Random-Projection Tree Stein Variational Gradient Descent

Shishuo Guo[1,*] **Xiaoyuan Cheng**[2,*] **Zhuo Sun**[1,3,*†]
[1]School of Statistics and Data Science, Shanghai University of Finance and Economics,
[2]University College London,
[3]Institute of Big Data Research, Shanghai University of Finance and Economics

Abstract

Vanilla SVGD is known to have a computational cost of $\mathcal{O}(N^2)$ per iteration. In this work, we introduce Random-Projection Tree Stein Variational Gradient Descent (RP-SVGD) to alleviate this computational challenge. This is achieved by restricting kernel interactions to spatially proximal particles clustered via a random projection tree, reducing to a cost of $\mathcal{O}(Nd(\log(N/C) + C))$, where C is a hyperparameter representing the maximum leaf node capacity of the spanning tree. To establish theoretical validity, we introduce a smoothed RP-Tree kernel for any fixed tree realization and prove that it belongs to the Stein class of the target distribution, thereby ensuring the generation of valid gradient flows. By considering an effective smoothed kernel as the expectation over random tree partitions, we verify that it preserves regularity along with other necessary conditions, which guarantee convergence under the approximate gradient flow framework. The practical algorithm approximates these dynamics with a single hard-partition tree per iteration, and we discuss the resulting gap. Extensive experimental results demonstrate that RP-SVGD tends to have competitive performance and speedups across various tasks.

1 Introduction

Characterizing complex and unnormalized probability distributions is one of the core challenges in fields such as Bayesian statistics, machine learning, and statistical physics [13, 7, 23]. To navigate this landscape, Stein Variational Gradient Descent (SVGD) [21] has emerged as a prominent particle-based inference framework, offering a principled gradient-based mechanism to transport an empirical measure toward a target density [20]. Unlike conventional sampling methods such as rejection sampling [24], importance sampling [28], and Markov Chain Monte Carlo (MCMC) [16], which rely on stochastic sampling with independent or weakly coupled samples, SVGD deterministically evolves a set of interacting particles, casting approximate inference as an optimization problem over an empirical measure. By simulating a gradient flow of the Kullback–Leibler (KL) divergence [18] under a kernelized Stein operator, SVGD evolves particles through functional gradient updates in a reproducing kernel Hilbert space (RKHS) [6], combining an attraction toward high-probability regions with a repulsive interaction that promotes diversity [20]. This flexibility and scalability have enabled SVGD to be successfully applied across a wide range of tasks, including Bayesian inference [21, 10], generative modeling [31], and reinforcement learning [22].

Motivation Despite these compelling theoretical advantages, the practical utility of SVGD is often limited by its computational cost. The core of this limitation lies in the quadratic complexity inherent in evaluating the Stein functional gradient [21]. As shown in Equation (2), updating each of the N particles requires computing kernel interactions with all other particles in the ensemble, resulting in an

*Equal Contribution.
†Corresponding Author. Correspondence to Zhuo Sun: sunzhuo@mail.shufe.edu.cn.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

$\mathcal{O}(N^2)$ cost per iteration. This quadratic scaling becomes particularly burdensome as the complexity of the target distribution increases, which typically necessitates a larger number of particles to achieve adequate coverage [4, 37]. Consequently, the computational and memory overhead grow rapidly with N, posing significant challenges in large-scale settings. Various acceleration strategies have been proposed to overcome this $\mathcal{O}(N^2)$ computational barrier [9, 10]. A common strategy to alleviate the computational burden is to approximate kernel interactions using subsets of particles. For instance, recent works, such as Global Batch SVGD (GB-SVGD) [10], explicitly restrict kernel evaluations to uniformly sampled subsets of particles. While this strategy reduces the per-iteration cost, uniform subsampling may fail to select the most representative subset that contains the most information of the update scheme. As a result, the sub-optimality of the selected subset would harm the finite-particle rate, see for instance Corollary 2 of [10], where the last term is of order $1/B$, worse than the standard $1/N$ rate without subsampling, with B being the batch size of GB-SVGD. For kernels widely adopted in practice, such as the Gaussian RBF, kernel values decay rapidly with distance, implying that particle interactions are predominantly local [26], which are not necessarily preserved under uniform subsampling. Meanwhile, uniformly random subsampling of the particles may disrupt these local distributional relationships. For instance, particles that would otherwise exert strong repulsive forces may be assigned to different subsets of particles. As a result, the strength of local repulsion could be underestimated. This structural disruption not only increases the variance of the Stein gradient estimates but can also weaken the repulsive interactions among particles, potentially leading to particle degeneracy or mode collapse as demonstrated in experiments in Section 4. As a result, the ability of SVGD to capture complex, multi-modal target distributions may be compromised [37].

These limitations of existing methods motivate this work: (i) the repulsive interactions in SVGD are primarily governed by local kernel interactions, making the full $\mathcal{O}(N^2)$ pairwise computation potentially redundant; (ii) while uniform subsampling reduces computational cost, it may disrupt the local neighborhood structure that is critical for these interactions. *This suggests the need for a partitioning scheme that can localize particle interactions, and enable the acceleration of SVGD.*

Our Approach To address both efficiency and performance, we employ Random Projection Trees (RP-Trees) [11] to adaptively group particles based on their geometric proximity, allowing us to restrict interactions within local partitions. This leads to a reduction in computational complexity while better preserving the local repulsive effects as observed in the experiments. Unlike standard spatial trees, such as KD-trees [5] suffering from the curse of dimensionality [34], an RP-tree recursively bisects the space using random hyperplanes, such that the diameter of its leaf cells decreases at a rate determined by the intrinsic low-dimensional structure of the data [12], rather than the ambient dimension d [11, 17]. Building upon this insight, we propose Random-Projection Tree Stein Variational Gradient Descent (RP-SVGD). At each iteration, an RP-tree is dynamically constructed to cluster spatially adjacent particles into shared leaf nodes, and kernel evaluations are restricted to these local neighborhoods. This partitioning scheme induces a structured sparsification of the kernel matrix, reducing the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(Nd(\log(N/C) + C))$, while preserving the essential local repulsive interactions. As a result, the proposed approach matches vanilla SVGD at lower cost.

Main Contributions We propose Random-Projection Tree SVGD, which dynamically constructs RP-Trees to restrict particle interactions to spatially proximal neighborhoods. This structured sparsification reduces the per-iteration computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(Nd(\log(N/C) + C))$, effectively bypassing the $O(N^2)$ barrier while explicitly preserving the local repulsive forces necessary to prevent mode collapse. We introduce a smoothed RP-Tree kernel and establish the convergence of RP-SVGD under the effective smoothed kernel. We provide an empirical ablation study on the maximum leaf node capacity C, demonstrating that RP-SVGD is robust to this hyperparameter, with a moderate capacity striking an optimal balance between approximation fidelity and computational speed. Finally, experiments on multi-modal 2D distributions, Bayesian logistic regression, Bayesian neural networks and reinforcement learning tasks validate that RP-SVGD yields wall-clock speedups over vanilla SVGD while maintaining competitive performance.

2 Background

In this section, we briefly review the two main related works: Stein Variational Gradient Descent and Random Projection Tree.

Stein Variational Gradient Descent Stein Variational Gradient Descent (SVGD), introduced by [21], is a particle-based variational inference algorithm that evolves a set of particles $\{\boldsymbol{x}_i\}_{i=1}^N$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ to approximate an unnormalized target posterior distribution $p(\boldsymbol{x}) \propto p_0(\boldsymbol{x})f(\boldsymbol{x})$ where p_0 denotes the prior of x and f denotes the likelihood or energy function that involves x. The core mechanism of SVGD is to find an optimal transport map $\mathbf{T}(\boldsymbol{x}) = \boldsymbol{x} + \epsilon\boldsymbol{\phi}(\boldsymbol{x})$ that minimizes the Kullback-Leibler (KL) divergence between the transformed particle distribution $q_{[\mathbf{T}]}$ and the target p. It was shown that the steepest descent direction in the unit ball of a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}^d$ is given by:

$$\boldsymbol{\phi}^*(\boldsymbol{x}) = \mathbb{E}_{\mathbf{z}\sim q}\left[\nabla_{\mathbf{z}} \log p(\mathbf{z})k(\mathbf{z}, \boldsymbol{x}) + \nabla_{\mathbf{z}} k(\mathbf{z}, \boldsymbol{x})\right], \tag{1}$$

where $k(\cdot, \cdot)$ is the reproducing kernel of $\mathcal{H}$. In practice, given N particles $\{\boldsymbol{x}_i^t\}_{i=1}^N$ at iteration t, the update rule is implemented as: $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \epsilon_t \hat{\boldsymbol{\phi}}^*(\boldsymbol{x}_i^t)$, with

$$\hat{\boldsymbol{\phi}}^*(\boldsymbol{x}_i^t) = \tfrac{1}{N}\textstyle\sum_{j=1}^N \left[\nabla_{\boldsymbol{x}_j^t} \log p(\boldsymbol{x}_j^t)k(\boldsymbol{x}_j^t, \boldsymbol{x}_i^t) + \nabla_{\boldsymbol{x}_j^t} k(\boldsymbol{x}_j^t, \boldsymbol{x}_i^t)\right]. \tag{2}$$

where ϵ_t is the step size. The first term in the summation, known as the driving force, pushes particles toward high-probability regions of p. While the second term, known as the repulsion force, prevents particles from collapsing into a single mode by leveraging the curvature of the kernel. This update takes $\mathcal{O}(N^2)$ computational cost per iteration step for the vanilla SVGD.

Random Projection Tree Random Projection Trees (RP-Trees) [11] are a class of space-partitioning data structures designed to adapt to the intrinsic low-dimensional structures of high-dimensional data. Formally, an RP-Tree recursively bisects a region $\Omega \subseteq \mathbb{R}^d$ using random hyperplanes. At each internal node u, the split is defined by a routing function $h_u(x) = \mathbb{I}(v_u^\top x \le m_u)$, where $v_u \in \mathbb{S}^{d-1}$ is a projection direction sampled uniformly at random, and $m_u \in \mathbb{R}$ is a threshold typically chosen as the median of the projected data. The primary theoretical appeal of RP-Trees lies in their manifold adaptation property: unlike axis-aligned structures such as KD-trees [5], the diameter of the cells in an RP-Tree decreases at a rate determined by the data's intrinsic dimension rather than the ambient dimension d. Originally utilized for accelerated nearest neighbor search and manifold learning, RP-Trees provide a computationally efficient mechanism to cluster points that are geometrically proximal while maintaining a hierarchical, convex partition of the space.

3 Method

To reduce the $\mathcal{O}(N^2)$ computational cost of vanilla SVGD, we reconsider the structure of particle interactions. Our approach is premised on the insight that kernel-based dynamics are intrinsically local: the influence of a particle decays rapidly with distance, rendering long-range interactions statistically negligible compared to local repulsion [27, 1, 36, 33]. Note that not all kernels decay rapidly, though many kernels used in SVGD in practice do so (e.g., Gaussian/Laplacian kernels). Leveraging this effective sparsity induced by the kernel, we propose a novel SVGD algorithm through the lens of RP-Trees. In Section 3.1, we formalize this with an effective RP-Tree kernel and introduce its smooth approximations. Section 3.2 defines an idealized effective kernel via expectation and relates the resulting dynamics to the convergence analysis of the approximate gradient flow framework [14]. Section 3.3 presents a practical algorithm, its computational complexity, and how it departs from the idealized setting.

3.1 Random Projection Tree Kernel

To capture local geometric structure and reduce the computational burden of kernel-based particle interactions, we introduce a stochastic kernel inspired by Random Projection Trees (RP-Trees) [11]. Let $\Omega = \mathbb{R}^d$ denote the support of the target distribution. An RP-Tree $\mathcal{T}$ of depth D recursively partitions Ω into a collection of non-overlapping regions via a sequence of random hyperplane splits. The tree is parameterized by a set of random variables $\xi = \{(\boldsymbol{v}_u, m_u)\}_{u\in\mathcal{U}}$, where $\mathcal{U}$ denotes the set of internal nodes. For each node u, the projection direction $\boldsymbol{v}_u$ is sampled uniformly from the unit sphere $\mathbb{S}^{d-1}$, and the threshold $m_u \in \mathbb{R}$ is typically chosen as the median of the projected data along $\boldsymbol{v}_u$. We define the routing function at node u: $h_u(\boldsymbol{x}) = \mathbb{I}(\boldsymbol{v}_u^\top \boldsymbol{x} \le m_u)$. Each leaf node $L \in \mathcal{L}_\mathcal{T}$ is uniquely identified by a path from the root to depth D. The corresponding leaf region $C_L \subset \Omega$ is a convex polytope defined by the intersection of D half-spaces: $C_L = \bigcap_{j=1}^D \left\{\boldsymbol{x} \in \Omega : (-1)^{b_{L,j}}\left(\boldsymbol{v}_{L,j}^\top \boldsymbol{x} - m_{L,j}\right) \le 0\right\}$, where $b_{L,j} \in \{0, 1\}$ encodes the routing decision

at depth j. The collection of leaf regions forms a measurable partition of the space, satisfying $\bigcup_{L\in\mathcal{L}_\mathcal{T}} C_L = \Omega$, and their interiors are mutually disjoint, i.e., $C_L \cap C_{L'} = \emptyset$ for $L \neq L'$.

To exploit this partition structure for scalable particle interactions, we incorporate the RP-Tree structure into the kernel design. Let $k(\boldsymbol{x}, \boldsymbol{y})$ be a base kernel, such as the Gaussian RBF kernel. We define a kernel that measures whether the two points are in the same leaf region as follows,

$$k_{\mathcal{P}(\xi)}(\boldsymbol{x}, \boldsymbol{y} \mid \mathcal{T}) = \textstyle\sum_{L\in\mathcal{L}_\mathcal{T}} \mathbb{I}(\boldsymbol{x} \in C_L)\, \mathbb{I}(\boldsymbol{y} \in C_L) = \mathbb{I}_{\text{same-leaf}}(\boldsymbol{x}, \boldsymbol{y}). \tag{3}$$

The resulting RP-Tree kernel is defined as

$$k_\xi(\boldsymbol{x}, \boldsymbol{y} \mid \mathcal{T}) := k(\boldsymbol{x}, \boldsymbol{y})\, k_{\mathcal{P}(\xi)}(\boldsymbol{x}, \boldsymbol{y} \mid \mathcal{T}). \tag{4}$$

This construction enforces strict locality by masking cross-leaf-region interactions, thereby inducing structural sparsity in the kernel Gram matrix. However, the use of hard indicator functions introduces jump discontinuities at the partition boundaries, rendering k_ξ non-differentiable. In particular, the gradient $\nabla_{\boldsymbol{x}} k_\xi(\boldsymbol{x}, \boldsymbol{y})$ involves distributional terms concentrated on the boundaries of the leaf regions. This may violate the regularity conditions required by Stein Variational Gradient Descent, which relies on kernels belonging to the Stein class to define a valid gradient flow.

To restore differentiability and establish theoretical validity, we introduce a smooth approximation to the RP-Tree kernel. Let $H_\tau : \mathbb{R} \to (0, 1)$ be a family of smooth soft-threshold functions indexed by $\tau > 0$, satisfying (i) $H_\tau \in C^\infty$, and (ii) $\lim_{\tau\to 0} H_\tau(z) = \mathbb{I}(z \leq 0)$ for all $z \neq 0$. A canonical choice is the sigmoid function $H_\tau(z) = \sigma(-z/\tau)$. For a leaf node L characterized by the path parameters $\{(\boldsymbol{v}_{L,j}, m_{L,j}, b_{L,j})\}_{j=1}^D$, we define the corresponding soft membership function as $\tilde{\mathbb{I}}_{\xi_L,\tau}(\boldsymbol{x}) = \prod_{j=1}^D H_\tau((-1)^{b_{L,j}} (\boldsymbol{v}_{L,j}^\top \boldsymbol{x} - m_{L,j}))$. Using these soft memberships, we define the smoothed RP-Tree kernel

$$k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y}) := k(\boldsymbol{x}, \boldsymbol{y}) \sum_{L\in\mathcal{L}_\mathcal{T}} \tilde{\mathbb{I}}_{\xi_L,\tau}(\boldsymbol{x})\, \tilde{\mathbb{I}}_{\xi_L,\tau}(\boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) \tilde{\mathbb{I}}_{\text{same-leaf},\tau}(\boldsymbol{x}, \boldsymbol{y}). \tag{5}$$

This kernel preserves the localized interaction structure induced by the RP-Tree while remaining smooth with respect to both arguments. We now formally present: (i) $k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})$ is a valid kernel; (ii) $k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})$ is also in the Stein class of the target distribution p when the base kernel k is in that Stein class in the following Proposition 1.

Proposition 1 (Validity of the Smoothed RP-Tree Kernel) *Let $k(\cdot, \cdot)$ be a positive semi-definite (PSD) kernel on $\mathcal{X} \times \mathcal{X}$ and $p(\boldsymbol{x})$ be a continuously differentiable target density on $\mathcal{X}$. For any $\tau > 0$, the smoothed RP-Tree kernel $k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})$ defined in Eq. (5) satisfies: (1) $k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})$ is a valid PSD kernel; (2) If $k(\cdot, \cdot) \in C^m(\mathcal{X} \times \mathcal{X})$, then $k_{\xi,\tau} \in C^m(\mathcal{X} \times \mathcal{X})$; (3) If k belongs to the Stein class of p, then $k_{\xi,\tau}$ also belongs to the Stein class of p. That is, for any fixed $\boldsymbol{y}$, the function $k_{\xi,\tau}(\cdot, \boldsymbol{y})$ is differentiable and satisfies the Stein identity: $\mathbb{E}_{\boldsymbol{x}\sim p}\left[\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y}) + \nabla_{\boldsymbol{x}} k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})\right] = \mathbf{0}$. See Section B.1 for proof.*

3.2 Stochastic Ascent on the Effective Kernel

While the smoothed partition kernel $k_{\xi,\tau}$ induced by a single realization of an RP-tree is a valid reproducing kernel, any fixed partition necessarily introduces boundary artifacts: geometrically nearby points separated by a partition boundary may experience artificially weakened interactions. To eliminate dependence on a specific partition and restore geometric invariance, we randomize the tree construction and consider the kernel dynamics in expectation.

Let $\xi \sim \Xi$ denote the random variables parameterizing the RP-tree construction, including random projection directions and split thresholds. For a fixed partition ξ, we define the *smooth random kernel direction* as the functional gradient in the RKHS induced by $k_{\xi,\tau}$:

$$\boldsymbol{\phi}^*_{\xi,\tau}(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{z}\sim q}\left[\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z}) k_{\xi,\tau}(\boldsymbol{z}, \boldsymbol{x}) + \nabla_{\boldsymbol{z}} k_{\xi,\tau}(\boldsymbol{z}, \boldsymbol{x})\right]. \tag{6}$$

To achieve geometric invariance, we consider the *smooth effective kernel direction*, $\boldsymbol{\phi}^*_\tau$, which is the expectation over the distribution of tree parameters:

$$\boldsymbol{\phi}^*_\tau(\boldsymbol{x}) := \mathbb{E}_\xi\left[\boldsymbol{\phi}^*_{\tau,\xi}(\boldsymbol{x})\right]. \tag{7}$$

By the linearity of the Stein operator, $\boldsymbol{\phi}^*_\tau$ corresponds to the update direction induced by the effective kernel $k_\tau(\boldsymbol{x}, \boldsymbol{y}) := \mathbb{E}_\xi[k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})]$. The expectation $\mathbb{E}_\xi[\cdot]$ captures the co-membership probability under random projection-induced partitions. Due to the properties of random tilings, this probability

decays monotonically with the angular distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ [8, 30]. Consequently, k_τ acts as a geometry-adaptive modulation of the base kernel, retaining strong local interactions while suppressing distant interactions that primarily contribute to variance. In the practical limit where the smoothing parameter $\tau \to 0$, we operate with hard partitions. We denote the corresponding update direction for a single tree as the random kernel direction $\boldsymbol{\phi}^*_\xi$, and the ideal update direction as the effective kernel direction $\boldsymbol{\phi}^*(\boldsymbol{x}) := \mathbb{E}_\xi[\boldsymbol{\phi}^*_\xi(\boldsymbol{x})]$.

By definition, the effective kernel is the expectation over tree partitions, $k_\tau(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}_\xi[k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})]$. Because expectation preserves linearity and limits, k_τ naturally inherits all the desirable theoretical properties established in Proposition 1, including positive semi-definiteness, smoothness, and the Stein class property.

Remark 1 (Convergence under k_τ) *Analyzing the expected dynamics of RP-SVGD is equivalent to studying the standard SVGD continuous dynamics driven by this effective kernel k_τ. Recent advances by [14] establish via the approximate gradient flow framework that SVGD guarantees convergence to the target distribution in Kullback-Leibler divergence, provided that the driving kernel satisfies four specific conditions: smoothness and boundedness (Assumption 1), integrally strictly positive definiteness (Assumption 2), bounded trace (Assumption 3), and strictly positive eigenvalue bounds (Assumption 4). As we verify in Section B.2, building upon Proposition 1, k_τ preserves all the aforementioned regularity and ISPD conditions of any valid base kernel.*

3.3 Practical Algorithm

While the effective smoothed kernel k_τ provide theoretical guarantees as established in Section 3.2, directly computing its expectation over the high-dimensional tree parameter space Ξ is practically intractable. To bridge the gap between theory and practice, we employ a stochastic approximation. At each iteration t, we sample an independent random tree realization ξ_t to construct the corresponding smoothed single-tree kernel k_{τ,ξ_t}. The empirical update direction is then computed as an unbiased stochastic estimate of the ideal effective direction:

$$\hat{\phi}_t(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim q}\left[\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z}) k_{\xi_t,\tau}(\boldsymbol{z}, \boldsymbol{x}) + \nabla_{\boldsymbol{z}} k_{\xi_t,\tau}(\boldsymbol{z}, \boldsymbol{x})\right]. \tag{8}$$

Although the smoothed kernel $k_{\xi_t,\tau}$ ensures analytical differentiability for any fixed temperature $\tau > 0$, explicitly evaluating its gradients across the soft boundaries for all particle pairs remains computationally expensive. Since our primary motivation is to accelerate SVGD, we operate in the limit $\tau \to 0$, where the soft memberships become hard indicators and particle interactions are confined to the leaves. This hard-partitioning scheme sparsifies the kernel matrix; the resulting algorithm, RP-SVGD, is summarized in Algorithm 1. The hard limit, together with the use of a single tree per iteration and of median splits computed from the current particles, takes the algorithm outside the idealized setting of Section 3.2; we discuss this gap in Section 5 and examine it empirically in Section D.4 and Section D.5.

Algorithm 1 RP-SVGD

Require: $\{\boldsymbol{x}_i^{(0)}\}, \mathbf{s}(\boldsymbol{x}) =: \nabla \log p(\boldsymbol{x}), \{\epsilon_t\}, C$
for $t = 0$ to $T-1$ **do**
  Construct RP-tree $\mathcal{T}^{(t)}$ via ξ_t
  $\boldsymbol{\phi}(\boldsymbol{x}_i) \leftarrow \mathbf{0}$
  for leaf $L_k \in \mathcal{T}^{(t)}$ **do**
    for $\boldsymbol{x}_i \in L_k$ **do**
      $\boldsymbol{\phi}(\boldsymbol{x}_i) \leftarrow \frac{1}{|L_k|} \sum_{\boldsymbol{x}_j \in L_k} [k\mathbf{s} + \nabla k]$
    end for
  end for
  $\boldsymbol{x}_i^{(t+1)} \leftarrow \boldsymbol{x}_i^{(t)} + \epsilon_t \boldsymbol{\phi}(\boldsymbol{x}_i)$
end for

We now analyze the per-iteration computational cost of RP-SVGD. Standard SVGD requires evaluating all-to-all particle interactions, leading to an $\mathcal{O}(N^2 d)$ complexity. Our method reduces this through a spatial partitioning scheme, which consists of two stages: tree construction and local kernel evaluation.

1. Tree Construction: Each iteration involves constructing a random projection tree to partition the space. With N particles in d-dimensional space, constructing a tree of depth D involves d-dimensional projections at each level. This incurs a cost of $\mathcal{O}(NdD)$.

2. Local Kernel Evaluation: Let L_k denote the set of particles in the k-

Table 1: Per-iteration complexity comparison. N: particle count; d: dimension; D: tree depth; C: max leaf capacity.

Method	*Tree*	*Kernel*	*Total*
Vanilla-SVGD	–	$\mathcal{O}(N^2 d)$	$\mathcal{O}(N^2 d)$
GB-SVGD	–	$\mathcal{O}(NBd)$	$\mathcal{O}(NBd)$
RP-SVGD	$\mathcal{O}(Nd\log(\frac{N}{C}))$	$\mathcal{O}(NCd)$	$\mathcal{O}(Nd(\log(\frac{N}{C}) + C))$

th leaf node. Interactions are restricted within these leaves, resulting in an interaction cost of $\mathcal{O}(d\sum_k |L_k|^2)$.

In practice, we enforce a constraint on the maximum leaf capacity by a constant C, such that $|L_k| \leq C$. Under this balanced partitioning, the tree depth D necessary to reach this capacity satisfies $N/2^D = C$, implying $D = \mathcal{O}(\log(N/C))$. Substituting $D \approx \mathcal{O}(\log N/C)$ and $|L_k| \leq C$ into our cost functions: *the construction cost* is $\mathcal{O}(NdD) = \mathcal{O}(Nd\log(N/C))$ and *the interaction cost* is $\mathcal{O}(d\sum_k |L_k|^2) \approx \mathcal{O}(d\cdot\frac{N}{C}\cdot C^2) = \mathcal{O}(NCd)$. Summing these, the overall per-iteration complexity of RP-SVGD is $\mathcal{O}(Nd\log(N/C) + NCd)$, which is summarized in Table 1. By effectively controlling the leaf capacity C, we reduce the original quadratic $\mathcal{O}(N^2d)$ complexity of standard SVGD to near-linear complexity in the number of particles N. This reduction makes RP-SVGD highly scalable for large-scale particle ensembles.

Remark 2 (Depths of RP-Trees and Leaf Node Capacity) *Proposition 1 and the verification in Appendix B.2 hold for any fixed finite depth and any $\tau > 0$. While it is tempting to increase the tree depth D (and decrease the leaf node capacity) to minimize the cost, this may degrade the approximation to the base kernel. In practice, a moderate value of D works sufficiently well. For the empirical sensitivity to the depth of tree D, please refer to Figure 5, Table 10 and Figure 3 for ablation results under the settings of 2-dimensional synthetic examples, BLR tasks and reinforcement learning tasks of this work.*

4 Experiments

In this section, we compare our method, RP-SVGD, mainly against vanilla SVGD[21], Matrix-SVGD[32] and GB-SVGD[10] on synthetic 2-dimensional examples, Bayesian logistic regression, Bayesian neural networks and reinforcement learning tasks. Due to page limitation, additional experimental results and ablation studies including enabling GPU execution for RP-SVGD, variance of single-tree approximation and slow-decaying kernels are provided in Appendix D.

4.1 Synthetic Examples

We first evaluate the approximate fidelity of RP-SVGD relative to vanilla-SVGD, GB-SVGD, and Matrix-SVGD on several complex two-dimensional target distributions. Closed-form probability density functions for each problem are detailed in Appendix C.2. For each task, we initialize $N = 500$ particles from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and perform 1000 steps of optimization. All methods employ an RBF kernel with its bandwidth adaptively set using a median heuristic. The approximate quality is qualitatively evaluated through particle visualization and quantitatively evaluated using the using the Maximum Mean Discrepancy (MMD) along with various metrics, the specific calculation formulas and their functions of which are detailed in Appendix C.3.

Table 2: Comparison of execution time per iteration for different algorithms: evaluated on the 2D synthetic tasks with $N = 500$ particles.

Algorithm	**Time / Iter. (ms)**
Vanilla-SVGD	3.689 ± 0.116
RP-SVGD	1.710 ± 0.037
GB-SVGD	0.410 ± 0.017
Matrix-SVGD	3.772 ± 0.140

The quantitative results in Table 8 and the visualization results in Figure 1 demonstrate that RP-SVGD achieves the best balance between approximate accuracy and computational efficiency. As shown in Table 8, RP-SVGD exhibits the lowest MMD, W_2, and Energy distance across all synthetic benchmarks, outperforming vanilla-SVGD in both speed and accuracy. Observing Figure 1 reveals that RP-SVGD generally requires fewer iterations to converge than Vanilla-SVGD, while achieving convergence faster in terms of overall wall-clock time. For instance, RP-SVGD reaches convergence in 0.42 seconds on the GMM-25 task and 0.59 seconds on the Two-Rings task, whereas Vanilla-SVGD takes 1.20 seconds and 1.97 seconds, respectively. In contrast, GB-SVGD fails to converge properly across these benchmarks (indicated by NA in the figures). Matrix-SVGD has the worst final MMD on three of the four tasks: 0.0214 versus 0.0084 on Two-Rings, 0.0100 versus 0.0003 on GMM-8, and 0.0145 versus 0.0013 on GMM-25. Its W_2, Energy, KL, and Precision results are also generally worse. Moreover, its per-iteration cost is approximately $2.2\times$ higher (Table 2). Furthermore, Table 2 details the average execution time per iteration across different algorithms. The results clearly demonstrate that our method drastically reduces the per-step computational cost. Furthermore, we investigate the sensitivity of RP-SVGD to its core spatial partitioning hyperparameter: the maximum leaf node capacity C. As detailed in Appendix D.1 and visually summarized in Figure 5.

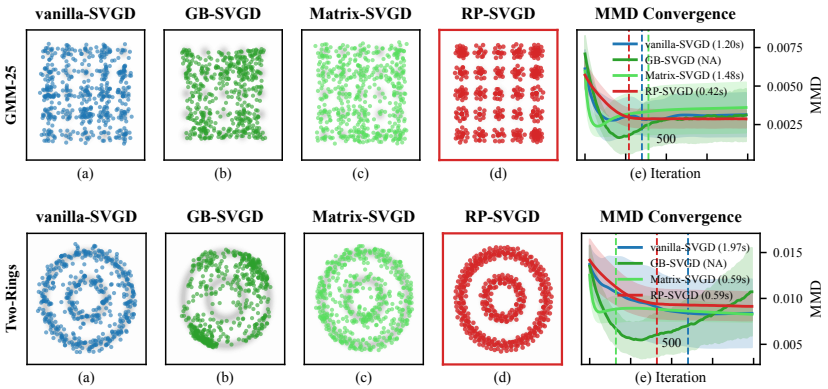

Figure 1: (a)-(d) show the particles obtained by various methods on the different target distributions at the final iteration. (e) MMD convergence trajectories over training iterations. The curves and shaded regions denote the mean and standard deviation across 10 independent trials, respectively. The vertical dashed lines mark the iteration at which each method converges, while the annotations above the plots indicate the corresponding wall-clock time to convergence.

4.2 Bayesian Logistic Regression

We consider the Bayesian Logistic Regression model for binary classification tasks. Let $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^N$ denote a dataset with feature vector $x_j \in \mathbb{R}^D$ and binary label $y_j \in \{0, 1\}$. The posterior distribution of interest is

$$p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)\, p(\theta), \quad \text{where } p(\mathcal{D} \mid \theta) = \textstyle\prod_{j=1}^N [\sigma(\theta^\top x_j)^{y_j} \left(1 - \sigma(\theta^\top x_j)\right)^{1-y_j}] \tag{9}$$

where $\theta \in \mathbb{R}^D$ is the weight vector and $\sigma(z) = 1/(1 + \exp(-z))$ denotes the sigmoid function. The prior distribution is chosen as the standard normal distribution $\theta \sim \mathcal{N}(0, I)$. We approximate the posterior using $N = 500$ particles. Our RP-SVGD is compared against Vanilla and GB-SVGD on four datasets (Breast Cancer, Ionosphere, German Credit, Covtype) by averaging Accuracy, Negative Log-Likelihood (NLL) and RMSE on a 20% test set over 10 random trials. Table 3 summarizes the predictive performance and computational efficiency of the evaluated methods on the BreastCancer and Covtype datasets from UCI benchmarks. RP-SVGD consistently matches or surpasses the performance of vanilla SVGD in terms of accuracy, NLL, and RMSE, while simultaneously reducing the total wall-clock time by over 50% on both datasets. While GB-SVGD also offers competitive runtimes, it often exhibits lower predictive accuracy and stability compared to our method. In contrast, RP-SVGD demonstrates well performance and robustness across diverse datasets. Overall, these results confirm that RP-SVGD provides a scalable and robust alternative for Bayesian inference.

Table 3: Performance on UCI benchmarks. We report the Mean $\pm$ Standard Deviation over 10 independent trials. For each metric, the best performing method is **bolded** and the second best is underlined. $\uparrow$ indicates higher values are better, and $\downarrow$ indicates lower values are better.

Dataset	Method	Test Accuracy $\uparrow$	Test NLL $\downarrow$	Test RMSE $\downarrow$	Time (s) $\downarrow$
BreastCancer	Vanilla-SVGD	0.9678 ± 0.0174	0.0778 ± 0.0070	0.1702 ± 0.0043	6.27 ± 0.12
	RP-SVGD	$\mathbf{0.9766 \pm 0.0083}$	$\mathbf{0.0757 \pm 0.0097}$	$\mathbf{0.1693 \pm 0.0056}$	3.06 ± 0.08
	Matrix-SVGD	0.9720 ± 0.0253	0.1008 ± 0.0018	0.1839 ± 0.0019	13.68 ± 0.21
	GB-SVGD	0.9668 ± 0.0083	0.0768 ± 0.0080	0.1719 ± 0.0067	$\mathbf{0.56 \pm 0.03}$
Covtype	Vanilla-SVGD	0.7527 ± 0.0070	$\mathbf{0.5186 \pm 0.0063}$	0.2833 ± 0.0008	10.68 ± 0.12
	RP-SVGD	$\mathbf{0.7562 \pm 0.0085}$	0.5235 ± 0.0063	$\mathbf{0.2827 \pm 0.0010}$	3.64 ± 0.18
	Matrix-SVGD	0.7496 ± 0.0071	0.6169 ± 0.0096	0.2955 ± 0.0040	30.55 ± 0.38
	GB-SVGD	0.7455 ± 0.0057	0.5238 ± 0.0069	0.2834 ± 0.0009	$\mathbf{1.34 \pm 0.07}$

4.3 Bayesian Neural Networks

We now consider Bayesian neural network regression as posterior inference over the network weights w, the observation-noise precision γ, and the weight-prior precision λ:

$$p(w, \gamma, \lambda \mid \mathcal{D}) \propto \textstyle\prod_{n=1}^{N_{\text{tr}}} \mathcal{N}\big(y_n \mid f(x_n; w), \gamma^{-1}\big)\, \mathcal{N}(w \mid 0, \lambda^{-1} I)\, p(\gamma)\, p(\lambda), \tag{10}$$

where $f(x; w)$ is the network output, $\mathcal{N}(w \mid 0, \lambda^{-1} I)$ is the prior of w, and $p(\gamma)$, $p(\lambda)$ are the priors of hyper-parameters. Each particle is $\theta = (w, \log \gamma, \log \lambda)$. We evaluate the performance on three tasks, `Concrete`, `Energy`, and `Kin8nm`, from UCI datasets. Each task has input dimension $d_{in} = 8$ and a scalar-valued target. The dimension of particle is $d_\theta = 503$. We maintain a population of $N = 500$ particles and run every method for $2,000$ iterations with learning rate 0.01 and RMSProp step-size adaptation, using the RBF kernel with the median-heuristic bandwidth. See Appendix C.5 for more details of the setting. Table 4 shows that RP-SVGD achieves predictive performance comparable to Vanilla-SVGD while reducing runtime by approximately $11 - 13\%$ across all three datasets. On `Concrete` and `Energy`, RP-SVGD yields slightly higher mean RMSE and NLL than Vanilla-SVGD, whereas on `Kin8nm` it achieves the best mean RMSE (0.0864) and NLL (-1.0335) among all methods. GB-SVGD is substantially faster but has consistently worse predictive performance than both methods. Matrix-SVGD incurs the highest runtime and performs worst on both predictive metrics across all datasets.

Table 4: Performance on BNNs. We report the Mean $\pm$ Standard Deviation over 10 independent trials. For each metric, the best performing method is **bolded** and the second best is underlined. $\uparrow$ indicates higher values are better, and $\downarrow$ indicates lower values are better.

Dataset	Method	Test RMSE $\downarrow$	Test NLL $\downarrow$	Time (s) $\downarrow$
Concrete	Vanilla-SVGD	$\mathbf{5.7515 \pm 0.3979}$	$\mathbf{3.1688 \pm 0.0736}$	48.4 ± 1.2
Concrete	RP-SVGD	5.7713 ± 0.4398	3.1738 ± 0.0821	42.0 ± 0.6
Concrete	GB-SVGD	6.1433 ± 0.4369	3.2549 ± 0.0432	$\mathbf{5.4 \pm 0.5}$
Concrete	Matrix-SVGD	10.3024 ± 0.8734	3.9398 ± 0.0272	55.7 ± 0.7
Energy	Vanilla-SVGD	$\mathbf{1.3974 \pm 0.2502}$	$\mathbf{1.7353 \pm 0.1384}$	47.7 ± 0.5
Energy	RP-SVGD	1.4098 ± 0.3106	1.7556 ± 0.1921	42.3 ± 0.7
Energy	GB-SVGD	2.6527 ± 0.1910	2.4524 ± 0.0463	$\mathbf{5.4 \pm 0.4}$
Energy	Matrix-SVGD	3.3356 ± 0.3491	3.3007 ± 0.0084	56.2 ± 0.9
Kin8nm	Vanilla-SVGD	0.0903 ± 0.0107	-0.9908 ± 0.1136	48.0 ± 0.3
Kin8nm	RP-SVGD	$\mathbf{0.0864 \pm 0.0054}$	$\mathbf{-1.0335 \pm 0.0545}$	42.5 ± 0.6
Kin8nm	GB-SVGD	0.1142 ± 0.0028	-0.7236 ± 0.0144	$\mathbf{5.8 \pm 0.2}$
Kin8nm	Matrix-SVGD	0.1687 ± 0.0048	-0.2058 ± 0.0110	57.0 ± 0.8

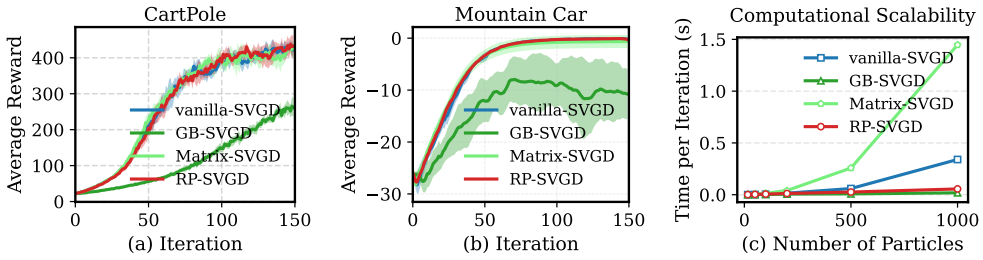

Figure 2: Training dynamics and computational scalability analysis. (a) Average episodic rewards on the `CartPole`. (b) Average episodic rewards on the `MountainCarContinuous-v0`. (c) Comparison of the average wall-clock time required per iteration as the number of particles N increases from 10 to 1000.

4.4 Reinforcement Learning

We assess RP-SVGD on continuous and discrete reinforcement learning (RL) tasks framed as inference problems [22]. Our objective is to sample policy parameters θ from a Boltzmann distribution defined over the policy-parameter space,

$$p(\theta) \propto \exp\left(\frac{J(\theta)}{\alpha}\right) p_0(\theta), \qquad p_0(\theta) \propto 1, \tag{11}$$

where each θ parameterizes a policy $\pi_\theta(a \mid s)$ that induces a trajectory distribution $p_\theta(\tau) = p_0(s_1)\prod_{t=1}^{T}\pi_\theta(a_t \mid s_t)p(s_{t+1} \mid s_t, a_t)$, and

$$J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t, a_t)\right] \tag{12}$$

is the expected return. We set $\alpha = 10.0$ and estimate the policy gradient $\nabla_\theta J(\theta)$ via the REINFORCE algorithm [35]. We evaluate on two environments using an ensemble of $N = 16$ interacting particles: (1) `MountainCarContinuous-v0`, utilizing an MLP with two 16-unit hidden layers (337 parameters), collecting $3,000$ steps per iteration; and (2) `CartPole`, utilizing a 3-hidden-layer MLP (100-50-25 units, $6,877$ parameters), collecting $1,000$ steps per iteration. Both MLPs use $\tanh$ activations, and episodes are truncated at a maximum of 500 steps.

Figures 2(a) and 2(b) illustrate the training dynamics (average episodic reward versus number of iterations) of `CartPole` and `MountainCarContinuous-v0`, respectively. RP-SVGD effectively learns competitive policies, achieving stable, high-reward convergence in both discrete and continuous control settings. In contrast, GB-SVGD exhibits performance degradation and instability, failing to converge to the optimal reward. Figure 2(c) shows the average runtime per iteration as the number of particles N increases from 10 to 1000. The computation time of Vanilla-SVGD increases rapidly with the number of particles, while the computation time of RP-SVGD is grows much more slowly.

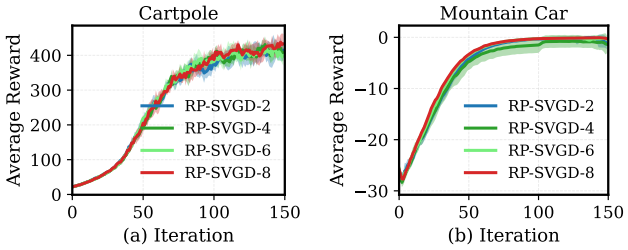

Figure 3: Ablation study on the leaf node capacity $C \in \{2, 4, 6, 8\}$ of RP-SVGD in Reinforcement Learning tasks. (a) Average episodic rewards on `CartPole`. (b) Average episodic rewards on `MountainCarContinuous-v0`. Solid lines represent the mean reward across multiple runs, and shaded regions denote the standard deviation.

These reinforcement learning experiments demonstrate that RP-SVGD provides an alternative for large-scale Bayesian policy optimization, which can be successfully extended to complex neural network architectures without degrading the quality of the learned policies.

Ablation Study on the Choice of Leaf Node Capacity To empirically verify the robustness of RP-SVGD in high-dimensional policy spaces, we conducted an ablation study on the maximum leaf node capacity $C \in \{2, 4, 6, 8\}$. Figure 3 shows the learning curves (relationship between average reward and number of iterations) under two reinforcement learning environments. The results show that RP-SVGD is highly robust to the choice of leaf node capacity. Regardless of whether we use fine-grained local neighborhoods (e.g., $C = 2$ or $C = 4$) or coarse-grained partitions (e.g., $C = 8$), the particles successfully explore the parameter space and consistently converge to stable policy returns in both discrete and continuous control tasks. Additional ablation experiments in Appendix D further confirm this robustness, showing that small changes in C have a negligible impact on the overall performance of the learned policy.

5 Conclusion

This paper proposes a novel SVGD method, RP-SVGD, based on random projection trees to reduce the computational complexity of SVGD. Our method utilizes RP trees to restrict the interaction of the kernel function to a local neighborhood, thereby reducing the complexity of each iteration from $\mathcal{O}(N^2)$ to $\mathcal{O}(Nd(\log(N/C) + C))$.

Experiments on synthetic targets, Bayesian logistic regression, Bayesian neural networks and reinforcement learning show that RP-SVGD attains accuracy comparable to vanilla SVGD at a lower wall-clock cost. Bounding the bias of the hard single-tree algorithm and deriving finite-particle convergence rates are natural directions for future work.

Acknowledgements

Zhuo Sun is supported by Fundamental Research Funds for the Central Universities 2025110590 of Shanghai University of Finance and Economics. This work is supported by the Shanghai Engineering Research Center of Finance Intelligence (Grant No,19DZ2254600).

References

[1] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001.

[2] Andreas Anastasiou, Alessandro Barp, François-Xavier Briol, Bruno Ebner, Robert E Gaunt, Fatemeh Ghaderinezhad, Jackson Gorham, Arthur Gretton, Christophe Ley, Qiang Liu, et al. Stein's method meets computational statistics: A review of some recent developments. *Statistical Science*, 38(1):120–139, 2023.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. Pmlr, 2017.

[4] 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 *International Conference on Learning Representations*, 2021.

[5] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[6] Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[7] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[8] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[9] Nhan Dam, Trung Le, Viet Huynh, and Dinh Phung. Stein variational gradient descent with variance reduction. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[10] Aniket Das and Dheeraj Nagaraj. Provably fast finite particle variants of svgd via virtual particle stochastic approximation. *Advances in Neural Information Processing Systems*, 36:49748–49760, 2023.

[11] Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 537–546, 2008.

[12] Sanjoy Dasgupta and Kaushik Sinha. Randomized partition trees for exact nearest neighbor search. In *Conference on learning theory*, pages 317–337. PMLR, 2013.

[13] Michael F Faulkner and Samuel Livingstone. Sampling algorithms in statistical physics: a guide for statistics and machine learning. *Statistical Science*, 39(1):137–164, 2024.

[14] Masahiro Fujisawa and Futoshi Futami. On the convergence of SVGD in KL divergence via approximate gradient flow. *Transactions on Machine Learning Research*, 2025.

[15] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773, 2012.

[16] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[17] Samory Kpotufe and Sanjoy Dasgupta. A tree-based regressor that adapts to intrinsic dimension. *Journal of Computer and System Sciences*, 78(5):1496–1515, 2012.

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

[19] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

[20] Qiang Liu. Stein variational gradient descent as gradient flow. *Advances in neural information processing systems*, 30, 2017.

[21] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

[22] Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.

[23] Radford M Neal. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, pages 47–95, 2011.

[24] Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 2004.

[25] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[26] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[27] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

[28] Alexander Shapiro. Monte carlo sampling methods. *Handbooks in operations research and management science*, 10:353–425, 2003.

[29] Gábor J Székely and Maria L Rizzo. Energy statistics: A class of statistics based on distances. *Journal of statistical planning and inference*, 143(8):1249–1272, 2013.

[30] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[31] Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.

[32] Dilin Wang, Ziyang Tang, Chandrajit Bajaj, and Qiang Liu. Stein variational gradient descent with matrix-valued kernels. *Advances in neural information processing systems*, 32, 2019.

[33] Dilin Wang, Zhe Zeng, and Qiang Liu. Stein variational message passing for continuous graphical models. In *International Conference on Machine Learning*, pages 5219–5227. PMLR, 2018.

[34] Roger Weber, Hans-Jörg Schek, Stephen Blott, et al. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, pages 194–205, 1998.

[35] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[36] Yang, Duraiswami, and Gumerov. Improved fast gauss transform and efficient kernel density estimation. In *Proceedings ninth IEEE international conference on computer vision*, pages 664–671. IEEE, 2003.

[37] Jingwei Zhuo, Chang Liu, Jiaxin Shi, Jun Zhu, Ning Chen, and Bo Zhang. Message passing stein variational gradient descent. In *International Conference on Machine Learning*, pages 6018–6027. PMLR, 2018.

Appendix

A Notation

Table 5: Key Notation Used in This Paper

Notation	Meaning
	Target Distribution & Particles
N	Total number of interacting particles
d	Ambient dimension of the state space
$\boldsymbol{x}_i \in \mathbb{R}^d$	State/position of the i-th particle
$p(\boldsymbol{x})$	Target posterior probability distribution
$q_t(\boldsymbol{x}), \mu_t$	Empirical particle distribution at iteration t
$\mathbf{s}(\boldsymbol{x})$	Score function of the target density, $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$
ϵ_t	Step size or learning rate at iteration t
	Kernel & SVGD Mechanics
$\mathcal{H}$	Reproducing Kernel Hilbert Space (RKHS)
$k(\boldsymbol{x}, \boldsymbol{y})$	Base positive semi-definite kernel (e.g., Gaussian RBF)
$\phi^*(\boldsymbol{x})$	Optimal transport perturbation direction / Functional gradient
	Random Projection Tree (RP-Tree)
$\mathcal{T}$	Random Projection Tree
D	Depth of the RP-Tree
C	Maximum leaf node capacity
u, L	Internal node and leaf node of the tree, respectively
$\boldsymbol{v}_u, m_u$	Random projection direction and split threshold at node u
ξ	Random variables parameterizing the RP-Tree construction
C_L	Spatial convex polytope defined by leaf node L
$\lvert L_k \rvert$	Number of particles located in leaf node L_k
	Smoothed RP-Tree Kernel & Theory
$\mathbb{I}_{\text{same-leaf}}$	Hard co-location indicator function for the RP-Tree partition
τ	Smoothing parameter for the partition boundary
$H_\tau(\cdot)$	Smooth soft-threshold function (e.g., sigmoid function)
$\tilde{\mathbb{I}}_{\xi_L,\tau}(\cdot)$	Soft membership function for leaf region L
$k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})$	Smoothed RP-Tree kernel for a fixed tree partition ξ
$k_\tau(\boldsymbol{x}, \boldsymbol{y})$	Smoothed effective kernel (expectation over ξ)

B Proof

B.1 Proof of Proposition 1

Proof of Part 1: Positive Semi-Definiteness (PSD) Since k is a PSD kernel on $\mathcal{X}$, by the Moore-Aronszajn theorem, there exists a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathcal{H}}, \ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \tag{13}$$

Substituting the feature map of k into the equation 5:

$$k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) \sum_{L \in \mathcal{L}} \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{x}) \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{y}) \tag{14}$$

$$= \sum_{L \in \mathcal{L}} \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{x}) \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{y}) \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathcal{H}} \tag{15}$$

$$= \sum_{L \in \mathcal{L}} \langle \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{x}) \phi(\boldsymbol{x}), \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{y}) \phi(\boldsymbol{y}) \rangle_{\mathcal{H}}. \tag{16}$$

Let us define a new Hilbert space $\tilde{\mathcal{H}}$ as the direct sum of $|\mathcal{L}|$ copies of $\mathcal{H}$:

$$\tilde{\mathcal{H}} = \bigoplus_{L \in \mathcal{L}} \mathcal{H}. \tag{17}$$

Elements in $\tilde{\mathcal{H}}$ are concatenated vectors (or sequences) of elements from $\mathcal{H}$. The inner product in $\tilde{\mathcal{H}}$ is the sum of the inner products of the components. We define the new feature map $\psi : \mathcal{X} \to \tilde{\mathcal{H}}$ as:

$$\psi(\boldsymbol{x}) := \left(\tilde{\mathbb{I}}_{C_1,\tau}(\boldsymbol{x}) \phi(\boldsymbol{x}), \ldots, \tilde{\mathbb{I}}_{C_{|\mathcal{L}|},\tau}(\boldsymbol{x}) \phi(\boldsymbol{x}) \right) \tag{18}$$

Now, we compute the inner product of $\psi(\boldsymbol{x})$ and $\psi(\boldsymbol{y})$ in $\tilde{\mathcal{H}}$:

$$\langle \psi(\boldsymbol{x}), \psi(\boldsymbol{y}) \rangle_{\tilde{\mathcal{H}}} = \sum_{L \in \mathcal{L}} \langle \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{x}) \phi(\boldsymbol{x}), \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{y}) \phi(\boldsymbol{y}) \rangle_{\mathcal{H}} \tag{19}$$

$$= \sum_{L \in \mathcal{L}} \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{x}) \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{y}) \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathcal{H}} \tag{20}$$

$$= \left(\sum_{L \in \mathcal{L}} \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{x}) \tilde{\mathbb{I}}_{C_L,\tau}(\boldsymbol{y}) \right) k(\boldsymbol{x}, \boldsymbol{y}) \tag{21}$$

$$= k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y}). \tag{22}$$

Since we have explicitly constructed a Hilbert space $\tilde{\mathcal{H}}$ and a feature map ψ such that $k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y}) = \langle \psi(\boldsymbol{x}), \psi(\boldsymbol{y}) \rangle_{\tilde{\mathcal{H}}}$, it follows by definition that $k_{\xi,\tau}$ is a Positive Semi-Definite (PSD) kernel. □

Proof of Part 2: Smoothness By definition (Section 3.1), the smooth soft-threshold function $H_\tau(\cdot)$ belongs to C^∞. Inside the function, the linear projection $\boldsymbol{v}_{L,j}^\top \boldsymbol{x} - m_{L,j}$ is also infinitely differentiable (C^∞) with respect to $\boldsymbol{x}$. Because the composition and finite multiplication of C^∞ functions remain C^∞, the soft membership function $\tilde{\mathbb{I}}_{\xi_L,\tau}(\boldsymbol{x}) = \prod_{j=1}^{D} H_\tau((-1)^{b_{L,j}}(\boldsymbol{v}_{L,j}^\top \boldsymbol{x} - m_{L,j}))$ is smooth and belongs to $C^\infty(\mathcal{X})$. Consequently, the scaling factor $f(\boldsymbol{x}, \boldsymbol{y}) := \sum_{L \in \mathcal{L}_\mathcal{T}} \tilde{\mathbb{I}}_{\xi_L,\tau}(\boldsymbol{x}) \tilde{\mathbb{I}}_{\xi_L,\tau}(\boldsymbol{y})$ is C^∞ with respect to both arguments. If the base kernel $k(\boldsymbol{x}, \boldsymbol{y})$ is C^m-differentiable, their product $k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{x}, \boldsymbol{y})$ guarantees that $k_{\xi,\tau} \in C^m(\mathcal{X} \times \mathcal{X})$. □

Proof of Part 3: Stein Identity To prove that $k_{\xi,\tau}$ belongs to the Stein class of p, we need to show that for any fixed $\boldsymbol{y}$, the expectation of the Stein operator applied to $k_{\xi,\tau}(\cdot, \boldsymbol{y})$ under $p(\boldsymbol{x})$ evaluates to zero. Let $g(\boldsymbol{x}) := k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})$. We evaluate the expectation:

$$\mathbb{E}_{\boldsymbol{x} \sim p}[\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) g(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} g(\boldsymbol{x})].$$

Using the identity $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) = \frac{\nabla_{\boldsymbol{x}} p(\boldsymbol{x})}{p(\boldsymbol{x})}$, the term inside the expectation can be rewritten as:

$$\int_{\mathcal{X}} \left(\frac{\nabla_{\boldsymbol{x}} p(\boldsymbol{x})}{p(\boldsymbol{x})} g(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} g(\boldsymbol{x}) \right) p(\boldsymbol{x}) d\boldsymbol{x} = \int_{\mathcal{X}} \left(g(\boldsymbol{x}) \nabla_{\boldsymbol{x}} p(\boldsymbol{x}) + p(\boldsymbol{x}) \nabla_{\boldsymbol{x}} g(\boldsymbol{x}) \right) d\boldsymbol{x}$$

$$= \int_{\mathcal{X}} \nabla_{\boldsymbol{x}} (p(\boldsymbol{x}) g(\boldsymbol{x})) d\boldsymbol{x}.$$

Applying the Gradient Theorem (a corollary of the Divergence Theorem for scalar fields), this volume integral of a gradient can be converted into a surface integral over the boundary $\partial\mathcal{X}$ of the domain:

$$\int_{\mathcal{X}} \nabla_{\boldsymbol{x}}(p(\boldsymbol{x})g(\boldsymbol{x}))d\boldsymbol{x} = \oint_{\partial\mathcal{X}} p(\boldsymbol{x})g(\boldsymbol{x})\mathbf{n}dS, \tag{23}$$

where $\mathbf{n}$ is the outward-pointing unit normal vector on $\partial\mathcal{X}$. Note that this is a scalar multiplication with the vector $\mathbf{n}$, not a dot product. For unconstrained domains like $\mathcal{X} = \mathbb{R}^d$, the boundary is at infinity ($\|\boldsymbol{x}\| \to \infty$).

By the definition of the Stein class [21], i.e., assuming mild zero conditions on g: $p(\boldsymbol{x})g(\boldsymbol{x}) = 0$ for $\forall \boldsymbol{x} \in \partial\mathcal{X}$), the base kernel $k(\boldsymbol{x}, \boldsymbol{y})$ ensures that the surface integral vanishes over the boundary of the unconstrained domain despite the $O(r^{d-1})$ growth of the surface area:

$$\lim_{r\to\infty} \oint_{\|\boldsymbol{x}\|=r} p(\boldsymbol{x})k(\boldsymbol{x}, \boldsymbol{y})\mathbf{n}dS = \mathbf{0}. \tag{24}$$

Notice that $g(\boldsymbol{x}) = k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y})f(\boldsymbol{x}, \boldsymbol{y})$. Because the soft-threshold functions $H_\tau \in (0, 1)$, the soft membership term $\tilde{\mathbb{I}}_{\xi_L,\tau}(\boldsymbol{x})$ is strictly bounded between 0 and 1. As a result, the scaling factor $f(\boldsymbol{x}, \boldsymbol{y})$ is a uniformly bounded scalar function ($0 < f(\boldsymbol{x}, \boldsymbol{y}) \leq 1$ since it represents a soft partition of unity).

Because multiplying an integrable function whose boundary integral vanishes by a uniformly bounded scalar function strictly preserves the zero-limit property, the asymptotic decay of the surface integral holds for the smoothed kernel as well:

$$\lim_{r\to\infty} \oint_{\|\boldsymbol{x}\|=r} p(\boldsymbol{x})k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})\mathbf{n}dS = \mathbf{0}. \tag{25}$$

Because this boundary integral evaluates to exactly zero, we conclude that:

$$\mathbb{E}_{\boldsymbol{x}\sim p}[\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y}) + \nabla_{\boldsymbol{x}} k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})] = \mathbf{0}. \tag{26}$$

Thus, $k_{\xi,\tau}$ is a valid kernel in the Stein class of p. □

B.2 Verification of Convergence Assumptions for the Effective Kernel

As discussed in Section 3.2, the expected dynamics of RP-SVGD are governed by the effective kernel $k_\tau(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}_\xi[k_{\xi,\tau}(\boldsymbol{x}, \boldsymbol{y})]$. By definition, this effective kernel can be factorized as:

$$k_\tau(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) \cdot C_\tau(\boldsymbol{x}, \boldsymbol{y}), \tag{27}$$

where $k(\boldsymbol{x}, \boldsymbol{y})$ is the base kernel (e.g., Gaussian RBF), and $C_\tau(\boldsymbol{x}, \boldsymbol{y}) := \mathbb{E}_\xi\left[\sum_{L\in\mathcal{L}_\mathcal{T}} \tilde{\mathbb{I}}_{\xi_L,\tau}(\boldsymbol{x})\tilde{\mathbb{I}}_{\xi_L,\tau}(\boldsymbol{y})\right]$ is the expected soft co-membership probability.

We relate the idealized RP-SVGD dynamics to the approximate gradient flow (AGF) framework of [14], who prove sub-linear convergence in Kullback–Leibler (KL) divergence of the discrete-time SVGD update with infinitely many particles. According to their *Theorem 1*, SVGD achieves convergence if the driving kernel satisfies the following key assumptions. We formally list these assumptions and verify that our effective kernel k_τ satisfies each of them.

Assumption 1 (Smoothness and Boundedness [14]) *The feature map $\nabla k(\cdot, \boldsymbol{x})$ is continuous. Moreover, the kernel and its first- and second-order derivatives must be uniformly bounded, i.e., there exists a constant $B > 0$ such that $\|k(\cdot, \boldsymbol{x})\|_\mathcal{H} \leq B$, and the norms of its derivatives are bounded by B^2.*

Verification for k_τ: Preserved. Assume the base kernel k satisfies Assumption 1 with bounded derivatives. By design, our soft co-membership function $C_\tau(\boldsymbol{x}, \boldsymbol{y})$ is constructed using the soft-threshold function $H_\tau(z) = \sigma(-z/\tau)$. For any smoothing temperature $\tau > 0$, H_τ is C^∞ smooth, and its output is strictly bounded in $(0, 1)$. Because C_τ is the expectation of the product of these smooth and bounded functions, C_τ itself is C^∞ smooth, strictly bounded ($0 < C_\tau \leq 1$), and possesses uniformly bounded derivatives. According to the product rule of calculus, the element-wise product $k_\tau = k \cdot C_\tau$ of two bounded functions with bounded derivatives yields a function that is also bounded and has bounded derivatives (bounded by a new constant B' depending on B and τ). Thus, k_τ preserves the smoothness and boundedness assumption.

Assumption 2 (Integrally Strictly Positive Definite (ISPD) [14]) *The kernel k must be integrally strictly positive definite, meaning that $\iint k(\boldsymbol{x}, \boldsymbol{y}) d\rho(\boldsymbol{x}) d\rho(\boldsymbol{y}) > 0$ holds for any finite non-zero signed Borel measure ρ.*

Verification for k_τ: Preserved. This is a critical assumption, as the ISPD property guarantees that projecting the true Wasserstein gradient onto the RKHS introduces no structural asymptotic bias ($\delta = 0$ in the AGF framework). Assume the chosen base kernel k is ISPD. We observe that the expected co-membership term $C_\tau(\boldsymbol{x}, \boldsymbol{y})$ is inherently a valid Positive Semi-Definite (PSD) kernel, because it is defined as an inner product over the random parameter space Ξ: $C_\tau(\boldsymbol{x}, \boldsymbol{y}) = \langle \tilde{\mathbb{I}}_{\cdot,\tau}(\boldsymbol{x}), \tilde{\mathbb{I}}_{\cdot,\tau}(\boldsymbol{y}) \rangle_{L^2(\Xi)}$. Furthermore, since the soft-threshold function strictly satisfies $H_\tau > 0$, the probability of soft co-membership is strictly positive everywhere, i.e., $C_\tau(\boldsymbol{x}, \boldsymbol{y}) > 0$. According to the closure properties of positive semi-definite kernels under pointwise multiplication (often referred to as the continuous analogue of the Schur Product Theorem), the element-wise (Hadamard) product of an ISPD kernel (k) and a strictly positive PSD kernel (C_τ) guarantees that the resulting kernel is also ISPD. Consequently, k_τ preserves the ISPD property of the base kernel, completely satisfying Assumption 2.

Assumption 3 (Bounded Trace [14]) *The trace of the kernel is bounded for any probability measure $\mu \in \mathcal{P}_2(\mathcal{X})$, i.e., $\int k(\boldsymbol{x}, \boldsymbol{x}) d\mu(\boldsymbol{x}) < \infty$.*

Verification for k_τ: Preserved. Assume the base kernel k has a bounded trace, $\int k(\boldsymbol{x}, \boldsymbol{x}) d\mu(\boldsymbol{x}) < \infty$. For the soft co-membership probability, a particle is always evaluated against itself in the same identical spatial location, yielding the maximum possible assignment weight. Because the membership functions are bounded by 1, it holds that $0 < C_\tau(\boldsymbol{x}, \boldsymbol{x}) \leq 1$. Therefore, $k_\tau(\boldsymbol{x}, \boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) C_\tau(\boldsymbol{x}, \boldsymbol{x}) \leq k(\boldsymbol{x}, \boldsymbol{x})$. Integrating both sides with respect to any probability measure μ yields $\int k_\tau(\boldsymbol{x}, \boldsymbol{x}) d\mu(\boldsymbol{x}) \leq \int k(\boldsymbol{x}, \boldsymbol{x}) d\mu(\boldsymbol{x}) < \infty$. The bounded trace property is strictly preserved.

Assumption 4 (Strictly Positive Eigenvalue Bounds [14]) *The eigenvalues $\{\lambda_i\}$ of the Hilbert-Schmidt integral operator $P_{\mu_t,k}$ are of constant order with respect to time t and are strictly positive; that is, there exist upper and lower bounds for $\{\lambda_i\}$ that are independent of t and strictly greater than 0.*

Verification for k_τ: Inherited as a standard premise. Assumption 4 is a sophisticated analytic condition required to ensure that the relative approximation error ϵ_t in the AGF framework can be bounded by a time-independent positive constant. In [14], it is acknowledged that formally proving this for standard characteristic kernels (like RBF) in an infinite-particle setting is analytically intractable, but it serves as a foundational working premise supported by empirical evidence. Because our effective kernel k_τ structurally operates as a geometry-adaptive modulation of the base kernel (preserving ISPD and infinite differentiability), it shares the fundamental spectral characteristics of standard universal kernels. Therefore, we follow the identical theoretical protocol and formally adopt Assumption 4 for the integral operator induced by k_τ.

C Experimental Details

C.1 Hyperparameter Tuning

To ensure a fair comparison across all baselines, we independently tuned the hyperparameters for each method. For all SVGD-based variants, we performed a grid search over the learning rate $\eta \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}$. The RBF kernel bandwidth h was adaptively determined using the median heuristic across all tasks. For our proposed RP-SVGD, we tuned the maximum leaf node capacity C, which controls the spatial partitioning granularity and the average leaf node size. For GB-SVGD, the subpopulation (or mini-batch) size B was tuned based on the recommendations from [2]: for 2D synthetic tasks and Bayesian logistic regression (BLR), we searched $B \in \{10, 20, 30, 40\}$; for reinforcement learning (RL) tasks, we set B to match the value of C used in the respective experiments. Regarding priors, we adhered to standard practices: we employed an isotropic Gaussian prior $\mathcal{N}(0, I)$ for BLR tasks, and a non-informative flat prior for RL experiments. A comprehensive list of the selected hyperparameters is provided in Table 6.

Table 6: Full Hyperparameter Overview across All Experimental Tasks.

Task	**Dim.**	h	**Vanilla**	**Matrix**	**GB-SVGD**		**RP-SVGD (Ours)**	
			η	η	η	B	η	C
2D Toy Examples (Particles $N = 500$)								
Two-Rings	2	med	0.05	0.05	0.005	40	0.1	50
GMM-8	2	med	0.01	0.05	0.01	40	0.1	50
GMM-25	2	med	0.005	0.1	0.1	40	0.1	50
Swiss-Roll	2	med	0.01	0.01	0.01	40	0.1	50
Bayesian Logistic Regression (Particles $N = 500$)								
BreastCancer	30	med	0.01	0.05	0.01	40	0.1	50
Ionosphere	34	med	0.01	0.1	0.1	40	0.1	50
German	20	med	0.05	0.05	0.005	40	0.1	50
Covtype	54	med	0.01	0.01	0.01	40	0.1	50
Bayesian Neural Networks (Particles $N = 1000$)								
Concrete	8	med	0.01	0.05	0.01	40	0.1	50
Energy	8	med	0.01	0.1	0.1	40	0.1	50
Kin8nm	8	med	0.05	0.05	0.005	40	0.1	50
Reinforcement Learning(Particles $N = 16$)								
CartPole	6877	med	0.01	0.01	0.01	8	0.01	8
MountainCar	337	med	0.01	0.01	0.01	8	0.01	8

Hyperparameter Legend:
Common: h is the kernel bandwidth, adaptively set using the median heuristic.
η: For vanilla SVGD, GB-SVGD, and RP-SVGD, we utilize the RMSProp optimizer; for Matrix-SVGD, we employ Standard Gradient Descent (SGD).
GB-SVGD: B is the particle sub-population size.
RP-SVGD: C is the Maximum leaf capacity generated by the RP tree.
Note: Learning rates were independently optimized for each task to ensure a fair comparison of convergence speed.

C.2 Mathematical Expressions of Synthetic Examples

We evaluate our proposed method on four synthetic 2D target distributions $\pi(\boldsymbol{x})$ with $\boldsymbol{x} \in \mathbb{R}^2$. These distributions are designed to test the model's ability to handle multi-modality, non-linear manifolds, and varying density scales.

Gaussian Mixture Model 8 The GMM-8 distribution is a mixture of $K = 8$ isotropic Gaussians whose means are uniformly arranged on a circle of radius $R = 6.0$. The density is given by:

$$p(\boldsymbol{x}) = \frac{1}{K} \sum_{i=1}^{K} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_i, \sigma \mathbf{I}), \quad \text{with} \quad \boldsymbol{\mu}_i = [R \cos \theta_i, R \sin \theta_i]^\top,$$

where $\theta_i = \frac{2\pi(i-1)}{K}$ for $i = 1, \ldots, K$. Note that the covariance is scaled by σ.

Gaussian Mixture Model 25 The GMM-25 distribution follows a 5×5 grid arrangement of 25 Gaussian modes. The density is defined as:

$$p(\boldsymbol{x}) = \frac{1}{K} \sum_{i=1}^{5} \sum_{j=1}^{5} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{i,j}, \sigma^2 \mathbf{I}),$$

where $K = 25$, and the means $\boldsymbol{\mu}_{i,j} = [s_i, s_j]^\top$ are grid points with $s \in \{-10, -5, 0, 5, 10\}$. This distribution tests the model's ability to capture numerous high-density modes.

Two Rings The Two Rings distribution consists of two concentric circular manifolds with additive Gaussian noise. The density is defined by the radial distance $r = \|\boldsymbol{x}\|_2$:

$$p(\boldsymbol{x}) = \frac{1}{Z} \sum_{i=1}^{2} \exp\left(-\frac{(\|\boldsymbol{x}\|_2 - R_i)^2}{2\sigma^2} \right),$$

where $R_1 = 5.0, R_2 = 12.0$ are the radii of the rings, σ is the noise scale, and Z is the normalization constant.

Swiss Roll The Swiss Roll distribution [25] represents a probability density concentrated along an Archimedean spiral manifold. Let $\mathcal{C} = \{(t \cos t, t \sin t)^\top \mid t \in [1.5, 8.5]\}$ be the skeleton of the manifold. The density is defined as:

$$p(\boldsymbol{x}) \propto \exp\left(-\frac{d(\boldsymbol{x}, \mathcal{C})^2}{2\sigma^2} \right),$$

where $d(\boldsymbol{x}, \mathcal{C}) = \inf_{\mathbf{p} \in \mathcal{C}} \|\boldsymbol{x} - \mathbf{p}\|_2$ denotes the Euclidean distance from $\boldsymbol{x}$ to the curve $\mathcal{C}$. In practice, $\mathcal{C}$ is approximated by a set of $M = 200$ equidistant points along the spiral.

Ground Truth Acquisition To evaluate the quality of the particle sets, we generate 1000 ground truth samples for each task. For the Gaussian Mixture distributions, samples are drawn directly from the closed-form density functions. For the Two Rings and Swiss Roll problems, we utilize the generative procedures defined by their respective manifold parameterizations to obtain exact samples.

C.3 Evaluation Metrics

To quantitatively assess the discrepancy between the generated particle set $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N_g}$ and the reference target set $\mathcal{Y} = \{\boldsymbol{y}_j\}_{j=1}^{N_r}$, we employ the following statistical metrics:

Maximum Mean Discrepancy (MMD). The Maximum Mean Discrepancy [15] measures the distance between two probability distributions embedded in a Reproducing Kernel Hilbert Space (RKHS). We utilize the Gaussian RBF kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 / 2h^2)$, where the bandwidth h is dynamically determined via the median heuristic. The empirical squared MMD is estimated as:

$$\widehat{\mathrm{MMD}}^2(\mathcal{X}, \mathcal{Y}) = \frac{1}{N_g^2} \sum_{i=1}^{N_g} \sum_{i'=1}^{N_g} k(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) + \frac{1}{N_r^2} \sum_{j=1}^{N_r} \sum_{j'=1}^{N_r} k(\boldsymbol{y}_j, \boldsymbol{y}_{j'}) - \frac{2}{N_g N_r} \sum_{i=1}^{N_g} \sum_{j=1}^{N_r} k(\boldsymbol{x}_i, \boldsymbol{y}_j).$$

Energy Distance (ED). The Energy Distance [29] is a statistical distance that characterizes the geometric discrepancy of the sample space based on pairwise Euclidean distances. It is computed as:

$$\widehat{\mathrm{ED}}(\mathcal{X}, \mathcal{Y}) = \frac{2}{N_g N_r} \sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2 - \frac{1}{N_g^2} \sum_{i=1}^{N_g} \sum_{i'=1}^{N_g} \|\boldsymbol{x}_i - \boldsymbol{x}_{i'}\|_2 - \frac{1}{N_r^2} \sum_{j=1}^{N_r} \sum_{j'=1}^{N_r} \|\boldsymbol{y}_j - \boldsymbol{y}_{j'}\|_2.$$

Kullback-Leibler (KL) Divergence. To estimate the Kullback-Leibler divergence [18] for low-dimensional tasks, we partition the spatial domain into $B \times B$ equal-sized bins. Let P and Q be the normalized histogram densities of the generated and target samples, respectively. The empirical KL divergence is then computed as:

$$D_{\mathrm{KL}}(P \| Q) = \sum_{b=1}^{B^2} P_b \log\left(\frac{P_b + \epsilon}{Q_b + \epsilon} \right),$$

where $\epsilon > 0$ is a small constant introduced to ensure numerical stability.

Wasserstein Distance (W_2). To efficiently approximate the optimal transport cost, we evaluate the 2-Wasserstein distance across 1D marginal distributions [3]. Denoting the d-th marginal distributions of $\mathcal{X}$ and $\mathcal{Y}$ as $P^{(d)}$ and $Q^{(d)}$, respectively, the metric over the D-dimensional space is defined as:

$$W_2^{\text{1D}}(\mathcal{X}, \mathcal{Y}) = \sqrt{\sum_{d=1}^{D} W_2^2(P^{(d)}, Q^{(d)})}.$$

Precision and Recall. These two metrics [19] are utilized to disentangle the approximation fidelity from mode coverage. Let $\rho_k(\mathbf{z}, \mathcal{Z})$ denote the Euclidean distance from a given point $\mathbf{z}$ to its k-th nearest neighbor within the set $\mathcal{Z}$. Precision (fidelity) measures the fraction of generated samples that fall within the estimated support of the target distribution:

$$\text{Precision} = \frac{1}{N_g} \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{1}\left[\exists \boldsymbol{y} \in \mathcal{Y} : \|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq \rho_k(\boldsymbol{y}, \mathcal{Y})\right].$$

Conversely, recall (diversity) evaluates the fraction of the target distribution's support that is successfully covered by the generated samples:

$$\text{Recall} = \frac{1}{N_r} \sum_{\boldsymbol{y} \in \mathcal{Y}} \mathbb{1}\left[\exists \boldsymbol{x} \in \mathcal{X} : \|\boldsymbol{y} - \boldsymbol{x}\|_2 \leq \rho_k(\boldsymbol{x}, \mathcal{X})\right].$$

Quantitative Convergence Criterion. To determine the convergence iterations and their corresponding wall-clock times reported in Figure 1, we employ a gradient-based plateau detection heuristic on the MMD trajectories. Let M_t denote the evaluated metric (e.g., MMD) at iteration t. We define the convergence point as the first iteration t^* where the moving average of the absolute step-wise differences over a sliding window of W iterations falls strictly below a predefined tolerance threshold δ. Mathematically, convergence is reached at t^* when:

$$\frac{1}{W} \sum_{k=0}^{W-1} |M_{t^*-k} - M_{t^*-k-1}| < \delta.$$

In our evaluations, we set the sliding window size to $W = 10$ and the tolerance threshold to $\delta = 10^{-5}$. The vertical dashed lines in the convergence plots and the reported wall-clock times correspond precisely to this analytically detected iteration t^*.

To quantitatively evaluate the predictive performance of the Bayesian Logistic Regression models, we compute three standard classification metrics on the held-out test set. Let $\mathcal{D}_{\text{test}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ denote the test set containing n instances, where $y_i \in \{0, 1\}$ is the true binary label. In our particle-based variational inference framework, the posterior predictive probability $\hat{p}_i \triangleq p(y_i = 1 \mid \boldsymbol{x}_i, \mathcal{D}_{\text{train}})$ is approximated by marginalizing over the empirical posterior ensemble of P particles: $\hat{p}_i \approx \frac{1}{P} \sum_{k=1}^{P} \sigma(\boldsymbol{\theta}_k^\top \boldsymbol{x}_i)$, where $\sigma(\cdot)$ is the sigmoid function.

Based on this ensemble predictive probability, we evaluate the following metrics:

Classification Accuracy (ACC). Accuracy measures the proportion of correctly classified test instances under a strict decision boundary. An instance is assigned to the positive class if its ensemble predictive probability is greater than or equal to 0.5:

$$\text{ACC} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[\,\hat{p}_i \geq 0.5 \iff y_i = 1\,]. \tag{28}$$

Negative Log-Likelihood (NLL). NLL evaluates the probabilistic calibration and the quality of the predictive uncertainty. Unlike ACC, which utilizes a hard threshold, NLL strictly penalizes overconfident incorrect predictions. It is computed as the average cross-entropy loss over the test set:

$$\text{NLL} = -\frac{1}{n} \sum_{i=1}^{n} \Big[\, y_i \log(\hat{p}_i + \varepsilon) + (1 - y_i) \log(1 - \hat{p}_i + \varepsilon) \Big], \tag{29}$$

where a small constant $\varepsilon = 10^{-12}$ is introduced to ensure numerical stability and prevent undefined logarithms in cases of extreme predictive confidence.

Root Mean Square Error (RMSE). In the context of binary classification, the squared error evaluates the Brier score, which captures both the accuracy and the calibration of the probabilistic predictions. We report the Root Mean Square Error, defined as the average Euclidean distance between the true binary labels and the continuous predictive probabilities:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{p}_i\big)^2}. \tag{30}$$

Lower values of NLL and RMSE indicate that the Bayesian posterior effectively captures the true data distribution with well-calibrated uncertainty estimates.

C.4 Bayesian Logistic Regression

We consider the Bayesian Logistic Regression model for binary classification tasks. Let $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^{|\mathcal{D}|}$ denote a dataset comprising feature vectors $\boldsymbol{x}_j \in \mathbb{R}^D$ and corresponding binary labels $y_j \in \{0, 1\}$. The posterior distribution of interest over the weight vector $\boldsymbol{\theta} \in \mathbb{R}^D$ is given by Bayes' theorem:

$$p(\boldsymbol{\theta} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta}).$$

The likelihood function is defined as a product of independent Bernoulli trials:

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{j=1}^{N}\left[\sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_j)^{y_j}\big(1 - \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_j)\big)^{1-y_j}\right],$$

where $\sigma(z) = 1/(1 + \exp(-z))$ denotes the standard sigmoid function. Following the established benchmark settings [21], we impose a non-informative standard isotropic Gaussian prior on the model weights, i.e., $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

We evaluate the inference quality of three SVGD variants (Vanilla SVGD, GB-SVGD, and RP-SVGD) across four standard UCI/OpenML datasets: Breast Cancer ($D = 30$), Ionosphere ($D = 34$), German Credit ($D = 20$), and Covtype ($D = 54$). To fit the binary logistic regression framework, we explicitly map the target variables to $y_j \in \{0, 1\}$. Specifically: Breast Cancer, Ionosphere, and German Credit are inherently binary classification tasks (e.g., predicting malignant vs. benign tumors, or good vs. bad credit risks). Their original categorical or $\{-1, 1\}$ labels are mapped to $\{0, 1\}$. Covtype is originally a 7-class forest cover type prediction dataset. Following standard evaluation protocols [21], we binarize the task by predicting the most frequent class (Lodgepole Pine) against all other classes. Due to its massive scale, we uniformly subsample $50,000$ instances for our experiments. Furthermore, to ensure the numerical stability of the radial basis function (RBF) kernel and the gradient updates, all continuous input features $\boldsymbol{x}_j$ are standardized to have zero mean and unit variance across the training set.

For all evaluated methods, the continuous posterior is approximated using a set of $N = 500$ interacting particles $\{\boldsymbol{\theta}_i\}_{i=1}^{N}$. All particles are independently initialized from the prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. To ensure computational efficiency during the evaluation of the score function $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid \mathcal{D})$, we employ unbiased stochastic gradient estimates. Specifically, at each iteration, the log-likelihood gradient is evaluated using stochastic or full-batch updates depending on the dataset scale. For larger datasets (German Credit and Covtype), we randomly sample a mini-batch of size $B = 500$ without replacement, scaling the log-likelihood by the ratio N_{train}/B to maintain an unbiased estimate with respect to the full training set. For smaller datasets (Breast Cancer and Ionosphere), where the 80% training split contains fewer than 500 instances, we evaluate the exact full-batch gradient over the entire training set without scaling.

For each dataset, we randomly partition the data into an 80% training set and a 20% held-out test set. Predictive performance on the test set is quantified using classification Accuracy, Negative Log-Likelihood (NLL) and RMSE. To obtain statistically robust results and eliminate the influence of initial conditions, each experiment is repeated over 10 independent random trials. Crucially, within a given trial, all SVGD variants are evaluated using the exact same random data splits and identical initial particle configurations.

C.5 Bayesian Neural Network

We consider Bayesian neural network regression as posterior inference over the network weights w, the observation-noise precision γ, and the weight-prior precision λ:

$$p(w, \gamma, \lambda \mid \mathcal{D}) \propto \prod_{n=1}^{N_{\mathrm{tr}}} \mathcal{N}\big(y_n \mid f(x_n; w), \gamma^{-1}\big)\, \mathcal{N}(w \mid 0, \lambda^{-1} I)\, p(\gamma)\, p(\lambda),$$

where $f(x; w)$ is the network output and $p(\gamma) = \mathrm{Gamma}(\gamma \mid 1, 0.1)$, $p(\lambda) = \mathrm{Gamma}(\lambda \mid 1, 0.1)$. Each particle is $\theta = (w, \log\gamma, \log\lambda)$, with the Jacobian of the log-transformation included in $\log p(\theta \mid \mathcal{D})$. The score $\nabla_\theta \log p(\theta \mid \mathcal{D})$ is unbiasedly estimated at every iteration from a random mini-batch of 100 training points, with the log-likelihood rescaled by $N_{\mathrm{tr}}/100$.

Tasks: `Concrete`, `Energy`, and `Kin8nm` (UCI Regression). The three datasets (OpenML ids 44959, 44960, and 189) contain $1,030$, 768, and $8,192$ instances, respectively, each with input dimension $d_{in} = 8$ and a scalar target. The regressor is an MLP with one hidden layer of 50 ReLU units and a linear output, so that the exact particle dimension, including the two log-precisions, is $d_\theta = (8 \times 50 + 50) + (50 \times 1 + 1) + 2 = 503$. We maintain a population of $N = 500$ particles and run every method for $2,000$ iterations with learning rate 0.01 and RMSProp step-size adaptation, using the RBF kernel with the median-heuristic bandwidth. RP-SVGD uses a leaf capacity of $C = 50$ (tree depth $\lceil \log_2(N/C) \rceil = 4$), GB-SVGD a subgroup of 40 particles per iteration, and Matrix-SVGD [32] the inverse particle covariance as its constant preconditioner with plain gradient steps.

Evaluation Protocol. Each dataset is randomly split into 90% training and 10% test data over 10 independent trials, and all methods share the same initial particles within a trial. Inputs and targets are standardized with training-set statistics, and all metrics are computed on the original target scale: the test RMSE of the particle-averaged prediction, the test NLL under the equally weighted Gaussian-mixture predictive density $\frac{1}{N}\sum_p \mathcal{N}\big(y \mid f_p(x), \sigma_y^2/\gamma_p\big)$, where σ_y is the training-set standard deviation of the targets, and the wall-clock time of the full run (single-precision PyTorch on an AMD Ryzen Threadripper PRO 5995WX CPU). We report mean $\pm$ standard deviation over the 10 trials.

C.6 Reinforcement Learning

Control as Inference Framework. In line with the experimental protocol of Stein Variational Policy Gradient [22], we frame policy optimization as a Bayesian inference task. The goal is to sample interacting policy parameters θ from a target Boltzmann distribution defined over the high-dimensional parameter space:

$$p(\theta) \propto \exp(J(\theta)/\alpha) p_0(\theta), \quad J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} r(s_t, a_t) \right],$$

where $J(\theta)$ is the expected episodic return, $p_\theta(\tau)$ is the trajectory distribution induced by policy π_θ, and $p_0(\theta)$ is a prior distribution. Consistent with standard implementations, we assume a non-informative flat prior $p_0(\theta) \propto 1$, effectively inducing the target density from the expected reward landscape. We set the temperature parameter $\alpha = 10.0$ across all variants.

To evolve the particles via SVGD and our proposed RP-SVGD, the required score function $\nabla_\theta \log p(\theta) = \frac{1}{\alpha}\nabla_\theta J(\theta)$ is unbiasedly estimated using the REINFORCE algorithm [35]. The particles (representing distinct policies) act independently in the environment to collect on-policy trajectories for gradient estimation.

Task 1: `MountainCarContinuous-v0` (Continuous Control). The Mountain Car task has a continuous state space (observation dimension $D_{obs} = 2$) and a continuous action space (action dimension $D_{act} = 1$). We parameterize the policy using a Multi-Layer Perceptron (MLP) with two hidden layers, each containing 16 units with $\tanh$ activations. The final layer outputs the mean μ of the continuous action. The exact parameter dimension of this policy network is $d_\theta = (2 \times 16 + 16) + (16 \times 16 + 16) + (16 \times 1 + 1) = 337$. For this environment, each particle collects a batch of $3,000$ environment steps per iteration, and the maximum length of an episode is truncated at 500 steps. For the training dynamics reported in Figures 2 and 3, we maintain a population of $N = 16$ policy particles.

Task 2: `CartPole` **(Discrete Control).** The CartPole task has a continuous state space (observation dimension $D_{obs} = 4$) and a discrete action space (action dimension $D_{act} = 2$). The policy is parameterized by a deeper MLP with three hidden layers of sizes 100, 50, and 25, respectively, utilizing $\tanh$ activations. The output layer produces the unnormalized logits for the discrete actions. The exact parameter dimension of this network is substantially higher: $d_\theta = (4 \times 100 + 100) + (100 \times 50 + 50) + (50 \times 25 + 25) + (25 \times 2 + 2) = 6,877$. For CartPole, each particle collects a batch of $1,000$ environment steps per iteration, with the maximum episode length similarly capped at 500 steps. The particle ensemble size is also set to $N = 16$.

C.7 Compute Resources and Hardware Implications

Hardware Specifications. Most experiments reported in this paper, including the main empirical simulations and wall-clock profiling, were conducted locally on an Apple MacBook Air equipped with an M4 chip (Apple Silicon, ARM architecture). Unless otherwise specified, the algorithms were executed primarily on the CPU. Additional Bayesian neural network experiments and supplementary CPU ablations were conducted on an AMD Ryzen Threadripper PRO 5995WX CPU, while the GPU scalability experiments were performed on an NVIDIA RTX A6000 GPU. The hardware used for each supplementary experiment is specified in the corresponding table caption.

Discussion on CPU vs. GPU Efficiency. The significant computational speedups observed in our experiments (e.g., Table 2 and Table 3) reflect the performance characteristics on a CPU-centric, unified-memory hardware setup. We find it necessary to provide a nuanced discussion regarding how hardware architectures may influence the comparative advantage of RP-SVGD:

1. The Nature of RP-SVGD (Tree Operations): RP-SVGD dramatically reduces the asymptotic computational complexity from $\mathcal{O}(N^2 d)$ to $\mathcal{O}(Nd \log(N/C) + NCd)$ by replacing dense all-to-all interactions with dynamic Random-Projection Tree construction and leaf-constrained interactions. However, constructing and traversing spatial trees inherently involves recursive branching, irregular memory access patterns, and pointer chasing. These operations are highly efficient on modern CPUs (such as the M4 chip) due to sophisticated branch prediction and deep cache hierarchies, but they are notoriously challenging to parallelize efficiently on standard GPUs (e.g., NVIDIA CUDA architectures) due to thread divergence.

2. The Nature of Vanilla SVGD (Dense Matrices): Conversely, the $\mathcal{O}(N^2)$ kernel evaluation in Vanilla SVGD, while asymptotically expensive, relies entirely on dense matrix multiplications and pairwise distance computations. These operations are extremely amenable to Single Instruction Multiple Data (SIMD) execution and can fully saturate the massive parallel computing cores of modern high-end GPUs.

Conclusion on Hardware: Therefore, our reported wall-clock speedups best represent scenarios where inference is deployed on edge devices, personal computers, or standard CPU servers, where memory and compute constraints render the $O(N^2)$ cost of Vanilla SVGD prohibitive. For ultra-large-scale scenarios running on high-end cluster GPUs, Vanilla SVGD might leverage dense matrix optimization to close the time gap, whereas optimizing RP-SVGD for GPU execution (e.g., via flattened array-based tree structures or batched query processing) remains a promising avenue for future system-level research.

D Additional Results and Ablation Studies

The experiments under various settings in this section are based on independent re-runs using different random seeds; consequently, the results may differ from those previously reported. Within the same table or figure, identical random seeds are used for the different methods to facilitate a direct comparison of their performance.

D.1 Synthetic Examples

Additional Results. Figure 4 illustrates the performance of various SVGD methods on the Swiss Roll and GMM-8 tasks. Table 8 shows the quantitative analysis results of various SVGD variants.

Ablation on Spatial Partitioning Granularity. The core hyperparameter of RP-SVGD is the maximum leaf node capacity C, which controls the spatial partitioning granularity of the RP-Tree and, consequently, the number of interacting particles within each local neighborhood. To investigate

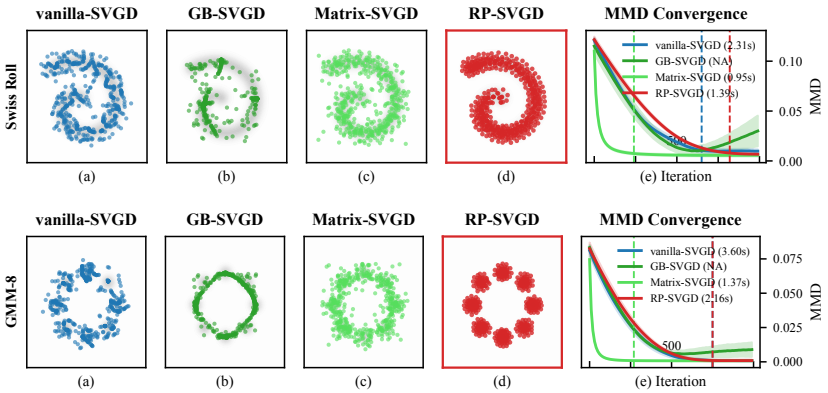

Figure 4: (a)-(d) show the particles obtained by various methods on the different target distributions at the final iteration. (e) MMD convergence trajectories over training iterations. The curves and shaded regions denote the mean and standard deviation across 10 independent trials, respectively. The vertical dashed lines mark the iteration at which each method converges, while the annotations above the plots indicate the corresponding wall-clock time to convergence.

its impact, we conduct an ablation study by varying $C \in \{10, 30, 50, 80\}$ across all four 2D synthetic distributions. The particle visualizations and the Maximum Mean Discrepancy (MMD) convergence trajectories are presented in Figure 5. Table 7 reports the average per-iteration execution time across these settings.

Table 7: Computational efficiency of RP-SVGD with varying maximum leaf capacity C (per-iteration execution time in ms).

Maximum Leaf Capacity (C)	Time per Iteration (ms)
10	3.404 ± 0.027
30	1.891 ± 0.025
50	1.472 ± 0.016
80	1.386 ± 0.160

Result. Our ablation study reveals that RP-SVGD exhibits robustness to the choice of tree-partitioning parameters. Across all synthetic benchmarks, the final particle distributions consistently recover the ground-truth density, while convergence speeds remain stable across different leaf capacities. The results confirm that a moderate leaf capacity C effectively strikes the balance between approximation fidelity and computational efficiency, demonstrating the practical scalability of our framework.

D.2 Bayesian Logistic Regression

Additional Results Table 9 presents the results of RP-SVGD compared to vanilla SVGD and GB-SVGD on the Ionosphere and German datasets.

Ablation Experiment We evaluate the performance of RP-SVGD on the Bayesian Logistic Regression task across four UCI datasets, focusing on the sensitivity of the hyperparameter C (maximum leaf node capacity). For all datasets, we approximate the posterior with $N = 500$ particles and mini-batches of size $B = 500$. We conduct an ablation study with $C \in \{10, 30, 50, 80\}$ to investigate the trade-off between local approximation fidelity and computational speed.

Result Analysis. Table 10 summarizes the predictive performance (Accuracy, NLL, and RMSE) and computational efficiency (Time) for different leaf capacities. Our findings reveal that RP-SVGD is highly robust to the choice of C. Across all four datasets, the variations in Accuracy, NLL, and RMSE remain within a narrow margin, demonstrating that the spatial partitioning induced by RP-Trees is robust regardless of minor variations in cluster granularity.

Regarding efficiency, we observe a clear trend: smaller values of C (e.g., $C = 10$) lead to higher computational costs due to the overhead of tree construction and the resulting larger number of

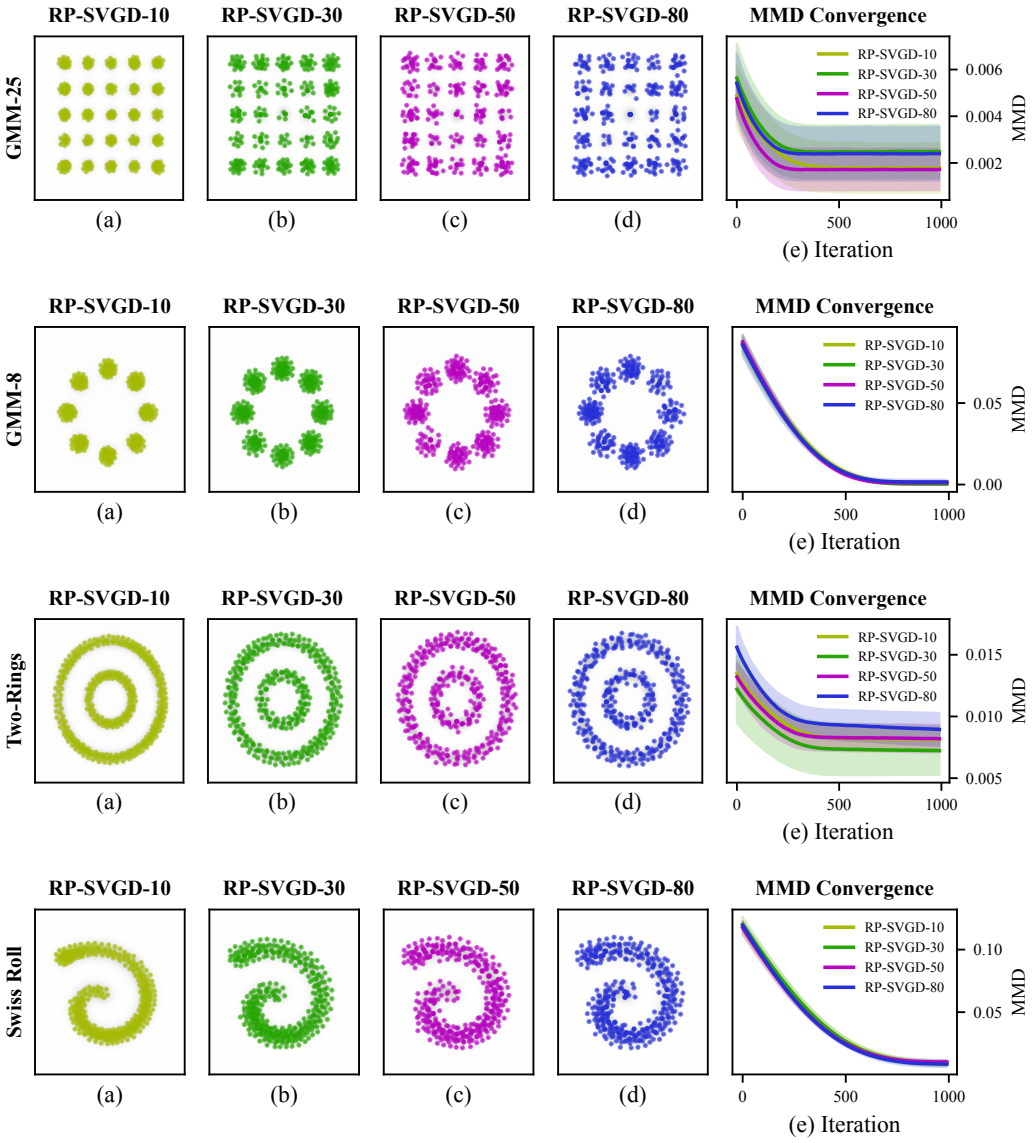

Figure 5: Ablation study on the maximum leaf node capacity $C \in \{10, 30, 50, 80\}$ of the RP-Tree. Columns (a)-(d) visualize the final particle distributions for different capacities across four synthetic datasets. Column (e) displays the corresponding MMD convergence trajectories. A moderate capacity ($C = 50$) achieves the best balance between approximation fidelity and computational efficiency.

Table 8: Performance Comparison of SVGD Variants across Synthetic Distributions. **Bold** and underline indicate the best and second-best results, respectively. Results are averaged over 10 independent trials.

Dataset	Method	Time (s) ↓	MMD ↓	W_2 ↓	Energy ↓	KL ↓	Prec. ↑	Rec. ↑
Two-Rings	**RP-SVGD**	1.3895	**0.0084**	**1.5078**	**0.2016**	16.1559	0.988	**0.998**
	vanilla-SVGD	3.2037	0.0088	1.6386	0.2240	15.4057	**0.996**	0.988
	GB-SVGD	**0.4342**	0.0150	2.6345	0.5236	18.7137	0.924	0.976
	Matrix-SVGD	4.7634	0.0214	4.3133	1.1124	21.5822	0.736	0.992
GMM-8	**RP-SVGD**	2.8025	**0.0003**	**0.3382**	**0.0153**	**13.4637**	**0.976**	0.998
	vanilla-SVGD	5.0036	0.0026	0.5714	0.0536	15.4711	0.954	0.988
	GB-SVGD	**1.5701**	0.0041	1.7167	0.2963	14.0507	0.576	**1.000**
	Matrix-SVGD	6.0197	0.0100	2.1360	0.6033	21.8972	0.260	**1.000**
GMM-25	**RP-SVGD**	1.4949	**0.0013**	**0.7239**	**0.0509**	**14.9339**	0.992	**0.994**
	vanilla-SVGD	3.0991	0.0043	1.2537	0.1207	16.9398	**0.996**	0.990
	GB-SVGD	**0.4253**	0.0058	1.8243	0.2249	19.2140	0.780	0.980
	Matrix-SVGD	4.5853	0.0145	2.8352	0.6142	18.6878	0.828	0.992
Swiss-Roll	**RP-SVGD**	1.5960	**0.0123**	**1.4337**	**0.3411**	17.8768	**0.988**	0.992
	vanilla-SVGD	3.5647	0.0248	1.5383	0.3598	**17.6155**	0.932	0.986
	GB-SVGD	**0.4799**	0.0350	2.2642	0.8147	19.5932	0.820	**1.000**
	Matrix-SVGD	4.8904	0.0144	1.9977	0.9850	21.3708	0.000	**1.000**

leaf nodes, whereas larger values of C (e.g., $C = 80$) further reduce the runtime by decreasing the granularity of spatial partitions. However, as C increases beyond 50, the marginal gains in speed diminish, while the approximation error (reflected in NLL and RMSE) remains consistently stable. This empirical evidence confirms that a moderate leaf capacity $C \approx 50$ strikes the optimal balance between computational efficiency and posterior approximation fidelity, validating the practical utility of our proposed RP-SVGD framework.

Table 9: Performance comparison on UCI benchmarks. We report the Mean ± Standard Deviation over 10 independent trials. For each metric, the best performing method is **bolded** and the second best is underlined. ↑ indicates higher values are better, and ↓ indicates lower values are better.

Dataset	Method	Test Accuracy ↑	Test NLL ↓	Test RMSE ↓	Time (s) ↓
Ionosphere	Vanilla-SVGD	0.9150 ± 0.0345	**0.2798 ± 0.0997**	0.2365 ± 0.0192	6.20 ± 0.12
	RP-SVGD	**0.9152 ± 0.0311**	0.2966 ± 0.1153	0.2375 ± 0.0177	3.10 ± 0.02
	Matrix-SVGD	0.9114 ± 0.0304	0.2815 ± 0.0295	**0.2350 ± 0.0056**	14.86 ± 0.17
	GB-SVGD	0.9108 ± 0.0289	0.3061 ± 0.1100	0.2374 ± 0.0180	**0.63 ± 0.05**
German	Vanilla-SVGD	0.6483 ± 0.0278	0.6627 ± 0.0572	0.3039 ± 0.0051	10.78 ± 2.24
	RP-SVGD	0.6654 ± 0.0178	**0.6569 ± 0.0441**	0.3032 ± 0.0052	3.68 ± 0.18
	Matrix-SVGD	**0.6733 ± 0.0165**	0.6583 ± 0.0492	**0.2999 ± 0.0043**	26.94 ± 3.72
	GB-SVGD	0.6550 ± 0.0147	0.6594 ± 0.0399	0.3012 ± 0.0037	**1.18 ± 0.18**

D.3 Enabling GPU Execution for RP-SVGD

Meanwhile, we have optimized our method to fully utilize GPU which is accomplished by considering the vectorized execution of Algo. 1 (the balanced tree built as a permutation refined by batched per-node random-projection sorts; leaf kernels as batched GEMMs; no CPU round-trips) as we already discussed in Appendix C.7. We report these experimental results in the table below. As shown in Tab. 11, when N is large, we observe a significant speedup: $3.6\times$ at $N = 4096$ and $13.0\times$ at $N = 8192$, with better final MMD.

Table 10: Ablation study on the effect of leaf node capacity C across four real-world datasets. Results are reported as Mean ± Std over 10 independent trials. This experiment involved a separate rerun, so the results differ from the previous figures.

Dataset	Method	Accuracy ↑	NLL ↓	RMSE ↓	Time (s) ↓
BreastCancer	RP-SVGD ($C = 10$)	0.9752 ± 0.0089	0.0795 ± 0.0099	0.1709 ± 0.0075	7.09 ± 0.27
	RP-SVGD ($C = 30$)	0.9762 ± 0.0105	0.0799 ± 0.0102	0.1710 ± 0.0066	3.52 ± 0.55
	RP-SVGD ($C = 50$)	$\mathbf{0.9769 \pm 0.0086}$	0.0814 ± 0.0119	$\mathbf{0.1707 \pm 0.0069}$	3.23 ± 0.42
	RP-SVGD ($C = 80$)	0.9754 ± 0.0070	$\mathbf{0.0782 \pm 0.0068}$	0.1717 ± 0.0051	$\mathbf{3.06 \pm 0.06}$
Ionosphere	RP-SVGD ($C = 10$)	0.9114 ± 0.0252	0.3066 ± 0.0923	0.2373 ± 0.0178	6.74 ± 0.05
	RP-SVGD ($C = 30$)	0.9142 ± 0.0301	0.2951 ± 0.0912	$\mathbf{0.2351 \pm 0.0189}$	3.24 ± 0.12
	RP-SVGD ($C = 50$)	$\mathbf{0.9170 \pm 0.0316}$	$\mathbf{0.2953 \pm 0.0904}$	0.2352 ± 0.0193	3.10 ± 0.07
	RP-SVGD ($C = 80$)	0.9114 ± 0.0252	0.2992 ± 0.0903	0.2387 ± 0.0180	$\mathbf{3.06 \pm 0.09}$
German	RP-SVGD ($C = 10$)	0.6680 ± 0.0108	0.6605 ± 0.0385	0.3032 ± 0.0041	7.80 ± 0.24
	RP-SVGD ($C = 30$)	0.6680 ± 0.0178	0.6577 ± 0.0425	$\mathbf{0.3028 \pm 0.0044}$	3.98 ± 0.12
	RP-SVGD ($C = 50$)	$\mathbf{0.6750 \pm 0.0089}$	0.6567 ± 0.0446	0.3031 ± 0.0044	3.63 ± 0.26
	RP-SVGD ($C = 80$)	0.6720 ± 0.0163	$\mathbf{0.6536 \pm 0.0345}$	0.3035 ± 0.0038	$\mathbf{3.48 \pm 0.24}$
Covtype	RP-SVGD ($C = 10$)	0.7543 ± 0.0063	0.5222 ± 0.0051	0.2835 ± 0.0009	7.41 ± 0.04
	RP-SVGD ($C = 30$)	0.7552 ± 0.0065	0.5227 ± 0.0045	0.2835 ± 0.0010	4.14 ± 0.08
	RP-SVGD ($C = 50$)	0.7558 ± 0.0070	0.5222 ± 0.0044	0.2834 ± 0.0007	3.27 ± 0.05
	RP-SVGD ($C = 80$)	$\mathbf{0.7569 \pm 0.0048}$	$\mathbf{0.5212 \pm 0.0035}$	$\mathbf{0.2833 \pm 0.0008}$	$\mathbf{3.26 \pm 0.06}$

Table 11: 2-D benchmark on GPUs: NVIDIA RTX A6000, Two-Rings task, $C = 64$.

N	Method	Final MMD ↓	Conv. iter ↓	Time-to-conv (s) ↓	Total time (s) ↓	ms/iter ↓
2048	vanilla-SVGD	0.0129 ± 0.0015	606 ± 96	1.27 ± 0.20	2.07 ± 0.00	2.07 ± 0.00
2048	RP-SVGD (C=64)	0.0072 ± 0.0007	394 ± 10	0.76 ± 0.02	1.91 ± 0.02	1.91 ± 0.02
4096	vanilla-SVGD	0.0129 ± 0.0030	412 ± 67	3.26 ± 0.53	7.83 ± 0.01	7.83 ± 0.01
4096	RP-SVGD (C=64)	0.0066 ± 0.0004	430 ± 11	0.93 ± 0.04	2.15 ± 0.06	2.15 ± 0.06
8192	vanilla-SVGD	0.0147 ± 0.0028	538 ± 124	16.29 ± 3.75	29.97 ± 0.02	29.97 ± 0.02
8192	RP-SVGD (C=64)	0.0063 ± 0.0003	472 ± 7	1.10 ± 0.03	2.31 ± 0.04	2.31 ± 0.04

D.4 Variance of the single-tree approximation.

Structurally, for fixed τ, the single-tree direction is an unbiased estimate of the effective direction, and an independent tree is resampled at every iteration, so the tree noise is zero-mean and independent across steps. The dynamics itself then averages it over time—similarly as minibatch noise is absorbed in SGD. In Tab. 12: we average the update direction over $M \in \{1, 2, 4, 8\}$ independent trees per iteration, yielding a direct $\times 1/M$ reduction in variance. Here, $M = 1$ corresponds to the algorithm in Algorithm 1. We evaluate all settings under the paper's 2-D experimental protocol. Final quality is flat with respect to M, whereas the computational cost grows with M.

Table 12: Multiple Trees: Final MMD across methods and tree counts (RBF kernel, $C = 50$, number of replica $= 10$, on CPU AMD Ryzen Threadripper PRO 5995WX).

Target	$M = 1$	$M = 2$	$M = 4$	$M = 8$	Conv. iter ($M = 1 \to 8$)	ms/iter ($M = 1 \to 8$)
Two-Rings	0.0088 ± 0.0013	0.0087 ± 0.0014	0.0086 ± 0.0014	0.0085 ± 0.0013	$419 \pm 57 \to 434 \pm 101$	$1.82 \pm 0.01 \to 12.38 \pm 0.24$
GMM-8	0.0011 ± 0.0009	0.0011 ± 0.0008	0.0011 ± 0.0009	0.0011 ± 0.0009	$744 \pm 33 \to 693 \pm 32$	$5.05 \pm 0.06 \to 15.46 \pm 0.07$
GMM-25	0.0028 ± 0.0012	0.0028 ± 0.0011	0.0028 ± 0.0012	0.0028 ± 0.0012	$270 \pm 16 \to 250 \pm 13$	$2.07 \pm 0.01 \to 12.32 \pm 0.05$
Swiss Roll	0.0096 ± 0.0024	0.0097 ± 0.0022	0.0099 ± 0.0023	0.0103 ± 0.0025	$853 \pm 27 \to 786 \pm 32$	$2.24 \pm 0.02 \to 12.64 \pm 0.05$

D.5 Ablation Study on the Effect of Taking $\tau \to 0$

In Tab. 13, we include ablation implementing the smoothed version and sweeps τ. For sampling of two-rings task, it is found that the Gram matrix remains dense for every $\tau > 0$, resulting in a per-iteration cost of 6.7–7.3 ms, compared with 1.8 ms for the hard algorithm (Algo. 1). Meanwhile, the hard algorithm achieves comparable solution quality in our experiments while providing the lowest computational cost among the tested settings.

Table 13: Ablation study of the smoothing temperature (τ) for soft RP-SVGD versus the hard-partition limit (Algorithm 1) and vanilla SVGD, on Two-Rings sampling. RBF base kernel, $C = 50$, number of replica $= 10$, on CPU AMD Ryzen Threadripper PRO 5995WX.

Method	Final MMD ↓	Conv. iter ↓	Time-to-conv (s) ↓	ms/iter ↓
vanilla-SVGD	0.0071 ± 0.0023	504 ± 40	1.84 ± 0.15	3.65 ± 0.12
RP-SVGD (hard)	0.0071 ± 0.0012	396 ± 34	0.73 ± 0.07	1.84 ± 0.09
RP-SVGD (τ=0.1)	0.0071 ± 0.0012	667 ± 233	4.55 ± 1.93	6.66 ± 0.64
RP-SVGD (τ=0.01)	0.0075 ± 0.0011	592 ± 149	4.29 ± 1.02	7.26 ± 0.18

D.6 Ablation Study on Slowly-decaying kernels

We ran the ablation for IMQ $k = (1 + r^2/h)^{-1/2}$, and Cauchy $k = (1 + r^2/h)^{-1}$ where r denotes the distance between two points, on all four 2-D targets, using the same protocol as in the main paper. Under slowly decaying kernels, RP-SVGD does not merely still work; it often outperforms vanilla SVGD. Under IMQ, vanilla SVGD fails to separate the GMM-8 modes (particles blur into a ring) and fails to fill the Swiss Roll; Cauchy behaves similarly (e.g., GMM-8: 0.0047 vs. 0.0011).

Table 14: Final MMD comparison: vanilla-SVGD vs. RP-SVGD. $N = 500$, number of replica $= 10$, on CPU AMD Ryzen Threadripper PRO 5995WX.

Kernel	Target	vanilla-SVGD	RP-SVGD (C=50)
IMQ	GMM-8	0.0090 ± 0.0059	0.0011 ± 0.0009
IMQ	GMM-25	0.0032 ± 0.0020	0.0020 ± 0.0010
IMQ	Swiss Roll	0.0488 ± 0.0088	0.0170 ± 0.0042
IMQ	Two-Rings	0.0080 ± 0.0022	0.0091 ± 0.0021
Cauchy	GMM-8	0.0047 ± 0.0032	0.0011 ± 0.0009
Cauchy	GMM-25	0.0024 ± 0.0011	0.0022 ± 0.0010
Cauchy	Swiss Roll	0.0292 ± 0.0031	0.0160 ± 0.0040
Cauchy	Two-Rings	0.0066 ± 0.0020	0.0093 ± 0.0019

D.7 Ablation Study on Spending the savings on extra updates

We measured the cases of reinvesting the savings in extra update rounds at a fixed wall-clock budget: within each (task, run), vanilla SVGD's standard run (1000 steps on 2-D / 800 steps on BLR) defines the budget, and each RP-SVGD variant ($C \in \{10, 30, 50, 80\}$) runs from the *same* initialization until the budget is exhausted, with the tree rebuilt every iteration *inside* the budget. As shown by the table, on Two-Rings RP-SVGD completes $1.70\times$–$2.03\times$ as many iterations within the same budget and matches vanilla's final MMD (0.0073–0.0075 vs. 0.0075) while its across-repeat spread is about half as large (± 0.0011 vs. ± 0.0024); on Covtype BLR it completes $1.41\times$–$1.62\times$ as many iterations with all accuracy/NLL differences inside one standard deviation.

D.8 Ablation Study on the Tradeoff between Locality Strength and number of Rounds

The per-iteration cost is a U-shaped function of C, because tree construction, $D = \lceil \log_2(N/C) \rceil$ levels over N/C leaves, may dominate at small C, while the $\mathcal{O}(NC)$ leaf-kernel computations dominate at large C.

Tab. 15 measures this, where the completed-iteration count is the inverse of the per-iteration cost: for Two-Rings, it peaks at $C = 50$ (2034 ± 5 rounds, $D = 4$), decreases to 1700 ± 7 at $C = 10$ ($D = 6$), where the most local setting requires 16% fewer rounds, and rises again at $C = 80$ ($D = 3$); Covtype BLR shows the same shape with its optimum at $C = 50$ ($D = 4$). At equal wall-clock, the performance is flat in C across $D \in \{3, \ldots, 6\}$. RP-SVGD completes $1.70\times$–$2.03\times$ (Two-Rings) and $1.41\times$–$1.62\times$ (Covtype) as many rounds inside vanilla's budget and matches its final performance. It does not beat it, and we can say why: under these two tasks, both methods are plateau-limited rather than iteration-limited so the surplus rounds after convergence cannot lower the plateau.

Table 15: Compute-matched comparison at an equal wall-clock budget: vanilla SVGD's standard run defines the budget; each RP-SVGD variant runs from the same initialization until it is exhausted. RBF kernel; all evaluation outside the timers. $N = 500$, number of replica $= 10$, measured on CPU AMD Ryzen Threadripper PRO 5995WX.

Two-Rings (sampling)

Method	D	Iterations	Wall-clock (s)	Final MMD ↓
vanilla-SVGD	–	1000	3.63 ± 0.10	0.0075 ± 0.0024
RP-SVGD (C=10)	6	1700 ± 7	3.63 ± 0.00	0.0073 ± 0.0011
RP-SVGD (C=30)	5	1972 ± 7	3.63 ± 0.00	0.0074 ± 0.0011
RP-SVGD (C=50)	4	2034 ± 5	3.63 ± 0.00	0.0075 ± 0.0011
RP-SVGD (C=80)	3	1985 ± 5	3.63 ± 0.00	0.0075 ± 0.0011

Covtype BLR

Method	D	Iterations	Wall-clock (s)	Accuracy ↑	NLL ↓
Vanilla-SVGD	–	800	3.44 ± 0.05	0.7531 ± 0.0047	0.5189 ± 0.0049
RP-SVGD (C=10)	6	1125 ± 2	3.45 ± 0.00	0.7535 ± 0.0046	0.5194 ± 0.0038
RP-SVGD (C=30)	5	1285 ± 29	3.44 ± 0.00	0.7523 ± 0.0052	0.5188 ± 0.0061
RP-SVGD (C=50)	4	1294 ± 2	3.44 ± 0.00	0.7534 ± 0.0056	0.5194 ± 0.0055
RP-SVGD (C=80)	3	1265 ± 4	3.44 ± 0.00	0.7525 ± 0.0068	0.5193 ± 0.0050

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly articulate the main contributions of this paper: a novel particle-based reasoning framework called RP-SVGD is proposed, which dynamically constructs random projection trees to restrict kernel interactions to spatially proximate neighborhoods. Both theoretical exposition and experimental results strongly support the following assertions: RP-SVGD alleviates the $O(N^2)$ computational bottleneck of traditional SVGD, explicitly preserves local repulsion to prevent mode collapse, and improves practical running speed. Furthermore, this paper acknowledges the limitations of RP-SVGD in highly parallel GPU computing and more rigorous convergence theory, and proposes the possibility of improving these limitations through future algorithmic optimization, which aligns with the research direction of this paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper includes a discussion of its main limitation in conclusion.

   Guidelines:

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

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: We provide assumptions and complete proof in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: All code and data are provided in supplementary materials.

   Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: See supplementary materials.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper specifies the necessary experimental details to understand and reproduce the results.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: The paper reports statistical variability for key experimental results using error bars that reflect standard deviation over multiple runs with different random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper provides details of the compute resources used in Appendix C.7.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research fully conforms to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

Justification: The paper includes a discussion of broader impacts. On the positive side, the proposed RP-SVGD framework has the potential to improve the computational efficiency and scalability of particle-based variational inference, which could benefit applications such as large-scale generative modeling, robust uncertainty quantification, and complex reinforcement learning systems. On the negative side, like other advanced approximate inference techniques, there is a risk that the method could be misapplied in high-stakes decision-making systems without sufficient verification. Specifically, because the practical algorithm relies on hard-partitioning approximations in the limit of $\tau \to 0$, over-reliance on its outputs without careful tuning could lead to miscalibrated uncertainty estimates, potentially resulting in unintended behavior. While this work is primarily algorithmic and demonstrated on standard synthetic benchmarks and simulations, awareness of deployment risks in real-world probabilistic systems is important. We believe transparency and further research into a full theoretical analysis of convergence can help mitigate these concerns.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: The models and data used in this work do not pose a high risk of misuse.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All external assets used in this work—such as code and datasets—are properly cited in the paper. The licenses and terms of use are respected and, where applicable, included in the references or appendix.

Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: The paper introduces new code assets implementing the proposed RP-SVGD framework. Documents are provided in supplementary materials.

Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

Answer: [N/A]

Justification: This work does not involve any crowdsourcing or research with human subjects. All experiments are conducted in simulated environments without human data or interaction.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A]

    Justification: This work does not involve human subjects, user studies, or the collection of human data. Therefore, IRB approval is not applicable.

    Guidelines:

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

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used as part of the core methodology, experiments, or technical contributions of this work. Any use of LLMs was solely for writing refinement and editing purposes.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.