Flow-Direct: Feedback-Efficient and Reusable Guidance for Flow Models via Non-Parametric Guidance Field

Kim Yong Tan [1] **Yueming Lyu** [2,3*] **Ivor Tsang**[1,2,3] **Yew-Soon Ong**[1,2,3]

[1]College of Computing and Data Science, Nanyang Technological University, Singapore
[2]Centre for Frontier AI Research, Agency for Science, Technology and Research, Singapore
[3]Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore

{KIMYONG001,ASYSOng}@ntu.edu.sg
{Lyu_Yueming,Ivor_Tsang}@cfar.a-star.edu.sg

Abstract

Training-free guidance enables pre-trained diffusion and flow models to optimize application-specific objectives using feedback from external black-box reward functions. However, existing methods are feedback-inefficient because reward feedback is used only transiently to inform a localized gradient approximation or a discrete search decision, and is subsequently discarded. To address this limitation, we propose Flow-Direct, a framework that guides the generation process via a persistent *guidance field*. Theoretically, this guidance field is analytically derived from the log-density ratio between the base and reward-weighted target distributions; it transports the pre-trained distribution to the target distribution. In practice, the field is implemented as a non-parametric estimator constructed from all accumulated reward-evaluated samples. As more samples are collected during optimization, this empirical guidance field becomes increasingly accurate. This persistent formulation yields two major advantages. First, Flow-Direct is highly *feedback-efficient*: because every evaluated sample is used to refine the global guidance field, no reward information is wasted. Second, the framework is naturally *reusable*: once optimization is complete, the collected dataset defines a reusable guidance field for generating novel target samples without additional reward evaluations, and distinct guidance fields can be combined to generate samples that simultaneously satisfy multiple objectives. The source code is publicly available at: `https://github.com/kimyong95/finetune-stable-diffusion`.

1 Introduction

Diffusion and flow-based generative models have become a dominant paradigm for high-dimensional generation. They achieve strong performance in image synthesis [1, 2], video generation [3, 4], and audio synthesis [5], and have increasingly been adopted in scientific domains such as molecule generation [6, 7], protein design [8], and materials discovery [9]. This success stems from their ability to model complex, multi-modal data distributions and to transform simple noise into realistic samples through a learned generative trajectory.

However, many applications do not only require samples from the broad distribution learned during pre-training. Instead, they require samples from a specific target distribution defined by application-

*Corresponding author

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

specific objectives. For example, protein design may seek structures with high binding affinity or stability, while aerodynamic design may seek shapes with low drag, high efficiency, or other physical properties. In such settings, the target distribution is often specified only implicitly through an external reward function that evaluates the performance of a generated sample. This reward evaluation process may be a laboratory experiment, a computational simulation, or a learned preference model. Because these evaluation procedures are often expensive, non-differentiable, and unavailable during model training, target generation must be achieved with limited black-box feedback.

A prominent approach for this setting is *training-free guidance*, which steers a pre-trained generative model at inference time using feedback from an external reward function. Since these methods do not update model parameters, they avoid the cost and instability of fine-tuning or reinforcement learning, and can be applied when the reward function is black-box or non-differentiable.

To achieve this, recent methods typically rely on gradient approximation, sequential Monte Carlo (SMC), or tree search. Specifically, during each intermediate denoising step, Evolvable [10] guides the generation trajectory via zero-order gradient estimation, and FK-Steering [11] performs particle resampling (SMC) to steer towards the target distribution. On the other hand, TreeG [12] formulates the sequential generation process as a search tree problem, and DSearch [13] incorporates dynamic tree expansion for efficient traversal. Despite their flexibility, they remain feedback-inefficient, requiring a massive number of reward evaluations for accurate gradient approximation, robust particle resampling, or comprehensive search. We defer a more detailed review to Appendix E.

We argue that a fundamental source of this inefficiency is that existing inference-time guidance methods use feedback only transiently. At each generation step, samples are evaluated to estimate a local gradient, select a search branch, or approximate a posterior distribution. Once this intermediate decision is made, the evaluated samples are discarded. As a result, valuable information from expensive reward evaluations is not preserved, leading to low feedback efficiency.

To address this limitation, we propose Flow-Direct, a framework that steers the flow models via a persistent *guidance field*. Theoretically, the guidance field is analytically derived from the log-density ratio between the base and reward-weighted target distributions; it transports the pre-trained distribution to the target distribution. In practice, the guidance field is implemented as a non-parametric estimator constructed from the reward-evaluated samples.

Flow-Direct is highly *feedback-efficient* because rather than utilizing reward feedback transiently, the non-parametric implementation allows us to aggregate the evaluated samples persistently, ensuring no reward-information loss. Over the course of the optimization process, all reward-evaluated samples are appended to a growing empirical dataset. As this dataset expands, the non-parametric estimator yields an increasingly accurate approximation of the exact guidance field, thereby guiding the flow model towards the target distribution effectively.

Furthermore, this framework is naturally *reusable*. Once a labeled dataset is collected for a specific target objective, its corresponding empirical guidance field can be reused to generate novel target samples without further reward evaluations. Additionally, the non-parametric estimators learned from distinct objectives can be linearly combined to produce samples that simultaneously satisfy multiple targets.

Our contributions are summarized as follows:

- We propose Flow-Direct, a training-free framework that guides pre-trained flow models using a *non-parametric guidance field*. This field is analytically derived from the log-density ratio between the base and reward-weighted target distributions, and is implemented as a non-parametric estimator.

- Moreover, Flow-Direct is highly *feedback-efficient*. Because the non-parametric implementation utilizes accumulated reward-evaluated samples rather than using feedback transiently, no reward information is discarded. The reward feedback is persistently preserved in the guidance field, which becomes increasingly accurate throughout the optimization process.

- Furthermore, the constructed guidance field is natively *reusable* after optimization. Once a labeled dataset is collected, it can steer the flow model to generate novel target samples without requiring further reward evaluations. Moreover, multiple empirical fields can be linearly combined to generate samples that simultaneously satisfy multiple objectives.

- Finally, we conduct comprehensive experiments on both image and 3D generation tasks, indicating Flow-Direct's superior effectiveness, scalability, and robust performance across multiple domains compared to existing methods.

2 Preliminaries

Flow Model. The flow matching framework [14] constructs a continuous-time transport from a noise distribution $p_0(x_0) = \mathcal{N}(0, I)$ to the data distribution $p_1(x_1)$ via linear interpolation $x_t = (1-t)\,x_0 + t\,x_1$ for $t \in [0,1]$. Conditioned on a data point x_1, the intermediate state admits a Gaussian distribution:

$$p_t(x_t|x_1) = \mathcal{N}\big(x_t\big|tx_1, (1-t)^2 I\big). \tag{1}$$

The trajectory of x_t is governed by the ODE $dx_t = v_t(x_t)\,dt$, with the time-dependent vector field given as:

$$v_t(x_t) = \frac{x_t}{t} + \frac{1-t}{t}\nabla_{x_t} \log p_t(x_t), \tag{2}$$

where we provide the derivation in Appendix A. In practice, a neural network $v_\theta(x_t)$ is trained to approximate the true velocity field, i.e., $v_\theta(x_t) \approx v_t(x_t)$. During inference, new samples are generated by drawing $x_0 \sim \mathcal{N}(0, I)$ and integrating the ODE. A standard and simple choice for solving the ODE is first-order Euler discretization:

$$x_{t+\Delta t} = \underbrace{x_t + \Delta t\, v_t}_{\texttt{step}(x_t, v_t)}, \tag{3}$$

where $v_t \coloneqq v_\theta(x_t)$. Iterating this update from $t = 0$ to $t = 1$ produces a final sample $x_1 \sim p_1^\theta(x_1)$ from the learned data distribution. In this work, we treat the numerical update operator `step` as a black-box sampler, which may be instantiated by more advanced ODE or SDE solvers, and we focus solely on modifying the vector field v_t.

3 Method

In this section, we introduce Flow-Direct, a training-free framework that steers pre-trained flow models via a *non-parametric guidance field*. We begin by formulating this guidance field in Section 3.1, showing how it can be constructed when target data is directly available. Since such target data is typically unavailable in real-world scenarios, Section 3.2 presents an optimization algorithm that estimates the guidance field using an external black-box reward. This optimization process naturally produces a labeled target dataset. In Section 3.3, we show how the accumulated dataset can be reused to generate novel target samples without additional reward evaluations. Finally, Section 3.4 highlights the theoretical properties and practical advantages of our framework.

3.1 Constructing the Guidance Field from Target Data

Problem Setup. Given a pre-trained flow model, we consider the problem of steering its generative distribution toward a desired target distribution using only two small datasets, without retraining. Formally, we assume access to a base dataset, sampled from a distribution that approximates the pre-trained distribution; and a target dataset, sampled from the desired distribution. Our goal is to construct a guidance formulation, computed entirely from these two datasets, that steers the pre-trained distribution from the base toward the target.

Guided Score. Let v_θ denote a pre-trained generative flow model, and let p_t^θ denote the marginal distribution that this model generates at time t. Suppose the flow model is well-trained to approximate the true vector field, i.e., $v_\theta(x_t) \approx v_t(x_t)$ in Equation (2), then:

$$v_\theta(x_t) \approx \frac{x_t}{t} + \frac{1-t}{t}\nabla_{x_t} \log p_t^\theta(x_t). \tag{4}$$

This identity shows that the learned flow implicitly encodes the score of p_t^θ, which allows us to perform guidance in score space.

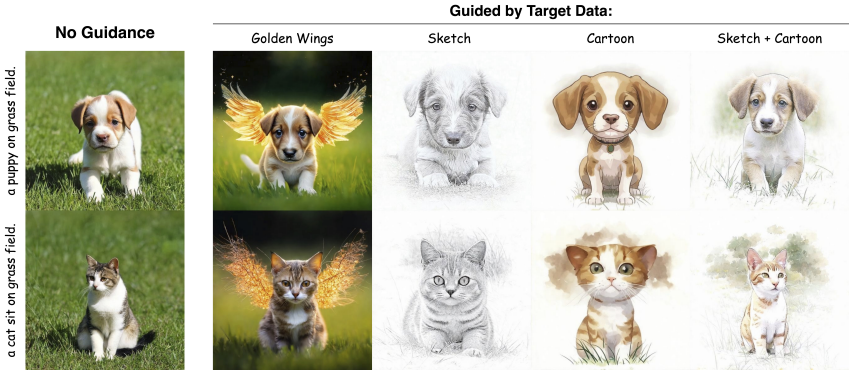


Figure 1: **Guided Flow Model by Target Data.** Each row shows generations from the same prompt: the first column is the unguided baseline, and the remaining columns apply Equation (8) with different target datasets. The *Sketch + Cartoon* column is produced by summing the two shift terms. **Top:** guidance steers the *puppy* generation toward each target distribution, and the composed shift meaningfully mixes both styles. **Bottom:** even when the prompt (*cat*) mismatches the guidance datasets (*puppy*), the target attributes still transfer. This shows that our formulation generalizes beyond the matched-distribution setting.

Let p_1^{base} and p_1^{target} denote the base and target data distributions. For each, we use Equation (1) to construct a corresponding probability path, p_t^{base} and p_t^{target} for $t \in [0, 1]$, by applying the flow matching interpolation between noise and the respective endpoint distribution. We guide the flow model in the score space by adding the log-density ratio between the base and target distribution:

$$v_\theta^{\text{guided}}(x_t) := \frac{x_t}{t} + \frac{1-t}{t}\left(\nabla_{x_t} \log p_t^\theta(x_t) + \nabla_{x_t} \log \frac{p_t^{\text{target}}(x_t)}{p_t^{\text{base}}(x_t)}\right) \tag{5}$$

$$= v_\theta(x_t) + \underbrace{\frac{1}{1-t}\left(\mathbb{E}_{p_1^{\text{target}}}[x_1|x_t] - \mathbb{E}_{p_1^{\text{base}}}[x_1|x_t]\right)}_{\text{guidance field } \Delta(x_t)}, \tag{6}$$

and we show that this guidance is equivalent to a guidance field $\Delta(x_t)$ constructed by the difference between the base and target posterior expectation, with the derivation provided in Appendix A. In Equation (5), the base distribution has a role of reference distribution, suppose $p_t^{\text{base}}(x_t) = p_t^\theta(x_t)$, then they cancel out, and the guided flow model generates the target distribution exactly. In Equation (6), the posterior expectation $\mathbb{E}[x_1|x_t]$ represents the ideal estimate of the clean data x_1 given the current state x_t at time t. Geometrically, it is a vector pointing from the base data towards the target data. In practice, we implement the guidance field by approximating the posterior expectation using the non-parametric estimator given in Proposition 1 with the derivation provided in Appendix A:

Proposition 1 (Estimator for Posterior Expectation) *Let $\mathcal{D} = \{x_1^i\}_{i=1}^N$ be a dataset where $x_1^i \overset{\text{iid}}{\sim} p_1^{\text{base}}(x_1)$. Under the flow matching framework, for any x_t with $t \in [0, 1)$, the posterior expectation can be estimated by:*

$$\sum_{i=1}^{N} \underbrace{\underset{i' \in [N]}{\text{softmax}}\left(-\frac{\|x_t - t x_1^{i'}\|^2}{2(1-t)^2}\right)_i}_{\hat{p}_1(x_1 = x_1^i | x_t)} x_1^i \quad \xrightarrow{N \to \infty} \quad \mathbb{E}_{p_1}[x_1|x_t], \tag{7}$$

and the estimator converges to the true posterior expectation as the dataset size grows to infinity.

Suppose we have access to the base and target datasets, $\mathcal{D}_{\text{base}} = \{x_1^i\}_{i=1}^N$ where $x_1^i \overset{\text{iid}}{\sim} p_1^{\text{base}}(x_1)$ and $\mathcal{D}_{\text{target}} = \{x_1^j\}_{j=1}^M$ where $x_1^j \overset{\text{iid}}{\sim} p_1^{\text{target}}(x_1)$. Applying Proposition 1 yields a non-parametric guidance field:[2]

[2]See Equation (8-practical) for the corresponding practical implementation.

$$\Delta(x_t; \mathcal{D}_{\text{base}}, \mathcal{D}_{\text{target}}) \coloneqq \frac{1}{1-t}\Bigg[\underbrace{\sum_{j=1}^{M} \underset{j' \in [M]}{\text{softmax}}\left(-\frac{\|x_t - t x_1^{j'}\|^2}{2(1-t)^2}\right)_j x_1^j}_{\approx \mathbb{E}_{p_1^{\text{target}}}[x_1|x_t]} - \underbrace{\sum_{i=1}^{N} \underset{i' \in [N]}{\text{softmax}}\left(-\frac{\|x_t - t x_1^{i'}\|^2}{2(1-t)^2}\right)_i x_1^i}_{\approx \mathbb{E}_{p_1^{\text{base}}}[x_1|x_t]}\Bigg] \tag{8}$$

where we use the shorthand notation for softmax[3] throughout this paper. Substituting this guidance field into Equation (6) yields the guided flow model. While this provides the exact theoretical formulation, direct computation in high-dimensional spaces can suffer from numerical instability. We discuss the practical implementation details in Appendix B.

Demonstration. We test the non-parametric guidance field formulation. Our goal is to steer the pre-trained flow model from the *puppy* base distribution toward three distinct targets: *puppy with golden wings*, *sketch drawing of puppy*, and *cartoon puppy*. For each target, we prepare a pair of datasets, $\mathcal{D}_{\text{base}}$ and $\mathcal{D}_{\text{target}}$, consisting of 64 images each (details provided in Appendix F), and construct the corresponding guidance field according to Equation (8). As shown in Figure 1, applying the guidance field successfully steers the flow model toward each respective target distribution. Furthermore, linearly combining the *sketch* and *cartoon* guidance fields yields meaningful compositional generation. We also evaluate a distribution-mismatch setting ($p_t^\theta \neq p_t^{\text{base}}$) by using the inference prompt *cat* while applying the guidance field derived from the *puppy* datasets. The target attributes are successfully transferred, demonstrating that our formulation remains robust even when the base dataset does not match the pre-trained distribution.

3.2 Optimizing the Guidance Field via Black-Box Rewards

Problem Setup. In practical settings, the target dataset is not directly accessible. In this section, we discuss how to learn the guidance field using an external reward function. Suppose we have access to a base data distribution $p_1^{\text{base}}(x_1)$ and an external reward function $r(x_1)$. Our goal is to steer generation toward a target distribution biased toward high-reward regions. We define the target distribution as a Boltzmann distribution:

$$p_1^{\text{target}}(x_1) \propto p_1^{\text{base}}(x_1) \exp(r(x_1)). \tag{9}$$

Following Equation (6), the guidance field toward this target distribution is determined by the target posterior expectation. We show that this expectation admits a non-parametric estimator in Proposition 2 with the derivation provided in Appendix A:

Proposition 2 (Estimator for Target Posterior Expectation) *For the target distribution $p_1^{\text{target}}(x_1)$ and reward function $r(x_1)$ defined in Equation* (9)*, let $\mathcal{D} = \{(x_1^i, r_i)\}_{i=1}^N$ be a labeled dataset, where $x_1^i \overset{\text{iid}}{\sim} p_1^{\text{base}}(x_1)$ and $r_i = r(x_1^i)$. Under the flow matching framework, for any x_t with $t \in [0, 1)$, the target posterior expectation can be estimated by:*

$$\sum_{i=1}^{N} \underset{i' \in [N]}{\text{softmax}}\left(-\frac{\|x_t - t x_1^{i'}\|^2}{2(1-t)^2} + r_{i'}\right)_i x_1^i \xrightarrow{N \to \infty} \mathbb{E}_{p_1^{\text{target}}}[x_1|x_t], \tag{10}$$

which converges to the true target posterior expectation as the dataset size grows to infinity.

Compared to Proposition 1, each softmax logit has the corresponding reward $r_{i'}$ added to it, biasing the distribution toward high-reward samples and thereby biasing the estimator toward the target distribution. The guidance field to transport $p_1^{\text{base}}(x_1)$ to $p_1^{\text{target}}(x_1)$ is thereby given as:[4]

$$\Delta(x_t; \mathcal{D}) \coloneqq \frac{1}{1-t}\sum_{i=1}^{N}\left[\underset{i' \in [N]}{\text{softmax}}\left(-\frac{\|x_t - t x_1^{i'}\|^2}{2(1-t)^2} + r_{i'}\right)_i - \underset{i' \in [N]}{\text{softmax}}\left(-\frac{\|x_t - t x_1^{i'}\|^2}{2(1-t)^2}\right)_i\right] x_1^i \tag{11}$$

[3] We use the shorthand notation: $\text{softmax}_{i' \in [N]}(z_{i'})_i \coloneqq \exp(z_i)/\sum_{i'=1}^N \exp(z_{i'})$ where i' is a dummy index ranging over $[N] = \{1, \cdots, N\}$, and the outer subscript i selects the i-th component of the resulting probability vector.

[4] See Equation (11-practical) for the corresponding practical implementation.

Algorithm 1: Flow-Direct

Input: Pre-trained flow model v_θ, reward function $r(\cdot)$, maximum optimization iterations L, batch size N, step size Δt, guidance field $\Delta(x_t; \mathcal{D})$ as defined in Equation (11-practical).

Output: Generated samples $\{x_1\}$ and accumulated labeled datasets $\mathcal{D}_{\text{all}} = \bigcup_{l=0}^{L-1} \mathcal{D}_l$.

for $l \leftarrow 0$ **to** $L-1$ **do**
  do parallel for N **instances**
    Sample $x_0 \sim \mathcal{N}(0, I)$
    for $t \in [0, 1)$ with step Δt **do**
      $v_t \leftarrow v_\theta(x_t, t) + \sum_{0 \le j < l} \Delta(x_t; \mathcal{D}_j)$
      $x_{t+\Delta t} \leftarrow \texttt{step}(x_t, v_t)$
    Evaluate reward $r \leftarrow r(x_1)$
  Collect dataset $\mathcal{D}_l \leftarrow \{(x_1^i, r_i)\}_{i=1}^N$

Iterative Optimization. A single application of Equation (11) produces a target distribution that scales the base by a factor of $\exp(r(x_1))$. To push further toward high-reward regions, we iteratively define a sequence of distributions, where the target of the current iteration becomes the base for the next. Specifically, starting from the pre-trained distribution $p_1^0 := p_1^{\text{base}}$, we define a sequence of distributions recursively as:

$$p_1^l(x_1) \propto p_1^{l-1}(x_1) \exp(r(x_1)) \propto p_1^0(x_1) \exp(l \cdot r(x_1)), \tag{12}$$

where the second proportional follows by unrolling the recursion. At each l-th iteration, the guidance field that transports $p_1^0(x_1)$ to $p_1^l(x_1)$ is therefore:

$$\frac{1}{1-t}\left(\mathbb{E}_{p_1^l}[x_1|x_t] - \mathbb{E}_{p_1^0}[x_1|x_t]\right) = \sum_{0 \le j < l} \frac{1}{1-t}\left(\mathbb{E}_{p_1^{j+1}}[x_1|x_t] - \mathbb{E}_{p_1^j}[x_1|x_t]\right) \approx \sum_{0 \le j < l} \Delta(x_t; \mathcal{D}_j), \tag{13}$$

where the first equality decomposes into a telescoping sum of per-step guidance fields, and the second approximation follows from the per-step non-parametric guidance field in Equation (11). After completing the l-th iteration, we obtain the new target dataset $\mathcal{D}_l \sim p_1^l(x_1)$. This process repeats. We summarize the full procedure in Algorithm 1.

3.3 Reusable Guidance Fields

Over the course of optimization, we aggregate the labeled datasets across all L iterations:

$$\mathcal{D}_{\text{all}} \leftarrow \mathcal{D}_0 \cup \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_{L-1} = \{(x_1^i, r_i)\}_{i=1}^{NL}. \tag{14}$$

and we denote the underlying distribution as $x_1^i \sim q(x_1)$. Since it contains samples drawn from a sequence of distributions ranging from the initial base to the final target, its distribution lies between them. We therefore define the contrastive distributions symmetrically around $q(x_1)$:

$$\begin{cases} p^+(x_1) \propto q(x_1) \exp(\alpha r(x_1)) \\ q(x_1) \propto p^-(x_1) \exp(\alpha r(x_1)) \end{cases} \tag{15}$$

where $\alpha \ge 0$ controls the gap between the $p^+(x_1)$ and $p^-(x_1)$. By rearranging $p^-(x_1)$ and applying Proposition 2 to both distributions, the guidance field to transport $p^-(x_1)$ to $p^+(x_1)$ is:[5]

$$\boxed{\Delta(x_t, \mathcal{D}_{\text{all}}, \alpha) := \frac{1}{1-t} \sum_{i=1}^{N} \left[\underset{i' \in [N]}{\text{softmax}} \left(-\frac{\|x_t - t x_1^{i'}\|^2}{2(1-t)^2} + \alpha r_{i'} \right)_i - \underset{i' \in [N]}{\text{softmax}} \left(-\frac{\|x_t - t x_1^{i'}\|^2}{2(1-t)^2} - \alpha r_{i'} \right)_i \right] x_1^i} \tag{16}$$

where α also controls the strength of the guidance signal. Furthermore, given a collection of K datasets $\{\mathcal{D}_{\text{all}}^k\}_{k=1}^K$ optimized for distinct rewards, we can naturally extend this formulation to achieve

[5] See Equation (16-practical) for the corresponding practical implementation.

compositional multi-target generation. Specifically, we construct a mixed guidance field by linearly combining the guidance fields associated with the K targets:

$$\Delta_{\text{mix}}(x_t) = \sum_{k=1}^{K} \Delta(x_t; \mathcal{D}_{\text{all}}^k, \alpha_k), \tag{17}$$

where each α_k controls the guidance strength and trade-off of each k-th objective. This mixture of guidance field guides the flow model to generate samples that simultaneously satisfy multiple target rewards, without requiring further reward evaluations or multi-objective optimization.

3.4 Properties and Advantages

In this section, we highlight several properties and advantages of Flow-Direct.

Regularized Guidance. The guidance direction induced by the non-parametric field is a linear combination of empirical data points. Consequently, the guidance vector is confined to the linear subspace spanned by the dataset. Under the data manifold hypothesis, the guidance field is approximately tangent to the underlying data manifold, which helps regularize the guided distribution and keeps it close to the data manifold, thereby mitigating the artifacts and degeneration commonly observed in unconstrained guidance methods.

Sampler-Agnostic. Flow-Direct directly modifies the continuous vector field, thus it is fundamentally agnostic to the choice of numerical sampler. This contrasts with SMC-based or search-based methods, which rely on specific SDE processes for stochastic exploration, and their performance can be sensitive to sampler configurations and noise schedules. As a result, our framework ensures stable performance and broader applicability across different sampler implementations.

Tuning-Free. Flow-Direct (Algorithm 1) relies on a closed-form derivation of the guidance field. In contrast to heuristic search or gradient-based methods, it does not introduce complex, sensitive parameters such as learning rates, momentum, or step sizes. Aside from the reward evaluation budget (batch size N and optimization iterations L), the formulation requires no optimization-specific tuning, significantly reducing implementation complexity.

4 Experiments

4.1 Target Distribution Optimization

In this section, we evaluate Flow-Direct (Algorithm 1) against baseline methods on image alignment tasks and 3D optimization task, using a matched reward-evaluation budget.

Pre-trained Flow Models. For the image task, we use Stable Diffusion 3 [2] (`stable-diffusion-3.5-medium`) as our text-to-image model; for the 3D task, we use TRELLIS [15] (`microsoft/TRELLIS-text-xlarge`) as our text-to-3D model. For both models, we adopt identical sampling parameters: sampling steps $T = 50$, classifier-free guidance scale $\texttt{cfg} = 4.5$, and the SDE sampler described in Equation (40) with noise scale $\eta = 0.7$.

Reward Functions. For the image task, we evaluate against six external reward functions. *Aesthetic* [16] and *HPSv3* [17] are pre-trained predictors that estimate perceptual image quality and human preference, respectively. *Compressibility* and *Incompressibility* [18] reward small and large JPEG-compressed file size, respectively. For *Attribute Alignment*, we use open source VLM Gemma [19] to evaluate whether the generated image matches a specified semantic attribute of the animal prompt; for *Prompt Alignment*, we use the VLM Gemini [20, 21] through paid API to evaluate the alignment with a specified target prompt. The implementation details are deferred to Appendix G and Appendix H.

For the 3D task, we consider vehicle aerodynamic optimization by minimizing the drag coefficient. Specifically, we prompt the model with *car* to generate a 3D car model, and utilize the DoMINO [22] to evaluate its drag coefficient, with implementation details provided in Appendix I.

We consider reward evaluations to be expensive and impose a fixed budget of 800 reward evaluations per run for the *Prompt Alignment* task, and 1600 for all other tasks.

Experimental Procedure. We implement Flow-Direct (Algorithm 1) with batch size $N = 16$ for all tasks, and set the number of optimization iterations to $L = 50$ for the *Prompt Alignment* task

and $L = 100$ for all other tasks, so that $N \cdot L$ matches the given reward evaluation budget. We compare Flow-Direct against four inference-time baseline methods: Evolvable [10], FK-Steering [11], TreeG [12], and DSearch [13]. For each, we adjust batch-size-related hyperparameters to match the reward evaluations budget and keep all remaining hyperparameters at their officially recommended values. We use the same hyperparameters for all methods for both image and 3D tasks.

For *Aesthetic*, *HPSv3*, *Compressibility*, and *Incompressibility*, we run each method on six simple animal prompts; for *Attribute Alignment*, we assign each animal with a semantic target attribute and repeat each over three seeds, for *Prompt Alignment*, we use the complex prompt and repeat each over three seeds. The specific prompts are stated in Table 11 and Appendix H. For the aerodynamic optimization, we repeat each experiment over three seeds.

Table 1: Reward value (higher is better) achieved by each method across all reward functions. For each run we record the average reward over the final batch of generated samples, and report the mean and standard deviation across runs. Note: As *Compressibility* and *Aerodynamic* are minimization tasks, their values are inverted.

Algorithm	Aesthetic	HPSv3	Compressibility	Incompressibility	Attribute Alignment						Prompt Alignment	Aerodynamic
					Happy	Fluffy	Running	Cute	Vivid	Fierce		
Evolvable	6.37±.54	8.57±.96	−26.18±2.18	262.09±47.35	0.18±.12	0.10±.03	0.00±.00	0.06±.01	0.00±.00	0.16±.05	2.0±2.6	−0.30±.18
FK-Steering	5.86±.30	6.02±1.86	−47.07±20.37	128.04±16.51	0.14±.06	0.01±.01	0.03±.04	0.27±.03	0.09±.01	0.20±.03	6.9±.7	−0.18±.02
TreeG	6.00±.34	8.40±1.13	−79.32±18.79	110.77±20.73	0.08±.03	0.05±.01	0.01±.01	0.10±.01	0.10±.01	0.10±.01	5.0±.4	−0.24±.02
DSearch	6.61±.33	9.90±1.50	−54.85±15.04	120.11±27.84	0.29±.02	0.26±.03	0.05±.05	0.47±.03	0.20±.02	0.26±.01	7.5±.8	−0.21±.01
Flow-Direct	**7.18**±.27	**10.94**±1.06	**−14.63**±7.99	**284.92**±28.09	**0.41**±.02	**0.42**±.02	**0.32**±.05	**0.52**±.07	**0.25**±.04	**0.30**±.01	**9.0**±.5	**−0.16**±.01

Table 2: Efficiency gain factor of Flow-Direct over each baseline. A factor of k indicates that Flow-Direct matches or surpasses the baseline's final reward using k times fewer reward evaluations.

Algorithm	Aesthetic	HPSv3	Compressibility	Incompressibility	Attribute Alignment						Prompt Alignment	Aerodynamic
					Happy	Fluffy	Running	Cute	Vivid	Fierce		
Evolvable	14.52×	45.40×	7.93×	3.07×	12.04×	21.43×	40.00×	83.33×	100.00×	25.00×	50.00×	77.78×
FK-Steering	61.70×	100.00×	16.02×	25.00×	16.39×	100.00×	10.07×	27.78×	25.00×	23.33×	4.09×	3.14×
TreeG	49.48×	46.94×	54.17×	54.63×	26.11×	27.78×	23.61×	66.67×	20.56×	50.00×	13.06×	31.48×
DSearch	5.81×	5.13×	20.50×	39.24×	3.62×	5.86×	8.13×	2.03×	2.15×	5.76×	2.78×	6.64×



Figure 2: Samples generated by each method across six representative reward functions. Flow-Direct produces images that clearly reflect the target reward while preserving image quality.

Quantitative Results. Table 1 shows the rewards achieved by all methods: for each run we record the average reward over the final batch of generated samples, and for each task we report the mean

and standard deviation across runs. Flow-Direct consistently achieves the highest reward across all seven reward functions. Table 2 shows the efficiency gain factor, computed as the reward evaluation budget divided by the number of reward evaluations Flow-Direct needed to match or surpass each baseline. A factor of k indicates that Flow-Direct reaches the same reward as the baseline using k times fewer reward evaluations.

In Appendix D, we additionally compare Flow-Direct against Best-of-N, which independently generates 1600 samples from the pre-trained model and selects the best sample. Table 8 shows Flow-Direct's final-batch mean already surpasses Best-of-N's maximum on 9 of 10 tasks, and significantly dominates on all 10 tasks under best-selection. This indicates that Flow-Direct optimizes the generative distribution, rather than merely retrieving the best sample from the accumulated dataset.

Qualitative Results. Figure 2 visualizes samples generated by each method across six representative reward functions, spanning both 2D image alignment and 3D vehicle aerodynamic optimization. Additional images and 3D vehicle results are provided in Appendices J.1 and J.2, respectively. Across both domains, Flow-Direct consistently produces samples that strongly reflect the target reward, while preserving the visual and structural quality. In contrast, baseline methods are less efficient at achieving these targets within the fixed reward evaluation budget.

Ablation Study. Flow-Direct (Algorithm 1) has only two hyperparameters: the number of optimization iterations L and the batch size N. We conduct an ablation study on these parameters in Appendix C. Results show that increasing either the optimization steps or the batch size yields a steady and stable improvement in the final reward, demonstrating the stability and scalability of Flow-Direct. Furthermore, unlike baseline methods, our formulation is compatible with both deterministic ODE and stochastic SDE samplers. In Table 6, we further compare Flow-Direct with the SDE sampler and the deterministic ODE sampler under the same reward-evaluation budget. The two variants achieve comparable rewards across reward functions, and both outperform all baselines, indicating that Flow-Direct is flexible and sampler-agnostic. Additionally, we analyze the computational overhead in Appendix C. The results show that the computational time and memory consumption of the guidance field $\Delta(x_t)$ both scale linearly with the dataset size N.

4.2 Reusable Target Datasets

We demonstrate that labeled datasets accumulated during optimization can be reused for target generation without additional reward evaluations. We construct a reusable guidance field by Equation (16) with $\alpha = 1$ using the dataset collected during main experiment (Table 1) with *Aesthetic* task and *dog* prompt. We then generate 1600 independent samples with the original (*dog*) and an unseen (*cat*) prompt, respectively. Note that no additional reward evaluation is used during this reusable guided generation. Table 3 shows that the guidance field improves both the mean and the best reward over the pre-trained model, even on the unseen prompt, and Figure 3 shows that the target style successfully transfers to both the original (*dog*) and an unseen (*cat*) prompt while preserving its visual quality.

Table 3: Mean and best reward value over 1600 independent samples generated by Flow-Direct's reusable guidance field. The reusable guidance field is constructed using the dataset collected during main experiment (Table 1) with *Aesthetic* task and *dog* prompt. No additional reward evaluation is used during the reusable run.

Reward	Method	*Dog* (seen)		*Cat* (unseen)	
		Mean	Best	Mean	Best
Aesthetic	Pre-trained	5.68	6.99	5.54	6.71
	Flow-Direct (reusable)	**6.44**	**7.41**	**6.13**	**7.10**
Compressibility	Pre-trained	−86.12	−41.91	−84.02	−43.04
	Flow-Direct (reusable)	**−8.65**	**−6.68**	**−9.02**	**−6.42**

Furthermore, multiple labeled datasets can be combined for multi-target generation. We apply Equation (17) using datasets optimized for the *Aesthetic* and *Compressibility* rewards on the *dog* prompt. We interpolate their scalar weights along $\alpha_{\text{aes}} + \alpha_{\text{com}} = 1$ to balance the trade-off. As seen in Figure 4, the images transition smoothly from high-compression to high-aesthetic. Additionally, this linear combination of distinct guidance fields can generalize to the unseen (*cat*) prompt, indicating its generalizability.



Figure 3: Novel and diverse images generated without additional reward evaluation by reusing the labeled dataset collected from the *Aesthetic*-optimized *dog* dataset. **Top:** prompt *dog* (matching the optimization). **Bottom:** prompt *cat* (unseen during optimization). The *Aesthetic* style transfers in both cases, demonstrating the generalization capability.

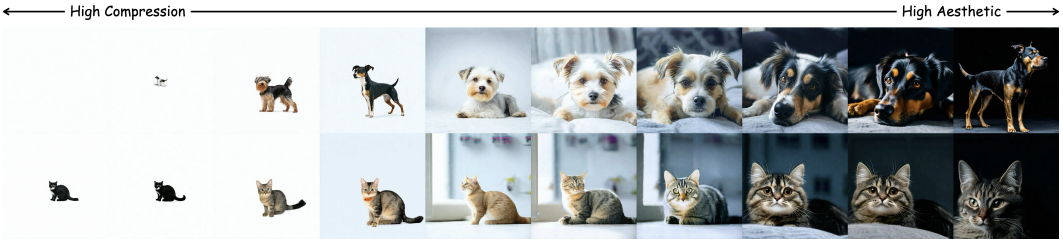


Figure 4: Compositional generation achieved without additional reward evaluation by mixing datasets optimized for *Aesthetic* and *Compressibility* on the *dog* prompt. **Top:** *dog* prompt (matching the optimization). **Bottom:** *cat* prompt (unseen during optimization). Both rows demonstrate a smooth interpolation between the two targets.

5 Conclusion, Limitations, and Broader Impacts

Conclusion. We presented Flow-Direct, a training-free framework that addresses the fundamental feedback inefficiency of existing inference-time guidance methods. By analytically deriving a guidance field from the log-density ratio of the base and target distributions and implementing it via a non-parametric estimator, Flow-Direct persistently aggregates all reward-evaluated samples. This approach is highly feedback-efficient because black-box feedback is preserved rather than transiently discarded. Our experiments across image alignment and 3D aerodynamic optimization demonstrate Flow-Direct's superior feedback-efficiency across diverse domains.

Limitations. While Flow-Direct is highly efficient in terms of reward evaluations, it is an iterative algorithm that requires executing the full generation trajectory multiple times. Consequently, unlike most existing guidance algorithms that perform guidance during a single generation pass, our approach incurs a higher computational cost with respect to neural function evaluations (NFEs). Therefore, Flow-Direct is intended for the expensive-reward regime where query-efficiency is the bottleneck. Additionally, because the framework assumes a continuous state space and differentiable score function to formulate the guidance vector field, it is not applicable to discrete diffusion models.

Broader Impacts. Flow-Direct may improve the efficiency of reward-guided generation for scientific and design applications, but it also inherits broader risks of generative models, including misuse in realistic synthetic content generation.

Acknowledgments and Disclosure of Funding

This research is partly supported by the National Research Foundation, Singapore under its NRF AI-for-Science (AI4S) Challenge Grant (Award NRF-AI4SCH-2025-0007). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] 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.

[2] 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. In *Forty-first international conference on machine learning*, 2024.

[3] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

[4] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024.

[5] Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, et al. Voicebox: Text-guided multilingual universal speech generation at scale. *Advances in neural information processing systems*, 36:14005–14034, 2023.

[6] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pages 8867–8887. PMLR, 2022.

[7] Jiaqi Guan, Wesley Wei Qian, Xingang Peng, Yufeng Su, Jian Peng, and Jianzhu Ma. 3d equivariant diffusion for target-aware molecule generation and affinity prediction. *arXiv preprint arXiv:2303.03543*, 2023.

[8] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[9] Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Sasha Shysheya, Jonathan Crabbé, Lixin Sun, Jake Smith, et al. Mattergen: a generative model for inorganic materials design. *arXiv preprint arXiv:2312.03687*, 2023.

[10] Zhao Wei, Chin Chun Ooi, Abhishek Gupta, Jian Cheng Wong, Pao-Hsiung Chiu, Sheares Xue Wen Toh, and Yew-Soon Ong. Evolvable conditional diffusion. *arXiv preprint arXiv:2506.13834*, 2025.

[11] 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. *arXiv preprint arXiv:2501.06848*, 2025.

[12] Yingqing Guo, Yukang Yang, Hui Yuan, and Mengdi Wang. Training-free guidance beyond differentiability: Scalable path steering with tree search in diffusion and flow models. *arXiv preprint arXiv:2502.11420*, 2025.

[13] Xiner Li, Masatoshi Uehara, Xingyu Su, Gabriele Scalia, Tommaso Biancalani, Aviv Regev, Sergey Levine, and Shuiwang Ji. Dynamic search for inference-time alignment in diffusion models. *arXiv preprint arXiv:2503.02039*, 2025.

[14] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[15] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025.

[16] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

[17] Yuhang Ma, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. Hpsv3: Towards wide-spectrum human preference score. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15086–15095, 2025.

[18] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

[19] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[20] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[21] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[22] Rishikesh Ranade, Mohammad Amin Nabian, Kaustubh Tangsali, Alexey Kamenev, Oliver Hennigh, Ram Cherukuri, and Sanjay Choudhry. Domino: A decomposable multi-scale iterative neural operator for modeling large scale engineering simulations. *arXiv preprint arXiv:2501.13350*, 2025.

[23] Quentin Bertrand, Anne Gagneux, Mathurin Massias, and Rémi Emonet. On the closed-form of flow matching: Generalization does not arise from target stochasticity. *arXiv preprint arXiv:2506.03719*, 2025.

[24] Mason Kamb and Surya Ganguli. An analytic theory of creativity in convolutional diffusion models. *arXiv preprint arXiv:2412.20292*, 2024.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[26] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla bayesian optimization performs great in high dimension. *arXiv preprint arXiv:2402.02229*, 2024.

[27] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

[28] Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models. *arXiv preprint arXiv:2509.25050*, 2025.

[29] Saurabh Singh and Ian Fischer. Stochastic sampling from deterministic flow models. *arXiv preprint arXiv:2410.02217*, 2024.

[30] Amita Kamath, Kai-Wei Chang, Ranjay Krishna, Luke Zettlemoyer, Yushi Hu, and Marjan Ghazvininejad. Geneval 2: Addressing benchmark drift in text-to-image evaluation. *arXiv preprint arXiv:2512.16853*, 2025.

A Derivations and Proofs

A.1 Preliminaries: Flow Matching Model

Given a noise distribution $x_0 \sim p_0 = \mathcal{N}(0, I)$ and a data distribution $x_1 \sim p_1$, flow matching framework [14] constructs a probability path between them via linear interpolation:

$$x_t = (1-t)x_0 + tx_1. \tag{18}$$

Conditional on a data point x_1, the intermediate state x_t is Gaussian:

$$p_t(x_t|x_1) = \mathcal{N}\big(x_t \big| tx_1, (1-t)^2 I\big) \tag{19}$$

$$= \frac{1}{(2\pi(1-t)^2)^{D/2}} \exp\left(-\frac{\|x_t - tx_1\|^2}{2(1-t)^2}\right). \tag{20}$$

Marginalizing over x_1 yields the marginal density along the probability path,

$$p_t(x_t) = \int p_t(x_t|x_1)p_1(x_1)dx_1, \tag{21}$$

and applying Bayes' rule yields the posterior over the clean data conditioned on x_t:

$$p_1(x_1|x_t) = \frac{p_t(x_t|x_1)p_1(x_1)}{p_t(x_t)} \propto p_1(x_1) \exp\left(-\frac{\|x_t - tx_1\|^2}{2(1-t)^2}\right). \tag{22}$$

Lipman et al. [14, Eq. 8] shows that the marginal probability path $p_t(x_t)$ is generated by a *marginal vector field* given as:

$$v_t(x_t) = \int v_t(x_t|x_1) \frac{p_t(x_t|x_1)p_1(x_1)}{p_t(x_t)}\, dx_1 \tag{23}$$

$$= \int \frac{x_1 - x_t}{1-t} p_1(x_1|x_t) dx_1 \tag{24}$$

$$= \frac{1}{1-t}\left(\int x_1 p_1(x_1|x_t)dx_1 - x_t \int p_1(x_1|x_t)dx_1\right) \tag{25}$$

$$= \frac{\mathbb{E}[x_1|x_t] - x_t}{1-t}, \tag{26}$$

where $v_t(x_t|x_1)$ is the conditional vector field that generates $p_t(x_t|x_1)$. The second line applies Bayes' rule from Equation (22), and the last line uses $\int p_1(x_1 \mid x_t)\, dx_1 = 1$.

A.2 Score–Vector Field Identity (Derivation of Equation (2))

We now derive the identity in Equation (2):

$$v_t(x_t) = \frac{x_t}{t} + \frac{1-t}{t}\nabla_{x_t} \log p_t(x_t), \tag{27}$$

which expresses the marginal vector field v_t in terms of the score of the probability path $p_t(x_t)$.

Score Function as Posterior Expectation. We first express the score function in terms of the posterior expectation by taking gradient of the marginal log-density defined in Equation (21):

$$\begin{aligned}
\nabla_{x_t} \log p_t(x_t) &= \frac{1}{p_t(x_t)} \nabla_{x_t} \int p_t(x_t|x_1)p_1(x_1)dx_1 \\
&= \frac{1}{p_t(x_t)} \int p_t(x_t|x_1) \nabla_{x_t} \log p_t(x_t|x_1) p_1(x_1)dx_1 \\
&= \int p_1(x_1|x_t) \nabla_{x_t} \log p_t(x_t|x_1) dx_1 \\
&= \int p_1(x_1|x_t) \frac{tx_1 - x_t}{(1-t)^2} dx_1 \\
&= \frac{t\mathbb{E}[x_1|x_t] - x_t}{(1-t)^2}, \\
\implies \mathbb{E}[x_1|x_t] &= \frac{x_t}{t} + \frac{(1-t)^2}{t} \nabla_{x_t} \log p_t(x_t),
\end{aligned} \tag{28}$$

where the first line applies the log-derivative identity $\nabla_{x_t} \log p = \nabla_{x_t} p / p$ and substitutes the marginal from Equation (21), the second line moves ∇_{x_t} inside the integral and reapplies $\nabla_{x_t} p = p \nabla_{x_t} \log p$ to the conditional, the third line applies Bayes' rule from Equation (22), the fourth line differentiates the log-density of Gaussian in Equation (20) with respect to x_t, and the fifth line uses the linearity of expectation. The last line rearranges in terms of the posterior expectation.

Vector field in terms of the score. Substituting Equation (28) into the marginal vector field in Equation (26),

$$\begin{aligned} v_t(x_t) &= \frac{\mathbb{E}[x_1|x_t] - x_t}{1-t} \\ &= \frac{1}{1-t}\left(\frac{x_t}{t} + \frac{(1-t)^2}{t}\nabla_{x_t} \log p_t(x_t) - x_t\right) \\ &= \frac{x_t}{t} + \frac{1-t}{t}\nabla_{x_t} \log p_t(x_t), \end{aligned} \tag{29}$$

which recovers Equation (2). Specializing this identity to the probability path p_t^θ generated by the pre-trained model and assuming the model is perfectly well-trained $v_\theta(x_t) \approx v_t(x_t)$ yields the Equation (4), completing the derivation.

A.3 Guided Vector Field as a Posterior-Expectation Difference (Derivation of Equation (6))

In Equation (5), the gradient of the log-density ratio between the two probability paths can be written as:

$$\begin{aligned} \nabla_{x_t} \log \frac{p_t^{\text{target}}(x_t)}{p_t^{\text{base}}(x_t)} &= \nabla_{x_t} \log p_t^{\text{target}}(x_t) - \nabla_{x_t} \log p_t^{\text{base}}(x_t) \\ &= \frac{t\,\mathbb{E}_{p_1^{\text{target}}}[x_1|x_t] - x_t}{(1-t)^2} - \frac{t\,\mathbb{E}_{p_1^{\text{base}}}[x_1|x_t] - x_t}{(1-t)^2} \\ &= \frac{t}{(1-t)^2}\left(\mathbb{E}_{p_1^{\text{target}}}[x_1|x_t] - \mathbb{E}_{p_1^{\text{base}}}[x_1|x_t]\right), \end{aligned} \tag{30}$$

where the second line applies the marginal-score identity from Equation (28) to each probability path, and the x_t terms cancel in the difference. Substituting Equation (30) into Equation (5) yields:

$$\begin{aligned} v_\theta^{\text{guided}}(x_t) &= \frac{x_t}{t} + \frac{1-t}{t}\left(\nabla_{x_t} \log p_t^\theta(x_t) + \frac{t}{(1-t)^2}\left(\mathbb{E}_{p_1^{\text{target}}}[x_1|x_t] - \mathbb{E}_{p_1^{\text{base}}}[x_1|x_t]\right)\right) \\ &= v_\theta(x_t) + \frac{1}{1-t}\left(\mathbb{E}_{p_1^{\text{target}}}[x_1|x_t] - \mathbb{E}_{p_1^{\text{base}}}[x_1|x_t]\right), \end{aligned} \tag{31}$$

which recovers Equation (6) and completes the derivation.

A.4 Proof of Proposition 1 (Estimator for Posterior Expectation)

Let $\mathcal{D} = \{x_1^i\}_{i=1}^N$ be a finite dataset of N samples drawn from the true data distribution $p_1(x_1)$, i.e., $x_1^i \overset{\text{iid}}{\sim} p_1(x_1)$. We approximate this unknown distribution using the empirical distribution $\hat{p}_1(x_1)$, represented as a uniform mixture of Dirac delta functions:

$$\hat{p}_1(x_1) \coloneqq \frac{1}{N}\sum_{i=1}^N \delta(x_1 - x_1^i) \xrightarrow{N\to\infty} p_1(x_1). \tag{32}$$

By the Glivenko-Cantelli theorem, the empirical measure $\hat{p}_1(x_1)$ converges to the true underlying data distribution $p_1(x_1)$ as the dataset size approaches infinity.

Following [23, 24], under the empirical data distribution, the empirical posterior expectation is:

$$\begin{aligned}
\mathbb{E}_{\hat{p}_1}[x_1|x_t] &= \frac{\int p_t(x_t|x_1)\hat{p}_1(x_1)x_1 dx_1}{\int p_t(x_t|x_1)\hat{p}_1(x_1)dx_1} \\
&= \frac{\int \mathcal{N}(x_t|tx_1,(1-t)^2 I)\big[\frac{1}{N}\sum_{i=1}^N \delta(x_1-x_1^i)\big]x_1 dx_1}{\int \mathcal{N}(x_t|tx_1,(1-t)^2 I)\big[\frac{1}{N}\sum_{i'=1}^N \delta(x_1-x_1^{i'})\big]dx_1} \\
&= \frac{\sum_{i=1}^N \mathcal{N}(x_t|tx_1^i,(1-t)^2 I)x_1^i}{\sum_{i'=1}^N \mathcal{N}(x_t|tx_1^{i'},(1-t)^2 I)} \\
&= \frac{\sum_{i=1}^N \exp\left(-\frac{\|x_t-tx_1^i\|^2}{2(1-t)^2}\right)x_1^i}{\sum_{i'=1}^N \exp\left(-\frac{\|x_t-tx_1^{i'}\|^2}{2(1-t)^2}\right)} \\
&= \sum_{i=1}^N \underset{i'\in[N]}{\mathrm{softmax}}\left(-\frac{\|x_t-tx_1^{i'}\|^2}{2(1-t)^2}\right)_i x_1^i. \qquad (33)
\end{aligned}$$

The first line expands the posterior expectation using Bayes' rule. In the second line, we substitute the conditional Gaussian density $p_t(x_t|x_1)$ and the empirical data distribution $\hat{p}_1(x_1)$. In the third line, by applying the sifting property of the Dirac delta function ($\int f(x)\delta(x-x_0)dx = f(x_0)$), the continuous integrals evaluate exactly to discrete sums over the dataset $\mathcal{D}$. The fourth line expands the Gaussian probability densities, which allows the shared normalization constant to cancel out and form the Softmax. When dataset size approaches infinity, then $\mathbb{E}_{\hat{p}_1}[x_1|x_t] \to \mathbb{E}_{p_1}[x_1|x_t]$, which recovers Equation (7) and completes the proof.

A.5 Proof of Proposition 2 (Estimator for Target Posterior Expectation)

Recall the target distribution is defined as:

$$p_1^{\text{target}}(x_1) \propto p_1^{\text{base}}(x_1)\exp(r(x_1)). \qquad (34)$$

Similarly, let $\mathcal{D} = \{x_1^i\}_{i=1}^N$ be a finite dataset of N samples drawn from the base distribution $p_1^{\text{base}}(x_1)$, i.e., $x_1^i \overset{\text{iid}}{\sim} p_1^{\text{base}}(x_1)$. We approximate the target distribution using the empirical target distribution:

$$\hat{p}_1^{\text{target}}(x_1) \coloneqq \frac{1}{\hat{Z}N}\sum_{i=1}^N \exp(r(x_1^i))\delta(x_1-x_1^i) \xrightarrow{N\to\infty} p_1^{\text{target}}(x_1), \qquad (35)$$

where $\hat{Z}$ is the empirical normalizing constant, and $r(x_1)$ is reward function. By the consistency of self-normalized importance sampling, the empirical measure $\hat{p}_1^{\text{target}}(x_1)$ converges to the true target data distribution $p_1^{\text{target}}(x_1)$ as the dataset size approaches infinity.

Following the similar derivation as Equation (33), the posterior expectation under this empirical target distribution is:

$$\begin{aligned}
\mathbb{E}_{\hat{p}_1^{\text{target}}}[x_1|x_t] &= \frac{\int p_t(x_t|x_1)\hat{p}_1^{\text{target}}(x_1)x_1 dx_1}{\int p_t(x_t|x_1)\hat{p}_1^{\text{target}}(x_1)dx_1} \\
&= \frac{\int \mathcal{N}(x_t|tx_1,(1-t)^2 I)\big[\frac{1}{\hat{Z}N}\sum_{i=1}^N \exp(r(x_1^i))\delta(x_1-x_1^i)\big]x_1 dx_1}{\int \mathcal{N}(x_t|tx_1,(1-t)^2 I)\big[\frac{1}{\hat{Z}N}\sum_{i'=1}^N \exp(r(x_1^{i'}))\delta(x_1-x_1^{i'})\big]dx_1} \\
&= \frac{\sum_{i=1}^N \mathcal{N}(x_t|tx_1^i,(1-t)^2 I)\exp(r(x_1^i))x_1^i}{\sum_{i'=1}^N \mathcal{N}(x_t|tx_1^{i'},(1-t)^2 I)\exp(r(x_1^{i'}))} \\
&= \frac{\sum_{i=1}^N \exp\left(-\frac{\|x_t-tx_1^i\|^2}{2(1-t)^2}+r(x_1^i)\right)x_1^i}{\sum_{i'=1}^N \exp\left(-\frac{\|x_t-tx_1^{i'}\|^2}{2(1-t)^2}+r(x_1^{i'})\right)} \\
&= \sum_{i=1}^N \underset{i'\in[N]}{\mathrm{softmax}}\left(-\frac{\|x_t-tx_1^{i'}\|^2}{2(1-t)^2}+r(x_1^{i'})\right)_i x_1^i. \qquad (36)
\end{aligned}$$

which recovers Equation (10) and completes the proof.

B Practical Implementation

We now discuss several practical considerations for implementing the posterior expectations in Propositions 1 and 2. Since both estimators share the same functional form, and the latter is simply the reward-weighted extension of the former, we focus our discussion on the latter; the following techniques apply equally to both. Recall the target posterior estimator in Equation (10):

$$\mathbb{E}_{p_1^{\text{target}}}[x_1|x_t] \approx \sum_{i=1}^{N} \underset{i' \in [N]}{\text{softmax}} \left(-\frac{\|x_t - t x_1^{i'}\|^2}{2(1-t)^2} + r_{i'} \right)_i x_1^i. \tag{10}$$

The Softmax logits dictate the influence of each data point by combining its ℓ_2 distance from the current state with its scalar reward. To prevent numerical instability during optimization, the distance metric must remain robust across high dimensionalities and shifting timesteps. Furthermore, the distance and reward terms must be calibrated to a similar scale so that neither dominates the guidance signal. We address these requirements through the following practical techniques:

- In high-dimensional space $\mathbb{R}^D$, the standard deviation of the squared ℓ_2 distance between two independent vectors scales with $\sqrt{D}$. We therefore normalize the squared distance term by $\sqrt{D}$ to ensure the variance for the Softmax logits is dimension-invariant.
- Although $t x_1^{i'} = \mathbb{E}[x_t^{i'}|x_1^{i'}]$ is the correct conditional mean, its norm shrinks toward the origin for $t < 1$, placing it on a different scale than the noisy state x_t. For numerical stability, we replace the conditional mean with a single stochastic sample:

$$t x_1^{i'} \leftarrow x_t^{i'} \coloneqq t x_1^{i'} + (1-t)\epsilon, \tag{37}$$

  for $\epsilon \sim \mathcal{N}(0, I)$. This ensures that x_t and $x_t^{i'}$ reside on the same scale.
- Under flow matching framework, the true posterior expectation must converge to current state, i.e., $\mathbb{E}[x_1|x_t] \to x_t$, when $t \to 1$. To strictly enforce this property in our estimator, we append the model's prediction of the clean data to the dataset:

$$\tilde{\mathcal{D}} \leftarrow \mathcal{D} \cup \{\hat{x}_{1|t}\}, \tag{38}$$

  where $\hat{x}_{1|t} = x_t + (1-t)v_\theta(x_t)$ is the predicted clean data.
- Building upon the previous technique, for the predicted clean data $\hat{x}_{1|t}$, we estimate its reward value using the existing dataset:

$$\mathbb{E}_{p_1^{\text{base}}}[r(x_1)|x_t] \approx \sum_{i=1}^{N} \underset{i' \in [N]}{\text{softmax}} \left(-\frac{\|x_t - x_t^{i'}\|^2}{2(1-t)^2\sqrt{D}} \right)_i r_i. \tag{39}$$

  This reuses existing reward labels and incurs no additional reward evaluations.
- Finally, we standardize the reward values to have zero mean and unit variance. For the iterative optimization described in Section 3.2, we normalize the rewards across all accumulated datasets $(\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{l-1})$ using the same global mean and standard deviation. This ensures that the reward scale remains consistent across all telescoping guidance terms.

We study the effect of these techniques in the ablation study in Appendix C.1.

Incorporating the above implementation techniques, we summarize the practical guidance fields used throughout the paper:[6]

- **Target Data Guidance (Section 3.1):**

$$\boxed{\Delta(x_t; \tilde{\mathcal{D}}_{\text{base}}, \tilde{\mathcal{D}}_{\text{target}}) \coloneqq \frac{1}{1-t}\left[\sum_{j=1}^{M} \underset{j' \in [M]}{\text{softmax}} \left(-\frac{\|x_t - x_t^{j'}\|^2}{2(1-t)^2\sqrt{D}} \right)_j x_1^j - \sum_{i=1}^{N} \underset{i' \in [N]}{\text{softmax}} \left(-\frac{\|x_t - x_t^{i'}\|^2}{2(1-t)^2\sqrt{D}} \right)_i x_1^i \right]} \tag{8-practical}$$

[6] With a slight overload of notation, N and M now denote the sizes of the augmented datasets $\tilde{\mathcal{D}}$.

- **Iterative Reward Optimization (Section 3.2):**

$$\Delta(x_t;\tilde{\mathcal{D}}) \coloneqq \frac{1}{1-t}\sum_{i=1}^{N}\left[\underset{i'\in[N]}{\mathrm{softmax}}\left(-\frac{\|x_t-x_t^{i'}\|^2}{2(1-t)^2\sqrt{D}}+r_{i'}\right)_i-\underset{i'\in[N]}{\mathrm{softmax}}\left(-\frac{\|x_t-x_t^{i'}\|^2}{2(1-t)^2\sqrt{D}}\right)_i\right]x_1^i$$

(11-practical)

- **Reusable Guidance Fields (Section 3.3):**

$$\Delta(x_t,\tilde{\mathcal{D}}_{\text{all}},\alpha) \coloneqq \frac{1}{1-t}\sum_{i=1}^{N}\left[\underset{i'\in[N]}{\mathrm{softmax}}\left(-\frac{\|x_t-x_t^{i'}\|^2}{2(1-t)^2\sqrt{D}}+\alpha r_{i'}\right)_i-\underset{i'\in[N]}{\mathrm{softmax}}\left(-\frac{\|x_t-x_t^{i'}\|^2}{2(1-t)^2\sqrt{D}}-\alpha r_{i'}\right)_i\right]x_1^i$$

(16-practical)

C Ablation Study

C.1 Practical Implementation Techniques

The practical implementation in Appendix B comprises four techniques:

1. Normalizing the softmax logits by $\sqrt{D}$.
2. Standardizing reward values to zero mean and unit variance.
3. Regularizing the dataset with the model's predicted clean sample (Equations (38) and (39)).
4. Estimating the conditional expectation with a noisy sample (Equation (37)).

Technique 1 is essential. It is standard rather than novel to our method, following the logit scaling used in attention layers [25] and RBF-kernel normalization [26]. Without it, the squared ℓ_2 distance grows with the dimension D, so the softmax saturates to a near one-hot weighting over the dataset and the estimated posterior expectation collapses onto a single sample.

Technique 2 is essential. It is standard rather than novel to our method, and is common practice in reward-based generative optimization [27, 18, 28]. Raw reward values differ by orders of magnitude across tasks (e.g., Aesthetic $\in [1, 10]$, Incompressibility $\approx 10^2$, Attribute Alignment $\in [0, 1]$). Without it, the reward weighting could either over-concentrate on a few samples or become uniform, requiring a per-task scaling coefficient to be tuned.

Technique 3 preserves perceptual image quality. We observe that adding the model's predicted clean sample to the dataset can improve the perceptual quality of the generated images. This is reflected in Table 4: with the regularized dataset, Flow-Direct performs better on the quality-oriented rewards (Aesthetic, HPSv3), but worse on the semantic rewards (Attribute Alignment). This is because appending the model's own predicted clean sample anchors the guidance field to the pre-trained data, which regularizes the guidance to remain near the pre-trained distribution.

Table 4: Ablation study on practical implementation techniques (regularized dataset and noisy expectation estimation). The last row is the default Flow-Direct reported in Table 1. Note: As *Compressibility* is a minimization task, its values are inverted.

Regularized Dataset (Technique 3)	Noisy Expectation Estimation (Technique 4)	Aesthetic	HPSv3	Compressibility	Incompressibility	Attribute Alignment					
						Happy	Fluffy	Running	Cute	Vivid	Fierce
✗	✗	6.980±.131	9.475±1.230	−14.550±5.591	290.547±11.676	0.501±.064	0.450±.014	0.334±.090	0.618±.066	0.309±.010	0.355±.027
✗	✓	6.968±.105	9.425±1.075	−12.897±6.068	287.992±9.231	0.478±.042	0.432±.019	0.273±.001	0.700±.036	0.312±.020	0.352±.014
✓	✗	7.185±.270	10.918±1.108	−12.117±6.328	295.753±30.628	0.323±.018	0.429±.021	0.299±.050	0.514±.041	0.256±.019	0.315±.015
✓	✓	7.181±.274	10.944±1.055	−14.632±7.993	284.920±28.089	0.406±.022	0.424±.018	0.324±.054	0.521±.065	0.253±.036	0.300±.007

Technique 4 preserves sample diversity. We visually observe that noisy estimation of the conditional expectation produces more diverse samples within the generated batch. To quantify this, we report the mean pairwise ℓ_2 distance within each generated batch in Table 5. With noisy expectation estimation, the generated batch consistently yields a higher pairwise distance, indicating higher diversity, across all ten reward functions. We did not visually observe that noisy expectation estimation affects the image quality.

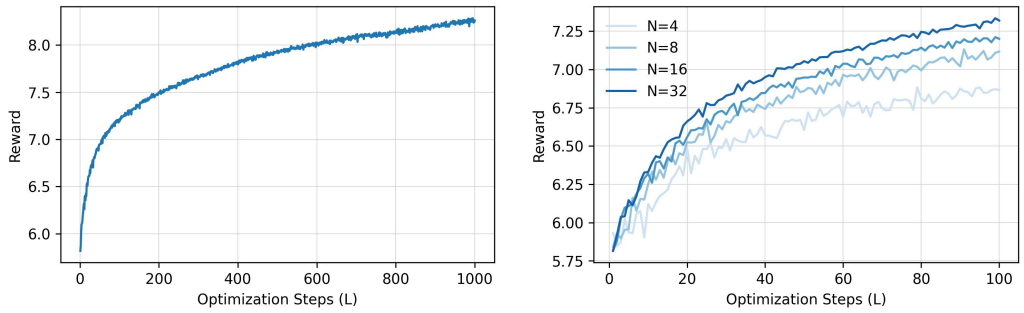
Table 5: Pairwise ℓ_2 distance on noisy expectation estimation.

Noisy Expectation Estimation (Technique 4)	Aesthetic	HPSv3	Compressibility	Incompressibility	Attribute Alignment					
					Happy	Fluffy	Running	Cute	Vivid	Fierce
✗	93.9±7.8	90.8±8.2	34.3±32.0	116.1±28.6	91.4±4.2	103.7±3.4	110.2±6.9	95.3±15.4	86.3±2.1	105.5±2.8
✓	95.8±9.2	95.3±7.1	50.6±48.1	120.7±25.1	98.8±2.5	107.8±3.9	115.5±9.9	112.1±8.1	91.4±3.1	117.2±1.4

C.2 Optimization Steps and Batch Size

We study the effect of the two hyperparameters in Flow-Direct: the number of optimization iterations L and the batch size N. For each setting, we run six independent runs with distinct prompts to optimize the Aesthetic reward, and report the average reward.

In Figure 5a, we extend the optimization iteration to $L = 1000$, which corresponds to 16,000 reward evaluations when using batch size $N = 16$. The reward increases steadily throughout optimization, indicating that Flow-Direct continues to benefit from additional reward-evaluated samples. In Figure 5b, we vary the batch size over $N \in \{4, 8, 16, 32\}$. Larger batch sizes achieve higher rewards and produce more stable improvement curves. This is because each iteration contributes more labeled samples to the empirical dataset, yielding a more accurate posterior-expectation estimation. These results indicate both scalability and stability: Flow-Direct improves steadily as either the optimization horizon or batch size increases.



(a) (b)

Figure 5: **Ablation study on scalability.** We report the average Aesthetic reward across six runs with distinct prompts. **(a)** Increasing the number of optimization steps continues to improve reward. **(b)** Larger batch sizes yield higher rewards.

C.3 ODE vs. SDE Sampler

We study the effect of the sampler used by Flow-Direct. Specifically, we compare the performance of Flow-Direct under the SDE sampler with $\eta = 0.7$ and the ODE sampler with $\eta = 0.0$ in Equation (40). All other hyperparameters and the reward-evaluation budget are kept the same as in the main experiment. We report the results in Table 6. We observe that both variants achieve comparable rewards across all reward functions. Moreover, both Flow-Direct (SDE) and Flow-Direct (ODE) achieve higher rewards than all baseline methods reported in Table 1, showing that Flow-Direct is effective under both stochastic and deterministic samplers.

Table 6: Reward value achieved by Flow-Direct with SDE and ODE samplers.

Algorithm	Aesthetic	HPSv3	Compressibility	Incompressibility	Attribute Alignment						Aerodynamic
					Happy	Fluffy	Running	Cute	Vivid	Fierce	
Flow-Direct (SDE)	**7.18**±0.27	10.94±1.05	−14.63±7.99	**284.92**±28.09	**0.41**±0.02	0.42±0.02	**0.32**±0.05	0.52±0.07	0.25±0.04	**0.30**±0.01	−0.16±0.01
Flow-Direct (ODE)	7.15±0.29	**11.26**±1.18	**−8.77**±6.66	273.07±33.69	**0.41**±0.03	**0.45**±0.01	0.23±0.10	**0.58**±0.07	**0.27**±0.02	0.29±0.00	**−0.14**±0.01

C.4 Computational Resource

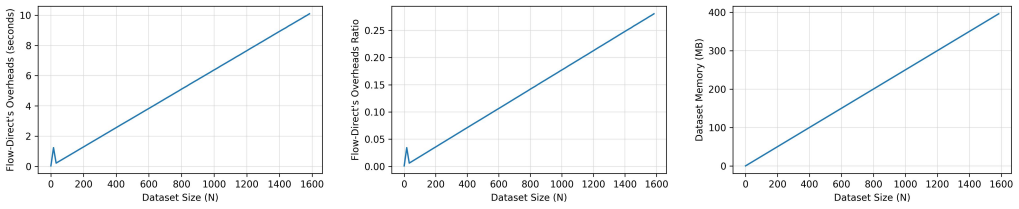
We report the total runtime for each task in Table 7, measured on a single NVIDIA A100 40GB GPU. To analyze the computational overhead introduced by our framework, we measure the total

time spent computing the guidance field $\Delta(x_t)$ over one generation pass with $T = 50$ timesteps and batch size $N = 16$, during the optimization process while the dataset size N expanding. Note that all reported time measurements exclude the time required for reward evaluations. Figure 6a reports the absolute wall-clock overhead in seconds, and Figure 6b reports the relative overhead ratio, defined as the Flow-Direct overhead divided by the wall-clock time of the (unguided) pre-trained flow model generation. The memory consumption is shown in Figure 6c.

Both runtime and memory scale linearly[7] with the dataset size N. At $N = 1600$, Flow-Direct incurs approximately 10 seconds of computational wall-clock time. This introduces roughly a 30% time overhead compared to the unguided pre-trained model's generation, while consuming 400 MB of memory.

Table 7: Total runtime for each task, measured on a single NVIDIA A100 40GB GPU.

Task	Runtime
Aesthetic	1h 31m
HPS-v3	1h 18m
Compress	1h 9m
Incompress	1h 12m
Attribute Alignment	1h 14m
Aerodynamic	5h 33m



(a) (b) (c)

Figure 6: **Ablation study on computational overhead and memory.** Flow-Direct's computation time and memory consumption scale linearly with the dataset size N.

D Best-of-N Experiment

In this section, we ablate whether Flow-Direct can produce better samples than the pre-trained distribution. We follow the same experiment procedure in Table 1, and implement Best-of-N, which independently samples $N = 1600$ samples from the pre-trained model, and report the best reward in Table 8. For comparison, we report the mean and best reward achieved by Flow-Direct in Table 8.

Table 8: Best-of-N under a matched reward-evaluation budget (1600 evaluations). *Best-of-N* reports the maximum reward over 1600 independently generated samples. *Flow-Direct (final-batch mean)* is the average reward over the final batch of 16 samples, as reported in Table 1. *Flow-Direct (best)* reports the maximum reward over its 1600 accumulated samples. Note: As *Compressibility* is a minimization task, its values are inverted.

Algorithm	Aesthetic	HPSv3	Compressibility	Incompressibility	Attribute Alignment					
					Happy	Fluffy	Running	Cute	Vivid	Fierce
Best-of-N	6.76±.28	10.27±1.34	−34.88±12.73	170.93±25.51	0.36±.03	0.28±.01	0.13±.00	0.53±.00	0.23±.03	0.26±.00
Flow-Direct (final-batch mean)	7.18±.27	10.94±1.06	−14.63±7.99	284.92±28.09	0.41±.02	0.42±.02	0.32±.05	0.52±.07	0.25±.04	0.30±.01
Flow-Direct (best)	7.45±.25	11.70±1.19	−10.81±6.51	298.48±27.95	0.50±.03	0.51±.02	0.43±.06	0.64±.06	0.30±.04	0.37±.00

Table 8 shows that Flow-Direct's average performance already matches or exceeds Best-of-N's *best* sample. The final-batch mean (row 2) surpasses Best-of-N (row 1) on nine of ten rewards, and is statistically indistinguishable on the remaining one (Cute: $0.521\pm.065$ vs. $0.527\pm.000$). This is the

[7]The small spike at the $N = 16$ is likely due to one-time GPU memory allocation during the first function calls.

stronger comparison because it is an unselected average over 16 samples in the final batch, whereas the Best-of-N is a maximum over 1600. This indicates that Flow-Direct optimizes the generative distribution, rather than merely retrieving the best sample from the accumulated dataset.

Moreover, comparing the two maxima over 1600 samples (rows 1 and 3), Flow-Direct achieves a consistently higher reward on all ten reward functions. This is because independent sampling is limited by the support of the pre-trained distribution: reranking can only retrieve the best sample the base model happens to produce, whereas Flow-Direct progressively transports the sampling distribution toward the high-reward region.

Finally, we note that Best-of-N returns a single retained sample, while Flow-Direct returns a persistent guidance field, from which novel high-reward samples can be drawn without any further reward evaluation.

E Related Works

In this section, we review training-free guidance methods that support black-box rewards. These methods guide the flow model generation process by using the scalar feedback from a non-differentiable reward function. Existing approaches differ in how they use reward feedback: some approximate a local update to the denoising distribution, some maintain multiple candidate trajectories and reallocate computation based on intermediate reward estimates, and others formulate generation as an explicit search over the denoising trajectory.

Evolvable. Evolvable [10] treats each denoising transition as a local black-box optimization problem. At an intermediate state, it samples multiple candidates from the current denoising distribution, queries the reward function on these candidates, and uses the resulting scalar rewards to update the mean of the next-step distribution toward higher-reward regions. This provides a derivative-free analogue of gradient guidance, since the update is estimated from reward-weighted perturbations rather than from backpropagation through the reward function.

FK-Steering. FK-Steering [11] adopts an SMC-based particle filtering procedure to steer generation toward a reward-tilted target distribution. It maintains a population of denoising trajectories as particles, evaluates their utility using the reward function, and performs resampling so that particles with higher reward are duplicated while lower-reward particles are discarded. This particle filtering view makes the method compatible with arbitrary black-box rewards, since guidance only requires scalar reward value rather than gradients.

TreeG and DSearch. TreeG [12] formulates training-free guidance as a tree search problem over the iterative generation process. At each timestep, it selects the high-reward nodes to branch out, while pruning low-reward nodes. This effectively steers the generation trajectory toward high-reward terminal sample. Because this process relies solely on scalar feedback, it natively supports black-box reward functions. DSearch [13] builds on the same framework and dynamically allocates the search budget across denoising steps. It maintains more active nodes in early denoising steps to preserve trajectory diversity, and gradually reduces the number of active nodes while increasing the branching factor in later steps. This leverages the coarse-to-fine nature of the diffusion process to allocate reward evaluations more efficiently across denoising steps.

SDE for Flow Model. The methods discussed above rely on stochastic sampling dynamics, either to approximate local guidance directions or to explore multiple candidate generation trajectories. To apply them to flow models, we use the stochastic flow sampler from [27, 29], whose discretization step is given by:

$$x_{t+\Delta t} = \underbrace{a_t x_t + b_t v_\theta(x_t, t) + c_t \epsilon_t}_{\texttt{step}(x_t, v_t)}, \tag{40}$$

for vector field $v_t = v_\theta(x_t, t)$, noise $\epsilon_t \sim \mathcal{N}(0, I)$, step size Δt, and the coefficients are given by:

$$a_t = 1 - \frac{\sigma_t^2 \Delta t}{2(1-t)}, \qquad b_t = \Delta t \left(1 + \frac{t\sigma_t^2}{2(1-t)}\right), \qquad c_t = \sigma_t \sqrt{\Delta t}. \tag{41}$$

Following Flow-GRPO [27], we set $\sigma_t = \eta\sqrt{(1-t)/t}$ with noise level $\eta = 0.7$ in all experiments. This SDE sampler preserves the same marginal probability path as the deterministic flow ODE. When $\eta = 0$, the update reduces to the deterministic ODE discretization in Equation (3).

F Implementation Details for Section 3.1 Demo

This section provides the implementation to produce the Figure 1.

Data Preparation. We consider three guidance tasks, where the goal is to steer the flow model toward a specific target attribute: *wings*, *sketch*, or *cartoon*. For each task, we construct a pair of base and target datasets, each containing 64 generated samples. All datasets are generated using the same pre-trained flow model, and the guidance computation is performed in latent space. The prompts used to construct these datasets are summarized in Table 9.

Table 9: Base and target datasets used in the dataset-guidance demonstration. Each base and target dataset contains 64 generated latent samples.

Task	Data pair	
Wings	$\mathcal{D}^{\text{base}}_{\text{wings}}$:	`a puppy on grass field.`
	$\mathcal{D}^{\text{target}}_{\text{wings}}$:	`a puppy with golden wings on grass field.`
Sketch	$\mathcal{D}^{\text{base}}_{\text{sketch}}$:	`a realistic photo of a puppy.`
	$\mathcal{D}^{\text{target}}_{\text{sketch}}$:	`a sketch drawing of a puppy.`
Cartoon	$\mathcal{D}^{\text{base}}_{\text{cartoon}}$:	`a realistic photo of a puppy.`
	$\mathcal{D}^{\text{target}}_{\text{cartoon}}$:	`a cartoon puppy.`

Guidance Field. For each target, we compute the term according to Equation (8). We also test composite target *sketch + cartoon* by summing the guidance field. In the demo, we additionally scale each guidance field by a scale α to make the visual effect more apparent. This scale is introduced only for the qualitative demonstration in this section and is not used in the rest of the paper. The concrete formulation is stated in Table 10.

Table 10: Guidance field implementation for each task.

Target	Guidance field	α
Wings	$\Delta_{\text{wings}}(x_t) \coloneqq \alpha\,\Delta(x_t; \mathcal{D}^{\text{base}}_{\text{wings}}, \mathcal{D}^{\text{target}}_{\text{wings}})$	3
Sketch	$\Delta_{\text{sketch}}(x_t) \coloneqq \alpha\,\Delta(x_t; \mathcal{D}^{\text{base}}_{\text{sketch}}, \mathcal{D}^{\text{target}}_{\text{sketch}})$	2
Cartoon	$\Delta_{\text{cartoon}}(x_t) \coloneqq \alpha\,\Delta(x_t; \mathcal{D}^{\text{base}}_{\text{cartoon}}, \mathcal{D}^{\text{target}}_{\text{cartoon}})$	2
Sketch + Cartoon	$\Delta_{\text{sketch+cartoon}}(x_t) \coloneqq \alpha\,(\Delta_{\text{sketch}}(x_t) + \Delta_{\text{cartoon}}(x_t))$	1

Generation. We evaluate the generation procedure using two inference prompts:

`a puppy on grass field.` `a cat sit on grass field.`

The first *puppy* prompt matches the dataset construction setting, whereas the second *cat* prompt creates a distribution mismatch with the guidance datasets. The generated images are reported in Figure 1.

G Implementation for Attribute Alignment

In this section, we describe the implementation details for the *attribute alignment* reward. The goal of this reward is to assess whether a generated image aligns with a specified semantic attribute.

We evaluate the generated image using an open-source vision-language model (VLM). Specifically, we use Gemma 4 [19] (`google/gemma-4-E4B-it`). We format the image and text as a joint multimodal input message, and ask the VLM the following question:

```
Is the {prompt} {attribute}?  Strictly answer "yes" or "no"
without any explanation.
```

The placeholder `{prompt}` represents the subject and `{attribute}` is the target semantic attribute.

Rather than decoding the VLM's output into a binary reward value, we implement Soft-TIFA [30] for a soft reward value. Specifically, Soft-TIFA extracts the VLM's response logits and computes the probability of the `"yes"` token. This probability then serves as a continuous reward within range $[0, 1]$, effectively quantifying the model's confidence that the image matches the target attribute.

In our experiments, each animal prompt is paired with one predefined target attribute, as stated in Table 11. During optimization, the flow model is optimized to generate images that match the corresponding target attribute.

Table 11: Target attribute used in our experiment.

Animal prompt	Target attribute
`dog`	`happy`
`cat`	`fluffy`
`horse`	`running`
`bear`	`cute`
`bird`	`vivid`
`lion`	`fierce`

H Implementation for Prompt Alignment

In this section, we describe the implementation details for the *prompt alignment* reward. The goal is to assess whether a generated image aligns with the specific complex prompt. This is a genuinely expensive reward, since each evaluation requires a paid LLM API query, so query efficiency becomes more important than memory or computational efficiency.

Specifically, we align the flow model to the following challenging prompt that involves object counting and spatial arrangement:

```
Seven apples arranged in a perfect circle, each in a distinct
rainbow color (red, orange, yellow, green, blue, indigo, and
violet), and a single transparent glass apple in the center.
```

Each generated image is evaluated by Gemini [20, 21] (`gemini-3.6-flash`) at a cost of $0.002611 (USD) per query. The actual query to the LLM is:

```
Rate how accurately this image matches the text prompt:
"{prompt}"
Assign a rating from 0 to 10, where 0 means does not match at
all, and 10 means perfect match.
```

The placeholder `{prompt}` represents the target text prompt. We constrain the LLM to respond with a structured output containing a single numerical rating, and use this rating directly as the reward value within range $[0, 10]$. We set the decoding temperature to 0, with the thinking level and the image resolution both set to `low`.

We compare all methods under matched query budgets of 200, 400, and 800 evaluations, repeat each experiment over three seeds, and report the average rating over the final batch of generated samples for each method in Table 12.

Table 12: Average rating under a matched LLM API query budget, with the corresponding cost. For each run we record the average rating over the final batch of generated samples, and report the mean and standard deviation across three seeds.

Algorithm	200 ($0.52)	400 ($1.04)	800 ($2.09)
Evolvable	1.8±2.8	1.2±2.0	2.0±2.6
FK-Steering	5.6±2.7	5.9±1.5	6.9±.7
TreeG	4.9±1.5	4.9±1.2	5.0±.4
DSearch	**7.7**±1.7	6.8±.9	7.5±.8
Flow-Direct	6.8±.3	**7.9**±.4	**9.0**±.5

I Implementation Details for 3D Vehicle Optimization

I.1 3D Generation Model

For the 3D vehicle generation task, we utilize the TRELLIS [15] text-to-3D model (`microsoft/TRELLIS-text-xlarge`). The TRELLIS pipeline consists of two sequential stages: 1) generating a sparse structure, and 2) decoding and refining this sparse structure into a dense, high-quality 3D mesh. In our experiments, we apply the guidance *exclusively* to the first stage. We fix the generation prompt to *car*, with the goal of guiding the first stage to generate highly aerodynamic cars.

I.2 DoMINO Reward Evaluation

To evaluate the aerodynamic performance of the generated cars, we utilize the DoMINO predictive model [22] (`nvcr.io/nim/nvidia/domino-automotive-aero:2.1.0-41313772`).

The primary optimization objective for this task is the *drag coefficient* (C_d). While the DoMINO API directly outputs the raw drag force (F_d) evaluated on the vehicle mesh, we calculate the dimensionless drag coefficient using the standard aerodynamic equation:

$$C_d = \frac{F_d}{\frac{1}{2}\rho V^2 A}$$

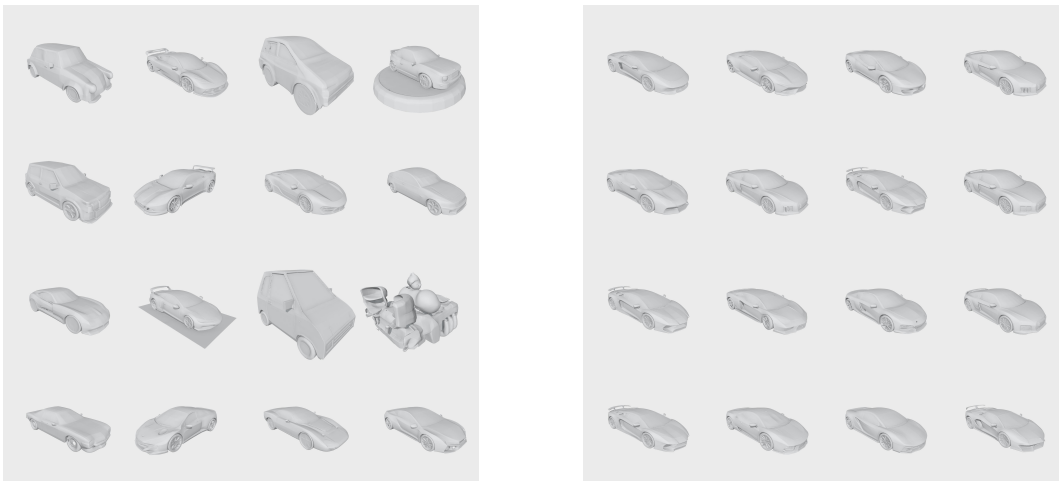
where:

- F_d is the drag force in Newtons returned by the DoMINO model.
- A is the frontal area of the vehicle mesh (in square meters), calculated by projecting the silhouette of the generated vehicle onto the YZ-plane.
- ρ is the standard air density at sea level, set to a constant $\rho = 1.225$ kg/m^3.
- V is the freestream velocity, set to $V = 30.0$ m/s.

J Additional Qualitative Results

J.1 3D Vehicle Aerodynamic Optimization

In this section, we provide additional qualitative results for the 3D vehicle aerodynamic optimization task. In Figure 7, we compare 16 randomly generated 3D vehicle meshes produced by the unguided pre-trained TRELLIS model with those optimized by Flow-Direct. We observe that the optimized car structures exhibit highly aerodynamic shapes while preserving their overall structural integrity.



(a) Unguided (b) Flow-Direct

Figure 7: Qualitative results for 3D vehicle aerodynamic optimization.

J.2 Image Alignment

In this section, we provide additional qualitative results for all reward functions considered in the main experiments. Each figure shows samples generated by different inference-time guidance methods under the same reward-evaluation budget. Columns correspond to prompts or target attributes, and rows correspond to the optimization method. These results complement Figure 2 and further illustrate that Flow-Direct consistently produces samples that better reflect the target reward while preserving image quality.

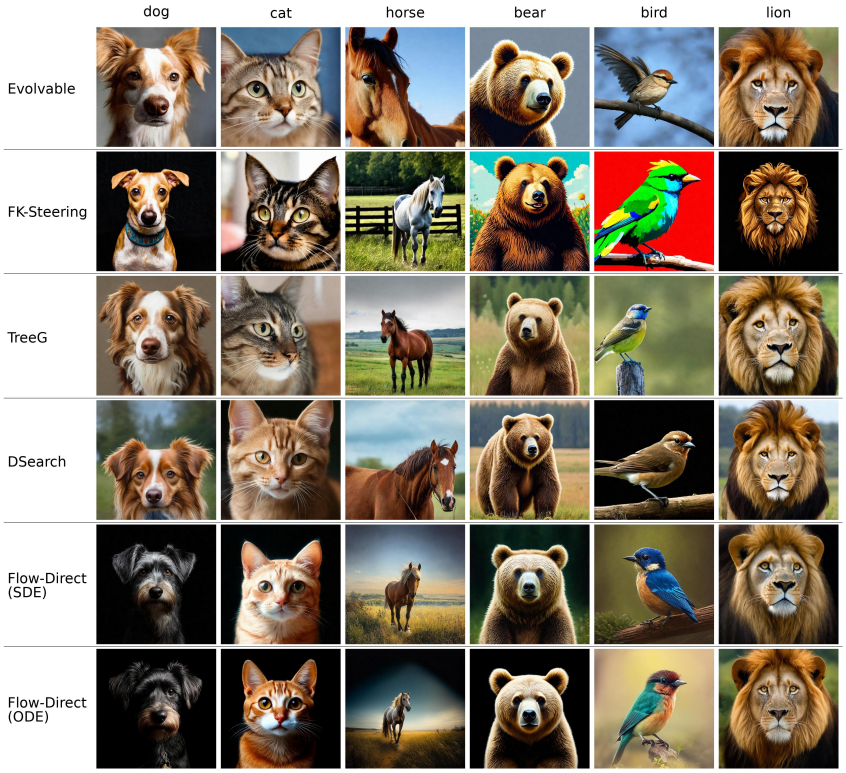


Figure 8: Additional qualitative results for the *Aesthetic* reward.

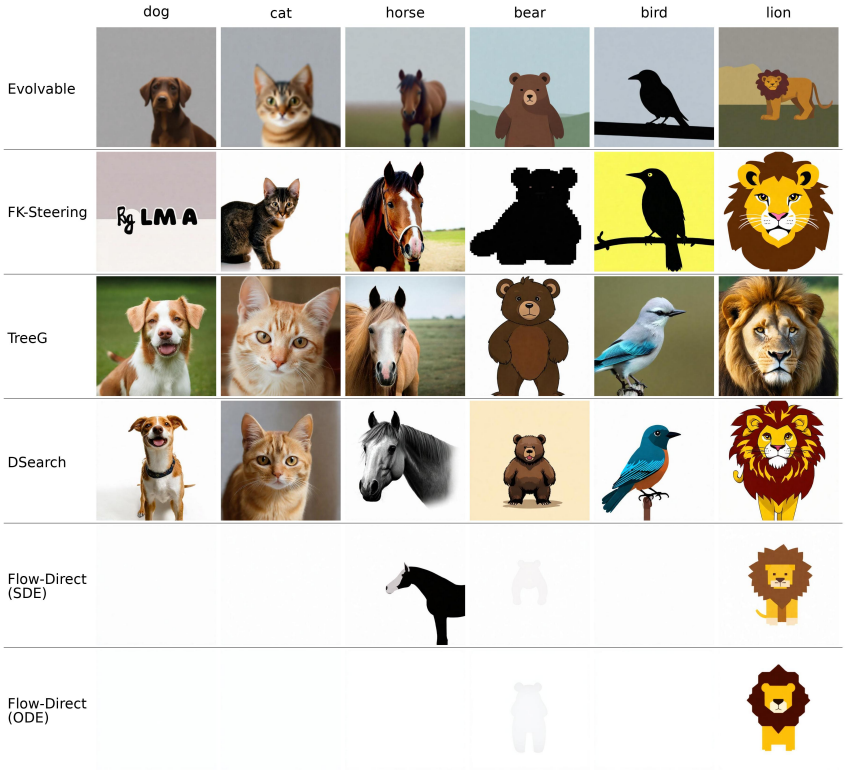


Figure 9: Additional qualitative results for the *Compressibility* reward.



Figure 10: Additional qualitative results for the *Incompressibility* reward.



Figure 11: Additional qualitative results for the *HPSv3* reward.

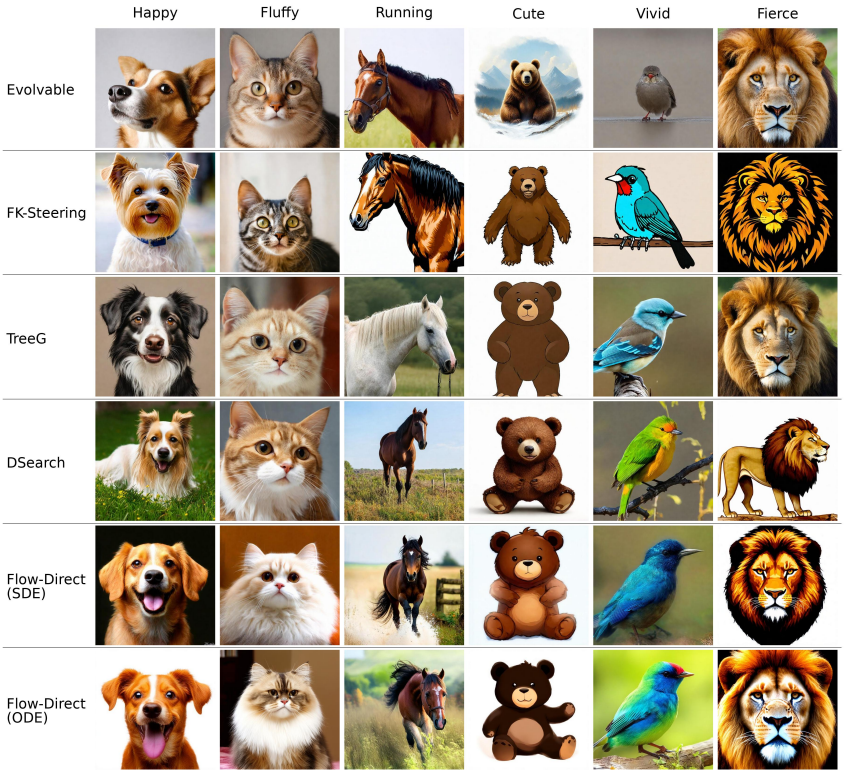


Figure 12: Additional qualitative results for the *Attribute Alignment* reward.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We make three claims: (1) a guidance field formulation, (2) feedback-efficiency, (3) reusability. Claim (1) is justified theoretically in Section 3 and Appendix A; claims (2) and (3) are justified empirically by the experiments in Sections 4.1 and 4.2 respectively.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 5 specifically discusses the limitations.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The two main theoretical results, Propositions 1 and 2, are stated formally with assumptions in Section 3, with the complete derivations are provided in Appendix A.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Algorithm 1 fully specifies the proposed method, with the practical implementation details provided in Appendix B. Other experimental details are provided in Appendices F to I.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The source code is publicly available at: `https://github.com/kimyong95/finetune-stable-diffusion`.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 4.1 specifies all the hyperparameters.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Tables 1 and 3 report mean reward value ± standard deviation across multiple runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The compute resources are stated in Appendix C.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics: it does not involve human subjects, the pre-trained generative models and reward predictors are publicly available academic releases, and no personally identifiable data is used.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Section 5 specifically discusses both positive and negative broader impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Flow-Direct operates on top of open source models during inference-time, we do not release any new data or models, so no additional release safeguards are required.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All pre-trained models and baseline methods used in the paper are properly cited, and we use them in accordance with their respective public release licenses and intended research use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper is algorithmic, and does not release any new datasets or pre-trained models.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve any crowdsourcing or human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve any human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [No]

    Justification: LLM is not as a component of the proposed method.

    Guidelines:

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