond 3 std from $q^{(0)}_{\theta_1}$ mean (Eq. 9, P-SVGD paper)	Metropolis-Hastings correction (Section 2.4)
Tr(Hessian) in Entropy	Omitted; invalid for finite particles (Theorem 3.3, P-SVGD paper)	Restored via Hutchinson estimator (Section 2.3)
Kernel Bandwidth σ	Median heuristic: $O(M^2)$	Learned via lightweight GNN (Section 2.2)
Step Size ϵ	Fixed	Learned via lightweight GNN (Section 2.2)
Number of Steps L	Fixed	Adaptive via Stein Identity (Section 2.2)
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), MH correction (Proposition 2.4), and convergence check ; GNN inference adds minor overhead (Section 2.2)
Memory	-	Two small GNNs for σ, ϵ (Section 2.2)
Convergence Guarantee	$L, M \to \infty$	$L \to \infty$
Empirical Performance	Sensitive to hyperparameters (Fig. **??**); mode collapse (Fig. **??**); poor scalability to non-smooth and high-dimensional targets (Fig. **??** and Fig. **??**)	SoTA entropy on G/GMM (Fig. 13 and Fig. 17); better FID, stability in EBMs for image generation (Fig. 3); improved MaxEnt RL returns (Fig. **??**)

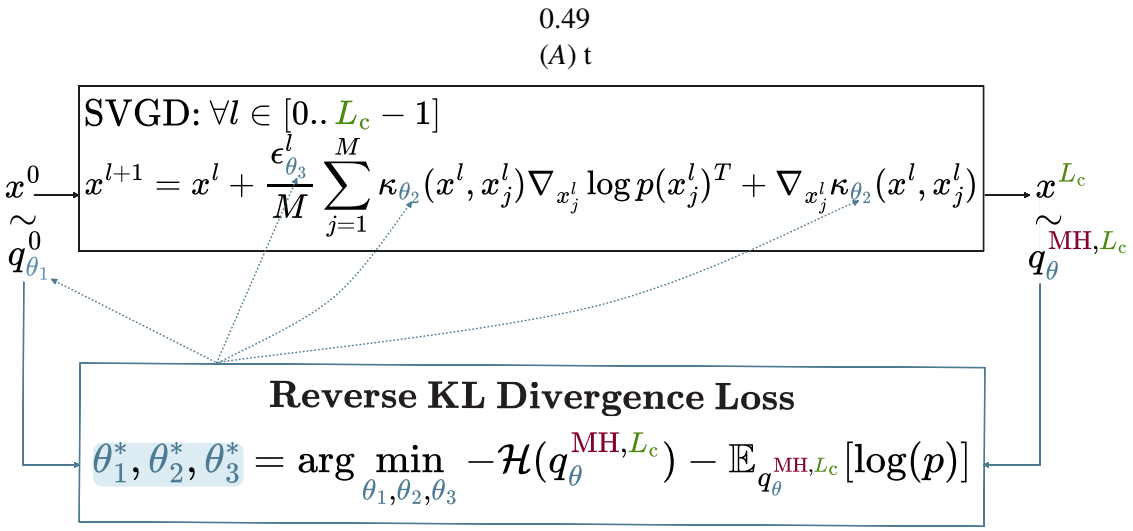


Figure 6: **SVGD**

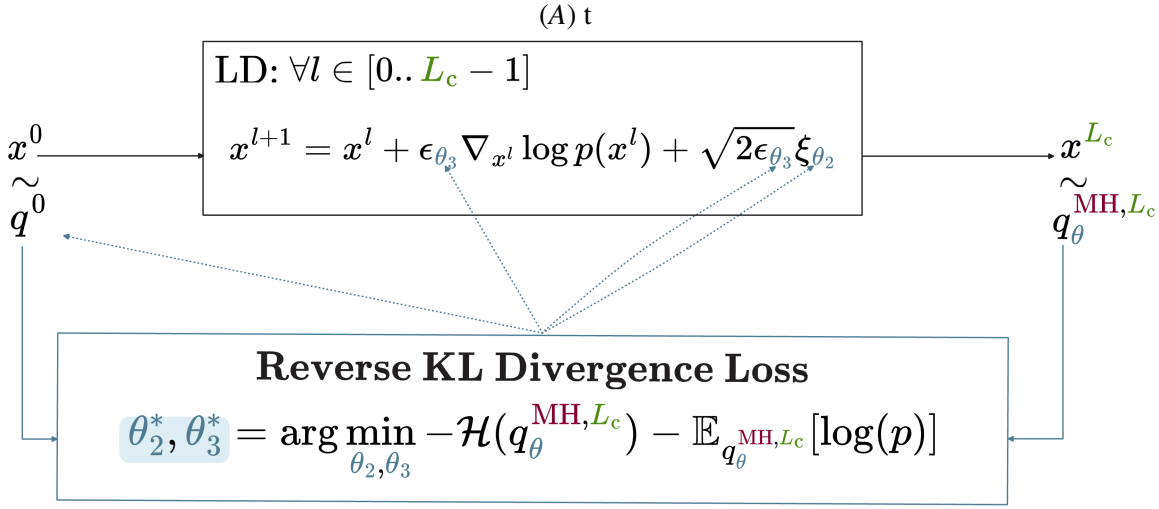


Figure 7: **LD**

Figure 8: **MET-SVGD provides a principled approach to learn sampler parameters** via first computing the particles induced density, then Learning the parameters through KLD minimization.

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 σ_{θ_2} and (v) learning rate deepnet ϵ_{θ_3}.

output : $\theta^* = \{\theta^*_1, \theta^*_2, \theta^*_3\}$.

for Each training iteration **do**
  $l = 0$ % initialize the number of SVGD steps
  $\{x^0_i\}_{i=0}^{M-1} \sim q^0_{\theta_1}$ % sample initial particles from $q^0_{\theta_1}$
  $q^0_{\mathrm{MH}} = q^0_{\theta_1}$ % Initialize q^0_{MH}
  % Run SVGD chain to convergence of si(Eq. 2.3)
  while $\left(l \leq L\right)$ and $\left(\Delta\mathrm{SI}(q_\theta^{\mathrm{MH},l}, p) \leq 0\right)$ **do**
    $\epsilon^l_{\theta_3} = \mathrm{GNN}(\{x^l_i\}_{i=0}^{M-1}; \theta_3)$ % Compute learning rate

5. 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.

5.1. Langevin Dynamics

SGLD [84] 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:

$$x^{l+1} = x^l + \epsilon \nabla_{x^l} \log p(x^l) + \sqrt{2\epsilon}\xi. \tag{5}$$

5.2. Change of Variable Formula (CVF)

We first introduce the concept of an Invertible Function.

According to [46], the following holds: if $F : Z \to X$ is an invertible function then:

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

5.3. 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 $\forall y \in W$.

5.4. Corollary of Jacobi's Formula

Given an invertible matrix A, the following equality holds:

$$\log(\det A) = \mathit{Tr}(\log A) = \mathit{Tr}\Big(\sum_{k=1}^{\infty} (-1)^{k+1} \frac{(A - I)^k}{k}\Big). \tag{6}$$

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.

5.5. 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 5 (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 (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:*

$$\exists K \in [0, 1[\quad s.t. \quad d(\phi(x), \phi(\tilde{x})) \leq K d(x, \tilde{x}), \quad \forall x, \tilde{x} \in \mathcal{X} \tag{7}$$

Theorem 7 (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*

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

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 $n < m$*

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

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: $\quad y^{i+1} = y^0 - g(y^i)$
3: **end for**
4: **Return** $y^n = g(x)^{-1}$

■

5.6. The Mean Value Theorem

Theorem 8 *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*

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

5.7. Stein Identity ([47])

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.

■

6. Related Work

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

6.1. Differential Entropy

Differential entropy, first introduced by Shannon in his foundational work on information theory [70], has been widely studied in statistics [1, 10, 88]. 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 [74], used in active learning [87]. (ii) In Bayesian Inference, the Maximum Entropy principle ensures the least informative prior [10]. (iii) In Reinforcement learning, it prevents overly deterministic policies by incorporating entropy into the reward function [3, 33]. (iv) Variational inference & generative Models: The entropy appears in ELBO [42] for posterior approximation and mitigates mode collapse in GANs and VAEs [4, 9].

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. [23] and later investigated by others using kernel density estimator [32, 37, 55, 59], histogram estimator [30, 32] and field-theoretic approaches [17]. Early approaches leverage kernels that capture pairwise distances between the particles. For instance, Parzen-Rosenblatt estimator [58, 64]: $\hat{p}(x) = \frac{1}{w^p n}\sum_{i=1}^M \kappa\Big(\frac{x-x_i}{w}\Big)$, where w denotes the bandwidth and κ is a kernel density. The resulting entropy estimator was analyzed by Ahmad and Lin [2]. Schraudolph [69] 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 [59] 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 [15] defined as: $\hat{p}^{\text{POIS}}(x) = k \sum_{i=0}^{\infty} \left(F_n(\frac{i+1}{k}) - F_n(\frac{i}{k}) \right) e^{-kx} \frac{(kx)^i}{i!}$, 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 [69, 83], providing a foundation for modern differentiable approaches.

- *Sample-spacing Estimates* use distances between ordered samples (*e.g.*, Vasicek estimator [81]). Sample spacing methods rely on the spacing of sorted samples and was initiated by Vasicek [81]: $H^{Vasicek}(p) \approx -\frac{1}{M} \sum_{i=1}^{M} \log \left(\frac{n}{2m} \left(x_{i+1} - x_i \right) \right)$, 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 [45]. This method estimates entropy using distances to the k-th nearest neighbor in the sample space [45], *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 $p(x)$ [41]. The entropy is computed as [41]: $\mathcal{H}(p) \approx -\mathbb{E}_{q(x)}[\log q(x)]$, 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 [60].

- *Mutual Information (MI) Estimators*: Approximate entropy indirectly via MI relationships, *i.e.*, $I(x, y) = \mathcal{H}(p_x) + \mathcal{H}(p_y) - \mathcal{H}(p_{x,y})$, [9], 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 [24]: $D_{\text{KL}}(p\|q) = \sup_{T \in \mathcal{T}} \left(\mathbb{E}_p[T(x)] - \log \mathbb{E}_q[e^{T(x)}] \right)$, 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 [77]. 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. [5] 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: $p(x) = p_1(x_1)\dots p_d(x_d)c\Big(F_1(x_1),\dots,F_d(x_d)\Big)$, where $c(u_1,\dots,u_d)$ 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. [39] 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	$-\frac{1}{M}\sum_{i=1}^{M}\log\left\{\frac{n}{2m}\Big(x_{i+1}-x_i\Big)\right\}$	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

6.2. Variational Inference

Variational Inference (VI) [26] approximates a target distribution $p(x) = \bar{p}(x)/Z$, known up to the normalizing constant Z, via a simpler-to-sample-from distribution $q^*(x)$ from a predefined family $\boldsymbol{Q} = \{q\}$, by maximizing the KL-divergence *i.e.*, $q^* = \arg\max_{q\in\boldsymbol{Q}} D_{\text{KL}}(p\|q)$. The choice of $\boldsymbol{Q}$ significantly impacts performance; more expressive families yield better approximations. At

convergence, the target entropy can be estimated as $\mathcal{H}(p) = -\mathbb{E}_{q^*}[\log q^*]$. While VI is scalable, it may not achieve the optimal q^* due to, either the limited expressivity of $\boldsymbol{Q}$, *i.e.*, easy to sample from distributions are usually over-simplistic (*e.g.*, Gaussians), or optimization challenges (*e.g.*, mode collapse in Normalizing Flows [57]).

6.3. 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 [27, 68] 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 [34] 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. [14] propose a deterministic Hamiltonian MCMC by removing resampling and the accept-reject step. However, this sacrifices MCMC guarantees, (iii) Thin et al. [79] 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. 5.2). 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 [21], 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 [43]. Our derived log-likelihood is more intuitive as it depends on the SVGD dynamics.

6.4. 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.5.2). 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:

$$p_x(x) = p_z(F^{-1}(x)) \left| \det \nabla_x g^{-1}(x) \right| \tag{8}$$

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 [11]. 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) [28, 36] and was achieved via partitioning units in each layer into two groups and defining coupling functions as:

$$y^A = x^A + F(x^B) y^B = x^B + G(y^A), \tag{9}$$

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. [7] proved the following statement:

Proposition 9 *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 5.6, 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. 7.1) . Controlling the Lipschitz constant of a neural network is not trivial. Note, that regularizing the spectral norm of the Jacobian of ϕ [75] only reduces it locally and does not guarantee the above condition. Instead, Jacobsen et al. [36] 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. 5.2). A different approach is proposed to approximate the log-det term:

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

Where (i) is obtained using the matrix identity result $\log\det(A) = \operatorname{Tr}(\log(A))$ for non-singular $A \in \mathbb{R}^{d \times d}$ [85] 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 [16]. In a model they called a Residual flow [16], 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) [31]:

$$\frac{d}{dt}\mathbf{x}(t) = F(\mathbf{x}(t), \theta(t)), \tag{10}$$

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

$$\mathbf{x}_{n+1} - \mathbf{x}_n = \varepsilon F(\mathbf{x}_n, \theta_n), \tag{11}$$

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

6.5. Stein Variational Gradient Descent

In the following, we provide an explanation of SVGD, 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.

Stein Variational Gradient Descent (SVGD) [49] is a sampling algorithm with update rule given by Eq. 1. Traditionally, an RBF kernel is used, with its bandwidth σ set via the median heuristic: $\sigma_{\text{med}} = \text{median}\{\|x_i^l - x_j^l\|\}_{i,j=1}^M / \log M$. Bandwidth σ determines the influence of neighboring particles $\{x_j^l\}$ on the update of each particle x_i^l: larger values lead to broader neighborhoods, while setting $\sigma = 0$ decouples the particles, making their updates independent (Fig. 10 in App. 6.5). SVGD has several advantages compared to other *approximate inference* approaches: unlike classical variational inference (VI) methods, SVGD can sample from arbitrary complex distributions under smoothness assumptions [82]. Compared to Markov Chain Monte Carlo (MCMC) methods [18], SVGD is more particle efficient and its convergence can be easily checked using the Stein Identity [40]. However, *SVGD convergence* is only proved under certain conditions, such as sub-Gaussian targets [72] or infinite particles [50, 67]. Additionally, SVGD suffers from *poor scalability in high-dimensional spaces* due to diminishing repulsive forces [90]. To address this, existing solutions are based on dimensionality reduction via projections into low-dimensional manifolds [29, 53], this however leads to inflated variance (*e.g.*, S-SVGD [29]) or impractical hyperparameters (*e.g.*, GSVGD [53]). Alternatively, [89, 90] propose using local kernels based on the Markov blanket, but this requires prior knowledge of the target's probabilistic graphical model. In this paper, we extend [54] work on deriving a closed-form expression of q^l enabling better scalability.

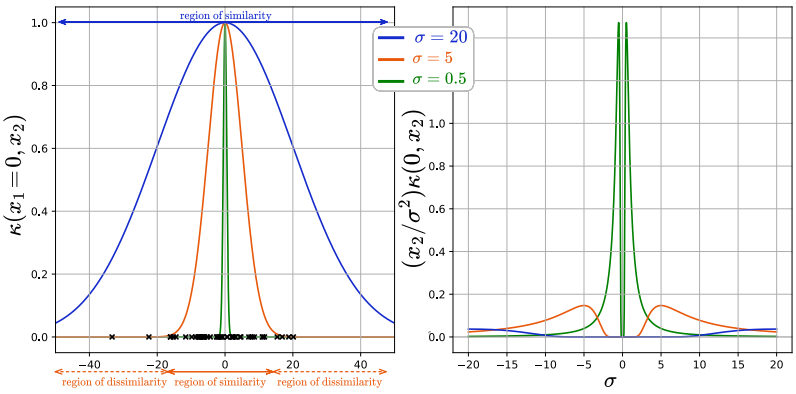


Figure 10: (a) Regions of similarity/dissimilarity for an RBF $\kappa(x_1, x_2)$ evaluated at $x_1 = 0$. (b) Repulsion term in the SVGD update as a function of σ.

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. 10 which also means that the particle update is impacted by its neighbors' gradients (b)).

Setting σ in the SVGD update rule;

$$x^{l+1} = x^l + \epsilon \mathbb{E}_{x_j^l}\Big[\underbrace{\kappa(x^l, x_j^l)\nabla_{x_j^l} \log p(x_j^l)}_{\text{drift term}} + \underbrace{\frac{(x^l - x_j^l)}{\sigma^2}\kappa(x^l, x_j^l)}_{\text{repulsion term}}\Big] \tag{12}$$

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. 10 (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. 2.2).

SVGD Derivation. [49] 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. 5.2):

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

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 $\arg\max_{\phi\in\mathcal{F}} \nabla_\epsilon D_{KL}(q_{[F]}||p) = \mathbb{E}_q[\mathrm{Tr}(\mathcal{A}_p\phi)]$. The optimal perturbation direction ϕ^* is, hence, the one that maximizes the Stein Discrepancy [51]:

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

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, [48] proved that SVGD converges (weakly) to p in KSD. [44, 67, 78] refined these results with path-independent constants, weaker smoothness conditions, and explicit rates of convergence. [25] provides conditions for exponential convergence. For the finite particles case, [48] 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. [44] 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.[52] show that SVGD with finite particles achieves linear convergence in KL divergence under a very limited setting where the target distribution is Gaussian. [73] 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.

6.6. Parametrized-SVGD

Parametrized SVGD (P-SVGD) [54] is a VI approach for entropy estimation from unnormalized densities. Under *invertibility assumption* of the SVGD update rule (Eq. 1), it computes the density of the SVGD particles $q^L(x^L)$ by sequentially applying the Change of Variable formula (CVF) [22] over L steps under an invertibility condition derived from the implicit function theorem (App. 5.3): $\log q^{l+1}(x^{l+1}) = \log q^l(x^l) - \log|\det(I+\epsilon\nabla_{x^l}\phi(x^l))|$. 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. 5.4) and leading to Eq. 3.

6.7. 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 [62]. 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 [80], 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

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

6.8. 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 [63]. 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 [13, 38, 65, 82]. 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 [80]. 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:

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

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

7. SVGD Density Derivation

Theorem 10 *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:*

$$\log q^L(x^L) = \log q^0(x^0) - \epsilon \sum_{l=0}^{L-1} \mathrm{Tr}(\nabla_{x^l}\phi(x^l)) + \mathcal{O}(\epsilon^2) \tag{13}$$

Proof Based on the change of variable formula (5.2), 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 $\log\left|\det\left(I + \epsilon\nabla_{x^l}\phi(x^l)\right)\right|$ 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 7.3, that's satisfied by the invertibility assumption (Sec.7.1) and derive the expression of $Tr(\nabla\phi)$ for the RBF, Bilinear and DKEF Kernels (Sec 7.5). ■

7.1. Sufficient Condition For $x + \epsilon\phi(x)$ Invertibility (Prop. 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 $\text{Lip}(\phi^l)$ the Lipschitz constant of the velocity ϕ^l at step l. f is invertible if $\epsilon\,\text{Lip}(\phi^l) < 1$, 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 $\epsilon\,\text{Lip}(\phi^l) < 1$. Since f is a composition of f^l ($l \in [1 \cdots L[$), we conclude that f is invertible. ■

7.2. A sufficient condition for invertibility check - an upper bound (Corr. 3)

Corollary 3.3. *The distribution induced by the SVGD update (Eq. 1) using an RBF kernel is given by Eq. 3 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. 7.1)

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 $\|\nabla\phi\|_2 \leq \sqrt{\text{Tr}(\nabla\phi\nabla\phi^T)}$. ■

7.3. A sufficient condition for log-det approximation (Prop. 2)

Proposition 3.2(Condition for log-det Approximation) *Let* $\phi^l : \mathbb{R}^d \to \mathbb{R}^d$, $\log|\det(I + \epsilon\nabla\phi^l)| = \epsilon\mathrm{Tr}(\nabla\phi^l)$ *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. 5.4) and get

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

In practice, since this bound is informal, [54] 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

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

(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 $\epsilon\nabla_{x^l}\phi(x^l)$:

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

(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 $\log\left|\det\left(I+\epsilon\nabla_{x^l}\phi(x^l)\right)\right| = \epsilon\mathrm{Tr}(\nabla_{x^l}\phi(x^l))$ holds exactly.

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

Even though the condition $\epsilon < \frac{\alpha}{|\lambda_{\max}|}$ is more exact than the one derived by [54], it's still impractical as it requires computing the Jaccobian.

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

Corollary 11 *Following [86], Let A be an $d \times d$ complex matrix, and let A^* be the Hermitian of A:*

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

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:

Term (1) $= \mathrm{Tr}(A_i^TA_i)$

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

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

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

Adding these sub-terms together

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

$$
\textbf{Term ②} = \mathrm{Tr}(B_i^T B_i)
$$

$$= \mathrm{Tr}\left(\frac{1}{M^2}\nabla_{x_i^l} s_p(x_i^l)\left(\nabla_{x_i^l} s_p(x_i^l)\right)^T\right)$$
$$= \frac{1}{VM^2}\sum_{t=1}^{V} v_t^T \nabla_{x_i^l} s_p(x_i^l)\left(\nabla_{x_i^l} s_p(x_i^l)\right)^T v_t$$
$$= \frac{1}{VM^2}\sum_{t=1}^{V} \left|\left|\nabla_{x_i^l}\left(v_t^T s_p(x_i^l)\right)\right|\right|^2$$

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

Using Hutchinson Trace Estimation [35]

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

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

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

■

7.5. 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

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

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:

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

Term ②:

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

Term ③: Using Hutchinson Trace Estimation [35]

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

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.

■

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

[52] 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:

$$\begin{cases} \mu^{l+1} = \mu^l + \epsilon^l \Big[\big(I - (\Sigma^l + \mu^l \mu^{l^T}) Q^{-1} + \mu^l b^T Q^{-1}\big) \frac{\mu^l}{C} + (b - \mu^l)^T Q^{-1} \Big], \\ \Sigma^{l+1} = \Sigma^l + \epsilon^l \Big[2\frac{\Sigma^l}{C} - \frac{\Sigma^l}{C} Q^{-1}(\Sigma^l + (\mu^l - b)\mu^{l^T}) - (\Sigma^l + \mu^l(\mu^l - b)^T) Q^{-1} \frac{\Sigma^l}{C} \Big]. \end{cases} \tag{14}$$

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:

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

Hence, substituting these into the dynamics we obtain:

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

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

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

Knowing that

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

Taking into consideration (i) and (ii)

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

■

In Fig. 11, we empirically verify that SVGD intermediate distributions coincide with the derived Gaussians.

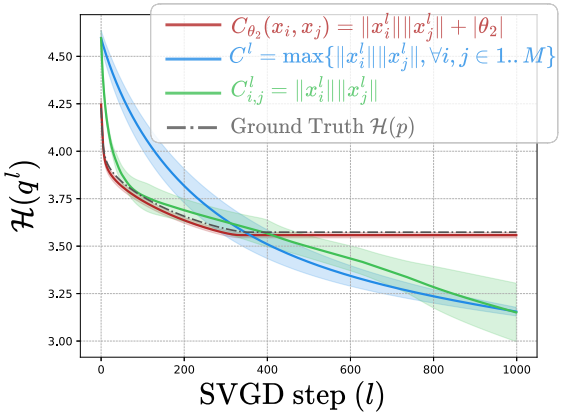


Figure 11: Bilinear kernel. q_θ^l coincides with theoretically derived intermediate distributions by Liu et al. [52].

7.7. 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:

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

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:

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

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

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. ■

7.8. 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 5.2)

(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:

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

■

8. Metropolis Hastings augmented entropy

In the following, we provide derivations for (1) the acceptance probability (Prop.**??**), (2) convergence check (Eq. 4) and (3) MH-augmented SVGD density (Eq. 4).

8.1. Acceptance Probability (Prop.??)

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:

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

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

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

■

8.2. Motivation: learning the SVGD learning rate (Sec. ??-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 4^{th}-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:

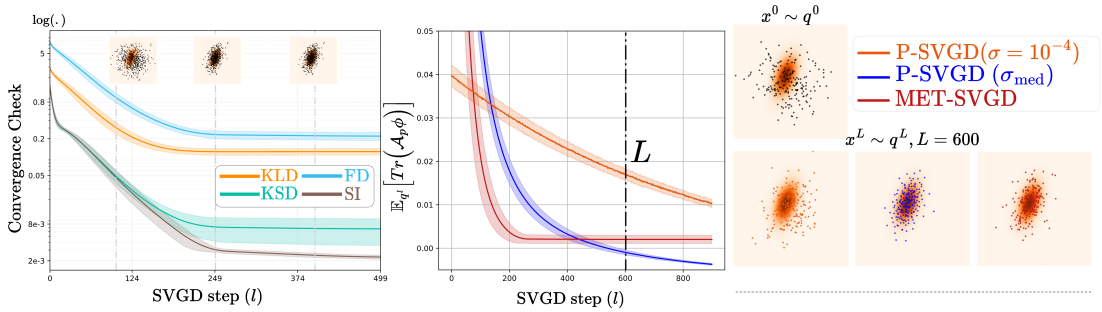
$$\exp\left(-\frac{\|x^{L_c}-x_j^{L_c}\|^2}{\sigma^2}\right) \approx 1 - \frac{\|x^{L_c}-x_j^{L_c}\|^2}{\sigma^2} + \frac{\|x^{L_c}-x_j^{L_c}\|^4}{2\sigma^4}$$

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. 12, we show that SI converges to 0 under different sampler configs of the experiments.

8.3. Convergence Check (Eq. 4)



(a) KSD, KLD, FD, SI at convergence (b) SI across SVGD steps (i)Quantitative (ii)Qualitative

Figure 12: (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}, $\sigma = 10^{-4}$).

8.3.1. DERIVATION OF THE STEIN IDENTITY

The following is a proof of proposition 4:

Proof The Stein Identity is the square of the Kernelized Stein Discrepancy [51]:

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

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:

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

And obtain

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

■

8.3.2. 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 13, which circumvents the intractability issue encountered due to the score of $q^l, \forall l \in [0 \cdots L[$:

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

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

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

■

8.4. MH-augmented SVGD Density (Eq 4)

Proposition 12 *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:

$$q_\theta^{\text{MH},l} = \alpha^{l-1}_{\theta_{2,3}} q_\theta^{\text{MH},l-1}(x^{l-1}) \Big| \det(I + \epsilon \nabla_{x^{l-1}} \phi(x^{l-1})) \Big|^{-1} + (1 - \alpha^{l-1}_{\theta_{2,3}}) q_\theta^{\text{MH},l-1} \tag{15}$$

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

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

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

Therefore:

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

Case $l > 1$: We assume

$$q_\theta^{\text{MH},l}(x^l) = q_\theta^l(x^l)\prod_{l=1}^{l}\alpha_{\theta_{2,3}}^l + \sum_{a_{1:l}\neq 1} q_\theta^{\text{MH},l}(x^l, a_{1:l}), \quad \forall l \in]1, L_c] \tag{16}$$

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

We know that:

$$q_\theta^{\text{MH},L_c+1}(x^{L_c+1}) = \alpha_{\theta_{2,3}}^{L_c+1} q_\theta^{\text{MH},L_c}(x^{L_c})\left|\det(I + \epsilon\nabla_{x^{L_c}}\phi(x^{L_c}))\right| + (1-\alpha^{L_c})q_\theta^{\text{MH},L_c}(x^{L_c}) \tag{17}$$

We plug Eq. 16 into Eq. 17:

$$q_\theta^{\text{MH},L_c+1}(x^{L_c+1}) = q_\theta^{\text{MH},L_c}(x^{L_c}) \times \left[\alpha_{\theta_{2,3}}^{L_c+1}\left|\det(I + \epsilon\nabla_{x^{L_c}}\phi(x^{L_c}))\right| + (1-\alpha^{L_c})\right]$$

$$= \left[q_\theta^{L_c}(x^{L_c})\prod_{l=1}^{L_c}\alpha_{\theta_{2,3}}^l + \sum_{a_{1..L_c}\neq 1} q_\theta^{\text{MH},L_c}(x^{L_c}, a_{1..L_c})\right] \times \left[\alpha_{\theta_{2,3}}^{L_c+1}\left|\det(I + \epsilon\nabla_{x^{L_c}}\phi(x^{L_c}))\right| + (1-\alpha^{L}\right.$$

$$= \underbrace{q_\theta^{L_c}(x^{L_c})\prod_{l=1}^{L_c+1}\alpha_{\theta_{2,3}}^l\left|\det(I + \epsilon\nabla_{x^{L_c}}\phi(x^{L_c}))\right|}_{q_\theta^{L_c+1}(x^{L_c+1})\prod_{l=1}^{L_c+1}\alpha_{\theta_{2,3}}^l} + \underbrace{q_\theta^{L_c}(x^{L_c})\prod_{l=1}^{L_c}\alpha_{\theta_{2,3}}^l(1-\alpha^{L_c})}_{q_\theta^{\text{MH},L_c+1}(x^{L_c+1}, a_{1..L_c}=1, a^{L_c+1}=1)}$$

$$+ \underbrace{\sum_{a_{1..L_c}\neq 1} q_\theta^{\text{MH},L_c}(x^{L_c}, a_{1..L_c})\alpha_{\theta_{2,3}}^{L_c+1}\left|\det(I + \epsilon\nabla_{x^{L_c}}\phi(x^{L_c}))\right|}_{q_\theta^{\text{MH},L_c+1}(x^{L_c+1}, a_{1..L_c}\neq 0, a^{L_c+1}=1)} + \underbrace{\sum_{a_{1..L_c}\neq 1} q_\theta^{\text{MH},L_c}(x^{L_c}, a_{1..L_c})(1-\alpha}_{q_\theta^{\text{MH},L_c+1}(x^{L_c+1}, a_{1..L_c}\neq 0, a^{L_c+1}=0)}$$

■

9. Additional Results on Entropy Estimation

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

9.1. Optimized SVGD parameters

9.1.1. KERNEL BANDWIDTH

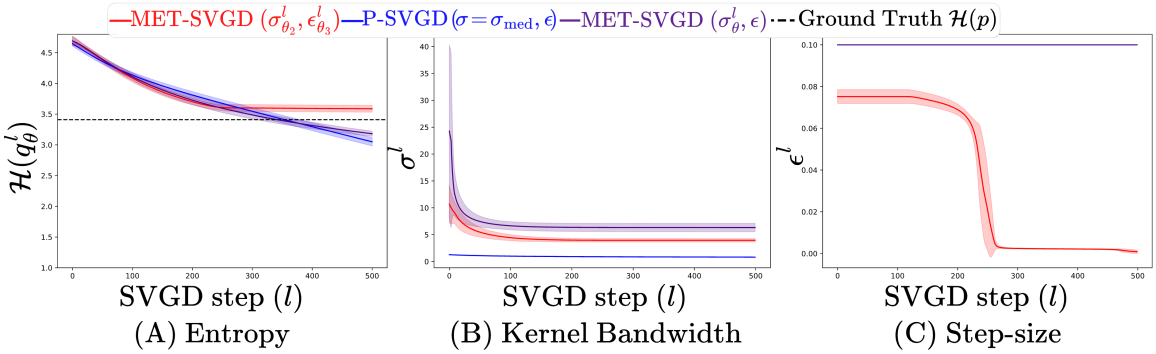


Figure 13: (A) Entropy, (B) RBF kernel bandwidth, and (C) step-size across SVGD steps. Target is the Gaussian target from Fig. **??**.

9.2. Gaussian Targets

Implementation Details for Fig. **??** are reported in Tab. 4.

Algorithm	Parameter	Value
LD P-SVGD MET-SVGD	Target p	$p = \mathcal{N}([-0.69, 0.8], [[1.13, 0.82], [0.82, 3.39]])$
	Number of Particles M	$M = 200$
	Number of Steps L	$L = 1500$
	Initial Distribution q^0	$\mathcal{N}(0, 6I)$
LD P-SVGD	Learning Rate ϵ	$\epsilon = 0.1$
P-SVGD	Kernel Bandwidth σ	$\sigma \in \{1, 5, \sigma_{\text{med}}\}$
	Kernel Architecture	
	Architecture	$\sigma_{\theta_2} = \text{GNN}(\{x^l_i\}_{i=1}^M; \theta_2)$
	# Layers	3
	Activation	{ReLU, Exponential, Truncate}
MET-SVGD	**Learning Rate Architecture ??**	
	Architecture	$\epsilon^l_{\theta_3} = \min\left(\epsilon^0_{\theta_3}, \epsilon^0_{\theta_3} d^{l/s_{\theta_3}}\right)$
	Initial Learning Rate $\epsilon^0_{\theta_3}$	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	Tesla V100-SXM2-32GB
	RAM	2 GB
	Per-epoch runtime	2.6 seconds

Table 4: Experimental setup for Fig. **??**

Implementation Details (Fig. 11) & (Fig. 12)

Parameter	**Figure 11**	**Figure 12**
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. 11 and 12.

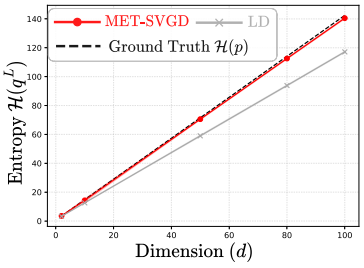


Figure 14: **MET-SVGD** is less sensitive to the *Tr* approximation than LD. Target is a slanted Gaussian (details in App. 6).

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	

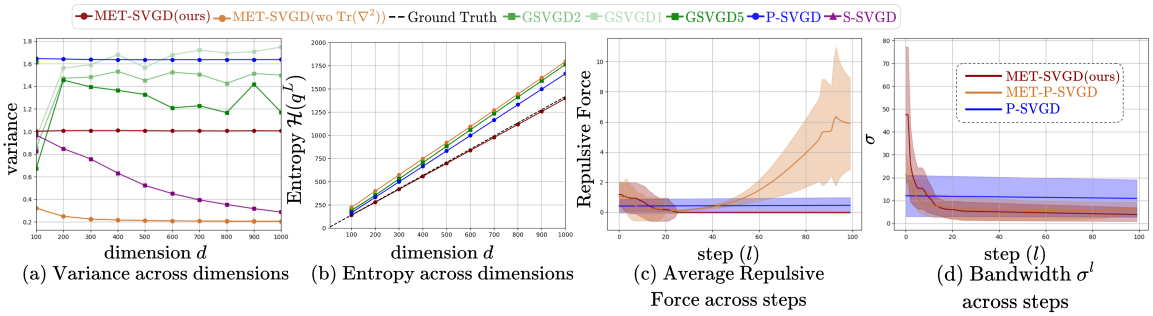
Table 6: Experimental setup for Fig. 14

Implementation Details (Fig. 14) **Langevin Dynamics.** In Fig. 14, 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:

$$\theta^* = \arg\min_\theta \mathbb{E}_{x^L \sim q_\theta^L}[\log q_\theta^L(x^L) - \log p(x^L)] \;\text{ s.t. }\; \epsilon_{\theta_3}^l \leq \epsilon_{\text{UB}}^l, \;\forall l \in [0, L-1]$$

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. 7.5). We show SVGD is less sensitive to this approximation than LD in high dimensions. **Scalability.** (a) Multivariate Gaussian. In Fig. 15, 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

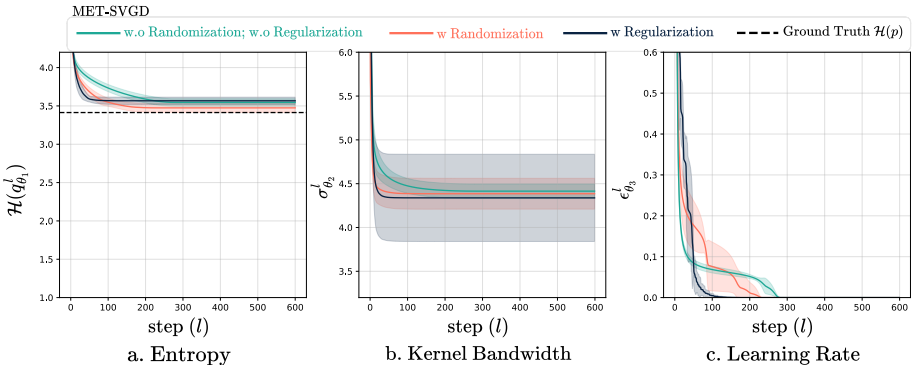
variance issue of SVGD. We show that MET-SVGD outperforms other baselines in high dimensional spaces as measured by the entropy and the variance across dimensions (Fig. 15-a,b). The SVGD repulsive force is large during the first updates, preventing the particles from collapsing, unlike other baselines (Fig. 15-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.



(a) Variance across dimensions (b) Entropy across dimensions (c) Average Repulsive Force across steps (d) Bandwidth σ^l across steps

Figure 15: Scalability Results: **MET-SVGD** achieves higher accuracy on both (a) Entropy Estimation and (b) Variance across dimensions

Accelerating Convergence. In Fig. 16, we show that, for the setup of Fig. 12, convergence can be accelerated either by (1) adding a regularization on the decay rate in the optimization objective (see Sec.2.2) or (2) randomizing the maximum number of steps during training (Eq. 4). We observe a quicker drop in the kernel variance (particles are less correlated initially).



a. Entropy b. Kernel Bandwidth c. Learning Rate

Figure 16: Accelerated convergence via regularization and number of SVGD steps randomization for the same MET-SVGD setup in Tab.6

9.3. Gaussian Mixture Model

9.3.1. 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.

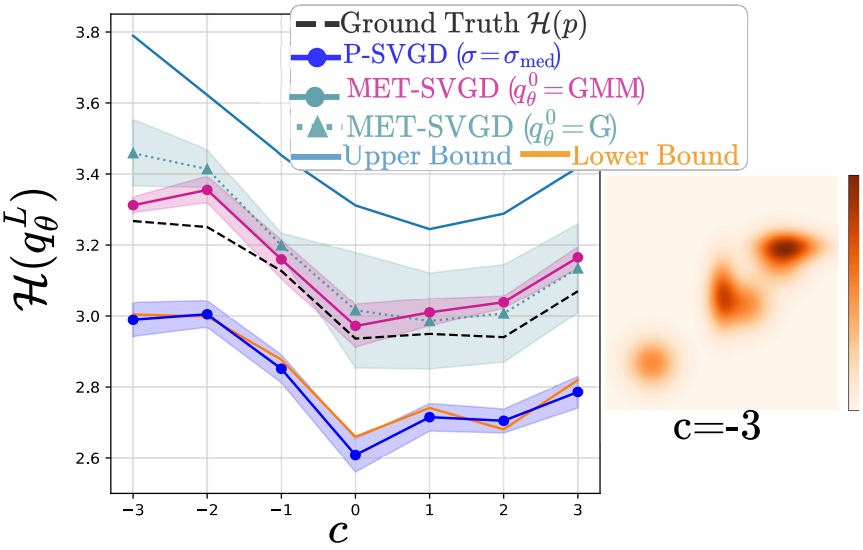


Figure 17: **MET-SVGD** outperforms **P-SVGD** on GMM entropy estimation.

Implementation Details for (Fig. 17) 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. 17

Qualitative results for different c values, as well as the KL-divergence, entropy, kernel Bandwidth and step-size are reported in Fig. 18. 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.

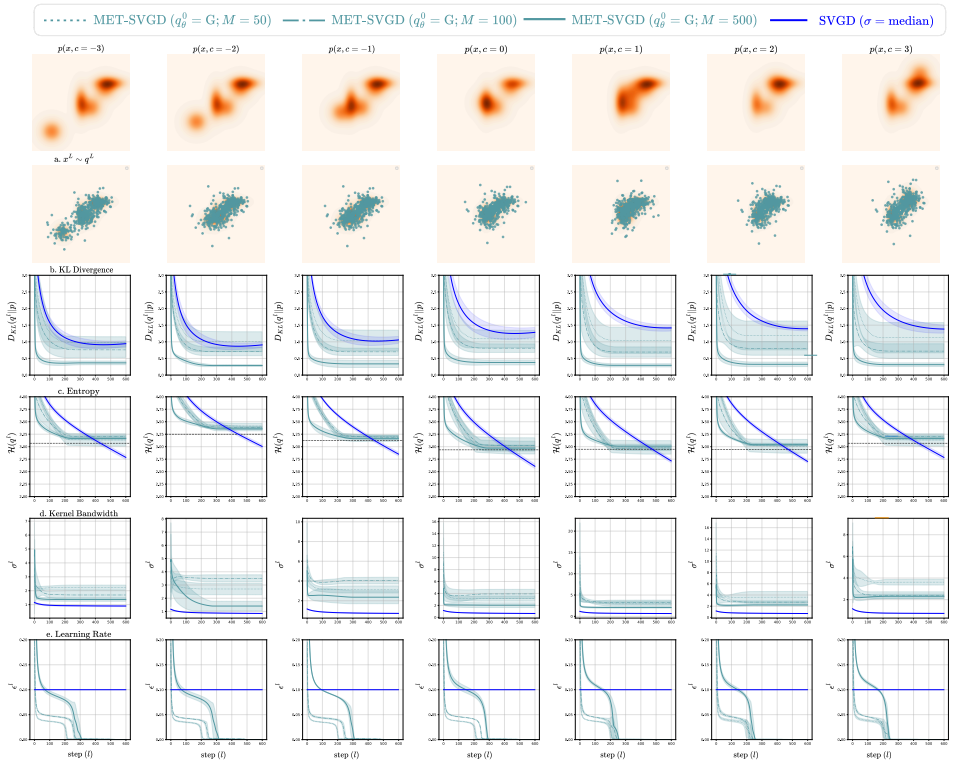


Figure 18: Results on entropy estimation for different c configurations. (a) KLD, (b) Entropy, (c) Learnt kernel bandwidth and (d) SVGD step-size.

Additionally, we show in Fig. 19 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:

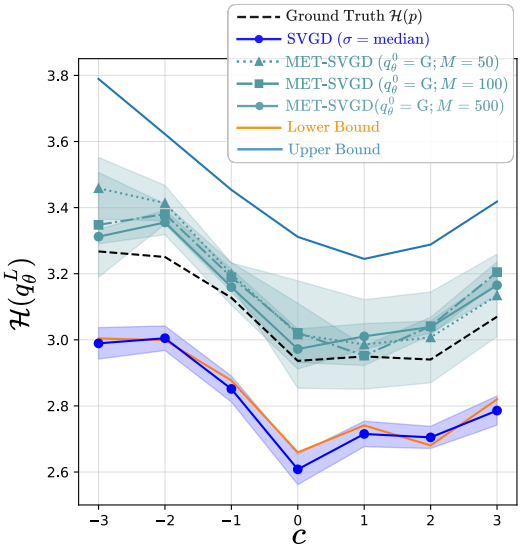


Figure 19: Entropy results on a 2D GMM with a moving component(Fig. 18). **MET-SVGD** significantly outperforms **P-SVGD**. Increasing the number of particles reduces the variances. Results are reported on 5 different seeds.

Next we explain the derivation of the Upper & Lower Bounds for a GMM target as seen in Fig. 19 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 $\mathrm{LB}(x) = -\sum_{i=1}^K \omega_i \log(\sum_{j=1}^K \omega_j \mathcal{N}(\mu_i; \mu_j, C_i + C_j))$.

Proof

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

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:

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

$$= \sum_{i=1}^{L} \omega_i \int_x xx^T \mathcal{N}(x; \mu_i, C_i) dx - \mu\mu^T$$

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:

$$C = \sum_{i=1}^{L} \omega_i (C_i + \mu_i \mu_i^T) - \mu\mu^T$$

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

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

9.3.2. TARGETS WITH DISTANT MODES

Implementation Details (Fig. 20) 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).

Effect of the Initial Distribution. In Fig. 21, 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.

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. 20

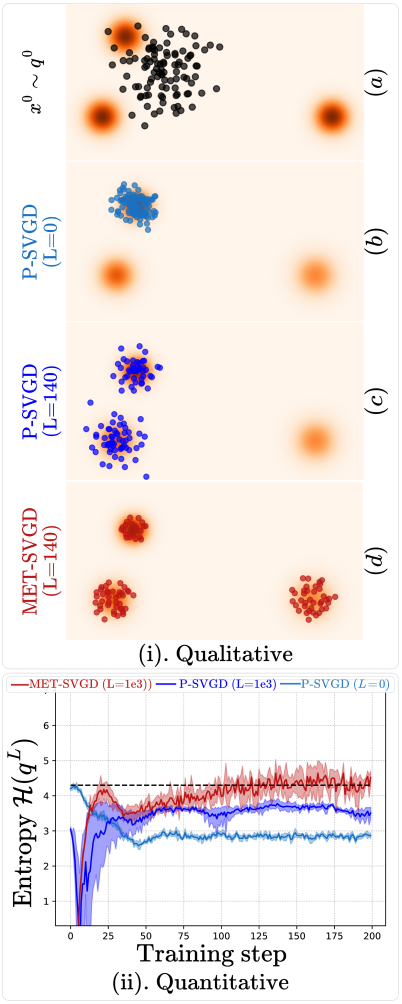
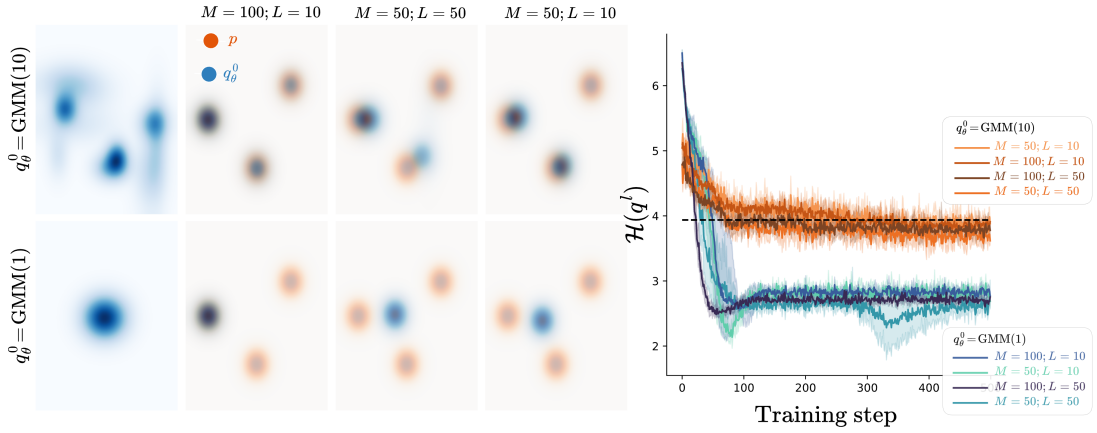


Figure 20: P-SVGD struggles with distributions with distant modes.



a. Learnt initial Distribution b. Entropy across training steps

Figure 21: 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. 21

9.4. 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. 26.A-D). Particles are initialized from $cN(0, I_d)$ and only the first two dimensions need to be learned. In Fig. 22, we show that **MET-SVGD** efficiently recovers the low-dimensional structure.

	Parameter	Value
	Target distribution	$p(x) = \sum_{k=1}^{4} 0.25\mathcal{N}(x, \mu_k, I_d)$
	Initial distribution	$q^0 = \mathcal{N}(0, I)$
Default SVGD parameters	Learning rate	$\epsilon_{\text{P-SVGD}} = 0.1$ $\epsilon_{\text{MET-SVGD}} = \text{nn.Parameter}(0.1)$
	Number of steps	$L = 100$
	Number of particles	$M = 100$
	Kernel variance	$\sigma_{\text{P-SVGD}} = \sqrt{\frac{\text{med}(\|\|x_i - x_j\|\|^2)}{2 \ln M}}$ $\sigma_{\text{MET-SVGD}} = \text{nn.Parameter}(1.0)$
Training	Optimizer	Adam
	Learning rate	10^{-2}
	Epochs	500

Table 10: Experimental configuration for high-dimensional GMM results.

10. Additional Results: Energy Based Models

Proposition 13 (Sec.3) *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:

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

$$\geq -\mathbb{E}_{x\sim p_d}[\log \bar{p}_\phi(x)] + \mathbb{E}_{x\sim q_\phi}[\log \bar{p}_\phi(x)] + \mathcal{H}(q_\phi)$$
$$\geq \mathcal{L}_{\text{ELBO}}(\phi).$$

■

10.1. Synthetic Experiment: Moon Distribution

We evaluate on the Moon dataset [61] with varying smoothness. **Implementation Details (Fig. 27)** are described in Tab. 11. **Performance:** As smoothness decreases, **MET-SVGD** consistently

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

Table 11: Experimental configuration for EBM results.

outperforms all baselines in terms of the MMD score [20], where $\text{MMD}(p,q) = \mathbb{E}_{x,x'\sim p}\left[\kappa(x,x')\right] + \mathbb{E}_{y,y'\sim q}\left[\kappa(y,y')\right] - 2\mathbb{E}_{x\sim p, y\sim q}\left[\kappa(x,y)\right], \quad \text{s.t} \quad \kappa(x,y) = \exp\left(-\|x-y\|^2/2\sigma^2\right)$

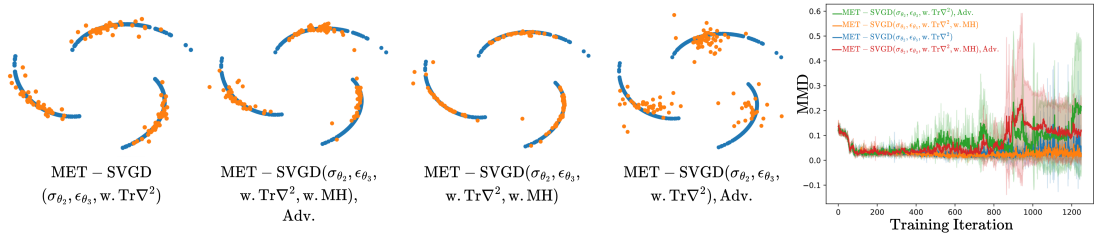


Figure 27: EBM Results. **MET-SVGD** outperforms **P-SVGD** and **LD** on learning EBMs to fit non-smooth data distributions.

10.2. Image generation

Implementation Details (Fig. 38) are reported in Tab. 12. All experiments were conducted on a single NVIDIA A100 80GB with 8GB allocated memory; average runtime was around 6 seconds per

	Parameter	Value
RED$_B OX$	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 covariances
Default SVGD parameters	Learning rate	$\epsilon_{\text{LD}} = \epsilon_{\text{P-SVGD}} = 64$ (this number is divided by m in the SVGD update formula) $\epsilon_{\text{MET-SVGD}} = \text{GNN}(\{x_i^l\}, \{\nabla_{x_i^l} \log p_\theta(x_i^l)\}; \theta_3)$
	Number of steps	$L = 5$ $L_c = 10$
	Number of particles	$M = 64$
	Kernel variance	$\sigma_{\text{LD}} = 0$ $\sigma_{\text{P-SVGD}} = \sqrt{\frac{\text{med}(\|\|x_i - x_j\|\|^2)}{2 \ln M}}$ $\sigma_{\text{MET-SVGD}} = \text{GNN}(\{x_i^l\}; \theta_2)$
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 12: Experimental configuration for EBM results.

iteration (processing one batch of data composed of 64 particles). **Qualitative Results.** In Fig. 38, we visualize generated images sampled from the different models. **FID and Inception Score** In Fig. 38, we report the FID and IS scores for baselines: (1) LD trained with contrastive divergence:

$$\min_\theta \mathcal{L}_{\text{CD}}(\theta) = \min_\theta -\mathbb{E}_{x\sim p_d}\left[f_\theta(x)\right] + \mathbb{E}_{x\sim q}\left[f_\theta(x)\right], \tag{18}$$

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

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

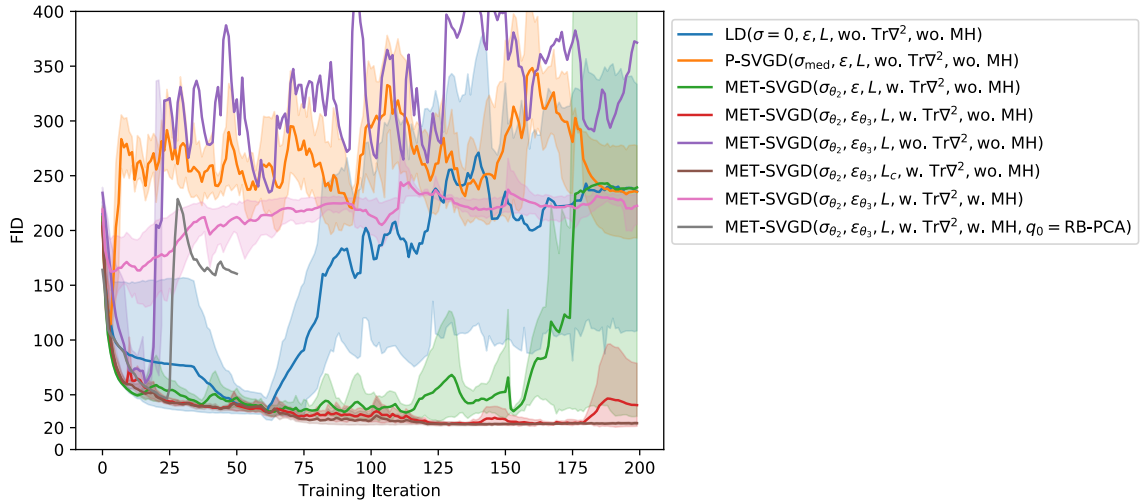
and minimizing the contrastive divergence

$$\min_\theta \mathcal{L}_{\text{CD}}(\theta) = \min_\theta \mathbb{E}_{x\sim q}[\log \bar{p}_\phi(x)] - \mathbb{E}_{x\sim p_d}[\log \bar{p}_\phi(x)] \tag{20}$$

in Fig. 45 and Fig. 41. (3) **MET-SVGD** variants trained adversarially using the ELBO loss for both learning the sampler and the energy:

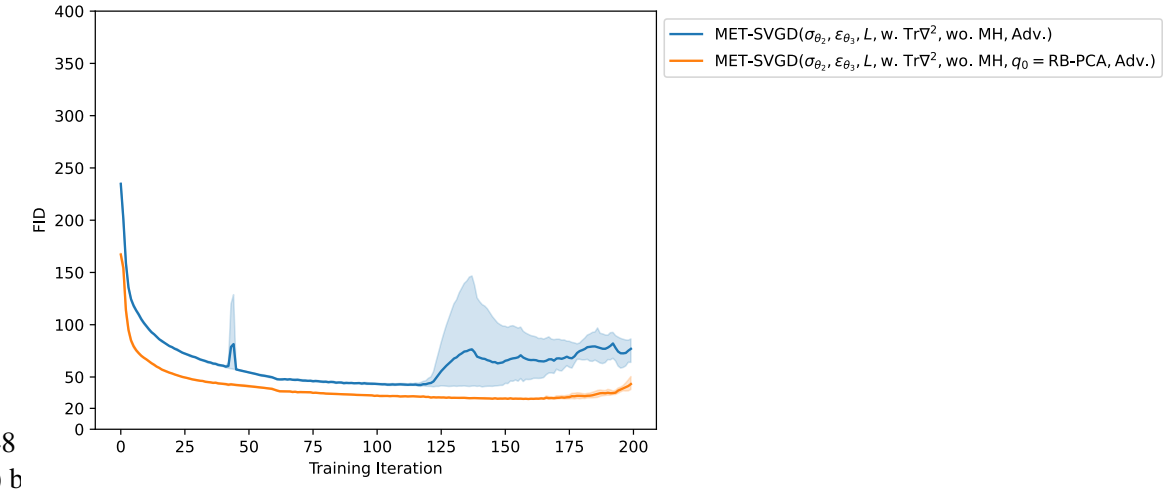
$$\min_\theta \max_\phi \mathcal{L}_{\text{ELBO}}(\theta, \phi) = \min_\theta \max_\phi \mathbb{E}_{x\sim q}[\log \bar{p}_\phi(x)] - \mathbb{E}_{x\sim p_d}[\log \bar{p}_\phi(x)] + \mathcal{H}(q_\phi), \tag{21}$$

in Fig. 46 and Fig. 42. The second setup (Fig. 45, Fig. 41) 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**(σ_{θ_2}), is due to the learning the step-size resulting in smoother energy landscapes (Fig. 44). The third setup (Fig. 46, Fig. 42), 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. 44, 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.



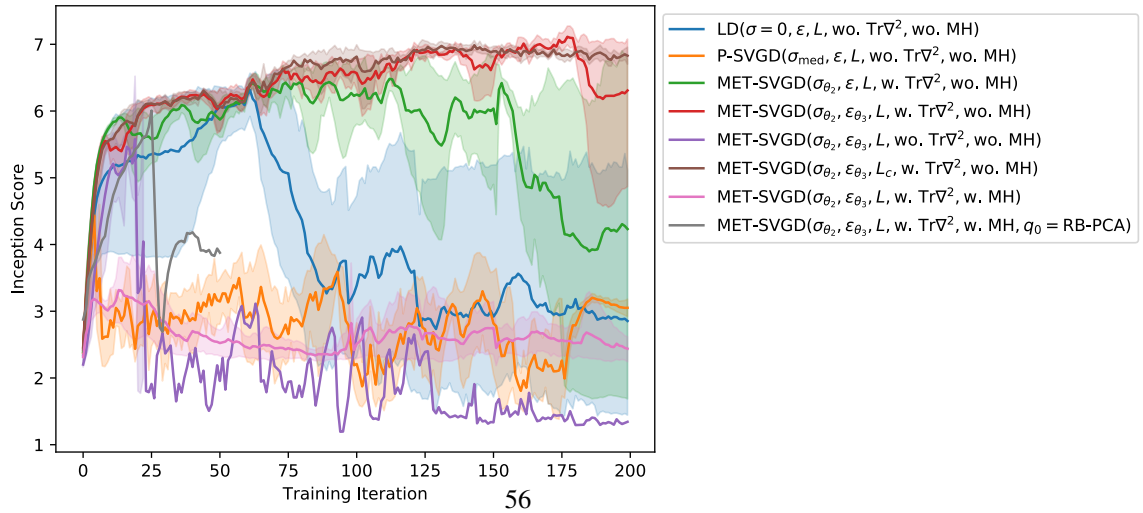
0.48
(*A*) b

Figure 39: FID score $\left(\mathcal{L}_{\text{ELBO}}(\phi), \mathcal{L}_{\text{CD}}(\theta)\right)$



0.48
(*A*) b

Figure 40: FID score $\left(\mathcal{L}_{\text{ELBO}}(\phi, \theta)\right)$



0.48
(*A*) b

Figure 41: IS score $\left(\mathcal{L}_{\text{ELBO}}(\phi), \mathcal{L}_{\text{CD}}(\theta)\right)$

Smoothness. In Fig. 44, 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. $Tr\nabla^2$). The diverging setups are associated with frequently exploding scores.

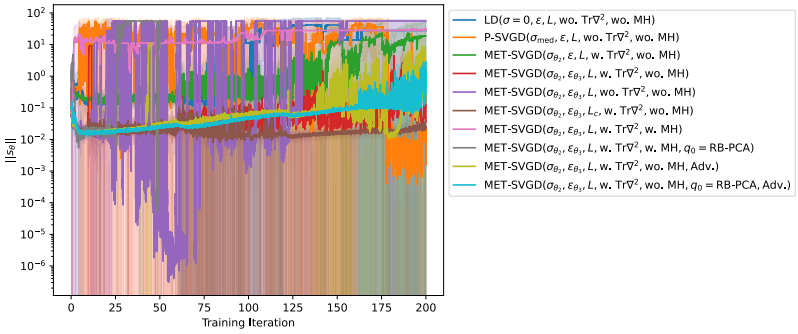


Figure 44: EBM results. The L2-norm of the learnt EBM score $\nabla_x f_\theta(x)$.

Trainable SVGD Hyperparameters. In Fig. 47, 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.

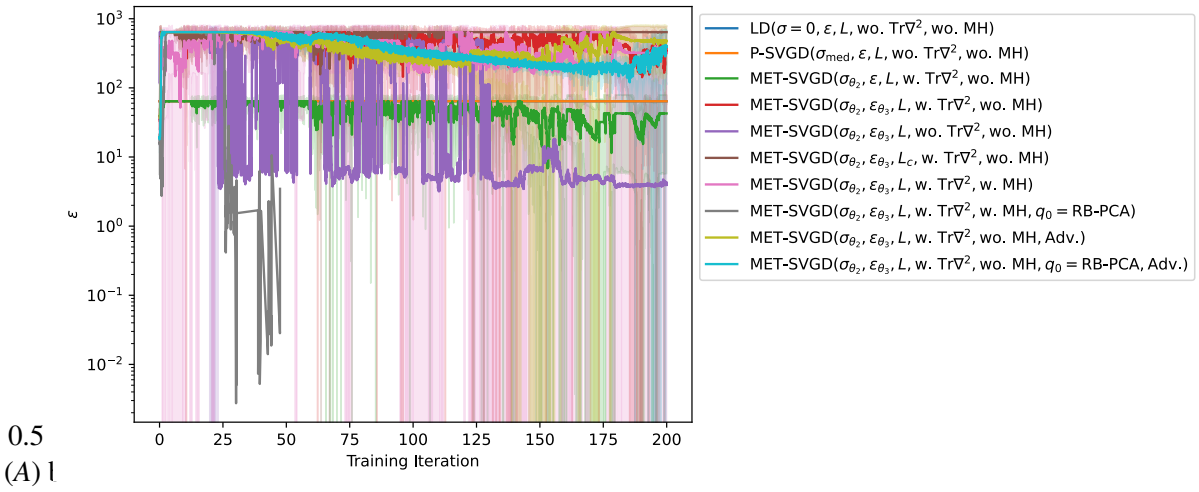


Figure 45: Average SVGD step-size $\epsilon^l_{\theta_3}$ across training iterations.

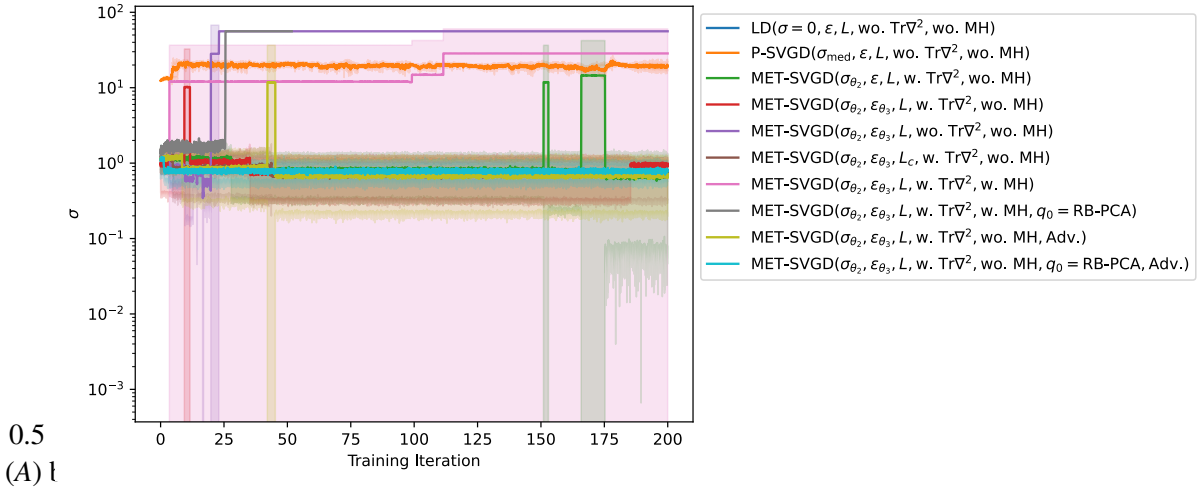


Figure 46: SVGD kernel bandwidth $\sigma^l_{\theta_2}$ across training iterations.

Figure 47: EBM Results. Visualization of the learnt kernel bandwidth and step-size across training iterations.

11. Additional Results: MaxEntr RL

(A) Return IQM (B) Final Return IQM

Figure 48: Return IQM for Walker2d-v2. MH variants yield the best returns.

Implementation Details for (Fig. 48) are reported in Tab. 13

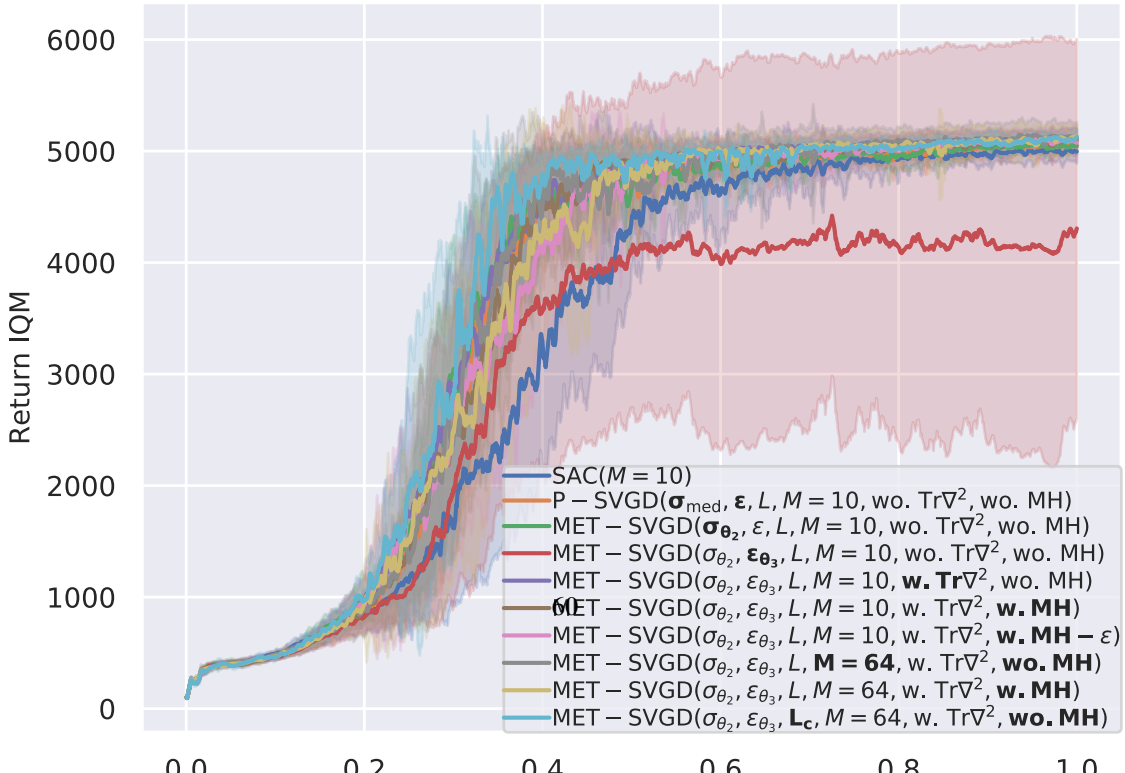
Performance. In Fig. 49 and Fig. 50, we report the Inter Quantile Mean (IQM) return values averaged over 5 runs, where every run is the average of 10 evaluations of the policy. We refer to the approach leveraging **P-SVGD** as proposed by [54] as S^{2}AC, and our approach as S^{2}AC$^+$. We follow the 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. $Tr\nabla^2$). In the Humanoid environment (Fig. 49), 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. 50), 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. 78), which is visible in the high variance of the score norm (Fig. 79). This is the same phenomenon observed in the other EBM experiments.

Table 13: 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 $\lvert\mathcal{D}\rvert$	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	$\sigma^2 = \frac{\sum_{i,j} \lVert a_i - a_j \rVert^2}{4(2 \log m + 1)}$
		$\sigma = \mathrm{GNN}(s_t, \{x_i^l\}, \{\nabla_{x_i^l} \log p(x_i^l)\}; \theta_2)$
	Learning rate	$\epsilon = 0.1$
		$\epsilon = \mathrm{GNN}(s_t, \{x_i^l\}, \{\nabla_{x_i^l} \log p(x_i^l)\}; \theta_3)$

0.48
(A) t



0.48
(A) t

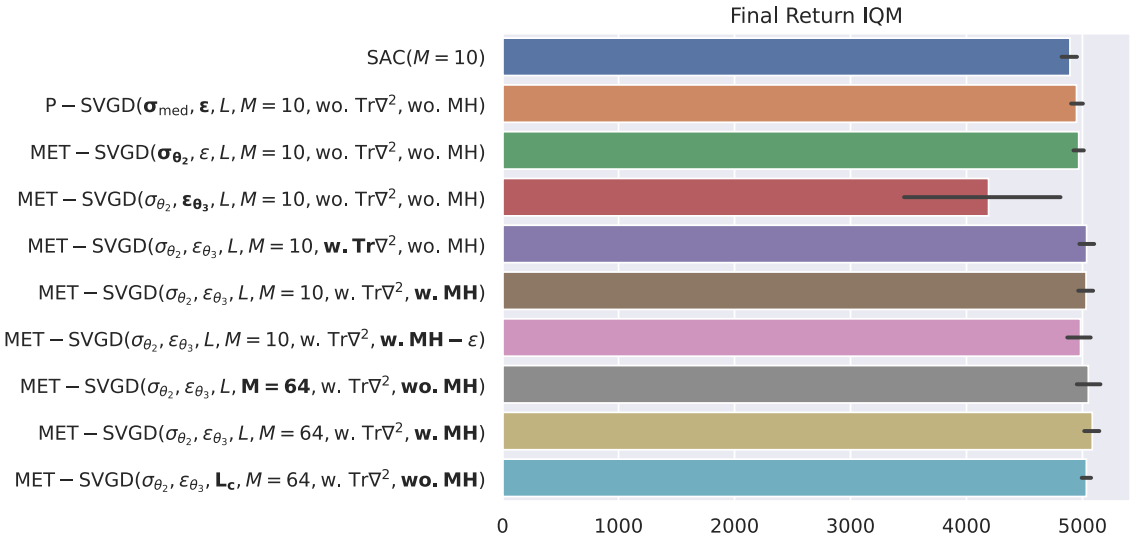


Figure 52: Humanoid's Final IQM Return.

0.48
(A) t

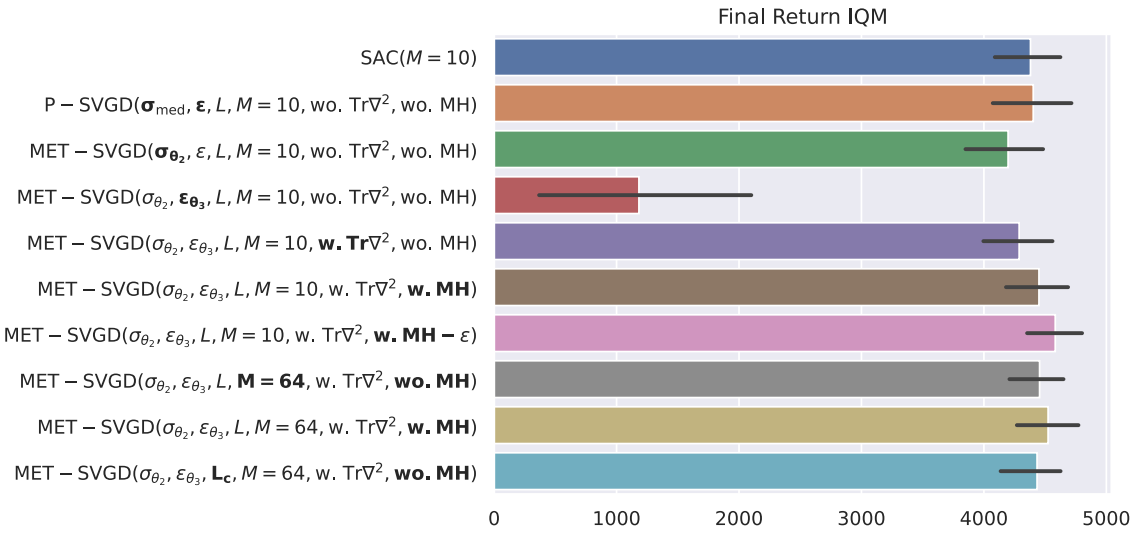


Figure 53: Walker's Final IQM Return.

Figure 54: Final IQM return scores across environments.

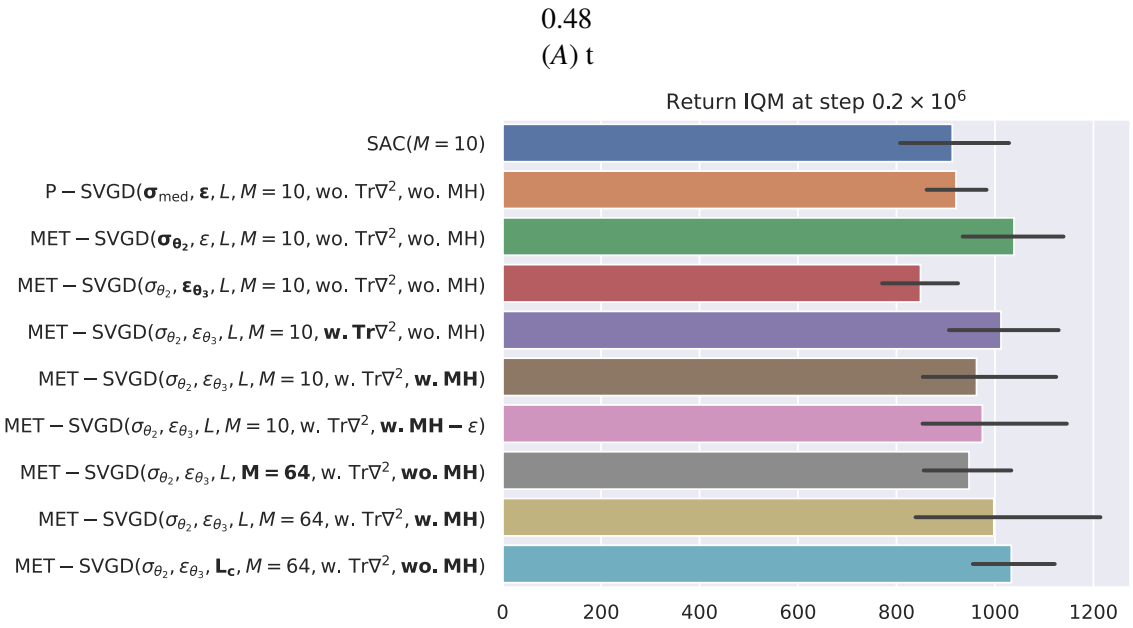


Figure 55: Humanoid's IQM Return at step 0.2×10^6.

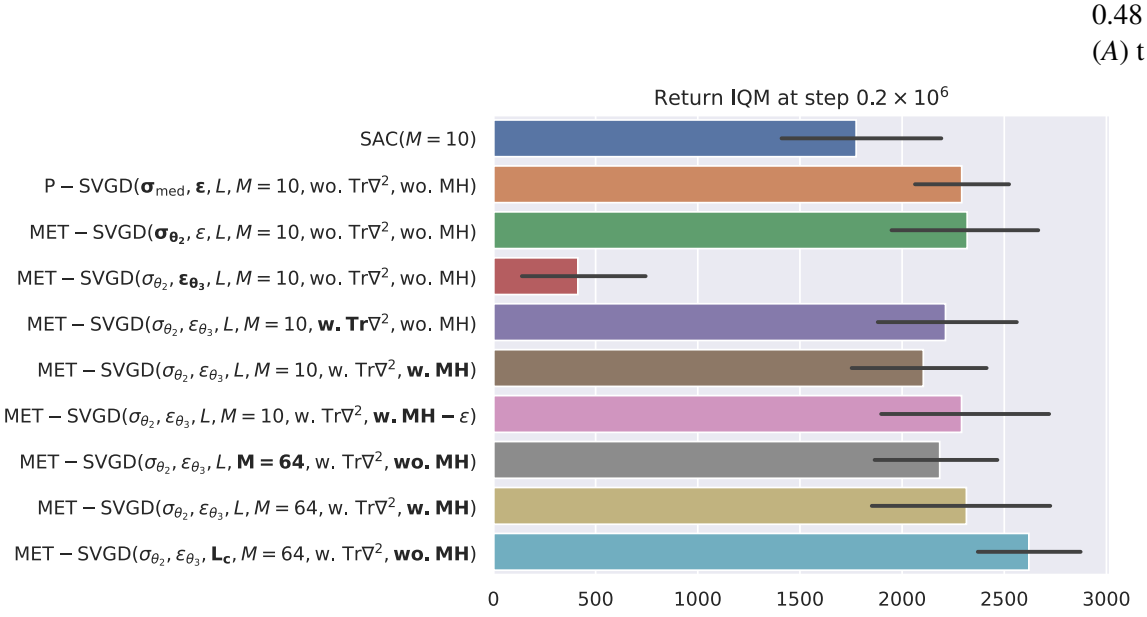


Figure 56: Walker's IQM Return at step 0.2×10^6.

Figure 57: IQM return scores at step 0.2×10^6 across environments.

Trainable SVGD Parameters (Humanoid Env) In Fig. 67, 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^2AC^+ 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.

In Fig. 80, 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. 77.

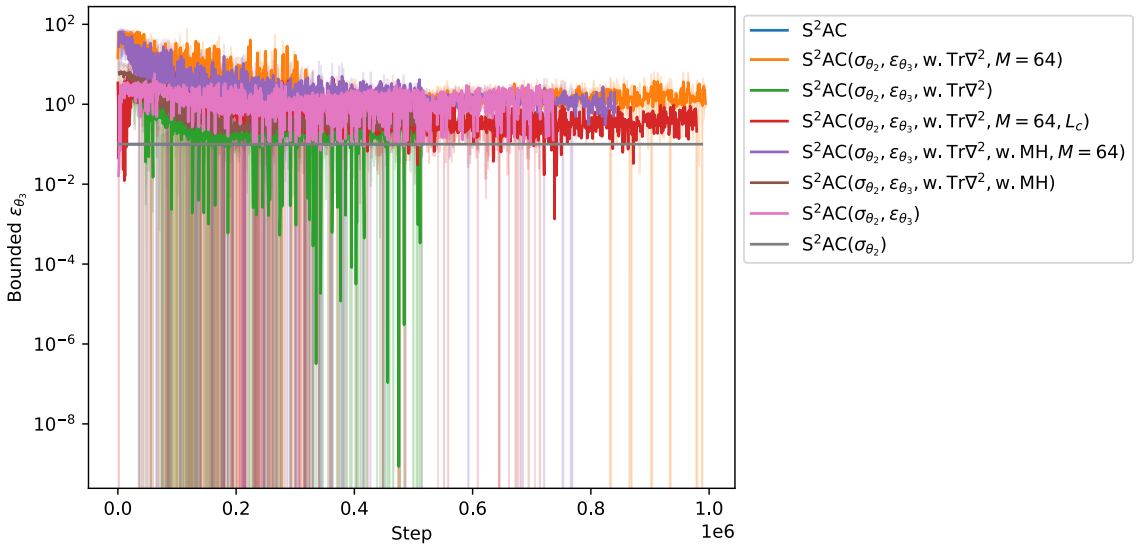


Figure 77: Learned step-size $\epsilon^l_{\theta_3}$ for Humanoid env.

Trainable SVGD Parameters (Walker Env). We visualize the scores and step-size in Fig. 78 and Fig. 79.

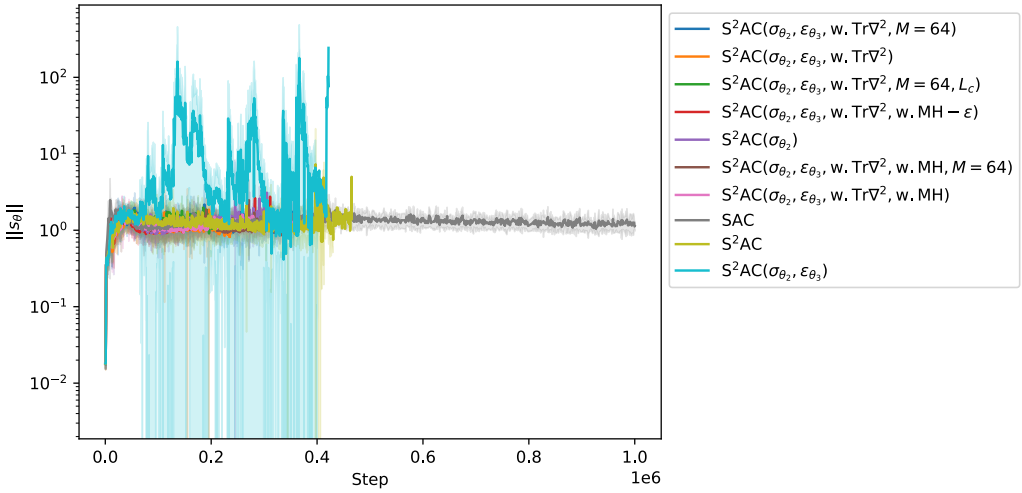


Figure 81: Scores (Walker env.)

0.35
(A) t

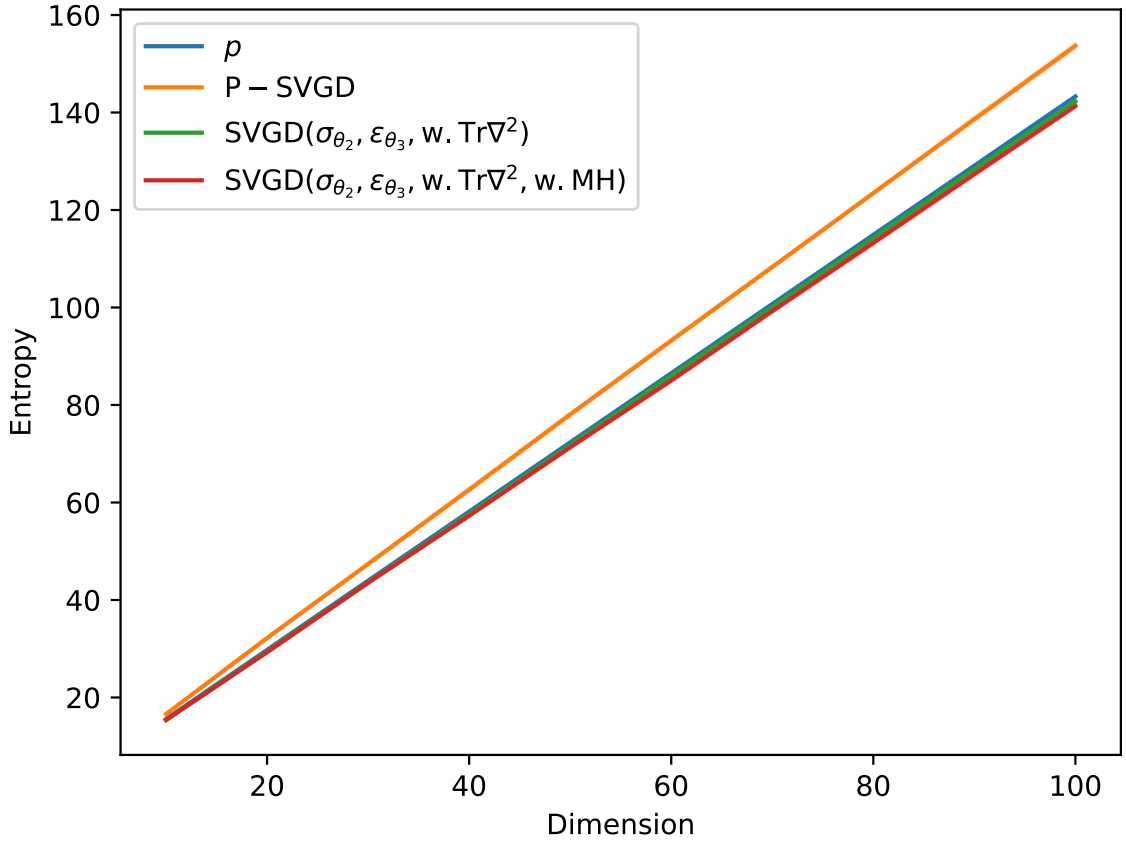
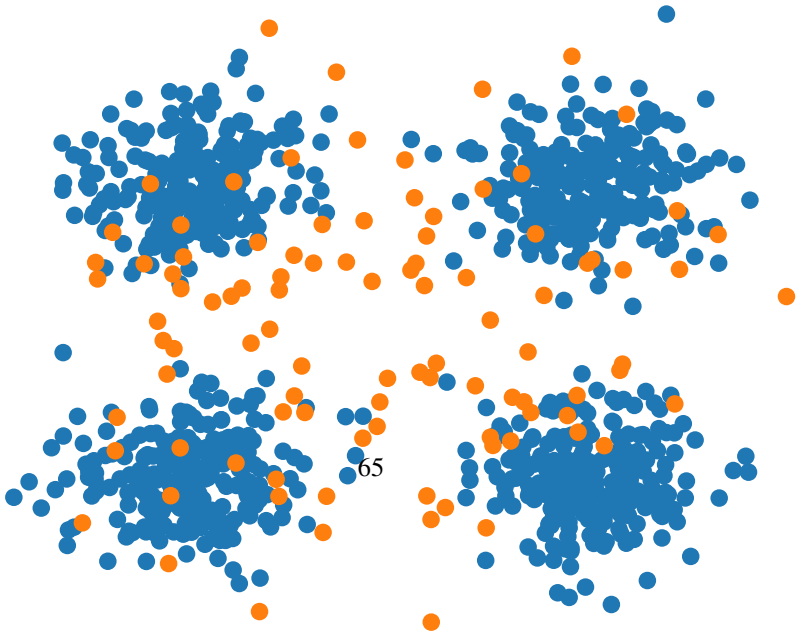


Figure 22: Estimated entropy
across dimensions.

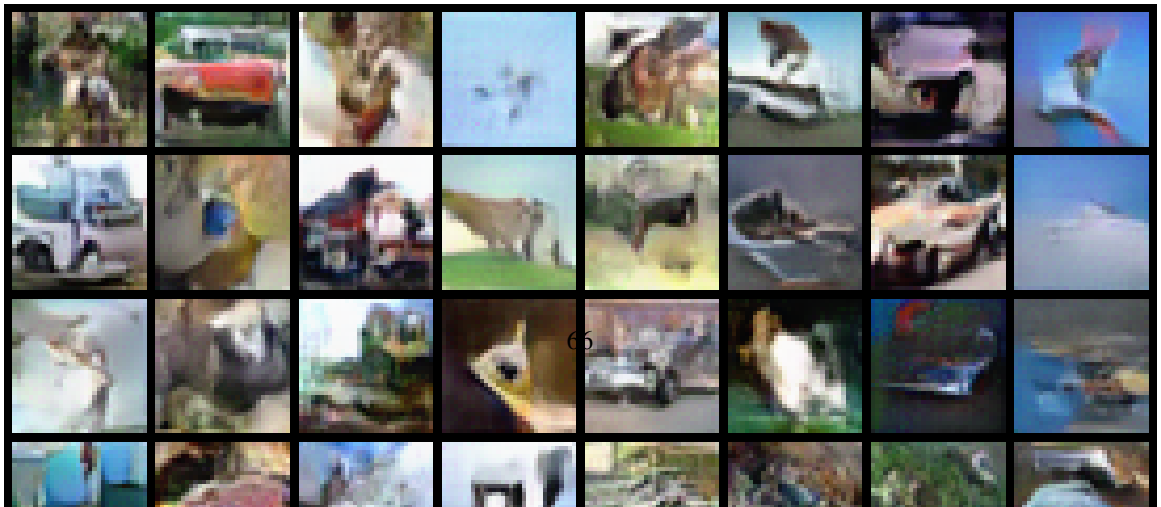
0.2
(A) t



0.3
(*A*) t

Figure 28: MET-SVGD(σ_{θ_2}, ϵ_{θ_3}, w.Tr∇^2, w.MH)

.3
(*A*) t



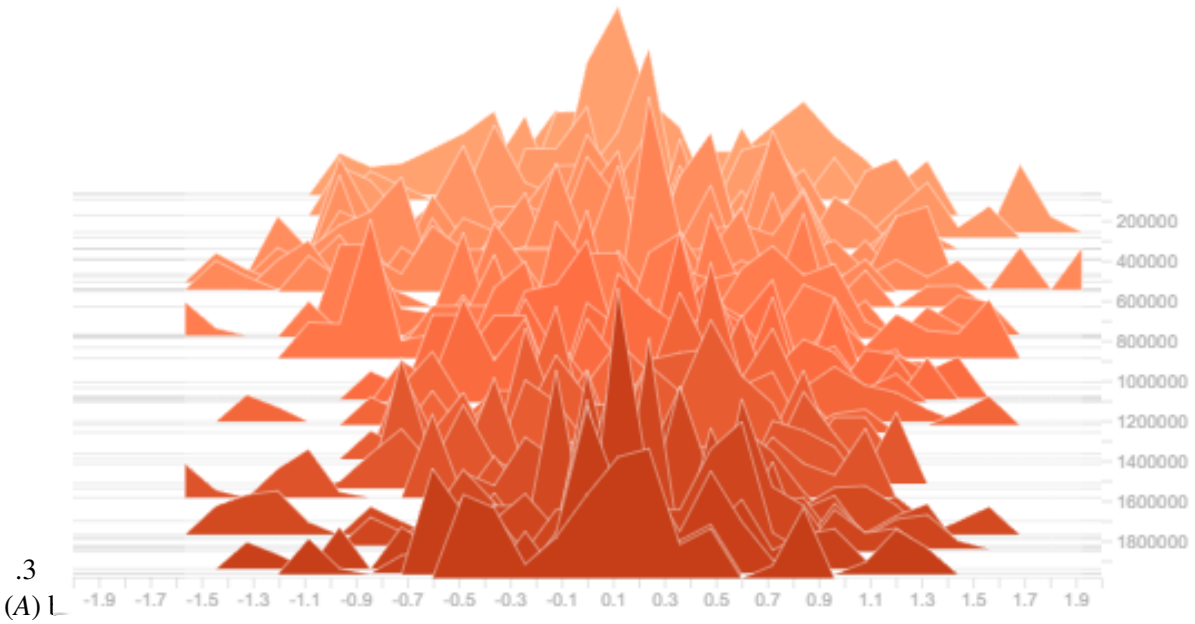


Figure 58: SAC

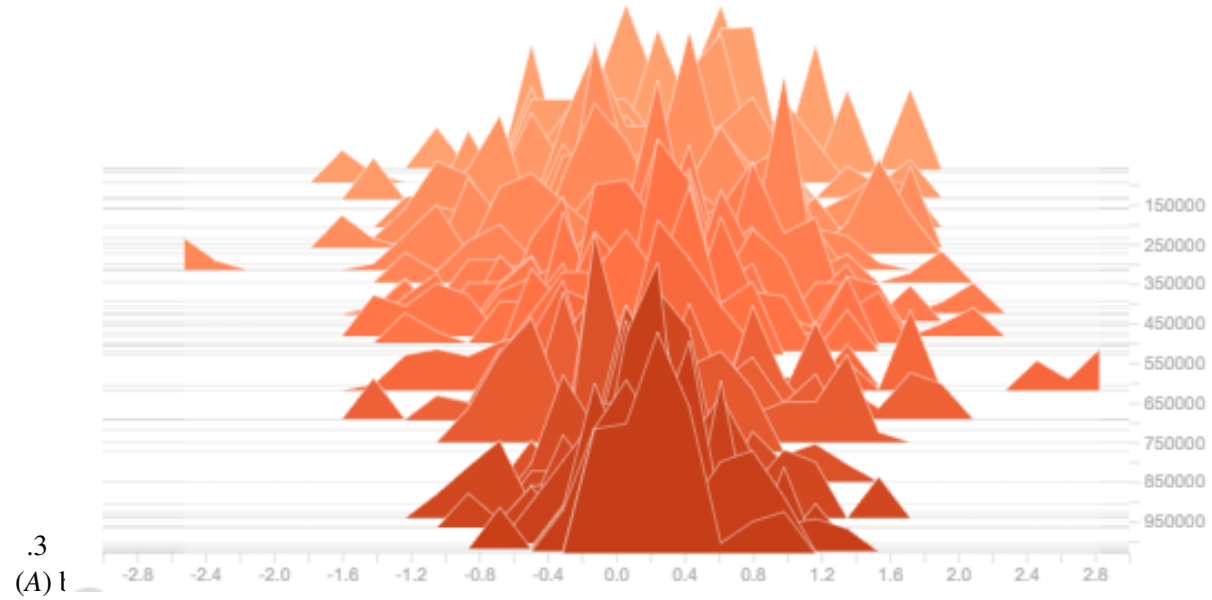


Figure 59: S^2AC

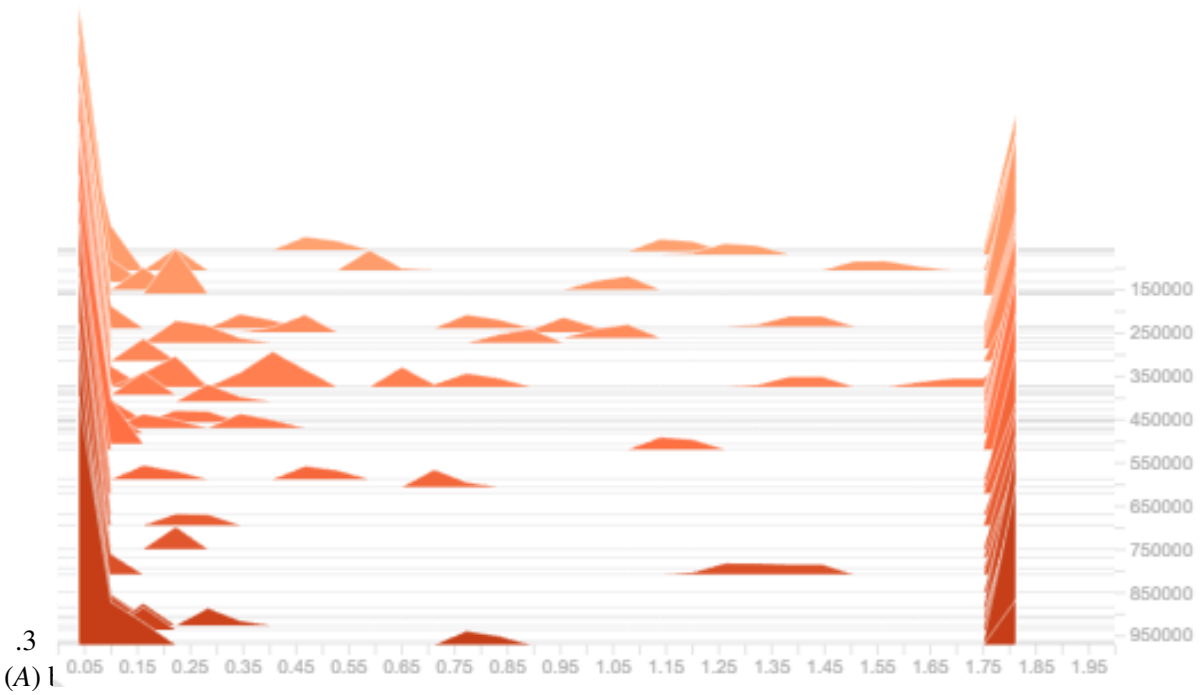


Figure 68: $\mathrm{S}^2\mathrm{AC}^+(\sigma_{\theta_2})$

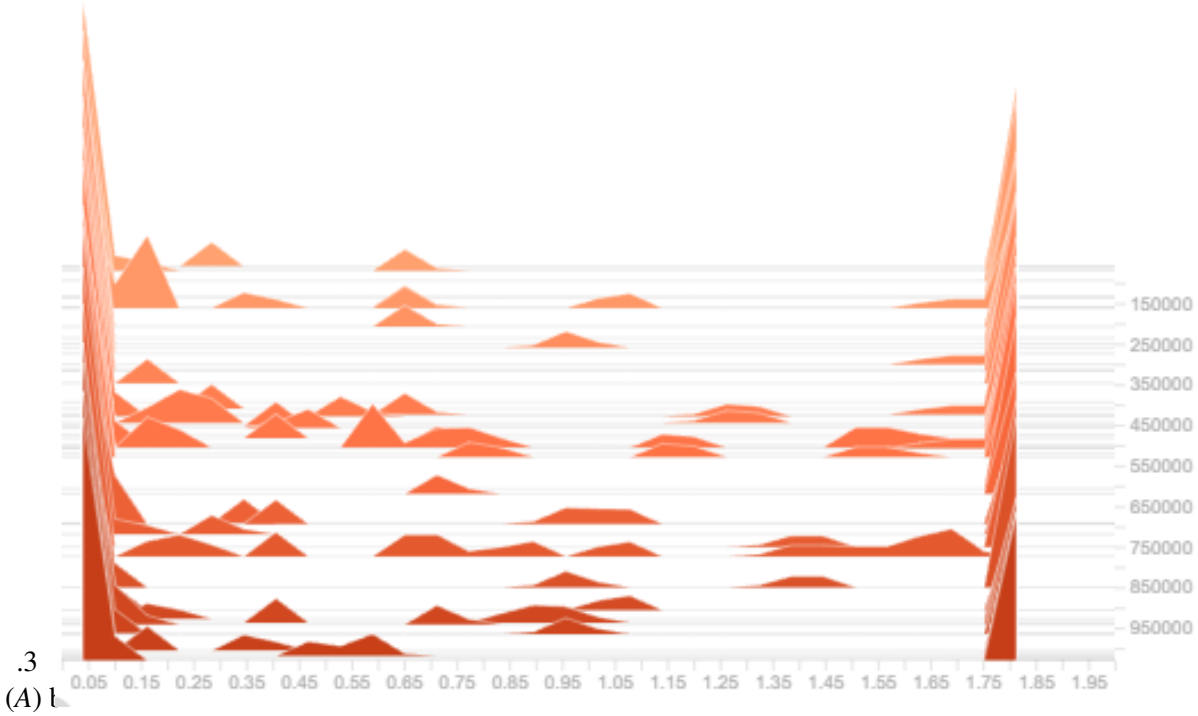
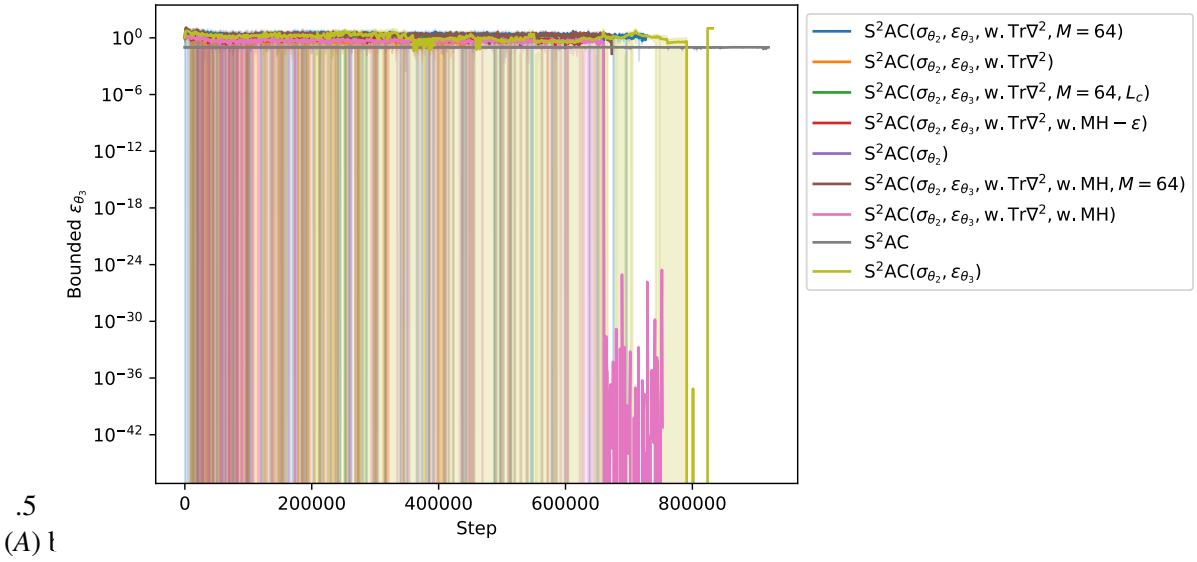
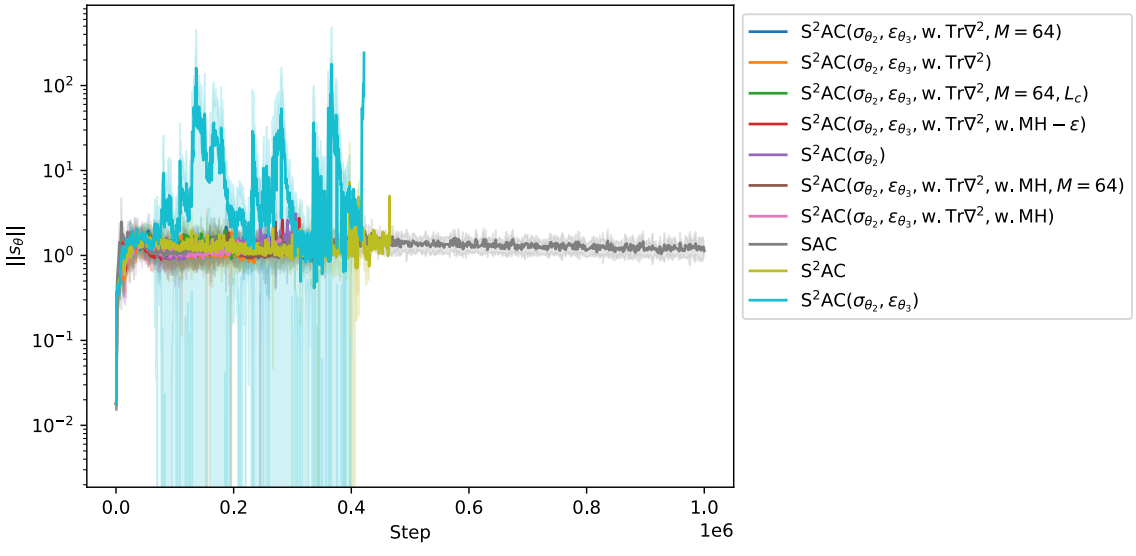


Figure 69: $\mathrm{S}^2\mathrm{AC}^+(\sigma_{\theta_2}, \epsilon_{\theta_3})$



.5
(*A*) l

Figure 78: Learnable step-size ϵ_3^l (Walker env.)



.5
(*A*) l

Figure 79: Scores (Walker env.)

Figure 80: Learned step-size and scores in Walker env.