

Best-of- N Guidance for Test-time Diffusion Alignment

Richard Lee Kim¹ Yeongmin Kim¹ Gyuwon Sim¹ Taekyu Kim¹
Minsang Park¹ Il-Chul Moon^{1,2}

¹KAIST, ²summary.ai

{swimskykim, alsdudrla10, gkwllaks4886, taekyu, pagemu, icmoon}@kaist.ac.kr

Abstract

Diffusion models achieve strong generative performance but often struggle to align generated samples with human preferences measured by a reward model. A simple yet effective algorithm for test-time alignment is Best-of- N (BoN) sampling, which draws N i.i.d. samples from a pre-trained diffusion model and outputs the single highest-reward sample. Despite its empirical success, BoN makes limited use of reward information, as it is incorporated only at the final selection stage without influencing the reverse diffusion trajectory during sampling. Consequently, BoN sampling does not improve the average alignment of generated samples and is primarily suited to single-output settings. We propose Best-of- N Guidance (BoNG), a novel method that integrates the principle of BoN sampling directly into the reverse diffusion process. BoNG performs online BoN selection over denoising particles and adjusts the reverse diffusion process to steer the particle population toward higher-reward regions during generation. Specifically, by introducing an asymmetric guidance interaction among denoising particles, BoNG uses the current BoN particle as a guidance signal to the rest of the particle population. This particle-level interaction reshapes the sampling process toward higher-reward regions, enabling BoNG to improve not only the final best sample beyond Vanilla BoN sampling, but also the average quality of generated samples. Over 36 empirical comparisons, BoNG achieves the best performance in 29 cases, ranking first in 80.56% of the comparisons against SMC and Vanilla BoN sampling. BoNG also supports multi-output capability, achieving $1.3\times$ ImageReward score of the latest sample-based guidance method with a $1.6\times$ speedup. We release the code at <https://github.com/aailab-kaist/BoNG>.

1 Introduction

Diffusion models [27, 8, 29] demonstrate strong generative capabilities, leading to a growing number of real-world applications that produce outputs conditioned on user inputs [17, 35]. However, the generated outputs still do not fully align with user preferences in terms of input alignment and sample fidelity. To address this issue, reward functions [36, 34, 16] that evaluate alignment with user conditions and output quality have been incorporated into the generation process. Recently, test-time alignment methods have shown superior performance compared to parameter-scaling [19, 14] or fine-tuning approaches [3, 33], attracting increasing research interest in this direction [26, 24, 13].

Best-of- N (BoN) sampling and Sequential Monte Carlo (SMC) [24, 15, 25] are widely used test-time alignment methods for diffusion models. However, both approaches have inherent limitations: BoN selects only a single sample, while SMC is prone to particle collapse. As a result, they are primarily suited to scenarios where only one sample is ultimately served. In contrast, guidance-based methods enable the generation of diverse samples by independently adjusting the denoising direction for each sample, making them applicable to settings that require multiple outputs. However, early approaches,

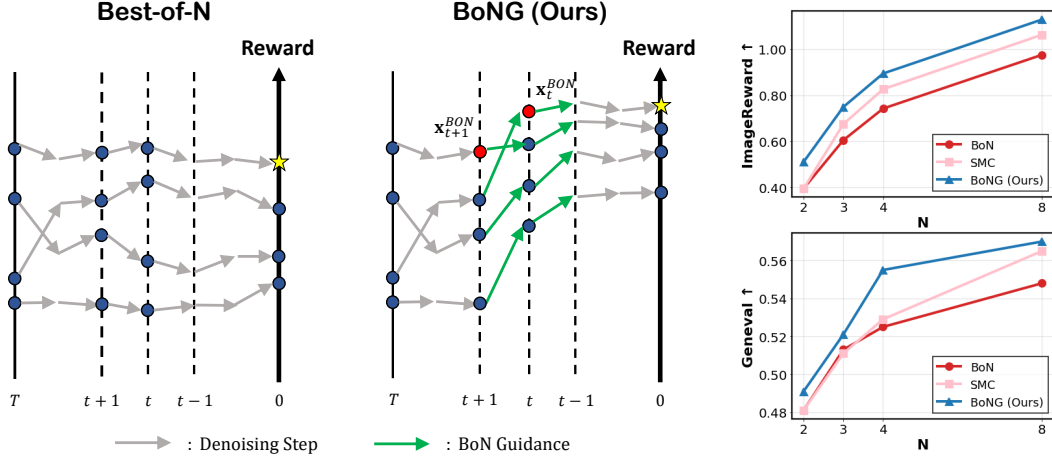


Figure 1: **Best-of- N Guidance (BoNG)** (Left) Vanilla BoN sampling selects the highest-reward sample only after generation. (Middle) BoNG performs online BoN selection (Algorithm 1) and uses the selected sample to guide the remaining particles (Algorithm 2). (Right) BoNG consistently improves single-output performance over Vanilla BoN and SMC across various sample budgets.

such as gradient guidance [12, 2, 18], require neural backward passes with differentiable rewards, leading to substantial computational overhead and are prone to reward hacking.

Sample-based guidance is emerging as a promising direction, as it can generate diverse samples without requiring neural backward passes. However, it often becomes a computational bottleneck, since iterative rollout sampling requires generating final samples at each time step. Consequently, this line of research has focused on reducing the cost of rollout [20, 10, 21, 13], and has recently achieved promising results in both performance and efficiency [13]. However, it still incurs rollout cost, resulting in 2–3 $\times$ higher sampling cost compared to vanilla sampling.

We propose *Best-of- N Guidance (BoNG)*, a rollout-free, sample-based guidance method inspired by BoN sampling. Unlike Vanilla BoN which uses the reward only once to select the best sample after generation, BoNG performs this selection online and injects it into the reverse diffusion process as a guidance signal (Figure 1). Specifically, we replace the iterative rollout process by selecting the highest-reward expected denoised sample among multiple particles at several intermediate timesteps. This approach steers all samples toward the direction of the highest-reward sample without incurring additional feed-forward computations of the score network. We theoretically analyze its connection to the reward-tilted distribution targeted by prior work, and empirically demonstrate gains over Vanilla BoN, SMC, and prior guidance methods with minimal additional inference cost.

2 Related Work

BoN sampling and approaches based on reward-tilted distributions have been widely studied in test-time diffusion alignment. Given a pre-trained diffusion model $p_\theta(\mathbf{x}_0)$ and a human reward function $r(\mathbf{x}_0)$, the reward-tilted distribution is defined as $p_\theta^r(\mathbf{x}_0) \propto p_\theta(\mathbf{x}_0) \exp(r(\mathbf{x}_0))$. To approximate this distribution, previous work has studied how to estimate the intractable expected future reward $r_t(\mathbf{x}_t) := \mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t)}[\exp(r(\mathbf{x}_0))]$ for a noisy sample $\mathbf{x}_t$, which induces the intermediate distribution $p_\theta^r(\mathbf{x}_t) \propto p_\theta(\mathbf{x}_t) r_t(\mathbf{x}_t)$ [31, 5]. SMC methods [24, 15, 25] generate multiple samples from the base diffusion model and perform resampling based on intermediate reward values. However, SMC suffers from particle collapse, which degrades the quality of the best sample in single-output generation and limits sample diversity in multi-output settings. Gradient-guidance methods address this issue by enabling i.i.d. sampling by approximating the target time-dependent score function. However, these methods require either training diffusion time-conditioned reward models [12] or performing multiple neural backward passes with Taylor approximation [2, 18] at every denoising step, resulting in significant latency and memory overhead.

Sample-based guidance methods [20, 13] have recently emerged as a promising direction because they avoid particle collapse and do not require neural backward propagation for guidance. Tilt [20] proposes a gradient-free method for incorporating expected future rewards into the reverse process. However, it remains computationally expensive because it requires generating multiple samples from $\mathbf{x}_t$ to $\mathbf{x}_0$ at every timestep t , known as backward rollout. GlassFlow [10] improves efficiency by reducing the number of rollout steps through ODE-based sampling, and MFM [21] further accelerates this process by replacing the ODE solver with a distilled model. LiDAR [13] proposes a forward rollout strategy for estimating intermediate rewards. This reduces per-step rollout sampling to a single marginal rollout. LiDAR significantly improves sample-based guidance and achieves state-of-the-art performance while being around 10× faster than gradient-guidance methods. However, it still requires additional lookahead sampling, making it 2–3× slower than vanilla diffusion sampling. The goal of this work is to develop an efficient and effective sample-based guidance method inspired by the idea of BoN sampling.

3 Preliminary

3.1 Diffusion Models

A diffusion model [27, 8, 29] consists of a pair of stochastic processes: a forward process and a reverse process. The forward diffusion process injects noise into the data $\mathbf{x}_0 \sim p(\mathbf{x}_0)$ to produce a perturbed sample $\mathbf{x}_t$ via a Gaussian kernel:

$$p(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \mathbf{x}_0, \sigma_t^2 \mathbf{I}),$$

for every $t \in [0, T]$, with a corresponding gradually increasing noise scale σ_t . This perturbation produces the marginal distribution $p(\mathbf{x}_t)$, and the terminal noise scale σ_T is chosen to be sufficiently large so that $p(\mathbf{x}_T)$ is close to a pure Gaussian prior.

The reverse process generates a data $\mathbf{x}_0$ from a noise $\mathbf{x}_T \sim p(\mathbf{x}_T)$, and is often represented as a stochastic differential equation [1, 29]:

$$d\mathbf{x}_t = [\mathbf{f}(\mathbf{x}_t, t) - g^2(t) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)] dt + g(t) d\mathbf{w}_t, \quad (1)$$

where the drift $\mathbf{f}$ and volatility g terms are defined to correspond to the noise schedule σ_t , and $\mathbf{w}_t$ is a standard Wiener process. The diffusion model constructs the reverse process by estimating the score function $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \approx \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t)$ using a score model trained with denoising score matching [32, 28] or noise prediction [8] loss functions.

3.2 Test-time Human Alignment

Problem Setup: With the advancement of diffusion models, many pre-trained models are now available as open-source [22, 19, 14], enabling generation conditioned on user input $\mathbf{c}$. However, the final generated sample $\mathbf{x}_0$ often does not align well with the user input $\mathbf{c}$. Accordingly, we assume access to a pre-trained diffusion model $p_\theta(\mathbf{x}_0 | \mathbf{c})$ and a human preference reward function $r : \mathcal{X}_0 \times \mathcal{C} \rightarrow \mathbb{R}$, and aim to obtain samples $\mathbf{x}_0$ such that reward $r(\mathbf{x}_0, \mathbf{c})$ is high, by manipulating the inference process. Here, $\mathcal{X}_0$ denotes the data space at time $t = 0$, and $\mathcal{C}$ denotes the user’s input condition space.

3.2.1 Best-of- N Sampling

Best-of- N (BoN) sampling is a post-hoc selection procedure that draws N i.i.d. samples from the base distribution $p_\theta(\mathbf{x}_0 | \mathbf{c})$ and returns a single sample with the highest reward. In finite N , the BoN sample from the base distribution is defined as:

$$\mathbf{x}_0^{\text{BoN}} := \operatorname{argmax}_{\{\mathbf{x}_0^i\}_{i=1}^N} r(\mathbf{x}_0^i, \mathbf{c}), \quad \mathbf{x}_0^i \stackrel{\text{i.i.d.}}{\sim} p_\theta(\mathbf{x}_0 | \mathbf{c}) \quad (2)$$

BoN sampling is a remarkably strong inference-time alignment strategy in practice [11], since it can substantially improve the quality of the final selected sample by increasing N , without modifying the underlying generative model. However, it remains fundamentally inefficient: the sampling process is done by Eq. (1), reward information is not used during the generation process. As a result, obtaining a sample from a high-reward region may require large N . Moreover, BoN sampling is inherently designed to return a single best output, and therefore does not directly address the multi-output setting [17, 35] in which all N generated candidates are expected to be of high quality.

3.2.2 Sampling from a Reward-tilted Distribution

A line of research [24, 10, 13] aims to sample from the reward-tilted distribution for alignment defined as:

$$p_\theta^r(\mathbf{x}_0|\mathbf{c}) \propto p_\theta(\mathbf{x}_0|\mathbf{c}) \exp(\lambda r(\mathbf{x}_0, \mathbf{c})), \quad (3)$$

where λ is a scalar hyperparameter that controls the strength of the reward. However, since this goal is defined on the final generated sample $\mathbf{x}_0$, it is challenging to determine how to modify the iterative denoising process in Eq. (1).

We follow a popular intermediate formulation from prior work [31, 5, 10] and adopt the score-based formulation of [13] as:

$$\nabla_{\mathbf{x}_t} \log p_\theta^r(\mathbf{x}_t|\mathbf{c}) = \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{c}) + \nabla_{\mathbf{x}_t} r_t^\lambda(\mathbf{x}_t, \mathbf{c}), \quad (4)$$

where $r_t^\lambda(\mathbf{x}_t, \mathbf{c}) := \log \mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})} [\exp(\lambda r(\mathbf{x}_0, \mathbf{c}))]$ is expected future reward at particle $\mathbf{x}_t$. Denoising via Eq. (1) with the score replaced by Eq. (4) yields samples from the reward-tilted distribution in Eq. (3). Note that computing r_t^λ is generally intractable, so most previous work focuses on how to approximate this term efficiently and accurately. Although this reverse-process modification is well suited for multi-output generation, existing approximations typically require costly gradient computation [2, 18] or rollout sampling [20, 13], making them less efficient for test-time alignment.

4 Method

To mitigate the inefficiency and limited capability in multi-output settings, we refine the idea of Best-of- N sampling as a diffusion-based guidance scheme. A natural approach is to inject the reward information of $\mathbf{x}_0^{\text{BoN}}$ in Eq. (2) directly into the reverse diffusion process, but doing so requires samples from $\mathcal{X}_0$. Following prior works [2, 18, 24], we use one-step denoised samples as an alternative to approximate $\mathbf{x}_0^{\text{BoN}}$, and guide the particle population towards it. We construct our Best-of- N guidance algorithm in two steps.

4.1 Best-of- N Guidance (BoNG)

Step 1: Best-of- N target in $\mathcal{X}_0$. Let $\{\mathbf{x}_t^i\}_{i=1}^N$ denote the current particle population. For each particle, we compute its one-step denoised sample

$$\hat{\mathbf{x}}_0^i := \mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t^i, \mathbf{c})}[\mathbf{x}_0] = \mathbf{x}_t^i + \sigma_t^2 \nabla_{\mathbf{x}_t^i} \log p_\theta(\mathbf{x}_t^i | \mathbf{c}). \quad (5)$$

These one-step denoised samples, $\{\hat{\mathbf{x}}_0^i\}_{i=1}^N$, define a candidate set in $\mathcal{X}_0$. We then perform Best-of- N selection over this set and define

$$i^* = \underset{i}{\operatorname{argmax}} \quad r(\hat{\mathbf{x}}_0^i, \mathbf{c}), \quad \hat{\mathbf{x}}_0^{\text{BoN}} := \hat{\mathbf{x}}_0^{i^*}, \quad \mathbf{x}_t^{\text{BoN}} := \mathbf{x}_t^{i^*}. \quad (6)$$

Step 2: Adjust each particle toward $\hat{\mathbf{x}}_0^{\text{BoN}}$. We define the **BoN target score** for each particle so that its one-step denoised sample matches $\hat{\mathbf{x}}_0^{\text{BoN}}$. Comparing this target with the one-step denoised sample in Eq. (5), we obtain

$$\underbrace{\nabla_{\mathbf{x}_t^i} \log p_\theta^{\text{BoN}}(\mathbf{x}_t^i | \mathbf{c})}_{\text{BoN target score}} := \nabla_{\mathbf{x}_t^i} \log p_\theta(\mathbf{x}_t^i | \mathbf{c}) + \underbrace{\frac{1}{\sigma_t^2} (\hat{\mathbf{x}}_0^{\text{BoN}} - \hat{\mathbf{x}}_0^i)}_{\substack{:= \nabla_{\mathbf{x}_t^i} r_t^{\text{BoN}}(\mathbf{x}_t^i, \mathbf{c}) \text{ (BoN guidance term)}}}. \quad (7)$$

The Best-of- N Guidance (BoNG) method is defined by using the **BoN guidance term** to construct the **BoN target score** at each timestep t . See Algorithms 1 and 2.

The BoN target score $\nabla_{\mathbf{x}_t^i} \log p_\theta^{\text{BoN}}(\mathbf{x}_t^i | \mathbf{c})$ in Eq. (7) admits a simple interpretation. For $\mathbf{x}_t^{\text{BoN}}$, the BoN guidance term $\nabla_{\mathbf{x}_t^i} r_t^{\text{BoN}}(\mathbf{x}_t^i, \mathbf{c})$ vanishes since $(\hat{\mathbf{x}}_0^{\text{BoN}} - \hat{\mathbf{x}}_0^{\text{BoN}})/\sigma_t^2 = 0$, and thus $\mathbf{x}_t^{\text{BoN}}$ is denoised according to the original reverse diffusion process without additional guidance.

Algorithm 1 Best-of- N selection

Input: $\{\mathbf{x}_t^i\}_{i=1}^N, t, \mathbf{c}, p_\theta, r$
1. **for** $i = 1, \dots, N$ **do**
 $\hat{\mathbf{x}}_0^i \leftarrow \mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t^i, \mathbf{c})}[\mathbf{x}_0]$
end for
2. Select i^* according to Eq. (6)
3. $\mathbf{x}_t^{\text{BoN}} \leftarrow \mathbf{x}_t^{i^*}, \hat{\mathbf{x}}_0^{\text{BoN}} \leftarrow \hat{\mathbf{x}}_0^{i^*}$
Output: $\mathbf{x}_t^{\text{BoN}}, \hat{\mathbf{x}}_0^{\text{BoN}}, \{\hat{\mathbf{x}}_0^i\}_{i=1}^N$

Algorithm 2 One-step BoNG update

Input: $\{\mathbf{x}_t^i\}_{i=1}^N, t, \mathbf{c}, p_\theta, r, \sigma_t$
1. Obtain $\mathbf{x}_t^{\text{BoN}}, \hat{\mathbf{x}}_0^{\text{BoN}}, \{\hat{\mathbf{x}}_0^i\}_{i=1}^N$ from Algorithm 1
2. **for** $i = 1, \dots, N$ **do**
Update $\mathbf{x}_t^i$ to $\mathbf{x}_{t-1}^i$ using Eq. (7)
end for
Output: $\{\mathbf{x}_{t-1}^i\}_{i=1}^N$

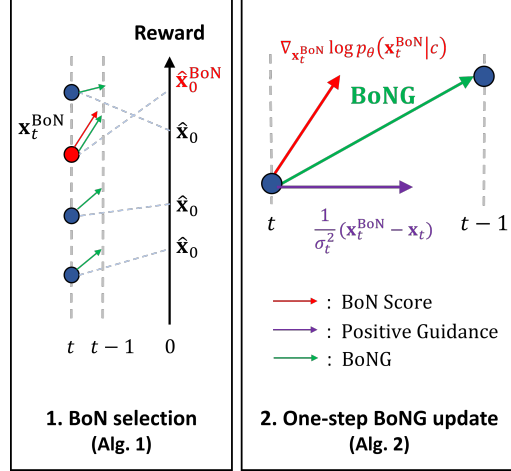


Figure 2: **Dynamics of BoNG.** (Left) Algorithm Summary. (Right) Decomposition of BoN target score in Eq. (7).

For any particle $\mathbf{x}_t^i$ with $i \neq i^*$, the update can be interpreted as the sum of two terms:

$$\nabla_{\mathbf{x}_t^i} \log p_\theta^{\text{BoN}}(\mathbf{x}_t^i | \mathbf{c}) = \underbrace{\nabla_{\mathbf{x}_t^{\text{BoN}}} \log p_\theta(\mathbf{x}_t^{\text{BoN}} | \mathbf{c})}_{\text{BoN Score}} + \underbrace{\frac{1}{\sigma_t^2} (\mathbf{x}_t^{\text{BoN}} - \mathbf{x}_t^i)}_{\text{Positive Guidance}}. \quad (8)$$

This decomposition shows that BoNG induces an asymmetric interaction among particles: each particle $\mathbf{x}_t^i$ with $i \neq i^*$ is attracted toward the current BoN particle by positive guidance, while simultaneously following the score-induced denoising direction of the BoN trajectory by BoN score. Overall, BoNG embeds the strong selection behavior of Best-of- N sampling via a guidance term without additional feed-forward computations of the score network.

4.2 Connection to Prior Sample-based Test-time Alignment Methods

We now connect BoNG to prior alignment methods that sample from the reward-tilted distribution in Eq. (3). Lemma 4.1 and Lemma 4.2 provide a general analysis of the reward-tilted distribution that holds for any $\lambda > 0$: they reduce reward guidance to estimating the tilted posterior mean and give its self-normalized importance sampling (SNIS) estimator. Theorem 4.3 then connects this general analysis to BoNG, showing that hard Best-of- N selection realizes the large- λ limit of the SNIS estimator, with Algorithms 1 and 2 stating the resulting procedure.

Lemma 4.1. *The gradient of the expected future reward can be expressed as*

$$\nabla_{\mathbf{x}_t} r_t^\lambda(\mathbf{x}_t, \mathbf{c}) = \frac{1}{\sigma_t^2} (m_\lambda^r(\mathbf{x}_t) - \mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0]), \quad m_\lambda^r(\mathbf{x}_t) := \mathbb{E}_{p_\theta^r(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0]. \quad (9)$$

Lemma 4.1 follows from the Fisher identity and related score-based formulations [13, 20]; its role is preparatory. It shows that reward guidance requires no reward derivatives and reduces to estimating the reward-tilted posterior mean $m_\lambda^r(\mathbf{x}_t)$. Derivation is in Section A.1.

We now provide a Monte Carlo approximation scheme of $m_\lambda^r(\mathbf{x}_t)$ via self-normalized importance sampling [30] (SNIS) by the following lemma, which holds for any $\lambda > 0$.

Lemma 4.2. *Let $q(\mathbf{x}_0)$ be any proposal distribution whose support is $\mathcal{X}_0$. Then, given proposal samples $\hat{\mathbf{x}}_0^1, \dots, \hat{\mathbf{x}}_0^N \stackrel{i.i.d.}{\sim} q(\mathbf{x}_0)$, the reward-tilted posterior mean $m_\lambda^r(\mathbf{x}_t)$ admits the self-normalized importance sampling approximation*

$$m_\lambda^r(\mathbf{x}_t) \approx \hat{m}_{N, \lambda}^r(\mathbf{x}_t) := \sum_{i=1}^N \alpha_i^{(\lambda)} \hat{\mathbf{x}}_0^i, \quad (10)$$

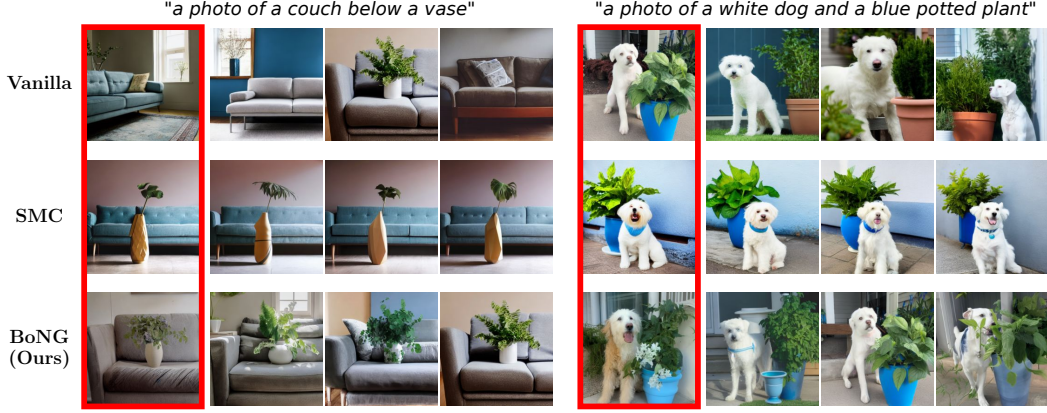


Figure 3: Qualitative comparison of generated samples from Vanilla, SMC, and BoNG (SD v1.5 DDPM 100 step sampler). Red boxes denote Best-of- N samples for each method. Vanilla sampling often fails to generate images which align well with specific prompts such as “below a vase” and “a blue potted plant”. SMC generates better aligned images, but resampling leads to premature particle collapse. In contrast, BoNG generates samples that remain well aligned with the prompt while preserving diverse layouts and appearances.

where

$$\alpha_i^{(\lambda)} = \text{softmax}_i \left(\log p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c}) - \log q(\hat{\mathbf{x}}_0^i) + \lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c}) \right).$$

Proof is in Section A.2. Lemma 4.2 provides a unified view of prior sample-based guidance methods: LiDAR [13] corresponds to the choice $q(\mathbf{x}_0) = p_\theta(\mathbf{x}_0 | \mathbf{c})$, while Tilt [20] uses $q(\mathbf{x}_0) = p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})$. Although these methods differ in their choice of proposal distribution, both approximate $m_\lambda^r(\mathbf{x}_t)$ through weighted proposal samples. As $\lambda \rightarrow \infty$, the softmax weights concentrate on the highest-reward proposal sample, with $\alpha_i^{(\lambda)} \rightarrow 1$ for the maximizer and $\alpha_i^{(\lambda)} \rightarrow 0$ otherwise, so the SNIS estimator reduces to the highest-reward sample in the proposal set. The following theorem shows that, in this large- λ limit, the process target score in Eq. (4) coincides with the BoN target score of BoNG in Eq. (7) when the proposal set is given by the one-step denoised samples $\{\hat{\mathbf{x}}_0^i\}_{i=1}^N$ with $\hat{\mathbf{x}}_0^i = \mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t^i, \mathbf{c})}[\mathbf{x}_0]$ in Eq. (5).

Theorem 4.3. *Let $\{\hat{\mathbf{x}}_0^i\}_{i=1}^N$ be the proposal set of one-step denoised samples in Eq. (5) with a unique maximizer $i^* = \arg \max_{1 \leq i \leq N} r(\hat{\mathbf{x}}_0^i, \mathbf{c})$, and let $\hat{\mathbf{x}}_0^{\text{BoN}} := \hat{\mathbf{x}}_0^{i^*}$. Then $\hat{m}_{N,\lambda}^r(\mathbf{x}_t^i) \rightarrow \hat{\mathbf{x}}_0^{\text{BoN}}$ as $\lambda \rightarrow \infty$, and the SNIS approximation of the reward-guidance gradient in Eq. (9) converges to the BoN guidance term for each particle:*

$$\nabla_{\mathbf{x}_t^i} r_t^\lambda(\mathbf{x}_t^i, \mathbf{c}) \approx \frac{1}{\sigma_t^2} (\hat{m}_{N,\lambda}^r(\mathbf{x}_t^i) - \hat{\mathbf{x}}_0^i) \xrightarrow{\lambda \rightarrow \infty} \nabla_{\mathbf{x}_t^i} r_t^{\text{BoN}}(\mathbf{x}_t^i, \mathbf{c}). \quad (11)$$

Proof is in Section A.3. Theorem 4.3 shows that BoNG realizes the large- λ limit of sample-based guidance toward the reward-tilted distribution, while eliminating the costly forward or backward rollouts of prior methods. Here, the one-step Tweedie estimates serve as a surrogate proposal set rather than exact i.i.d. proposal samples. The same construction also admits a finite- λ relaxation: replacing the hard selection with the soft SNIS weights $\alpha_i^{(\lambda)}$ of Lemma 4.2 yields a guidance target that interpolates between the average of the proposal set and the BoN target, so that λ directly controls the strength of reward tilting. We refer to Section E for the full derivation, the approximation gaps of this soft variant, and a λ sweep verifying its predicted reward-diversity scaling.

4.3 Instantiation of BoNG

BoNG can be applied flexibly during inference time, which allows the method to interpolate between aggressive reward optimization and diversity preservation. Similarly to SMC [24], we instantiate BoNG by specifying a guidance timestep window, i.e. the subset of reverse diffusion steps at which the BoN guidance term is activated. The guidance window determines how often the particle

Table 1: Best-of- N (BoN) performance on GenEval prompts with different numbers of generated images N , using SD v1.5 [22] and SDXL [19] backbones. ImageReward (IR) is used as the reward for all methods. **Bold** entries denote the top-performing results within each backbone group.

Backbone	Method	Performance ($\uparrow$)								
		$N = 2$			$N = 3$			$N = 4$		
		IR	CLIP	GenEval	IR	CLIP	GenEval	IR	CLIP	GenEval
SD v1.5 (0.9B) (w/ DDIM 50 steps)	Vanilla	0.317	0.277	0.473	0.545	0.281	0.496	0.682	0.283	0.520
	SMC	0.317	0.277	0.473	0.532	0.281	0.512	0.636	0.282	0.509
	BoNG	0.407	0.280	0.480	0.659	0.283	0.507	0.793	0.285	0.535
SD v1.5 (0.9B) (w/ DDPM 100 steps)	Vanilla	0.396	0.279	0.481	0.605	0.282	0.513	0.742	0.284	0.525
	SMC	0.396	0.279	0.481	0.675	0.283	0.511	0.826	0.286	0.529
	BoNG	0.511	0.280	0.491	0.749	0.284	0.521	0.894	0.286	0.555
SDXL (2.6B) (w/ DDIM 50 steps)	Vanilla	0.899	0.285	0.578	1.060	0.288	0.600	1.140	0.289	0.613
	SMC	0.899	0.285	0.578	1.011	0.287	0.596	1.093	0.289	0.612
	BoNG	0.911	0.287	0.573	1.073	0.289	0.594	1.149	0.292	0.606
SDXL (2.6B) (w/ DDPM 100 steps)	Vanilla	1.037	0.286	0.600	1.162	0.289	0.617	1.228	0.290	0.634
	SMC	1.037	0.286	0.600	1.191	0.288	0.623	1.264	0.290	0.631
	BoNG	1.059	0.287	0.597	1.194	0.289	0.618	1.259	0.290	0.648

population is steered toward the current Best-of- N target during reverse diffusion. Applying BoNG over more timesteps yields stronger steering toward high-reward regions, whereas restricting the window leaves more denoising steps governed by the original diffusion process. In this sense, the guidance timestep window in BoNG plays a role analogous to the reward-strength hyperparameter λ in prior methods [2, 18, 15] targeting the reward-tilted distribution in Eq. (3). Consistent with Theorem 4.3, increasing the guidance window corresponds to a more reward-tilted target distribution. In addition, following prior guidance methods [4, 9], we introduce a guidance scale multiplying the BoN guidance term in Eq. (7). This provides an additional degree of control over the magnitude of guidance at each timestep. Further analysis of BoNG is provided in Sections C and D.1.

5 Experiments

We use GenEval [6] prompts for evaluation and ImageReward (IR) [36] as the reward function during inference time. We report IR, CLIP Score [7], and GenEval score for all methods. For single-output performance we compare against Vanilla BoN sampling and Sequential Monte Carlo (SMC) [24], since both methods are designed for best-sample generation under N , with comparable computational cost. We report the performance of the BoN sample (i.e., highest IR particle) accordingly. For multi-output performance we compare against representative guidance-based baselines, including gradient-guidance [2, 18] and sample-based guidance [13] methods. For each method, we report the average reward of 4 generated samples. See Section B for more details.

5.1 Single-Output Performance

As shown in Table 1, BoNG consistently outperforms baselines across different backbone models with varying sample budget N . Over 36 comparisons, BoNG achieves the highest score in 29 cases, ranking first in 80.56% of all comparisons against baselines. Even with 2 particles, BoNG shows steep performance gains over baselines and further shows superior performance across $N = 2, 3, 4$. In particular, BoNG with $N = 4$ outperforms Vanilla BoN with $N = 8$ in GenEval (see Figure 1). This advantage scales to larger budgets where BoNG continues to outperform SMC and Vanilla BoN (see Figure 7). Overall, these results demonstrate that even with small N budget, BoNG outputs a high reward sample compared to Vanilla BoN and SMC. BoNG is also applicable to modern flow-matching backbones, retaining its advantage on FLUX.1-dev (12B) [14] (Table 5).

Why does BoNG outperform SMC? SMC also steers the generative process by incorporating reward information by interaction of particles, similar to BoNG. The main difference is that SMC

Table 2: Average performance on GenEval prompts with SD v1.5 backbone. Metrics are averaged over the 4 generated images per prompt, and cost is measured for generating 4 images (5 for Best-4-of-5) on a single A100 GPU. **Bold** entries indicate the best results.

Method	Performance ($\uparrow$)			Cost ($\downarrow$)	
	IR	CLIP	GenEval	Time (s)	Mem. (G)
Vanilla	-0.001	0.271	0.426	7.07	8.90
Vanilla (250 steps)	0.003	0.272	0.430	17.42	8.90
Vanilla Best-4-of-5	0.180	0.275	0.453	8.83	9.97
UG [2]	0.326	0.262	0.355	58.36	28.16
DATE [18]	0.364	0.274	0.438	32.89	24.71
LiDAR [13]	0.384	0.278	0.478	13.41	8.90
BoNG	0.506	0.280	0.511	8.06	8.90

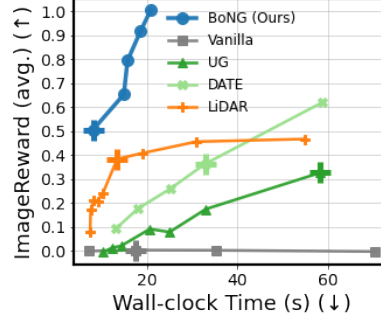


Figure 4: Reward-time trade-off. The reported time denotes the average time required to generate four images. The cross marker on each curve corresponds to the performance reported in Table 2.

relies on resampling, whereas BoNG does not. In SMC, even if denoising starts from N independent particles, repeated resampling based on intermediate rewards can collapse the particle population onto only a few trajectories, or even a single one. Such particle collapse reduces exploration and degrades the quality of the final Best-of- N sample. In contrast, BoNG preserves all particles and instead performs Best-of- N selection in the predicted clean-sample space at each intermediate timestep. The selected high-reward target is then used as a guidance signal so that the particle population is steered to high-reward regions without particle copy. As a result, BoNG maintains a better exploration-exploitation tradeoff than SMC, which leads to stronger final BoN performance under the same particle budget. Moreover, since guidance and resampling act on different components of the particle system, the two are composable: applying BoNG on top of SMC improves both the mean and the BoN reward at every particle count (Section F.4).

5.2 Multi-Output Performance

As shown in Table 2, BoNG achieves the best performance across all three metrics, demonstrating that its benefit is not limited to improving only the final selected sample (BoN) but also for overall samples. More importantly, these gains are obtained with very small test-time overhead. Compared with prior guidance methods, BoNG shows the strongest average performance while remaining close to vanilla diffusion sampling in inference time.

Why does BoNG outperform prior guidance methods? These results are consistent with Theorem 4.3, which shows that BoNG can also be interpreted as a sample-based guidance method targeting the reward-tilted distribution Eq. (3). Its strong empirical performance comes from combining this connection with the core strength of Best-of- N sampling: selecting a single high-reward output from N generated candidates. Rather than applying Best-of- N sampling only after generation, BoNG injects the same principle directly into reverse diffusion by repeatedly using the current Best-of- N target to steer the denoising particle population toward higher-reward regions. At the same time, unlike prior sample-based guidance methods, BoNG does not require additional forward or backward rollouts to construct guidance samples. Instead, it directly reuses the base scores and one-step denoised samples of particles already present in the denoising process. As a result, BoNG inherits the strong best-sample behavior of Best-of- N sampling while avoiding the computational overhead of prior guidance methods, leading to a substantially better efficiency performance tradeoff of the overall quality of multiple generated samples.

5.3 Analysis

Reward-Diversity Trade-off against SMC. Similar to BoNG, SMC can also be adapted to the multi-output setting. SMC does it by tuning the reward-weighting parameter λ that determines particle potentials, whereas BoNG controls the guidance time window in Section 4.3. Figure 5 shows that BoNG achieves a consistently better reward-diversity Pareto frontier than SMC. SMC incorporates reward information through resampling, which necessarily duplicates some particles and removes others, while BoNG incorporates reward information through guidance and thus modifies

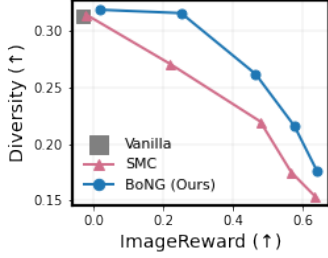


Figure 5: Reward-Diversity trade-off of BoNG and SMC.

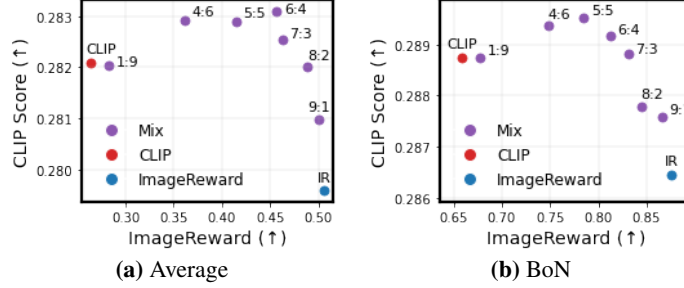


Figure 6: Average and BoN scores over four particles under different reward mixing weights (IR : CLIP).

Table 3: Guidance with HPSv2 [34] as the reward model (SD v1.5, DDPM 100 steps, $N = 4$).

Method	Mean (↑)				BoN (↑)			
	HPSv2	IR	CLIP	GenEval	HPSv2	IR	CLIP	GenEval
Vanilla	0.263	-0.018	0.272	0.427	0.287	0.488	0.282	0.517
BoNG	0.276	0.316	0.277	0.487	0.288	0.495	0.282	0.514

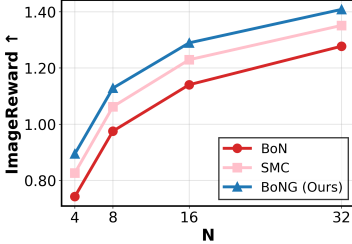


Figure 7: BoN performance across large sample budget N (SD v1.5, DDPM 100 steps).

Table 4: Comparison with LiDAR under matched end-to-end wall-clock time (SD v1.5, DDPM 100 steps, $\delta = 5$ for LiDAR; BoN performance). **Bold** entries indicate the best results.

Method	Performance (↑)			Cost (↓)
	IR	CLIP	GenEval	Time (s)
LiDAR ($N = 2, n = 50$)	0.734	0.284	0.515	10.59
LiDAR ($N = 3, n = 25$)	0.760	0.284	0.515	8.92
LiDAR ($N = 4, n = 9$)	0.866	0.286	0.549	8.24
BoNG ($N = 4$)	0.894	0.286	0.555	8.06

Table 5: BoN performance on FLUX.1-dev (12B) [14] with the flow-matching Euler 28-step sampler.

Method	$N = 2$			$N = 3$			$N = 4$		
	IR	CLIP	GenEval	IR	CLIP	GenEval	IR	CLIP	GenEval
Vanilla	1.220	0.285	0.671	1.303	0.287	0.687	1.353	0.287	0.696
BoNG	1.252	0.285	0.680	1.341	0.287	0.696	1.370	0.286	0.700

particle trajectories more gradually. Consequently, BoNG preserves diversity more effectively while maintaining strong reward improvement, giving it a clear advantage in the multi-output setting.

Efficiency–Performance comparison among gradient guidance methods. The guidance baselines in Table 2 incur substantial additional computation: UG and DATE require backward passes through the score and reward models, and LiDAR requires additional lookahead rollouts with score-network evaluations (see Section D.2 for the full cost breakdown). BoNG avoids both sources of overhead, yet achieves better average performance than LiDAR. We attribute this to the construction of the guidance signal: LiDAR relies on random lookahead samples, whereas BoNG steers all particles toward the current highest-reward denoising particle. Figure 4 extends the efficiency–performance Pareto points of Table 2, showing how performance changes as more computation is used; BoNG achieves the steepest performance gain relative to the amount of computation.

Performance with different reward functions. Figure 6 shows results when BoNG is guided by CLIP, ImageReward (IR), or their weighted combination. To account for the different reward scales, we Z-score normalize each reward before applying the mixing weights. As the CLIP weight

increases, the generated samples achieve higher CLIP scores, whereas increasing the IR weight improves ImageReward scores. This trend shows that BoNG can effectively leverage other reward functions as guidance signals. Furthermore, combining IR and CLIP yields a clear trade-off curve between the two metrics. The mixed rewards produce frontier points, indicating that reward mixing can improve the balance across complementary evaluation metrics. These results suggest that BoNG provides a reward-agnostic guidance mechanism that can operate with different reward functions and can further benefit from combining multiple rewards. We further test HPSv2 [34] as the guidance reward in Table 3, where BoNG improves both the mean and the BoN scores over Vanilla, confirming that the guidance mechanism transfers to a different human-preference reward. Results with a heavier reward model are provided in Section F.3.

Budget-matched comparison with LiDAR. Table 1 compares methods whose end-to-end cost stays close to that of Vanilla BoN sampling. LiDAR [13], a representative sample-based guidance method, can also serve the single-output setting, but its cost is higher: independently of N , it must first generate n lookahead samples in advance, each requiring δ score-network evaluations, and its performance is governed by n (see Section D.2 for the full cost breakdown). We therefore compare BoNG against LiDAR under matched end-to-end wall-clock time, trading off N against n for LiDAR. As shown in Table 4, BoNG achieves the best results on every metric with the shortest generation time. This is because BoNG constructs its guidance signal from the N particles currently being sampled, whereas LiDAR relies on separately generated lookahead samples. Additional experiments are provided in Section F.2.

Scalability of BoNG. BoNG scales with both the sample budget and the backbone. Figure 7 shows that on SD v1.5 (DDPM 100 steps), BoNG consistently outperforms SMC and Vanilla BoN for $N \in \{8, 16, 32\}$, and its BoN performance keeps improving with N , inheriting the scaling behavior of Best-of- N sampling. BoNG also benefits from a stronger generative process: in Table 1, its performance increases as the backbone grows from SD v1.5 (0.9B) to SDXL (2.6B) and as the sampler becomes more accurate (DDIM 50 to DDPM 100 steps). This trend extends to flow-matching backbones, to which BoNG applies directly since it requires only the one-step denoised estimate in Eq. (5). On FLUX.1-dev (12B) [14], Table 5 shows that BoNG continues to improve over Vanilla BoN at every N , confirming that its gains carry over to a larger and stronger backbone.

6 Conclusion

We propose BoNG, a test-time alignment method that integrates Best-of- N sampling directly into the reverse diffusion process of diffusion models. By performing online Best-of- N selection among denoising particles, BoNG steers particles to high reward regions during generation rather than using rewards only for post-hoc selection. We further show that BoNG is theoretically connected to sample-based guidance for reward-tilted diffusion sampling, while avoiding the costly rollout procedure required by prior methods. Empirically, BoNG improves both single-output (BoN) performance and average multi-output quality, achieving strong alignment gains with minimal additional inference cost. These results demonstrate that BoNG is an efficient and flexible framework for test-time diffusion alignment.

Limitations and Future Work. BoNG assumes that the reward model faithfully reflects the target preference; an inaccurate reward model can steer the particles toward unwanted regions, a limitation shared by all reward-based alignment methods. As future work, the finite- λ relaxation in Section E, which replaces the hard Best-of- N selection with a soft-weighted posterior mean, opens a promising direction: exploring this relaxation in greater depth, for example by scaling λ to control the strength of alignment, would be a meaningful extension of BoNG.

Acknowledgments and Disclosure of Funding

This work was supported by the IITP (Institute of Information & Communications Technology Planning & Evaluation)-ITRC (Information Technology Research Center) grant funded by the Korea government (Ministry of Science and ICT) (IITP-2026-RS-2024-00437268, 50%), and by AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data (IITP) funded by the Ministry of Science and ICT (RS-2022-II220077, 50%).

References

- [1] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.
- [2] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Roni Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.
- [3] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.
- [4] Hyungjin Chung, Jeongsol Kim, Geon Yeong Park, Hyelin Nam, and Jong Chul Ye. Cfg++: Manifold-constrained classifier free guidance for diffusion models. *arXiv preprint arXiv:2406.08070*, 2024.
- [5] Carles Domingo-Enrich, Michal Drozdal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [6] Dhruva Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36:52132–52152, 2023.
- [7] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528, 2021.
- [8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [9] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.
- [10] Peter Holderrieth, Uriel Singer, Tommi Jaakkola, Ricky T. Q. Chen, Yaron Lipman, and Brian Karrer. GLASS flows: Efficient inference for reward alignment of flow and diffusion models. In *The Fourteenth International Conference on Learning Representations*, 2026.
- [11] Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, Kenshi Abe, Mitsuki Sakamoto, and Eiji Uchibe. Evaluation of best-of-n sampling strategies for language model alignment, 2025.
- [12] Dongjun Kim, Yeongmin Kim, Se Jung Kwon, Wanmo Kang, and Il-Chul Moon. Refining generative process with discriminator guidance in score-based diffusion models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 16567–16598. PMLR, 23–29 Jul 2023.
- [13] Yeongmin Kim, Donghyeok Shin, Byeonghu Na, Minsang Park, Richard Lee Kim, and Il-Chul Moon. Lookahead sample reward guidance for test-time scaling of diffusion models. In *Forty-third International Conference on Machine Learning*, 2026.
- [14] Black Forest Labs. Flux. <https://github.com/black-forest-labs/flux>, 2024.
- [15] Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gökçen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, and Masatoshi Uehara. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- [16] Yuhang Ma, Yunhao Shui, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. Hpsv3: Towards wide-spectrum human preference score, 2025.
- [17] Microsoft Team. <https://www.bing.com/images/create>, 2026.

- [18] Byeonghu Na, Minsang Park, Gyuwon Sim, Donghyeok Shin, HeeSun Bae, Mina Kang, Se Jung Kwon, Wanmo Kang, and Il chul Moon. Diffusion adaptive text embedding for text-to-image diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- [19] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024.
- [20] Peter Potaptchik, Cheuk-Kit Lee, and Michael S Albergo. Tilt matching for scalable sampling and fine-tuning. In *Forty-third International Conference on Machine Learning*, 2026.
- [21] Peter Potaptchik, Adhi Saravanan, Abbas Mammadov, Alvaro Prat, Michael S Albergo, and Yee Whye Teh. Meta flow maps enable scalable reward alignment. In *Forty-third International Conference on Machine Learning*, 2026.
- [22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- [23] Seyedmorteza Sadat, Otmar Hilliges, and Romann M Weber. Eliminating oversaturation and artifacts of high guidance scales in diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2024.
- [24] Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.
- [25] Marta Skreta, Tara Akhound-Sadegh, Viktor Ohanesian, Roberto Bondesan, Alan Aspuru-Guzik, Arnaud Doucet, Rob Brekelmans, Alexander Tong, and Kirill Neklyudov. Feynman-kac correctors in diffusion: Annealing, guidance, and product of experts. In *Forty-second International Conference on Machine Learning*, 2025.
- [26] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.
- [27] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.
- [28] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 11918–11930, 2019.
- [29] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.
- [30] Adith Swaminathan and Thorsten Joachims. The self-normalized estimator for counterfactual learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- [31] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024.
- [32] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.
- [33] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil 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.

- [34] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.
- [35] xAI Team. <https://grok.com/imagine>, 2026. Accessed: 2026-01.
- [36] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

A Proofs and Derivations

A.1 Derivation of Lemma 4.1

Lemma 4.1. *The gradient of the expected future reward can be expressed as*

$$\nabla_{\mathbf{x}_t} r_t^\lambda(\mathbf{x}_t, \mathbf{c}) = \frac{1}{\sigma_t^2} (m_\lambda^r(\mathbf{x}_t) - \mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0]), \quad m_\lambda^r(\mathbf{x}_t) := \mathbb{E}_{p_\theta^r(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0]. \quad (9)$$

Proof. We have a forward Gaussian Kernel and its Stein score as:

$$p_t(\mathbf{x}_t|\mathbf{x}_0) = \frac{1}{\sigma_t \sqrt{2\pi}} \exp\left(-\frac{(\mathbf{x}_t - \mathbf{x}_0)^2}{2\sigma_t^2}\right), \quad \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_0) = -\frac{\mathbf{x}_t - \mathbf{x}_0}{\sigma_t^2}.$$

With Bayes' rule, we can rewrite expected future reward as follows:

$$\mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})} [\exp(\lambda r(\mathbf{x}_0, \mathbf{c}))] = \int p_\theta(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c}) \exp(\lambda r(\mathbf{x}_0, \mathbf{c})) d\mathbf{x}_0 \quad (12)$$

$$= \int \frac{p_\theta(\mathbf{x}_0 | \mathbf{c}) p_t(\mathbf{x}_t | \mathbf{x}_0)}{\int p_\theta(\mathbf{x}_0 | \mathbf{c}) p_t(\mathbf{x}_t | \mathbf{x}_0) d\mathbf{x}_0} \exp(\lambda r(\mathbf{x}_0, \mathbf{c})) d\mathbf{x}_0 \quad (13)$$

$$= \frac{\int p_\theta(\mathbf{x}_0 | \mathbf{c}) p_t(\mathbf{x}_t | \mathbf{x}_0) \exp(\lambda r(\mathbf{x}_0, \mathbf{c})) d\mathbf{x}_0}{\int p_\theta(\mathbf{x}_0 | \mathbf{c}) p_t(\mathbf{x}_t | \mathbf{x}_0) d\mathbf{x}_0} \quad (14)$$

$$= \frac{z_r(\mathbf{x}_t)}{z_0(\mathbf{x}_t)}, \quad (15)$$

where

$$z_r(\mathbf{x}_t) := \int p_\theta(\mathbf{x}_0 | \mathbf{c}) p_t(\mathbf{x}_t | \mathbf{x}_0) \exp(\lambda r(\mathbf{x}_0, \mathbf{c})) d\mathbf{x}_0 \quad (16)$$

$$z_0(\mathbf{x}_t) := \int p_\theta(\mathbf{x}_0 | \mathbf{c}) p_t(\mathbf{x}_t | \mathbf{x}_0) d\mathbf{x}_0 = p_\theta(\mathbf{x}_t | \mathbf{c}). \quad (17)$$

The gradient of the expected future reward $\nabla_{\mathbf{x}_t} r_t^\lambda(\mathbf{x}_t, \mathbf{c})$ can be expressed as

$$\nabla_{\mathbf{x}_t} r_t^\lambda(\mathbf{x}_t, \mathbf{c}) = \nabla_{\mathbf{x}_t} \log \mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})} [\exp(\lambda r(\mathbf{x}_0, \mathbf{c}))] \quad (18)$$

$$= \nabla_{\mathbf{x}_t} \log z_r(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log z_0(\mathbf{x}_t). \quad (19)$$

The first term $\nabla_{\mathbf{x}_t} \log z_r(\mathbf{x}_t)$ can be expressed as follows:

$$\nabla_{\mathbf{x}_t} \log z_r(\mathbf{x}_t) = \frac{\nabla_{\mathbf{x}_t} z_r(\mathbf{x}_t)}{z_r(\mathbf{x}_t)} \quad (20)$$

$$= \int \frac{p_\theta(\mathbf{x}_0 | \mathbf{c}) \exp(\lambda r(\mathbf{x}_0, \mathbf{c})) \nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t | \mathbf{x}_0)}{z_r(\mathbf{x}_t)} d\mathbf{x}_0 \quad (21)$$

$$= \int \frac{p_\theta(\mathbf{x}_0 | \mathbf{c}) \exp(\lambda r(\mathbf{x}_0, \mathbf{c})) p_t(\mathbf{x}_t | \mathbf{x}_0) \frac{\nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t | \mathbf{x}_0)}{p_t(\mathbf{x}_t | \mathbf{x}_0)}}{z_r(\mathbf{x}_t)} d\mathbf{x}_0 \quad (22)$$

$$= \int \frac{p_\theta(\mathbf{x}_0 | \mathbf{c}) p_t(\mathbf{x}_t | \mathbf{x}_0)}{z_0(\mathbf{x}_t)} \frac{z_0(\mathbf{x}_t)}{z_r(\mathbf{x}_t)} \exp(\lambda r(\mathbf{x}_0, \mathbf{c})) \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{x}_0) d\mathbf{x}_0 \quad (23)$$

$$= \int \frac{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c}) \exp(\lambda r(\mathbf{x}_0, \mathbf{c}))}{\underbrace{\mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})} [\exp(\lambda r(\mathbf{x}_0, \mathbf{c}))]}_{:= p_\theta^r(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})}} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{x}_0) d\mathbf{x}_0 \quad (24)$$

$$= \mathbb{E}_{p_\theta^r(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})} [\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{x}_0)]. \quad (25)$$

We can rewrite the second term $\nabla_{\mathbf{x}_t} \log z_0(\mathbf{x}_t)$ as follows:

$$\nabla_{\mathbf{x}_t} \log z_0(\mathbf{x}_t) = \frac{\nabla_{\mathbf{x}_t} z_0(\mathbf{x}_t)}{z_0(\mathbf{x}_t)} \quad (26)$$

$$= \int \frac{p_\theta(\mathbf{x}_0 | \mathbf{c}) \nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t | \mathbf{x}_0)}{z_0(\mathbf{x}_t)} d\mathbf{x}_0 \quad (27)$$

$$= \int \frac{p_\theta(\mathbf{x}_0 | \mathbf{c}) p_t(\mathbf{x}_t | \mathbf{x}_0) \frac{\nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t | \mathbf{x}_0)}{p_t(\mathbf{x}_t | \mathbf{x}_0)}}{z_0(\mathbf{x}_t)} d\mathbf{x}_0 \quad (28)$$

$$= \int \frac{p_\theta(\mathbf{x}_0 | \mathbf{c}) p_t(\mathbf{x}_t | \mathbf{x}_0)}{p_\theta(\mathbf{x}_t | \mathbf{c})} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{x}_0) d\mathbf{x}_0 \quad (29)$$

$$= \mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})} [\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{x}_0)]. \quad (30)$$

Substituting the two identities above into Eq. 19 and simplifying yields

$$\nabla_{\mathbf{x}_t} r_t^\lambda(\mathbf{x}_t, \mathbf{c}) = \mathbb{E}_{p_\theta^r(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})} [\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{x}_0)] - \mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})} [\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{x}_0)] \quad (31)$$

$$= \mathbb{E}_{p_\theta^r(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})} \left[-\frac{\mathbf{x}_t - \mathbf{x}_0}{\sigma_t^2} \right] - \mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})} \left[-\frac{\mathbf{x}_t - \mathbf{x}_0}{\sigma_t^2} \right] \quad (32)$$

$$= -\frac{1}{\sigma_t^2} (\mathbf{x}_t - \mathbb{E}_{p_\theta^r(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0]) + \frac{1}{\sigma_t^2} (\mathbf{x}_t - \mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0]) \quad (33)$$

$$= \frac{1}{\sigma_t^2} (\mathbb{E}_{p_\theta^r(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0] - \mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0]) \quad (34)$$

$$= \frac{1}{\sigma_t^2} (m_\lambda^r(\mathbf{x}_t) - \mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0]). \quad (35)$$

This completes the proof. $\square$

A.2 Derivation of Lemma 4.2

Lemma 4.2. *Let $q(\mathbf{x}_0)$ be any proposal distribution whose support is $\mathcal{X}_0$. Then, given proposal samples $\hat{\mathbf{x}}_0^1, \dots, \hat{\mathbf{x}}_0^N \stackrel{i.i.d.}{\sim} q(\mathbf{x}_0)$, the reward-tilted posterior mean $m_\lambda^r(\mathbf{x}_t)$ admits the self-normalized importance sampling approximation*

$$m_\lambda^r(\mathbf{x}_t) \approx \hat{m}_{N,\lambda}^r(\mathbf{x}_t) := \sum_{i=1}^N \alpha_i^{(\lambda)} \hat{\mathbf{x}}_0^i, \quad (10)$$

where

$$\alpha_i^{(\lambda)} = \text{softmax}_i \left(\log p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c}) - \log q(\hat{\mathbf{x}}_0^i) + \lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c}) \right).$$

Proof. We derive the self-normalized importance sampling representation of the reward-tilted posterior mean. Specifically, we obtain

$$m_\lambda^r(\mathbf{x}_t) = \mathbb{E}_{p_\theta^r(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0] = \frac{\int p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c}) \exp(\lambda r(\mathbf{x}_0, \mathbf{c})) \mathbf{x}_0 d\mathbf{x}_0}{\mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}[\exp(\lambda r(\mathbf{x}_0, \mathbf{c}))]} \quad (36)$$

$$= \frac{\mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0 \exp(\lambda r(\mathbf{x}_0, \mathbf{c}))]}{\mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}[\exp(\lambda r(\mathbf{x}_0, \mathbf{c}))]}. \quad (37)$$

Let $q(\mathbf{x}_0)$ be any proposal distribution whose support contains that of $p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})$. By change of measure, both the numerator and denominator in Eq. (37) can be rewritten under $q(\mathbf{x}_0)$, yielding

$$m_\lambda^r(\mathbf{x}_t) = \frac{\mathbb{E}_{q(\mathbf{x}_0)} \left[\frac{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}{q(\mathbf{x}_0)} \mathbf{x}_0 \exp(\lambda r(\mathbf{x}_0, \mathbf{c})) \right]}{\mathbb{E}_{q(\mathbf{x}_0)} \left[\frac{p_\theta(\mathbf{x}_0 | \mathbf{x}_t, \mathbf{c})}{q(\mathbf{x}_0)} \exp(\lambda r(\mathbf{x}_0, \mathbf{c})) \right]}. \quad (38)$$

Now let $\hat{\mathbf{x}}_0^1, \dots, \hat{\mathbf{x}}_0^N \stackrel{i.i.d.}{\sim} q(\mathbf{x}_0)$. Applying Monte Carlo approximation to both expectations in Eq. (38), we obtain

$$m_\lambda^r(\mathbf{x}_t) \approx \frac{\frac{1}{N} \sum_{i=1}^N \frac{p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c})}{q(\hat{\mathbf{x}}_0^i)} \hat{\mathbf{x}}_0^i \exp(\lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c}))}{\frac{1}{N} \sum_{i=1}^N \frac{p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c})}{q(\hat{\mathbf{x}}_0^i)} \exp(\lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c}))} \quad (39)$$

$$= \frac{\sum_{i=1}^N \frac{p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c})}{q(\hat{\mathbf{x}}_0^i)} \hat{\mathbf{x}}_0^i \exp(\lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c}))}{\sum_{i=1}^N \frac{p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c})}{q(\hat{\mathbf{x}}_0^i)} \exp(\lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c}))}. \quad (40)$$

Define the unnormalized importance weights

$$w_i^{(\lambda)} := \frac{p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c})}{q(\hat{\mathbf{x}}_0^i)} \exp(\lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c})). \quad (41)$$

Then Eq. (40) can be written as

$$m_\lambda^r(\mathbf{x}_t) \approx \sum_{i=1}^N \frac{w_i^{(\lambda)}}{\sum_{j=1}^N w_j^{(\lambda)}} \hat{\mathbf{x}}_0^i. \quad (42)$$

Therefore, if we define

$$\alpha_i^{(\lambda)} := \frac{w_i^{(\lambda)}}{\sum_{j=1}^N w_j^{(\lambda)}}, \quad (43)$$

we obtain the self-normalized importance sampling estimator

$$\hat{m}_{N,\lambda}^r(\mathbf{x}_t) := \sum_{i=1}^N \alpha_i^{(\lambda)} \hat{\mathbf{x}}_0^i. \quad (44)$$

Finally, taking logarithms of the unnormalized weights gives

$$\log w_i^{(\lambda)} = \log p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c}) - \log q(\hat{\mathbf{x}}_0^i) + \lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c}). \quad (45)$$

Hence the normalized weights admit the softmax form

$$\alpha_i^{(\lambda)} = \text{softmax}_i \left(\log p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c}) - \log q(\hat{\mathbf{x}}_0^i) + \lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c}) \right), \quad (46)$$

which proves the result. $\square$

A.3 Derivation of Theorem 4.3

Theorem 4.3. Let $\{\hat{\mathbf{x}}_0^i\}_{i=1}^N$ be the proposal set of one-step denoised samples in Eq. (5) with a unique maximizer $i^* = \arg \max_{1 \leq i \leq N} r(\hat{\mathbf{x}}_0^i, \mathbf{c})$, and let $\hat{\mathbf{x}}_0^{\text{BoN}} := \hat{\mathbf{x}}_0^{i^*}$. Then $\hat{m}_{N,\lambda}^r(\mathbf{x}_t^i) \rightarrow \hat{\mathbf{x}}_0^{\text{BoN}}$ as $\lambda \rightarrow \infty$, and the SNIS approximation of the reward-guidance gradient in Eq. (9) converges to the BoN guidance term for each particle:

$$\nabla_{\mathbf{x}_t^i} r_t^\lambda(\mathbf{x}_t^i, \mathbf{c}) \approx \frac{1}{\sigma_t^2} (\hat{m}_{N,\lambda}^r(\mathbf{x}_t^i) - \hat{\mathbf{x}}_0^i) \xrightarrow{\lambda \rightarrow \infty} \nabla_{\mathbf{x}_t^i} r_t^{\text{BoN}}(\mathbf{x}_t^i, \mathbf{c}). \quad (11)$$

Proof. By Eq. (9), the target process score in Eq. (4) depends on the reward-tilted posterior mean $m_\lambda^r(\mathbf{x}_t)$. By Lemma 4.2, this posterior mean admits the self-normalized importance sampling approximation

$$m_\lambda^r(\mathbf{x}_t) \approx \hat{m}_{N,\lambda}^r(\mathbf{x}_t) = \sum_{i=1}^N \alpha_i^{(\lambda)} \hat{\mathbf{x}}_0^i,$$

where

$$\alpha_i^{(\lambda)} = \text{softmax}_i \left(\log p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c}) - \log q(\hat{\mathbf{x}}_0^i) + \lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c}) \right).$$

Let

$$i^* = \arg \max_{1 \leq i \leq N} r(\hat{\mathbf{x}}_0^i, \mathbf{c}), \quad \hat{\mathbf{x}}_0^{\text{BoN}} := \hat{\mathbf{x}}_0^{i^*},$$

which is unique by assumption. Then for any $i \neq i^*$,

$$\begin{aligned} \frac{\alpha_i^{(\lambda)}}{\alpha_{i^*}^{(\lambda)}} &= \exp\left(\log p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c}) - \log q(\hat{\mathbf{x}}_0^i) + \lambda r(\hat{\mathbf{x}}_0^i, \mathbf{c})\right) \\ &\quad \times \exp\left(-\log p_\theta(\hat{\mathbf{x}}_0^{i^*} | \mathbf{x}_t, \mathbf{c}) + \log q(\hat{\mathbf{x}}_0^{i^*}) - \lambda r(\hat{\mathbf{x}}_0^{i^*}, \mathbf{c})\right) \\ &= C_i \exp\left(\lambda(r(\hat{\mathbf{x}}_0^i, \mathbf{c}) - r(\hat{\mathbf{x}}_0^{i^*}, \mathbf{c}))\right), \end{aligned} \quad (47)$$

where

$$C_i := \exp\left(\log p_\theta(\hat{\mathbf{x}}_0^i | \mathbf{x}_t, \mathbf{c}) - \log q(\hat{\mathbf{x}}_0^i) - \log p_\theta(\hat{\mathbf{x}}_0^{i^*} | \mathbf{x}_t, \mathbf{c}) + \log q(\hat{\mathbf{x}}_0^{i^*})\right)$$

is independent of λ . Since i^* is the unique maximizer, we have

$$r(\hat{\mathbf{x}}_0^i, \mathbf{c}) - r(\hat{\mathbf{x}}_0^{i^*}, \mathbf{c}) < 0 \quad \text{for all } i \neq i^*.$$

Therefore,

$$\frac{\alpha_i^{(\lambda)}}{\alpha_{i^*}^{(\lambda)}} \xrightarrow{\lambda \rightarrow \infty} 0 \quad \text{for all } i \neq i^*.$$

Since $\sum_{i=1}^N \alpha_i^{(\lambda)} = 1$, it follows that

$$\alpha_i^{(\lambda)} \xrightarrow{\lambda \rightarrow \infty} 0 \quad (i \neq i^*), \quad \alpha_{i^*}^{(\lambda)} \xrightarrow{\lambda \rightarrow \infty} 1.$$

Hence,

$$\hat{m}_{N,\lambda}^r(\mathbf{x}_t) = \sum_{i=1}^N \alpha_i^{(\lambda)} \hat{\mathbf{x}}_0^i \xrightarrow{\lambda \rightarrow \infty} \hat{\mathbf{x}}_0^{i^*} = \hat{\mathbf{x}}_0^{\text{BoN}}. \quad (48)$$

Now evaluate at a query particle $\mathbf{x}_t = \mathbf{x}_t^i$. By the proposal construction in Eqs. (5) and (6),

$$\mathbb{E}_{p_\theta(\mathbf{x}_0 | \mathbf{x}_t^i, \mathbf{c})}[\mathbf{x}_0] = \hat{\mathbf{x}}_0^i, \quad \hat{\mathbf{x}}_0^{\text{BoN}} = \hat{\mathbf{x}}_0^{i^*}.$$

Substituting the SNIS approximation into Eq. (9) evaluated at $\mathbf{x}_t^i$, we obtain

$$\nabla_{\mathbf{x}_t^i} r_t^\lambda(\mathbf{x}_t^i, \mathbf{c}) \approx \frac{1}{\sigma_t^2} (\hat{m}_{N,\lambda}^r(\mathbf{x}_t^i) - \hat{\mathbf{x}}_0^i). \quad (49)$$

By Eq. (48), the right-hand side converges as $\lambda \rightarrow \infty$ to

$$\frac{1}{\sigma_t^2} (\hat{\mathbf{x}}_0^{\text{BoN}} - \hat{\mathbf{x}}_0^i) = \nabla_{\mathbf{x}_t^i} r_t^{\text{BoN}}(\mathbf{x}_t^i, \mathbf{c}),$$

which is the BoN guidance term in Eq. (7). This establishes Eq. (11) and proves the theorem. $\square$

B Experimental Settings

B.1 Experiment Setup

We use SD v1.5 [22] and SDXL [19] backbones with CFG [9] scale 7.5 for all experiments. For each backbone, we experiment with the DDPM 100-step sampler and the DDIM 50-step sampler (probability-flow ODE) in Table 1, the latter being substantially faster. We additionally evaluate on the flow-matching backbone FLUX.1-dev [14], using the flow-matching Euler sampler with 28 steps and guidance scale 3.5. All results are averaged over three random seeds, except FLUX.1-dev, which uses a single seed due to computational cost.

B.2 BoNG Configuration

BoNG has three hyperparameters: the window position t_{start} , the window size K , and the guidance scale s . The selection protocol and the corresponding sensitivity analyses are given in Sections C and D; here we record the values used in the main experiments.

For a compute budget comparable to the SMC baseline [24], which performs five resampling steps, we set the number of guided transitions to $K \leq 4$ in all configurations of Table 1. A guidance window $[t_{\text{start}}, t_{\text{end}}]$ denotes the set of guided transitions $t \rightarrow t+1$ for $t_{\text{start}} \leq t < t_{\text{end}}$; e.g., $[40, 44]$ denotes the four transitions $40 \rightarrow 41, \dots, 43 \rightarrow 44$ ($K = 4$). The guidance windows are $[40, 44]$ for SD v1.5 with DDPM 100 steps (where step 40 corresponds to $t = 0.6$, with $t = 1$ the prior and $t = 0$ the data), $[21, 23]$ for SD v1.5 with DDIM 50 steps, $[24, 28]$ for SDXL with DDPM 100 steps, $[20, 22]$ for SDXL with DDIM 50 steps, $[12, 14]$ for FLUX with Euler 28 steps. The guidance scale is $s = 6$ for SD v1.5, $s = 4$ for SDXL, $s = 2$ for FLUX. Table 2 uses the same configuration as Table 1.

B.3 Baseline Configuration

We apply the shared settings of Section B.1 to all baselines. In Table 1, Vanilla runs standard diffusion sampling N times and selects the best sample according to ImageReward [36]. The baseline denoted SMC is FK-Steering [24] under its recommended text-to-image configuration: max potential, $\lambda = 10$, five evenly spaced resampling steps, adaptive resampling, and intermediate rewards on denoised estimates. Best- k -of- N involves no guidance: $N = 5$ samples are generated with the base model, and the top $k = 4$ samples by reward are returned.

For Table 2, we follow the standard practice of quoting the reported numbers for LiDAR [13], UG [2], and DATE [18] from the concurrent work LiDAR, which achieves state-of-the-art performance in the multi-output setting. Please refer to Sections B.1 and B.2 of LiDAR for the corresponding configurations. For transparency, we additionally reproduce all baselines of Table 2 in our unified codebase under the same configurations and report the results in Section F.1.

B.4 Evaluation Metrics

We report the metrics used throughout our experiments.

ImageReward [36]: a learned human-preference reward model for text-to-image generation. It assigns a scalar preference score to each generated image conditioned on the text prompt, and we use it to evaluate overall image quality and prompt satisfaction from a human-centric perspective.

CLIP Score [7]: measures the semantic alignment between the text prompt and the generated image by computing the similarity between their CLIP embeddings. It provides a reference-free estimate of prompt-image correspondence, although it may be less sensitive to fine-grained compositional errors.

GenEval [6]: an object-centric evaluation benchmark for text-to-image generation. It uses detection-based checks to assess whether generated images correctly satisfy prompt-specified entities, attributes, counts, and spatial relations, making it useful for measuring compositional accuracy.

We measure the diversity of generated samples by computing pairwise distances in the CLIP image-embedding space, following the evaluation protocol of FK-Steering.¹

C Hyperparameter Selection Protocol of BoNG

BoNG has three hyperparameters: the window position t_{start} , the window size K , and the guidance scale s . For a new backbone or sampler, we set them by the following protocol:

- **Window position** t_{start} : place the window where the correlation between the Tweedie-estimate reward $r(\hat{x}_0)$ and the final reward becomes reliable (≈ 0.6).
- **Window size** K : choose K by the compute budget, at or below the resampling budget of the SMC baseline.

¹https://github.com/zacharyhorvitz/D-Diffusion-Steering/blob/main/text_to_image/fkd_diffusers/rewards.py

- **Guidance scale s :** tune $s \in [1, 6]$ to select the desired point on the reward–diversity frontier, at no extra compute.

This procedure was followed for every configuration in the paper. The remainder of this section justifies each rule with controlled experiments: Section C.1 shows why the window position trades off the accuracy of the guidance signal against the steerability of the samples, Section C.2 shows that the window size moves gradually and monotonically along the reward–diversity frontier, and Section C.3 shows that the guidance scale sweeps the same frontier at zero additional cost.

C.1 Window Position

A meaningful guidance signal requires the one-step denoised estimate $\hat{\mathbf{x}}_0^{\text{BoN}}$ to be aligned with the final reward, so we place the window near the point where the correlation between $r(\hat{\mathbf{x}}_0)$ and the final reward exceeds 0.6 (cf. Figure 5 of FK-Steering [24]). Placing the window too late is ineffective for the opposite reason: by then the high-level features of each sample are already determined, so particles committed to low-reward regions can no longer be steered.

Table 6: Ablation over the window position t_{start} (SD v1.5, DDPM 100 steps, $N = 4$, $K = 4$, $s = 6$; BoN performance).

Window	IR ($\uparrow$)	CLIP ($\uparrow$)	GenEval ($\uparrow$)
Vanilla	0.713	0.284	0.532
[20, 24]	0.753	0.282	0.523
[40, 44]	0.875	0.286	0.559
[60, 64]	0.849	0.287	0.526
[80, 84]	0.866	0.287	0.528

Table 6 reports an ablation over t_{start} with all other factors fixed (SD v1.5, DDPM 100 steps, $N = 4$, $K = 4$, $s = 6$, single seed; BoN performance). The metric-wise breakdown reveals why the optimum sits mid-range: window placement trades off the *accuracy* of the guidance signal against the *steerability* of the samples. An early window ([20, 24]) fails on both metrics because the Tweedie-reward correlation is still low. In a late window ([80, 84]), the correlation is at its highest and ImageReward improves accordingly; however, the high-level compositional content that GenEval measures is already determined by that stage, so accurate guidance can refine samples but can no longer re-steer particles committed to compositionally weaker configurations, and GenEval does not improve over Vanilla. The mid-range window [40, 44], where the correlation first becomes reliable while the high-level content can still be shaped, is the only regime that improves all metrics simultaneously, which is precisely what the window-position rule above is designed to find.

C.2 Window Size

The window size K determines the additional cost of BoNG, which scales linearly as $NK(D + R)$ (see Section D.2). Enlarging K applies the reward signal over more transitions and correspondingly reduces the influence of the base diffusion process, concentrating the particles in high-reward regions; in this sense, K plays a role analogous to the reward-strength hyperparameter λ in prior methods [2, 18, 15]. Table 7 reports a controlled scan over K (SD v1.5, DDPM 100 steps, $s = 1$, $t_{\text{start}} = 40$). Both effects are gradual and monotone: reward increases and diversity decreases as K grows. At the main-experiment setting $K = 4$, the reward improves over Vanilla sampling with no loss of diversity (0.319 vs. 0.313), indicating that the main experiments operate in a regime free of particle collapse. To keep the compute budget below that of the SMC baseline, we set $K = 4$ for 100-step sampling and $K = 2$ for 50-step sampling.

Table 7: Controlled scan over the window size K (SD v1.5, DDPM 100 steps, $N = 4$, $s = 1$, $t_{\text{start}} = 40$).

Window	K	mean IR ($\uparrow$)	Diversity ($\uparrow$)
Vanilla	–	−0.001	0.313
[40, 44]	4	0.019	0.319
[40, 50]	10	0.252	0.316
[40, 55]	15	0.465	0.262
[40, 60]	20	0.576	0.216
[40, 65]	25	0.641	0.176
[40, 70]	30	0.678	0.146
[40, 75]	35	0.707	0.121

C.3 Guidance Scale

CFG [9] improves image fidelity without increasing inference time or the number of function evaluations (NFEs) by introducing a guidance scale $\omega \in \mathbb{R}$ with $\omega \geq 1$ into the conditional score function, as defined in Eq. (50). In the same spirit, following prior guidance methods that expose an analogous scale [2, 13], we introduce a guidance scale $s \in \mathbb{R}$ with $s \geq 1$ on the BoN guidance term, which controls how strongly each non-BoN particle ($\mathbf{x}_t^i$ with $i \neq i^*$) is steered, as described in Eq. (51).

$$\nabla_{\mathbf{x}_t} \log p_{\theta}^{\omega}(\mathbf{x}_t|\mathbf{c}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \omega \times \frac{\mathbb{E}_{p_{\theta}(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0] - \mathbb{E}_{p_{\theta}(\mathbf{x}_0|\mathbf{x}_t)}[\mathbf{x}_0]}{\sigma_t^2} \quad (50)$$

$$\nabla_{\mathbf{x}_t} \log p_{\theta}^{\text{BoN}}(\mathbf{x}_t|\mathbf{c}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{c}) + s \times \frac{\mathbb{E}_{p_{\theta}(\mathbf{x}_0|\mathbf{x}_t^{\text{BoN}}, \mathbf{c})}[\mathbf{x}_0] - \mathbb{E}_{p_{\theta}(\mathbf{x}_0|\mathbf{x}_t, \mathbf{c})}[\mathbf{x}_0]}{\sigma_t^2} \quad (51)$$

Within the protocol, the scale s complements K : since it only rescales the guidance term, it sweeps the same reward–diversity frontier at zero additional compute. Table 8 reports a scan at the fixed main-experiment window [40, 44] ($K = 4$): increasing s raises the mean reward while diversity decreases gradually, so s selects the operating point on the frontier once t_{start} and K are fixed.

Table 8: Scan over the guidance scale s (SD v1.5, DDPM 100 steps, $N = 4$, window [40, 44], $K = 4$).

s	mean IR ($\uparrow$)	Diversity ($\uparrow$)
Vanilla	−0.001	0.313
1	0.019	0.319
2	0.224	0.315
3	0.490	0.255
4	0.473	0.245
6	0.506	0.224

We further compare the sensitivity of BoNG to s against UG and LiDAR, which expose the same hyperparameter, on GenEval [6] test prompts. Figure 8 reports the ImageReward–CLIP score trade-off as s varies, and we make three observations. First, in the range $s \in [1, 7.5]$, UG exhibits reward-hacking behavior: as s increases, ImageReward improves while the CLIP score decreases. In contrast, LiDAR and BoNG achieve more favorable trade-offs across the two metrics. Second, BoNG achieves a stronger ImageReward–CLIP score trade-off than LiDAR. Finally, when $s = 10$, we observe a degradation of BoNG in CLIP score due to issues such as off-manifold steering and color saturation, consistent with the findings of CFG++ [4] and APG [23]; this motivates the protocol range $s \in [1, 6]$. Qualitative results are shown in Figure 9, where we present generations for five randomly selected prompts as s varies from 1 to 10.

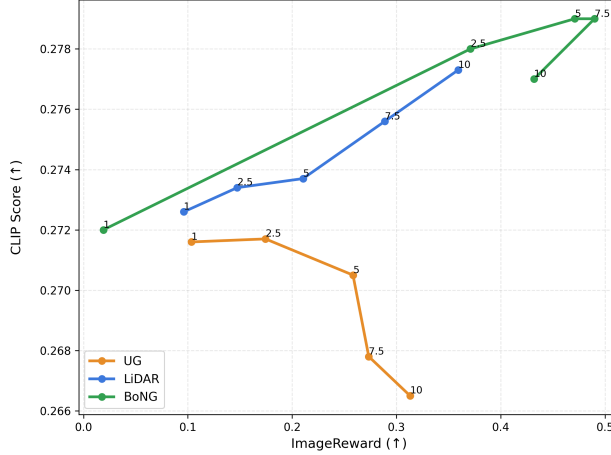


Figure 8: ImageReward [36] and CLIP score [7] trade-offs between UG, LiDAR, and BoNG. Across all methods, we report results under guidance scale $s \in \{1, 2.5, 5, 7.5, 10\}$.

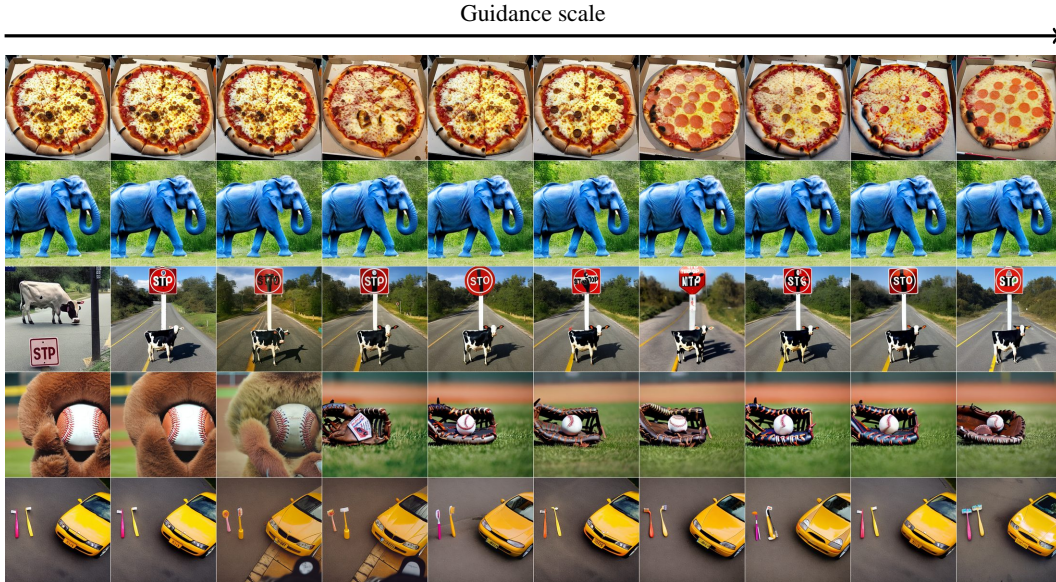


Figure 9: Changes in Best-of- N samples of BoNG as the guidance scale $s \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ varies. The prompts from the top row to the bottom row are “a photo of a pizza”, “a photo of a blue elephant”, “a photo of a cow left of a stop sign”, “a photo of a baseball glove right of a bear”, and “a photo of a yellow car and an orange toothbrush”.

D Additional Analysis of BoNG

D.1 Effect of Guidance Timestep Window

The guidance timestep window jointly determines reward optimization, diversity preservation, and inference-time overhead. As discussed in Sections 4.3 and 5.3, the window plays the role of λ in the reward-tilted distribution: applying BoNG over more timesteps yields stronger reward optimization, but may reduce sample diversity by making the generation process closer to sampling from a highly reward-tilted distribution. At the same time, the window size sets the additional cost $NK(D + R)$ of BoNG (Section D.2). Figure 5 shows the resulting reward–diversity trade-off, Table 7 confirms that both effects are gradual and monotone, and Figure 4 shows that BoNG achieves the best performance–

efficiency trade-off among the compared methods; the selection rules for the window are given in Section C.

D.2 Computational Cost

BoNG introduces no extra score network evaluations over Vanilla diffusion sampling. Its additional cost comes only from constructing the BoN target at guided timesteps, which requires a VAE decoder pass and a reward model pass for each denoising particle. For N particles and K guided timesteps, letting D and R denote the cost of one VAE decoder and reward model forward pass, respectively, the additional function evaluations are $NK(D + R)$.

Table 9 compares each method’s additional cost over Vanilla sampling in hardware-independent units. SMC and BoNG add no additional score-network evaluations; their overhead comes solely from scoring denoised estimates, $NS(D + R)$ and $NK(D + R)$, respectively. UG and DATE evaluate the decoder and reward at every step, $NT(D + R)$, and additionally require backward passes through the score network, decoder, and reward model to obtain reward gradients. LiDAR generates lookahead samples: each of the n lookaheads costs δ score-network evaluations plus one decoder and one reward pass, adding $n\delta + n(D + R)$ in total. Since $K \leq S \ll T$ in all our experiments ($K = 4$, $S = 5$), BoNG incurs the smallest additional cost among the steering methods, with no gradient computation.

Table 9: Additional cost over Vanilla sampling in hardware-independent units. D : VAE-decoder pass, R : reward pass; T : diffusion steps; S : resampling steps of SMC; n , δ : lookahead samples and score evaluations per lookahead of LiDAR.

Method	Score-net	VAE decoder	Reward	Backward
SMC	–	$N \cdot S$	$N \cdot S$	–
UG / DATE	–	$N \cdot T$	$N \cdot T$	required
LiDAR	$n \cdot \delta$	n	n	–
BoNG	–	$N \cdot K$	$N \cdot K$	–

To make the comparisons concrete, Table 10 reports the wall-clock breakdown of one guided denoising step on SD v1.5 ($N = 16$, mean of 5 runs).

Table 10: Wall-clock breakdown of one guided denoising step (SD v1.5, $N = 16$, mean of 5 runs).

Component	Time (ms)	Per particle (ms)	Time %	TFLOPs	TFLOPs / particle
UNet fwd + CFG	238.88	14.93	20.6%	25.705	1.607
VAE decode	522.98	32.69	45.1%	40.232	2.515
Postprocess (tensor→PIL)	167.94	10.50	14.5%	0.000	0.000
Reward (ImageReward)	230.81	14.43	19.9%	2.205	0.138

Regarding memory, we report peak GPU memory during generation in GiB in Table 2. All methods share an identical configuration (backbone, resolution, $N = 4$, GPU), so the differences do not stem from image size or encoding. Notably, sample count alone does not predict the measurements: UG and DATE store no extra samples yet require roughly $3\times$ the memory of Vanilla, while LiDAR stores $n = 50$ lookahead latents yet matches Vanilla’s footprint. We instead characterize the measurements through the memory decomposition itself. With the same networks resident across methods, peak memory is approximately $M \approx P_{\text{score}} + P_{\text{dec}} + P_{\text{reward}} + N \cdot A$, where only the activation term A differs: gradient guidance (UG, DATE) retains the activation graph for backpropagation (A_{bwd}), whereas Vanilla, LiDAR, and BoNG discard activations layer by layer ($A_{\text{fwd}} \ll A_{\text{bwd}}$). BoNG retains no graph and allocates nothing beyond the N particles of Vanilla sampling, so its memory footprint equals Vanilla’s by construction, independently of hardware.

D.3 Anchor-Switching Ablation

To test whether a single particle dominates the guidance signal, we tracked the identity of the Best-of- N anchor $\mathbf{x}_t^{\text{BoN}}$ at every guided transition (window $[40, 44]$: three consecutive transitions per

prompt) across all 553 GenEval prompts (total of 1,659 transitions) across 3 runs. We report the mean, median, and standard deviation of the switch rate of the 3 runs. Switch rate 1.0 implies that for every transition $\mathbf{x}_t^{\text{BoN}}$ was changed, and 0 implies that $\mathbf{x}_t^{\text{BoN}}$ is not changed at all. The experiment shows that the anchor is dynamically re-elected from the population rather than fixed: no single trajectory dominates the guidance target, and the guidance signal genuinely emerges from particle interaction.

Table 11: Anchor-switching statistics over 553 GenEval prompts (1,659 transitions), 3 runs.

Anchor switching (553 prompts, 1,659 transitions)	Value
Mean	0.2656
Median	0.2737
Std	0.0161

E Finite- λ Soft Relaxation of BoNG

BoNG applies hard Best-of- N selection over the one-step denoised candidates and therefore corresponds to the large- λ limit described in Theorem 4.3. However, Lemma 4.2 itself holds for every finite $\lambda > 0$, and the same rollout-free one-step Tweedie construction admits a natural finite- λ relaxation, which we derive and evaluate in this section. We emphasize that this relaxation is a variant beyond the hard-selection algorithm evaluated in the main experiments; all results in the main text correspond to the hard-selection BoNG.

Derivation. Choosing the proposal distribution as $q(\mathbf{x}_0) = p_\theta(\mathbf{x}_0 \mid \mathbf{c})$, the SNIS weights of Lemma 4.2 for a query particle $\mathbf{x}_t^i$ can be written, using Bayes’ rule, as

$$\text{softmax}_j \left(\log p_t(\mathbf{x}_t^i \mid \mathbf{x}_0^j) + \lambda r(\mathbf{x}_0^j, \mathbf{c}) \right), \quad (52)$$

where $\mathbf{x}_0^j \sim p_\theta(\mathbf{x}_0 \mid \mathbf{c})$ and $p_t(\mathbf{x}_t \mid \mathbf{x}_0)$ is the tractable Gaussian forward kernel; the finite- λ estimator is thus computable in principle when exact proposal samples are available. Obtaining such samples, however, requires full reverse-process rollouts. To construct a rollout-free relaxation, we replace the exact samples with the one-step Tweedie estimates already available from the current particle population, $\hat{\mathbf{x}}_0^j = \mathbb{E}_{p_\theta(\mathbf{x}_0 \mid \mathbf{x}_t^j, \mathbf{c})}[\mathbf{x}_0]$, which gives the soft weights

$$\tilde{\alpha}_{ij}^{(\lambda)} = \text{softmax}_j \left(\log p_t(\mathbf{x}_t^i \mid \hat{\mathbf{x}}_0^j) + \lambda r(\hat{\mathbf{x}}_0^j, \mathbf{c}) \right), \quad (53)$$

and the corresponding particle-specific relaxed guidance target

$$\hat{m}_{N,\lambda}^r(\mathbf{x}_t^i) = \sum_{j=1}^N \tilde{\alpha}_{ij}^{(\lambda)} \hat{\mathbf{x}}_0^j. \quad (54)$$

Rather than selecting only the highest-reward candidate, this target softly combines the candidate set according to both the forward compatibility with the query particle and the reward. Smaller values of λ distribute the guidance across multiple candidates, while larger values increasingly concentrate the weights on the highest-reward candidate; assuming a unique reward maximizer, the hard-selection BoNG update is recovered as $\lambda \rightarrow \infty$, consistent with Theorem 4.3.

Approximation gaps. This construction remains approximate in two respects. First, the one-step Tweedie estimates are deterministic posterior means rather than exact proposal samples from $p_\theta(\mathbf{x}_0 \mid \mathbf{c})$. Second, once particle interaction has been applied, the guided particles need not exactly follow the original marginal $p_\theta(\mathbf{x}_t \mid \mathbf{c})$. Consequently, the exact importance-sampling interpretation of Lemma 4.2 does not directly transfer to this rollout-free relaxation: BoNG admits a rollout-free finite- λ relaxation, rather than exactly targeting the finite- λ reward-tilted distribution.

Empirical verification of the predicted scaling. The finite- λ analysis predicts that λ traces the reward-diversity trade-off, with hard-selection BoNG recovered at the large- λ end of the spectrum. To verify this, we implement the soft-weighted relaxation, replacing hard Best-of- N selection with

the softmax weights $\tilde{\alpha}_{ij}^{(\lambda)}$ over the one-step denoised candidates, with the same guidance window as the main experiments (SD v1.5, DDPM 100 steps). Table 12 reports the mean ImageReward and diversity as λ varies. Increasing λ raises the mean reward while diversity decreases gradually, which is precisely the behavior the finite- λ analysis predicts. These results suggest a soft-weighted family of BoNG variants beyond the hard-selection algorithm evaluated in this paper, which we view as a natural direction for future work opened up by our framework.

Table 12: Soft-weighted finite- λ relaxation of BoNG under varying λ ($N = 4$).

Backbone	Method	Mean IR ($\uparrow$)	Diversity ($\uparrow$)
SD v1.5 (0.9B) (w/ DDPM 100 steps)	Vanilla	-0.001	0.313
	$\lambda = 100$	0.001	0.313
	$\lambda = 500$	0.004	0.311
	$\lambda = 1000$	0.131	0.300
	$\lambda = 5000$	0.564	0.234
	$\lambda = 10000$	0.595	0.214

F Additional Experiment Results

This section supplements the main results along both evaluation axes. For the multi-output setting of Table 2, Section F.1 provides a unified-codebase reproduction with the SMC entry and diversity added, and Section F.4 shows that BoNG composes with SMC. For the single-output setting of Table 1, Section F.2 extends the matched wall-clock comparison against LiDAR in Section 5.3 to a large sample budget, and Section F.3 tests a heavier guidance reward beyond the HPSv2 result in Table 3.

F.1 Unified-Codebase Reproduction of Table 2

For transparency, we reproduce all baselines of Table 2 in our unified codebase, following the LiDAR paper’s configurations for the UG and DATE baselines, and additionally report the SMC entry and the diversity of each method. Table 13 reports the results. The relative ordering of Table 2 is preserved under reproduction. SMC attains a higher mean reward than BoNG in this table, but at a substantially lower diversity (0.155 vs. 0.224), since resampling duplicates high-scoring ancestors; this makes it unsuitable for the multi-output setting that Table 2 targets, where N distinct outputs must be served.

Table 13: Unified-codebase reproduction of Table 2 with the SMC entry and diversity added.

Method	IR ($\uparrow$)	CLIP ($\uparrow$)	GenEval ($\uparrow$)	Diversity ($\uparrow$)	Time (s) ($\downarrow$)	Mem. (GiB) ($\downarrow$)
Vanilla	-0.001	0.271	0.426	0.313	7.07	8.9
UG	0.271	0.261	0.364	0.461	57.89	28.2
DATE	0.382	0.275	0.450	0.327	32.68	24.7
LiDAR ($\delta = 5, n = 50$)	0.394	0.278	0.473	0.280	13.57	8.9
SMC	0.641	0.283	0.517	0.155	8.23	8.9
BoNG	0.506	0.280	0.511	0.224	8.06	8.9

F.2 Comparison with LiDAR under Matched Wall-Clock Budgets

Table 4 in Section 5.3 compares BoNG with LiDAR under matched end-to-end wall-clock time at $N = 4$. Here we extend the comparison to a large sample budget. At $N = 32$, we set LiDAR to $n = 100$ and $\delta = 5$ so that its additional cost $n\delta + n(D + R)$ consumes similar time as the $NK(D + R)$ cost of BoNG (see Section D.2), and compare the average performance over the N samples in Table 14. BoNG outperforms LiDAR on every metric with a shorter end-to-end time. This is because LiDAR’s performance gain is independent of N and relies on n , whereas BoNG inherits the scaling behavior of Best-of- N sampling as N increases.

Table 14: Budget-matched comparison with LiDAR at $N = 32$. **Bold** entries indicate the best results.

Backbone	Method	Performance ($\uparrow$)			Cost ($\downarrow$)
		IR	CLIP	GenEval	Time (s)
SD v1.5 (0.9B) (w/ DDPM 100 steps)	Vanilla	-0.012	0.272	0.425	49.5
	LiDAR ($n = 100, \delta = 5$)	0.405	0.281	0.478	61.4
	BoNG	0.854	0.285	0.538	56.1

F.3 Performance with a Heavier Reward Model

Table 3 in Section 5.3 shows that BoNG transfers to HPSv2 [34] as the guidance reward. To further test whether the guidance mechanism holds under a heavier reward backbone, we run Vanilla, SMC, and BoNG with HPSv3 [16] as the guidance reward ($N = 4$, SD v1.5, DDPM 100 steps) and report the BoN performance in Table 15. BoNG improves the HPSv3 reward over Vanilla, confirming that the guidance mechanism transfers to a heavier reward backbone, while SMC attains a slightly higher HPSv3 score in this setting. We also observe that ImageReward decreases under HPSv3 guidance for both SMC and BoNG; we attribute this to HPSv3 using a non-CLIP encoder, so that the two reward models emphasize different features.

Table 15: Guidance with HPSv3 as the reward model ($N = 4$; BoN performance). **Bold** entries indicate the best results.

Backbone	Method	HPSv3 ($\uparrow$)	IR ($\uparrow$)	CLIP ($\uparrow$)	GenEval ($\uparrow$)
SD v1.5 (0.9B) (w/ DDPM 100 steps)	Vanilla	7.287	0.440	0.281	0.520
	SMC	7.418	0.439	0.280	0.520
	BoNG	7.306	0.399	0.279	0.512

F.4 Composing BoNG with SMC

Inference-time steering methods fall into two distinct classes [24]: reward-guidance methods, which steer particles by modifying the sampling dynamics, and particle-based resampling methods, which steer by reweighting and resampling intermediate particles with potentials. BoNG shares a structural connection with SMC, as both maintain N interacting particles and evaluate intermediate rewards on denoised estimates, but the two belong to different classes. BoNG is a reward-guidance method, in which the reward signal enters through the guidance term $\frac{s}{\sigma_t^2} (\hat{\mathbf{x}}_0^{\text{BoN}} - \hat{\mathbf{x}}_0^i)$ added to the score function, whereas SMC is a potential-based resampling method, in which the reward signal enters through the resampling weights G_t while the transition kernel is left unchanged. Concretely, while the outer loop of BoNG resembles that of SMC (propagate N particles, score their denoised estimates with the reward), SMC applies multinomial resampling based on G_t , so particles are copied and deleted, whereas BoNG applies no resampling and instead modifies the score function. In this sense, BoNG brings the strength of the guidance class, with no particle copying or deletion, to the black-box reward setting.

This distinction has a direct practical consequence: since guidance and resampling act on different components of the particle system (the transition kernel and the resampling weights, respectively), they are orthogonal and composable. To verify this empirically, we apply BoNG on top of SMC and report ImageReward with N particles in Table 16, for both the mean over the N returned samples and the Best-of- N sample. Composing BoNG with SMC improves both the mean and the Best-of- N reward at every particle count, with no reward gradients and no additional score-network evaluations. This empirically confirms that BoNG occupies the proposal-generator slot of the FK-IPS framework [24] and complements resampling-based steering.

Table 16: Composing BoNG with SMC. **Bold** entries indicate the best results.

Backbone	Particle Count	Mean IR ($\uparrow$)		BoN IR ($\uparrow$)	
		SMC	+BoNG	SMC	+BoNG
SD v1.5 (0.9B) (w/ DDPM 100 steps)	$N = 2$	-0.039	0.276	0.377	0.417
	$N = 3$	0.463	0.549	0.679	0.684
	$N = 4$	0.641	0.706	0.823	0.833

G Qualitative Comparison

Figure 10 shows the Best-of- N samples of Vanilla, SMC, and BoNG for various prompts under the configurations of Table 1. BoNG produces samples that are better aligned with the prompts across backbones and samplers.

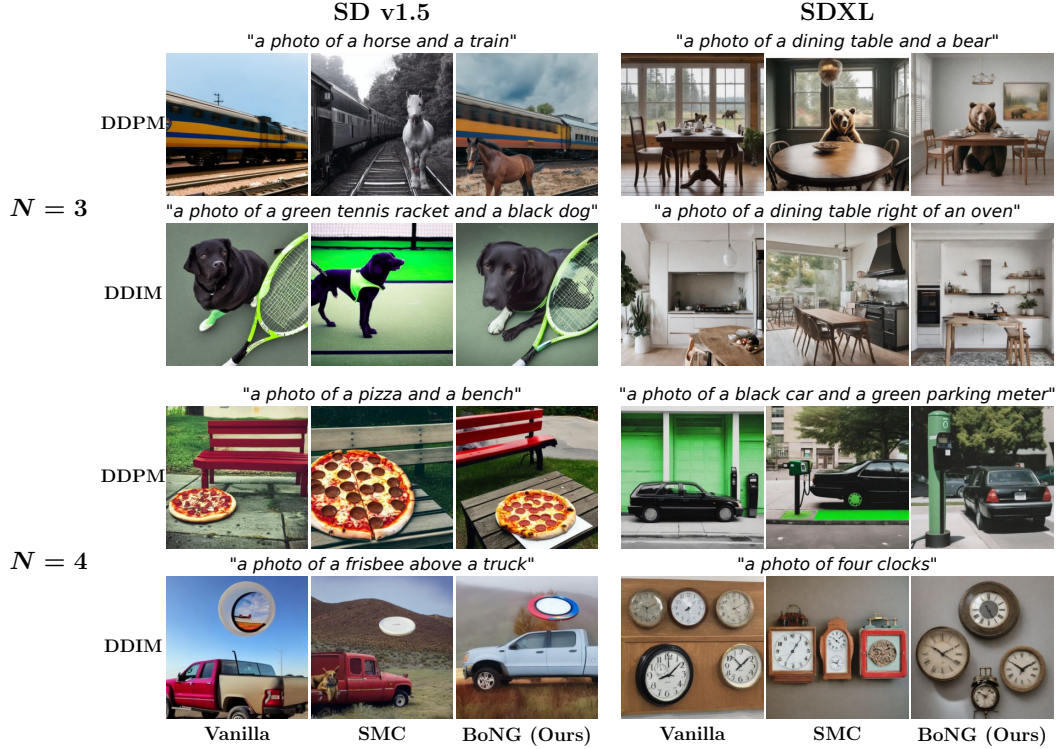


Figure 10: Qualitative comparison of Best-of- N samples from Vanilla, SMC, and BoNG.

H Pseudocode of BoNG

Algorithm 3 presents the full sampling procedure of BoNG, combining the Best-of- N selection of Algorithm 1 and the guided update of Algorithm 2 into a single loop over the reverse diffusion process.

Algorithm 3 BoNG: full sampling procedure

```
1: Input: prompt  $\mathbf{c}$ , number of particles  $N$ , diffusion steps  $T$ , guidance scale  $s$ , guidance window  
    $[t_{\text{start}}, t_{\text{end}}]$ , reward  $r$ , pretrained noise network  $\epsilon_\theta$   
2: Initialize  $\mathbf{x}_T^i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  for  $i = 1, \dots, N$   
3: for  $t = T, \dots, 1$  do  
4:    $\epsilon^i \leftarrow \epsilon_\theta(\mathbf{x}_t^i, t, \mathbf{c})$  for  $i = 1, \dots, N$   
5:   if  $t_{\text{start}} \leq t < t_{\text{end}}$  then  
6:      $\hat{\mathbf{x}}_0^i \leftarrow (\mathbf{x}_t^i - \sigma_t \epsilon^i) / \sqrt{\alpha_t}$  for  $i = 1, \dots, N$  ▷ Alg. 1  
7:      $R^i \leftarrow r(\text{Decode}(\hat{\mathbf{x}}_0^i), \mathbf{c})$  for  $i = 1, \dots, N$   
8:      $i^* \leftarrow \arg \max_i R^i$ ,  $\hat{\mathbf{x}}_0^{\text{BoN}} \leftarrow \hat{\mathbf{x}}_0^{i^*}$   
9:      $\epsilon^i \leftarrow \epsilon^i - \sqrt{\alpha_t} s (\hat{\mathbf{x}}_0^{\text{BoN}} - \hat{\mathbf{x}}_0^i) / \sigma_t$  for  $i = 1, \dots, N$  ▷ Alg. 2  
10:   end if  
11:    $\mathbf{x}_{t-1}^i \leftarrow \text{SchedulerStep}(\epsilon^i, \mathbf{x}_t^i, t)$  for  $i = 1, \dots, N$   
12: end for  
13: Output:  $\{\mathbf{x}_0^i\}_{i=1}^N$ 
```
