Stein Variational Evolution Strategies

Cornelius V. Braun[1] **Robert T. Lange**[2] **Marc Toussaint**[1]

[1]TU Berlin
[2]Sakana AI

Abstract

Efficient global optimization and sampling are fundamental challenges, particularly in fields such as robotics and reinforcement learning, where gradients may be unavailable or unreliable. In this context, jointly optimizing multiple solutions is a promising approach to avoid local optima. While Stein Variational Gradient Descent (SVGD) provides a powerful framework for sampling diverse solutions, its reliance on first-order information limits its applicability to differentiable objectives. Existing gradient-free SVGD variants often suffer from slow convergence and poor scalability. To improve gradient-free sampling and optimization, we propose Stein Variational CMA-ES, a novel gradient-free SVGD-like method that combines the efficiency of evolution strategies with SVGD-based repulsion forces. We perform an extensive empirical evaluation across several domains, which shows that the integration of the ES update in SVGD significantly improves the performance on multiple challenging benchmark problems. Our findings establish SV-CMA-ES as a scalable method for zero-order sampling and blackbox optimization, bridging the gap between SVGD and evolution strategies.

1 INTRODUCTION

Many optimization problems – such as neural network parameter search – involve highly non-convex objective functions, which makes the optimization process very sensitive to its initialization [Sullivan et al., 2022, Li et al., 2018]. Thus, these hard optimization problems are commonly approached by generating multiple solution candidates from which the best is selected [Toussaint et al., 2024, Parker-Holder et al., 2020]. This allows to frame the task of minimizing a function $f : \mathbb{R}^d \to \mathbb{R}$ as an approximate inference problem which can be formulated as follows:

$$\min_q D(q \parallel p) \qquad p(\mathbf{x}) = \frac{e^{-f(\mathbf{x})}}{Z},$$

where the normalization constant $Z = \int_{\mathbb{R}^d} e^{-f(\mathbf{x})} \mathrm{d}\mathbf{x}$ is typically intractable, p and q are probability distributions supported on $\mathbb{R}^d$, and D is a suitable divergence, such as the Kullback-Leibler (KL) divergence.

Stein Variational Gradient Descent (SVGD) is a powerful algorithm to solve this optimization problem through iteratively updating a particle set [Liu and Wang, 2016]. As the approach is non-parametric and does not require the lengthy burn-in periods of Markov chain Monte Carlo (MCMC) methods [Andrieu et al., 2003], it is a computationally efficient method to approximate complex distributions. Due to these properties, SVGD is an increasingly popular first-order method for sampling and non-convex optimization [Zhang et al., 2019, Maken et al., 2021, Pavlasek et al., 2023].

Unfortunately, the reliance of SVGD on the score function limits its applicability to differentiable objectives. In many real-world problems – such as robotics and chemistry – however, the energy function f may not yield reliable gradients or be non-differentiable altogether [Lambert et al., 2021, Englert and Toussaint, 2018, Maus et al., 2023]. To facilitate *gradient-free* Stein variational inference, prior work introduced a zero-order version of SVGD that uses analytical gradients from a surrogate distribution [Han and Liu, 2018, GF-SVGD]. While the algorithm provably minimizes the KL divergence, fitting the surrogate to the objective function is challenging in practice, especially in higher dimensions (cf. Fig. 2). Alternatively, other works used simple Monte Carlo (MC) gradients in the SVGD update [Liu et al., 2017, Lambert et al., 2021, Lee et al., 2023]. Again, this approach comes with limitations as the MC step estimate has high variance, which often leads to noisy updates and thus poor computational efficiency.

To address the aforementioned shortcomings of existing gradient-free SVGD methods, we propose a novel approach,

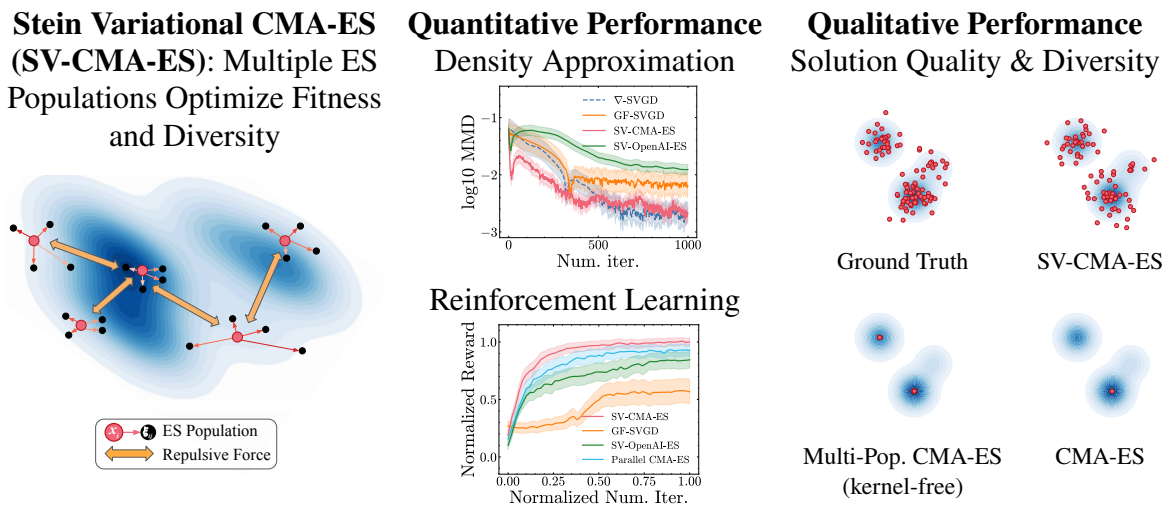


Figure 1: **Left**: Illustration of Stein Variational CMA-ES. Multiple ES search distributions are updated in parallel, similar to the SVGD step. **Middle**: Quantitative comparison of different methods for sampling and RL control tasks. SV-CMA-ES obtains higher quality solutions than existing gradient-free SVGD-based approaches. **Right**: Qualitative comparison of different CMA-ES-based methods unveils that SV-CMA-ES generates more diverse samples than other methods. The full experimental details can be found in Section 5.

Stein Variational CMA-ES (SV-CMA-ES). Our method bridges the fields of Evolution Strategies (ES) and distribution approximation by updating multiple ES search distributions in parallel. The idea of SV-CMA-ES is to perform the distribution updates in a coordinated manner using a kernel-based repulsion term, which ensures an inter-population diversity similar to that in SVGD. We motivate our work based on prior results that established ES as a competitive alternative to gradient-based optimization algorithms, achieving higher performance and robustness on difficult objectives due to their inherent exploration capabilities [Salimans et al., 2017, Wierstra et al., 2014]. In particular, the Covariance Matrix Adaptation Evolution Strategy [Hansen and Ostermeier, 2001, CMA-ES] is one of the most popular ES across many domains [Hansen et al., 2010, Jankowski et al., 2023], due to its adaptive and efficient search process, which leverages a dynamic step-size adaptation mechanism to increase convergence speeds [Akimoto et al., 2012].

We evaluate our proposed approach on a wide range of challenging problems from multiple domains, such as robot trajectory optimization and reinforcement learning. Our experimental results demonstrate that SV-CMA-ES improves considerably over existing gradient-free SVGD approaches. Fig. 1 summarizes our findings. Not only can our method be used to sample from challenging densities efficiently, but also as a blackbox optimizer on non-convex objectives. We outline our contributions as follows:

1. We introduce a novel zero-order method for diverse sampling and global optimization that combines ideas of SVGD with gradient-free ES, thus bypassing the need for a surrogate distribution required by previous gradient-free SVGD approaches (Section 3).
2. We validate our method, SV-CMA-ES, on a range of problems and demonstrate that it improves over prior gradient-free SVGD approaches in sampling and optimization tasks (Fig. 1 middle; Sec. 5.1-5.3).
3. We show that our presented method improves over prior CMA-ES-based methods because it combines the fast convergence rate of CMA-ES with the entropy-preserving optimization dynamics of SVGD (Fig. 1 right; Sec. 5.3).

2 PRELIMINARIES

2.1 STEIN VARIATIONAL GRADIENT DESCENT

Stein Variational Gradient Descent [Liu and Wang, 2016, SVGD] is a non-parametric inference algorithm that approximates a target distribution with a set of $\varrho \in \mathbb{N}^+$ particles $X = \{\mathbf{x}_i\}_{i=1}^{\varrho}$ as $q(\mathbf{x}) = \sum_{\mathbf{x}_i \in X} \delta(\mathbf{x} - \mathbf{x}_i)/\varrho$, where we use $\delta(\cdot)$ to denote the Dirac delta function. Given an initial set of particles, the goal is to determine an optimal particle transformation $\phi^* : \mathbb{R}^d \to \mathbb{R}^d$ that maximally decreases the KL divergence $D_{\mathrm{KL}}(q \parallel p)$:

$$\begin{aligned} \mathbf{x}_i &\leftarrow \mathbf{x}_i + \epsilon \phi^*(\mathbf{x}_i), \quad \forall \mathbf{x}_i \in X \\ \text{s.t. } \phi^* &= \arg\min_{\phi \in \mathcal{F}} \left\{ \frac{d}{d\epsilon} D_{\mathrm{KL}}(q_{[\epsilon\phi]} \parallel p) \Big|_{\epsilon=0} \right\}, \end{aligned} \tag{1}$$

where $\epsilon \in \mathbb{R}$ is a sufficiently small step-size, $q_{[\epsilon\phi]}$ denotes the distribution of the updated particles, and $\mathcal{F}$ is a set of candidate transformations.

The main result by Liu and Wang [2016] is the derivation of a closed form solution to this optimization problem. By choosing $\mathcal{F}$ as the unit sphere $\mathcal{B}_k$ in a vector-valued reproducing kernel Hilbert space $\mathcal{H}_k^d$, i.e. $\mathcal{F}_k = \mathcal{B}_k\{\phi \in \mathcal{H}_k^d : \|\phi\|_{\mathcal{H}_k^d} \leq 1\}$, with its kernel function $k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, the authors show that the solution to Eq. (1) is:

$$\phi_k^*(\cdot) \propto \mathop{\mathbb{E}}_{\mathbf{x}\sim q}\Big[\underbrace{\nabla_{\mathbf{x}} \log p(\mathbf{x}) k(\mathbf{x},\cdot)}_{\text{driving force}} + \underbrace{\nabla_{\mathbf{x}} k(\mathbf{x},\cdot)}_{\text{repulsive force}}\Big]. \tag{2}$$

This result can be used to update the particle set iteratively using Eq. (1) and (2), where the expectation is estimated via MC approximation over the entire particle set X. Intuitively, the particle update balances likelihood maximization and particle repulsion: the first term drives particles toward regions of higher probability, while the second term counteracts this by repulsing particles based on the kernel gradient [D'Angelo and Fortuin, 2021, Ba et al., 2021].

Because vanilla SVGD is prone to the initialization of particles and mode collapse [Zhuo et al., 2018, Ba et al., 2021, Zhang et al., 2020], prior work proposed *Annealed SVGD* [Liu et al., 2017, D'Angelo and Fortuin, 2021]. This extension of SVGD, reweighs the terms in the update based on the optimization progress [D'Angelo and Fortuin, 2021]. Given the timestep-dependent temperature parameter $\gamma(t) \in \mathbb{R}$, the annealed update is:

$$\phi_k^*(\cdot) \propto \mathop{\mathbb{E}}_{\mathbf{x}\sim q}\left[\nabla_{\mathbf{x}} \log p(\mathbf{x}) k(\mathbf{x},\cdot) + \gamma(t)\nabla_{\mathbf{x}} k(\mathbf{x},\cdot)\right]. \tag{3}$$

2.2 COVARIANCE MATRIX ADAPTATION EVOLUTION STRATEGY

The Covariance Matrix Adaptation Evolution Strategy [Hansen and Ostermeier, 2001, CMA-ES] is one of the most popular ES algorithms. We therefore choose it as the starting point for our ES-based SVGD method. The core idea of the CMA-ES algorithm is to iteratively optimize the parameters of a Gaussian search distribution $\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{C})$ from which the candidate solutions are sampled. While the algorithmic intuition of CMA-ES is similar to MC gradient approaches [Salimans et al., 2017], CMA-ES updates the search distribution following natural gradient steps, which has been shown to produce more efficient steps than standard gradient descent on multiple problems [Martens, 2020, Akimoto et al., 2012, Glasmachers et al., 2010].

We note that our notation in the following deviates from the default notation in the ES literature, as some of its variable names are typically associated with a different meaning compared to the variational inference (VI) literature. For instance, μ commonly is the symbol for the mean of a Gaussian in VI literature, while it refers to the number of elites in CMA-ES. To improve clarity, we thus use the variable n to denote the size of a sampled CMA-ES population and m for the number of selected elites. In our notation, CMA-ES is therefore an (m, n) strategy. The CMA-ES algorithm relies on multiple hyperparameters which we fix to the default values from Hansen [2016]. For completeness, we include the definitions of these variables – $w_i, \alpha_1, \alpha_m, \alpha_\sigma, h_\sigma, d(h_\sigma)$ and $\bar{w}_i$ – in the Appendix. Further, we slightly overload notation by using $\mathbf{p}$ to denote the evolution path updates following Hansen [2016] (unlike pdf's which we denote by p).

Given a population of n candidate samples $\boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{x}, \sigma^2\mathbf{C})$, each iteration of CMA-ES updates the search parameters as follows. First, the samples $\boldsymbol{\xi}_i$ are evaluated and ranked by their fitness $f(\boldsymbol{\xi}_i)$ in ascending order. To simplify notation, we assume ranked solutions in the following, i.e., we assume that index $i < j \to f(\boldsymbol{\xi}_i) \leq f(\boldsymbol{\xi}_j)$. This allows to assign each sample to a mutation weight w_i, where weights for better solutions are higher. For details on the exact computation of the weights, we refer to Hansen [2016] and to Appendix A of our work. The mean of the search distribution is then updated by mutating the $m \leq n$ best samples from the current generation of candidates:

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}_{\text{CMA}} \tag{4}$$

where

$$\Delta\mathbf{x}_{\text{CMA}} = \sigma \textstyle\sum_{i=1}^m w_i \mathbf{y}_i, \text{ and } \mathbf{y}_i = (\boldsymbol{\xi}_i - \mathbf{x})/\sigma. \tag{5}$$

Next, the parameters of the search distribution are updated. First, the step-size σ is updated based on the history of prior steps. Given

$$m_{\text{eff}} = (\Sigma_{i=1}^m w_i^2)^{-1}, \text{ and} \tag{6}$$

$$\mathbf{p}_\sigma \leftarrow (1-\alpha_\sigma)\mathbf{p}_\sigma + \sqrt{\alpha_\sigma(2-\alpha_\sigma)\, m_{\text{eff}}}\, \mathbf{C}^{-\frac{1}{2}} \Delta\mathbf{x}_{\text{CMA}}/\sigma, \tag{7}$$

we define

$$\sigma \leftarrow \sigma \times \exp\Big(\frac{\alpha_\sigma}{d_\sigma}\Big(\frac{\|\mathbf{p}_\sigma\|}{\mathbb{E}\|\mathcal{N}(0,\mathbf{I})\|} - 1\Big)\Big), \tag{8}$$

where $\mathbf{p}_\sigma$ is a moving average over the optimization steps, which unprojects the steps using $C^{-1/2}/\sigma$, so the resulting vector follows a standard normal. Thus, Eq. (8) automatically adapts the step-size based on the expected length of steps, similar to momentum-based optimizers [Kingma and Ba, 2015, Nesterov, 1983], with the hyperparameters α_σ and d_σ governing the rate of the step-size changes. Finally, the covariance $\mathbf{C}$ is updated based on the covariance of the previous steps and current population fitness values:

$$\begin{aligned}\mathbf{C} \leftarrow\ & (1 + \alpha_1 d(h_\sigma) - \alpha_1 - \alpha_m \textstyle\sum_{j=1}^n w_j)\mathbf{C} \\ & + \alpha_1 \mathbf{p}_c \mathbf{p}_c{}^T + \alpha_m \textstyle\sum_{i=1}^n \bar{w}_i \mathbf{y}_i \mathbf{y}_i^T,\end{aligned} \tag{9}$$

with

$$\mathbf{p}_c \leftarrow (1-\alpha_c)\mathbf{p}_c + h_\sigma \sqrt{\alpha_c(2-\alpha_c) m_{\text{eff}}}\, \Delta\mathbf{x}_{\text{CMA}}/\sigma. \tag{10}$$

In words, the covariance update performs smoothing over the optimization path to update $\mathbf{C}$ based on the within- and between-step covariance of well-performing solutions. Thus, Eq. (9) scales $\mathbf{C}$ along directions of successful steps to make the search converge faster.

3 A STEIN VARIATIONAL EVOLUTION STRATEGY

This section introduces a novel framework of using a multi-population ES for efficient discovery of multiple high-quality solutions to an optimization problem. The idea of this work is to represent each SVGD particle by the mean of an ES search distribution and use the estimated steps of the ES algorithm as the *driving force* in the SVGD particle update. Hence, our approach exploits the CMA-ES step-size adaptation mechanism to make gradient-free inference more efficient. Intuitively, the reformulated update permits larger particle updates, similar to momentum, especially in flat regions of the target. Since ES are easily parallelizable on modern GPUs [Lange, 2023, Tang et al., 2022], this approach comes at a small additional runtime cost. In the following, we use $\varrho \in \mathbb{N}^+$ to refer to the number of ES search distributions, $n \in \mathbb{N}^+$ to denote the size of each sampled population, and $m \in \mathbb{N}^+$ for the number of elite samples. This amounts to a total population size of ϱn for ES-based algorithms.

Based on the SVGD update in Eq. (2) and the CMA-ES update of the search distribution mean in Eq. (4), we now define *Stein Variational CMA-ES (SV-CMA-ES)*. The full algorithm is listed in Algorithm 1. SV-CMA-ES is a multi-population version of CMA-ES, where ϱ search distributions are updated in parallel, each representing an SVGD particle $\mathbf{x}_i$ via their distribution mean. In other words, for each particle, there is a corresponding Gaussian search distribution that is centered at the particle and parametrized as $\mathcal{N}(\mathbf{x}_i, \sigma_i^2 \mathbf{C}_i)$. Given the standard CMA-ES distribution update step $\Delta\mathbf{x}_{\text{CMA}}$ from Eq. (5) and a sampled population $\boldsymbol{\xi}_{ij} \sim \mathcal{N}(\mathbf{x}_i, \sigma_i^2 \mathbf{C}_i)$, we propose the following SVGD-based update:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon\, \phi(\mathbf{x}_i) \quad \text{with} \tag{11}$$

$$\begin{aligned}\phi(\mathbf{x}_i) &= \mathop{\mathbb{E}}_{\mathbf{x}_j \sim q}\left[k(\mathbf{x}_j, \mathbf{x}_i)\Delta\mathbf{x}_{j_{\text{CMA}}} + \nabla_{\mathbf{x}_j} k(\mathbf{x}_j, \mathbf{x}_i)\right] \\ &= \frac{1}{\varrho}\sum_{j=1}^{\varrho}\Bigg[\underbrace{\left[\sum_{\ell=1}^{m} w_{j\ell}(\boldsymbol{\xi}_{j\ell} - \mathbf{x}_j)\right] k(\mathbf{x}_j, \mathbf{x}_i)}_{\text{driving force}} + \underbrace{\nabla_{\mathbf{x}_j} k(\mathbf{x}_j, \mathbf{x}_i)}_{\text{repulsive force}}\Bigg]\end{aligned} \tag{12}$$

where we assume the same sorting by fitness in our sum as in vanilla CMA-ES and $w_{j\ell}$ are the sample weights that are computed based on the fitness values $f(\boldsymbol{\xi}_{j\ell})$ following Hansen [2016]. Further, we use an additional step-size hyperparameter ϵ for notational consistency with SVGD, but we always fix it to $\epsilon = 1$.

Eq. (12) defines how to update each particle search distribution mean. It now remains to define the remaining SV-CMA-ES parameter updates. The original CMA-ES step-size update (8) is based on the length of the distribution mean update step. In the particle update in Eq. (11), this quantity corresponds to the effective update step $\phi(\mathbf{x}_i)$. Given this particle shift, the smoothened step estimate $\mathbf{p}_{\sigma_i}$ is computed analogously to the CMA-ES optimization path update in Eq. (7):

$$\mathbf{p}_{\sigma_i} \leftarrow (1-\alpha_\sigma)\mathbf{p}_{\sigma_i} + \sqrt{\alpha_\sigma(2-\alpha_\sigma)\, m_{\text{eff},i}}\; \mathbf{C}_i^{-\frac{1}{2}}\phi(\mathbf{x}_i)/\sigma_i \tag{13}$$

Using the same construction, we update $\mathbf{p}_{c_i}$ based on $\phi(\mathbf{x}_i)$, from which the covariance $\mathbf{C}_i$ can be computed using Equation (9).

3.1 PRACTICAL CONSIDERATIONS

We now discuss some modifications to the algorithm that we found beneficial in practice. As noted earlier, the update of the particle in Eq. (12) smoothens the gradient approximation across all particles. As a result, the magnitude of the effective steps is reduced compared to standard CMA-ES. Since CMA-ES reduces the step-size σ automatically when small steps are taken, this may lead to premature convergence. An example that illustrates this problem is a bimodal distribution with both modes far apart, such that for most particles $k(\mathbf{x}, \mathbf{y})$ is close to zero for all pairs $\mathbf{x}, \mathbf{y}$ that are sampled from different modes. In this scenario, the driving force term of the update corresponds to the vanilla CMA-ES update, scaled down by the factor of $1/\varrho$. Hence, the proposed steps in this scenario would shrink iteratively. To address this issue, we propose the following simplified particle update:

$$\phi(\mathbf{x}_i) = \frac{1}{\varrho}\sum_{j=1}^{\varrho}\left[\left[\sum_{\ell=1}^{m} w_{i\ell}(\boldsymbol{\xi}_{i\ell} - \mathbf{x}_i)\right] + \nabla_{\mathbf{x}_j} k(\mathbf{x}_j, \mathbf{x}_i)\right]. \tag{14}$$

This update uses only the particle $\mathbf{x}_i$ to estimate the first term of the update, i.e., the driving force. We note that this corresponds to a hybrid kernel SVGD setting [D'Angelo et al., 2021, MacDonald et al., 2023], which uses two separate kernels to compute the repulsion and driving force terms: $\phi_{\text{hybrid}}(\mathbf{x}_i) = \mathbb{E}_{\mathbf{x}\sim q}\left[\nabla_{\mathbf{x}} f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}_i) + \nabla_{\mathbf{x}} k_2(\mathbf{x}, \mathbf{x}_i)\right]$ if we choose $k_1(\mathbf{x}, \mathbf{y}) = n\mathbb{1}(\mathbf{x} = \mathbf{y})$. This kernel can be approximated by an RBF kernel with small bandwidth $h \to 0$.

While the update in Eq. (14) does not possess the same capabilities of transporting particles "along a necklace" as the vanilla SVGD update (cf. Fig. 1 of Liu and Wang [2016]), it has been noted that these SVGD capabilities play a limited role for practical problems in the first place [D'Angelo and Fortuin, 2021]. Instead, prior work proposed the annealed update in Eq. (3) to transport the particles to regions of high density [D'Angelo and Fortuin, 2021, Liu et al., 2017]. In practice, we observe that using the annealed version of the above update, i.e.,

$$\begin{aligned}\phi(\mathbf{x}_i) &= \frac{1}{\varrho}\sum_{j=1}^{\varrho}\left[\left[\sum_{\ell=1}^{m} w_{i\ell}(\boldsymbol{\xi}_{i\ell} - \mathbf{x}_i)\right] + \gamma(t)\nabla_{\mathbf{x}_j} k(\mathbf{x}_j, \mathbf{x}_i)\right] \\ &= \sum_{\ell=1}^{m} w_{i\ell}(\boldsymbol{\xi}_{i\ell} - \mathbf{x}_i) + \frac{\gamma(t)}{\varrho}\sum_{j=1}^{\varrho}\nabla_{\mathbf{x}_j} k(\mathbf{x}_j, \mathbf{x}_i)\end{aligned} \tag{15}$$

ensures sufficient mode coverage to efficiently sample from distributions.

The substitution of the score function with the CMA-ES step introduces a bias in comparison to the SVGD update in Eq. (3), meaning it does not strictly adhere to the canonical SVGD framework and does not inherit its robust convergence properties. Still, we find that, in practice, the update makes a useful tradeoff which combines the computational efficiency of CMA-ES with the particle set entropy preservation capabilities of SVGD. We leave a more in-depth theoretical analysis of the algorithm for future work, and present our empirical findings in the subsequent section. For an empirical convergence analysis, we refer to Appendix C.4.

4 RELATED WORK

Stein Variational Gradient Descent Extensions SVGD is a popular method for sampling from unnormalized densities. As such, SVGD has been an active field of research and many extensions have been proposed. These include approaches to improve the performance in high dimensions, for instance using projections [Chen et al., 2019] or by adjusting the particle update to reduce its bias [D'Angelo and Fortuin, 2021, Ba et al., 2021]. Other extensions include non-Markovian steps [Ye et al., 2020, Liu et al., 2022], learning-based methods [Langosco et al., 2021, Zhao et al., 2023], and domain-specific kernel functions [Sharma et al., 2023, Barcelos et al., 2024]. While our focus lies on gradient-free SVGD approaches, most of these ideas could be integrated into our approach, which would be an interesting direction of future research.

Gradient-free sampling Many gradient-free sampling methods, like those in the MCMC family, iteratively update a proposal distribution to match the target [Andrieu et al., 2003]. A shortcoming of these approaches is their slower sampling procedure compared to SVGD, as they are prone to be trapped in a single mode over long periods of time on multimodal objectives. Population-based MCMC methods improve over this by running multiple chains in parallel, which exchange information over time [Laskey and Myers, 2003]. Notably, parallel tempering methods simulate chains with different temperatures in parallel to improve mode coverage [Swendsen and Wang, 1986]. Still, these methods commonly require sample rejections and potentially long burning-in periods. Gradient-free SVGD [Han and Liu, 2018, GF-SVGD] addresses this by estimating the gradient for SVGD on a surrogate distribution, which allows for interactions between all chains at each update step and fast convergence rates. Further work improved the computational efficiency of this method by fitting the surrogate to a limited set of points [Yan and Zou, 2021]. However, these surrogate-based methods require a well-chosen prior for surrogate initialization, as they lack an explicit exploration mechanism. Thus, in practical scenarios, a different gradient-free SVGD approach has been presented which relies on MC estimates of the gradient [Liu et al., 2017]. In this work, we present a novel perspective on gradient-free SVGD, which combines ideas from the literature on ES. Different from prior work, we propose a particle update that is based on CMA-ES, a highly efficient ES [Hansen and Kern, 2004].

Evolution Strategies ES are a specific class of blackbox optimization methods that iteratively improve a search dis-

Algorithm 1 Stein Variational CMA-ES. Differences to the parallel CMA-ES algorithm are highlighted in **blue**.

Input: Kernel $k(\cdot,\cdot)$; num. particles ϱ; subpop. size n; num. elites m; elite weights $w_{i=1\dots m}$; learning rates $\epsilon, \alpha_\sigma, \alpha_1, \alpha_m, \alpha_c$; damping hyperparam. d_σ; dimension d; num. iterations T

1: Initialize population parameters $\mathbf{x}_i, \sigma_i, \mathbf{C}_i$ for particles $i = 1, \dots, \varrho$
2: **for** iteration $t = 1, \dots, T$ **do**
3: **for** particle $i = 1, \dots, \varrho$ **do**
4: **Sample & evaluate new population**:
$\boldsymbol{\xi}_{ij} \sim \mathcal{N}(\mathbf{x}_i, \sigma_i^2 \mathbf{C}_i)$, for $j = 1, \dots, n$
$\mathbf{y}_{ij} = (\boldsymbol{\xi}_{ij} - \mathbf{x}_i)/\sigma_i$
5: **Estimate gradient & shift particle (Eq. 15)**:
Sort samples by $f_{ij} = f(\boldsymbol{\xi}_{ij})$ in ascending order
$$\phi(\mathbf{x}_i) = \sum_{\ell=1}^{m} w_{i\ell}(\boldsymbol{\xi}_{i\ell} - \mathbf{x}_i) + \frac{\gamma(t)}{\varrho} \sum_{j=1}^{\varrho} \nabla_{\mathbf{x}_j} k(\mathbf{x}_j, \mathbf{x}_i)$$
$\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon \phi(\mathbf{x}_i)$
6: **Cumulative step-size adaptation**:
$m_{\text{eff},i} = (\sum_{\ell=1}^{m} w_{i\ell}^2)^{-1}$
$\mathbf{p}_{\sigma_i} \leftarrow (1-\alpha_\sigma)\mathbf{p}_{\sigma_i}$
$+ \sqrt{\alpha_\sigma(2-\alpha_\sigma)\, m_{\text{eff},i}}\, \mathbf{C}_i^{-\frac{1}{2}} \phi(\mathbf{x}_i)/\sigma_i$
$\sigma_i \leftarrow \sigma_i \times \exp\Big(\frac{\alpha_\sigma}{d_\sigma}\Big(\frac{\|\mathbf{p}_{\sigma_i}\|}{\mathbb{E}\|\mathcal{N}(0,\mathbf{I})\|} - 1\Big)\Big)$
7: **Covariance matrix adaptation**:
$\bar{h} = \|\mathbf{p}_{\sigma_i}\| / \sqrt{1-(1-\alpha_\sigma)^{2(t+1)}}$
$h_{\sigma_i} = 1$ if $\bar{h} < (1.4 + \frac{2}{d+1})\,\mathbb{E}\|\mathcal{N}(0,\mathbf{I})\|$ else 0
$d(h_{\sigma_i}) = 1$ if $\alpha_c(1-h_\sigma)(2-\alpha_c) \le 1$ else 0
$\bar{w}_{ij} = w_{ij}$ if $w_{ij} \ge 0$ else $d/\|\mathbf{C}_i^{-\frac{1}{2}}\mathbf{y}_{ij}\|^2$
$\mathbf{p}_{c_i} \leftarrow (1-\alpha_c)\mathbf{p}_{c_i} + h_{\sigma_i}\sqrt{\alpha_c(2-\alpha_c)m_{\text{eff},i}}\, \phi(\mathbf{x}_i)/\sigma_i$
$\mathbf{C}_i \leftarrow (1 + d(h_{\sigma_i}) - \alpha_1 - \alpha_m \sum_{j=1}^{n} w_{ij})\mathbf{C}_i$
$+ \alpha_1 \mathbf{p}_{c_i}\mathbf{p}_{c_i}^T + \alpha_m \sum_{j=1}^{n} \bar{w}_{ij}\mathbf{y}_{ij}\mathbf{y}_{ij}^T$
8: **end for**
9: **end for**

tribution over solution candidates by implementing specific sampling, evaluation and update mechanisms [Rechenberg, 1978]. While ES commonly use a single distribution [Li et al., 2020], it has been demonstrated that their efficiency can be improved by employing restarts or multiple runs in parallel [Auger and Hansen, 2005, Pugh et al., 2016]. For instance, restarts with increasing population sizes have been demonstrated to improve the performance of CMA-ES [Loshchilov et al., 2012]. A downside of restarting approaches is their sequential nature, which makes them slower and prohibits exploiting the benefits of modern GPUs. Our method is different as it uses the SVGD update to sample multiple subpopulations in parallel, which naturally enables to explore multiple modes. In particular, our proposed SVGD-based update is simpler to compute than other distributed updates [Wang et al., 2019b], yet more informed than uncoordinated parallel runs.

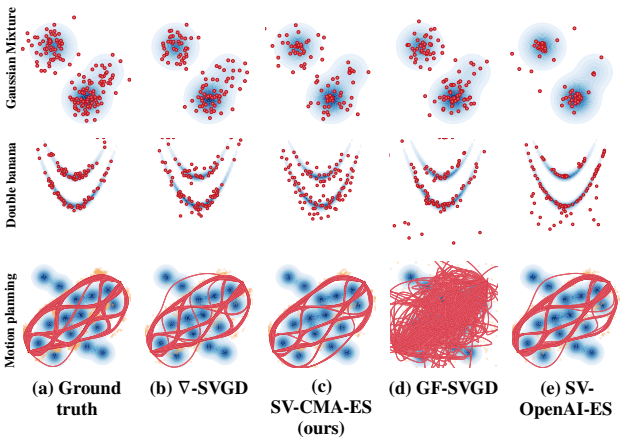


Figure 2: Samples obtained by various methods. Gradient-based SVGD (b) captures all target densities effectively, while SV-CMA-ES produces the highest quality samples among gradient-free methods. GF-SVGD struggles on more complex targets, and SV-OpenAI-ES tends to converge slowly due to taking small steps in flat regions of the target.

5 EXPERIMENTS

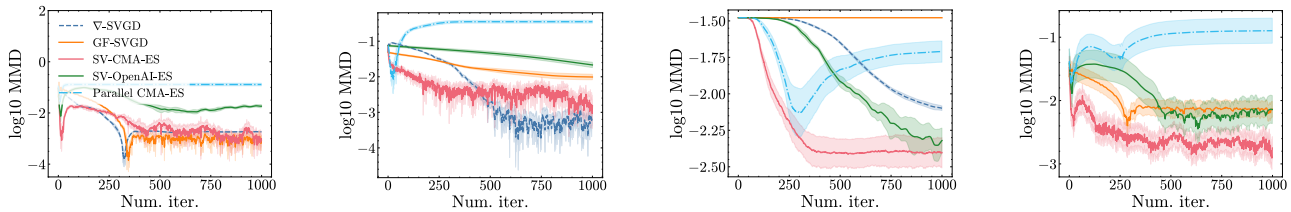
We compare SV-CMA-ES against the two existing approaches for zero-order SVGD from the literature: *GF-SVGD* as state-of-the-art method for surrogate-based inference, and the MC gradient SVGD as state-of-the-art gradient approximation method. We refer to the latter as *SV-OpenAI-ES* throughout the remainder of the paper following the naming convention of the ES community. Furthermore, we compare against gradient-based SVGD, which we denote as ∇-SVGD in the following. All strategies have been implemented based on the evosax library [Lange, 2023].

To guarantee a fair comparison, we keep the number of function evaluations equal for all methods. In other words, if the ES-based methods are evaluated for 4 particles, each sampling subpopulations of size 16, we evaluate GF-SVGD and ∇-SVGD with 64 particles. For each kernel-based method, we use the standard RBF kernel. For GF-SVGD, we follow the setup of Han and Liu [2018] and use the same kernel function for the SVGD kernel k and the surrogate kernel k_ρ, as well as an isotropic Gaussian prior $\mathcal{N}(0, \sigma^2\mathbf{I})$. The optimal scale σ^2 of the prior is found via a hyperparameter grid search. For each method, we implement the annealed version of ∇-SVGD (cf. Eq. (3)) using a logarithmic schedule $\gamma(t) = \max(\log(T/t), 1)$. Unless specified differently, this choice is followed in all experiments. For all methods that require an internal optimizer, we use Adam [Kingma and Ba, 2015]. We carefully search for the best hyperparameters for each algorithm separately, to guarantee a fair comparison. The full details of our experimental setup can be found in the Appendix B. Moreover, we refer to Appendix C for additional results including ablation studies and empirical runtime and convergence analyses.

5.1 SAMPLING FROM SYNTHETIC DENSITIES

Setting We first evaluate our method on multiple synthetic densities to illustrate the quality of the generated samples. The closed form pdf for every problem is listed in Appendix B. We use a total population size of 400 for all methods, which is split across 100 particles for the ES-based algorithms. In other words, each particle samples an ES population of 4. Following common practice in the literature, we quantify sampling performance by evaluating the Maximum Mean Discrepancy [Gretton et al., 2012, MMD] of the particles with respect to ground truth samples. We additionally evaluate the scaling to higher particle numbers in Sec. 5.4.

Results Figures 2 and 3 display the qualitative and quantitative sampling results. As expected, ∇-SVGD generates high-quality samples for all problems. We find that among the gradient-free methods, SV-CMA-ES performs the best across all problems. While GF-SVGD generates high-quality samples for the Gaussian mixture, the variance of the generated samples on the double banana density is too high, and the samples for the motion planning problem are of poor quality. Concurrently, SV-OpenAI-ES performs well on the motion planning problem, but on the others it underestimates the variance (Gaussian mixture) or converges slowly (double banana, also see Fig. 13). These results highlight the fast convergence properties of our method, as it employs the automatic step-size adaptation of CMA-ES. Additionally, Figure 3 (d) displays the MMD w.r.t. the samples that were obtained by ∇-SVGD, aggregated across all sampling tasks. Our results demonstrate that SV-CMA-ES can indeed quickly converge to a set of samples that approximates the outcomes of ∇-SVGD well, and better than the two other



(a) Gaussian Mixture (b) Double Banana (c) Motion Planning (d) MMD w.r.t. ∇-SVGD

Figure 3: **(a)-(c):** MMD w.r.t. *ground truth* samples on the synthetic densities depicted in Figure 2. **(d):** Mean log10 MMD across all three sampling tasks w.r.t. the *samples obtained by gradient-based SVGD*. All results are averaged across 10 independent runs (±1.96 standard error). SV-CMA-ES approximates the ground truth samples and results by gradient-based SVGD (blue line) the best out of all gradient-free methods.

gradient-free baselines. We further illustrate these results in Appendix C.5 where we display the sample sets for all sampling tasks. Moreover, we illustrate the benefit of using the presented algorithm compared to other CMA-ES-based methods in Fig. 1 – since prior CMA-ES methods only maximize likelihood, the diversity of samples is low.

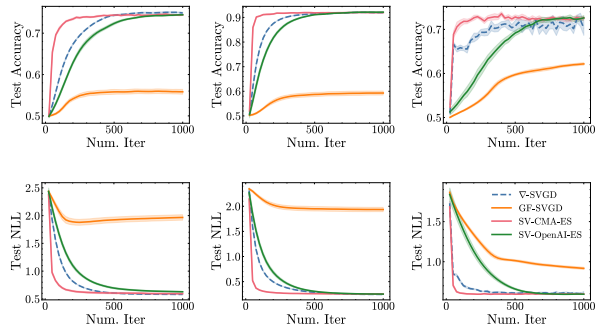
5.2 BAYESIAN LOGISTIC REGRESSION

Setting Next, we evaluate our method on Bayesian logistic regression for binary classification. We follow the setup of Langosco et al. [2021], which uses a hierarchical prior $p(\theta)$ on the parameters $\theta = [\alpha, \beta]$, where $\beta \sim \mathcal{N}(0, \alpha^{-1})$ and $\alpha \sim \Gamma(a_0, b_0)$. Given data D, the task is to approximate samples from the posterior

$$p(\theta \mid D) = p(D \mid \theta)p(\theta) \quad \text{with:}$$
$$p(D \mid \theta) = \prod_{i=1}^{N} \Big[y_i \frac{\exp(x_i^T \beta)}{1+\exp(x_i^T \beta)} + (1 - y_i) \frac{\exp(-x_i^T \beta)}{1+\exp(-x_i^T \beta)} \Big].$$

We consider the binary *Covtype*, *Spambase*, and the *German credit* datasets from the UCI Machine Learning Repository [Kelly et al., 2023], as suggested in prior work [Liu and Wang, 2016, Arenz et al., 2020, Futami et al., 2018]. For all experiments, we use a total population of 256, which is split across 8 particles for the ES-based methods.

Results For each dataset, we report the accuracy and negative log-likelihood (NLL) across the entire particle set, and report the mean performance across 10 runs. Our results demonstrate that SV-CMA-ES outperforms the remaining gradient-free algorithms. On both datasets, our method is the fastest converging among the gradient-free methods. Furthermore, its final performance is considerably better than GF-SVGD on all datasets. While the performance of ∇-SVGD is slightly better on the Covtype dataset, SV-CMA-ES is on par with it for the Spam dataset. Additionally, on the credit data, we find that ES-based methods are both more accurate and exhibit greater stability than the gradient-based SVGD, which underlines the potential of zero-order methods in this context.



(a) Covtype (b) Spam (c) Credit

Figure 4: Results of Bayesian logistic regression. We report mean (±1.96 standard error) across 10 independent runs. SV-CMA-ES converges the faster than other gradient-free methods, and achieves similar performance levels at convergence as gradient-based SVGD (dashed line).

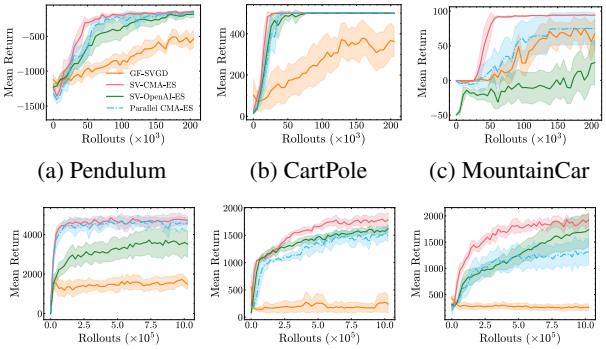
5.3 REINFORCEMENT LEARNING

Setting We further assess the performances of the gradient-free SVGD methods on six classic reinforcement learning (RL) problems. The goal of each RL task is to maximize the expected episodic return $J(\theta)$, where each particle θ now parametrizes a multi-layer perceptron (MLP). The corresponding inference objective is to sample policy parameters θ from the following Boltzmann distribution:

$$p(\theta) \propto \exp(J(\theta)), \quad J(\theta) = \mathop{\mathbb{E}}_{(s_t, a_t) \sim \pi_\theta} \Big[\sum_{t=1}^{T} r(s_t, a_t) \Big]$$

where $(s_t, a_t) \sim \pi_\theta$ represent a trajectory sampled from the distribution that is induced by the policy parametrized by θ. For each problem, we train a 2-hidden layer MLP with 16 units per layer, which implies high-dimensional optimization problems as each MLP has several hundred parameters. The

specific numbers vary across the benchmarks and are listed in Table 2 in the Appendix. We use a total population size of 64 which we split into 4 subpopulations for the ES-based methods and estimate the expected return across 16 rollouts with different seeds. To make the results comparable to other works on ES for RL, we follow the approach of Lee et al. [2023] and extend the optimization by a phase that attempts to find exact optima. We realize this by fading out the repulsive term via the schedule $\gamma(t) = \log(T/t)$.



(a) Pendulum (b) CartPole (c) MountainCar

(d) Halfcheetah (e) Hopper (f) Walker

Figure 5: Results of sampling MLP parameters for RL tasks. Plotted is the best expected return across all particles for each method. We report the mean (±1.96 standard error) across 10 independent runs. SV-CMA-ES performs better than the gradient-free baselines across all tasks.

Results We display the aggregated results across all RL tasks in Fig. 1, and the individual task performances in Fig. 5. Our results showcase a strong performance of SV-CMA-ES. In comparison to other gradient-free versions of SVGD, it is the only method that generates high scoring solutions for all problems. In particular, we observe that SV-CMA-ES is the only method that solves the MountainCar problem consistently, while it is the fastest to converge on Pendulum. Both of these environments feature a local optimum at which agents remain idle to avoid control costs [Eberhard et al., 2023]. It is on these problems that GF-SVGD converges to such optima in certain runs, which we further illustrate in Fig. 8 in the Appendix. These results illustrate that SV-CMA-ES improves over GF-SVGD by sampling stochastic ES steps, which leads to a higher exploration of the domain. Interestingly, our results further show that SV-OpenAI-ES may deliver good samples in some runs, but the high standard error on several problems underline its sensitivity to initialization. These findings confirm that SV-CMA-ES is a strong gradient-free SVGD scheme, capable of sampling from densities and optimizing blackbox objectives. Further, we would like to note that our final performances are comparable to those reported in prior, gradient-based, work [Jesson et al., 2024], which again underlines the potential of our method.

Further, we analyze the benefits of the kernel term by comparing our method to uncoordinated parallel runs of CMA-ES. Overall, we observe a clear performance improvement when using the kernel term. In particular, in the more challenging Hopper and Walker tasks, the benefits of using SV-CMA-ES over parallel CMA-ES are large. We extend our analysis of SV-CMA-ES in Appendix C.1 where we compare it to vanilla CMA-ES and OpenAI-ES, and conduct additional experiments on sparse reward environments. This analysis reveals that SV-CMA-ES consistently outperforms competing ES, underscoring its superior performance in environments where effective exploration is essential.

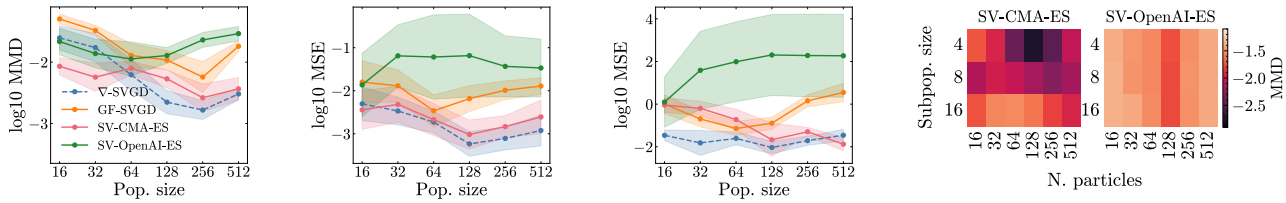
5.4 ABLATION STUDIES

Choice of Population Size In the experiments above, we investigate the performance for fixed particle numbers and population sizes. To gain further insights into the scalability of our method, we conduct an additional analysis on the same sampling problems as in Section 5.1 using varying particle numbers. The results of these experiments are displayed in Figure 6. In addition to the MMD after 1 000 iterations, we report the error when estimating the first two central moments of the target distribution from the generated samples. We observe the clear trend that SV-CMA-ES performs better than GF-SVGD and SV-OpenAI-ES with increasing particle numbers. Furthermore, we observe in Fig. 6 (d) that SV-CMA-ES requires fewer samples than SV-OpenAI-ES to estimate good steps.

Choice of Annealing Schedule In the experiments above, we used annealed SVGD. This decision was made due to its widespread use in the community and many desirable properties. However, to assess the quality of our method, it is important to consider the sensitivity to the choice of annealing schedule. In Table 1, we show the key performance metrics from 10 seeds for SV-CMA-ES with and without annealing. As we see, performance in all cases is strong, with little difference between the two conditions.

Task	Annealing	No Annealing
GMM	-3.03 (0.30)	-2.92 (0.26)
Double Banana	-2.59 (0.16)	-2.83 (0.21)
Motion Planning	-2.40 (0.10)	-2.44 (0.13)
Covertype NLL	0.59 (0.01)	0.59 (0.01)
MountainCar RL	93.68 (0.10)	93.68 (0.08)
Hopper RL	1781.31 (132.64)	1788.13 (79.22)

Table 1: Kernel annealing ablation. This table shows SV-CMA-ES performances across 10 seeds with 1.96 standard error in parentheses. All runs use the identical setup, aside from the kernel annealing. No annealing means that we use a constant $\gamma(t) = 1$.



(a) MMD w.r.t. ground truth (b) $\mathbb{E}[x]$ (c) $\mathbb{V}[x]$ (d) Subpopulation Scaling

Figure 6: Scaling analysis. Depicted are the final performances for different total population sizes. **(a)**: MMD vs. sample size after 1000 iterations. **(b)-(c)**: MSE vs. sample size when estimating the first two central moments of the ground truth distribution. For ES, we use the same subpop. size per particle as in Figure 3. **(d)**: Subpopulation size scaling for ES-based SVGD. The results are averaged across 10 independent runs of all synthetic sampling tasks from Figure 3. SV-CMA-ES performs the best out of all gradient-free methods (solid lines) across different particle numbers.

6 CONCLUSION

Summary We proposed a new gradient-free algorithm that combines elements from evolution strategies and SVGD. The resulting method, SV-CMA-ES, achieves high computational efficiency by replacing the score term in the SVGD update with an ES step. On several problems with different characteristics, we demonstrated that SV-CMA-ES outperforms prior gradient-free SVGD-based algorithms consistently. We could thus confirm our hypothesis that the incorporation of the CMA-ES update enables faster convergence than SVGD with MC gradients, and better overall performance than GF-SVGD across multiple problems.

Limitations For stable convergence, we selected a fixed kernel bandwidth via grid search, while prior work used the median heuristic. Selecting the kernel bandwidth via grid search is costly and thus constitutes a disadvantage. Furthermore, our approach can be computationally expensive due to the decomposition required for each covariance matrix, leading to a runtime complexity in $\mathcal{O}(\varrho^2 d + \varrho d^3)$ in d dimensions with ϱ particles. In contrast, SV-OpenAI-ES and GF-SVGD achieve a complexity in $\mathcal{O}(\varrho^2 d)$. Future work could address this by exploring diagonal covariance matrices, which are commonly used to speed up CMA-ES [Ros and Hansen, 2008]. Additionally, we would like to stress that the most time-consuming part of ES is often the fitness evaluation. We illustrate this aspect in Appendix C.3 where we present additional plots including empirical runtimes. This analysis shows that the wallclock time that SV-CMA-ES requires to produce high-quality solutions is competitive with the baselines.

Future work In our experiments, we used the standard RBF kernel, following the convention of many prior works. Recent work suggested adjusting the size of the considered neighborhood adaptively in the context of particle swarm optimization [Zhang et al., 2024]. One potential extension is to integrate this idea into our approach to improve particle repulsion. Moreover, we see, for instance in Figure 3, that our method has higher variance compared to other methods. Future work could investigate mechanisms to make the optimization more stable. Finally, we see a potential to scaling up our method to a high number of particles to parallelize ES in an informed way. An investigation of scaling laws would be an intriguing avenue of research.

Acknowledgements

This research was funded by the Amazon Development Center Germany GmbH. The authors want to thank anonymous reviewers for constructive feedback which helped improve the paper.

References

Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. Theoretical foundation for cma-es from information geometry perspective. *Algorithmica*, 64:698–716, 2012.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50:5–43, 2003.

Oleg Arenz, Mingjun Zhong, and Gerhard Neumann. Trust-region variational inference with gaussian mixture models. *Journal of Machine Learning Research*, 21(163):1–60, 2020.

Anne Auger and Nikolaus Hansen. A restart cma evolution strategy with increasing population size. In *2005 IEEE congress on evolutionary computation*, volume 2, pages 1769–1776. IEEE, 2005.

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 (ICLR)*, 2021.

Lucas Barcelos, Tin Lai, Rafael Oliveira, Paulo Borges, and Fabio Ramos. Path signatures for diversity in probabilistic trajectory optimisation. *The International Journal of Robotics Research*, 0(0):02783649241233300, 2024. doi: 10.1177/02783649241233300. URL https://doi.org/10.1177/02783649241233300.

Jock A Blackard. *Comparison of neural networks and discriminant analysis in predicting forest cover types*. Colorado State University, 1998.

Peng Chen, Keyi Wu, Joshua Chen, Tom O'Leary-Roseberry, and Omar Ghattas. Projected stein variational newton: A fast and scalable bayesian inference method in high dimensions. *Advances in Neural Information Processing Systems*, 32, 2019.

Francesco D'Angelo and Vincent Fortuin. Annealed stein variational gradient descent. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2021. URL https://openreview.net/forum?id=pw2v8HFJIYg.

Francesco D'Angelo, Vincent Fortuin, and Florian Wenzel. On stein variational neural network ensembles. *arXiv preprint arXiv:2106.10760*, 2021.

Gianluca Detommaso, Tiangang Cui, Youssef Marzouk, Alessio Spantini, and Robert Scheichl. A stein variational newton method. *Advances in Neural Information Processing Systems*, 31, 2018.

Onno Eberhard, Jakob Hollenstein, Cristina Pinneri, and Georg Martius. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Peter Englert and Marc Toussaint. Learning manipulation skills from a single demonstration. *The International Journal of Robotics Research*, 37(1):137–154, 2018.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Futoshi Futami, Issei Sato, and Masashi Sugiyama. Variational inference based on robust divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 813–822. PMLR, 2018.

Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, and Jürgen Schmidhuber. Exponential natural evolution strategies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 393–400, 2010.

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.

Jun Han and Qiang Liu. Stein variational gradient descent without gradient. In *International Conference on Machine Learning*, pages 1900–1908. PMLR, 2018.

Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Nikolaus Hansen and Stefan Kern. Evaluating the cma evolution strategy on multimodal test functions. In *International conference on parallel problem solving from nature*, pages 282–291. Springer, 2004.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, pages 1689–1696, 2010.

Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. Spambase data set, 1999.

Julius Jankowski, Lara Brudermüller, Nick Hawes, and Sylvain Calinon. Vp-sto: Via-point-based stochastic trajectory optimization for reactive robot behavior. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10125–10131. IEEE, 2023.

Andrew Jesson, Chris Lu, Gunshi Gupta, Nicolas Beltran-Velez, Angelos Filos, Jakob N Foerster, and Yarin Gal. Relu to the rescue: improve your on-policy actor-critic with positive advantages. In *Proceedings of the 41st International Conference on Machine Learning*, pages 21577–21605, 2024.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The uci machine learning repository, 2023. URL https://archive.ics.uci.edu.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Alexander Lambert, Fabio Ramos, Byron Boots, Dieter Fox, and Adam Fishman. Stein variational model predictive control. In *Conference on Robot Learning*, pages 1278–1297. PMLR, 2021.

Robert Tjarko Lange. gymnax: A jax-based reinforcement learning environment library, 2022. URL http://github.com/RobertTLange/gymnax.

Robert Tjarko Lange. evosax: Jax-based evolution strategies. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 659–662, 2023.

Lauro Langosco, Vincent Fortuin, and Heiko Strathmann. Neural variational gradient descent. *arXiv e-prints*, pages arXiv–2107, 2021.

Kathryn Blackmond Laskey and James W Myers. Population markov chain monte carlo. *Machine Learning*, 50:175–196, 2003.

Yewon Lee, Yizhou Huang, Krishna Murthy Jatavallabhula, Andrew Zou Li, Fabian Damken, Eric Heiden, Kevin A Smith, Derek Nowrouzezahrai, Fabio Ramos, and Florian Shkurti. Stamp: Differentiable task and motion planning via stein variational gradient descent. In *CoRL 2023 Workshop on Learning Effective Abstractions for Planning (LEAP)*, 2023.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in Neural Information Processing Systems*, 31, 2018.

Zhenhua Li, Xi Lin, Qingfu Zhang, and Hailin Liu. Evolution strategies for continuous optimization: A survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 56:100694, 2020.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in Neural Information Processing Systems*, 29, 2016.

Xing Liu, Harrison Zhu, Jean-Francois Ton, George Wynne, and Andrew Duncan. Grassmann stein variational gradient descent. In *International Conference on Artificial Intelligence and Statistics*, pages 2002–2021. PMLR, 2022.

Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. In *33rd Conference on Uncertainty in Artificial Intelligence, UAI 2017*, 2017.

Open Data LMU. Kreditscoring zur klassifikation von kreditnehmern, 2010. URL `https://data.ub.uni-muenchen.de/23/`.

Ilya Loshchilov, Marc Schoenauer, and Michele Sebag. Alternative restart strategies for cma-es. In *International Conference on Parallel Problem Solving from Nature*, pages 296–305. Springer, 2012.

Anson MacDonald, Scott A Sisson, and Sahani Pathiraja. Hybrid kernel stein variational gradient descent, 2023. URL `https://openreview.net/forum?id=cbu1lIYQ19`.

Fahira Afzal Maken, Fabio Ramos, and Lionel Ott. Stein icp for uncertainty estimation in point cloud matching. *IEEE robotics and automation letters*, 7(2):1063–1070, 2021.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

Natalie Maus, Kaiwen Wu, David Eriksson, and Jacob Gardner. Discovering many diverse solutions with bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 1779–1798. PMLR, 2023.

Bogdan Mazoure, Thang Doan, Audrey Durand, R Devon Hjelm, and Joelle Pineau. Leveraging exploration in off-policy algorithms via normalizing flows. *Proceedings of the 3rd Conference on Robot Learning (CoRL 2019)*, 2019.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Dokl. Akad. Nauk. SSSR*, volume 269, page 543, 1983.

Jack Parker-Holder, Luke Metz, Cinjon Resnick, Hengyuan Hu, Adam Lerer, Alistair Letcher, Alexander Peysakhovich, Aldo Pacchiano, and Jakob Foerster. Ridge rider: Finding diverse solutions by following eigenvectors of the hessian. *Advances in Neural Information Processing Systems*, 33:753–765, 2020.

Jana Pavlasek, Stanley Robert Lewis, Balakumar Sundaralingam, Fabio Ramos, and Tucker Hermans. Ready, set, plan! planning to goal sets using generalized bayesian inference. In *Conference on Robot Learning*, pages 3672–3686. PMLR, 2023.

Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:202845, 2016.

Ingo Rechenberg. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie: Workshop, Hannover, 29. Sept.–1. Okt. 1977*, pages 83–114. Springer, 1978.

Raymond Ros and Nikolaus Hansen. A simple modification in cma-es achieving linear time and space complexity. In *International conference on parallel problem solving from nature*, pages 296–305. Springer, 2008.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Madhav Shekhar Sharma, Thomas Power, and Dmitry Berenson. Task-space kernels for diverse stein variational mpc. In *IROS 2023 Workshop on Differentiable Probabilistic Robotics: Emerging Perspectives on Robot Learning*, 2023.

Ryan Sullivan, Jordan K Terry, Benjamin Black, and John P Dickerson. Cliff diving: Exploring reward surfaces in reinforcement learning environments. In *International Conference on Machine Learning*, pages 20744–20776. PMLR, 2022.

Robert H Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

Yujin Tang, Yingtao Tian, and David Ha. Evojax: Hardware-accelerated neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 308–311, 2022.

Marc Toussaint, Cornelius V. Braun, and Joaquim Ortiz-Haro. Nlp sampling: Combining mcmc and nlp methods for diverse constrained sampling. *arXiv preprint arXiv:2407.03035*, 2024.

Dilin Wang, Ziyang Tang, Chandrajit Bajaj, and Qiang Liu. Stein variational gradient descent with matrix-valued kernels. *Advances in Neural Information Processing Systems*, 32, 2019a.

Zi-Jia Wang, Zhi-Hui Zhan, and Jun Zhang. Distributed minimum spanning tree differential evolution for multimodal optimization problems. *Soft Computing*, 23:13339–13349, 2019b.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Liang Yan and Xiling Zou. Gradient-free stein variational gradient descent with kernel approximation. *Applied Mathematics Letters*, 121:107465, 2021.

Mao Ye, Tongzheng Ren, and Qiang Liu. Stein self-repulsive dynamics: Benefits from past samples. *Advances in Neural Information Processing Systems*, 33:241–252, 2020.

Jianyi Zhang, Ruiyi Zhang, Lawrence Carin, and Changyou Chen. Stochastic particle-optimization sampling and the non-asymptotic convergence theory. In *International Conference on Artificial Intelligence and Statistics*, pages 1877–1887. PMLR, 2020.

Yanbo Zhang, Benedikt Hartl, Hananel Hazan, and Michael Levin. Diffusion models are evolutionary algorithms. *arXiv preprint arXiv:2410.02543*, 2024.

Yao Zhang et al. Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning. *Chemical science*, 10(35):8154–8163, 2019.

Qian Zhao, Hui Wang, Xuehu Zhu, and Deyu Meng. Stein variational gradient descent with learned direction. *Information Sciences*, 637:118975, 2023.

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.

Supplementary Material

The supplementary material is structured as follows:

- **Appendix A** lists the vanilla CMA-ES algorithm for comparison to our method and the computation of all hyperparameters that we used.
- **Appendix B** presents the full experimental details of our work.
- **Appendix C** presents additional experimental results. These include ablations of SV-CMA-ES, an empirical runtime analysis, and an empirical convergence analysis.

Algorithm 2 The Vanilla CMA-ES Update. Adapted from Hansen [2016].

Input: Generation index t, search distribution parameters $\mathbf{x}, \sigma, \mathbf{C}$; pop. size n, num. elites m; learning rates $\epsilon, \alpha_\sigma, \alpha_1, \alpha_m, \alpha_c$; damping hyperparam. d_σ

1: **Sample and evaluate new population of search points**, for $i = 1, \ldots, n$

$$\mathbf{y}_i = (\boldsymbol{\xi}_i - \mathbf{x})/\sigma, \quad \text{where } \boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{x}, \sigma^2\mathbf{C})$$
$$f_i = f(\boldsymbol{\xi}_i)$$

2: **Selection and recombination**
Sort $\mathbf{y}_i$ by f_i in ascending order, for $i = 1, \ldots, n$

$$\Delta\mathbf{x}_{\text{CMA}} = \sigma \sum_{i=1}^{m} w_i \mathbf{y}_i \qquad \text{where } \sum_{i=1}^{m} w_i = 1, \text{ and } \forall i \in \{1, \ldots, m\}.\ w_i > 0$$
$$\mathbf{x} \leftarrow \mathbf{x} + \epsilon\, \Delta\mathbf{x}_{\text{CMA}}$$

3: **Cumulative step-size adaptation**

$$\mathbf{p}_\sigma \leftarrow (1-\alpha_\sigma)\mathbf{p}_\sigma + \sqrt{\alpha_\sigma(2-\alpha_\sigma)\, m_{\text{eff}}}\, \mathbf{C}^{-\frac{1}{2}} \Delta\mathbf{x}_{\text{CMA}}/\sigma, \qquad \text{where } m_{\text{eff}} = (\textstyle\sum_{i=1}^{m} w_i^2)^{-1}$$
$$\sigma \leftarrow \sigma \times \exp\left(\frac{\alpha_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma\|}{\mathbb{E}\|\mathcal{N}(0,\mathbf{I})\|} - 1\right)\right)$$

4: **Covariance matrix adaptation**

$$h_\sigma = \mathbb{1}\left(\frac{\|\mathbf{p}_\sigma\|}{\sqrt{1-(1-\alpha_\sigma)^{2(t+1)}}} < (1.4 + \tfrac{2}{d+1})\, \mathbb{E}\|\mathcal{N}(0,\mathbf{I})\|\right)$$
$$\bar{w}_j = w_j \times \left(1 \text{ if } w_j \geq 0 \text{ else } d/\|\mathbf{C}^{-\frac{1}{2}}\mathbf{y}_j\|^2\right)$$
$$d(h_\sigma) = \mathbb{1}\left(\alpha_c(1-h_\sigma)(2-\alpha_c) \leq 1\right)$$
$$\mathbf{p}_c \leftarrow (1-\alpha_c)\mathbf{p}_c + h_\sigma\sqrt{\alpha_c(2-\alpha_c)m_{\text{eff}}}\, \Delta\mathbf{x}_{\text{CMA}}/\sigma$$
$$\mathbf{C} \leftarrow (1 + \alpha_1 d(h_\sigma) - \alpha_1 - \alpha_m \textstyle\sum_{i=1}^{n} w_i)\mathbf{C} + \alpha_1 \mathbf{p}_c {\mathbf{p}_c}^T + \alpha_m \textstyle\sum_{i=1}^{n} \bar{w}_i \mathbf{y}_i \mathbf{y}_i^T$$

A VANILLA CMA-ES ALGORITHM

The listing in Algorithm 2 displays the update step of vanilla CMA-ES at iteration t. Parallel CMA-ES, on which we base our method, uses the same update, but carries it out ϱ times in parallel, once for each search distribution. We would like to

point out that our notation differs from the standard notation of the ES community. While the number of sampled proposals in each iteration is classically denoted by λ and the number of selected elites by μ [Hansen, 2016], we deviate from this notation in the present work. Following the main paper, we denote the number of sampled candidates by $n \in \mathbb{N}^+$, and the number of elites by m. In our experiments, we tune the elite population size $m \in \mathbb{N}^+$, the initial value for σ using a grid search (cf. Appendix B) and use the defaults from the evosax codebase [Lange, 2023] for the remaining parameters. These hyperparameters should coincide with those of Hansen [2016].

A.1 COMPUTATION OF THE RECOMBINATION WEIGHTS AND HYPERPARAMETERS

For completeness, we list the computation of the recombination weights and all remaining hyperparameters in this section. Each population uses the same recombination weights, so $w_{ik} = w_{jk}$ for populations i and j. For simplicity, we therefore drop the population indices in the weights and define them for the single population case. For this, we use the equations (49)-(58) from Hansen [2016] with an *elite* population size of m. Please note that the aforementioned work uses different notation and denotes the population size by λ, and the number of elites by μ.

Given population size n and problem dimensionality d, we compute the set of weights $w_1, \dots, w_n$ as follows:

$$\epsilon = 1 \tag{16}$$

$$w'_i = \ln\left(\frac{n+1}{2}\right) - \ln(i), \quad \text{for } i = 1, \dots, n \tag{17}$$

$$m_{\text{eff}} = \frac{(\sum_{i=1}^{m} w'_i)^2}{\sum_{i=1}^{m} w'^2_i} \qquad \text{Note that this is equivalent to the previous definition in (6)} \tag{18}$$

$$m^-_{\text{eff}} = \frac{(\sum_{i=m+1}^{n} w'_i)^2}{\sum_{i=m+1}^{n} w'^2_i} \tag{19}$$

$$\alpha_\sigma = \frac{m_{\text{eff}} + 2}{d + m_{\text{eff}} + 5} \tag{20}$$

$$d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{m_{\text{eff}} - 1}{d+1}} - 1\right) + \alpha_\sigma \tag{21}$$

$$\alpha_c = \frac{4 + m_{\text{eff}}/d}{d + 4 + 2m_{\text{eff}}/d} \tag{22}$$

$$\alpha_1 = \frac{2}{(d+1.3)^2 + m_{\text{eff}}} \tag{23}$$

$$\alpha_m = \min\left(1 - \alpha_1, 2\frac{1/4 + m_{\text{eff}} + 1/m_{\text{eff}} - 2}{(d+2)^2 + m_{\text{eff}}}\right) \tag{24}$$

$$\beta_m = 1 + \alpha_1/\alpha_m \tag{25}$$

$$\beta_{m_{\text{eff}}} = 1 + \frac{2m^-_{\text{eff}}}{m_{\text{eff}} + 2} \tag{26}$$

$$\beta_{\text{pos def}} = \frac{1 - \alpha_1 - \alpha_m}{d\alpha_m} \tag{27}$$

$$w_i = \begin{cases} \frac{1}{\sum_{i=1}^{n} \max(w'_i, 0)} w'_i & \text{if } w'_i \geq 0 \\ \frac{\min(\beta_m, \beta_{m_{\text{eff}}}, \beta_{\text{pos def}})}{\sum_{i=1}^{n} |\min(w'_i, 0)|} w'_i & \text{else} \end{cases} \tag{28}$$

B EXPERIMENTAL DETAILS

This section lists the full experimental details for this paper. The code is partially based on Langosco et al. [2021] and Lange [2023]. For the experiments including existing ES such as vanilla CMA-ES, we use the implementations in `evosax` [Lange, 2023]. All the experiments are performed on an internal cluster with eight NVIDIA A40 GPUs. The code to reproduce our experiments and plots will be made available upon conference publication.

Each experiment is repeated 10 times using randomly generated seeds. In each plot, we report the mean performance across all 10 runs and 1.96 standard error bars. To compute the maximum mean discrepancy [Gretton et al., 2012, MMD] we use the

RBF kernel for which we select the bandwidth based on the median distance between the ground truth samples [Han and Liu, 2018]. For all experiments, we use the standard RBF kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/2h)$ for which we find the bandwidth h via grid search. Following Salimans et al. [2017] we use a rank transformation for fitness shaping for all OpenAI-ES-based methods. For GF-SVGD, we follow Han and Liu [2018] in using a Gaussian prior $\mathcal{N}(0, \sigma^2\mathbf{I})$, where σ^2 is determined via grid search.

B.1 HYPERPARAMETER TUNING

We tune the hyperparameters for each method separately. For all methods, this includes the kernel bandwidth. For SVGD, GF-SVGD and SV-OpenAI-ES, we additionally tune the Adam learning rate, while the initial step-size σ for SV-CMA-ES is selected analogously. For GF-SVGD, we follow Han and Liu [2018] in using a Gaussian prior centered at the origin, with isotropic covariance. The scale of the prior covariance is also determined via grid search. The ranges over which we search are listed in the following subsections in the corresponding hyperparameter paragraph of each experimental subsection. For SV-OpenAI-ES, we additionally tune the width σ of the proposal distribution, and for SV-CMA-ES we select the elite population size m via grid search. All remaining hyperparameters for the CMA-ES- and OpenAI-ES-based algorithms are chosen to be the defaults from evosax. The ranges over which we search the hyperparameters differs across the problems due to their different characteristics. We list the specific ranges in the following subsections. The full list of hyperparameters can be found in Table 2.

Table 2: Full Hyperparameter Overview

Task	**Dim.**	**SVGD**		**GF-SVGD**			**SV-CMA-ES**			**SV-OpenAI-ES**		
Hyperparameter		ϵ	h	ϵ	h	σ^2	m	h	σ^2	ϵ	h	σ^2
Gaussian Mixture	2	0.05	0.223	1.0	0.889	2.72	2	0.889	0.50	0.50	0.001	0.10
Double Banana	2	1.0	10^{-4}	0.001	0.011	1.116	2	0.011	0.5	0.001	10^{-4}	0.15
Motion Planning	10	0.01	0.01	0.001	0.67	2.67	2	0.01	0.10	0.05	0.01	0.10
Credit	22	0.1	10^{-3}	0.01	0.67	3.34	9	10^{-3}	0.35	0.005	10^{-3}	0.05
Covtype	55	0.01	0.45	0.05	1.0	2.28	12	0.78	0.15	0.01	0.334	0.2
Spam	58	0.01	0.11	0.05	1.0	0.56	9	0.45	0.20	0.01	0.11	0.2
Pendulum	353	–	–	0.05	16.67	0.34	2	3.33	0.47	0.10	30.0	0.05
CartPole	386	–	–	0.10	13.33	0.45	3	30.0	0.89	1.0	30.0	1.0
MountainCar	337	–	–	0.05	23.33	0.56	2	30.0	0.68	1.0	30.0	0.68
Halfcheetah	678	–	–	0.05	6.67	0.01	5	16.67	0.68	0.05	30.0	0.05
Hopper	515	–	–	0.10	16.67	0.01	5	3.33	0.05	0.10	30.0	0.26
Walker	662	–	–	0.10	10.0	0.01	8	10.0	0.79	0.05	30.0	0.16

SVGD: ϵ is the Adam learning rate, h is the kernel bandwidth.
GF-SVGD: ϵ is the Adam learning rate, h is the bandwidth, σ^2 is the scale of the prior covariance.
SV-CMA-ES: m is the number of elites, σ^2 is the init. step-size, h is the bandwidth.
SV-OpenAI-ES: ϵ is the Adam learning rate, σ^2 is the step-size, h is the bandwidth.

B.2 SAMPLING FROM SYNTHETIC DENSITIES

Gaussian Mixture We construct a Gaussian Mixture by uniformly sampling N modes $\boldsymbol{\mu}_i$ over the fixed interval of $[-6, 6]^2$. Furthermore, we sample the associated weights $w_i \in \mathbb{R}$ uniformly over $[0, 10]$. The resulting density is given by:

$$p(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^{N} w_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \mathbf{I}), \qquad K = \textstyle\sum_{i=1}^{N} w_i.$$

In our setting, we set $N = 4$. We note that in our experiments one of the sampled weights is close to zero, which is why Fig. 2 depicts only three modes.

Double Banana We use the double banana density that was introduced by Detommaso et al. [2018]:

$$p(\mathbf{x}) \propto \exp\Big(-\frac{\|x\|_2^2}{2\sigma_1} - \frac{(y - F(\mathbf{x}))^2}{2\sigma_2}\Big),$$
$$\text{where } \mathbf{x} = [x_1, x_2] \in \mathbb{R}^2, \qquad F(\mathbf{x}) = \log((1 - x_1)^2 + 100(x_2 - x_1^2)^2).$$

We choose the same hyperparameters as Wang et al. [2019a], which are $y = \log(30), \sigma_1 = 1.0, \sigma_2 = 0.09$.

Motion Planning The motion planning problem that we sample from was introduced by Barcelos et al. [2024]. The goal is to sample from the density over N waypoints that induce a trajectory from a predefined start point t_0 to a target t_T. The resulting density defines a cost that interpolates obstacle avoidance and smoothness penalties:

$$p(\mathbf{x}) \propto \exp\Big(\sum_{t=1}^{W} p_{\text{collision}}(\mathbf{x}_t) + \alpha\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2\Big),$$
$$\text{where } p_{\text{collision}}(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \sigma^2\mathbf{I}).$$

The distribution $p_{\text{collision}}$ induces a randomized 2D terrain for navigation, with the valleys between modes having lower cost. We sample a total number of 15 obstacles $\boldsymbol{\mu}_i \in \mathbb{R}^2$ from a Halton sequence following the procedure of Barcelos et al. [2024]. We choose the hyperparameters $\sigma = 0.25$ and choose $N = 5$. This results in an optimization problem over $\mathbb{R}^{10}$, as each waypoint is two-dimensional.

Ground Truth Samples For each task, we sample 256 ground truth samples to compute the MMD. For the Gaussian Mixtures, samples can be drawn by sampling from the closed form Gaussian mixture. For the other two problems, ground truth samples are drawn with the help of standard Metropolis-Hastings MCMC. For this, we sample from one independent chain per sample, using 100,000 burn-in steps.

Hyperparameter Tuning We carefully tune the hyperparameters of each method by carrying out a grid search over the following hyperparameter ranges. For the Adam learning rate, we tune over the values proposed by Wang et al. [2019a]: $[0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0]$. For the kernel parameter h, we use 10 equidistant values from the intervals $[0.001, 1]$, $[0.0001, 0.1]$, $[0.001, 3.0]$ for the Gaussian mixture, double banana and motion planning tasks respectively. For the elite ratio hyperparameter in SV-CMA-ES, we use 10 equidistant values from the interval $[0.15, 0.5]$, and for the initial value of σ that is used in SV-CMA-ES and SV-OpenAI-ES, we use 10 equidistant values from the interval $[0.05, 0.5]$. For GF-SVGD σ refers to the prior covariance scaling which we search over 10 equidistant values from the intervals $[0.1, 6.0]$, $[0.01, 2.0]$, and $[0.01, 4.0]$ for the Gaussian mixture, double banana and motion planning tasks respectively.

Table 3: Sampling Task Hyperparameters

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Number of iterations	1000	Total Pop. Size	400
Number of seeds	10	Number of Subpop.	100

B.3 LOGISTIC REGRESSION

Datasets We follow the implementation of Langosco et al. [2021], which uses a hierarchical prior $p(\theta)$ on the parameters $\theta = [\alpha, \beta]$, where $\beta \sim \mathcal{N}(0, \alpha^{-1})$ and $\alpha \sim \Gamma(a_0, b_0)$. Following the aforementioned work, we use $a_0 = 1$ and $b_0 = 0.01$. Given data D, the task is to approximate samples from the posterior

$$p(\theta \mid D) = p(D \mid \theta)p(\theta)$$

with

$$p(D \mid \theta) = \prod_{i=1}^{N}\Big[y_i \tfrac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)} + (1 - y_i)\tfrac{\exp(-x_i^T\beta)}{1+\exp(-x_i^T\beta)}\Big].$$

We use three datasets from the UCI Machine Learning repository [Kelly et al., 2023] in our experiments: Covtype [Blackard, 1998], Spambase [Hopkins et al., 1999], and Credit score data [LMU, 2010]. For the Covtype dataset, we create a binary version following Liu and Wang [2016], which groups the samples into the two groups: *Lodgepole Pine* versus *rest*. For each task, we partition the data into 70% for training, 10% for validation and 20% for testing and use a batch size of 128. The dimensionalities of the problems are defined by the number of features in the respective datasets, with an additional degree of freedom for the parameter α of the prior.

Hyperparameter Tuning We tune the hyperparameters of each method by carrying out a grid search over the following hyperparameter ranges on the validation set. For the Adam learning rate, elite ratio, and initial σ values for ES, we tune over the same parameter ranges as for the toy densities. For the kernel parameter h, we use 10 equidistant values from the interval $[0.001, 1.0]$ and for the prior σ in GF-SVGD we use the interval $[0.01, 5.0]$.

Table 4: Logistic Regression Task Hyperparameters

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Number of iterations	1000	Total Pop. Size	256
Number of seeds	10	Number of Subpop.	8
Batch size	128	–	–

B.4 REINFORCEMENT LEARNING

Setup The goal of each RL task is to maximize the expected return. The corresponding objective is to sample from the following posterior:

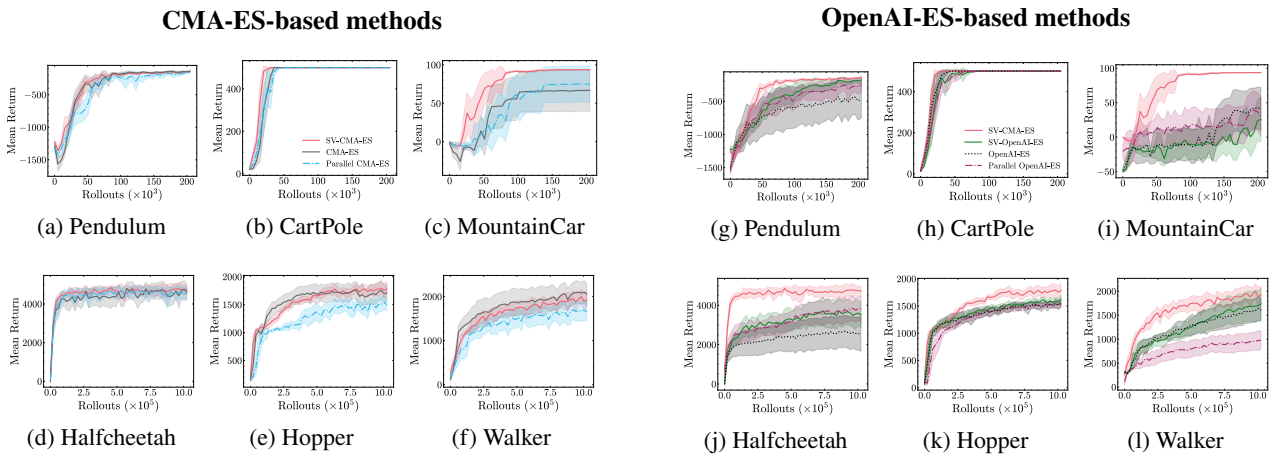
$$p(\theta) \propto \exp(J(\theta)), \quad J(\theta) = \mathop{\mathbb{E}}_{(s_t, a_t) \sim \pi_\theta} \Big[\sum_{t=1}^{T} r(s_t, a_t) \Big].$$

Here, we estimate the expected return via MC approximation across 16 independent policy rollouts for each iteration at which we evaluate the current particle set performance. In our experiments, we report the best expected return across all particles. Every such experiment is repeated 10 times, as with all previous tasks. We use the standard maximum episode length for all classic tasks as specified in the gym environment implementation, except for MountainCar, where we use a maximum episode length of 500 (instead of the default 1000 steps) to make the task more difficult. For the brax control tasks, we use a maximum episode length of 1000 steps. For the classic tasks Pendulum, CartPole and MountainCar, we use the `gymnax` implementations [Lange, 2022], and for the remaining environments, we use the `brax-v1` implementations [Freeman et al., 2021]. We use the continuous version of MountainCar. Please note that we use the brax legacy dynamics in our experiments.

Hyperparameter Tuning We tune the hyperparameters of each method by carrying out a grid search over the following hyperparameter ranges. For the Adam learning rate, elite ratio, and initial σ values, we tune over the same parameter ranges as for the toy densities. For the kernel parameter h, we use 10 equidistant values from the interval $[0.001, 30.0]$ and for the prior σ in GF-SVGD we use the interval $[0.01, 1.0]$. For brax environments, we tune the hyperparameters for 500 generations and report results for running the selected configurations over 10 seeds for 1000 generations. Please note that while we use 5 seeds for hyperparameter tuning, all evaluation runs and plots in this paper use 10 runs as indicated.

Table 5: RL Task Hyperparameters

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Number of iterations	200 (classic) / 1000 (brax)	Total Pop. Size	64
Number of seeds	10	Number of Subpop.	4
MLP Layers	2	MC Evaluations	16
Hidden Units	16	Hidden Activation	ReLU
–	–	Output Activation	Tanh



(a) Pendulum (b) CartPole (c) MountainCar (g) Pendulum (h) CartPole (i) MountainCar

(d) Halfcheetah (e) Hopper (f) Walker (j) Halfcheetah (k) Hopper (l) Walker

Figure 7: Comparison of CMA-ES-based methods (left) and OpenAI-ES-based methods (right). In all experiments, a total population size of 64 is used, split across 4 subpopulations in parallel methods. We report the mean (± 1.96 standard error) across 10 independent runs.

C ADDITIONAL RESULTS

C.1 ALGORITHM ABLATIONS

Setting To evaluate the potential of SV-CMA-ES for pure blackbox optimization, we compare it to other ES-based methods that perform pure maximum-likelihood estimation. The two natural baselines to compare against are first simple CMA-ES, but with a population size of n = Num. particles $\times$ Subpopsize, which ablates the effect of sampling multiple subpopulations in parallel. Second, we compare against multiple independent parallel runs of CMA-ES, which ablates the coordination of the runs via the SVGD update.

Results In Fig. 7, we display the results of our evaluation. When comparing SV-CMA-ES against single population CMA-ES we observe similar performances on most problems. The most notable difference that stands out is the better performance of SV-CMA-ES on the MountainCar problem. What happens in this experiment? The MountainCar problem is an MDP in which a car must be accelerated to reach a goal on a hill. Since there are negative rewards for large accelerations, there is a large local optimum at which policies are idle and do not accelerate at all to avoid negative rewards. However, once the goal is reached, the agent receives a reward of 100. We speculate due to these results that SV-CMA-ES may be superior to other CMA-ES-based methods on such sparse reward environments. We will further investigate this hypothesis in Appendix C.2.

Furthermore, we compare the performance to multiple parallel CMA-ES runs. This experiment essentially ablates the kernel term in our update. We find that in comparison to multiple parallel CMA-ES runs, our method achieves superior performance on half of the problems, and equal on the rest. These results are encouraging, as they suggest that our SV-ES is a viable alternative to independent parallelizations of a given ES algorithm.

Additionally, we include Fig. 8 to display the per-seed results on the MountainCar problem. The results highlight that GF-SVGD converges to the local optimum of being idle on two out of the ten runs.

C.2 ENVIRONMENT ABLATIONS

Setting To investigate the hypothesis that SV-CMA-ES is superior to simple parallel CMA-ES runs or single large-population CMA-ES in sparse reward environments, we construct sparse reward versions of the Walker and Hopper environments following the procedure of Mazoure et al. [2019]. The goal of the environment ablations is to make the optimization landscape more similar to that of MountainCar. To achieve this goal, we increase the control cost in the reward and remove the positive rewards for the agent being healthy. Additionally, the original forward reward is replaced by a positive reward for being upright that is only awarded once the agent has moved beyond a certain position in space. Essentially, the resulting environments only reward successful locomotion policies once the behavior is learned, instead of rewarding intermediate learning outcomes as

well. In this setting, the reward is negative for policies that move the agent not far enough and close to zero for policies that do not move the agent at all, similar to the MountainCar reward. The resulting reward function is:

$$r_{\text{ablated}}(s_t, a_t) = \alpha \mathbb{1}(pos(s_t) \geq 1) - \beta \|a_t\|_2^2,$$

where $pos(s_t) \in \mathbb{R}$ is the agent's position at state s_t in the task space. In our experiments, we choose $\alpha = 2.0$ and $\beta = 0.1$. Since the resulting ablated Walker problem is significantly more difficult, we run experiments on this benchmark for 1500 instead of 1000 iterations on this problem.

Results Our results demonstrate that SV-CMA-ES outperforms other CMA-ES-based methods considerably in the constructed scenarios. Both problems are more difficult to solve than the standard problems in Fig. 7 because a positive reward is only received once the agent moves far enough. As hypothesized, we observe a similar pattern as for MountainCar. In other words, SV-CMA-ES outperforms the other methods on sparse reward environments. These results suggest two things. First, we find that the coordination of multiple CMA-ES runs via the SVGD step improves the algorithmic performance on problems for which exploration is crucial. Second, we observe that this effect only comes to fruition if the population updates are coordinated via SVGD, as SV-CMA-ES outperforms parallel CMA-ES runs considerably. The combination of these findings suggests that SV-CMA-ES constitutes a strong alternative to other CMA-ES-based optimizers if problems require exploration. We thus believe that analysis of the scalability of the algorithm to a high number of particles beyond our computational capabilities (i.e., several thousands) would be an intriguing direction of future research.

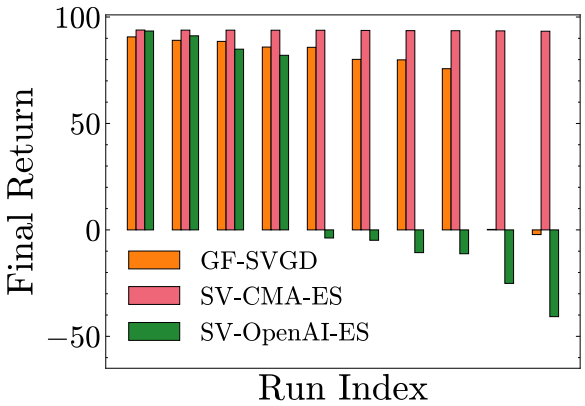
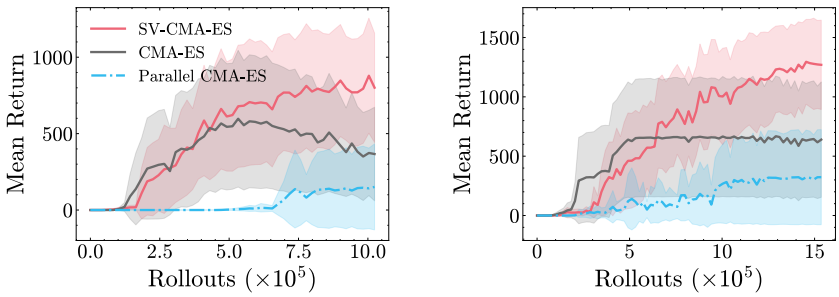


Figure 8: Per-seed performances on the MountainCar task. SV-CMA-ES is the only method to achieve optimal performance across all seeds. GF-SVGD converges to idle agents (i.e., a reward of zero) on two out of the ten seeds.



(a) Hopper-sparse (b) Walker-sparse

Figure 9: Comparison of CMA-ES-based methods on sparse reward environments. Coordinating parallel ES runs via the SVGD update clearly improves the performance. In all experiments, a total population size of 64 is used, split across 4 subpopulations in parallel methods. We report the mean (±1.96 standard error) across 10 independent runs.

C.3 EMPIRICAL RUNTIME ANALYSIS

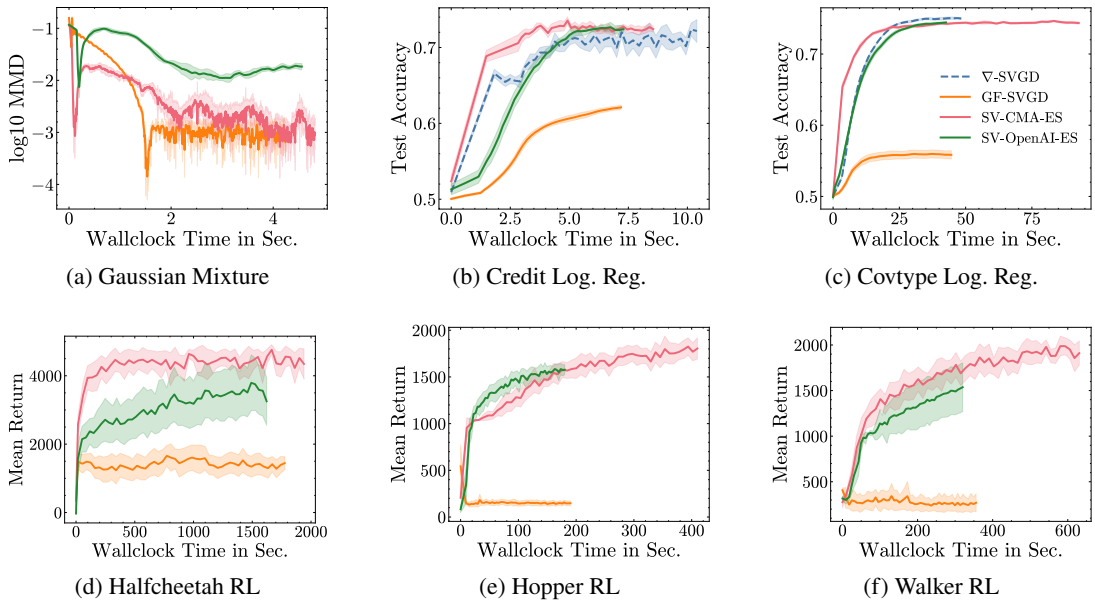
Setting Given the higher theoretical runtime complexity of SV-CMA-ES compared to the baselines, we perform an empirical runtime analysis. The goal of this investigation is to evaluate the significance of the theoretical algorithm properties in practice. To this end, we select multiple problems from the main paper to illustrate the performances over time. We select the synthetic Gaussian mixture as a low dimensional and fast-to-evaluate objective, a logistic regression tasks as moderately dimensional tasks with fast-to-evaluate objectives, and the brax RL tasks as the most high-dimensional task of the paper which require expensive simulator calls for evaluations of the ground truth density. For each problem, we run 1000 iterations, just as in prior experiments. For the RL tasks, we use the GPU accelerated brax simulator. To provide a fair analysis of

wall-clock times, we disable deterministic operations on the GPU, which may lead to slight differences in performances, w.r.t. to the results reported in the main paper, but is more realistic in terms of runtimes. We would like to point out, however, that we did not observe any visible differences in the results. The experiment is carried out on a single NVIDIA A40 GPU and reported times are for all 10 independent runs being run in parallel.

Results The results of the runtime analysis are depicted in Fig. 10. Overall, we observe competitive convergence behavior of SV-CMA-ES in terms of wall-clock time. While the overall runtime time required to run 1000 iterations is highest for SV-CMA-ES, we do observe that our method obtains good samples efficiently under this metric.

Generally, we observe that the influence of the update computation costs on the overall runtime varies greatly between problems. While the differences between all methods are relatively small on the synthetic 2d problems, we observe that SV-CMA-ES is slower on high dimensional problems such as the Covtype logistic regression task. This reflects that the computational cost of SV-CMA-ES increases with the number of problem dimensions, while the cost of the other baselines is dominated by the number of particles. However, we also observe that the quality of the updates of SV-CMA-ES is still generally better in these cases, as it samples well performing solutions the fastest.

More importantly, we observe that the main factor that drives overall runtime is the evaluation of the objective function. Furthermore, we observe smaller runtime differences between the methods on the Halfcheetah task despite it being the most high dimensional task that we evaluate on. This underlines that the significance of theoretical algorithm complexity depends heavily on the problem and the costs that are associated with target density evaluations. Since our method requires fewer iterations than the baselines to reach comparable performance (as shown previously), we find that this demonstrates the practical relevance of the presented approach.



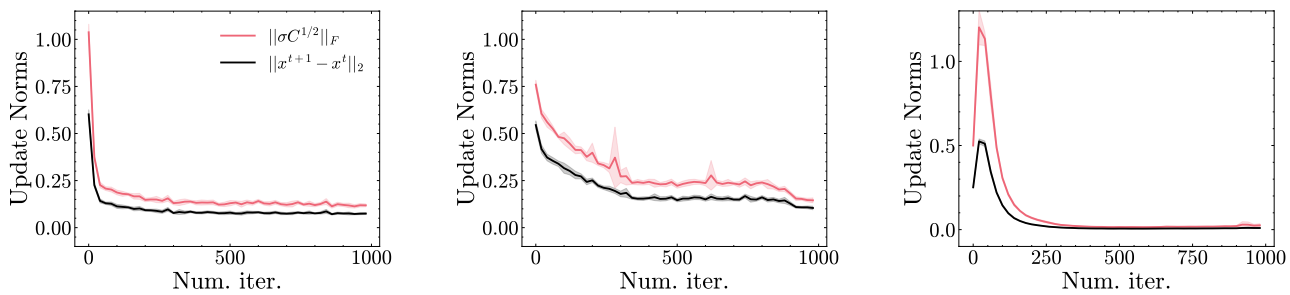
(a) Gaussian Mixture (b) Credit Log. Reg. (c) Covtype Log. Reg.

(d) Halfcheetah RL (e) Hopper RL (f) Walker RL

Figure 10: Performance vs. wallclock time. We run all methods for 1000 iterations and display the elapsed wallclock time. The plot shows that SV-CMA-ES also performs well w.r.t. this metric. All results are averaged across 10 independent runs (± 1.96 standard error).

C.4 EMPIRICAL CONVERGENCE ANALYSIS

To evaluate the convergence of our method, we expand our analysis beyond the pure MMD computations in Fig. 3 and analyze the length of the particle steps. To this end, we use the same setting as the one in Fig. 3 to plot the Frobenius norm of the particle preconditioning matrix $\sigma C^{1/2}$ across iterations in Fig. 11. In the same figure, we further plot the length of the steps that were actually taken by the particles, i.e., the size of the update in Eq. (15). The results show that the algorithm exhibits stable convergence. A key finding in this context is that for each problem, there exists a stationary point at which the CMA-ES steps that are sampled from $\mathcal{N}(\mathbf{x}, \sigma^2 C)$ are counteracted by the kernel gradients, as the total step length is



(a) Gaussian Mixture (b) Double Banana (c) Motion Planning

Figure 11: Mean step lengths per iteration. We record the step length every 20 steps across 1 000 iterations. The SV-CMA-ES steps and estimated gradient steps by CMA-ES converge to a stable equilibrium. We report the mean (±1.96 standard error) across all 100 particles and 10 independent runs.

shorter than the matrix norm. This shows that our method is capable of finding the point of convergence of SVGD where the kernel gradient and objective gradient balance each other out. We see the fact that the resulting total step length does not reach zero across all problems as an artifact of the stochastic gradient estimates that CMA-ES provides. We do not see this as a problem, however, as the MMD analysis in Fig. 3 shows that the particles only move within areas of high density. Since the approximation of the target density in SVGD is based on a sum of delta function which approximates the target as $q(x) = \sum_{\mathbf{x}_i \in X} \delta(\mathbf{x} - \mathbf{x}_i)/\varrho$ it permits small particle movements at the equilibrium between repulsive and driving force in the update.

C.5 CONVERGENCE PLOTS

For completeness, we depict the convergence results on the synthetic sampling tasks. For each task and method combination, we display the evolution of the sample set across the optimization. To this end, we split the total number of iterations into 10 equally-sized bins and generate a plot for each. In other words, we plot the sample set every 100 iterations. The results for the Gaussian mixture are listed in Fig. 12, those for the Double Banana density in Fig. 13, and Fig. 14 shows the convergence on the trajectory optimization task. All plots confirm the quantitative results, i.e., SV-CMA-ES converges quickly compared to the baselines. We note that we refer to convergence once all particles are in high density areas. Due to the stochastic nature of the update, there is no guarantee that there is an actual stationary distribution of the finite optimization process. For storage reasons, we include these images as raster images. We encourage readers to reach out to us in case they are interested in the vector graphics version for these last three plots.

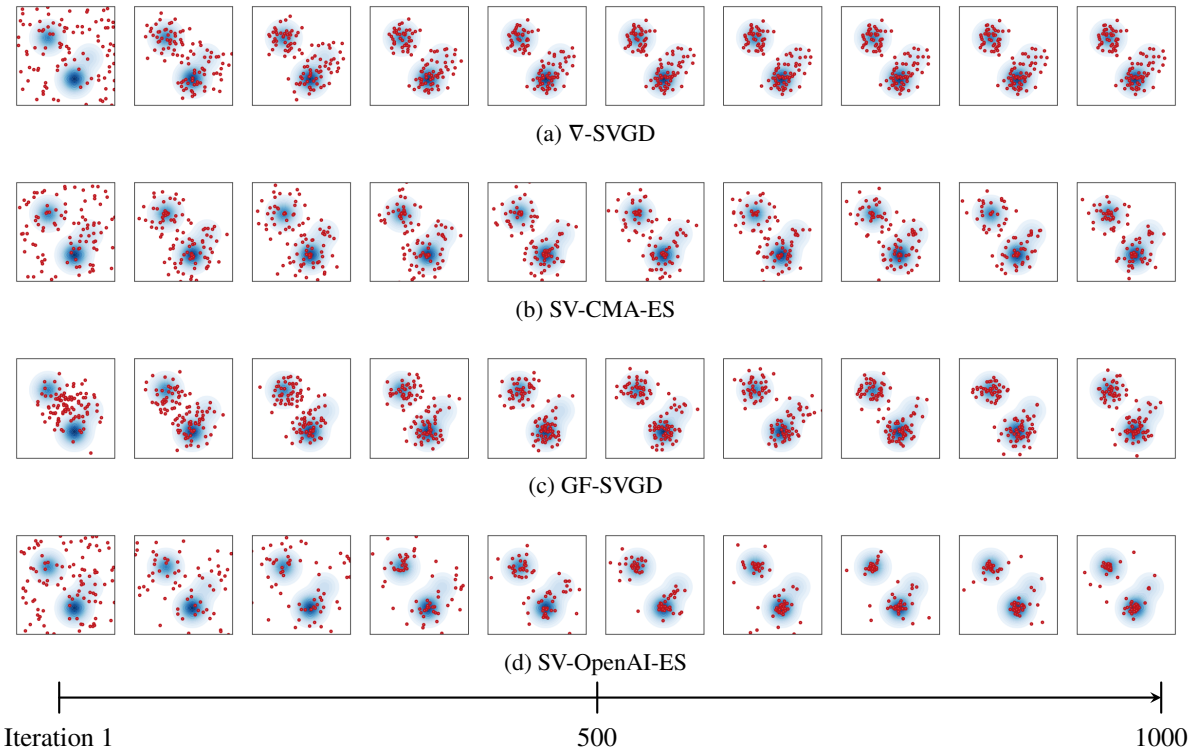


Figure 12: Convergences on GMM sampling task. For each method, the sample convergence across the full 1000 sampling iterations is displayed.

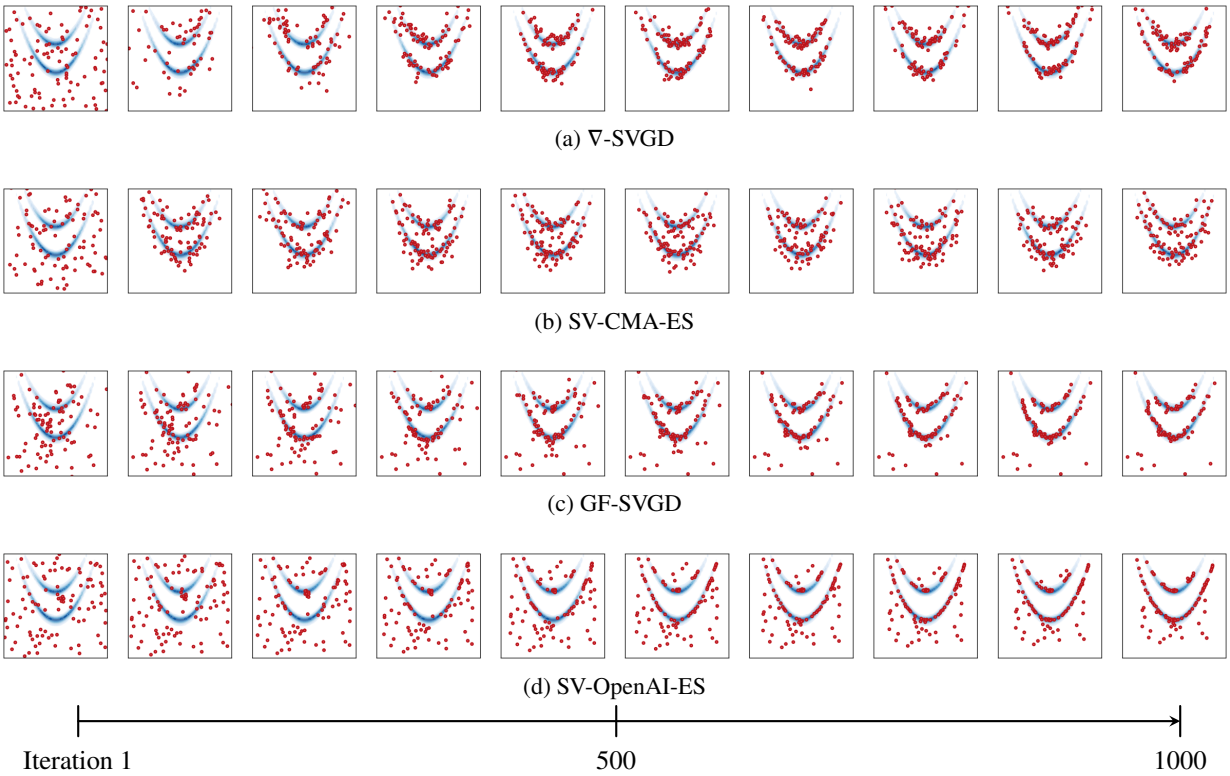


Figure 13: Convergences on Double banana sampling task. For each method, the sample convergence across the full 1000 sampling iterations is displayed.

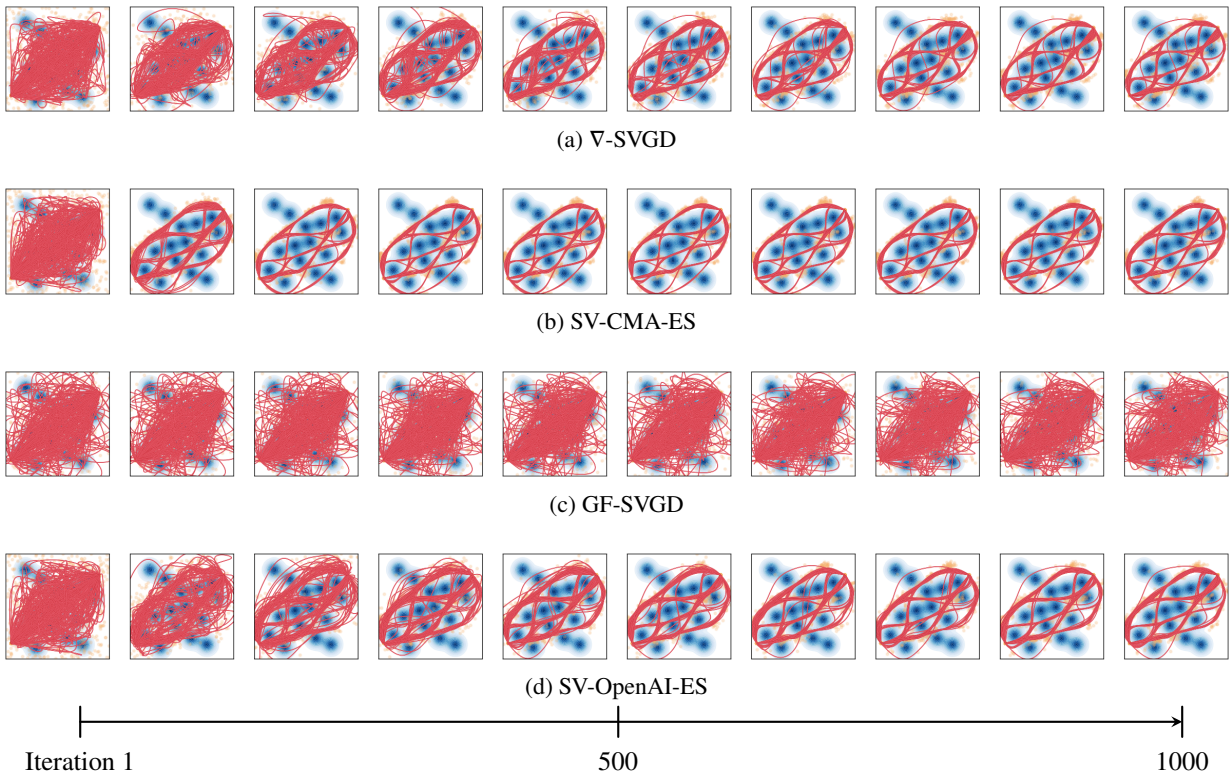


Figure 14: Convergences on motion planning sampling task. For each method, the sample convergence across the full 1000 sampling iterations is displayed.