# Inference-Time Scaling for Flow Models via Stochastic Generation and Rollover Budget Forcing

Jaihoon Kim*  Taehoon Yoon*  Jisung Hwang*  Minhyuk Sung
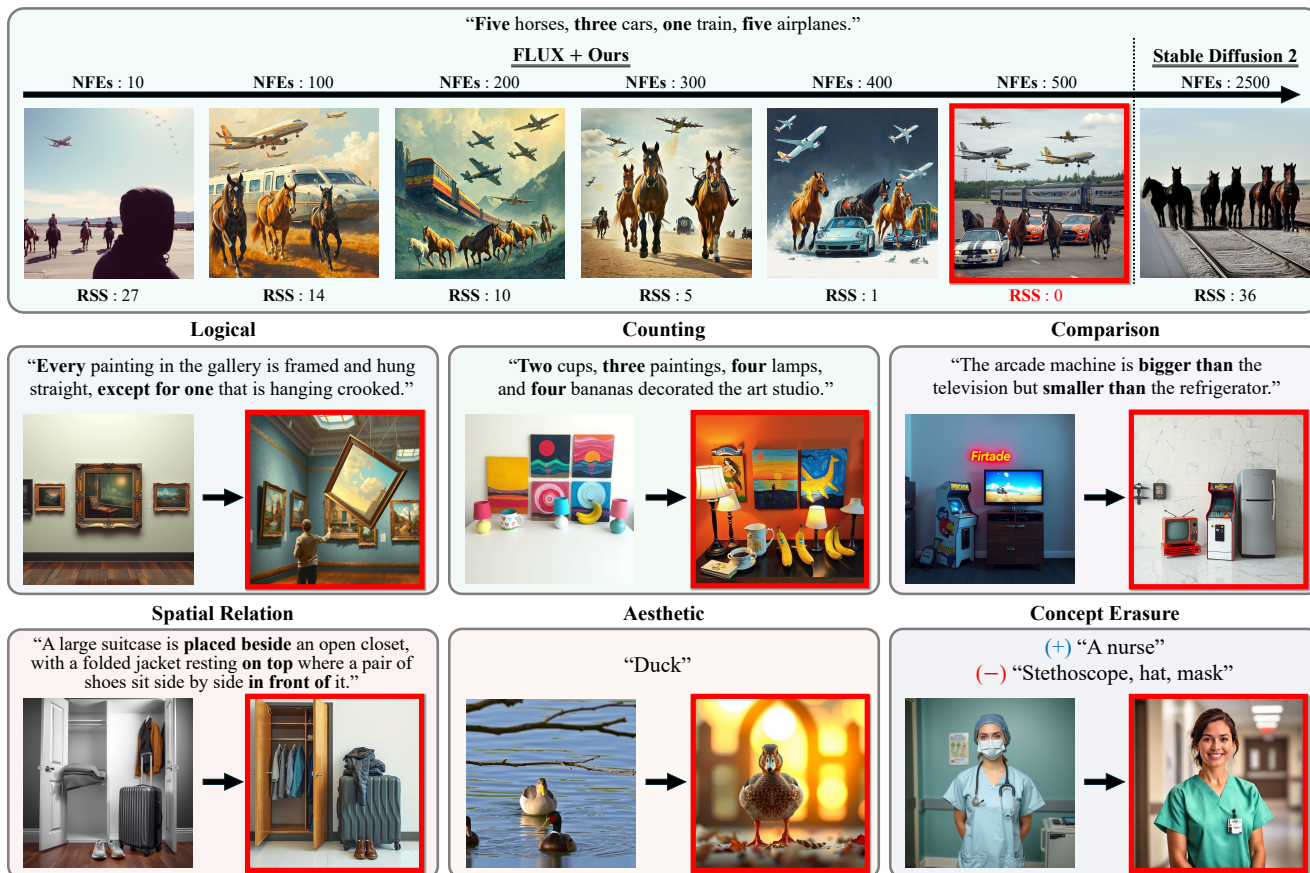
KAIST

{jh27kim,taehoon,4011hjs,mhsung}@kaist.ac.kr

Figure 1. **Diverse applications of our inference-time scaling method.** Our inference-time scaling method extends the capabilities of a pretrained flow model [25] to generate images that more precisely align with user preferences. More computation during inference improves alignment, reducing Residual Sum of Squares (RSS) over time (top row). Our flow-based method outperforms diffusion models, even with five times fewer number of function evaluations (NFEs) (top-right). For compositional text-to-image generation applications ("logical", "comparison", "spatial relation"), we use the reward from VQAScore [28] to ensure precise alignment with the input text, where the description is particularly challenging for typical text-to-image generative models to satisfy (see the results on the left side of each case). We use the object detection score [31] for the "counting" application and the aesthetic score [44] for the "aesthetic" application. For "concept erasure", the reward is the number of removed concepts computed using VLM [3] queries. The red box denotes the results of our method.

## Abstract

*We propose an inference-time scaling approach for pretrained flow models. Recently, inference-time scaling has gained significant attention in LLMs and diffusion models, improving sample quality or better aligning outputs with user preferences by leveraging additional computation. For diffusion models, particle sampling has allowed more efficient scaling due to the stochasticity at intermediate denoising steps. On the contrary, while flow models have gained popu-*

---

*Equal contribution.

*larity as an alternative to diffusion models–offering faster generation and high-quality outputs in state-of-the-art image and video generative models–efficient inference-time scaling methods used for diffusion models cannot be directly applied due to their deterministic generative process. To enable efficient inference-time scaling for flow models, we propose three key ideas: 1) SDE-based generation, enabling particle sampling in flow models, 2) Interpolant conversion, broadening the search space and enhancing sample diversity, and 3) Rollover Budget Forcing (RBF), an adaptive allocation of computational resources across timesteps to maximize budget utilization. Our experiments show that SDE-based generation, particularly variance-preserving (VP) interpolant-based generation, improves the performance of particle sampling methods for inference-time scaling in flow models. Additionally, we demonstrate that RBF with VP-SDE achieves the best performance, outperforming all previous inference-time scaling approaches. Project page:* `flow-inference-time-scaling.github.io/`.

## 1. Introduction

Over the past years, scaling laws of AI models have mainly focused on increasing model size and training data. However, recent advancements have shifted attention toward *inference-time scaling* [49, 51], leveraging computational resources during inference to enhance model performance. OpenAI o1 [38] and DeepSeek R1 [11] exemplify this approach, demonstrating consistent output improvements with increased inference computation. Recent research in LLMs [37] attempting to replicate such improvements has introduced *test-time budget forcing*, achieving high efficiency with limited token sampling during inference.

For diffusion models [46, 48], which are widely used for generation tasks, research on inference-time scaling has been growing in the context of reward-based sampling [21, 27, 45]. Given a reward function that measures alignment with user preferences [23] or output quality [28, 44], the goal is to find the sample from the learned data distribution that best aligns with the reward through repeated sampling. Fig. 1 showcases diverse applications of inference-time scaling, enabling the generation of faithful images that accurately align with complex user descriptions involving objects quantities, logical relationships, and conceptual attributes. Notably, naïve generation from text-to-image models [25, 43] often fails to fully meet user specifications, highlighting the effectiveness of inference-time scaling.

Our goal in this work is to extend the inference-time scaling capabilities of diffusion models to flow models. Flow models [29] power state-of-the-art image [14, 25] and video generation [7, 60], achieving high-quality synthesis with few inference steps, enabled by trajectory stratification techniques during training [32]. Beyond just speed, recent pretrained flow models, equipped with enhanced text-image

embeddings [41] and advanced architectures [14], significantly outperform previous pretrained diffusion models in both image and video generation quality.

Despite their advantages in generating high-quality results more efficiently than diffusion models, flow models have an inherent limitation in the context of inference-time scaling. Due to their ODE-based deterministic generative process, they cannot directly incorporate particle sampling at intermediate steps, a key mechanism for effective inference-time scaling in diffusion models. Building on the formulation of stochastic interpolant framework [1], we adopt an SDE-based sampling method for flow models at inference-time, enabling particle sampling for reward alignment.

Furthermore, we observe that converting an ODE to its corresponding SDE in the generative process of a flow model does not provide sufficient diversity in particle sampling to effectively seek high-reward samples. To expand the exploration space, we consider not only stochasticity but also the choice of the *interpolant* itself. While typical flow models use a linear interpolant, diffusion models commonly adopt a Variance-Preserving (VP) interpolant [17, 48]. Inspired by this, for the first time, we incorporate the VP interpolant into the generative process of flow models and demonstrate its effectiveness in increasing sample diversity, ultimately enhancing the likelihood of discovering high-reward samples.

We emphasize that while we propose converting the generative process of a pretrained flow model to align with that of diffusion models—*i.e.*, VP-SDE-based generation—inference-time scaling with flow models offers significant advantages over diffusion models. Flow models, particularly those with rectification fine-tuning [32, 33], produce much clearer expected outputs at intermediate steps, enabling more precise future reward estimation and, in turn, more effective particle sampling.

We additionally explore a strategy for tight budget enforcement in terms of the number of function evaluations (NFEs) of the velocity prediction network. Previous particle-sampling-based inference-time scaling approaches for diffusion models [27, 45] allocate the NFEs budget *uniformly* across timesteps in the generative process, which we empirically found to be ineffective in practice. To optimize budget utilization, we propose *Rollover Budget Forcing*, a method that adaptively reallocates NFEs across timesteps. Specifically, we perform a denoising step upon identifying a new particle with a higher expected future reward and allocate the remaining NFEs to subsequent timesteps.

Experimentally, we demonstrate that our inference-time SDE conversion and VP interpolant conversion enable particle sampling in flow models, leading to consistent improvements in reward alignment across two challenging tasks: compositional text-to-image generation and quantity-aware image generation. Additionally, our Rollover Budget Forcing (`RBF`) provides further performance gains, outperform-

ing all previous particle sampling approaches. We also demonstrate that for differentiable rewards, such as aesthetic image generation, integrating `RBF` with a gradient-based method [8] creates a synergistic effect, leading to further performance improvements.

In summary, we introduce an inference-time scaling method for flow models, analyzing three key factors and proposing the best options for each:

- ODE vs. SDE: We introduce an *SDE-based* generative process for flow models to enable particle sampling.
- Interpolant: We demonstrate that replacing the linear interpolant of flow models with *Variance Preserving (VP) interpolant* expands the search space, facilitating the discovery of higher-reward samples.
- NFEs Allocation: We propose *Rollover Budget Forcing* that adaptively allocates NFEs across timesteps to ensure efficient utilization of the available compute budget.

## 2. Related Work

### 2.1. Reward Alignment in Diffusion Models

In the literature of diffusion models, reward alignment approaches can be broadly categorized into fine-tuning-based methods [5, 9, 39, 55, 57, 58] and inference-time-scaling-based methods [6, 12, 27, 45, 56]. While fine-tuning diffusion models enables the generation of samples aligned with user preferences, it requires fine-tuning for each task, potentially limiting scalability. In contrast, inference-time scaling approaches offer a significant advantage as they can be applied to any reward without requiring additional fine-tuning. Moreover, they can also be applied to fine-tuned models to further enhance alignment with the reward. Since our proposed approach is an inference-time scaling method, we focus our review on related literature in this domain.

Furthermore, for *differentiable* rewards, gradient-based methods [4, 8, 15, 16, 54, 59] have been extensively studied. We note that inference-time scaling can be integrated with gradient-based approaches to achieve synergistic performance improvements.

### 2.2. Particle Sampling with Diffusion Models

The simplest iterative sampling method that can be applied to any generative model is Best-of-N (BoN) [49–51], which generates $N$ batches of samples and selects the one with the highest reward. For diffusion models, however, incorporating particle sampling during the denoising process has been shown to be far more effective than naïve BoN [27, 45]. This idea has been further developed through various approaches that sample particles at each step. For instance, SVDD [27] proposed simply selecting the particle with the highest reward at every step. CoDe [45] extends this idea by selecting the highest-reward particle only at specific intervals. On the other hand, methods based on Sequential Monte Carlo (SMC) [6, 12, 21, 56] employ a probabilistic selection ap-

proach, in which particles are sampled from a multinomial distribution according to their importance weights. Despite the success of particle sampling approaches for diffusion models, they have not been applicable to flow models due to the absence of stochasticity in their generative process. In this work, we present the first inference-time scaling method for flow models based on particle sampling by introducing stochasticity into the generative process and further increasing sampling diversity through trajectory modification.

### 2.3. Inference-Time Scaling with Flow Models

To our knowledge, SoP [36], a concurrent work to ours, is the only inference-time scaling method proposed for flow models, which applies a forward kernel to sample particles from the deterministic sampling process of flow models. However, SoP does not explore the possibility of modifying the reverse kernel, which could enable more diverse particle-sampling-based methods [21, 27, 45]. To the best of our knowledge, we are the *first* to investigate the application of particle sampling to flow models through the lens of the reverse kernel.

## 3. Problem Definition and Background

### 3.1. Inference-Time Reward Alignment

Given a pretrained flow model that maps the source distribution, a standard Gaussian distribution $p_1$, into the data distribution $p_0$, our objective is to generate high-reward samples $\mathbf{x}_0 \in \mathbb{R}^d$ from the pretrained flow model without additional training–a task known as inference-time reward alignment. We denote the given reward function as $r : \mathbb{R}^d \to \mathbb{R}$, which measures text alignment or user preference for a generated sample. Following previous works [24, 52, 53], our objective can be formulated as finding the following target distribution:

$$p_0^* = \arg\max_q \mathbb{E}_{\mathbf{x}_0 \sim q} \underbrace{[r(\mathbf{x}_0)]}_{\text{Reward}} - \beta \underbrace{\mathcal{D}_{\text{KL}}[q(\mathbf{x}_0)|p_0(\mathbf{x}_0)]}_{\text{KL Regularization}}, \quad (1)$$

which maximizes the expected reward while the KL divergence term prevents $p_0^*(\mathbf{x}_0)$ from deviating too far from $p_0(\mathbf{x}_0)$, with its strength controlled by the hyperparameter $\beta$. As shown in previous work [40], the target distribution $p_0^*$ can be computed as:

$$p_0^*(\mathbf{x}_0) = \frac{1}{Z} p_0(\mathbf{x}_0) \exp\left(\frac{r(\mathbf{x}_0)}{\beta}\right), \quad (2)$$

where $Z$ is a normalization constant. We present details in Appendix A.1. However, sampling from the target distribution is non-trivial.

A notable approach for sampling from the target distribution is *particle sampling*, which maintains a set of candidate samples—referred to as particles—and iteratively propagates high-reward samples while discarding lower-reward ones. When combined with the denoising process of diffusion models, particle sampling can improve the efficiency of limited

computational resources in inference-time scaling. In the next section, we review particle sampling methods used in diffusion models and explore insights for adapting them to flow models.

## 3.2. Particle Sampling Using Diffusion Model

A pretrained diffusion model generates data by drawing an initial sample from the standard Gaussian distribution and iteratively sampling from the learned conditional distribution $p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)$. Building on this, previous works [26, 53] have shown that data from the target distribution in Eq. 2 can be generated by performing the same denoising process while replacing the conditional distribution $p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)$ with the *optimal policy*:

$$p_\theta^*(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t) = \frac{p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)\exp\left(\frac{v(\mathbf{x}_{t-\Delta t})}{\beta}\right)}{\int p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)\exp\left(\frac{v(\mathbf{x}_{t-\Delta t})}{\beta}\right)d\mathbf{x}_{t-\Delta t}}, \quad (3)$$

where the details are presented in Appendix A.2. We denote $v(\cdot) : \mathbb{R}^d \to \mathbb{R}$ as the optimal value function that estimates the expected future reward of the generated samples at current timestep. Following previous works [4, 8, 21, 27], we approximate the value function using the posterior mean computed via Tweedie's formula [42], given by $v(\mathbf{x}_t) \approx r(\mathbf{x}_{0|t})$, where $\mathbf{x}_{0|t} := \mathbb{E}_{\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0|\mathbf{x}_t)}[\mathbf{x}_0]$.

Since directly sampling from the optimal policy distribution in Eq. 3 is nontrivial, one can first approximate the distribution using importance sampling while taking $p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)$ as the *proposal distribution*:

$$p_\theta^*(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t) \approx \sum_{i=1}^K \frac{w_{t-\Delta t}^{(i)}}{\sum_{j=1}^K w_{t-\Delta t}^{(j)}} \delta_{\mathbf{x}_{t-\Delta t}^{(i)}} \quad (4)$$

$$\{\mathbf{x}_{t-\Delta t}^{(i)}\}_{i=1}^K \sim p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t),$$

where $K$ is the number of particles, $w_{t-\Delta t}^{(i)} = \exp\left(v(\mathbf{x}_{t-\Delta t}^{(i)})/\beta\right)$ is the weight, and $\delta_{\mathbf{x}_{t-\Delta t}^{(i)}}$ is a Dirac distribution. Li et al. [27] proposed an approximate sampling method for the optimal policy by selecting the sample with the largest weight from Eq. 4.

Notably, a key factor in seeking high-reward samples using particle sampling is defining the proposal distribution to sufficiently cover the distribution of high-reward samples. Consider a scenario where high-reward samples reside in a low density region of the original data distribution, which is common when generating complex or highly specific samples that deviate from the mode of the pretrained model distribution. In this case, the proposal distribution must have a large variance to effectively explore these low density regions. This highlights the importance of the *stochasticity* of the proposal distribution, which has been instrumental in the successful adoption of particle sampling in diffusion models. In contrast, flow models [29] employ a *deterministic* sampling process, where all particles $\mathbf{x}_{t-\Delta t}$ drawn from $\mathbf{x}_t$ are
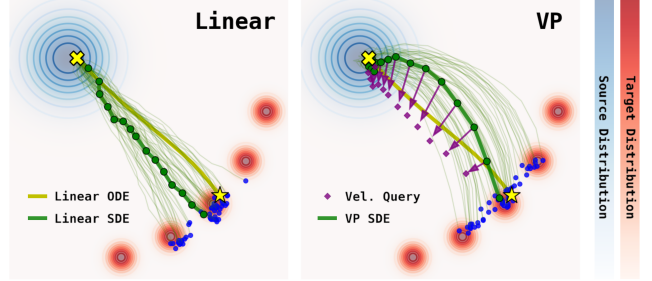


Figure 2. **Comparison of Linear-SDE and VP-SDE**. Starting from the same initial noise latent, we generate 50 samples using Linear-ODE, Linear-SDE, and VP-SDE.

identical. This restricts the applicability of particle sampling methods in flow models.

To this end, we propose an inference-time approach that introduces stochastic sampling into the generative process of flow models to enable particle sampling. We first transform the deterministic sampling process of flow models into a stochastic process (Sec. 4.2). We further expand the search space by modifying the sampling trajectory of flow models to align with that of diffusion models (Sec. 4.3).

Additionally, previous particle sampling methods in diffusion models allocated a fixed computational budget (*i.e.*, a uniform number of particles) across all denoising timesteps, potentially limiting exploration. We explore sampling with the rollover strategy, which adaptively allocates compute across timesteps during the sampling process (Sec. 5).

## 4. SDE-Based Generation Using Flow Models

In this section, we review flow and diffusion models within the unified stochastic interpolant framework (Sec. 4.1) and introduce our inference-time approaches for efficient particle sampling in flow models (Sec. 4.2 and 4.3).

### 4.1. Background: Stochastic Interpolant Framework

At the core of both diffusion and flow models is the construction of probability paths $\{p_t\}_{0\le t\le 1}$, where $\mathbf{x}_t \sim p_t$ serves as a bridge between $\mathbf{x}_1 \sim p_1$ and $\mathbf{x}_0 \sim p_0$:

$$\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \mathbf{x}_1, \quad (5)$$

where $\alpha_t$ and $\sigma_t$ are smooth functions satisfying $\alpha_0 = \sigma_1 = 1$, $\alpha_1 = \sigma_0 = 0$, and $\dot{\alpha}_t < 0, \dot{\sigma}_t > 0$; we denote the dot as a time derivative. This formulation provides a flexible choice of *interpolant* $(\alpha_t, \sigma_t)$ which determines the sampling trajectory.

### 4.2. Inference-Time SDE Conversion

Flow models [29, 32] learn the velocity field $u_t : \mathbb{R}^d \to \mathbb{R}^d$, which enables sampling of $\mathbf{x}_0$ by solving the Probability

4

Flow-ODE [48] backward in time:

$$d\mathbf{x}_t = u_t(\mathbf{x}_t)dt. \qquad (6)$$

The deterministic process in Eq. 6 accelerates the sampling process enabling few-step generation of high-fidelity samples. However, as discussed in Sec. 3.2, the deterministic nature of this sampling process limits the applicability of particle sampling in flow models.

To address this, we transform the deterministic sampling process into a stochastic process. Song et al. [48] proposed the reverse-time SDE that shares the same marginal probability densities as the deterministic process in Eq. 6:

$$d\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_t)dt + g_t d\mathbf{w}, \qquad (7)$$

$$\mathbf{f}_t(\mathbf{x}_t) = u_t(\mathbf{x}_t) - \frac{g_t^2}{2}\nabla \log p_t(\mathbf{x}_t), \qquad (8)$$

where $\mathbf{f}_t(\mathbf{x}_t)$ and $g_t$ represent the drift and diffusion coefficient, respectively, and $\mathbf{w}$ is the standard Wiener process. Ma et al. [35] have shown that $g_t$, which determines the level of stochasticity of the proposal distribution, can be freely chosen within the stochastic interpolant framework [1, 20] (details in Appendix B). Note that in the case where $g_t = 0$ the process reduces to deterministic sampling in Eq. 6.

Using the velocity $u_t(\mathbf{x}_t)$ predicted by a pretrained flow model, the score function $\nabla \log p_t(\mathbf{x}_t)$ appearing in the drift coefficient $\mathbf{f}_t(\mathbf{x}_t)$ can be computed as:

$$\nabla \log p_t(\mathbf{x}_t) = \frac{1}{\sigma_t}\frac{\alpha_t u_t(\mathbf{x}_t) - \dot{\alpha}_t \mathbf{x}_t}{\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t}. \qquad (9)$$

This enables the conversion of the deterministic sampling to stochastic sampling, which we refer to as inference-time SDE conversion. Given the drift coefficient term $\mathbf{f}_t(\mathbf{x}_t)$ and diffusion coefficient $g_t$, the proposal distribution in the discrete-time domain is derived as follows:

$$p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t - \mathbf{f}_t(\mathbf{x}_t)\Delta t, \ g_t^2 \Delta t \ \mathbf{I}). \qquad (10)$$

Since flow models utilize the linear interpolant ($\alpha_t = 1 - t, \sigma_t = t$), we refer to the generative processes of the flow models using Eq. 6 and Eq. 7 as Linear-ODE and Linear-SDE.

In Fig. 2 (left), we visualize the sampling trajectories of Linear-ODE and Linear-SDE. The samples generated using Linear-ODE are identical and collapse to a single point, restricting exploration. In contrast, Linear-SDE introduces sample variance, allowing for broader exploration and increasing the likelihood of discovering high-reward samples.

However, in practice, we empirically observe that the search space of Linear-SDE remains constrained. In Fig. 3 (a-b), we visualize images sampled from Linear-ODE and Linear-SDE using FLUX [25], respectively. As discussed previously, the particles drawn from the proposal distribution of Linear-ODE are identical. While Linear-SDE introduces variations across different particles, we find that



(a) Linear-ODE     (b) Linear-SDE     (c) VP-SDE

Figure 3. **Sample variance test using FLUX [25] under linear and Variance Preserving (VP) interpolant.** All samples share the same initial latent. Prompt: *"A steaming cup of coffee"*.

the sample variance remains insufficient for effectively exploring low-density regions. In the next section, we introduce *inference-time interpolant conversion*, which further increases the search space.

### 4.3. Inference-Time Interpolant Conversion

To address the limitation of Linear-SDE, we draw inspiration from the effective use of particle sampling in diffusion models, where we identified a key difference: the *interpolant*. While the forward process in diffusion models follows the Variance Preserving (VP) interpolant ($\alpha_t = \exp^{-\frac{1}{2}\int_0^t \beta_s ds}, \sigma_t = \sqrt{1 - \exp^{-\int_0^t \beta_s ds}}$), where $\beta_s$ is a predefined variance schedule, flow models instead follow a linear interpolant. To bridge this gap, we introduce inference-time interpolant conversion, which transforms the linear interpolant into a VP interpolant.

As shown in the previous works [20, 30], we note that given a velocity model $u_t$ based on an interpolant ($\alpha_t, \sigma_t$) (*e.g.*, linear), one can transform the vector field and generate a sample based on a new interpolant ($\bar{\alpha}_s, \bar{\sigma}_s$) (*e.g.*, VP) at inference-time. The two paths $\bar{\mathbf{x}}_s = \bar{\alpha}_s \mathbf{x}_0 + \bar{\sigma}_s \mathbf{x}_1$ and $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \mathbf{x}_1$ are connected through scale-time transformation:

$$\bar{\mathbf{x}}_s = c_s \mathbf{x}_{t_s} \quad t_s = \rho^{-1}(\bar{\rho}(s)) \quad c_s = \bar{\sigma}_s/\sigma_{t_s}, \qquad (11)$$

where $\rho(t) = \frac{\alpha_t}{\sigma_t}$ and $\bar{\rho}(s) = \frac{\bar{\alpha}_s}{\bar{\sigma}_s}$ define the signal-to-noise ratio of the original and the new interpolant, respectively.

The velocity for the new interpolant is given as:

$$\bar{u}_s(\bar{\mathbf{x}}_s) = \frac{\dot{c}_s}{c_s}\bar{\mathbf{x}}_s + c_s \dot{t}_s u_{t_s}\left(\frac{\bar{\mathbf{x}}_s}{c_s}\right), \qquad (12)$$

$$\dot{c}_s = \frac{\sigma_{t_s}\dot{\bar{\sigma}}_s - \bar{\sigma}_s \dot{\sigma}_{t_s}\dot{t}_s}{\sigma_{t_s}^2} \quad \dot{t}_s = \frac{\sigma_{t_s}^2\left(\bar{\sigma}_s\dot{\bar{\alpha}}_s - \bar{\alpha}_s\dot{\bar{\sigma}}_s\right)}{\bar{\sigma}_s^2\left(\sigma_{t_s}\dot{\alpha}_{t_s} - \alpha_{t_s}\dot{\sigma}_{t_s}\right)}.$$

Plugging the transformed velocity into the SDE in Eq. 7 after computing the score using Eq. 9 gives our inference-time interpolant conversion. Since the new trajectory follows the VP interpolant, we refer to this as VP-SDE. We visualize VP-SDE sampling in Fig. 2 (right). At inference-time, we query the velocity of the new interpolant from the original interpolant (purple arrow).

In Fig. 3 (c), we visualize the sample variance under VP-SDE using FLUX [25]. As shown in the figure, VP-SDE produces significantly more diverse samples compared to Linear-SDE, which exhibits smaller variations across particles. This property of VP-SDE effectively expands the search space, improving particle sampling in flow models.

Note that this takes the opposite approach of previous works on diffusion models, which aimed to convert the SDE into an ODE [34, 47, 48] and transform trajectories into straighter paths [20, 30] for faster generation. Importantly, while we modify the generative process of flow models to align with that of diffusion models, inference-time scaling with flow models still provides distinct advantages. The rectified trajectories of flow models [25, 32, 33] allow for a much clearer posterior mean via Tweedie's formula [42], leading to more precise future reward estimation and, in turn, more effective particle filtering. Furthermore, when generating samples using the SDE in Eq. 7, we enhance exploration by taking smaller time intervals during the early stages of the generative process, where the variance level is high, then gradually increasing the interval to accelerate generation–enabled by the few-step generation ability of flow models (details in Appendix C).

## 5. Rollover Budget Forcing

In this section, we propose a new budget-forcing particle sampling strategy to maximize the use of limited compute in inference-time scaling. To the best of our knowledge, previous inference-time scaling approaches based on particle sampling for diffusion models [27, 45] use a fixed number of particles throughout all denoising steps.

However, our analysis shows that this uniform allocation leads to inefficiency, where the NFEs required at each denoising step to obtain a sample $\mathbf{x}_{t-\Delta t}$ with a higher reward than the current sample $\mathbf{x}_t$ significantly varies across different runs. We present the analysis results in Appendix C.

This motivates us to adopt a *rollover* strategy that adaptively allocates NFEs across timesteps. First, given the total NFEs budget, the NFEs quota $Q$ is allocated uniformly across timesteps. At each timestep, if a particle $\mathbf{x}_{t-\Delta t}$ achieves a higher reward than the current sample $\mathbf{x}_t$ within the NFEs quota, we immediately proceed to the next timestep, rolling over the remaining NFEs. Otherwise, we select the particle with the highest expected future reward from the current set of particles as done by Li et al. [27]. We present the pseudocode of our method, RBF, in Appendix E. In the next section, we demonstrate the effectiveness of RBF, along with SDE conversion and interpolant conversion.

## 6. Applications

In this section, we present the experimental results of particle sampling methods for inference-time reward alignment. In Appendix, we present i) aesthetic image generation, ii) implementation details of the search algorithms, iii) additional

| L.O. | L.S. | V.S. |
|---|---|---|

*"A mouse pad has two pencils on it,
the shorter pencil is green and the longer one is not"*
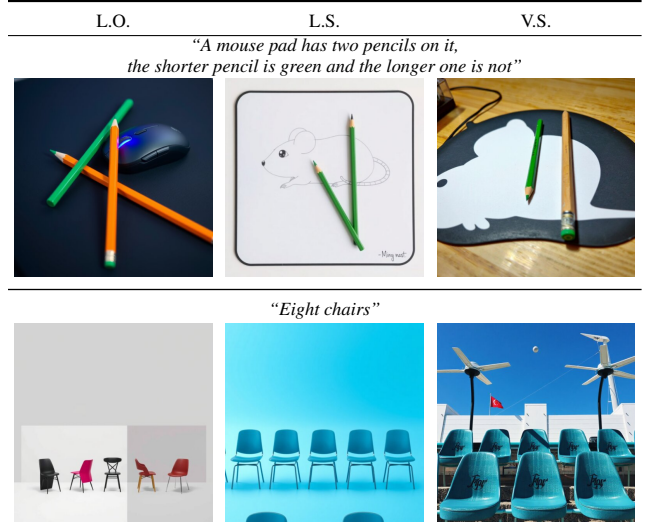


*"Eight chairs"*



Figure 4. **Qualitative results of inference-time SDE and interpolant conversion.** Employing SDE and VP interpolant conversion effectively scales model performance. Each column presents results from Linear-ODE (L.O.), Linear-SDE (L.S.), and VP-SDE (V.S.).

qualitative results, and iv) scaling comparison of Best-of-N (BoN) and RBF.

### 6.1. Experiment Setup

**Tasks.** In this section, we present the results for the following applications: compositional text-to-image generation and quantity-aware image generation, where the rewards are non-differentiable. For the differentiable reward case, we consider aesthetic image generation (Appendix D.1). In compositional text-to-image generation, we use all 121 text prompts from GenAI-Bench [19] that contain three or more advanced compositional elements. For quantity-aware image generation, we use 100 randomly sampled prompts from T2I-CompBench++ [18] numeracy category.

For all applications, we use FLUX [25] as the pretrained flow model. We fix the total number of function evaluations (NFEs) to 500 and set the number of denoising steps to 10, which allocates 50 NFEs per denoising step. For comparison with diffusion models, we use Stable Diffusion 2 [43]. For diffusion models, we additionally report a case where the number of denoising steps is set to 50 (totaling 2, 500 NFEs), allocating five times the compute budget compared to the flow model. As a reference, we also include the results of the base pretrained models without inference-time scaling.

**Baselines.** We evaluate inference-time search algorithms discussed in Sec. 2, including Best-of-N (BoN), Search over Paths (SoP) [36], SMC [21], CoDe [45], and SVDD [27]. We categorize BoN and SoP as Linear-ODE-based methods, as their generative processes follow the deterministic process in Eq. 6. For SMC, we adopt DAS [21]; however, when the reward is non-differentiable, we use the reverse transition kernel of the pretrained model as the proposal distribution.

Table 1. **Quantitative results of compositional image generation.** [†] denotes the given reward used in inference-time scaling. For particle-based sampling methods, the relative improvement in each cell is computed with respect to the Linear-ODE case. The best result in the given and held-out reward is highlighted by **bold**, and the runner-up is underlined. L.O., L.S., and V.S. indicate Linear-ODE, Linear-SDE, and VP-SDE, respectively. [*] denotes results with 50 denoising steps.

| | Diffusion Model | | | | Flow Model | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | SD2 | SD2* | SVDD [27] SD2 | SVDD [27] SD2* | FLUX | BoN | SoP [36] | SMC [21] L.O. | SMC [21] L.S. | SMC [21] V.S. | CoDe [45] L.O. | CoDe [45] L.S. | CoDe [45] V.S. | SVDD [27] L.O. | SVDD [27] L.S. | SVDD [27] V.S. | RBF (Ours) L.O. | RBF (Ours) L.S. | RBF (Ours) V.S. |
| | V.S. | | V.S. | | L.O. | | | | | | | | | | | | | | |
| VQAScore[†] [28] ↑ | 0.671 | 0.647 | 0.867 | 0.886 | 0.726 | 0.879 | 0.844 | 0.841 | 0.862 +2.50% | 0.877 +4.28% | 0.788 | 0.887 +12.56% | 0.914 +15.99% | 0.788 | 0.893 +13.32% | 0.915 +16.12% | 0.788 | 0.900 +14.21% | **0.925** +17.39% |
| Inst. BLIP [10] ↑ (held-out) | 0.741 | 0.739 | 0.790 | 0.799 | 0.775 | 0.820 | 0.799 | 0.815 | 0.811 −0.49% | 0.836 +2.58% | 0.786 | 0.815 +3.69% | 0.839 +6.74% | 0.789 | 0.813 +3.04% | **0.847** +7.35% | 0.789 | 0.813 +3.04% | 0.843 +6.84% |
| Aesthetic Score [44] ↑ | 5.107 | 5.115 | 5.181 | 5.213 | 5.338 | 5.162 | 5.254 | 5.233 | 5.016 | 5.156 | 5.190 | 5.023 | 5.177 | 5.200 | 5.052 | 5.249 | 5.200 | 5.072 | 5.237 |

| BoN | SoP [36] | SMC [21] | CoDe [45] | SVDD [27] | **RBF** (Ours) |
|---|---|---|---|---|---|

*"Three mugs are placed side by side; the two closest to the faucet each contain a toothbrush, while the one furthest away is empty."*



*"Five ants are carrying biscuits, and an ant that is not carrying biscuits is standing on a green leaf directing them."*
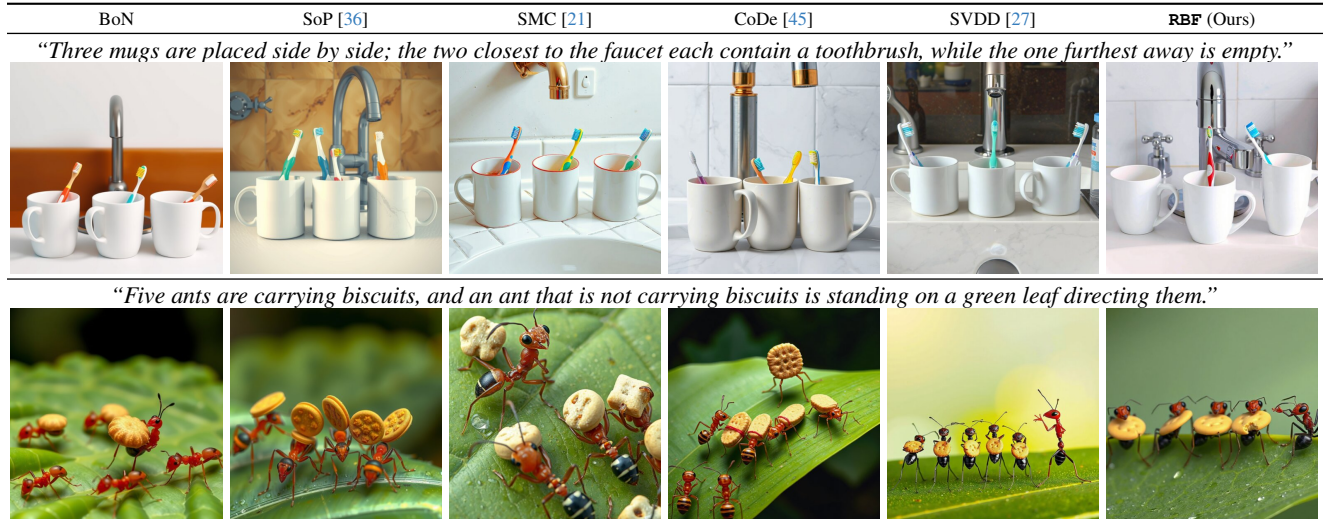


Figure 5. **Qualitative results of compositional text-to-image generation.** At inference-time, we guide the generation process using VQAScore [28] as the given reward, which evaluates the **image-text alignment**.

## 6.2. Compositional Text-to-Image Generation

**Evaluation Metrics.** In this work, we refer to the reward used for inference-time scaling as the given reward. Here, the given reward is VQAScore, measured with CLIP-FlanT5 [28], which evaluates text-image alignment. For the held-out reward, which is not used during inference, we evaluate the score using a different model, InstructBLIP [10]. Additionally, we evaluate aesthetic score [44] to assess the quality of the generated images.

**Inference-Time SDE and Interpolant Conversion.** The quantitative and qualitative results of compositional text-to-image generation are presented in Tab. 1 and Fig. 5, respectively. As discussed in Sec. 4.2, the deterministic sampling process of flow models limits the efficiency of particle sampling. The results in Tab. 1 support this, showing that Linear-SDE consistently improves the given reward over the Linear-ODE case for all particle sampling methods. However, the limited search space of Linear-SDE, as discussed in Sec. 4.3, leads to suboptimal particle sampling performance, falling behind Linear-ODE-based sampling meth-

ods, BoN and SoP [36], in the held-out reward. Through inference-time interpolant conversion, VP-SDE further improves performance across all particle sampling methods. Notably, particle sampling methods generate high-reward samples without significantly compromising image quality, as reflected in the aesthetic score, which remains comparable to the base FLUX model [25]. Lastly, we note that the quality of the pretrained Stable Diffusion 2 (SD2) [43] falls behind that of FLUX [25], even with 5× the compute budget (SD2*), highlighting the importance of adopting particle sampling through flow models.

Qualitatively, SDE conversion and interpolant conversion shown in Fig. 4 bring consistent performance improvements. Note that these results are obtained by using SVDD [27]; however, a similar trend is observed across all other particle sampling methods (see Appendix G.1).

**Rollover Budget Forcing.** As discussed in Sec. 5, instead of fixing the number of particles throughout the denoising process, we explore adaptive budget allocation through RBF. Quantitatively, we demonstrate that budget forcing provides additional performance improvements, outperforming all

Table 2. **Quantitative results of quantity-aware image generation.** [†] denotes the given reward used in inference-time scaling. For particle-based sampling methods, the relative improvement in each cell is computed with respect to the Linear-ODE case. The best result in the given and held-out reward is highlighted by **bold**, and the runner-up is underlined. L.O., L.S., and V.S. indicate Linear-ODE, Linear-SDE, and VP-SDE, respectively. [*] denotes results with 50 denoising steps.

| Metric | Diffusion Model | | | | Flow Model | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SD2 | SD2* | SVDD [27] SD2 | SD2* | FLUX | BoN | SoP [36] | SMC [21] | | | CoDe [45] | | | SVDD [27] | | | **RBF** (Ours) | | |
| | V.S. | | V.S. | | L.O. | | | L.O. | L.S. | V.S. | L.O. | L.S. | V.S. | L.O. | L.S. | V.S. | L.O. | L.S. | V.S. |
| RSS[†] [31] ↓ | 22.200 | 18.590 | 5.820 | 5.140 | 11.430 | 1.760 | 3.460 | 4.870 | 1.140 +76.59% | 1.070 +78.03% | 11.740 | 1.400 +88.07% | 1.090 +90.72% | 11.740 | 1.140 +90.29% | 0.840 +92.84% | 11.740 | 0.890 +92.42% | **0.540** +95.40% |
| Accuracy [31] ↑ | 0.020 | 0.050 | 0.470 | 0.470 | 0.130 | 0.580 | 0.470 | 0.530 | 0.630 +18.87% | 0.620 +16.98% | 0.090 | 0.650 +622.22% | 0.650 +622.22% | 0.090 | 0.660 +633.33% | 0.750 +733.33% | 0.090 | 0.700 +677.78% | **0.800** +788.89% |
| VQAScore [28] ↑ (held-out) | 0.523 | 0.532 | 0.617 | 0.648 | 0.648 | 0.756 | 0.740 | 0.731 | 0.768 +5.06% | 0.756 +3.42% | 0.633 | <u>0.772</u> +21.96% | **0.778** +22.91% | 0.633 | 0.761 +20.22% | 0.765 +20.85% | 0.633 | 0.761 +20.22% | 0.769 +21.48% |
| Aesthetic Score [44] ↑ | 5.060 | 5.223 | 4.983 | 5.081 | 5.182 | 5.420 | 5.676 | 5.464 | 5.304 | 5.506 | 5.179 | 5.471 | 5.571 | 5.179 | 5.447 | 5.576 | 5.179 | 5.343 | 5.581 |

| BoN | SoP [36] | SMC [21] | CoDe [45] | SVDD [27] | **RBF** (Ours) |
|---|---|---|---|---|---|

*"Seven balloons, four bears and four swans."*



*"Six horses and six deer and four balloons."*
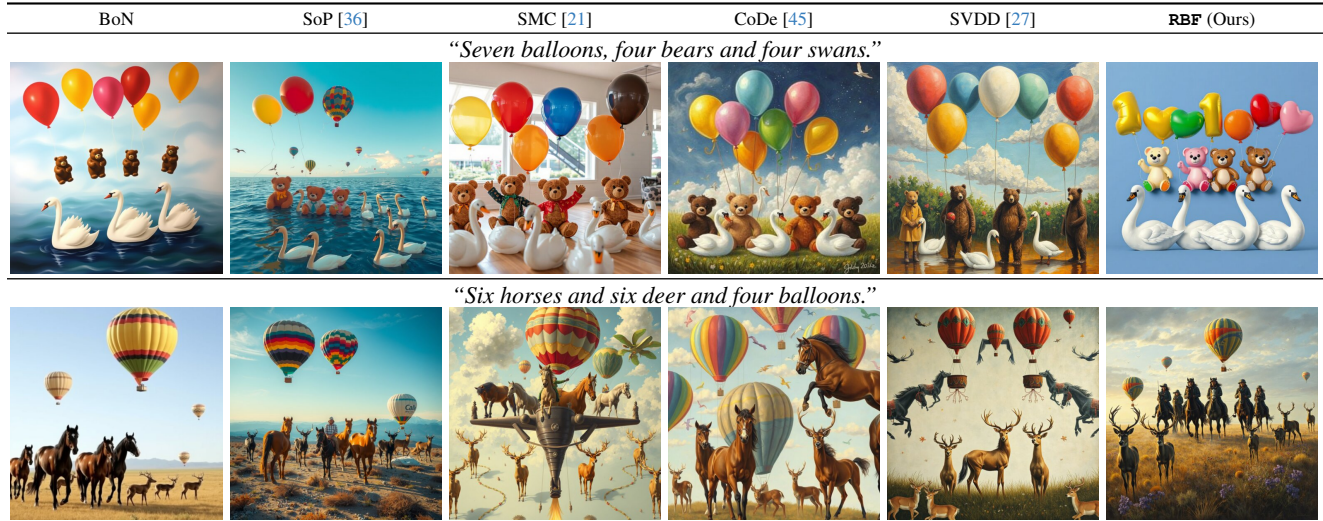


Figure 6. **Qualitative results of quantity-aware image generation.** At inference-time, we guide generation using the negation of RSS (Residual Sum of Squares) as the given reward, which measures **the discrepancy between detected and target object counts.**

other particle sampling methods in the given reward. Qualitatively, Fig. 5 shows that our method, RBF efficiently searches for high-reward samples that best align with the given text prompt (see Appendix G.2 for additional qualitative results).

## 6.3. Quantity-Aware Image Generation

**Evaluation Metrics.** Here, the given reward is the negation of the Residual Sum of Squares (RSS) between the target counts and the detected object counts, computed using GroundingDINO [31] and SAM [22] (details in Appendix F). Additionally, we report object count accuracy, which evaluates whether all object quantities are correctly shown in the image. For the held-out reward, we report VQAScore measured with CLIP-FlanT5 [28]. As in the previous application, we evaluate the quality of the generated images using the aesthetic score [44].

**Results.** The quantitative and qualitative results of quantity-aware image generation are presented in Tab. 2 and Fig. 6, respectively. The trend in Tab. 2 align with those in Sec. 6.2, demonstrating that SDE conversion and interpolant conversion synergistically enhance the identification

of high-reward samples. Notably, particle sampling methods with Linear-SDE already outperform Linear-ODE-based methods, while interpolant conversion further enhances accuracy, achieving a $4 \sim 5\times$ improvement over the base FLUX model [25]. Notably, our RBF achieves the highest accuracy, outperforming all other particle-based sampling methods. Qualitatively, Fig. 6 shows that RBF effectively identifies high-reward samples that precisely match the specified object categories and quantities (additional qualitative results are presented in Appendix G.2).

## 7. Conclusion

We introduced a novel inference-time scaling method for flow models with three key contributions: (1) ODE-to-SDE conversion for particle sampling, (2) Linear-to-VP interpolant conversion for enhanced diversity and search efficiency, and (3) Rollover Budget Forcing (RBF) for adaptive compute allocation. We demonstrated the effectiveness of VP-SDE-based generation in applying off-the-shelf particle sampling to flow models and showed that our RBF combined with VP-SDE generation outperforms previous methods.

## Acknowledgments

## References

[1] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv*, 2023. 2, 5, 13

[2] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 1982. 14

[3] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023. 1

[4] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *CVPRW*, 2023. 3, 4

[5] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv*, 2024. 3, 16

[6] Gabriel Cardoso, Yazid Janati El Idrissi, Sylvain Le Corff, and Eric Moulines. Monte carlo guided diffusion for bayesian linear inverse problems. In *ICLR*, 2024. 3

[7] Shoufa Chen, Chongjian Ge, Yuqi Zhang, Yida Zhang, Fengda Zhu, Hao Yang, Hongxiang Hao, Hui Wu, Zhichao Lai, Yifei Hu, Ting-Che Lin, Shilong Zhang, Fu Li, Chuan Li, Xing Wang, Yanghua Peng, Peize Sun, Ping Luo, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Goku: Flow based video generative foundation models. *arXiv preprint arXiv:2502.04896*, 2025. 2

[8] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *ICLR*, 2023. 3, 4, 15, 16

[9] Kevin Clark, Paul Vicol, Kevin Swersky, and Fleet David J. Directly fine-tuning diffusion models on differentiable rewards. In *ICLR*, 2024. 3

[10] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. In *NeurIPS*, 2023. 7, 19

[11] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. 2

[12] Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *ICLR*, 2024. 3

[13] Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. *Sequential Monte Carlo methods in practice*. Springer, 2001. 17

[14] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis, 2024. 2024. 2

[15] Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. Reno: Enhancing one-step text-to-image models through reward-based noise optimization. In *NeurIPS*, 2024. 3

[16] Xiefan Guo, Jinlin Liu, Miaomiao Cui, Jiankai Li, Hongyu Yang, and Di Huang. InitNO: Boosting text-to-image diffusion models via initial noise optimization. In *CVPR*, 2024. 3

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2

[18] Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2I-CompBench++: An Enhanced and Comprehensive Benchmark for Compositional Text-to-Image Generation . *IEEE Transactions on Pattern Analysis Machine Intelligence*, 2025. 6, 16

[19] Dongfu Jiang, Max Ku, Tianle Li, Yuansheng Ni, Shizhuo Sun, Rongqi Fan, and Wenhu Chen. Genai arena: An open evaluation platform for generative models. *arXiv*, 2024. 6

[20] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. 2022. 5, 6

[21] Sunwoo Kim, Minkyu Kim, and Dongmin Park. Test-time alignment of diffusion models without reward over-optimization. In *ICLR*, 2025. 2, 3, 4, 6, 7, 8, 13, 17, 21, 23, 24

[22] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *ICCV*, 2023. 8, 19, 20

[23] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *NIPS*, 2023. 2

[24] Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetmant. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. In *NeurIPS*, 2022. 3

[25] Black Forest Labs. FLUX. https://github.com/black-forest-labs/flux, 2024. 1, 2, 5, 6, 7, 8, 15

[26] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv*, 2018. 4

[27] Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Aviv Regev, Sergey Levine, and Masatoshi Uehara. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv*, 2024. 2, 3, 4, 6, 7, 8, 13, 15, 16, 18, 19, 22, 23, 24

[28] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. *arXiv*, 2024. 1, 2, 7, 8, 16, 19

[29] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023. 2, 4

[30] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024. 5, 6

[31] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *ECCV*, 2024. 1, 8, 16, 19, 20

[32] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023. 2, 4, 6

[33] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and qiang liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *ICLR*, 2024. 2, 6

[34] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *NIPS*, 2022. 6

[35] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *ECCV*, 2024. 5, 13

[36] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv*, 2025. 3, 6, 7, 8, 17, 23, 24

[37] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025. 2

[38] OpenAI. Learning to Reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/, 2024. 2

[39] Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv*, 2023. 3

[40] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NIPS*, 2023. 3

[41] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020. 2

[42] Herbert E. Robbins. *An Empirical Bayes Approach to Statistics*. Springer, 1992. 4, 6

[43] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 6, 7

[44] C. Schuhmann. Laion aesthetics. https://laion.ai/blog/laion-aesthetics, 2022. 1, 2, 7, 8, 15, 16

[45] Anuj Singh, Sayak Mukherjee, Ahmad Beirami, and Hadi Jamali-Rad. Code: Blockwise control for denoising diffusion models. *arXiv*, 2025. 2, 3, 6, 7, 8, 15, 18, 21, 23, 24

[46] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. 2015. 2

[47] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *ICLR*, 2021. 6

[48] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021. 2, 5, 6

[49] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. In *NeurIPS*, 2020. 2, 3, 17

[50] Luming Tang, Nataniel Ruiz, Chu Qinghao, Yuanzhen Li, Aleksander Holynski, David E Jacobs, Bharath Hariharan, Yael Pritch, Neal Wadhwa, Kfir Aberman, and Michael Rubinstein. Realfill: Reference-driven generation for authentic image completion. *ACM TOG*, 2024. 17

[51] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv*, 2023. 2, 3

[52] 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*, 2024. 3, 13

[53] Masatoshi Uehara, Yulai Zhao, Ehsan Hajiramezanali, Gabriele Scalia, Gökcen Eraslan, Avantika Lal, Sergey Levine, and Tommaso Biancalani. Bridging model-based optimization and generative modeling via conservative fine-tuning of diffusion models. In *NeurIPS*, 2024. 3, 4, 13, 16

[54] Bram Wallace, Akash Gokul, Stefano Ermon, and Nikhil Naik. End-to-end diffusion latent optimization improves classifier guidance. In *ICCV*, 2023. 3

[55] 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 *CVPR*, 2024. 3

[56] Luhuan Wu, Brian L Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. In *NeurIPS*, 2023. 3

[57] 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. In *NeurIPS*, 2023. 3, 15, 16

[58] Kai Yang, Jian Tao, Jiafei Lyu, Chunjiang Ge, Jiaxin Chen, Qimai Li, Weihan Shen, Xiaolong Zhu, and Xiu Li. Using human feedback to fine-tune diffusion models without any reward model. In *CVPR*, 2024. 3

[59] Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *ICCV*, 2023. 3

[60] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, 2024. 2

# Appendix

## A. Proofs

### A.1. Derivation of the Target Distribution

From Eq. 1, we obtain the target distribution $p_0^*$, which maximizes the reward while maintaining proximity to the distribution of the pretrained model $p_0$:

$$p_0^*(\mathbf{x}_0) = \arg\max_q \mathbb{E}_{\mathbf{x}_0 \sim q} \left[ r(\mathbf{x}_0) \right] - \beta \mathcal{D}_{\text{KL}} \left[ q(\mathbf{x}_0) | p_0(\mathbf{x}_0) \right],$$

$$= \arg\max_q \mathbb{E}_{\mathbf{x}_0 \sim q} \left[ r(\mathbf{x}_0) - \beta \log \frac{q(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} \right]$$

$$= \arg\min_q \mathbb{E}_{\mathbf{x}_0 \sim q} \left[ \log \frac{q(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} - \frac{1}{\beta} r(\mathbf{x}_0) \right]$$

$$= \arg\min_q \int q(\mathbf{x}_0) \log \frac{q(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} \mathrm{d}\mathbf{x}_0 - \frac{1}{\beta} \int q(\mathbf{x}_0) r(\mathbf{x}_0) \mathrm{d}\mathbf{x}_0.$$

This can be solved via calculus of variation where the functional $\mathcal{J}$ is given as follows:

$$\mathcal{J}\left[ q(\mathbf{x}_0) \right] := \int q(\mathbf{x}_0) \left( \log \frac{q(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} - \frac{1}{\beta} r(\mathbf{x}_0) \right) \mathrm{d}\mathbf{x}_0.$$

Substituting $\tilde{q}(\mathbf{x}_0, \epsilon) := q(\mathbf{x}_0) + \epsilon \eta(\mathbf{x}_0)$ gives:

$$\mathcal{J}\left[ \tilde{q}(\mathbf{x}_0, \epsilon) \right] = \int \tilde{q}(\mathbf{x}_0, \epsilon) \left( \log \frac{\tilde{q}(\mathbf{x}_0, \epsilon)}{p_0(\mathbf{x}_0)} - \frac{1}{\beta} r(\mathbf{x}_0) \right) \mathrm{d}\mathbf{x}_0,$$

where $\eta(\mathbf{x}_0)$ is an arbitrary smooth function, and $\epsilon$ is a scalar parameter.

Introducing a Lagrange multiplier $\lambda$ to constraint $\int q(\mathbf{x}_0) \mathrm{d}\mathbf{x}_0 = 1$ gives:

$$\mathcal{J}\left[ \tilde{q}(\mathbf{x}_0, \epsilon) \right] = \int \tilde{q}(\mathbf{x}_0, \epsilon) \left( \log \frac{\tilde{q}(\mathbf{x}_0, \epsilon)}{p_0(\mathbf{x}_0)} - \frac{1}{\beta} r(\mathbf{x}_0) \right) + \lambda \tilde{q}(\mathbf{x}_0, \epsilon) \mathrm{d}\mathbf{x}_0$$

$$:= \int f\{\tilde{q}; \mathbf{x}_0\} \mathrm{d}\mathbf{x}_0$$

Then the problem boils down to finding a function $\tilde{q}(\mathbf{x}_0, \epsilon)$ satisfying:

$$\left. \frac{\partial \mathcal{J}}{\partial \epsilon} \right|_{\epsilon=0} = 0$$

This can be solved using the Euler-Lagrange equation:

$$\frac{\partial f}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}\mathbf{x}_0} \frac{\partial f}{\partial q'} = 0,$$

where $q'$ is a derivative of $q$ with respect to $\mathbf{x}_0$ and tilde notation is dropped since the condition is to be satisfied at $\epsilon = 0$.

Note that $q'$ does not appear in $f$, so the Euler-Lagrange equation simplifies to:

$$\frac{\partial f}{\partial q} = \frac{\partial}{\partial q} \left( q(\mathbf{x}_0) \left( \log \frac{q(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} - \frac{1}{\beta} r(\mathbf{x}_0) \right) + \lambda q(\mathbf{x}_0) \right) = 0$$

$$= \log \frac{q(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} - \frac{1}{\beta} r(\mathbf{x}_0) + 1 + \lambda = 0. \tag{13}$$

Solving Eq. 13 gives the target distribution $p_0^*$, which minimizes the objective function in Eq. 1:

$$p_0^*(\mathbf{x}_0) = p_0(\mathbf{x}_0) \exp\left( \frac{r(\mathbf{x}_0)}{\beta} - 1 - \lambda \right) \tag{14}$$

Lastly, the Lagrangian multiplier $\lambda$ is obtained from the normalization constraint, $\exp(\lambda) = \int p_0(\mathbf{x}_0) \exp\left(\frac{r(\mathbf{x}_0)}{\beta} - 1\right) d\mathbf{x}_0$. Plugging this into Eq. 14 gives the target distribution presented in Eq. 2:

$$p_0^*(\mathbf{x}_0) = \frac{p_0(\mathbf{x}_0) \exp\left(\frac{r(\mathbf{x}_0)}{\beta}\right)}{\int p_0(\mathbf{x}_0) \exp\left(\frac{r(\mathbf{x}_0)}{\beta}\right) d\mathbf{x}_0}, \tag{15}$$

### A.2. Derivation of the Optimal Policy

Here, we provide the derivations of the optimal policy given in Eq. 3 for completeness, which is proposed in previous works [52, 53].

To sample from the target distribution defined in Eq. 15, previous studies utilize an optimal policy $p_\theta^*(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)$. The optimal value function $v(\mathbf{x}_t)$ is defined as the expected future reward at current timestep $t$:

$$v(\mathbf{x}_t) = \beta \log \mathbb{E}_{\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0|\mathbf{x}_t)}\left[\exp\left(\frac{r(\mathbf{x}_0)}{\beta}\right)\right] \tag{16}$$

The optimal policy is the policy that maximizes the objective function:

$$p_\theta^*(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t) = \arg\max_q \mathbb{E}_q\left[v(\mathbf{x}_{t-\Delta t})\right] - \beta \mathcal{D}_{KL}\left[q(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)\|p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)\right]$$

$$= \frac{p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)\exp\left(\frac{1}{\beta}v(\mathbf{x}_{t-\Delta t})\right)}{\int p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)\exp\left(\frac{1}{\beta}v(\mathbf{x}_{t-\Delta t})\right)d\mathbf{x}_{t-\Delta t}} \tag{17}$$

$$= \frac{p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)\exp\left(\frac{1}{\beta}v(\mathbf{x}_{t-\Delta t})\right)}{\exp\left(\frac{1}{\beta}v(\mathbf{x}_t)\right)} \tag{18}$$

where the last equality follows from the soft-Bellman equations [53]. For completeness, we present the theorem.

**Theorem 1.** (Theorem 1 of Uehara et al. [53]). *The induced distribution of the optimal policy in Eq. 17 is the target distribution in Eq. 15.*

$$p_0^*(\mathbf{x}_0) = \int \left\{ p_1(\mathbf{x}_1) \prod_{s=T}^{1} p_\theta^*(\mathbf{x}_{\frac{s}{T}-\frac{1}{T}}|\mathbf{x}_{\frac{s}{T}}) \right\} d\mathbf{x}_{\frac{1}{T}:1}.$$

However, computing the optimal value function in Eq. 16 is non-trivial. Hence, we follow the previous works [21, 27] and approximate it using the posterior mean $\mathbf{x}_{0|t} := \mathbb{E}_{\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0|\mathbf{x}_t)}[\mathbf{x}_0]$:

$$v(\mathbf{x}_t) = \beta \log\left(\int \exp\left(\frac{r(\mathbf{x}_0)}{\beta}\right) p_\theta(\mathbf{x}_0|\mathbf{x}_t) d\mathbf{x}_0\right)$$

$$\approx \beta \log\left(\exp\left(\frac{r(\mathbf{x}_{0|t})}{\beta}\right)\right) = r(\mathbf{x}_{0|t}). \tag{19}$$

### B. Choice of Diffusion Coefficient

Ma et al. [35] have shown that the diffusion coefficient can be chosen freely within the stochastic interpolant framework [1]. We present detailed derivations for completeness. Here, we use $\mathbf{w}$ interchangeably to denote the standard Wiener process for both forward and reverse time flows.

**Proposition 1.** *For a linear stochastic process $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \mathbf{x}_1$ and the Probability-Flow ODE $d\mathbf{x}_t = u_t(\mathbf{x}_t)dt$ that yields the marginal density $p_t(\mathbf{x}_t)$, the following forward and reverse SDEs with an arbitrary diffusion coefficient $g_t \geq 0$ share the same marginal density:*

$$\text{Forward SDE: } d\mathbf{x}_t = \left[u_t(\mathbf{x}_t) + \frac{g_t^2}{2}\nabla \log p_t(\mathbf{x}_t)\right]dt + g_t d\mathbf{w} \tag{20}$$

$$\text{Reverse SDE: } d\mathbf{x}_t = \left[u_t(\mathbf{x}_t) - \frac{g_t^2}{2}\nabla \log p_t(\mathbf{x}_t)\right]dt + g_t d\mathbf{w}. \tag{21}$$

*Proof.* When velocity field $u_t$ generates a probability density path $p_t$, it satisfies the continuity equation:

$$\frac{\partial}{\partial t} p_t(\mathbf{x}_t) = -\nabla \cdot (p_t(\mathbf{x}_t) u_t(\mathbf{x}_t)). \tag{22}$$

Similarly, for the SDE $d\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_t)dt + g_t d\mathbf{w}$, the Fokker-Planck equation describes the time evolution of $\tilde{p}_t$:

$$\frac{\partial}{\partial t} \tilde{p}_t(\mathbf{x}_t) = -\nabla \cdot (\tilde{p}_t(\mathbf{x}_t)\mathbf{f}_t(\mathbf{x}_t)) + \frac{1}{2} g_t^2 \nabla^2 \tilde{p}_t(\mathbf{x}_t) \tag{23}$$

where $\nabla^2$ denotes the Laplace operator.

To find an SDE that yields the same marginal probability density as the ODE, we equate the probability density functions in Eq. 23 and Eq. 22, resulting in the following equation:

$$-\nabla \cdot (p_t(\mathbf{x}_t)\mathbf{f}_t(\mathbf{x}_t)) + \frac{1}{2} g_t^2 \nabla^2 p_t(\mathbf{x}_t) = -\nabla \cdot (p_t(\mathbf{x}_t) u_t(\mathbf{x}_t))$$

$$\nabla \cdot (p_t(\mathbf{x}_t)(\mathbf{f}_t(\mathbf{x}_t) - u_t(\mathbf{x}_t))) = \frac{1}{2} g_t^2 \nabla^2 p_t(\mathbf{x}_t) \tag{24}$$

This implies that any SDE with drift coefficient $\mathbf{f}_t(\mathbf{x}_t)$ and diffusion coefficient $g_t$ that satisfies Eq. 24 will generate $p_t$. One particular choice is to set $p_t(\mathbf{x}_t)(\mathbf{f}_t(\mathbf{x}_t) - u_t(\mathbf{x}_t))$ proportional to $\nabla p_t(\mathbf{x}_t)$, *i.e.*, $p_t(\mathbf{x}_t)(\mathbf{f}_t(\mathbf{x}_t) - u_t(\mathbf{x}_t)) = A_t \nabla p_t(\mathbf{x}_t)$. Then Eq. 24 can be rewritten as:

$$A_t \nabla^2 p_t(\mathbf{x}_t) = \frac{1}{2} g_t^2 \nabla^2 p_t(\mathbf{x}_t),$$

which leads to the relation $A_t = \frac{1}{2} g_t^2$. Similarly, the drift coefficient is given by:

$$\mathbf{f}_t(\mathbf{x}_t) = u_t(\mathbf{x}_t) + \frac{1}{2} g_t^2 \frac{\nabla p_t(\mathbf{x}_t)}{p_t(\mathbf{x}_t)}$$

$$= u_t(\mathbf{x}_t) + \frac{1}{2} g_t^2 \nabla \log p_t(\mathbf{x}_t)$$

Thus, a family of SDEs that generate $p_t$ takes the following form:

$$d\mathbf{x}_t = \left[ u_t(\mathbf{x}_t) + \frac{1}{2} g_t^2 \nabla \log p_t(\mathbf{x}_t) \right] dt + g_t d\mathbf{w},$$

which is the forward SDE presented in Eq. 20. Similarly, the reverse SDE in Eq. 21 can be derived by applying the time reversal formula, following Anderson [2]. □

In the experiment, for all SDEs, we use $g_t = t^2$, with its norm scaled by a factor of 3, which we found to work well for all inference-time search algorithms.

## C. Adaptive Time Scheduling and Rollover Strategy

In this section, we provide details of adaptive time scheduling and NFE analysis result which inspired rollover strategy.

**Adaptive Time Scheduling.** As discussed in Sec. 4.3, to maximize the exploration space in VP-SDE sampling, we design the time scheduler to take smaller steps during the initial phase—when variance is high—and gradually increase the step size in later stages. Specifically, we define the time scheduler as $t_{\text{new}} = \sqrt{1 - (1 - t)^2}$. While this approach can be problematic when the number of steps is too low—resulting in excessively large discretization steps in later iterations—we find that using a reasonable number of steps (*e.g.*, 10) works well in practice, benefiting from the few-step generation capability of flow models. This setup effectively balances a broad exploration space with fast inference time, highlighting one of the key advantages of flow models over diffusion models.
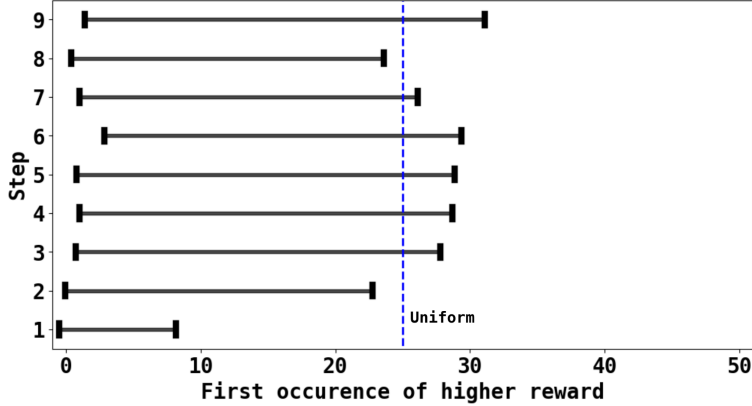
Figure 7. **Analysis of number of function evaluations (NFEs) across timesteps.** The NFEs required to achieve a higher reward for each timestep. The plot illustrates the $\pm 1$ sigma variation band. The blue-dotted line represents the uniform allocation of compute (NFEs) across timesteps. We observe that the NFEs required to identify a higher-reward sample may exceed the uniformly allocated budget (blue dotted line).

Table 3. **Quantitative results of aesthetic image generation.** [†] denotes the given reward used in inference time. The relative improvement in each cell is computed with respect to the base model. The best result in each row is highlighted in **bold**.

| Metric | FLUX [25] | DPS [8] | SVDD [27] +DPS [8] | **RBF** (Ours) +DPS [8] |
|---|---|---|---|---|
| Aesthetic Score[†] [44] | 5.795 | 6.438 +11.10% | 6.887 +18.85% | **7.170** +23.73% |
| ImageReward [57] (held-out) | 0.991 | 0.605 −38.97% | 1.077 +8.61% | **1.152** +16.20% |

| FLUX [25] | DPS [8] | SVDD [27] + DPS [8] | **RBF** (Ours) + DPS [8] |
|---|---|---|---|

*"Bird"*



*"Bat"*
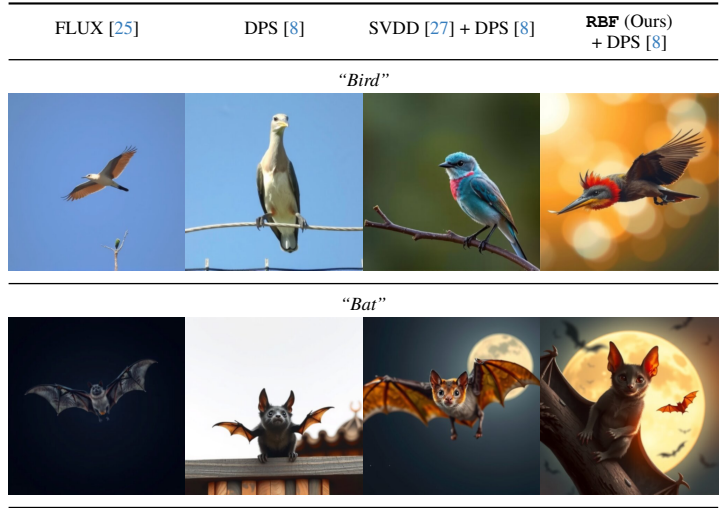


Figure 8. **Qualitative results of aesthetic image generation.** At inference-time, we guide generate using the aesthetic score [44] as the given reward, which assesses **visual appeal**.

**NFE Analysis.** As discussed in Sec. 5, we analyze the number of function evaluations (NFEs) required to obtain a sample with a higher reward than the current one. In Fig. 7, we visualize the variance band of the required NFEs across timesteps, with the blue-dotted line representing the uniform allocation used in previous particle sampling methods [27, 45]. Notably, uniform compute allocation may constrain exploration and fail to identify high-reward samples, as evidenced by crossings within the variance band. This observation motivates the use of a rollover strategy to optimize compute utilization efficiently. As demonstrated in Sec. 6, our experiments confirm that RBF provides additional improvements over previous particle-sampling-based methods [27, 45].

# D. Additional Results

## D.1. Aesthetic Image Generation

In this section, we demonstrate that inference-time scaling can also be applied to gradient-based methods, such as DPS [8], for differentiable rewards. Specifically, we consider aesthetic image generation and show that RBF leads to synergistic performance improvements. We first derive the formulation of the proposal distribution for differentiable rewards and then present qualitative and quantitative results.

Table 4. **Quantitative results of quantity-aware image generation in NFE scaling expriment.** We use the same 100 prompts from T2I-CompBench [18] as in the quantity-aware image generation task. $^\dagger$ denotes the given reward used in the inference-time.

| | NFEs | RSS$^\dagger$ [31] ↓ | Accuracy ↑ | VQAScore [28] (held-out) ↑ | Aesthetic Score [44] ↑ |
|---|---|---|---|---|---|
| BoN | 50 | 4.360 | 0.400 | 0.758 | 5.408 |
| | 100 | 3.280 | 0.510 | 0.750 | 5.522 |
| | 300 | 2.190 | 0.570 | 0.755 | 5.463 |
| | 500 | 1.760 | 0.580 | 0.756 | 5.420 |
| | 1000 | 1.340 | 0.590 | 0.759 | 5.466 |
| RBF (Ours) | 50 | 3.250 | 0.410 | 0.756 | 5.560 |
| | 100 | 1.860 | 0.590 | 0.760 | 5.627 |
| | 300 | 0.690 | 0.720 | 0.779 | 5.503 |
| | 500 | 0.540 | 0.800 | 0.769 | 5.581 |
| | 1000 | 0.290 | 0.880 | 0.777 | 5.526 |
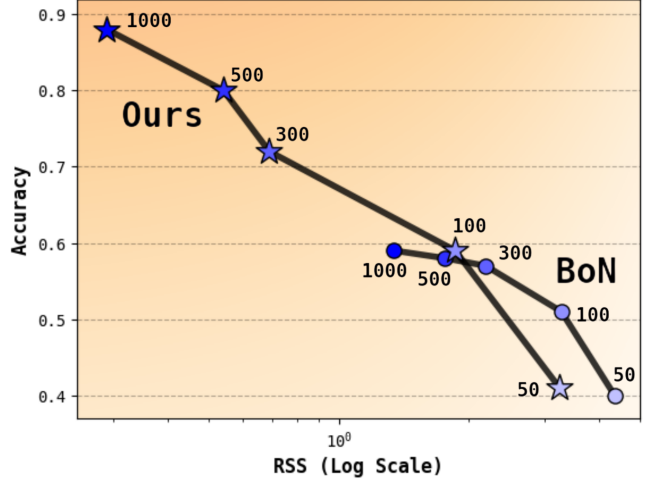


Figure 9. **Scaling behavior comparison of BoN and RBF.** We plot the known reward (RSS) [31] against accuracy for different numbers of function evaluations: $\{50, 100, 300, 500, 1,000\}$. Note that the horizontal axis is displayed on a logarithmic scale.

### D.1.1. Gradient-Based Guidance

Uehara et al. [53] have shown that the marginal distribution $p_t^*(\mathbf{x}_t)$ is computed as follows:

$$p_t^*(\mathbf{x}_t) \propto \exp\left(\frac{v(\mathbf{x}_t)}{\beta}\right) p_t(\mathbf{x}_t) \approx \exp\left(\frac{r(\mathbf{x}_{0|t})}{\beta}\right) p_t(\mathbf{x}_t),$$

where the approximation follows from Eq. 19. When the reward is differentiable (*e.g.*, aesthetic score [44]), one can simulate samples from $p_t^*(\mathbf{x}_t)$ by computing its score function:

$$\nabla \log p_t^*(\mathbf{x}_t) = \nabla \log\left[\exp\left(\frac{r(\mathbf{x}_{0|t})}{\beta}\right) p_t(\mathbf{x}_t)\right]$$
$$= \frac{1}{\beta} \underbrace{\nabla r(\mathbf{x}_{0|t})}_{\text{Guidance}} + \underbrace{\nabla \log p_t(\mathbf{x}_t)}_{\text{Pretrained Score}}. \tag{25}$$

For differentiable rewards, we incorporate the gradient-based guidance defined in Eq.25 into the SDE sampling process described in Eq.7. Notably, this approach is orthogonal to inference-time scaling, and RBF can be additionally utilized to further enhance performance. In the next section, we experimentally demonstrate that RBF can be effectively integrated with gradient-based guidance.

### D.1.2. Results

The aesthetic image generation task aims to sample images that best capture human preferences, such as visual appeal. We use 45 animal prompts from previous work, DDPO [5]. The aesthetic score [44] serves as the given reward, while ImageReward [57] is used as the held-out reward.

We present quantitative and qualitative results of aesthetic image generation in Tab.3 and Fig.8. Notably, RBF, implemented with DPS [8], achieves significant improvements on both the given and held-out rewards, even surpassing SVDD [27]. Qualitatively, RBF effectively adapts the pretrained flow model to better align with human preferences, particularly in terms of visual appeal.

### D.2. Scaling Behavior Comparison

As discussed in Sec. 4, expanding the exploration space and applying budget forcing significantly enhance the efficiency of RBF, leading to superior performance improvements over BoN. Here, we compare the scaling behavior of BoN, a representative Linear-ODE-based method, with RBF across different numbers of function evaluations (NFEs).

We present qualitative and quantitative results of scaling behavior in quantity-aware image generation in Fig.9 and Tab. 4, respectively. Our results indicate that allocating more compute leads to performance improvements for both BoN and RBF. However, the accuracy of BoN plateaus after 300 NFEs, whereas RBF continues to scale and achieves an accuracy of 0.88. Notably, RBF shows similar trend in the held-out reward, outperforming BoN and demonstrating its efficiency.
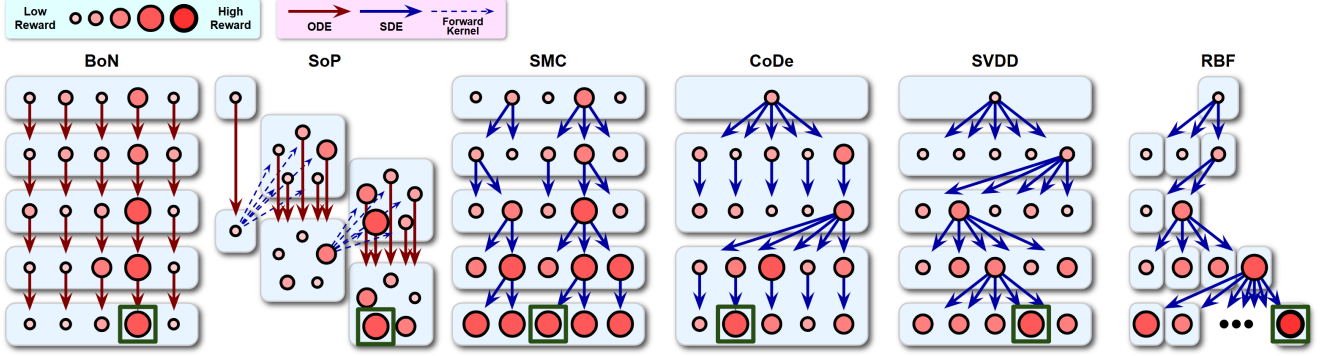


Figure 10. **Schematics of inference-time search algorithms.** Linear-ODE-based methods, BoN and SoP use a deterministic sampling process, whereas particle-sampling-based methods follow a stochastic process. Note that RBF adaptively allocates NFEs across denoising timesteps.

## E. Search Algorithms

In this section, we introduce the inference-time search algorithms discussed in Sec.2 along with their implementation details. An illustrative figure of the algorithms is provided in Fig.10. Here, we define the batch size $(N)$ as the number of initial latent samples and the particle size $(K)$ as the number of samples drawn from the proposal distribution $p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)$ at each denoising step.

**Best-of-N** (BoN) [49, 50] is a form of rejection sampling. Given $N$ generated samples $\{\mathbf{x}_0^{(i)}\}_{i=1}^N$, BoN selects the sample with the highest reward.

$$\mathbf{x}_0 = \arg\max_{\{\mathbf{x}_0^{(i)}\}_{i=1}^N} r(\mathbf{x}_0^{(i)}).$$

As presented in Sec. 6, we fixed the total compute budget to 500 NFEs and the number of denoising steps to 10, which sets the batch size of BoN to $N = 50$.

**Search over Paths** (SoP) [36] begins by sampling $N$ initial noises and running the ODE solver up to a predefined timestep $t_0$. Then the following two operations iterate until reaching $t = 0$:

1. Applying the forward kernel: For each sample in the batch at time $t$, $K$ particles are sampled using the forward kernel, which propagates them from $t$ to $t + \Delta_f$.
2. Solving the ODE: The resulting $N \cdot K$ particles are then evolved from $t + \Delta_f$ to $t + \Delta_f - \Delta_b$ by solving the ODE. The top $N$ candidates with the highest rewards are selected.

We followed the original implementations [36] for $\Delta_f$ and $\Delta_b$. We used $N = 2$ and $K = 5$.

**Sequential Monte Carlo** (SMC) [13, 21] extends the idea of importance sampling to a time-sequential setting by maintaining $N$ samples and updating their importance weights over time:

$$w_{t-\Delta t}^{(i)} = \frac{p_\theta^*(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)}{q(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)}w_t^{(i)} = \frac{p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)\exp\left(v(\mathbf{x}_{t-\Delta t}^{(i)})/\beta\right)}{q(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)\exp\left(v(\mathbf{x}_t^{(i)})/\beta\right)}w_t^{(i)},$$

where $q(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)$ is a proposal distribution and the last equality follows from the optimal policy Eq. 18. We used the reverse process of the pretrained model as the proposal distribution, which leads to the following importance weight equation:

$$w_{t-\Delta t}^{(i)} = \frac{\exp\left(v(\mathbf{x}_{t-\Delta t}^{(i)})/\beta\right)}{\exp\left(v(\mathbf{x}_t^{(i)})/\beta\right)} w_t^{(i)}. \tag{26}$$

At each step when effective sample size $\left(\sum_{j=1}^{N} w_t^{(j)}\right)^2 / \sum_{i=1}^{N}(w_t^{(i)})^2$ is below the threshold, we perform resampling, *i.e.*, indices $\{a_t^{(i)}\}_{i=1}^{N}$ are first sampled from a multinomial distribution based on the normalized importance weights:

$$\{a_t^{(i)}\}_{i=1}^{N} \sim \text{Multinomial}\left(N, \quad \left\{\frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}}\right\}_{i=1}^{N}\right).$$

These ancestor indices $a_t^{(i)}$ are then used to replicate high-weight particles and discard low-weight ones, yielding the resampled set $\{\mathbf{x}_t^{(a_t^{(i)})}\}_{i=1}^{N}$. If resampling is not performed, the indices are simply set as $a_t^{(i)} = i$. Lastly, one-step denoised samples are obtained from $\{\mathbf{x}_t^{(a_t^{(i)})}\}_{i=1}^{N}$:

$$\mathbf{x}_{t-\Delta t}^{(i)} \sim p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t^{(a_t^{(i)})}).$$

When resampling is performed, the importance weights are reinitialized to one, *i.e.*, $w_t = 1$. The importance weights for the next step, $w_{t-\Delta t}$ are subsequently computed according to Eq. 26, regardless of whether resampling was applied. We used $N = 50$ for all applications.

**Controlled Denoising** (CoDe) [45] extends BoN by incorporating an interleaved selection step after every $L$ denoising steps.

$$\mathbf{x}_{t-L\Delta t} = \underset{\{\mathbf{x}_{t-L\Delta t}^{(i)}\}_{i=1}^{K}}{\arg\max} \ \exp\left(v(\mathbf{x}_{t-L\Delta t}^{(i)})/\beta\right)$$

We used $N = 2$, $K = 25$, and $L = 2$ for all applications.

**SVDD** [27] approximates the optimal policy in Eq. 3 by leveraging weighted $K$ particles:

$$p_\theta^*(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t) \approx \sum_{i=1}^{K} \frac{w_{t-\Delta t}^{(i)}}{\sum_{j=1}^{K} w_{t-\Delta t}^{(j)}} \delta_{\mathbf{x}_{t-\Delta t}^{(i)}} \tag{27}$$

$$\{\mathbf{x}_{t-\Delta t}^{(i)}\}_{i=1}^{K} \sim p_\theta(\mathbf{x}_{t-\Delta t}|\mathbf{x}_t)$$

$$w_{t-\Delta t}^{(i)} = \exp\left(v(\mathbf{x}_{t-\Delta t}^{(i)})/\beta\right).$$

At each timestep, the approximate optimal policy in Eq. 27 is sampled by first drawing an index $a_{t-\Delta t}$ from a categorical distribution:

$$a_{t-\Delta t} \sim \text{Categorical}\left(\left\{\frac{w_{t-\Delta t}^{(i)}}{\sum_{j=1}^{K} w_{t-\Delta t}^{(j)}}\right\}_{i=1}^{K}\right) \tag{28}$$

This index is then used to select the sample from $\{\mathbf{x}_{t-\Delta t}^{(i)}\}_{i=1}^{K}$, *i.e.*, $\mathbf{x}_{t-\Delta t} \leftarrow \mathbf{x}_{t-\Delta t}^{(a_{t-\Delta t})}$. In practice, SVDD uses $\beta = 0$, replacing sampling from the categorical distribution with a direct $\arg\max$ operation, *i.e.*, selecting the particle with the largest importance weight. Following the original implementation [27], we used $N = 2$ and $K = 25$ for all applications.

**Rollover Budget Forcing** (RBF) adaptively allocates compute across denoising timesteps. At each timestep, when a particle with a higher reward than the previous one is discovered, it immediately takes a denoising step, and the remaining

NFEs are rolled over to the next timestep, ensuring efficient utilization of the available compute. To maintain consistency with SVDD [27], we set $N = 2$, with the compute initially allocated uniformly across all timesteps. We present the pseudocode for sampling from the stochastic proposal distribution with interpolant conversion in Alg. 1. Specifically, the pseudocode for RBF with SDE conversion and interpolant conversion is provided in Alg. 2. Here, we denote $\{S^{(i)}\}_{i=1}^{M}$ as a sequence of timesteps in descending order, where $S^{(1)} = 1$ and $S^{(M)} = 0$, and $M$ is the total number of denoising steps.

---

**Algorithm 1:** `stoch_denoise`: 1-step stochastic denoising

**Inputs:** original velocity field $u$, original interpolant $(\alpha, \sigma)$, new interpolant $(\bar{\alpha}, \bar{\sigma})$, diffusion coefficient $g$, current sample $\bar{\mathbf{x}}_s$, current timestep $s$, denoising step size $\Delta s$

**Outputs:** Stochastically denoised sample $\bar{\mathbf{x}}_{s-\Delta s}$

1   $t_s \leftarrow \rho^{-1}(\bar{\rho}(s)) \quad c_s \leftarrow \bar{\sigma}_s / \sigma_{t_s}$

2   $\bar{\mathbf{u}}_s \leftarrow \frac{\dot{c}_s}{c_s} \bar{\mathbf{x}}_s + c_s \dot{t}_s u_{t_s}\left(\frac{\bar{\mathbf{x}}_s}{c_s}\right)$      // Eq. 12

3   $\mathbf{s}_s \leftarrow \frac{1}{\bar{\sigma}_s} \frac{\bar{\alpha}_s \bar{\mathbf{u}}_s - \dot{\bar{\alpha}}_s \bar{\mathbf{x}}_s}{\dot{\bar{\alpha}}_s \bar{\sigma}_s - \bar{\alpha}_s \dot{\bar{\sigma}}_s}$      // Eq. 9

4   $\mathbf{f}_s = \bar{\mathbf{u}}_s - \frac{g_s^2}{2} \mathbf{s}_s$      // Eq. 8

5   $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

6   $\bar{\mathbf{x}}_{s-\Delta s} \leftarrow \bar{\mathbf{x}}_s - \mathbf{f}_s \Delta s + g_s \sqrt{\Delta s} \, \mathbf{z}$

---

**Algorithm 2:** Rollover Budget Forcing (RBF)

**Inputs:** Number of denoising steps $M$, timesteps $\{S^{(i)}\}_{i=1}^{M}$, NFE quota $\{Q^{(i)}\}_{i=1}^{M}$

**Outputs:** Aligned sample $\bar{\mathbf{x}}_0$

1   $\bar{\mathbf{x}}_1 \sim \mathcal{N}(0, \boldsymbol{I}) \quad r^* \leftarrow r(\bar{\mathbf{x}}_{0|1})$

2   **for** $i \in \{1, \ldots, M\}$ **do**

3     $s \leftarrow S^{(i)} \quad \Delta s \leftarrow S^{(i)} - S^{(i+1)} \quad q \leftarrow Q^{(i)}$

4     **for** $j \in \{1, \ldots, q\}$ **do**

5        $\bar{\mathbf{x}}_{s-\Delta s}^{(j)} \leftarrow$ `stoch_denoise`$(\bar{\mathbf{x}}_s, s, \Delta s)$   // Alg. 1

6        **if** $r^* < r(\bar{\mathbf{x}}_{0|s-\Delta s}^{(j)})$ **then**

7           $Q^{(i+1)} \leftarrow Q^{(i+1)} + Q^{(i)} - j$    // Sec. 5

8           $r^* \leftarrow r(\bar{\mathbf{x}}_{0|s-\Delta s}^{(j)}) \quad \bar{\mathbf{x}}_{s-\Delta s} \leftarrow \bar{\mathbf{x}}_{s-\Delta s}^{(j)}$

9           **break**

10       **if** $j = q$ **then**

11          $k^* \leftarrow \arg\max_{k \in \{1, \ldots, q\}} r(\bar{\mathbf{x}}_{0|s-\Delta s}^{(k)})$

12          $\bar{\mathbf{x}}_{s-\Delta s} \leftarrow \bar{\mathbf{x}}_{s-\Delta s}^{(k^*)}$

---

## F. Implementation Details

### F.1. Compositional Text-to-Image Generation

In the compositional text-to-image generation task, we use the VQAScore metric as the reward, which evaluates image-text alignment using a visual question-answering (VQA) model (CLIP-FlanT5 [28] and InstructBLIP [10]). Specifically, VQAScore measures the probability that a given attribute or object is present in the generated image. To compute the reward, we scale the probability value by setting $\beta = 0.1$ in Eq. 3.

### F.2. Quantity-Aware Image Generation

In quantity-aware image generation, text prompts specify objects along with their respective quantities. To generate images that accurately match the specified object counts, we use the negation of the Residual Sum of Squares (RSS) as the given reward. Here, RSS is computed to measure the discrepancy between the detected object count $\hat{C}_i$ and the target object count $C_i$ specified in the given text prompt:

$$\text{RSS} = \sum_{i=1}^{n} \left(C_i - \hat{C}_i\right)^2,$$

where $n$ is the total number of object categories in the prompt. We additionally report accuracy, which is defined as 1 when $\text{RSS} = 0$ and 0 otherwise. For the held-out reward, we report VQAScore measured with CLIP-FlanT5 [28] model.

**Object Detection Implementation Details.** To compute the given reward, RSS, it is necessary to detect the number of objects per category, $\hat{C}_i$. Here, we leverage the state-of-the-art object detection model, GroundingDINO [31] and the object segmentation model SAM [22], which is specifically used to filter out duplicate detections.

We observe that naïvely using the detection model [31] to compute RSS leads to poor detection accuracy due to two key issues: inner-class duplication and cross-class duplication. Inner-class duplication occurs when multiple detections are

assigned to the same object within a category, leading to overcounting. This often happens when an object is detected both individually and as part of a larger group. Cross-class duplication arises when an object is assigned to multiple categories due to shared characteristics (*e.g.*, a toy airplane being classified as both a toy and an airplane), making it difficult to assign it to a single category.

To address inner-class duplication, we refine the object bounding boxes detected by GroundingDINO [31] using SAM [22] and filter out overlapping detections. Smaller bounding boxes are prioritized, and larger ones that significantly overlap with existing detections are discarded. This ensures that each object is counted only once within its category. To resolve cross-class duplication, we assign each object to the category with the highest GroundingDINO [31] confidence score. This ensures that objects are classified into a single category, preventing double counting across multiple classes.

**More qualitative results are presented in the following pages.**

# G. Additional Qualitative Results

## G.1. Comparisons of Inference-Time SDE Conversion and Interpolant Conversion



Figure 11. **Qualitative comparisons of inference-time SDE conversion and interpolant conversion.** Each column represents Linear-ODE (L.O.), Linear-SDE (L.S.), and VP-SDE (V.S.).
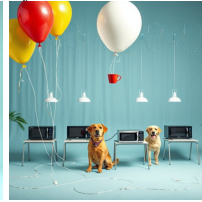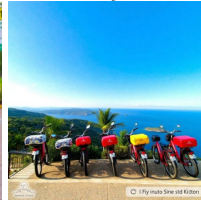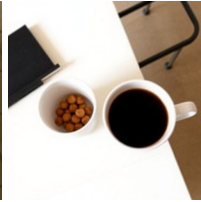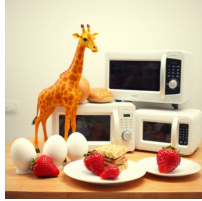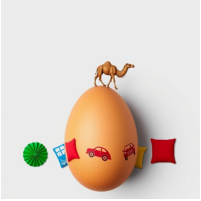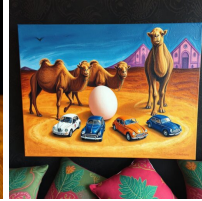
| | L.O. | L.S. | V.S. | | L.O. | L.S. | V.S. |

*SVDD [27] Quantity*

*"Four balloons, one cup, four desks, two dogs and four microwaves."* — *"Five hamburgers sizzled on the grill."*

*SVDD [27] Quantity*

*"Two men, four vases, four chickens and four ships."* — *"Six bicycles."*

*SVDD [27] Composition*
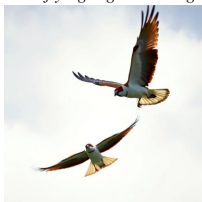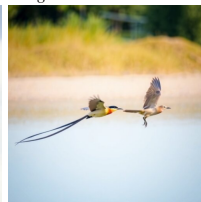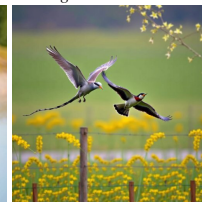
*"Two people and two bicycles in the street, the bicycle with the larger wheels belongs to the taller person."* — *"There are two cups on the table, the cup without coffee is on the left of the other filled with coffee."*

| | L.O. | L.S. | V.S. | | L.O. | L.S. | V.S. |

*RBF (Ours) Quantity*

*"Two giraffes, three eggs, two breads, three microwaves and four strawberries."* — *"Four pears, four desks, three paddles and two rabbits."*

*RBF (Ours) Quantity*

*"One egg, three camels, four cars and four pillows."* — *"Seven women."*

*RBF (Ours) Composition*

*"Three sailboats on the water, each with sails of a different color."* — *"Two birds are chasing each other in the air, with the one flying higher having a long tail and the other bird having a short tail."*

Figure 12. **Qualitative comparisons of inference-time SDE conversion and interpolant conversion.** Each column represents Linear-ODE (L.O.), Linear-SDE (L.S.), and VP-SDE (V.S.).

# G.2. Comparisons of Inference-Time Scaling

| BoN | SoP [36] | SMC [21] | CoDe [45] | SVDD [27] | **RBF** (Ours) |
|---|---|---|---|---|---|

*"In a room, all the chairs are occupied except one."*



*"In a pack of wolves, each one howls at the moon, but one remains silent."*



*"An open biscuit tin contains three biscuits, one without sultanas is square-shaped and the other two are round-shaped."*



*"A rose that is not fully bloomed is higher than a rose that is already in bloom."*



*"There are two colors of pots in the flower garden; all green pots have tulips in them and all yellow pots have no flowers in them."*



*"Two dragons fly towards the castle, a dragon with a backpack and no hat on the left of the dragon without a backpack and with a hat."*



Figure 13. **Additional qualitative results of compositional text-to-image generation task**.

| BoN | SoP [36] | SMC [21] | CoDe [45] | SVDD [27] | **RBF** (Ours) |
|-----|----------|----------|-----------|-----------|-----------------|

*"Eight apples, three bicycles and five rabbits."*



*"Six helicopters buzzed over eight pillows."*



*"Five swans and seven ducks swam in the pond."*



*"Four drums, seven tomatoes, and five candles."*



*"Three chickens, four birds, and eight pears."*



*"Six airplanes flying over a desert with seven camels walking below."*
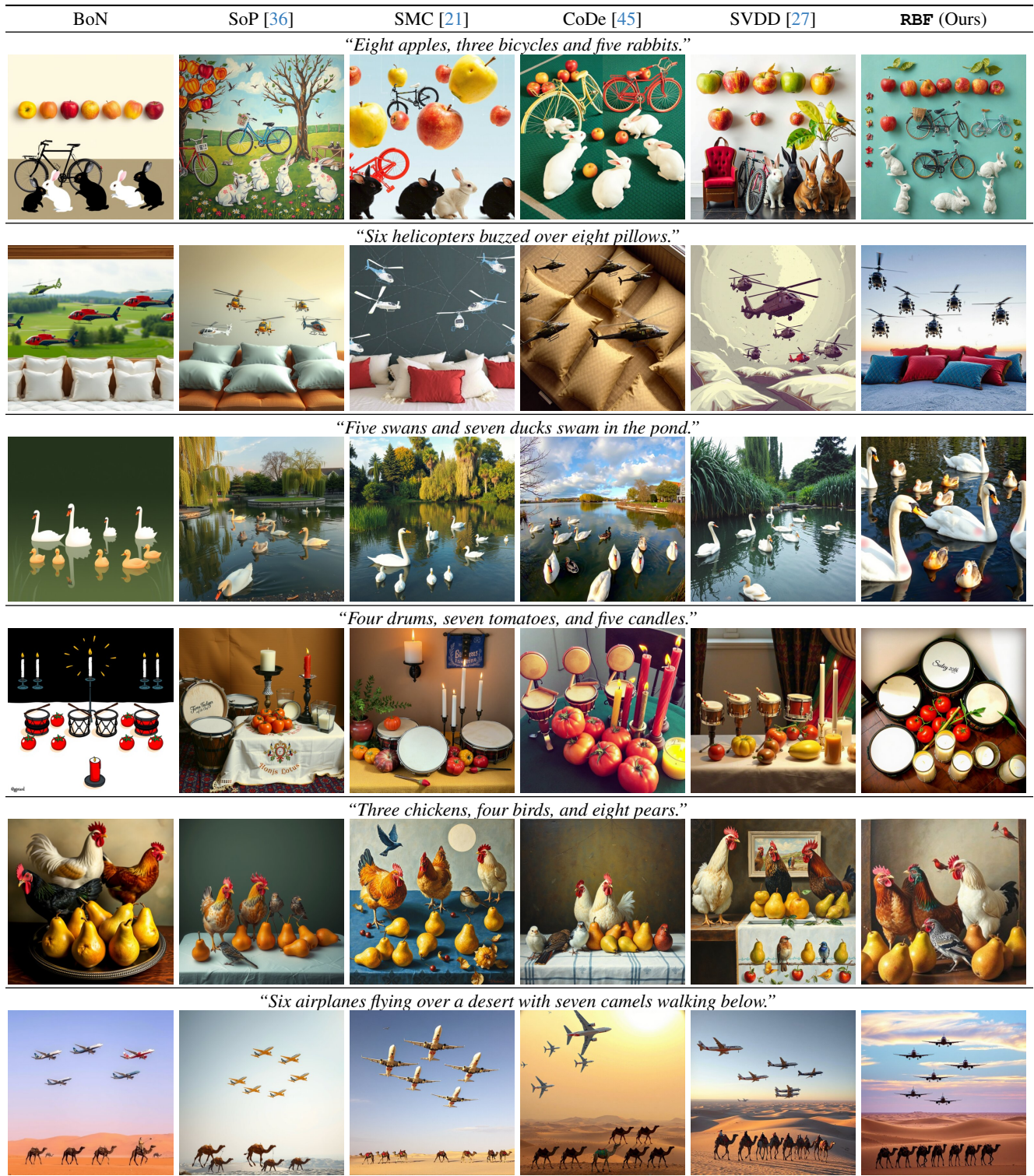


Figure 14. **Additional qualitative results of quantity-aware image generation task**.