Inference-time Scaling of Diffusion Models through Classical Search

Anonymous Author(s)
Affiliation
Address
email

Abstract

Classical search algorithms have long underpinned modern artificial intelligence. In this work, we tackle the challenge of inference-time control in diffusion models—adapting generated outputs to meet diverse test-time objectives—using principles from classical search. We propose a general framework that orchestrates local and global search to efficiently navigate the generative space. It performs compute-efficient global exploration using breadth-first and depth-first tree search and employs a theoretically grounded scalable local search via annealed Langevin MCMC. We evaluate our approach on a range of challenging domains, including planning, offline reinforcement learning, and image generation. Across all tasks, we observe significant gains in both performance and efficiency over baseline methods. These results demonstrate that classical search offers a principled, practical foundation for inference-time scaling in diffusion models, and that our method, which jointly scales local and global search, establishes a new Pareto frontier.

1 Introduction

Classical search algorithms have laid the foundation for modern artificial intelligence [59]. In discrete settings, graph search algorithms are widely used to explore the state space. Breadth-first search (BFS) [50] and depth-first search (DFS) [73] traverse the search tree in a fixed order. To better leverage problem-specific information, best-first search methods [56], such as A* [22], use a heuristic to evaluate and prioritize states. Alternatively, local search methods, such as hill-climbing [59, Sec. 4.1], explore neighboring states. More recent techniques like gradient descent and Markov Chain Monte Carlo (MCMC) have become widely adopted in optimization and probabilistic inference, underpinning many modern AI models.

Diffusion models [26] have shown impressive performance in generative modeling for continuous domains such as images [10], videos [28], and world modeling [92]. They are also increasingly used in robotics and decision-making [44, 4, 74] to generate diverse actions [8]. However, generated samples may not always align with physical laws [68] or human intent [79], and the vast generative space often necessitates multiple trials to produce satisfactory outputs [87]. To address this, we scale up *inference-time compute* using strategic search methods that navigate the generative manifold for high-quality samples. We formalize sample evaluation using a verifier function $f(\boldsymbol{x}_0)$ defined on *ground truth* samples, which measures the quality of the sample. Such verifiers could be reward functions [89], Q-functions [45], classifier conditions $p(c|\boldsymbol{x}_0)$ [94, 10], and multi-modal LLMs [29].

To efficiently search the generative space of diffusion models, we revisit classical search principles. To capture diverse modes in the complex distributions generated by diffusion models, we view sampling as traversing a search tree, employing BFS and DFS to progressively explore states during denoising. Similar to best-first search, we evaluate intermediate states $\boldsymbol{x}_t$ with a verifier $f(\boldsymbol{x}_{0|t})$, prioritizing high-quality paths. To go beyond the base model and obtain higher-quality samples, we

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

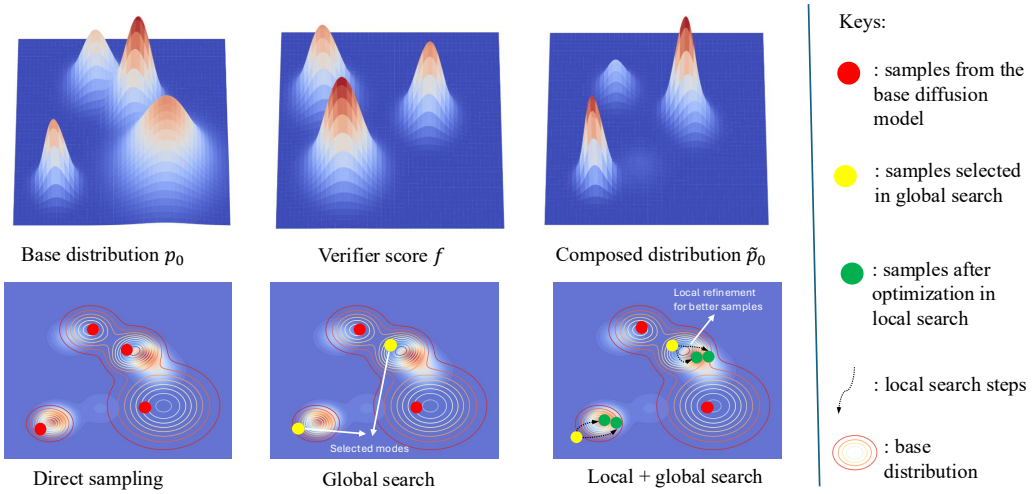


Figure 1: **Illustration of our search framework**. **Bottom left**: direct sampling results in samples with low verifier scores. **Bottom middle**: global search identifies high score modes within the base distribution. **Bottom right**: local search further optimizes the samples for higher quality, driven by the gradient signal.

perform local search via Langevin MCMC, exploring the neighborhood of current samples under guidance from both the verifier gradient and the diffusion model's "score function" [78]. By jointly optimizing the compositional objective of the diffusion model and the verifier [12], our local search surpasses the capabilities of the base model. An overview of our framework is shown in Fig. 1.

Recent works scale diffusion model inference via particle-based SMC [33, 63, 84] and tree-based methods [19], typically as BFS with fixed schedules. We generalize these with a BFS-based framework, clarifying prior design choices and establishing a strong baseline. Inspired by DFS, we add adaptive backtracking to allocate compute adaptively, surpassing BFS baselines. While global search remains limited to base-distribution modes, scaling local search with Langevin MCMC explores high-reward regions beyond the model, proving effective in challenging decision-making tasks.

Our key contributions are summarized as follows:

- **i)** For global tree search, we elucidate the design space of prior BFS-style methods and provide an improved BFS baseline. We further present the first adaptive DFS algorithm for diffusion inference scaling, offering superior efficiency and adaptivity.
- **ii)** We introduce a theoretically grounded local search method using annealed Langevin MCMC, demonstrating superior performance in challenging domains.
- **iii)** We propose a unified framework for efficient inference-time search in diffusion models grounded in classical search principles. By jointly scaling local and global search for the first time, we advance the Pareto frontier of inference-time scaling across diverse domains.

2 Related Works

Here, we provide a brief overview of inference-time scaling with diffusion models. For a more comprehensive literature review and discussion of concurrent works, see Appendix B.

Recent works such as [33, 63] propose SMC-based particle filtering methods, scaling inference compute by increasing the number of particles. Tree-search-based methods [39, 19, 48] evaluate intermediate nodes and expand promising candidates, scaling inference compute by increasing the width of the search tree. Both approaches can be seen as special cases of our BFS framework. Alternatively, Du et al. [13] propose iterative reasoning via Langevin MCMC, scaling inference by increasing the number of refinement steps. To utilize the verifier gradient, classifier guidance [10] trains a noise-dependent classifier for gradient guidance, and training-free guidance methods [94, 9, 68, 95, 24] improve sample quality using an additional pretrained classifier. In TFG [94] the authors have observed that more recurrent steps can yield better performance on challenging tasks.

However, their theoretical foundations remain poorly understood, and their scaling behaviors are largely unexplored.

3 Backgrounds

3.1 Diffusion Probabilistic Models

Suppose we have D-dimensional random variable $\boldsymbol{x}_0 \in \mathbb{R}^D$ with distribution $p_0(\boldsymbol{x}_0)$. Diffusion models [26, 67] and the more general flow models [42, 2] are generative models that turn noise into data via a stochastic process $\{\boldsymbol{x}_t\}_{t=0}^T$. The forward "noising" process with $t > s$ can be defined as:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_s) = \mathcal{N}\left(\boldsymbol{x}; \frac{\alpha_t}{\alpha_s}\boldsymbol{x}_s, \alpha_t^2\left(\frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_s^2}{\alpha_s^2}\right)\boldsymbol{I}\right) . \tag{1}$$

where α_t, σ_t are referred as the noise schedule with $\alpha_0 = \sigma_T = 1$, $\alpha_T = \sigma_0 = 0$. We can thus write the random variables $\boldsymbol{x}_t$ as an interpolation between data and noise [47]:

$$\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon} ,$$

and denote $q_t(\boldsymbol{x}_t)$ as the marginal distribution of $\boldsymbol{x}_t$. To model the reverse "denoising" process, we train the model using the denoising objective [26]:

$$L(\theta) = \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{\epsilon}}\left[\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}\right] ,$$

which is equivalent of learning the score function of $q_t(\boldsymbol{x}_t)$ [78], as the ground truth of $\epsilon_\theta(\boldsymbol{x}_t, t)$ is $-\sigma_t \nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t)$. To generate samples, we transform noise into data via the reverse transition kernel $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. In practice, we either sample $\boldsymbol{x}_{t-1}$ using deterministic samplers like DDIM [67]:

$$\boldsymbol{x}_{t-1} = \frac{\alpha_{t-1}}{\alpha_t}(\boldsymbol{x}_t - \sigma_t \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)) + \sigma_{t-1}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$$

or stochastic samplers like DDPM [26, 54]:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)\right) .$$

3.2 Compositional and Controllable Generation of DPMs

Given a base diffusion model with data distribution $p_0(\boldsymbol{x}_0)$, one may wish to sample $\boldsymbol{x}_0$ with some constraints or conditions $f(\boldsymbol{x}_0)$. Exact diffusion sampling from the composed distribution $\tilde{p}_0(\boldsymbol{x}_0) \propto p_0(\boldsymbol{x}_0) f(\boldsymbol{x}_0)$ would require training a time-dependent f on data generated by p_0 [10, 45], which may not be applicable in practice. Thus, we adopt optimization based methods to approximate the target distribution.

Compositional generation via annealed Langevin MCMC. When sampling from a compositional distribution composed of multiple probability distributions, $\tilde{p}_0(\boldsymbol{x}_0) \propto p_0(\boldsymbol{x}_0)\hat{p}_0(\boldsymbol{x}_0)$, [12] proposes annealed Langevin MCMC sampling. In this approach, a sequence of annealed distributions $\tilde{q}_t(\boldsymbol{x}_t) \propto q_t(\boldsymbol{x}_t)\hat{q}_t(\boldsymbol{x}_0)$ is constructed, and samples are drawn using Langevin dynamics [82]:

$$\boldsymbol{x}_t^{i+1} = \boldsymbol{x}_t^i + \eta \nabla_{\boldsymbol{x}} \log \tilde{q}_t(\boldsymbol{x}_t^i) + \sqrt{2\eta}\boldsymbol{\epsilon}^i , \quad \boldsymbol{\epsilon}^i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) . \tag{2}$$

Since the distribution of $\boldsymbol{x}_t^i$ converges to $\tilde{q}_t(\boldsymbol{x}_t)$ asymptotically as $i \to \infty$, $\eta \to 0$, we can sample from $\tilde{p}_0(\boldsymbol{x}_0)$ following the annealing path $\{\tilde{q}_t\}_{t=0}^T$ with $\tilde{q}_0 = \tilde{p}_0$. Moreover, since the score of $\tilde{q}_t$ can be directly computed by composing the score of two distributions:

$$\nabla_{\boldsymbol{x}} \log \tilde{q}_t(\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log \hat{q}_t(\boldsymbol{x}_t) ,$$

thus do not require extra training.

Controllable generation through training-free guidance. During the sampling process, training-free guidance propose to update $\boldsymbol{x}_t$ using gradient ascent

$$\tilde{\boldsymbol{x}}_t = \boldsymbol{x}_t + \boldsymbol{\Delta}_t , \quad \boldsymbol{\Delta}_t = \rho_t \nabla_{\boldsymbol{x}_t} \log f(\boldsymbol{x}_{0|t}) + \mu_t \alpha_t \nabla_{\boldsymbol{x}_{0|t}} \log f(\boldsymbol{x}_{0|t}) . \tag{3}$$

where $\boldsymbol{x}_{0|t} = \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] = \frac{\boldsymbol{x}_t - \sigma_t \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)}{\alpha_t}$. This method approximates the intractable posterior with the posterior mean: $\mathbb{E}_{\boldsymbol{x}_0|\boldsymbol{x}_t}[f(\boldsymbol{x}_0)] \approx f(\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t])$. To enhance the guidance strength, [95] propose to apply a recurrence strategy, which first samples $\boldsymbol{x}_{t-1}$ via $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, add the guidance gradient, then add noise back to $\boldsymbol{x}_t$ through the forward process $q_t(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$:

$$\boldsymbol{x}_{t-1}^i \sim p_\theta(\cdot|\boldsymbol{x}_t^i) , \quad \tilde{\boldsymbol{x}}_{t-1}^i = \boldsymbol{x}_{t-1}^i + \frac{\alpha_{t-1}}{\alpha_t}\boldsymbol{\Delta}_t , \quad \boldsymbol{x}_t^{i+1} \sim q_t(\cdot|\tilde{\boldsymbol{x}}_{t-1}^i) , \quad i = 1, 2, \cdots, N_{\text{recur}} , \tag{4}$$

with N_{recur} being the total number of recurrence steps.

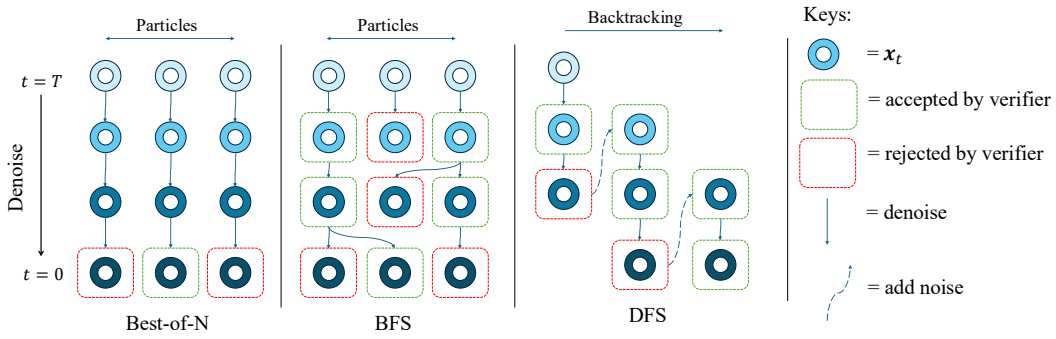


Figure 2: **Illustration of global tree search algorithms.**

4 Methods

Problem Formulation. Given a pretrained diffusion model $\epsilon_\theta(\boldsymbol{x}_t, t)$ with a base distribution $p_0(\boldsymbol{x}_0)$, at test-time, we often wish to optimize the generation process to satisfy task-specific objectives. For example, RL may require generating high-value actions, image synthesis may seek constraint-satisfying images, and trajectory generation may demand physically valid outputs. In this paper, we are interested in how to scale test-time inference to follow such objectives.

We consider an inference-time scaling strategy that adjusts the sampling process based on a verifier function. Specifically, we define a verifier $f(\boldsymbol{x}_0) : \mathbb{R}^D \to \mathbb{R}^+$ which specifies the degree to which samples optimize a specified objective. We then aim to bias sampling toward regions of the sample space where $f(\boldsymbol{x}_0)$ is high. This leads to the objective of sampling from a compositional distribution that combines the original model distribution with the verifier:

$$\tilde{p}_0(\boldsymbol{x}_0) \propto p_0(\boldsymbol{x}_0) f(\boldsymbol{x}_0)^\lambda \,, \tag{5}$$

where λ controls the weight of verifier scores.

Since exact sampling from the distribution is often impractical, we aim to search the manifold for the target samples at *inference time*, both *globally* and *locally*. First, we explore the diverse modes in the complex generative landscape of diffusion models through global graph search algorithms. However, global search alone can not generate samples beyond the pretrained model. We then propose to search the vicinity of the sample using hill-climbing style local search methods, guided by the verifier gradient.

4.1 Global Search for Mode Identification

To efficiently explore the modes of the diffusion model, we represent the Markov chain of the denoising process as a *fixed-depth tree*, where the transition kernel $\tilde{p}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ may correspond to either deterministic or stochastic samplers. This abstraction allows the application of classical tree search heuristics to design compute-efficient exploration methods. By expanding nodes with higher score estimates and backtracking from low-quality nodes in the tree, we can efficiently navigate the generative space and sample from high-quality modes. An illustration is provided in Fig. 2.

4.1.1 Unified BFS-style linear search

Inspired by breadth-first search (BFS), which expands nodes level by level, we denoise a set of particles in parallel at each noise level. The simplest approach is best-of-N sampling: generate N candidate trajectories and select the one with the highest verifier score at the final step. While straightforward, this strategy ignores information from intermediate stages.

To improve efficiency, and following the idea of best-first search [56], we score each intermediate particle $\left\{\boldsymbol{x}_t^k\right\}_{k=1}^N$ using estimates of its verifier score $f(\boldsymbol{x}_{0|t}^k)$, and dynamically reallocate computation by sampling more children n_t^k for high-scoring nodes. We provide a general design space for tempering, scoring, and resampling that unifies previous tree-search-based and particle-based baselines such as SVDD [39], DAS [33], and FK-steering [64]. The pseudocode is shown in Alg. 3.

Tempering. To reduce estimation bias in early steps, DAS [33] increases weights on smaller time steps so that $\tau_T < \tau_{T-1} \cdots < \tau_0$, re-weighting scores with $\tau_t f(\boldsymbol{x}_{0|t}^k)$. SVDD [39] samples only from the top-scoring particle, i.e., $\tau_t = \infty$. We consider: **Constant** : $\tau_t = \tau$, **Increase** : $\tau_t = \left((1+\gamma)^{T-t} - 1\right)\tau$, **Inf** : $\tau_t = \infty$.

Scoring. Following [33, 64], we propose to score intermediate particles $\widehat{f}(\boldsymbol{x}_t^k)$ via: **Current** : $\tau_t f(\boldsymbol{x}_{0|t}^k)$, **Difference** : $\tau_t f(\boldsymbol{x}_{0|t}^k) - \tau_{t+1} f(\boldsymbol{x}_{0|t+1}^k)$, **Max** : $\max_{s \geq t} \tau_s f(\boldsymbol{x}_{0|s}^k)$.

Resampling. Given $\widehat{f}(\boldsymbol{x}_t^k)$, we allocate particles as $n_t^k = \text{Resample}\left(N, \text{softmax}\left(\widehat{f}(\boldsymbol{x}_t^k)\right)\right)$, where n_t^k is the number of children for $\boldsymbol{x}_t^k$. We compare the baseline **Multinomial** resampling [19, 64] and the variance-reduced **SSP** [33]; see [17] for other methods.

Prior methods are special cases of BFS: SVDD [39] = **BFS** (**Inf**, **Current**, **Multinomial**); DAS [33] = **BFS** (**Increase**, **Difference**, **SSP**); FK-steering [63] = **BFS** (**Constant**, **Max**, **Multinomial**). Ablations (Sec. 5.1) show **SSP** resampling is key for performance, and our baseline **BFS** (**Increase**, **Max**, **SSP**) consistently outperforms prior methods in efficiency.

4.1.2 DFS-style non-linear search

Depth-first search (DFS) explores one branch of the search tree as deeply as possible before backtracking. In our setting, this corresponds to iteratively denoising a single particle until its verifier score drops below a predefined threshold: $f(\boldsymbol{x}_{0|t}) \leq \delta_t$, where δ_t is a scheduled threshold for timestep t.

Once the constraint is violated, the algorithm backtracks by reintroducing noise, moving to a higher noise level $t_{\text{next}} = t + \Delta_T$ using the forward diffusion process $q(\boldsymbol{x}_{t_{\text{next}}}|\boldsymbol{x}_t)$ in Eq. 1. This allows the model to restart the denoising process from a different region of the manifold, encouraging exploration of diverse modes. Unlike the small noise injection and fixed schedule used in SoP [48] for local exploration, DFS performs global exploration with $\Delta_T \geq \frac{T}{4}$ and an adaptive exploration strategy.

A key strength of DFS is its ability to allocate compute adaptively: difficult prompts and low-quality trajectories naturally trigger more backtracking and exploration, while easier instances are solved more directly. This dynamic behavior is driven purely by the verifier signal, without needing to know the difficulty in advance as in [66]. Also, the threshold acts as a control knob for users to balance output quality and computation resources, where higher threshold automatically scales compute for better output. As shown in Sec. 5.2, this adaptive strategy leads to substantial gains in efficiency and performance over prior methods, and even our strengthened BFS baseline.

4.2 Scaling Local Search via Langevin MCMC with Verifier Gradient

Global search can efficiently discover the high score modes from the base diffusion model, but can not generate higher quality samples that exceed the pretrained model. Thus, we aim to sample from the compositional distribution $\tilde{p}_0$ in Eq. 5 for higher quality samples. To optimize the compositional objective, we conduct local-search with hill-climbing methods, aiming to find the local maximum with high $\tilde{p}_0$. Specifically, we view the sampling problem as compositional optimization in measure space [83], and follow the gradient flow of KL-divergence, performing Langevin MCMC steps (details see Appendix. C.1).

Similar to annealed Langevin MCMC in [12], we could construct a series of annealed functions $\hat{f}_t(\boldsymbol{x}_t)$ with $\hat{f}_0(\boldsymbol{x}_0) = f(\boldsymbol{x}_0)$. Then we sample from the distributions $\tilde{q}_t(\boldsymbol{x}_t) \propto q_t(\boldsymbol{x}_t)\hat{f}(\boldsymbol{x}_t)$ through Langevin MCMC in Eq. 2 (details see Appendix. C.2). Alternatively, training-free guidance in Eq. 3 utilizes the gradient of $f(\boldsymbol{x}_{0|t})$ to optimize $\boldsymbol{x}_t$, which can be computed directly using the diffusion model output. However, naive gradient updates have been observed to produce OOD and adversarial samples [61]. In [94], recurrence (Eq. 4) was found to help avoid adversarial samples in challenging guidance tasks, though its theoretical underpinnings remain poorly understood. We unify these two approaches by demonstrating that training-free guidance with recurrence, in the continuous limit, constitutes an instance of Langevin MCMC. For details see Appendix. C.3, and a rigorous convergence bound is in Theorem. 1.

Proposition 1. *In the continuous limit where the number of diffusion denoising steps $T \to \infty$, training-free guidance with recurrence is equivalent to running Langevin MCMC on a series of annealed distributions $\{\tilde{q}_t(\boldsymbol{x}_t)\}_{t=0}^{T}$, with $\tilde{q}_0(\boldsymbol{x}_0) = \tilde{p}_0(\boldsymbol{x}_0) \propto p_0(\boldsymbol{x}_0) f(\boldsymbol{x}_0)^{\lambda}$.*

Thus, the recurrence step (without guidance) can be interpreted as Langevin MCMC applied to the original distribution of the diffusion model $q_t(\boldsymbol{x}_t)$, and the guidance term $\boldsymbol{\Delta}_t$ in Eq. 3 then serves as defining a practical annealing path $\hat{f}_t(\boldsymbol{x}_t)$ that bias the sampling path towards high reward areas beyond the modes of the base model. We are the *first* to propose this theoretical unification of the two lines of work, providing insights into efficient local search of diffusion models via gradients.

We implement local search by parameterizing the reverse transition kernel $\tilde{p}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ as a sequence of Langevin MCMC steps (Eq. 2), followed by a denoising step using DDIM (Eq. 11) or DDPM (Eq. 12); see Appendix C.5 for details. Unlike classifier-guidance or naive training-free guidance, which apply only gradient guidance in the denoising step, our approach incorporates explicit Langevin MCMC steps. In Sec. 5.3, we scale the number of local search steps for the first time and observe substantial improvements over pretrained models across multiple tasks.

5 Experiments

In this section, we apply inference-time scaling with our search strategy across a range of domains. In Sec. 5.1, we present a strengthened BFS baseline that outperforms previous particle-based methods. In Sec. 5.2, we demonstrate the adaptivity and efficiency of our DFS method. In Sec. 5.3, we scale up local search in challenging decision-making domains, highlighting the importance of jointly scaling local and global search.

5.1 Elucidating the Design Space of BFS for a Strengthened Baseline

In this section, we explore the design choices of BFS and present a strengthened baseline. To ensure a fair comparison, we directly use the official implementation of FK-steering [63] with the ImageReward [89] verifier and the SD v1.5 model. For details, see Appendix E.1.

N	BoN	Multinomial	SSP
4	0.702 ± 0.057	0.743 ± 0.037	$\mathbf{0.834 \pm 0.041}$
8	0.896 ± 0.031	0.926 ± 0.042	$\mathbf{1.032 \pm 0.035}$

(a) Results for different sampling choices with **Constant** tempering and **Max** scoring

N	Current	Difference	Max
4	0.812 ± 0.037	0.823 ± 0.036	$\mathbf{0.834 \pm 0.041}$
8	0.996 ± 0.029	1.013 ± 0.032	$\mathbf{1.032 \pm 0.035}$

(b) Results for different scoring choices with **SSP** resampling and **Constant** tempering

N	Increase	Inf	Constant
4	$\mathbf{0.882 \pm 0.029}$	0.667 ± 0.076	0.834 ± 0.041
8	$\mathbf{1.087 \pm 0.031}$	0.775 ± 0.087	1.032 ± 0.035

(c) Results for different tempering choices with **SSP** resampling and **Max** scoring

Table 1: Ablation of BFS design choices

We begin with the baseline design of FK using **BFS** (**Constant**, **Max**, **Multinomial**) and evaluate different resampling strategies. As shown in Table 1a, **SSP** significantly improves performance over naive multinomial resampling, and we adopt it in our design. We then ablate the scoring methods and tempering options in Tables 1b and 1c, arriving at our improved **BFS** (**Increase**, **Max**, **SSP**).

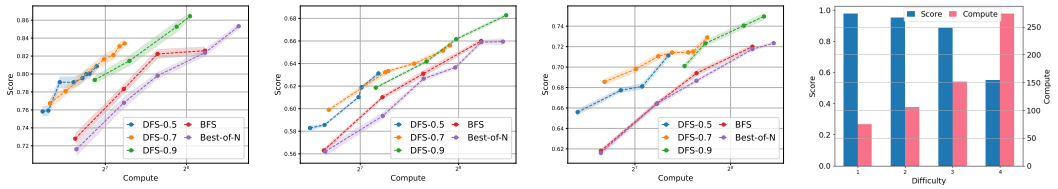
Model	N	BoN	FK[63]	DAS[33]	TreeG [19]	SVDD[39]	BFS (ours)
SD v1.5	4	0.702 ± 0.057	0.743 ± 0.037	0.878 ± 0.028	0.860 ± 0.033	0.667 ± 0.076	$\mathbf{0.882 \pm 0.029}$
SD v1.5	8	0.896 ± 0.031	0.926 ± 0.042	1.052 ± 0.033	1.023 ± 0.018	0.775 ± 0.087	$\mathbf{1.087 \pm 0.031}$
SD XL	4	1.085 ± 0.013	1.131 ± 0.022	1.181 ± 0.023	1.152 ± 0.023	1.036 ± 0.062	$\mathbf{1.194 \pm 0.024}$
SDXL	8	1.198 ± 0.021	1.251 ± 0.011	1.265 ± 0.019	1.261 ± 0.021	1.225 ± 0.027	$\mathbf{1.291 \pm 0.018}$

Table 2: Comparison of our BFS with prior methods

To compare our improved **BFS** (**Increase**, **Max**, **SSP**) with prior baselines, we additionally experiment with the SDXL model [58], which differs from the model used in our ablations. As shown in Table 2, our improved BFS consistently outperforms previous methods across compute budgets and models. In the following experiments, we use the improved BFS as our baseline.

5.2 Adaptive and Efficient Inference-Scaling with DFS

In this section, we evaluate the adaptivity and efficiency of DFS on the CompBench dataset [29], using the SSD-1B model [20] and a VLM [37] as our verifier. The detailed setup is provided in Appendix E.2. Through these experiments, we address the following questions:



(a) (b) (c) (d)

Figure 3: **CompBench [29] text-to-image results with DFS.** DFS-δ denotes DFS with threshold $\delta_t = \delta$. Figs. 3a, 3b, and 3c show DFS outperforming baseline BFS on the color, shape, and texture datasets, with up to $2\times$ lower cost than Best-of-N. Fig. 3d shows average compute allocation by DFS for prompts of increasing difficulty on the color dataset.

- *Can DFS outperform Best-of-N and prior particle-based methods?* As shown in Figs. 3a, 3b, and 3c, DFS consistently outperforms BFS and Best-of-N across datasets and threshold parameters, achieving up to $2\times$ lower computational cost.
- *Can DFS adjust compute allocation with different thresholds?* We evaluate DFS across a wide range of practical threshold values (0.5, 0.7, and 0.9) and find that lower thresholds automatically allocate less compute, while higher thresholds scale up compute for better quality. DFS consistently outperforms baseline methods across all threshold choices, demonstrating the robustness of our method.
- *Can DFS dynamically adjust compute allocation for different instances?* We measure the computational cost of DFS on prompts of varying difficulty in the color dataset. Threshold parameters are fixed, and the difficulty of a prompt is defined as the average score over four independent trials. As shown in Fig. 3d, difficult prompts with lower scores automatically consume more compute, without prior knowledge of difficulty as in [66].

Unlike linear-search methods that use a fixed exploration schedule, DFS offers higher efficiency and adaptivity, which may be of independent interest to the broader community.

5.3 Joint Scaling Local and Global Search

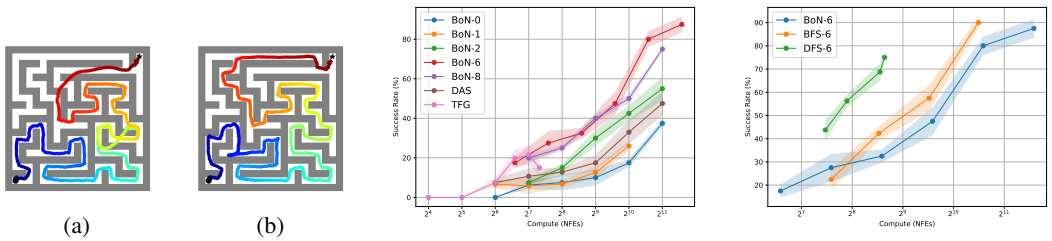
Although global search methods such as BFS and DFS can efficiently explore the generative space of the diffusion model, they are restricted to the modes of the base distribution and therefore cannot exceed the capabilities of the base model. To optimize the compositional objective in Eq. 5 and sample from high-reward regions beyond the base model, we propose scaling up local search steps via annealed Langevin MCMC, introducing a new scaling dimension for diffusion models. We validate the effectiveness of scaling local search in challenging decision-making domains, such as long-horizon planning and offline RL.

Baselines. To demonstrate the effectiveness of scaling local search steps, we compare with DAS [33], which also utilizes verifier gradients but applies only gradient guidance without multiple local search steps. We also compare with the state-of-the-art training-free guidance method TFG [94], which scales up the number of recurrence steps without any global search. Compute is measured as the total NFEs of both local and global search, ensuring a fair comparison. As shown in the following experiments, scaling local and global search separately yields suboptimal performance, while our joint scaling strategy establishes a new Pareto frontier.

5.3.1 Long Horizon Planning

Diffusion models have been widely adopted in planning for trajectory synthesis [75]. We evaluate long-horizon planning in a challenging PointMaze environment, using the base model trained following Diffuser [31], with the verifier defined as the total number of collisions between the trajectory and maze walls (see Appendix E.3 for details). Importantly, naively maximizing the verifier score does not guarantee a successful plan, and planning remains challenging even with full access to the maze layout [46, 49].

As shown in Fig. 4c, scaling up local search improves the overall Pareto frontier and significantly outperforms baseline methods. Scaling local search alone in TFG [94] is efficient with a low compute



(a) (b) (c) (d)

Figure 4: **(Illustration and results for maze planning)** Fig. 4a shows a failed trajectory without local search (start: •, goal: ⋆). Fig. 4b shows a successful trajectory after scaling local search. Fig. 4c presents Pareto curves for inference-scaling with varying local search steps, where BoN-i is best-of-N with i steps. Fig. 4d shows global search efficiency with 6 local search steps fixed.

budget but fails to scale with increased compute, as local search alone can become trapped in local optima. DAS [33] is more efficient than the corresponding BoN-0 baseline without local search, but underperforms best-of-N when more local search steps are used. In Fig. 4d, we show that local search can be combined with global search techniques such as BFS and DFS to further improve scaling efficiency, demonstrating the flexibility of our framework.

5.4 Offline Reinforcement Learning

Dataset	Environment	IQL	SfBC	DD	Diffuser	D-QL	QGPO	TFG	DAS	TTS(ours)
Medium-Expert	HalfCheetah	86.7	**92.6**	90.6	79.8	**96.1**	**93.5**	90.2 ± 0.2	$\mathbf{93.3 \pm 0.3}$	$\mathbf{93.9 \pm 0.3}$
Medium-Expert	Hopper	91.5	108.6	**111.8**	**107.2**	**110.7**	**108.0**	100.2 ± 3.5	105.4 ± 5.1	104.4 ± 3.1
Medium-Expert	Walker2d	**109.6**	**109.8**	**108.8**	**108.4**	**109.7**	**110.7**	$\mathbf{108.1 \pm 0.1}$	$\mathbf{111.4 \pm 0.1}$	$\mathbf{111.4 \pm 0.1}$
Medium	HalfCheetah	47.4	45.9	49.1	44.2	50.6	**54.1**	$\mathbf{53.1 \pm 0.1}$	$\mathbf{53.4 \pm 0.1}$	$\mathbf{54.8 \pm 0.1}$
Medium	Hopper	66.3	57.1	79.3	58.5	82.4	**98.0**	$\mathbf{96.2 \pm 0.5}$	71.3 ± 2.7	$\mathbf{99.5 \pm 1.7}$
Medium	Walker2d	78.3	77.9	**82.5**	79.7	**85.1**	**86.0**	$\mathbf{83.2 \pm 1.4}$	$\mathbf{83.9 \pm 0.9}$	$\mathbf{86.5 \pm 0.2}$
Medium-Replay	HalfCheetah	44.2	37.1	39.3	42.2	**47.5**	**47.6**	$\mathbf{45.0 \pm 0.3}$	42.2 ± 0.1	$\mathbf{47.8 \pm 0.4}$
Medium-Replay	Hopper	94.7	86.2	**100.0**	**96.8**	**100.7**	**96.9**	93.1 ± 0.1	$\mathbf{96.7 \pm 3.0}$	$\mathbf{97.4 \pm 4.0}$
Medium-Replay	Walker2d	73.9	65.1	75.0	61.2	**94.3**	84.4	69.8 ± 4.0	63.8 ± 2.0	79.3 ± 9.7
Average (Locomotion)		76.9	75.6	81.8	75.3	**86.3**	**86.6**	82.1	80.2	**86.1**

Table 3: Performance on D4RL locomotion tasks. For more details see Appendix. E.4.

Recently, diffusion models have emerged as a powerful action prior in robotics due to their ability to model complex and multimodal distributions [8, 44]. However, these diffusion policies are typically trained on offline datasets and struggle to adapt to reinforcement learning or test-time requirements. Following prior work [57], we formulate the offline RL problem as sampling from a Q-regularized distribution: $\pi^*(\boldsymbol{a}|\boldsymbol{s}) \propto \mu(\boldsymbol{a}|\boldsymbol{s})e^{\beta Q_\psi(\boldsymbol{s},\boldsymbol{a})}$, where Q_ψ is a learned Q-function representing preferences over actions, and μ is the behavior policy, which we model using a diffusion prior. We approach this problem from the inference-scaling perspective, composing an off-the-shelf pretrained diffusion policy with ground-truth Q-functions, without additional training.

Among the baselines, Diffuser [31], QGPO [45], and D-QL [81] are training-based methods that require joint training of the diffusion model and Q-function, while SfBC [6] can be viewed as a naive best-of-N approach. To demonstrate the effectiveness of our method, we allow TFG [94] and DAS [33] to use up to twice the compute of our method. As shown in Table 3, our method achieves performance comparable to training-based baselines, while DAS struggles on the Medium and Medium-Replay datasets where the model's capabilities are limited.

6 Limitations and Conclusion

In this work, we present a unified and principled framework for inference-time scaling of diffusion models. Our approach includes an improved BFS baseline, an adaptive DFS method for efficient global search, and a scalable local search strategy based on annealed Langevin MCMC. A potential limitation of our method is the risk of generating adversarial samples that exploit weaknesses in the verifier. To address this, we introduce a *double-verifier* strategy, employing separate verifiers for local and global search. Further details and evaluations are provided in Appendix F.

References

[1] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022. 24, 25

[2] M. S. Albergo and E. Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022. 3

[3] A. Bansal, H.-M. Chu, A. Schwarzschild, S. Sengupta, M. Goldblum, J. Geiping, and T. Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 843–852, 2023. 15

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. 1, 15, 24

[5] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023. 15

[6] H. Chen, C. Lu, C. Ying, H. Su, and J. Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023. 8, 24

[7] X. Cheng and P. Bartlett. Convergence of langevin mcmc in kl-divergence. In *Algorithmic learning theory*, pages 186–211. PMLR, 2018. 16

[8] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023. 1, 8, 15, 24

[9] H. Chung and J. C. Ye. Score-based diffusion models for accelerated mri. *Medical image analysis*, 80:102479, 2022. 2, 15

[10] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 1, 2, 3, 15

[11] K. Dong and T. Ma. STP: Self-play LLM Theorem Provers with Iterative Conjecturing and Proving, Mar. 2025. arXiv:2502.00212 [cs]. 14

[12] Y. Du, C. Durkan, R. Strudel, J. B. Tenenbaum, S. Dieleman, R. Fergus, J. Sohl-Dickstein, A. Doucet, and W. S. Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International conference on machine learning*, pages 8489–8510. PMLR, 2023. 2, 3, 5, 15, 16

[13] Y. Du, J. Mao, and J. B. Tenenbaum. Learning iterative reasoning through energy diffusion. *arXiv preprint arXiv:2406.11179*, 2024. 2

[14] Y. Du and I. Mordatch. Implicit generation and modeling with energy based models. *Advances in neural information processing systems*, 32, 2019. 16

[15] A. Durmus and E. Moulines. Non-asymptotic convergence analysis for the unadjusted langevin algorithm. *arXiv preprint arXiv:1507.05021*, 2015. 16

[16] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024. 15

[17] M. Gerber, N. Chopin, and N. Whiteley. Negative association, ordering and convergence of resampling methods, 2020. 5

[18] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 14

[19] Y. Guo, Y. Yang, H. Yuan, and M. Wang. Training-free guidance beyond differentiability: Scalable path steering with tree search in diffusion and flow models, 2025. 2, 5, 6, 21, 22

[20] Y. Gupta, V. V. Jaddipal, H. Prabhala, S. Paul, and P. Von Platen. Progressive knowledge distillation of stable diffusion xl using layer level loss. *arXiv preprint arXiv:2401.02677*, 2024. 6

[21] X. Han, S. Kumar, and Y. Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*, 2022. 15

[22] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. 1

[23] H. He, J. Liang, X. Wang, P. Wan, D. Zhang, K. Gai, and L. Pan. Scaling image and video generation via test-time evolutionary search, 2025. 15

[24] Y. He, N. Murata, C.-H. Lai, Y. Takida, T. Uesaka, D. Kim, W.-H. Liao, Y. Mitsufuji, J. Z. Kolter, R. Salakhutdinov, et al. Manifold preserving guided diffusion. *arXiv preprint arXiv:2311.16424*, 2023. 2, 15

[25] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016. 27

[26] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 3, 15, 18

[27] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 25

[28] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022. 1, 15

[29] K. Huang, K. Sun, E. Xie, Z. Li, and X. Liu. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems*, 36:78723–78747, 2023. 1, 6, 7

[30] A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. 14

[31] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022. 7, 8, 15, 23, 24, 25

[32] D. Kahneman. *Thinking, fast and slow*. macmillan, 2011. 14

[33] S. Kim, M. Kim, and D. Park. Test-time alignment of diffusion models without reward over-optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. 2, 4, 5, 6, 7, 8, 21, 22, 25

[34] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983. 16

[35] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021. 24

[36] G. Lee, T. N. N. Bao, J. Yoon, D. Lee, M. Kim, Y. Bengio, and S. Ahn. Adaptive inference-time scaling via cyclic diffusion search, 2025. 15

[37] J. Li, D. Li, C. Xiong, and S. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022. 6

[38] S. Li, K. Kallidromitis, A. Gokul, A. Koneru, Y. Kato, K. Kozuka, and A. Grover. Reflect-dit: Inference-time scaling for text-to-image diffusion transformers via in-context reflection. *arXiv preprint arXiv:2503.12271*, 2025. 15

[39] X. Li, Y. Zhao, C. Wang, G. Scalia, G. Eraslan, S. Nair, T. Biancalani, S. Ji, A. Regev, S. Levine, et al. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024. 2, 4, 5, 6, 21

[40] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023. 14

[41] S. Lin, B. Liu, J. Li, and X. Yang. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 5404–5411, 2024. 21

[42] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 3

[43] N. Liu, S. Li, Y. Du, A. Torralba, and J. B. Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pages 423–439. Springer, 2022. 15

[44] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024. 1, 8, 15, 24, 25

[45] C. Lu, H. Chen, J. Chen, H. Su, C. Li, and J. Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pages 22825–22855. PMLR, 2023. 1, 3, 8, 24, 25

[46] Y. Luo, C. Sun, J. B. Tenenbaum, and Y. Du. Potential based diffusion motion planning. *arXiv preprint arXiv:2407.06169*, 2024. 7

[47] N. Ma, M. Goldstein, M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, and S. Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pages 23–40. Springer, 2024. 3, 17

[48] N. Ma, S. Tong, H. Jia, H. Hu, Y.-C. Su, M. Zhang, X. Yang, Y. Li, T. Jaakkola, X. Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025. 2, 5, 15

[49] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake. Motion planning around obstacles with convex optimization. *Science robotics*, 8(84):eadf7843, 2023. 7, 22

[50] E. F. Moore. The shortest path through a maze. In *Proc. of the International Symposium on the Theory of Switching*, pages 285–292. Harvard University Press, 1959. 1

[51] M. Nakamoto, O. Mees, A. Kumar, and S. Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *8th Annual Conference on Robot Learning*, 2024. 24

[52] A. Newell, J. C. Shaw, and H. A. Simon. Report on a general problem solving program. In *IFIP congress*, volume 256, page 64. Pittsburgh, PA, 1959. 14

[53] A. Newell, H. A. Simon, et al. *Human problem solving*, volume 104. Prentice-hall Englewood Cliffs, NJ, 1972. 14

[54] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021. 3, 18

[55] S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. *arXiv preprint arXiv:2410.20092*, 2024. 22, 23

[56] J. Pearl. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., 1984. 1, 4

[57] J. Peters, K. Mulling, and Y. Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1607–1612, 2010. 8

[58] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 6

[59] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009. 1

[60] S. Sahoo, M. Arriola, Y. Schiff, A. Gokaslan, E. Marroquin, J. Chiu, A. Rush, and V. Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024. 15

[61] Y. Shen, X. Jiang, Y. Yang, Y. Wang, D. Han, and D. Li. Understanding and improving training-free loss-based diffusion guidance. *Advances in Neural Information Processing Systems*, 37:108974–109002, 2024. 5, 25

[62] A. Singh, J. D. Co-Reyes, R. Agarwal, A. Anand, P. Patil, X. Garcia, P. J. Liu, J. Harrison, J. Lee, K. Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023. 14

[63] R. Singhal, Z. Horvitz, R. Teehan, M. Ren, Z. Yu, K. McKeown, and R. Ranganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025. 2, 5, 6, 14, 15, 21, 22

[64] R. Singhal, Z. Horvitz, R. Teehan, M. Ren, Z. Yu, K. McKeown, and R. Ranganath. A general framework for inference-time scaling and steering of diffusion models, 2025. 4, 5

[65] S. A. Sloman. The empirical case for two systems of reasoning. *Psychological bulletin*, 119(1):3, 1996. 14

[66] C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. 5, 7, 14

[67] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3, 17

[68] J. Song, Q. Zhang, H. Yin, M. Mardani, M.-Y. Liu, J. Kautz, Y. Chen, and A. Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pages 32483–32498. PMLR, 2023. 1, 2, 15

[69] Y. Song and S. Ermon. Generative Modeling by Estimating Gradients of the Data Distribution, Oct. 2020. arXiv:1907.05600. 16

[70] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 15

[71] V. Subramaniam, Y. Du, J. B. Tenenbaum, A. Torralba, S. Li, and I. Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*, 2025. 14

[72] Z. Tan, S. Liu, X. Yang, Q. Xue, and X. Wang. Ominicontrol: Minimal and universal control for diffusion transformer. *arXiv preprint arXiv:2411.15098*, 2024. 15

[73] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972. 1

[74] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024. 1, 15

[75] T. Ubukata, J. Li, and K. Tei. Diffusion model for planning: A systematic literature review. *arXiv preprint arXiv:2408.10266*, 2024. 7

[76] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016. 15, 25

[77] C. Villani. *Topics in optimal transportation*. Graduate studies in mathematics ; v. 58. American Mathematical Society, Providence, R.I, 2003. 20

[78] P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011. 2, 3

[79] B. Wallace, M. Dang, R. Rafailov, L. Zhou, A. Lou, S. Purushwalkam, S. Ermon, C. Xiong, S. Joty, and N. Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8228–8238, 2024. 1, 15

[80] G. Wang, S. Zhang, T. Zhan, Z. Shen, J. Li, X. Hu, X. Sun, F. Wu, G. Deng, J. Zhang, et al. Unlocking the mysteries of openai o1: A survey of the reasoning abilities of large language models. 14

[81] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022. 8, 24, 25

[82] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011. 3

[83] A. Wibisono. Sampling as optimization in the space of measures: The langevin dynamics as a composite optimization problem. In *Conference on learning theory*, pages 2093–3027. PMLR, 2018. 5, 15, 19

[84] L. Wu, B. Trippe, C. Naesseth, D. Blei, and J. P. Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023. 2, 14

[85] Y. Wu, Z. Sun, S. Li, S. Welleck, and Y. Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024. 14

[86] Z. Wu, S. Huang, Z. Zhou, H. Ying, J. Wang, D. Lin, and K. Chen. Internlm2. 5-stepprover: Advancing automated theorem proving via expert iteration on large-scale lean problems. *arXiv preprint arXiv:2410.15700*, 2024. 14

[87] E. Xie, J. Chen, Y. Zhao, J. Yu, L. Zhu, C. Wu, Y. Lin, Z. Zhang, M. Li, J. Chen, et al. Sana 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. *arXiv preprint arXiv:2501.18427*, 2025. 1

[88] Y. Xie, V. Jampani, L. Zhong, D. Sun, and H. Jiang. Omnicontrol: Control any joint at any time for human motion generation. *arXiv preprint arXiv:2310.08580*, 2023. 15

[89] J. Xu, X. Liu, Y. Wu, Y. Tong, Q. Li, M. Ding, J. Tang, and Y. Dong. Imagereward: learning and evaluating human preferences for text-to-image generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 15903–15935, 2023. 1, 6, 15

[90] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022. 15

[91] Y. Xu, M. Deng, X. Cheng, Y. Tian, Z. Liu, and T. Jaakkola. Restart sampling for improving generative processes. *Advances in Neural Information Processing Systems*, 36:76806–76838, 2023. 14

[92] S. Yang, Y. Du, S. K. S. Ghasemipour, J. Tompson, L. P. Kaelbling, D. Schuurmans, and P. Abbeel. Learning interactive real-world simulators. In *The Twelfth International Conference on Learning Representations*, 2024. 1

[93] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023. 14

[94] H. Ye, H. Lin, J. Han, M. Xu, S. Liu, Y. Liang, J. Ma, J. Y. Zou, and S. Ermon. Tfg: Unified training-free guidance for diffusion models. *Advances in Neural Information Processing Systems*, 37:22370–22417, 2024. 1, 2, 5, 7, 8, 15, 19, 20, 25

[95] J. Yu, Y. Wang, C. Zhao, B. Ghanem, and J. Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23174–23184, 2023. 2, 3, 15

[96] E. Zelikman, Y. Wu, J. Mu, and N. Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022. 14

[97] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023. 15

[98] Y. Zhang, E. Tzeng, Y. Du, and D. Kislyuk. Large-scale reinforcement learning for diffusion models. In *European Conference on Computer Vision*, pages 1–17. Springer, 2024. 15

[99] S. Zhao, R. Brekelmans, A. Makhzani, and R. Grosse. Probabilistic inference in language models via twisted sequential monte carlo. *arXiv preprint arXiv:2404.17546*, 2024. 14

A Appendix Overview

In Sec. B, we provide a in-depth review of literature related to inference-time scaling and diffusion models. In Sec. C, we elaborate on local search with Langevin MCMC, and in Sec. D we provide the pseudo code and design of global search algorithms BFS and DFS. In Sec. E, we provide the details of all the experiments. In Sec. F we provide the details of double-verifier for mitigating reward hacking.

B Additional Related Works

Inference-time scaling. Scaling compute in inference-time with "slow thinking" has its long history grounded in cognitive science, known as "system 2" thinking [32, 65]. In [52, 53], Newell and his colleagues formalized problem solving as tree search in a combinatorial space, and [93] uses tree-of-thoughts to enable LLM reasoning with multiple exploration paths, using BFS and DFS as strategic search algorithms.

Recently, long chain-of-though (CoT) reasoning has demonstrated remarkable performance for LLM reasoning [30, 18], where the long CoT reasoning ability is incentivized through reinforcement learning [18]. Notably, the CoT process demonstrates reasoning activities such as self-verification, backtracking and self-correction. Using a process reward model [40], we can also conduct explicit tree search without training the language model. [85] propose reward-balanced search (REBASE) which is a special instance of BFS, and [66] applied beam-search to difficult math problems, showing compute-optimal inference can be achieved via selecting different strategy for problems with different difficulty. We refer the readers to [80] for a comprehensive review.

Inference-time scaling could also be used to improve the model itself, known as expert iteration. In [32], they proposed reinforcement learning with expert iteration in games, where the expert is constructed combining the base policy with Monte Carlo Tree Search (MCTS), and [86] applied expert iteration to automated theorem proving. Self improvements can also be achieved through iterative self-training [71, 96, 11, 62].

Inference scaling in diffusion models. The inference-time compute of diffusion models depends heavily on the number of denoising steps. [91] showed that recursive restart sampling can reduce cumulative error during sampling, which can be regarded as scaling the number of denoising steps. More recently, [63] proposed a Sequential Monte Carlo (SMC) [84, 99] style method, known as Feynman-Kac steering, which can be seen as a instance of BFS. Besides image generation, they also

applied it to diffusion language models [21, 60]. Additionally, [48] explored inference-time scaling of diffusion models with local zero order search and global search over paths for image generation. They also experimented with different verifiers, such as oracle verifiers, self-supervised verifiers, and studied the verifier-task alignment problem. There zero-order search can be understood as an uninformed version of local search which exhibits low efficiency, and when utilizing gradients, they need to back-propagate through the entire diffusion sampling chain, causing high computation and memory overheads. Compared with their work, we propose efficient gradient-based local search with theoretically grounded Langevin MCMC, which we show is crucial in many tasks. Also, we provide systematic experiments on the compute efficiency of global search methods. Their proposed methods can thus be understood as a instance within our search framework.

Apart from search, [38] exploits the in-context learning abilities of foundation models to provide revision during sampling. Specifically, they leverage the multi-modal capabilities of VLMs to provide feedbacks on past generated images, and train the model to condition on past images and feedback.

Diffusion models and applications. Diffusion models [26, 70, 76] has shown great performance in generative modeling for continuous data domains, such as image [16, 10], videos [28] and molecules [90]. Due to its expressive power on modeling multimodal and complex distributions, they have also been widely used as a decision prior in robotics. [31] proposes the first work on using diffusion models to generate plans. [8] uses the diffusion model for visuo-motor policy in robotics. Recently, a series of robotics foundation models utilize diffusion heads as action experts [4, 74], while [44] trains a end-to-end diffusion transformer for bimanual manipulation. In this work, we demonstrate that inference-time scaling can be especially helpful for decision making tasks with diffusion models.

Control and alignment of diffusion models. To align the diffusion model with flexible objectives, training-free guidance [94, 3, 9, 68, 95, 24] and compositional generation [12, 43] combines the diffusion models with classifiers or other diffusion models at inference time, while RL-based methods [98, 79, 89, 5] finetune the diffusion model using reward or preference signals. ControlNet [97] style approaches have also been used to add additional conditions for sampling, where [72] designs a control block for diffusion transformers, and [88] uses a combination of guidance and controlnet for controllable human motion generation.

Discussions on concurrent works. Adaptive Bi-directional Cyclic Diffusion (ABCD) [36] propose a search based inference scaling framework which could be seen as a combination of DFS and BFS. Unlike DFS that determines backtracking with a quality threshold, ABCD maintains a set of particles and backtracks by sending the particles to all different noise levels. The termination condition is determined by whether backtracking to higher noise levels increases sample quality. Compared with DFS, ABCD can have smaller score estimation errors since they evaluate particles when fully denoised, and can explore the generative space sufficiently via a combination of BFS and DFS. However, ABCD can not adaptively adjust compute allocation on different instances due to its special termination condition, and requires more compute on easy instances since it will denoise a set of particles regardless of whether sample quality is satisfactory.

EvoSearch [23] propose to use evolutionary search to scale inference compute in image and video generation. At selected time steps, they evaluate the particles via full denoising, and maintain high score particles and mutate low score particles with adding noise. Their method demonstrated improved performance over the FK-steering [63] baseline. It improves upon naive BFS with local search via mutation for low quality particles.

C Details about local search with Langevin MCMC

In this section we provide a comprehensive and detailed overview of (annealed) Langevin MCMC based methods used in local search, as well as proving Proposition. 1.

C.1 Langevin MCMC as gradient flow in measure space

Following [83], the Langevin SDE in sample space corresponds to gradient flow of the KL-divergence in measure space. Here we provide a brief overview.

Define our target distribution that we wish to sample from as ν, and the distribution of our current sample as ρ. We define the KL-divergence (relative entropy) as:

$$H_\nu(\rho) = \int \rho \log \frac{\rho}{\nu} \,. \tag{6}$$

Thus, sampling from ν can be seen as minimizing H, since the minimum of H is achieved at $\rho = \nu$ with $H_\nu(\rho) = 0$. Furthermore, ν is the only stationary point of H even for multimodal distributions. Thus we can sample from ν when optimizing H via gradient based methods.

We have the gradient flow of H in Eq. 6 follows the following PDE:

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (\rho \nabla(-\log \nu)) + \Delta \rho \,, \tag{7}$$

which is known as the Fokker-Planck equation. Here, $\rho = \rho(\boldsymbol{x}, t)$ is a smooth positive density evolving through time, driven by the dynamics of the sample $\boldsymbol{x}$. The dynamics in sample space corresponding to Eq. 7 is the Langevin SDE:

$$d\boldsymbol{x}_t = \nabla \log \nu(\boldsymbol{x}_t) dt + \sqrt{2} d\boldsymbol{w}_t \,. \tag{8}$$

where $(\boldsymbol{x}_t)_{t \geq 0}$ is a stochastic process with measure ρ_t, and $(\boldsymbol{w}_t)_{t \geq 0}$ is standard Brownian motion. That is, if $\boldsymbol{x}_t \sim \rho_t$ evolves according to the dynamics in Eq. 8, then the measure $\rho(\boldsymbol{x}, t) = \rho_t$ evolves according to the PDE in Eq. 7, conducting gradient optimization in measure space.

In practice, we implement Eq. 8 through discretization, which is known as the unadjusted Langevin algorithm (ULA):

$$\boldsymbol{x}^{i+1} = \boldsymbol{x}^i + \eta \nabla_{\boldsymbol{x}^i} \log \nu(\boldsymbol{x}^i) + \sqrt{2\eta} \boldsymbol{\epsilon}^i \,, \tag{9}$$

with $\boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. When $\eta \to 0$, the ULA converges to Langevin SDE, providing exact sampling.

Previous works [15, 7] show that ULA can efficiently converge to the target measure ν if ν is log-concave and smooth. However, when facing complex and multimodal distributions, we can only guarantee convergence to the concave vicinity.

C.2 Annealed Langevin MCMC Sampling

Langevin MCMC have been used to perform implicit sampling in energy-based models [14] and score-based models [69]. However, these methods suffer from inaccurate score estimation and low density regions [69]. In [69] they propose to perturb the data with gaussian noise, eventually smoothing the data distribution:

$$q(\boldsymbol{x}_t) = \int_{\boldsymbol{x}_0} p_0(\boldsymbol{x}_0) \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}_0, \sigma_t^2 \boldsymbol{I}) \,,$$

and creating a sequence of annealed distributions $\{q(\boldsymbol{x}_t)\}_{t=0}^T$ which converges to $p_0(\boldsymbol{x}_0)$. Since they are smoothed by gaussian noise, we can improve the mixing time of Langevin MCMC on multimodal distributions by sampling from these intermediate distributions, sharing similar spirits with simulated annealing [34].

In [12], they extend this method to compositional generation of diffusion models. Specifically, we consider sampling from a product distribution $p_0^{\text{prod}}(\boldsymbol{x}_0) \propto p_0^1(\boldsymbol{x}_0) p_0^2(\boldsymbol{x}_0)$, where $p_0^1(\boldsymbol{x}_0)$ and $p_0^2(\boldsymbol{x}_0)$ are distributions of different diffusion models. Since we have access to the score functions $\nabla_{\boldsymbol{x}_t} \log q_t^1(\boldsymbol{x}_t)$ and $\nabla_{\boldsymbol{x}_t} \log q_t^2(\boldsymbol{x}_t)$ through the diffusion model, we can construct a sequence of annealing distributions $\tilde{q}_t^{\text{prod}}(\boldsymbol{x}_t)$ such that:

$$\nabla_{\boldsymbol{x}_t} \log \tilde{q}_t^{\text{prod}}(\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}_t} \log q_t^1(\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log q_t^2(\boldsymbol{x}_t) \,.$$

By sampling from the sequence $\left\{\tilde{q}_t^{\text{prod}}(\boldsymbol{x}_t)\right\}$, we can arrive at $\tilde{q}_0^{\text{prod}}(\boldsymbol{x}_0)$ which is equal to $p_0^{\text{prod}}(\boldsymbol{x}_0)$. A key difference from sampling from $\left\{\tilde{q}_t^{\text{prod}}(\boldsymbol{x}_t)\right\}$ and direct diffusion sampling is that the diffusion process with $p_0^{\text{prod}}(\boldsymbol{x}_0)$ defined as

$$q_t^{\text{prod}}(\boldsymbol{x}_t) = \int_{\boldsymbol{x}_0} p_0^{\text{prod}}(\boldsymbol{x}_0) q(\boldsymbol{x}_t | \boldsymbol{x}_0)$$

is different from $\tilde{q}_t^{\text{prod}}(\boldsymbol{x}_t)$. The score of $q_t^{\text{prod}}(\boldsymbol{x}_t)$ can be derived as:

$$\nabla_{\boldsymbol{x}_t} \log q_t^{\text{prod}}(\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}_t} \log \left(\int_{\boldsymbol{x}_0} p_0^1(\boldsymbol{x}_0) p_0^2(\boldsymbol{x}_0) q(\boldsymbol{x}_t|\boldsymbol{x}_0) \right) ,$$

which is not equal to

$$\nabla_{\boldsymbol{x}_t} \log \tilde{q}_t^{\text{prod}}(\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}_t} \log \left(\int_{\boldsymbol{x}_0} p_0^1(\boldsymbol{x}_0) q(\boldsymbol{x}_t|\boldsymbol{x}_0) \right) + \nabla_{\boldsymbol{x}_t} \log \left(\int_{\boldsymbol{x}_0} p_0^2(\boldsymbol{x}_0) q(\boldsymbol{x}_t|\boldsymbol{x}_0) \right) ,$$

and thus intractable to compute directly.

A key distinction between annealed Langevin MCMC sampling and reverse diffusion sampling is that we run multiple Langevin MCMC steps on the *same noise level*, while reverse diffusion goes from high noise level to low noise level via denoising. A minimal pseudo code is shown in Alg. 1.

Algorithm 1 Annealed Langevin MCMC sampling

Input: sequence of annealing distributions $\{\tilde{q}_t(\boldsymbol{x}_t)\}_{t=0}^T$, number of MCMC steps N, step size $\{\eta_t\}_{t=0}^T$. (Optional) reverse transition kernel $\{\tilde{p}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\}_{t=0}^T$.
Init: $\boldsymbol{x}_T^0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $t = T, \cdots, 1$ **do**
  for $i = 0, 1, \cdots, N-1$ **do**
    Perform Langevin MCMC steps:

$$\boldsymbol{x}_t^{i+1} = \boldsymbol{x}_t^i + \eta_t \nabla_{\boldsymbol{x}_t} \log \tilde{q}_t(\boldsymbol{x}_t^i) + \sqrt{2\eta_t}\boldsymbol{\epsilon}_t^i , \quad \boldsymbol{\epsilon}_t^i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) .$$

  end for
  (Optional) transit to next time step: $\boldsymbol{x}_{t-1}^0 \sim \tilde{p}_\theta(\cdot|\boldsymbol{x}_t^N)$. If no reverse kernel initialize $\boldsymbol{x}_{t-1}^0 = \boldsymbol{x}_t^N$.
end for
Return $\boldsymbol{x}_0$

C.3 Annealed Langevin MCMC with recurrent training-free guidance

In this section, we prove the connection between annealed Langevin MCMC (Alg. 1) and training-free guidance (Alg. 2) in Proposition. 1. We divide the proof into two parts. In Sec. C.3.1 we prove the equivalence between naive recurrence steps and Langevin MCMC. Then in Sec. C.3.2, we prove that adding the guidance term is defining an annealing path that biases towards high score regions. Finally, we provide a rigorous convergence analysis in Sec. C.3.3.

C.3.1 Equivalence between Langevin MCMC and naive recurrence

Consider the diffusion process with the following stochastic interpolant [47]:

$$\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon} .$$

We denote the score function of $q_t(\boldsymbol{x}_t)$ as $\nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t) = s(\boldsymbol{x}_t, t)$. Recall the forward process in Eq. 1:

$$\boldsymbol{x}_t = \frac{\alpha_t}{\alpha_{t-1}} \boldsymbol{x}_{t-1} + \sqrt{\alpha_t^2 \left(\frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_{t-1}^2}{\alpha_{t-1}^2} \right)} \boldsymbol{\epsilon} . \tag{10}$$

In a recurrence step in Line. 5, we first solve $\boldsymbol{x}_{t-1}^i$ from $\boldsymbol{x}_t^i$ using the learned score function $s(\boldsymbol{x}_t^i, t)$, then add noise to $\boldsymbol{x}_{t-1}^i$ to obtain the recurrent sample $\boldsymbol{x}_t^{i+1}$, where the superscript denotes the recurrence step index: $i = 0, 1, \cdots, N_{\text{recur}}$. Depending on different solvers, we have different formulations of $\boldsymbol{x}_t^{i+1}$.

DDIM sampler. When using DDIM [67] sampler, we have the reverse step as:

$$\boldsymbol{x}_{t-1} = \frac{\alpha_{t-1}}{\alpha_t} \boldsymbol{x}_t + \sigma_t^2 \left(\frac{\alpha_{t-1}}{\alpha_t} - \frac{\sigma_{t-1}}{\sigma_t} \right) s(\boldsymbol{x}_t, t) , \tag{11}$$

where $s(\boldsymbol{x}_t, t)$ is the score function $\nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t)$. Thus, we have:

$$\begin{aligned}\boldsymbol{x}_t^{i+1} &= \frac{\alpha_t}{\alpha_{t-1}}\boldsymbol{x}_{t-1}^i + \alpha_t\sqrt{\frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_{t-1}^2}{\alpha_{t-1}^2}}\boldsymbol{\epsilon}^i \\ &= \boldsymbol{x}_t^i + \sigma_t^2\left(1 - \frac{\alpha_t}{\alpha_{t-1}}\frac{\sigma_{t-1}}{\sigma_t}\right)s(\boldsymbol{x}_t^i, t) + \sigma_t\sqrt{1 - \frac{\alpha_t^2}{\alpha_{t-1}^2}\frac{\sigma_{t-1}^2}{\sigma_t^2}}\boldsymbol{\epsilon}^i\,.\end{aligned}$$

Denote $\lambda_t = \log\frac{\alpha_t}{\sigma_t}$, then we have:

$$\begin{aligned}\boldsymbol{x}_t^{i+1} &= \boldsymbol{x}_t^i + \sigma_t^2\left(1 - e^{\lambda_t - \lambda_{t-1}}\right)s(\boldsymbol{x}_t^i, t) + \sigma_t\sqrt{1 - e^{2(\lambda_t - \lambda_{t-1})}}\boldsymbol{\epsilon}^i \\ &= \boldsymbol{x}_t^i + \sigma_t^2\left(1 - e^{\lambda_t - \lambda_{t-1}}\right)s(\boldsymbol{x}_t^i, t) + \sigma_t\sqrt{\left(1 - e^{\lambda_t - \lambda_{t-1}}\right)\left(1 + e^{\lambda_t - \lambda_{t-1}}\right)}\boldsymbol{\epsilon}^i\,,\end{aligned}$$

where $1 + e^{\lambda_t - \lambda_{t-1}} \to 2$ when $T \to \infty$ and denoising step size approaches 0, as $\lambda_t - \lambda_{t-1} \to 0$.

DDPM sampler. In DDPM [26], we parametrize the posterior distribution as:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}\left(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \Sigma_\theta(\boldsymbol{x}_t, t)\right)\,, \tag{12}$$

where the posterior mean is:

$$\mu_\theta(\boldsymbol{x}_t, t) = \frac{\alpha_{t-1}}{\alpha_t}\boldsymbol{x}_t + \left(\sigma_t^2\frac{\alpha_{t-1}}{\alpha_t} - \sigma_{t-1}^2\frac{\alpha_t}{\alpha_{t-1}}\right)s(\boldsymbol{x}_t, t)\,.$$

[26] parameterizes the posterior variance as $\Sigma_\theta(\boldsymbol{x}_t, t) = \beta_t\boldsymbol{I}$ or $\Sigma_\theta(\boldsymbol{x}_t, t) = \tilde{\beta}_t\boldsymbol{I}$:

$$\begin{aligned}\beta_t &= \alpha_t^2\left(\frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_{t-1}^2}{\alpha_{t-1}^2}\right)\,, \\ \tilde{\beta}_t &= \frac{\sigma_{t-1}^2}{\sigma_t^2}\beta_t\,,\end{aligned}$$

while [54] propose to train the posterior variance as $\Sigma_\theta(\boldsymbol{x}_t, t) = \exp\left(v\log\beta_t + (1-v)\log\tilde{\beta}_t\right)$.

Thus, a backward step can be written as:

$$\boldsymbol{x}_{t-1} = \frac{\alpha_{t-1}}{\alpha_t}\boldsymbol{x}_t + \left(\sigma_t^2\frac{\alpha_{t-1}}{\alpha_t} - \sigma_{t-1}^2\frac{\alpha_t}{\alpha_{t-1}}\right)s(\boldsymbol{x}_t, t) + \Sigma_\theta^{1/2}(\boldsymbol{x}_t, t)\boldsymbol{\epsilon}_{\text{post}}\,,$$

where $\boldsymbol{\epsilon}_{\text{post}}$ denotes the noise added in the posterior sampling step. Then, we can write the recurrence step as:

$$\begin{aligned}\boldsymbol{x}_t^{i+1} &= \boldsymbol{x}_t^i + \left(\sigma_t^2 - \sigma_{t-1}^2\frac{\alpha_t^2}{\alpha_{t-1}^2}\right)s(\boldsymbol{x}_t^i, t) + \Sigma_\theta^{1/2}(\boldsymbol{x}_t^i, t)\boldsymbol{\epsilon}_{\text{post}}^i + \alpha_t\sqrt{\frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_{t-1}^2}{\alpha_{t-1}^2}}\boldsymbol{\epsilon}_{\text{forward}}^i \\ &= \boldsymbol{x}_t^i + \beta_t s(\boldsymbol{x}_t^i, t) + \sqrt{\Sigma_\theta(\boldsymbol{x}_t, t) + \beta_t\boldsymbol{I}}\boldsymbol{\epsilon}^i\,,\end{aligned}$$

where $\Sigma_\theta(\boldsymbol{x}_t, t) \to \beta_t\boldsymbol{I}$ when $T \to \infty$, and the denoising step size approaches 0.

Putting together. In general, we can write the recurrence step as:

$$\boldsymbol{x}_t^{i+1} = \boldsymbol{x}_t^i + a_t r_t s(\boldsymbol{x}_t^i, t) + \sqrt{2a_t}\boldsymbol{\epsilon}^i\,. \tag{13}$$

with $a_t \to 0$ and $r_t \to 1$ as the denoising step size approaches 0:

- For DDIM sampler, we have $a_t = \frac{1}{2}\alpha_t^2\left(\frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_{t-1}^2}{\alpha_{t-1}^2}\right)$ and $r_t = \frac{2}{1+e^{\lambda_t - \lambda_{t-1}}}$.
- For DDPM sampler, we have $a_t = \frac{1}{2}\alpha_t^2\left(\frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_{t-1}^2}{\alpha_{t-1}^2}\right)$ and $1 \le r_t \le \frac{2}{1+\frac{\sigma_{t-1}^2}{\sigma_t^2}}$.

Thus, it can be seen as a approximation of the ULA in Eq. 9, and also a discretization of the Langevin SDE in Eq. 8.

C.3.2 Annealed Langevin MCMC with guidance

When applying training free guidance [94] during the recurrence, we have:

$$\boldsymbol{x}_t^{i+1} = \boldsymbol{x}_t^i + a_t r_t s(\boldsymbol{x}_t^i, t) + \sqrt{2a_t}\boldsymbol{\epsilon}^i + \boldsymbol{\Delta}(\boldsymbol{x}_t, t)\,,$$

where a_t, b_t are the coefficients of the recurrence equation in Eq. 13 without guidance. In general, $\boldsymbol{\Delta_t} = \rho_t \nabla_{\boldsymbol{x}_t} \log f(\boldsymbol{x}_{0|t}) + \mu_t \alpha_t \nabla_{\boldsymbol{x}_{0|t}} \log f(\boldsymbol{x}_{0|t})$, where ρ_t, μ_t controls the guidance strength. We then show that the guidance term can be considered as the score function of a set of annealed verifiers $\left\{\hat{f}(\boldsymbol{x}_t)\right\}_{t=0}^{T}$.

When considering 'variance guidance' in Line. 7, we have $\boldsymbol{\Delta}_{\text{var}} = \rho_t \nabla_{\boldsymbol{x}_t} \log f(\boldsymbol{x}_{0|t})$. Thus, we can define $\hat{f}_t^{\text{var}}(\boldsymbol{x}_t) = f(\boldsymbol{x}_{0|t})$, which satisfies $\hat{f}_0^{\text{var}}(\boldsymbol{x}_0) = f(\boldsymbol{x}_0)$. Similarly, for 'mean guidance' in Line. 8, we have

$$\begin{aligned}\boldsymbol{\Delta}_{\text{mean}} &= \mu_t \alpha_t \nabla_{\boldsymbol{x}_{0|t}} \log f(\boldsymbol{x}_{0|t}) \\ &= \mu_t \frac{\sigma_t^2}{\boldsymbol{\Sigma}_{0|t}} \nabla_{\boldsymbol{x}_t} \log f(\boldsymbol{x}_{0|t})\,,\end{aligned}$$

where the second Equation follows from Lemma 3.3 in [94]. Thus, there exists a set of functions $\hat{f}_t^{\text{mean}}(\boldsymbol{x}_t)$ such that $\nabla_{\boldsymbol{x}_t} \log \hat{f}_t^{\text{mean}}(\boldsymbol{x}_t) = \frac{\sigma_t^2}{\boldsymbol{\Sigma}_{0|t}} \nabla_{\boldsymbol{x}_t} \log f(\boldsymbol{x}_{0|t})$, and we can see that when $t \to 0, \nabla_{\boldsymbol{x}_t} \log \hat{f}_t^{\text{mean}}(\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}_0} \log f(\boldsymbol{x}_0)$. If we additionally incorporate the 'implicit dynamics' in Line. 4, our arguments still stands since the smoothed objective $\tilde{f}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{\delta}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})} f(\boldsymbol{x} + \bar{\gamma}\sigma_t\boldsymbol{\delta})$ converges to f with $t \to 0$ and $\sigma_t \to 0$.

Combining the two terms together, we have $\boldsymbol{\Delta}_t = c_t \nabla_{\boldsymbol{x}_t} \log \hat{f}_t(\boldsymbol{x}_t)$ with $\hat{f}_t = \hat{f}_t^{\text{var}} \cdot \hat{f}_t^{\text{mean}}$. Thus, recurrence with guidance can be written as:

$$\begin{aligned}\boldsymbol{x}_t^{i+1} &= \boldsymbol{x}_t^i + a_t r_t s(\boldsymbol{x}_t^i, t) + \sqrt{2a_t}\boldsymbol{\epsilon}^i + c_t \nabla_{\boldsymbol{x}_t} \log \hat{f}_t(\boldsymbol{x}_t) \\ &= \boldsymbol{x}_t^i + a_t r_t \nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t) \hat{f}_t(\boldsymbol{x}_t)^{c_t/a_t r_t} + \sqrt{2a_t}\boldsymbol{\epsilon}^i\,,\end{aligned}$$

Thus, we have defined the annealing path as $\tilde{q}_t(\boldsymbol{x}_t) = q_t(\boldsymbol{x}_t)\hat{f}_t(\boldsymbol{x}_t)^{c_t/a_t r_t}\,,\ t = 1, 2, \cdots, T$.

C.3.3 Convergence analysis

In this section, we provide a rigorous convergence analysis of recurrence to the target distribution $\tilde{q}_t(\boldsymbol{x}_t)$.

Theorem 1. *Suppose $\tilde{q}_t(\boldsymbol{x}_t)$ has bounded support, is α-strongly log-concave and L-log-smooth, and $-\nabla^2 \log \tilde{q}_t$ is M-Lipschitz. Denote $\boldsymbol{x}_t^{N_{recur}}$ as the sample after N_{recur} steps of recurrence, we can bound the Wasserstein distance between the distribution of $\boldsymbol{x}_t^{N_{recur}}$ and $\tilde{q}_t$ as:*

$$W_2(p(\boldsymbol{x}_t^{N_{recur}}), \tilde{q}_t) = \mathcal{O}\left(\sqrt{\lambda_{t-1} - \lambda_t} + e^{-2\lambda_t} - e^{-2\lambda_{t-1}} + (1 - e^{-2\lambda_t} + e^{-2\lambda_{t-1}})^{N_{recur}}\right)\,,$$

where $\lambda_t = \log \frac{\alpha_t}{\sigma_t}$ is half of the log SNR.

Proof. Recall recurrence is equivalent to the following recursion equation:

$$\begin{aligned}\boldsymbol{x}_t^{i+1} &= \boldsymbol{x}_t^i + a_t r_t \nabla_{\boldsymbol{x}_t} \log \tilde{q}_t(\boldsymbol{x}_t^i) + \sqrt{2a_t}\boldsymbol{\epsilon}^i \\ &= \boldsymbol{x}_t^i + a_t \nabla_{\boldsymbol{x}_t} \log \tilde{q}_t(\boldsymbol{x}_t^i)^{r_t} + \sqrt{2a_t}\boldsymbol{\epsilon}^i\,.\end{aligned}$$

Thus, recurrence is equivalent to running unadjusted Langevin algorithm (ULA) on the tempered distribution $p^{\text{tempered}} \propto \tilde{q}_t^{r_t}$. Using Lemma 1 and Lemma 2 from [83], given the regularity conditions on $\tilde{q}_t$, we can bound the discretization error from ULA as:

$$\begin{aligned}W_2(p^{\text{tempered}}, p(\boldsymbol{x}_t^{N_{\text{recur}}})) &= \mathcal{O}\left(a_t + (1 - a_t)^{N_{\text{recur}}}\right) \\ &= \mathcal{O}\left(\frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_{t-1}^2}{\alpha_{t-1}^2} + (1 - \frac{\sigma_t^2}{\alpha_t^2} + \frac{\sigma_{t-1}^2}{\alpha_{t-1}^2})^{N_{\text{recur}}}\right) \\ &= \mathcal{O}\left(e^{-2\lambda_t} - e^{-2\lambda_{t-1}} + (1 - e^{-2\lambda_t} + e^{-2\lambda_{t-1}})^{N_{\text{recur}}}\right)\,.\end{aligned}$$

To bound $W_2(p^{\text{tempered}}, \tilde{q}_t)$, we can bound the TV distance as $\text{TV}(p^{\text{tempered}}, \tilde{q}_t) \leq \mathcal{O}(r_t - 1)$. Following Proposition 7.10 in [77] for distributions with bounded support, we have:

$$
\begin{aligned}
& W_2(p^{\text{tempered}}, \tilde{q}_t) \\
&= \mathcal{O}\left(\sqrt{\text{TV}(p^{\text{tempered}}, \tilde{q}_t)}\right) \\
&= \mathcal{O}\left(\sqrt{r_t - 1}\right) \\
&= \mathcal{O}\left(\sqrt{1 - \min\left(\frac{\alpha_t \sigma_{t-1}}{\alpha_{t-1}\sigma_t}, \frac{\sigma_{t-1}^2}{\sigma_t^2}\right)}\right) \\
&= \mathcal{O}\left(\sqrt{\log \frac{\sigma_t}{\sigma_{t-1}} + \max\left(\log \frac{\alpha_{t-1}}{\alpha_t}, \log \frac{\sigma_t}{\sigma_{t-1}}\right)}\right) \\
&= \mathcal{O}\left(\log \frac{\alpha_{t-1}}{\alpha_t} + \log \frac{\sigma_t}{\sigma_{t-1}}\right) \\
&= \mathcal{O}\left(\lambda_{t-1} - \lambda_t\right) .
\end{aligned}
$$

Putting together we obtain our desired bound. □

C.4 Relationship between Langevin MCMC and gradient ascent

In training-free guidance, most prior works only apply gradient ascent without recurrence. Here we provide a theoretical analysis of both methods.

Recall the KL-divergence objective in Eq. 6, which can be further decomposed when we are sampling from a compositional distribution of $p_0(\boldsymbol{x}_0)$ and verifier $f(\boldsymbol{x}_0)$, with $\nu \propto p_0 \cdot f$:

$$H_\nu(\rho) = \mathbb{E}_\rho[-\log f] + H_{p_0}(\rho) + \log Z \,.$$

where $Z = \int p_0 f$ is a normalization constant. Thus, gradient ascent is optimizing the verifier objective $\mathbb{E}_\rho[-\log f]$, while Langevin MCMC in Eq. 13 is optimizing the divergence between current sample and base distribution $H_{p_0}(\rho)$. This explains why naive gradient updates leads to OOD samples, and recurrence effectively mitigates this issue, acting as a contraction force pulling the sample back to the original manifold. However, since we start from p_0 as the distribution of our initial sample, sometimes we can omit the recurrence if the guidance strength is small. But if we wish to traverse different modes with multiple gradient updates, introducing recurrence helps to avoid OOD during optimization.

C.5 Implementing Local Search with TFG hyper-parameter space

Due to the equivalence between annealed Langevin MCMC and training-free guidance with recurrence, we can implement local search with Langevin MCMC using the TFG framework of [94], efficiently searching the hyperparameters. Here we provide a overview of the algorithm and design space. Following Sec. C.3, every iteration of recurrence in Line. 5 is equivalent to an annealed Langevin MCMC step, thus N_{recur} is equal to the number of local search steps.

For time varying schedules ρ_t, μ_t, we follow [94] and propose to use either the 'increase' schedule:

$$s_t = T \frac{\alpha_t / \alpha_{t-1}}{\sum_{t=1}^{T} \alpha_t / \alpha_{t-1}} \,, \tag{14}$$

where we increase the guidance strength as we denoise: $s_T < s_{T-1} < \cdots < s_1$; or the 'constant' schedule

$$s_t = 1 \,, \tag{15}$$

which uses constant parameters throughout the denoising process. Thus, the time-varying schedules can be computed as $\rho_t = s_t \bar{\rho}$ and $\mu_t = s_t \bar{\mu}$, and we only need to determine the average $\bar{\rho}$ and $\bar{\mu}$.

D Global Search of Denoising Diffusion Models

In this section, we provide details about the global search algorithms: BFS and DFS.

Algorithm 2 **T**raining-**F**ree **G**uidance

Input: Unconditional diffusion model $\boldsymbol{\epsilon}_\theta$, verifier f, guidance strength $\boldsymbol{\rho}, \boldsymbol{\mu}, \bar{\gamma}$, number of steps $T, N_{\text{recur}}, N_{\text{iter}}$
$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $t = T, \cdots, 1$ **do**
  Define function $\tilde{f}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} f(\boldsymbol{x} + \bar{\gamma}\sigma_t \boldsymbol{\delta})$
  for $r = 1, \cdots, N_{\text{recur}}$ **do**
    $\boldsymbol{x}_{0|t} = (\boldsymbol{x}_t - \sigma_t \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t))/\alpha_t$
    $\boldsymbol{\Delta}_{\text{var}} = \rho_t \nabla_{\boldsymbol{x}_t} \log \tilde{f}(\boldsymbol{x}_{0|t})$
    $\boldsymbol{\Delta}_{\text{mean}} = \boldsymbol{\Delta}_{\text{mean}} + \mu_t \alpha_t \nabla_{\boldsymbol{x}_{0|t}} \log \tilde{f}(\boldsymbol{x}_{0|t} + \boldsymbol{\Delta}_{\text{mean}})$ ▷ *Iterate N_{iter} times starting from $\boldsymbol{\Delta}_{mean} = \mathbf{0}$*
    $\boldsymbol{x}_{t-1} = \text{Sample}(\boldsymbol{x}_t, \boldsymbol{x}_{0|t}, t) + \frac{\alpha_{t-1}}{\alpha_t}(\boldsymbol{\Delta}_{\text{var}} + \boldsymbol{\Delta}_{\text{mean}})$ ▷ *Sample follows DDIM or DDPM*
    $\boldsymbol{x}_t \sim \mathcal{N}\left(\frac{\alpha_t}{\alpha_{t-1}}\boldsymbol{x}_{t-1}, \alpha_t^2\left(\frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_{t-1}^2}{\alpha_{t-1}^2}\right)\boldsymbol{I}\right)$ ▷ *Recurrent strategy*
  end for
end for
Output: Conditional sample $\boldsymbol{x}_0$

D.1 BFS-Based Search

We present the pseudo code for BFS in Alg. 3.

Here, we provide an overview of prior methods.

SVDD [39]. In SVDD, the best sample is selected at each time step, from which M children are generated. This approach can be viewed as a variant of BFS with $\tau = \infty$ and M particles. Nodes are evaluated using the current score $f(\boldsymbol{x}_{0|t})$.

TreeG [19]. In TreeG, particles are ranked and the top M are either selected directly or resampled based on their scores to obtain M samples. From each selected particle, K children are sampled, resulting in an effective tree width of KM. Particles are evaluated using their current score $f(\boldsymbol{x}_{0|t})$.

DAS [33]. In DAS, the authors propose an exponentially increasing tempering schedule as the default, given by $\tau_t = (1+\gamma)^{T-t} - 1$, and also introduce an adaptive tempering schedule. They adopt advanced SSP resampling instead of multinomial resampling, and evaluate particles based on the difference in rewards from the previous evaluation.

FK-steering [63]. In FK, the authors propose several options for evaluating intermediate particles, including difference, max, and sum, with *max* adopted as the default. In the official implementation, multinomial resampling is used, which may lead to suboptimal performance.

D.2 DFS-based search

In this section, we provide the details and pseudo code for DFS in Alg. 4. To better utilize previously explored sampling paths, we employ a buffer to store prior results. When no particles pass the threshold constraint, we retrieve the best sample from the buffer.

Similar to BFS, controlling the set of evaluation steps allows a trade-off between efficiency and accuracy. Evaluating at earlier time steps introduces higher uncertainty but enables backtracking. Additionally, adjusting the backtracking depth Δ_T governs the search scope: a small Δ_T reduces computation and favors local search, while a larger Δ_T enables broader exploration at the cost of increased computation.

In practice, we set the evaluation steps to $\mathcal{S} = \left\{\frac{1}{2}T, \frac{1}{4}T\right\}$ for image experiments to save compute, and to $\mathcal{S} = \left\{\frac{3}{4}T, \frac{3}{4}T - 1, \cdots, 1\right\}$ for PointMaze experiments. We set the recurrence depth to $T/2$ for image tasks and $T/4$ for PointMaze, corresponding to the denoised steps at which samples are first evaluated. The threshold schedule δ_t is also set to 'increase' as in Eq.14, enforcing tighter constraints for samples with lower noise.

In our experiments, we observed that when backtracking to $t_{\text{next}} = T$—thus fully restarting—the nonzero terminal SNR α_T/σ_T in many diffusion schedules [41] can lead to cumulative errors with

Algorithm 3 Diffusion BFS

Diffusion input: diffusion model ϵ_θ with diffusion time steps T and proposal transition kernel $\{\tilde{p}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\}_{t=1}^N$. Verifier f.
BFS input: Set of evaluation time steps $\mathcal{S}$. Tempering schedule τ_t. Budget of particles N. Scoring rule and Resampling function rule.
Init: Random sample N particles $\boldsymbol{x}_N^k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), k = 1, 2, \cdots N$.
for $t = T, \cdots, 1$ **do**
  if $t \in \mathcal{S}$ **then**
    for $k = 1, 2, \cdots, N$ **do**
      Estimate the conditional mean: $\boldsymbol{x}_{0|t}^k = \frac{\boldsymbol{x}_t^k - \sigma_t \epsilon_\theta(\boldsymbol{x}_t^k, t)}{\alpha_t}$. Compute the verifier score $f(\boldsymbol{x}_{0|t}^k)$.
      Compute the verifier scores according to the scoring rules: textbfCurrent: $\widehat{f}(\boldsymbol{x}_t^k) = \tau_t f(\boldsymbol{x}_{0|t}^k)$. **Difference**: $\widehat{f}(\boldsymbol{x}_t^k) = \tau_t f(\boldsymbol{x}_{0|t}^k) - \widehat{f}_{\text{prev}}^k$. **Max**: $\widehat{f}(\boldsymbol{x}_t^k) = \max\left(\tau_t f(\boldsymbol{x}_{0|t}^k), \widehat{f}_{\text{prev}}^k\right)$
      Resample the particles. Compute the weights $w_t^k = \text{softmax}(\widehat{f}(\boldsymbol{x}_t^k))$, and sample the children $n_t^k = \text{Resample}(N, w_t^k)$, where Resample can be **Multinomial** or **SSP**. Update the score buffers $\widehat{f}_{\text{prev}}^k = \tau_t f\left(\boldsymbol{x}_{0|t}^{\text{parent(k)}}\right)$.
    end for
  else
    $n_t^k = 1$
  end if
  for $k = 1, \cdots, N$ **do**
    Sample n_t^k particles from $\boldsymbol{x}_t^k$: $\boldsymbol{x}_{t-1}^j \sim \tilde{p}_\theta(\cdot|\boldsymbol{x}_t^k), j = 1, 2, \cdots, n_t^k$
  end for
end for
Return $\boldsymbol{x}_0 = \text{argmax}_{k=1,\cdots,K_0} f(\boldsymbol{x}_0^k)$

repeated backtracking. Therefore, when backtracking to $t_{\text{next}} = T$, we initialize with fresh Gaussian noise.

E Experiment Details

In this section we provide the details of experimental setup and implementation for all our experiments. We run our experiments on clusters with Nvidia A100 GPUs, with over 1000 GPU hours used.

E.1 Ablation of BFS design space

We directly adopt the official code base of FK-steering [63] and use the samping methods provided in the code base of DAS [33]. We use the ImageReward prompts as in [63] and report the average and standard deviation over 4 independent trials. For the temperature and resampling interval, we directly follow the implementation of FK-steering. For TreeG [19] we use a fixed branch out size of 2.

E.2 Text-to-Image Compositional Generation with DFS

We use the SSD-1B model[1] which is distilled from SDXL, and we use the default sampling configuration with 50 steps of DDIM sampler. For DFS and BFS, we evaluate at time steps $\{25, 35, 45\}$ and set the backtrack depth $\Delta_T = 25$. For BFS we additionally sweep the temperature in range $\{0.5, 1, 2, 4, 8\}$ and report the best performance.

E.3 Long Horizon Maze Planning

Maze environment. For all our maze experiments we use the OGBench PointMaze environment [55]. We created our maze layout using the same protocol of Figure 5 in [49][2], but with a smaller size

[1] https://huggingface.co/segmind/SSD-1B
[2] https://github.com/mpetersen94/gcs/blob/main/models/maze.py

Algorithm 4 Diffusion DFS

Diffusion input: diffusion model ϵ_θ with diffusion time steps T and proposal transition kernel $\{\tilde{p}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\}_{t=1}^N$. Verifier f.
DFS input Budget for total number of backtracking $B = K$, backtracking depth Δ_T and threshold $\{\delta_t\}_{t=1}^T$. Set of evaluation time steps $\mathcal{S}$.
Init $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, $t = T$. Init buffer with empty sets: $\texttt{buffer}(t) \leftarrow \{\}$, $t = 1, 2, \cdots, T$.
while $t > 0$ **do**
  if $t \in \mathcal{S}$ **then**
    Estimate the conditional mean and verifier score: $\boldsymbol{x}_{0|t} = \frac{\boldsymbol{x}_t - \sigma_t \epsilon_\theta(\boldsymbol{x}_t, t)}{\alpha_t}$, $f(\boldsymbol{x}_{0|t})$.
    if $f(\boldsymbol{x}_{0|t}) < \delta_t$ and budget $B > 0$ **then**
      Backtrack: $t_{\text{next}} \leftarrow \min(t + \Delta_T, T)$, $\boldsymbol{x}_{t_{\text{next}}} \sim q(\boldsymbol{x}_{t_{\text{next}}}|\boldsymbol{x}_t)$ with q in Eq. 1
      Decrease the budget: $B \leftarrow B - 1$
      Add the score-value pair to the buffer: $\texttt{buffer}(t).\texttt{add}\left(f(\boldsymbol{x}_{0|t}) : \boldsymbol{x}_t\right)$
    else
      if $B = 0$ **then**
        Pop the best sample from buffer: $\boldsymbol{x}_t \leftarrow \texttt{buffer}(t).\texttt{max}$ ▷ *select the best sample from past explorations*
      end if
      Sample posterior: $t_{\text{next}} \leftarrow t - 1$, $\boldsymbol{x}_{t_{\text{next}}} \sim \tilde{p}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$
    end if
  else
    Sample posterior: $t_{\text{next}} \leftarrow t - 1$, $\boldsymbol{x}_{t_{\text{next}}} \sim \tilde{p}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$
  end if
  $t \leftarrow t_{\text{next}}$, $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{\text{next}}$
end while
Return $\boldsymbol{x}_0$

of 20x20 cells. Dataset is collected following the protocal in OGBench [55]. We evaluate the model on the default task 1 of OGBench [55], which is navigating from bottom left to top right. Empirically we discover that the diffusion model can perform well on short-horizon tasks without extra inference compute, but struggles heavily in the long horizon tasks.

Model Training. We train the model following diffuser [31], where we use a temporal U-Net to denoise the trajectory

$$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{s}_1 & \boldsymbol{s}_2 & \cdots & \boldsymbol{s}_H \\ \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_H \end{bmatrix} .$$

Since our objective start and goal is more distant than trajectories in dataset, we sample at longer horizons than training, which is enabled by the temporal U-Net architecture. We train the model for 1.2M steps using the same configuration as [31].

Inference. We found that the model performance saturates with 16 denoising steps, which we use for all our experiments. For all the data points we report the average success rate with over 40 samples.

For verifier design, we use the ground-truth maze layout, and calculate the violation of each point in the trajectory using the position coordinates. Specifically, if a point (x, y) is inside a maze wall box with center (c_x, c_y) and half-width d, then the point loss can be calculated as the minimum distance from the point to box walls:

$$L(x, y) = \min\left(x - (c_x - d), (c_x + d) - x, y - (c_y - d), (c_y + d) - y\right) .$$

and the total verifier score is computed as:

$$f(\boldsymbol{\tau}) = \exp\left(-\sum_{i=1}^{H} L(x_i, y_i)^2\right) .$$

So if all the points are free of violation in the trajectory, then $f(\boldsymbol{\tau}) = 1$. We point out that this does not indicate a successful plan as the connection between consecutive points $(x_i, y_i) \to (x_{i+1}, y_{i+1})$ may violate the maze layout, and using only the verifier function can not generate a successful plan.

For local search, we search the hyper-parameters $\bar{\rho}$ and $\bar{\mu}$ in Sec. C.5 with $\bar{\gamma} = 0$. For global search with BFS we evaluate at steps $\{12, 8, 4\}$, and for DFS we evaluate at $\{12, 11, \cdots, 1\}$ with backtracking depth $\Delta_T = 4$. We also observe that increasing backtracking depth to 12 and evaluate at smaller time steps $\{4, 3, 2, 1\}$ helps to scale up the performance with more compute. The hyperparameter search results are below:

N	$\tau = 0.2$	$\tau = 0.005$	$\tau = 0.1$
2	27.5 ± 4.3	32.5 ± 1.1	31.2 ± 4.2
4	42.5 ± 5.2	48.1 ± 1.1	45.5 ± 2.3
8	67.6 ± 1.1	71.2 ± 2.2	70.1 ± 1.1

Table 4: Hyperparameter search for temperature τ in PointMaze BFS

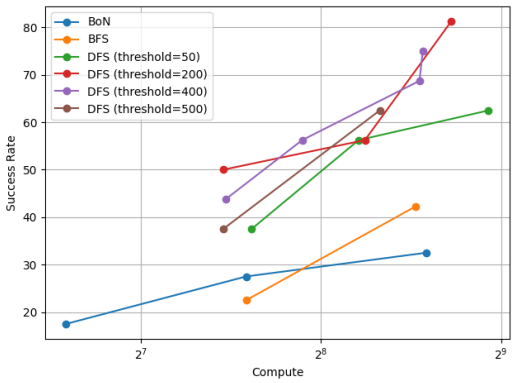


Figure 5: Hyperparameter search for threshold δ in PointMaze DFS

E.4 Offline RL

Background. Diffusion policy [8] is widely used for action generation in robot foundation models [4, 44]. At inference time, policies can be guided by human trajectory constraints [81] or LLM-based value functions [51]. Exact sampling requires training a noise-dependent energy function [45], but this can degrade pretrained knowledge and demands additional data—often impractical in data-scarce robotic settings. In contrast, inference-scaling provides a more flexible approach, allowing seamless composition of pretrained diffusion policies with Q-functions without retraining.

Setup. We follow the setup in [45], and we directly use their pretrained diffusion model and Q-function, omitting the time-dependent energy function. The diffusion model was trained to generate action $\boldsymbol{a}$ given state $\boldsymbol{s}$, and we sample with 15 steps of DDIM.

For hyper-parameter search, we disable the implicit dynamics and set $\bar{\gamma} = 0$, and use the 'increase' schedule for $\boldsymbol{\rho}$ and $\boldsymbol{\mu}$. For strength parameters $\bar{\rho}$ and $\bar{\mu}$, we first search for the right magnitude. Then, we also follow [45] and search with step size [1,2,3,5,8,10] within the magnitude. Same as [45], we use 5 different seeds with 10 samples per seed for each task. To avoid over fitting, we use different seeds for parameter search and evaluation. We report the hyper-parameters and the performance within the parameter-searching dataset and evaluation dataset.

For global search, we use 4 particles for Medium-Expert and Medium datasets, and 2 particles for Medium-Replay datasets. Since the number of particles are small, we do not carry out BFS or DFS methods and simply use Best-of-N. We point out that the number of particles we use are much smaller than the 50 particles in [81] and the 32 particles in [6], highlighting the effectiveness of local search.

Baseline. We compare our method to a variety of baselines, including traditional state-of-the-art methods IQL [35] and diffusion-based policies such as diffuser [31], decision-diffuser (DD) [1], Diffusion-QL (D-QL) [81], SfBC [6] and QGPO [45]. We directly take the numbers from [45].

Among the baseline diffusion-based methods, both Diffuser [31] and QGPO [45] requires training a noise-dependent guidance function, and D-QL [81] requires updating the diffusion model during training using the Q-function iteratively, which needs to back-propagate through the diffusion sampling chain, introducing high computation and memory overheads. DD [1] uses classifier-free guidance [27] to generate high-return trajectories that requires training a return-conditional model on labeled datasets, which can be expensive to obtain in robotics where only demonstration data is available [44].

For our reproduced baselines, TFG [94] is allowed up to 8 recurrence steps and DAS [33] up to 16 particles, resulting in a hyperparameter space and computational cost approximately twice that of our method. We sweep across all configurations for the baseline methods and report the best performance. For fair comparison we evaluate our method on different seeds used for hyperparameter search, with the results shown in Table. 5.

Dataset	**Environment**	particles	N_{recur}	N_{iter}	$\bar{\rho}$	$\bar{\mu}$	**Eval set**	**Search set**
Medium-Expert	HalfCheetah	4	1	1	0.008	0.02	93.9 ± 0.3	$\mathbf{94.3 \pm 0.5}$
Medium-Expert	Hopper	4	1	4	0.001	0.00	104.4 ± 3.1	$\mathbf{109.4 \pm 5.2}$
Medium-Expert	Walker2d	4	1	1	0.005	0.10	$\mathbf{111.4 \pm 0.1}$	111.4 ± 0.2
Medium	HalfCheetah	4	1	4	0.003	0.05	$\mathbf{54.8 \pm 0.1}$	54.8 ± 0.2
Medium	Hopper	4	4	4	0.003	0.02	99.5 ± 1.7	$\mathbf{100.1 \pm 0.1}$
Medium	Walker2d	4	1	6	0.003	0.08	$\mathbf{86.5 \pm 0.2}$	85.2 ± 3.2
Medium-Replay	HalfCheetah	2	1	6	0.005	0.03	47.8 ± 0.4	$\mathbf{48.4 \pm 0.1}$
Medium-Replay	Hopper	2	1	1	0.003	0.20	97.4 ± 4.0	$\mathbf{100.4 \pm 2.2}$
Medium-Replay	Walker2d	2	2	4	0.003	0.03	79.3 ± 9.7	$\mathbf{83.2 \pm 2.8}$
Average							86.1	**87.5**

Table 5: Hyper-parameters on D4RL locomotion tasks with test-time scaling. We report the performance on hyper-parameter search dataset and the evaluation dataset, highlighting the best number.

F Mitigating reward hacking with double verifier

In this section, we show that reward hacking caused by adversarial examples can be mitigated by employing separate verifiers for local and global search. As observed in [61], training-free guidance with verifier gradients is vulnerable to adversarial examples: generated samples can exploit weaknesses in the verifier, causing it to classify them as belonging to the target class despite being out-of-distribution (OOD). We find, however, that such adversarial examples do not transfer well between independently trained verifiers. Inspired by double-Q learning in reinforcement learning [76], we propose a *double-verifier* approach, assigning distinct verifiers to local and global search to efficiently detect and reject adversarial samples.

We evaluate the proposed double-verifier on the challenging conditional ImageNet generation task, generating target-class samples from an unconditional model guided by a pretrained classifier. Specifically, we use two independent classifiers as verifiers[3][4] for global and local search. We report the Fréchet Inception Distance (FID) computed on 256 generated samples against the corresponding ImageNet class, and measure class accuracy using a separate classifier[5]. Since we only apply the global verifier sparsely, double-verfier introduces negligible computational costs.

[3] https://huggingface.co/google/vit-base-patch16-224
[4] https://huggingface.co/google/vit-base-patch16-384
[5] https://huggingface.co/facebook/deit-small-patch16-224

Table 6: Best-of-N results for ImageNet conditional generation, with FID and Accuracy averaged across the two labels.

#Particles	BoN-Single		BoN-Double		BFS-Single		BFS-Double	
	FID↓	Acc↑	FID↓	Acc↑	FID↓	Acc↑	FID↓	Acc↑
4	171.5	31.8%	151.2	37.5%	156.2	36.1%	**145.5**	**44.3**%
8	155.7	35.8%	127.8	49.2%	133.3	46.5%	**118.2**	**55.9**%

#Particles	BoN-Single	BoN-Double
4	0.161	**0.164**
8	0.165	**0.184**

Table 7: MSP scores of Best-of-N with single and double verifier. Double verifier significantly reduces OOD samples with higher MSP score.

As shown in Table. 6, using double-verifier significantly improves performance over single verifier with Best-of-N and BFS, using 2x less compute. We also evaluate the OOD of generated samples using the MSP score [25], with higher MSP score indicating less OOD samples. As shown in Table. 7, using double-verifier significantly reduces OOD samples indicated by the higher MSP score.

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 reflect the paper's contribution within inference-time scaling of diffusion models

   Guidelines:

   - The answer NA 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 NA 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: In the conclusions section we discussed the limitations of our methods.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: We provide complete proof in the Appendix

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide detailed pseudo code and hyper-parameters in the appendix

   Guidelines:

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

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

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

Answer: [No]

Justification: We run our experiments with publicly available models and dataset.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: We provide all the experimental details in the appendix

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: We provide detailed numbers with standard deviation in the locomotion setting

Guidelines:

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

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: we report the compute resources in the Appendix experiments section

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: the research follows NIPS code of ethics

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: inference-time search with verifiers could potentially be used for verifier hacking against classifiers

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: the paper uses pretrained models, with safeguards already in place.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: we provide citation for all resources used

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: the paper does not release new assets

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: the paper does not involve research with human objects

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- The answer NA 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 (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.