VALUE MATCHING: SCALABLE AND GRADIENT-FREE REWARD-GUIDED FLOW ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting large-scale flow and diffusion models to downstream tasks through reward optimization is essential for their adoption in real-world applications, including scientific discovery and image generation. While recent fine-tuning methods based on reinforcement learning and stochastic optimal control achieve compelling performance, they face severe scalability challenges due to high memory demands that scale with model complexity. In contrast, methods that disentangle reward adaptation from base model complexity, such as Classifier Guidance (CG), offer flexible control over computational resource requirements. However, CG suffers from limited reward expressivity and a train-test distribution mismatch due to its offline nature. To overcome the limitations of fine-tuning methods and CG, we propose **V**alue **M**atching (VM), an online algorithm for learning the value function within an optimal control setting. VM provides tunable memory and compute demands through flexible value network complexity, supports optimization of non-differentiable rewards, and operates on-policy, which enables going beyond the data distribution to discover high-reward regions. Experimentally, we evaluate VM across image generation and molecular design tasks. We demonstrate improved stability and sample efficiency over CG and achieve comparable performance to fine-tuning approaches while requiring less than 5% of their memory usage.

1 INTRODUCTION

Large-scale generative foundation models have recently made remarkable progress. Among them, flow (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023; Liu et al., 2023) and diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015; Song et al., 2020) stand out for their ability to generate high-fidelity samples across a wide range of domains, including images (Ho et al., 2020), chemistry (Hoogeboom et al., 2022), biology (Corso et al., 2023), and robotics (Chi et al., 2023). For many applications (*e.g.*, controllable image editing and drug discovery (Olivecrona et al., 2017)), adapting such large pre-trained models to downstream rewards is essential. Existing approaches based on reinforcement learning (RL) (Zhao et al., 2025; Black et al., 2023; Fan et al., 2023; Hu et al., 2025) and stochastic optimal control (SOC) (Domingo-Enrich et al., 2025; Uehara et al., 2024; Tang, 2024; Domingo-Enrich et al., 2024) rely on backpropagating through the full model to update model weights. This makes them increasingly memory-intensive as model sizes scale to billions of parameters, as illustrated in Figure 1.

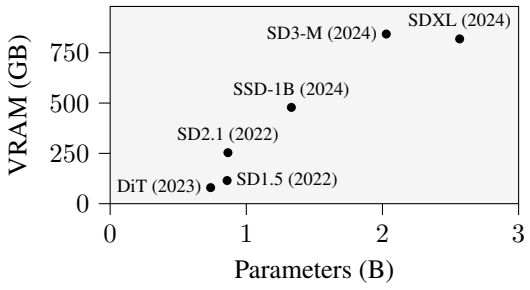


Figure 1: The recent trend toward more complex flow and diffusion generative models leads to prohibitively high fine-tuning memory (VRAM) requirements.

Despite their promising performance (*e.g.*, Domingo-Enrich et al., 2025; Zhao et al., 2025), RL and SOC-based fine-tuning methods remain fundamentally limited by resource requirements that scale with model size. Moreover, many state-of-the-art approaches require the reward function to be differentiable (*e.g.*, Domingo-Enrich et al., 2025), which severely limits applicability to black-box optimization settings where only function evaluations are available without access to structural information. This constraint is problematic for real-world applications such as drug discovery, where molecular

property prediction relies on complex simulators (Bannwarth et al., 2019; Sholl & Steckel, 2009; Forli et al., 2016) or experimental measurements (Hughes et al., 2011). This raises a key question:

How can we leverage SOC (or RL) machinery to adapt flow and diffusion models to non-differentiable rewards with controllable resource usage and competitive performance?

Limitations of fine-tuning methods suggest reconsidering approaches like classifier guidance (CG) (Dhariwal & Nichol, 2021) that offer computational advantages by avoiding updates to base model parameters and enables adaptation with non-differentiable rewards. However, we show that CG has notable limitations, such as limited expressivity and a train-test distribution mismatch. We address these shortcomings by proposing **V**alue **M**atching (VM), a control-theoretic algorithm that learns the value function online, enabling discovery of high-reward regions beyond the data distribution. Further, we analyze how VM differs from fine-tuning approaches such as Adjoint Matching (AM) (Domingo-Enrich et al., 2025) and Continuous-Time PPO (CT-PPO) (Zhao et al., 2025), notably not requiring fine-tuning and avoiding reward gradients. Lastly, we show that VM achieves performance competitive with CT-PPO at significantly lower resource requirements.

Table 1: Algorithm capabilities.

	VM	CG	AM	CT-PPO
No fine-tuning	✓	✓	✗	✗
Gradient-free	✓	✓	✗	✓
Online data	✓	✗	✓	✓
# Hyperparams.	1	1	1	5

Our Contributions:

- A control-theoretic viewpoint shedding light on how the offline nature of classifier guidance hinders the discovery of high-reward samples beyond the data distribution (Section 4).
- **V**alue **M**atching (VM), a theoretically-grounded online method for reward adaptation of flow models through value function learning, highlighting advantages compared to AM and CT-PPO (Section 5).
- An experimental evaluation across image and molecular generation tasks using non-differentiable rewards, showing that VM (i) achieves superior stability and sample efficiency compared to classifier guidance, (ii) reduces resource requirements by 95% relative to fine-tuning methods while maintaining comparable performance, and (iii) enables effective adaptation of molecular and large-scale text-to-image models through value networks significantly smaller than the base model (Section 6).

2 Background and Notation

Flow Models. In this work, we consider the problem of adapting flow (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023; Liu et al., 2023) and diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015; Song et al., 2020), which have emerged as leading approaches for generative modeling across various domains, including images (Rombach et al., 2022; Peebles & Xie, 2023; Podell et al., 2024; Gupta et al., 2024; Esser et al., 2024), text (Gat et al., 2024), and molecular design (Hoogeboom et al., 2022; Dunn & Koes, 2024a;b). Typically, flow models are sampled through an ordinary differential equation (ODE) (Lipman et al., 2023), however, they can also be sampled via a stochastic differential equation (SDE) with equal time marginals (Maoutsa et al., 2020). We sample by simulating an SDE:

$$\mathrm{d}\mathbf{x}_t = b(\mathbf{x}_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \quad \mathbf{x}_0 \sim p_0, \tag{1}$$

where $b : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is the drift, $\sigma : [0,1] \to \mathbb{R}^{d\times d}$ is the diffusion, and B_t is a Brownian motion. For unbiased adaptation, we use a memoryless parameterization (Domingo-Enrich et al., 2025):

$$b(\mathbf{x}, t) \triangleq \frac{\dot{\alpha}_t}{\alpha_t}\mathbf{x} + \sigma^2(t)\nabla \log p_t(\mathbf{x}_t), \quad \sigma^2(t) = 2\beta_t\left(\frac{\dot{\alpha}_t}{\alpha_t}\beta_t - \dot{\beta}_t\right), \tag{2}$$

with (α_t, β_t) defined as the flow schedule (Lipman et al., 2023) and $(\dot{\alpha}_t, \dot{\beta}_t)$ being time derivatives.

Stochastic Optimal Control (SOC). For our approach, we frame KL-regularized adaptation as an SOC problem (Bellman & Dreyfus, 2015; Fleming & Rishel, 2012), a general framework that deals with optimization over stochastic processes. Specifically, we restrict ourselves to a quadratic cost control-affine Bolza problem (Bolza, 1904) on a finite time horizon $[0, 1]$ with dynamics:

$$\mathrm{d}\mathbf{x}_t = (b(\mathbf{x}_t, t) + \sigma(t)u(\mathbf{x}_t, t))\,\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \quad \mathbf{x}_0 \sim p_0. \tag{3}$$

Further, the cost functional J is defined as the total cost of a control u starting from a point $(\mathbf{x}, t)$, composed of a quadratic running cost $\frac{1}{2}\|u(\mathbf{x}_t, t)\|^2$ and an arbitrary terminal cost $g : \mathbb{R}^d \to \mathbb{R}$. The

control problem is to find the control u that minimizes the cost functional J at every point $(\mathbf{x}, t)$:

$$u^\star \in \arg\min_{u \in \mathcal{U}} J(u; \mathbf{x}, t) \triangleq \mathbb{E}_{p^u}\left[\frac{1}{2}\int_t^1 \|u(\mathbf{x}_s, s)\|^2 \,\mathrm{d}s + g(\mathbf{x}_1) \;\middle|\; \mathbf{x}_t = \mathbf{x}\right]. \tag{4}$$

From here, the value function is defined as the optimal value of the cost functional:

$$V(\mathbf{x}, t) \triangleq \inf_{u \in \mathcal{U}} J(u; \mathbf{x}, t) = J(u^\star; \mathbf{x}, t). \tag{5}$$

Further, V can be defined through the base distribution (Domingo-Enrich et al., 2024, Appendix B):

$$V(\mathbf{x}, t) = -\log \mathbb{E}_{p_1^{\mathrm{pre}}}[\exp(-g(\mathbf{x}_1)) \mid \mathbf{x}_t = \mathbf{x}]. \tag{6}$$

3 Problem Setting

We consider the generative optimization problem (Li et al., 2024b; De Santi et al., 2025a;b) of adapting a pre-trained flow or diffusion model p^{pre} to maximize a reward function $r : \mathbb{R}^d \to \mathbb{R}$ in expectation, weighted by $\lambda \in \mathbb{R}_{\geq 0}$, and remain close to p^{pre} in terms of Kullback-Leiber (KL) divergence. Formally, we optimize over policies π with induced last-timestep marginal p_1^π:

$$\arg\max_{\pi} \mathbb{E}_{p_1^\pi}[\lambda r(\mathbf{x}_1)] - D_{\mathrm{KL}}(p_1^\pi \parallel p_1^{\mathrm{pre}}) \;\; \text{s.t.} \;\; \mathrm{d}\mathbf{x}_t = b^\pi(\mathbf{x}_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t. \tag{7}$$

The optimal solution $\pi^\star$ of Problem (7) induces the tilted distribution $p_1^\star(\mathbf{x}) \propto p_1^{\mathrm{pre}}(\mathbf{x}) \exp(\lambda r(\mathbf{x}))$ if σ follows a memoryless noise schedule (Domingo-Enrich et al., 2025). In flow models, this objective is equivalent to the following quadratic cost control-affine control problem (Tang, 2024):

$$\begin{aligned} &\arg\min_{u:\mathbb{R}^d \times [0,1] \to \mathbb{R}^d} \mathbb{E}\left[\frac{1}{2}\int_t^1 \|u(\mathbf{x}_s, s)\|^2 \,\mathrm{d}s - \lambda r(\mathbf{x}_1) \;\middle|\; \mathbf{x}_t = \mathbf{x}\right] \\ &\text{s.t.} \;\; \mathrm{d}\mathbf{x}_t = (b^{\mathrm{pre}}(\mathbf{x}_t, t) + \sigma(t) u(\mathbf{x}_t, t))\,\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t. \end{aligned} \tag{8}$$

Existing fine-tuning methods (*e.g.*, Domingo-Enrich et al., 2025; Zhao et al., 2025) encounter scalability challenges because they require updating the base model parameters. To perform these updates, the gradients of the loss function must be backpropagated through the entire model. This process is highly memory-intensive, as calculating the gradients requires storing all intermediate activations of the model. As a result, the memory footprint scales with model size, posing a major bottleneck as these models grow to billions of parameters. Next, we introduce an approach that addresses this issue.

4 Disentangling Optimization through Value Function Learning

In this work, we advocate for learning the value function V of Equation (5) to solve Problem (7), which offers compelling advantages over current fine-tuning methods: (1) support of non-differentiable rewards (*cf*. Xu et al., 2023; Clark et al., 2024; Domingo-Enrich et al., 2025) and (2) controllable resource usage. Once learned, the value function allows us to find the optimal control $u^\star$ through Pontryagin's minimum principle (Pontryagin, 1962), where the first-order optimality condition gives:

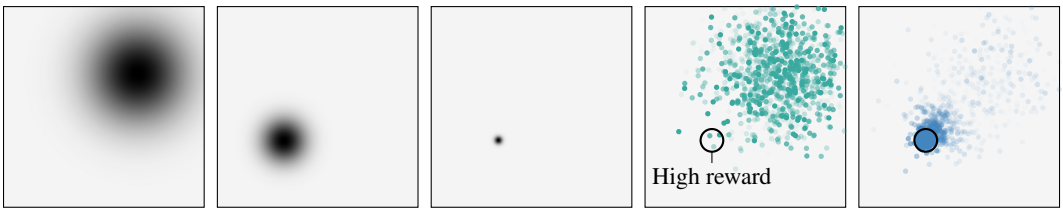
$$u^\star(\mathbf{x}, t) = -\sigma^\intercal(t) \nabla_{\mathbf{x}} V(\mathbf{x}, t). \tag{9}$$

A key insight for Problem (7) is that the value function retains differentiability even when the reward function is not, ensuring well-defined optimal control. Intuitively, this occurs because stochastic noise acts as a smoothing kernel; the value function V at time t averages over all noise realizations from $\mathbf{x}_t$ to $\mathbf{x}_1$, effectively regularizing reward discontinuities (see Figure 3). We formalize this as follows.

Proposition 1. *Under the memoryless noise schedule and assuming that r is bounded and measurable, the value function V is differentiable in $\mathbf{x}$ at $t < 1$.*

Proof outline. We write $V(\mathbf{x}, t) = -\log \psi(\mathbf{x}, t)$ where $\psi(\mathbf{x}, t) = \mathbb{E}_{p^{\mathrm{pre}}}[\exp(\lambda r(\mathbf{x}_1)) \mid \mathbf{x}_t = \mathbf{x}]$. Then apply the chain rule and show that $\psi > 0$ and that ψ is differentiable (see Appendix A.3).

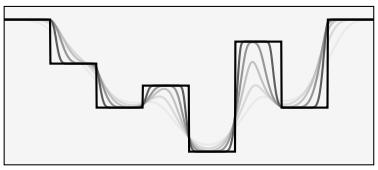
This proposition enables the optimization of non-differentiable reward functions commonly used in practical applications, rendering this approach *gradient-free*. It proves particularly valuable in settings where reward functions are treated as black boxes, providing only function evaluations without access to gradients or structural information. For instance, in molecular generation, reward functions often rely on external simulators (Bannwarth et al., 2019; Sholl & Steckel, 2009; Forli et al., 2016)



(a) Base model p_1^{pre}. (b) Reward r. (c) Optimal $p_1^\star$. (d) CG training data. (e) VM training data.

Figure 2: Training distribution evolution for VM and CG on a 2D environment, showing that VM aligns its training distribution with the optimal distribution throughout training, whereas CG does not. (2d, 2e) Points with lower opacity represent earlier training stages.

or experimental measurements (Hughes et al., 2011) that only return scalar values. In such cases, value function learning makes reward adaptation possible, as opposed to methods that rely on reward gradients (*e.g.*, Xu et al., 2023; Clark et al., 2024; Domingo-Enrich et al., 2025).



$t = 0.66$ $t = 0.74$ $t = 0.81$
$t = 0.88$ $t = 0.94$ $t = 1.00$

Figure 3: Evolution of value function for a discontinuous reward.

Beyond handling non-differentiable rewards, learning the value function disentangles reward adaptation from base model optimization. Consequently, the dominating computational cost shifts from base model training to base model inference and value function learning. This leads to *controllable resource usage* by allowing for the flexible choice of a value network architecture. As we show in Section 6, the value function can be made significantly smaller than the base model, resulting in a substantial reduction in cost, while achieving comparable performance to fine-tuning methods. Because of this significant cost reduction, the value function learning approach is highly practical for the adaptation of large-scale models where fine-tuning would be prohibitively expensive.

Having established the advantages of value function learning, we next recall how classifier guidance can be viewed as an offline algorithm to learn the value function V.

Classifier Guidance as Offline Value Function Learning. As previously established by Pandey et al. (2025), Classifier Guidance (CG) (Dhariwal & Nichol, 2021) admits an interpretation as value function learning. The approach involves training a classifier $p_{Y|t}(y \mid \mathbf{x})$ over noisy samples $\mathbf{x}$ at timesteps $t \in [0, 1]$ and using its log-gradient $\nabla_{\mathbf{x}} \log p_{Y|t}(y \mid \mathbf{x})$ to guide generation. This procedure can be understood as leveraging Equation (6) to solve Problem (7) with the reward function $r(\mathbf{x}) = \log p_{Y|1}(y \mid \mathbf{x})$ and value function $V(\mathbf{x}, t) = -\log p_{Y|t}(y \mid \mathbf{x})$ (see Appendix A.5 for the derivation), which in turn motivates a generalized loss function for arbitrary rewards:

$$\mathcal{L}_{\text{CG}}(\boldsymbol{\theta}; \mathbf{x}_{[0,1]}) \triangleq \frac{1}{2} \int_0^1 |\exp(-V_{\boldsymbol{\theta}}(\mathbf{x}_t, t)) - \exp(\lambda r(\mathbf{x}_1))|^2 \, \mathrm{d}t, \qquad (10)$$
$$\mathrm{d}\mathbf{x}_t = b^{\text{pre}}(\mathbf{x}_t, t) \, \mathrm{d}t + \sigma(t) \, \mathrm{d}B_t, \quad \mathbf{x}_0 \sim p_0.$$

Throughout the remainder of this work, we use CG to refer to this loss formulation. Next, we discuss issues with this approach for generative optimization (Li et al., 2024b; De Santi et al., 2025a;b).

Limitations of Classifier Guidance for Generative Optimization. The CG formulation in Equation (10) represents an offline value function learning approach because it trains on samples from the fixed, pre-trained distribution p_t^{pre} rather than the current policy distribution p_t^u. This offline nature is inherent to the loss formulation itself: the expectation in Equation (6) is computed over the pre-trained distribution, necessitating that the value function is trained on samples from this fixed source.

However, for generative optimization tasks such as reward maximization, we wish to sample high reward designs beyond regions of high data availability (Li et al., 2024b; De Santi et al., 2025a;b). This objective reveals a fundamental limitation of the offline approach: a *distribution mismatch* arises between the fixed training distribution p_t^{pre} and the target distribution $p_t^\star$ that the learned policy encounters during inference (see Figure 2). As the policy shifts probability mass toward higher-reward regions, the training samples from p_t^{pre} become increasingly less informative, reducing sample efficiency and potentially limiting the method's ability to discover optimal solutions.

Algorithm 1 Value Matching algorithm.

Require: Pre-trained base model with sampling SDE $\mathrm{d}\mathbf{x}_t = b(\mathbf{x}_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t$, Reward function $r : \mathbb{R}^d \to \mathbb{R}$, Untrained value function approximator $V_{\boldsymbol{\theta}} : \mathbb{R}^d \times [0,1] \to \mathbb{R}$, Number of iterations $N \in \mathbb{N}$, Trajectories per iteration $m \in \mathbb{N}$, Timestep-dependent weighting $w : [0,1] \to \mathbb{R}_{>0}$.

1: **for** N iterations **do**

2: Sample m trajectories under the *current policy*:

$$\mathrm{d}\mathbf{x}_{i,t} = \left(b^{\mathrm{pre}}(\mathbf{x}_{i,t}, t) - \sigma^2(t)\nabla V_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_{i,t}, t)\right)\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \quad \mathbf{x}_{i,0} \sim p_0, \quad i \in [m].$$

3: Estimate the *cost functional* for each timestep in each trajectory with $\bar{\boldsymbol{\theta}} = \texttt{stopgrad}(\boldsymbol{\theta})$:

$$\hat{J}_{i,t} = \frac{1}{2}\int_t^1 \|\sigma(s)\nabla V_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_{i,s}, s)\|^2\,\mathrm{d}s - \lambda r(\mathbf{x}_{i,1}), \quad t \in [0,1], i \in [m].$$

4: Compute the *loss function*:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\int_0^1 w(t)\cdot|V_{\boldsymbol{\theta}}(\mathbf{x}_{i,t}, t) - \hat{J}_{i,t}|^2\,\mathrm{d}t.$$

5: Make an *optimization step* with $\nabla\mathcal{L}(\boldsymbol{\theta})$.

6: **end for**

Additionally, the exponential terms in Equation (10) create numerical stability issues during training, due to overflows when $\lambda r(\mathbf{x}) > 90$ under 32-bit floating-point precision. This constrains the method's expressivity to small reward scalings λ, as we show in Section 6. To address these limitations, we leverage the control-theoretic viewpoint to next introduce an online value function learning approach that aligns the training and inference distributions while maintaining numerical stability.

5 Value Matching: Scalable and Gradient-Free Reward-Guided Adaptation

We introduce **V**alue **M**atching (VM), an online method for learning the value function that overcomes a fundamental limitation of CG by training on trajectories from the current policy. As detailed in Algorithm 1, VM leverages Equations (5) and (9) to estimate the value function. This is achieved by iteratively regressing the approximator $V_{\boldsymbol{\theta}}$ onto the cost functional J, computed using the current policy.

$$\begin{aligned}
\mathcal{L}_{\mathrm{VM}}(\boldsymbol{\theta}; \mathbf{x}_{[0,1]}) &\triangleq \frac{1}{2}\int_0^1 w(t)\cdot|V_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \hat{J}(-\sigma^{\intercal}\nabla_{\mathbf{x}}V_{\bar{\boldsymbol{\theta}}}; \mathbf{x}_{[0,1]}, t)|^2\,\mathrm{d}t, \\
\hat{J}(u; \mathbf{x}_{[0,1]}, t) &\triangleq \frac{1}{2}\int_t^1 \|u(\mathbf{x}_s, s)\|^2\,\mathrm{d}s - \lambda r(\mathbf{x}_1), \quad \bar{\boldsymbol{\theta}} = \texttt{stopgrad}(\boldsymbol{\theta}) \\
\mathrm{d}\mathbf{x}_t &= \left(b^{\mathrm{pre}}(\mathbf{x}_t, t) - \sigma^2(t)\nabla_{\mathbf{x}}V_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_t, t)\right)\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \quad \mathbf{x}_0 \sim p_0.
\end{aligned} \tag{11}$$

The VM algorithm follows a simple iterative procedure. Each iteration begins by sampling trajectories using the control policy $u(\mathbf{x}, t) = -\sigma^{\intercal}(t)\nabla_{\mathbf{x}}V_{\boldsymbol{\theta}}(\mathbf{x}, t)$, derived from the current value function approximator $V_{\boldsymbol{\theta}}$. These trajectories serve a dual purpose in the training loop. First, they are used to compute single-sample Monte Carlo estimates $\hat{J}_t$ of the cost functional J, which serves as the regression target:

$$\hat{J}_t = \frac{1}{2}\int_t^1 \sigma^2(s)\|\nabla_{\mathbf{x}}V_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_s, s)\|^2\,\mathrm{d}s - \lambda r(\mathbf{x}_1). \tag{12}$$

Here, $\bar{\boldsymbol{\theta}} = \texttt{stopgrad}(\boldsymbol{\theta})$ prevents gradients from flowing through the target, ensuring that $\hat{J}_t$ is treated as a fixed target value during backpropagation. Second, the states $\mathbf{x}_t$ along these same trajectories provide the input data for the value function. The network parameters $\boldsymbol{\theta}$ are updated by regressing the model's prediction $V_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ onto the target $\hat{J}_t$ using an ℓ_2-loss:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\int_0^1 w(t)\cdot|V_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \hat{J}_t|^2\,\mathrm{d}t. \tag{13}$$

To stabilize training under the memoryless schedule where $\sigma(t) \to \infty$ as $t \to 0$, we incorporate a weighting $w : [0, 1] \to \mathbb{R}_{>0}$. Empirically, we find the following scheme effective (see Figure 10):

$$w(t) = \frac{1}{\lambda^2\left(1 + \frac{1}{2}\int_t^1 \sigma^2(s)\,\mathrm{d}s\right)}. \tag{14}$$

This scheme normalizes rewards by the scaling factor λ and down-weights timesteps with high future variance. For models employing multiple schedulers, weights are averaged across schedulers.

Finally, the weights $\boldsymbol{\theta}$ are updated using a gradient descent step with $\nabla\mathcal{L}(\boldsymbol{\theta})$. For numerical implementation, the underlying SDE is simulated with the Euler-Maruyama method (Kloeden & Platen, 1992) using T steps, and the integrals are approximated as Riemann sums (Anton, 1999) over the same discretization. The procedure to efficiently compute Equation (12) is outlined in Appendix B.3. In the following result, we show that this approach converges to the true value function in expectation.

Proposition 2. *The value function V is the unique critical point of $\mathbb{E}[\mathcal{L}_{\mathrm{VM}}]$.*

Proof outline. We first establish that V is a stationary point by computing the functional derivative of $\mathbb{E}[\mathcal{L}_{\mathrm{VM}}]$. Uniqueness then follows as a standard result in SOC (see Appendix A.4).

This proposition provides theoretical justification for using gradient-based methods to optimize $V_{\boldsymbol{\theta}}$, ensuring that VM converges to the correct value function under appropriate conditions. To build intuition for VM, we next show how it relates to current state-of-the-art fine-tuning algorithms.

Value Matching as the Gradient-Free Analogue of Adjoint Matching. Conceptually, Adjoint Matching (AM) (Domingo-Enrich et al., 2025) can be understood as learning the value function gradient $\nabla_{\mathbf{x}} V$ by iteratively matching it to single-sample Monte Carlo estimates of $\nabla_{\mathbf{x}} J$. Our method represents a zeroth-order analogue, where V is learned by regressing onto estimates of J, and the gradient is obtained via backpropagation. Thus, AM and VM are procedurally very similar.

Value Matching Simplifies Continuous-Time PPO. Zhao et al. (2025) introduced a Continuous-Time Proximal Policy Optimization (CT-PPO) algorithm that learns the optimal control by iteratively alternating between training a value function and using it to optimize an actor network, starting from the pre-trained model (see Algorithm 2). We argue that by setting the actor to $s^{\mathrm{pre}}(\mathbf{x}, t) - \nabla_{\mathbf{x}} V_{\boldsymbol{\theta}}(\mathbf{x}, t)$, the actor optimization step becomes redundant and the VM algorithm emerges. This substantially simplifies the algorithm and eliminates the need for fine-tuning the base model. Moreover, VM requires fewer hyperparameters to achieve optimal performance, as shown in Appendix E.1.

6 Results

We now evaluate **V**alue **M**atching (VM), aiming to showcase four primary insights: (i) we verify that VM recovers the correct tilted distribution in an illustrative environment; (ii) we demonstrate the scalability of VM to high-dimensional image and molecular domains; (iii) we show that VM is more sample efficient and expressive than CG; and (iv) we find that VM can reduce resource requirements by over 95% compared to the fine-tuning method CT-PPO, while achieving comparable performance.

To ensure fairness, we limit the number of sampled trajectories to 128K in all direct comparisons. For CT-PPO, we conduct an extensive hyperparameter search (Appendix E.1) and adopt the value network from (Zhao et al., 2025) that defines the value function as a convex combination between $r \circ \hat{\mathbf{x}}_{\boldsymbol{\theta}}$, where $\hat{\mathbf{x}}_{\boldsymbol{\theta}}$ predicts $\mathbf{x}_1$ from $\mathbf{x}_t$, and a residual network $F_{\boldsymbol{\phi}}$. For $V_{\boldsymbol{\theta}}$ in VM and $F_{\boldsymbol{\phi}}$ in CT-PPO, we use a 1.8M-parameter CNN for images and a 2.5M-parameter GNN for molecules. Additional training details in Appendix B.

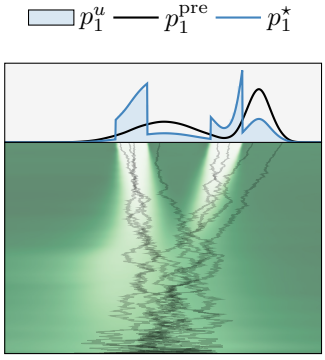


Figure 4: 1D toy experiment.

VM recovers the tilted distribution with non-differentiable rewards. To confirm that VM optimizes the intended objective, we test it in a simple one-dimensional setting, where the base distribution is a Gaussian mixture and the reward is binary and non-differentiable. As shown in Figure 4, VM successfully converges to the tilted distribution, the optimal solution. Moreover, it reveals the value function at intermediate timesteps. Consistent with the theory, the diffusion introduces an attenuating effect: values blur out and move to the origin as $t \to 0$. The blurring effect intuitively explains why the value function is differentiable for timesteps $t < 1$, even when r is not. Next, we consider high-dimensional domains.

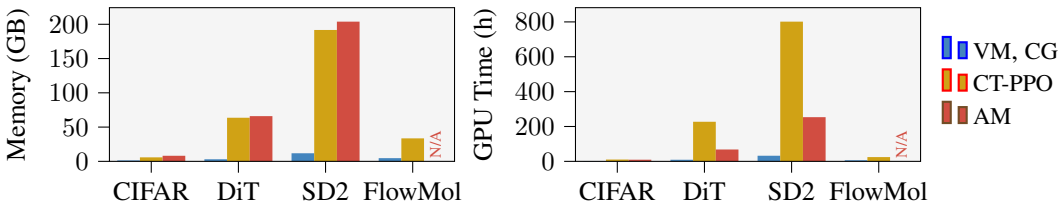


Figure 5: Resource requirements for adaptation methods across base models (16-bit, A100 GPUs). We do not show requirements for AM on FlowMol, because molecular rewards are inherently non-differentiable. LEFT: Memory requirements in GB. RIGHT: Training wall-clock time in hours.

VM is remarkably resource-efficient. As shown in Figure 5, VM demonstrates exceptional computational efficiency in comparison to fine-tuning methods. It requires less than 12 GB of memory across all evaluated models, whereas CT-PPO and AM demand up to 250 GB for SD2; a reduction of over 95%. The time requirements show similar advantages: VM completes training in under 35 hours for all models, while CT-PPO requires up to 800 GPU-hours for SD2. This efficiency gap widens with model scale: while the resource cost for fine-tuning methods grows substantially from CIFAR to SD2, VM maintains a consistently low overhead. These results establish that VM can be orders of magnitude more efficient than fine-tuning alternatives. Next, we show that small value networks are sufficient for effective reward adaptation and give comparable performance to fine-tuning methods.

VM effectively adapts large-scale image generation models. To demonstrate the general efficacy of VM, we apply it to the Diffusion Transformer (DiT) (Peebles & Xie, 2023) trained on the 256×256 ImageNet dataset (Deng et al., 2009), and the text-to-image model Stable Diffusion 2 (SD2) (Rombach et al., 2022). For training prompts, we randomly selected 40K captions from the LVIS dataset (Schuhmann & Bevan, 2023). The reward functions are compression and incompression, which correspond to minimizing and maximizing the bits per pixel (BPP) of the sample's JPEG-compressed version, at quality level 85. By learning to exploit JPEG's frequency-based method, we find that VM generates less detailed, low-frequency images under the compression reward and high-frequency Moiré patterns under the incompression reward (see Appendix D). Further, quantitative evaluation of DiT over 10K samples reveals that VM achieves 0.6 ± 0.3 and 3.1 ± 1.1 BPP under the compression and incompression rewards. Next, we investigate the performance of VM on molecular generation.

Figure 6: Samples with same random seed. TOP: SD2 base model. BOTTOM: VM with compression reward.

VM can effectively adapt molecular generation models. In molecular design, we evaluate VM on the continuous FlowMol model (Dunn & Koes, 2024b), pre-trained on the GEOM-Drugs dataset (Axelrod & Gomez-Bombarelli, 2022). The reward function is the dipole moment, computed using `GFN2-xTB` (Bannwarth et al., 2019) following geometry relaxation with `GFN-FF` (Spicher & Grimme, 2020). Due to the discrete nature of molecules and the geometry relaxation step, this reward function is non-differentiable. To prevent reward exploitation, wherein FlowMol frequently generates fragmented molecules, these outputs are assigned a zero reward. We find that optimizing this reward increases the frequency of heteroatoms and halogens, causing a 5-fold rise in highly electronegative fluorine (Figure 24). Moreover, over 10K samples, VM increases the average dipole moment to 7.5 ± 3.8 Debye from the base model's 6.4 ± 3.5 Debye, while simultaneously reducing the fragmentation rate from 31% to 28%. From this, we conclude that VM can successfully optimize for the target property without resorting to reward hacking. In the remainder of this section, we compare VM against CG and CT-PPO.

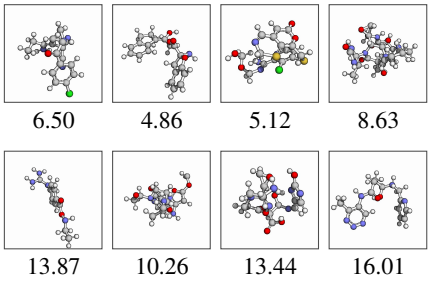


Figure 7: Samples with same random seed. TOP: FlowMol base model. BOTTOM: VM with dipole moment reward.

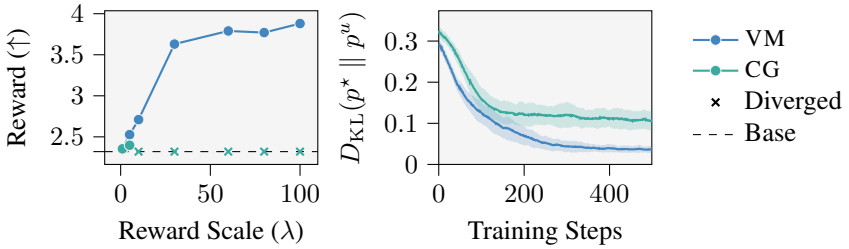


Figure 8: Comparison between VM and CG in large- and small-scale settings. LEFT: On the CIFAR image model with aesthetic reward, CG is unstable under moderate reward scaling. RIGHT: In a simple 1D environment, VM converges significantly faster than CG to the tilted distribution in terms of KL.

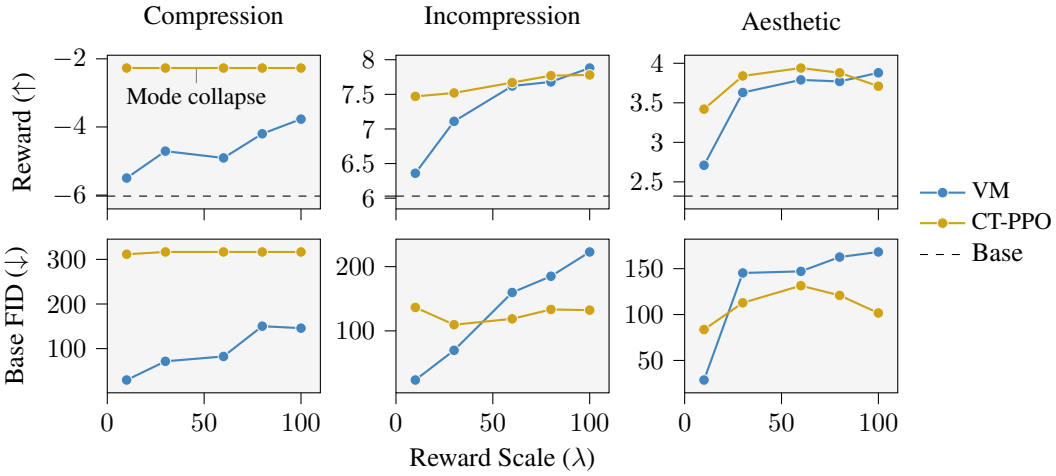


Figure 9: Comparison between VM and CT-PPO across various reward functions and scalings. VM demonstrates performance comparable to CT-PPO but with more predictable and stable behavior, shown by a consistent reward improvement as λ increases. In contrast, CT-PPO suffers from mode collapse on the compression task. TOP: Mean reward. BOTTOM: FID to base model.

VM is more sample efficient and expressive than classifier guidance. In this comparison, we demonstrate two key advantages of VM over CG: higher reward expressivity and greater sample efficiency. To evaluate the first, we use a 32×32 CIFAR-10 base model and the LAION aesthetics reward (Schuhmann, 2022). As shown in Figure 8 (left), the experiment reveals that CG becomes unstable at moderate reward scales ($\lambda \geq 10$), a significant practical limitation given that meaningful optimization often requires higher λ values. In contrast, VM maintains stable training and leads to much higher rewards. We then evaluate sample efficiency in a one-dimensional environment by tracking the KL divergence to the optimal distribution during training. The results (Figure 8, right) show VM consistently converging to superior optima, indicating a more effective use of training samples. The combination of enhanced stability and improved sample efficiency makes VM a more robust and practical alternative to CG for reward adaptation tasks, especially in resource-constrained settings.

VM demonstrates superior stability and controllability relative to CT-PPO while maintaining comparable performance. In the comparison with the fine-tuning method CT-PPO, we use the same base model as in the previous experiment across three reward functions: compression, incompression, and LAION aesthetics. The results, shown in Figure 9, show that VM is more robust and practical. While CT-PPO suffers from catastrophic mode collapse on the compression task (producing only all-white and all-black samples), VM's performance improves consistently as λ increases. On the other two rewards, their performance is comparable. However, achieving this with CT-PPO requires an extensive hyperparameter search (Appendix E.1), a significant practical drawback from which VM does not suffer. Finally, VM exhibits more controllability through the reward scaling, where it maintains the expected behavior of deviating further from the base model, measured in FID (Heusel et al., 2017) to the base model, in order to increase rewards. Taken together, VM represents a more stable, efficient, and reliably controllable algorithm for reward adaptation.

7 Related Work

Reward-Guided Flow Fine-Tuning for Generative Optimization. Recent work has explored fine-tuning flow and diffusion models for objectives beyond likelihood estimation, with leading approaches formulating the problem through RL and SOC frameworks. In this view, the generation process is a sequential decision problem where a policy is learned to steer the model toward desirable outcomes. An early approach, DDPO (Black et al., 2023) applies a policy gradient method to directly optimize for arbitrary rewards but often suffers from "reward collapse", where it overfits to a few high-reward samples at the cost of diversity. To counter this, DPOK (Fan et al., 2023) incorporated KL regularization to preserve diversity, though the KL term was approximated by an upper bound. A key insight was that the KL divergence can be computed with a quadratic running cost, enabling a control-theoretic interpretation. Leveraging this insight, SOCM (Domingo-Enrich et al., 2024) casts the control problem as an importance-weighted regression task. Further advancing this line, Adjoint Matching (Domingo-Enrich et al., 2025) resolved a critical value function bias in earlier methods, enabling provably unbiased reward adaptation. In an effort to address limitations of discretization, Zhao et al. (2025) introduced a continuous-time RL framework. Recent work also renders it possible to maximize rewards while preserving information from p^{pre} more generally than KL, as well as enabling risk-averse and risk-sensitive reward optimization (De Santi et al., 2025a). Our work advances this research line by introducing an algorithm that preserves the online nature of control-theoretic schemes, while lowering the memory requirements significantly, thereby easing its practical adoption.

Classifier(-Free) Guidance. A widely used alternative to fine-tuning is to steer the generative process at inference time. Classifier guidance (Dhariwal & Nichol, 2021) leverages the gradients of a separately trained classifier to push the sampling trajectory toward samples that exhibit desired attributes. To eliminate the need for an external model, classifier-free guidance (Ho & Salimans, 2022) modifies the training of the generative model itself to learn both a conditional and an unconditional distribution. At inference, the model is guided by amplifying the difference between the two, effectively steering generations toward the desired condition. These guidance mechanisms are foundational in diffusion model research, and improving upon them has become an active field of study (*e.g.*, Karras et al., 2024; Sadat et al., 2024; 2025; Rajabi et al., 2025). In this work, we establish a connection to these methods by showing that VM can be viewed as an online generalization of classifier guidance.

Inference-Time Schemes Beyond Guidance. Another family of methods performs reward optimization at inference-time through local, step-wise decisions. Many of these approaches can be understood as approximating an optimal denoising process by leveraging the pre-trained model as a look-ahead function to predict future rewards (Uehara et al., 2025). For instance, at each denoising step, methods like SVDD (Li et al., 2024a) and SCG (Huang et al., 2024) evaluate multiple candidate states and select the next state based on these predictions, employing strategies such as resampling or greedy selection. In contrast to such local methods, OC-Flow (Wang et al., 2025) adopts a global perspective, optimizing the entire trajectory at once by framing the task as an optimal control problem. The downside of inference-time schemes is a substantially increased wall-clock time for generation. Our method does not suffer from this by amortizing the optimization cost during training.

8 Conclusion and Outlook

We introduce **V**alue **M**atching (VM), a scalable and efficient online method for adapting pre-trained flow models to arbitrary reward functions. Drawing from fundamental insights in optimal control theory, VM learns the value function, which yields several key advantages. First, the resource requirements are controllable by a flexible choice of the value network architecture. Second, by learning the value function, VM naturally handles non-differentiable rewards, a crucial capability for black-box optimization problems.

Our experiments on image and molecular generation tasks demonstrate these benefits empirically. VM reduces memory and compute requirements by up to 95% compared to fine-tuning methods while achieving comparable performance. Furthermore, VM shows higher reward expressivity and proves more sample efficient than classifier guidance. By providing a theoretically grounded and practical framework for reward-guided adaptation, VM opens up promising opportunities for future research, such as applying VM to more complex, real-world problems.

Reproducibility Statement

We provide comprehensive details to ensure the reproducibility of our work. For all algorithms introduced, pseudocode is included, and benchmarks are performed against existing, publicly documented methods. In Appendix B.1, we detail the value network architectures used in this work. Further, in Appendix B.3, we show how to compute the integrals in practice. Moreover, in Appendix C, we give descriptions of the evaluation metrics, including how they are computed. Lastly, in Appendix E, we report the hyperparameters used for the CT-PPO experiments.

References

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023.

Howard Anton. *Calculus: A new horizon*. New York: Wiley, 1999.

Simon Axelrod and Rafael Gomez-Bombarelli. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):185, 2022.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Christoph Bannwarth, Sebastian Ehlert, and Stefan Grimme. Gfn2-xtb—an accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions. *Journal of chemical theory and computation*, 15 (3):1652–1671, 2019.

Richard E Bellman and Stuart E Dreyfus. *Applied dynamic programming*. Princeton university press, 2015.

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

Oskar Bolza. *Lectures on the Calculus of Variations*, volume 14. University of Chicago Press, 1904.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, pp. 02783649241273668, 2023.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024.

Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi S. Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. In *The Eleventh International Conference on Learning Representations*, 2023.

Riccardo De Santi, Marin Vlastelica, Ya-Ping Hsieh, Zebang Shen, Niao He, and Andreas Krause. Flow density control: Generative optimization beyond entropy-regularized fine-tuning. In *The Exploration in AI Today Workshop at ICML 2025*, 2025a.

Riccardo De Santi, Marin Vlastelica, Ya-Ping Hsieh, Zebang Shen, Niao He, and Andreas Krause. Provable maximum entropy manifold exploration via diffusion models. In *Forty-second International Conference on Machine Learning*, 2025b.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Carles Domingo-Enrich, Jiequn Han, Brandon Amos, Joan Bruna, and Ricky TQ Chen. Stochastic optimal control matching. *Advances in Neural Information Processing Systems*, 37:112459–112504, 2024.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ian Dunn and David Ryan Koes. Exploring discrete flow matching for 3d de novo molecule generation. *ArXiv*, pp. arXiv–2411, 2024a.

Ian Dunn and David Ryan Koes. Mixed continuous and categorical flow matching for 3d de novo molecule generation. *ArXiv*, pp. arXiv–2404, 2024b.

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.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Reinforcement learning for fine-tuning text-to-image diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS) 2023*. Neural Information Processing Systems Foundation, 2023.

Wendell H Fleming and Raymond W Rishel. *Deterministic and stochastic optimal control*, volume 1. Springer Science & Business Media, 2012.

Wendell H Fleming and H Mete Soner. *Controlled Markov processes and viscosity solutions*. Springer, 2006.

Stefano Forli, Ruth Huey, Michael E Pique, Michel F Sanner, David S Goodsell, and Arthur J Olson. Computational protein–ligand docking and virtual drug screening with the autodock suite. *Nature protocols*, 11(5):905–919, 2016.

Avner Friedman. Stochastic differential equations and applications. In *Stochastic differential equations*, pp. 75–148. Springer, 1975.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Yatharth Gupta, Vishnu V Jaddipal, Harish Prabhala, Sayak Paul, and Patrick Von Platen. Progressive knowledge distillation of stable diffusion xl using layer level loss. *arXiv preprint arXiv:2401.02677*, 2024.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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

Zijing Hu, Fengda Zhang, Long Chen, Kun Kuang, Jiahui Li, Kaifeng Gao, Jun Xiao, Xin Wang, and Wenwu Zhu. Towards better alignment: Training diffusion models with reinforcement learning against sparse rewards. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 23604–23614, 2025.

Yujia Huang, Adishree Ghatare, Yuanzhe Liu, Ziniu Hu, Qinsheng Zhang, Chandramouli Shama Sastry, Siddharth Gururani, Sageev Oore, and Yisong Yue. Symbolic music generation with non-differentiable rule guided diffusion. In *Forty-first International Conference on Machine Learning*, 2024.

James P Hughes, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011.

Mark Kac. On distributions of certain wiener functionals. *Transactions of the American Mathematical Society*, 65(1):1–13, 1949.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. *Advances in Neural Information Processing Systems*, 37:52996–53021, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*. Springer Berlin, Heidelberg, 1992.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical report*, 2009.

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, et al. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024a.

Zihao Li, Hui Yuan, Kaixuan Huang, Chengzhuo Ni, Yinyu Ye, Minshuo Chen, and Mengdi Wang. Diffusion model for data-driven black-box optimization. *CoRR*, 2024b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

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

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. Interacting particle solutions of fokker–planck equations through gradient–log–density estimation. *Entropy*, 22(8):802, 2020.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

K. Pandey, F. Sofian, F. Draxler, T. Karaletsos, and S. Mandt. Variational control for guidance in diffusion models. In *International Conference on Machine Learning (ICML)*, 2025.

Amey P Pasarkar and Adji Bousso Dieng. Cousins of the vendi score: A family of similarity-based diversity metrics for science and machine learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3808–3816. PMLR, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024.

Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. Routledge, 1962.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Javad Rajabi, Soroush Mehraban, Seyedmorteza Sadat, and Babak Taati. Token perturbation guidance for diffusion models. *arXiv preprint arXiv:2506.10036*, 2025.

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*, pp. 10684–10695, 2022.

Seyedmorteza Sadat, Otmar Hilliges, and Romann M Weber. Eliminating oversaturation and artifacts of high guidance scales in diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2024.

Seyedmorteza Sadat, Manuel Kansy, Otmar Hilliges, and Romann M. Weber. No training, no problem: Rethinking classifier-free guidance for diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Christoph Schuhmann. Laion aesthetics predictor v1, 2022. URL https://github.com/LAION-AI/aesthetic-predictor.

Christoph Schuhmann and Peter Bevan. Gpt4vision captions for livis, 2023. URL https://huggingface.co/datasets/laion/220k-GPT4Vision-captions-from-LIVIS.

David S Sholl and Janice A Steckel. *Density functional theory: a practical introduction*. John Wiley & Sons, 2009.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Sebastian Spicher and Stefan Grimme. Robust atomistic modeling of materials, organometallic, and biochemical systems. *Angewandte Chemie International Edition*, 59(36):15665–15673, 2020.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Wenpin Tang. Fine-tuning of diffusion models via stochastic control: entropy regularization and beyond. *arXiv preprint arXiv:2403.06279*, 2024.

Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024.

Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review. *arXiv preprint arXiv:2501.09685*, 2025.

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.

Luran Wang, Chaoran Cheng, Yizhen Liao, Yanru Qu, and Ge Liu. Training free guided flow-matching with optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

Richard Zhang. Making convolutional networks shift-invariant again. In *International conference on machine learning*, pp. 7324–7334. PMLR, 2019.

Hanyang Zhao, Haoxian Chen, Ji Zhang, David Yao, and Wenpin Tang. Score as action: Fine tuning diffusion generative models by continuous-time reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

APPENDICES

CONTENTS

A PROOFS

A.1 ASSUMPTIONS

Assumption 1. $a(t) \triangleq \sigma(t)\sigma^\intercal(t)$ is uniformly elliptic.

Assumption 2. The base drift $b : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is Lipschitz continuous in $\mathbf{x}$ and continuous in t.

Assumption 3. The norm of the base drift $\|b\|$ is bounded.

Assumption 4. The reward function $r : \mathbb{R}^d \to \mathbb{R}$ is bounded.

A.2 USEFUL LEMMAS

Lemma 1 (Application of the Feynman-Kac formula (Kac, 1949))**.** *Let V be the value function defined in Equation (5), then:*

$$V(\mathbf{x}, t) = -\log \mathbb{E}_{p^{\text{pre}}}[\exp(\lambda r(\mathbf{x}_1)) \mid \mathbf{x}_t = \mathbf{x}]. \tag{15}$$

Lemma 2 (Friedman (1975); Chapter 6, Theorem 4.5)**.** *Under Assumptions 1 to 3, the transition density $p_{s|t}(\mathbf{y} \mid \mathbf{x})$ of the uncontrolled SDE satisfies the following upper bound on its norm for $0 \le t < s \le 1$:*

$$\|\nabla_{\mathbf{x}} p_{s|t}(\mathbf{y} \mid \mathbf{x})\| \le C(s-t)^{-\frac{d+1}{2}} \exp\left(-c\frac{\|\mathbf{y}-\mathbf{x}\|^2}{s-t}\right), \tag{16}$$

where $C, c > 0$ are constants and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. Further, $\nabla_{\mathbf{x}} p_{s|t}(\mathbf{y} \mid \mathbf{x})$, $\nabla^2_{\mathbf{x}} p_{s|t}(\mathbf{y} \mid \mathbf{x})$, and $\partial_t p_{s|t}(\mathbf{y} \mid \mathbf{x})$ are uniformly continuous.

Lemma 3 (Fleming & Soner (2006), Chapter 5, Theorem 9.1)**.** *Consider the following Hamilton-Jacobi-Bellman equation:*

$$-\partial_t W(\mathbf{x}, t) + \mathcal{H}\big(\mathbf{x}, t, \nabla_{\mathbf{x}} W(\mathbf{x}, t), \nabla^2_{\mathbf{x}} W(\mathbf{x}, t)\big) = 0, \tag{17}$$

where in our case the Hamiltonian, $\mathcal{H}$, is:

$$\mathcal{H}(\mathbf{x}, t, \mathbf{p}, \mathbf{A}) = -\frac{1}{2}\operatorname{tr}(a(t)\mathbf{A}) - \langle b(\mathbf{x}, t), \mathbf{p}\rangle + \frac{1}{2}\|\sigma^\intercal(t)\mathbf{p}\|^2. \tag{18}$$

Assume Assumptions 1 to 4. Let W be a bounded viscosity subsolution and V be a bounded viscosity supersolution. Then,

$$\sup_{(\mathbf{x},t)\in\mathbb{R}^d\times[0,1]} (W(\mathbf{x}, t) - V(\mathbf{x}, t)) = \sup_{\mathbf{x}\in\mathbb{R}^d} (W(\mathbf{x}, 1) - V(\mathbf{x}, 1)). \tag{19}$$

Lemma 4 (Uniqueness)**.** *Under the assumptions of Lemma 3, the viscosity solution to Equation (17) is unique.*

Proof. We show this result by a comparison principle. Assume that Equation (17) has two viscosity solutions V_1 and V_2 with terminal condition $V_1(\mathbf{x}, 1) = V_2(\mathbf{x}, 1) = -\lambda r(\mathbf{x})$. Since they are viscosity solutions, they are also viscosity sub- and supersolutions. By their terminal condition, we know that:

$$\sup_{\mathbf{x}\in\mathbb{R}^d} (W(\mathbf{x}, 1) - V(\mathbf{x}, 1)) = \sup_{\mathbf{x}\in\mathbb{R}^d} -\lambda r(\mathbf{x}) + \lambda r(\mathbf{x}) = 0. \tag{20}$$

We will first show that $V_1 \le V_2$. Apply Lemma 3 with $W = V_1$ and $V = V_2$, then we have:

$$V_1(\mathbf{x}, t) - V_2(\mathbf{x}, t) \le \sup_{(\mathbf{x},t)} (V_1(\mathbf{x}, t) - V_2(\mathbf{x}, t)) = \sup_{\mathbf{x}\in\mathbb{R}^d} (V_1(\mathbf{x}, 1) - V_2(\mathbf{x}, 1)) = 0. \tag{21}$$

As such we have $V_1(\mathbf{x}, t) \le V_2(\mathbf{x}, t)$ for all $(\mathbf{x}, t)$.

Now we show that $V_2 \le V_1$. Again, apply Lemma 3 with $W = V_2$ and $V = V_1$, then we have:

$$V_2(\mathbf{x}, t) - V_1(\mathbf{x}, t) \le \sup_{(\mathbf{x},t)} (V_2(\mathbf{x}, t) - V_1(\mathbf{x}, t)) = \sup_{\mathbf{x}\in\mathbb{R}^d} (V_2(\mathbf{x}, 1) - V_1(\mathbf{x}, 1)) = 0. \tag{22}$$

Thus we also have $V_2(\mathbf{x}, t) \le V_1(\mathbf{x}, t)$ for all $(\mathbf{x}, t)$.

In conclusion, we have $V_1 - V_2 = 0$, meaning that they are equal. □

Putting it all together, we have that the following HJB equation has a unique solution:

$$\begin{gathered}\partial_t W(\mathbf{x}, t) + \frac{1}{2}\operatorname{tr}\big(a(t)\nabla^2_{\mathbf{x}} W(\mathbf{x}, t)\big) + \langle b(\mathbf{x}, t), \nabla_{\mathbf{x}} W\rangle - \frac{1}{2}\|\sigma^\intercal(t)\nabla_{\mathbf{x}} W\|^2 = 0, \\ W(\mathbf{x}, 1) = -\lambda r(\mathbf{x}).\end{gathered} \tag{23}$$

A.3 Proposition 1

Under Assumptions 1 to 4, the value function $V : \mathbb{R}^d \times [0,1] \to \mathbb{R}$ defined in Equation (5) is differentiable in $\mathbf{x}$ for $t < 1$

Proof. Let $t < 1$. Define $\psi(\mathbf{x}, t) \triangleq \mathbb{E}_{p^{\text{pre}}}[\exp(\lambda r(\mathbf{x}_1)) \mid \mathbf{x}_t = \mathbf{x}]$. Then from Lemma 1, we have $V(\mathbf{x}, t) = -\log \psi(\mathbf{x}, t)$. Thus, it suffices to show that (1) $\psi > 0$ and (2) ψ is differentiable in $\mathbf{x}$:

1. We assume that r is bounded, so $\exp(\lambda r(\mathbf{x})) > 0$. Hence, $\psi(\mathbf{x}, t) > 0$.
2. Writing ψ as an integral we have:

$$\psi(\mathbf{x}, t) = \int \exp(\lambda r(\mathbf{y})) p_{1|t}(\mathbf{y} \mid \mathbf{x}) \, \mathrm{d}\mathbf{y}. \tag{24}$$

Using that r is bounded such that $\exp(\lambda r) < M$ for some M and Lemma 2, we can show that the gradient norm of the integrand is dominated by an integrable function:

$$\|\nabla_{\mathbf{x}} \exp(\lambda r(\mathbf{y})) p_{1|t}(\mathbf{y} \mid \mathbf{x})\| = \exp(\lambda r(\mathbf{y})) \|\nabla_{\mathbf{x}} p_{1|t}(\mathbf{y} \mid \mathbf{x})\| \tag{25}$$

$$\leq MC(1-t)^{-\frac{d+1}{2}} \exp\left(-c \frac{\|\mathbf{y} - \mathbf{x}\|^2}{1-t}\right), \tag{26}$$

where $M, C, c, d > 0$ are constants. Thus, we can differentiate under the integral:

$$\nabla \psi(\mathbf{x}, t) = \int \exp(\lambda r(\mathbf{y})) \nabla_{\mathbf{x}} p_{1|t}(\mathbf{y} \mid \mathbf{x}) \, \mathrm{d}\mathbf{y}. \tag{27}$$

Further using Lemma 2, the transition density is continuously differentiable in $\mathbf{x}$. Thus, ψ is differentiable.

This fails for $t = 1$ since r might be non-differentiable and we have $V(\mathbf{x}, 1) = -\lambda r(\mathbf{x})$. In conclusion, by the chain rule:

$$\nabla V(\mathbf{x}, t) = -\frac{\nabla \psi(\mathbf{x}, t)}{\psi(\mathbf{x}, t)} \tag{28}$$

Therefore, V is continuously differentiable in $\mathbf{x}$ for $t < 1$. □

A.4 Proposition 2

The value function V is the unique critical point of $\mathbb{E}[\mathcal{L}_{\text{VM}}]$.

Proof. Let $W : \mathbb{R}^d \times [0,1] \to \mathbb{R}$ be a value function approximator and denote $\bar{W} = \texttt{stopgrad}(W)$ where the argument of `stopgrad` is treated as constant w.r.t. differentiation. In this proof, assume that any trajectory $\mathbf{x}_{[0,1]}$ is sampled from the current policy without gradients w.r.t. weights:

$$\mathrm{d}\mathbf{x}_t = \big(b(\mathbf{x}_t, t) - \sigma(t)\sigma^\intercal(t) \nabla \bar{W}(\mathbf{x}_t, t)\big) \, \mathrm{d}t + \sigma(t) \, \mathrm{d}B_t. \tag{29}$$

Critical point. In order to find the critical points, we will derive the functional derivative of $\mathbb{E}[\mathcal{L}_{\text{VM}}]$. Let $C : \mathbb{R}^d \times [0,1] \to \mathbb{R}$ be an arbitrary function, then:

$$\frac{\mathrm{d}}{\mathrm{d}\epsilon} \mathbb{E}[\mathcal{L}_{\text{VM}}(W + \epsilon C; \mathbf{x}_{[0,1]})] \Big|_{\epsilon=0} \tag{30}$$

$$= \frac{\mathrm{d}}{\mathrm{d}\epsilon} \mathbb{E}\left[\frac{1}{2} \int_0^1 w(t) \cdot \left|(W + \epsilon C)(\mathbf{x}_t, t) - \hat{J}\big(-\sigma^\intercal \nabla \bar{W}; \mathbf{x}_{[0,1]}, t\big)\right|^2 \mathrm{d}t\right] \Big|_{\epsilon=0} \tag{31}$$

$$= \mathbb{E}\left[\frac{1}{2} \int_0^1 w(t) \cdot \frac{\mathrm{d}}{\mathrm{d}\epsilon} \left|(W + \epsilon C)(\mathbf{x}_t, t) - \hat{J}\big(t - \sigma^\intercal \nabla \bar{W}; \mathbf{x}_{[0,1]}, t\big)\right|^2 \Big|_{\epsilon=0} \mathrm{d}t\right] \tag{32}$$

$$= \mathbb{E}\left[\int_0^1 C(\mathbf{x}_t, t) \cdot w(t) \cdot \Big(W(\mathbf{x}_t, t) - \hat{J}\big(-\sigma^\intercal \nabla \bar{W}; \mathbf{x}_{[0,1]}, t\big)\Big) \, \mathrm{d}t\right] \tag{33}$$

Using the tower property of expectation:

$$= \mathbb{E}\left[\int_0^1 C(\mathbf{x}_t, t) \cdot w(t) \cdot \left(W(\mathbf{x}_t, t) - \mathbb{E}\left[\hat{J}\left(-\sigma^\intercal \nabla \bar{W}; \mathbf{x}_{[0,1]}, t\right) \,\middle|\, \mathbf{x}_t\right]\right) \mathrm{d}t\right]. \tag{34}$$

So, the functional derivative is:

$$\frac{\delta}{\delta W}\mathbb{E}[\mathcal{L}_{\mathrm{VM}}(W)(\mathbf{x}, t)] = w(t) \cdot \left(W(\mathbf{x}, t) - \mathbb{E}\left[\hat{J}\left(-\sigma^\intercal \nabla \bar{W}; \mathbf{x}_{[0,1]}, t\right) \,\middle|\, \mathbf{x}_t = \mathbf{x}\right]\right). \tag{35}$$

Thus, any critical point (a point where the functional derivative equals zero) must satisfy:

$$W^\star(\mathbf{x}, t) = \mathbb{E}\left[\hat{J}\left(-\sigma^\intercal \nabla W^\star; \mathbf{x}_{[0,1]}, t\right) \,\middle|\, \mathbf{x}_t = \mathbf{x}\right] \tag{36}$$

$$= \mathbb{E}\left[\frac{1}{2}\int_t^1 \|\sigma^\intercal(s)\nabla W^\star(\mathbf{x}_s, s)\|^2 \,\mathrm{d}s - \lambda r(\mathbf{x}_1) \,\middle|\, \mathbf{x}_t = \mathbf{x}\right]. \tag{37}$$

By plugging Equation (9) into Equation (5), we know that the value function can be written as:

$$V(\mathbf{x}, t) = J(u^\star; \mathbf{x}, t) \tag{38}$$

$$= J(-\sigma^\intercal \nabla V; \mathbf{x}, t) \tag{39}$$

$$= \mathbb{E}\left[\frac{1}{2}\int_t^1 \|\sigma^\intercal(s)\nabla V(\mathbf{x}_s, s)\|^2 \,\mathrm{d}s - \lambda r(\mathbf{x}_1) \,\middle|\, \mathbf{x}_t = \mathbf{x}\right]. \tag{40}$$

Therefore V is a critical point of $\mathbb{E}[\mathcal{L}_{\mathrm{VM}}]$.

Uniqueness. As shown, a critical point W must satisfy the fixed-point:

$$W(\mathbf{x}, t) = \mathbb{E}\left[\frac{1}{2}\int_t^1 \|\sigma^\intercal(s)\nabla W(\mathbf{x}_s, s)\|^2 \,\mathrm{d}s - \lambda r(\mathbf{x}_1) \,\middle|\, \mathbf{x}_t = \mathbf{x}\right], \tag{41}$$

where the expectation is over trajectories from the controlled SDE:

$$\mathrm{d}\mathbf{x}_s = (b(\mathbf{x}_s, s) - a(s)\nabla W(\mathbf{x}_s, s))\,\mathrm{d}s + \sigma(s)\,\mathrm{d}B_s, \quad \mathbf{x}_t = \mathbf{x}. \tag{42}$$

By the Feynman-Kac formula, W satisfies the following PDE:

$$\partial_t W + \langle b - a\nabla W, \nabla W\rangle + \frac{1}{2}\operatorname{tr}\left(a\nabla^2 W\right) + \frac{1}{2}\|\sigma^\intercal \nabla W\|^2 = 0, \quad W(\mathbf{x}, 1) = -\lambda r(\mathbf{x}). \tag{43}$$

Noticing that $\langle \nabla W, a\nabla W\rangle = \|\sigma^\intercal \nabla W\|^2$, the PDE simplifies to:

$$\partial_t W + \langle b, \nabla W\rangle + \frac{1}{2}\operatorname{tr}\left(a\nabla^2 W\right) - \frac{1}{2}\|\sigma^\intercal \nabla W\|^2 = 0, \quad W(\mathbf{x}, 1) = -\lambda r(\mathbf{x}). \tag{44}$$

Using Lemma 4, we know that this HJB equation has a unique solution. This concludes the proof of Proposition 2: V is the unique critical point of $\mathbb{E}[\mathcal{L}_{\mathrm{VM}}]$. □

A.5 Derivation of Classifier Guidance Value Function

In this setting, we have $r(\mathbf{x}) = \log p^{\mathrm{pre}}_{Y|1}(y \mid \mathbf{x})$ for some class label y and $\lambda = 1$. From Lemma 1, we have:

$$V(\mathbf{x}, t) = -\log \mathbb{E}_{p^{\mathrm{pre}}}[\exp(\lambda r(\mathbf{x}_1)) \mid \mathbf{x}_t = \mathbf{x}] \tag{45}$$

$$= -\log \mathbb{E}_{p^{\mathrm{pre}}}[p^{\mathrm{pre}}(y \mid \mathbf{x}_1) \mid \mathbf{x}_t = \mathbf{x}] \tag{46}$$

$$= -\log \int p^{\mathrm{pre}}_{1|t}(\mathbf{x}_1 \mid \mathbf{x}) p^{\mathrm{pre}}_{Y|1}(y \mid \mathbf{x}_1)\,\mathrm{d}\mathbf{x}_1 \tag{47}$$

We have $y \perp \mathbf{x}_t \mid \mathbf{x}_1$, so by the chain rule:

$$= -\log \int p^{\mathrm{pre}}_{1,Y|t}(\mathbf{x}_1, y \mid \mathbf{x})\,\mathrm{d}\mathbf{x}_1 \tag{48}$$

By marginalization:

$$= -\log p^{\mathrm{pre}}_{Y|t}(y \mid \mathbf{x}). \tag{49}$$

This concludes the derivation of the statement.

B Experimental Details

All VM experiments employ the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 1×10^{-4}, a batch size of 128, and 100 SDE discretization steps. Unless otherwise specified, image experiments utilize a 1.8M-parameter convolutional neural network (CNN) and molecular experiments employ a 2.5M-parameter graph neural network (GNN) to parameterize the value function approximator $V_{\boldsymbol{\theta}}$. Additionally, to normalize the CG loss function (as we do for VM by adding the $1/\lambda^2$ term to $w(t)$), we normalize $\mathcal{L}_{\mathrm{CG}}$ by dividing it by $\exp(2\lambda)$.

B.1 Value Network Architectures

Convolutional Neural Network. We employ a standard CNN architecture consisting of an input convolution, three downsampling stages, an adaptive average pool, and a final linear head. Timesteps are embedded using a sinusoidal timestep embedder (Vaswani et al., 2017). Each downsampling stage comprises two layers with the following structure: convolution with a 3×3 kernel $\rightarrow$ group normalization (Wu & He, 2018) $\rightarrow$ FiLM (Perez et al., 2018) to incorporate timestep information $\rightarrow$ sigmoid linear unit (Hendrycks & Gimpel, 2016) activation function. Finally, the input is added residually and the result is downsampled using blur pool (Zhang, 2019). The convolutional layers in the downsampling stages use a base hidden dimensionality of 64, which doubles at each stage.

Graph Neural Network. The GNN architecture follows a similar design to the CNN architecture (excluding downsampling), where we replace the input convolution with a linear layer, convolutions with graph convolutions (Kipf & Welling, 2017), and group normalization with layer normalization (Ba et al., 2016). We utilize all node information available from the FlowMol model: atom position, atom type, and formal charge. We also incorporate edge data by linearly transforming the edge features and adding the mean of all incoming edge features to the node features after the input linear layer. Each block uses a hidden dimensionality of 256 across 6 stages.

B.2 Efficiently Computing Rewards

To efficiently compute reward functions for latent diffusion models, we decode samples individually. This approach significantly reduces VRAM requirements, as decoded samples are typically very large. We find that this strategy does not result in substantially increased wall-clock time.

B.3 Efficiently Computing the Cost Functional Estimate

We discretize the time horizon into T evenly spaced points. On this discretization, we perform the Euler-Maruyama method for sampling. Thus, at each step, the gradient $\nabla_{\mathbf{x}} V$ is computed. Based on this gradient, we compute the running cost at every step:

$$L_t = \frac{1}{2} \|\sigma^{\intercal}(t) \nabla_{\mathbf{x}} V(\mathbf{x}_t, t)\|^2. \tag{50}$$

At the final time step, the reward $R = r(\mathbf{x}_1)$ is received. The estimated cost functional $\hat{J}_t$ is then computed by summing the running costs from time t onward and subtracting the scaled terminal reward:

$$\hat{J}_t = \frac{1}{T} \sum_{\tau=t}^{T} L_{\tau/T} - \lambda R. \tag{51}$$

In total, computing the cost functional estimate involves computing T d-dimensional norms and adding T scalars using a reverse cumulative sum.

B.4 Weighting Functions

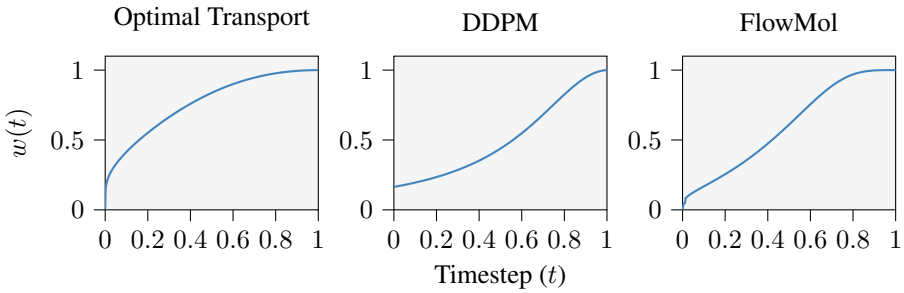


Figure 10: Weighting function $w(t)$ under various (α_t, β_t)-schedules, $\lambda = 1$, and the memoryless noise schedule.

Figure 10 displays the weighting function defined in Equation (14) for the schedules of models considered in this work. As can be seen, it down-weights earlier timesteps, which intuitively have the highest variance.

C Evaluation Metrics

Throughout this work, we employ three key metrics to assess performance: average reward, Fréchet inception distance (FID) (Heusel et al., 2017), and Vendi diversity (Friedman & Dieng, 2023; Pasarkar & Dieng, 2024). The average reward measures an algorithm's ability to exploit the reward function effectively, while FID relative to the base model captures the extent of deviation from the base model required to achieve this performance. Vendi diversity quantifies the variety within generated samples. Our objective is to achieve high reward and diversity while maintaining low FID. However, an inherent trade-off exists between reward optimization and sample diversity. In this section, we detail the computation of these metrics. For each metric, we assume access to a dataset of n samples.

C.1 Fréchet Inception Distance

FID (Heusel et al., 2017) is computed by first embedding each data point through a pre-trained Inception network (Szegedy et al., 2015) and extracting feature activations from the final layer. The Fréchet distance computes the means $(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)$ and covariance matrices $(\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ of both datasets, then calculates:

$$d_F(\mathbf{X}_1, \mathbf{X}_2) \triangleq \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2 + \operatorname{tr}\Big(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 - 2(\boldsymbol{\Sigma}_1\boldsymbol{\Sigma}_2)^{1/2}\Big). \tag{52}$$

Typically, $\mathbf{X}_1$ represents a reference dataset and $\mathbf{X}_2$ contains samples from the generative model. In this work, however, we set the reference dataset to samples from the base model and $\mathbf{X}_2$ to samples from the reward-adapted model. This provides a measure of how much the reward-adapted version has deviated from the base model.

C.2 Vendi Score

The Vendi score (Friedman & Dieng, 2023; Pasarkar & Dieng, 2024) is a diversity metric that requires only a positive semi-definite similarity function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with $k(\mathbf{x}, \mathbf{x}) = 1$ for all $\mathbf{x} \in \mathcal{X}$. It computes pairwise similarities between all samples and organizes them into a matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ where $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. The Vendi score is defined as the exponential of the entropy of the eigenvalues of $\mathbf{K}/n$:

$$\mathrm{VS}_k(\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}) \triangleq \exp\left(-\sum_{i=1}^{n} \lambda_i \log \lambda_i\right). \tag{53}$$

In this work, we employ the following similarity function:

$$k(\mathbf{x}, \mathbf{y}) = \langle \mathrm{clip}(\mathbf{x}), \mathrm{clip}(\mathbf{y}) \rangle, \tag{54}$$

where $\mathrm{clip}(\cdot)$ represents a CLIP image encoder (Radford et al., 2021) that produces normalized embeddings.

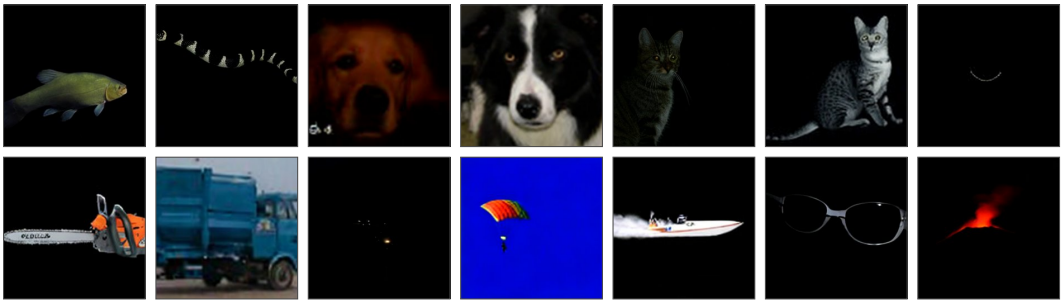
D SAMPLES AND TRAINING CURVES

The plotted costs reflect the deviation of the fine-tuned model from the base model; they correspond to the KL divergence between the base and controlled processes p^u, conditioned on the same initial state (Domingo-Enrich et al., 2025):

$$D_{\mathrm{KL}}\big(p^u(\mathbf{x}_{[0,1]} \mid \mathbf{x}_0) \,\big\|\, p^{\mathrm{pre}}(\mathbf{x}_{[0,1]} \mid \mathbf{x}_0)\big) = \mathbb{E}_{p^u}\left[\frac{1}{2}\int_0^1 \|u(\mathbf{x}_t, t)\|^2 \,\mathrm{d}t\right] \tag{55}$$

D.1 DIFFUSION TRANSFORMER

(a) Base model (mean size: 1.89 bits/pixel).



(b) Compression reward ($\lambda = 25$; mean size: 0.49 bits/pixel).

(c) Incompression reward ($\lambda = 25$; mean size: 3.31 bits/pixel).

Figure 11: Samples from Diffusion Transformer generated under the same random seed. Inference with CFG weight 2, whereas training was done without CFG.

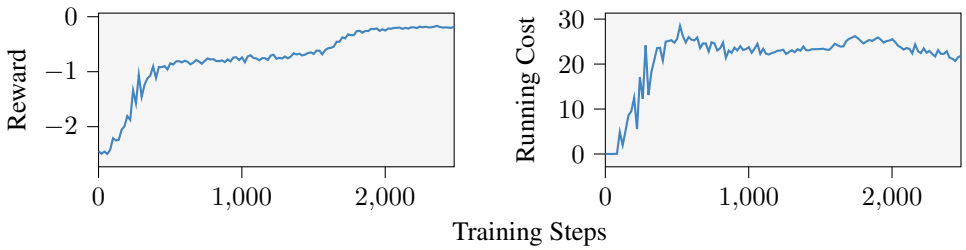


Figure 12: Training curves for VM on the DiT model with compression reward ($\lambda = 25$).

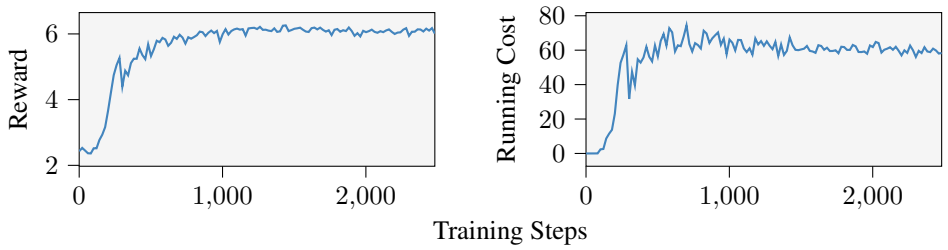


Figure 13: Training curves for VM on the DiT model with incompression reward ($\lambda = 25$).

D.2 STABLE DIFFUSION 2

Figure 14: Prompt: *A chocolate cake on a plate with decorative pattern, a fork beside it, giving off a sense of indulgence or celebration.* Reward: Compression ($\lambda = 2500$). CFG weight: 4.0.

Figure 15: Prompt: *Skiing scene with multiple individuals dressed in ski gear, engaging in skiing activities amidst snowy surroundings, suggesting a resort or slope ambiance.* Reward: Compression ($\lambda = 2500$). CFG weight: 4.0.

Figure 16: Prompt: *Dog seated on a red lounge chair in a cozy, sophisticated room with a painting, various decorations, and multiple lampshades while wearing a collar.* Reward: Compression ($\lambda = 2500$). CFG weight: 4.0.

Figure 17: Prompt: *A surfer in a wet suit performs a carving turn by a pier, on a beach break with no other surfers or boats present.* Reward: Incompression ($\lambda = 2500$). CFG weight: 4.0.

Figure 18: Prompt: *An orange cat with a blue hat featuring a logo, resting on a dark-colored background.* Reward: Incompression ($\lambda = 2500$). CFG weight: 4.0.

Figure 19: Prompt: *A freight train with cargo containers passes through a railroad crossing.* Reward: Incompression ($\lambda = 2500$). CFG weight: 4.0.

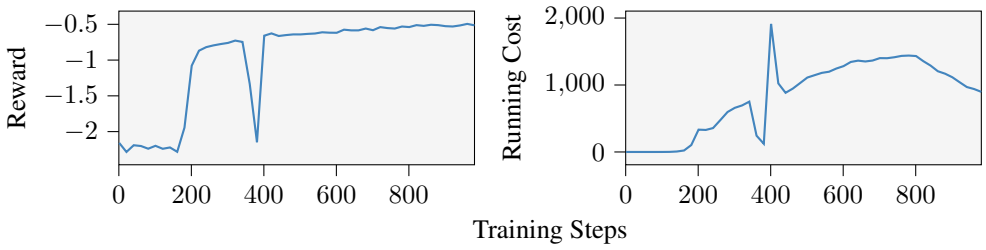


Figure 20: Training curves for VM on the SD2 model with compression reward ($\lambda = 2500$).

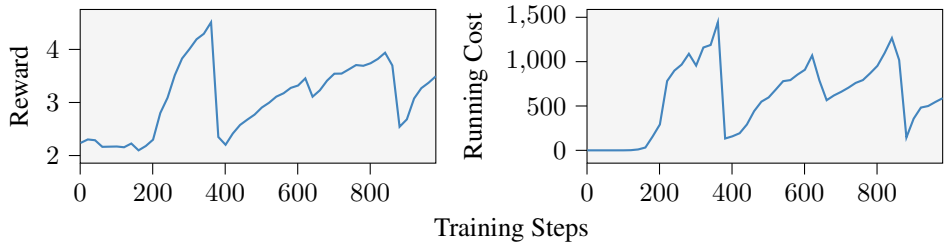
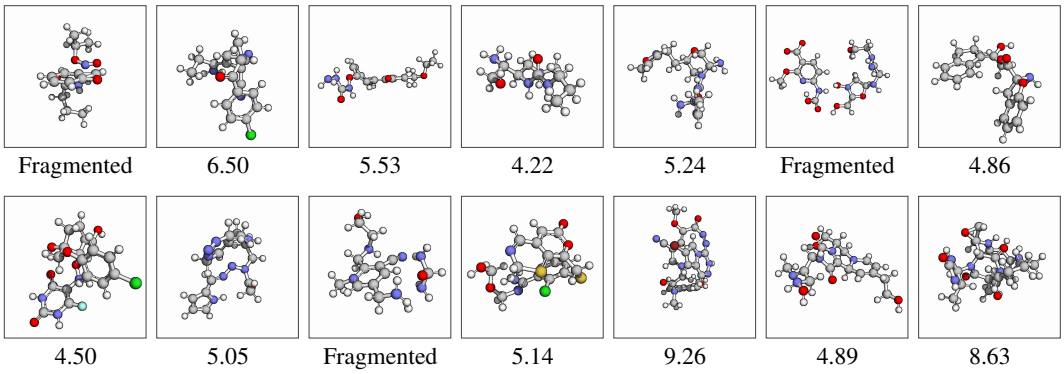
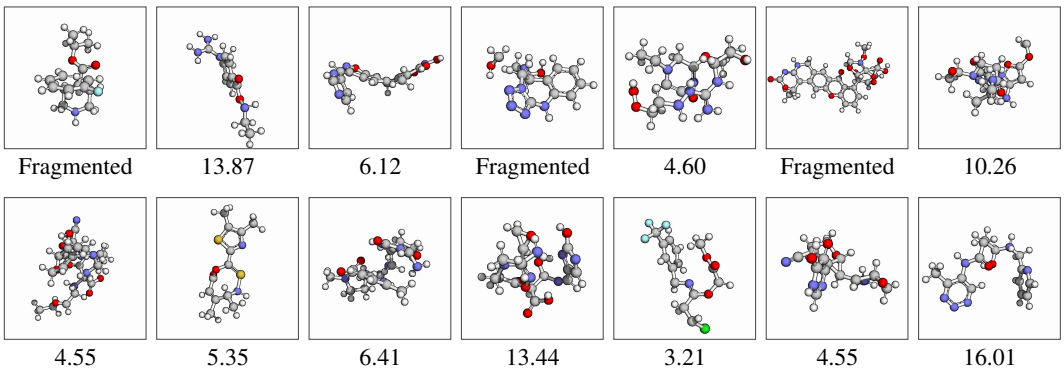


Figure 21: Training curves for VM on the SD2 model with incompression reward ($\lambda = 2500$).

D.3 GEOM-DRUGS



(a) Base model (mean reward: 4.56).



(b) VM on dipole moment reward ($\lambda = 100$; mean reward: 6.31).

Figure 22: Samples from the continuous GEOM-Drugs FlowMol base and VM model under the same random seed.

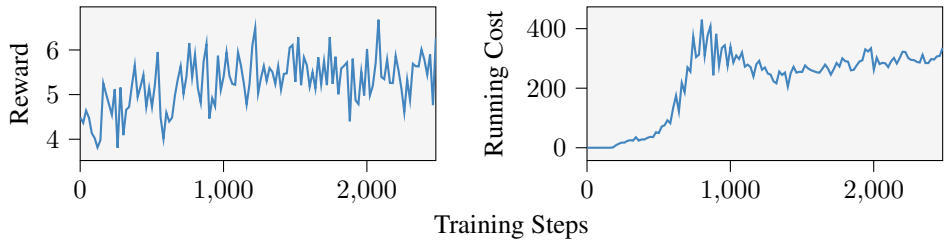


Figure 23: Training curves for VM on the continuous GEOM-Drugs FlowMol model with dipole moment reward ($\lambda = 100$).

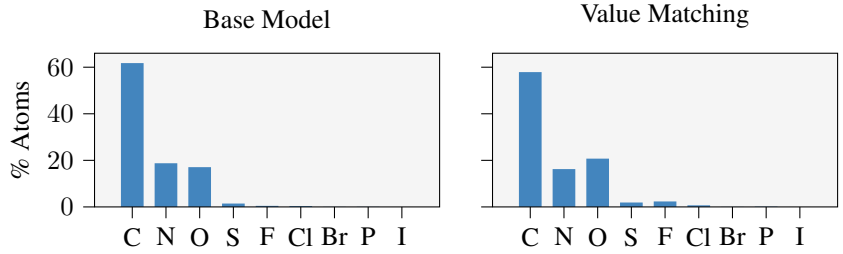


Figure 24: Atom type frequency distributions for molecules generated by FlowMol base model and VM with dipole moment reward ($\lambda = 100$). Results shown for 10K generated samples.

E PROXIMAL POLICY OPTIMIZATION ALGORITHM

Algorithm 2 Continuous-Time PPO (CT-PPO) algorithm.

Require: Initial policy parameters $\boldsymbol{\theta}$ (pre-trained), Initial value function parameters $\boldsymbol{\phi}$, Reward function $r : \mathbb{R}^d \to \mathbb{R}$, Number of iterations $N \in \mathbb{N}$, Training steps per iteration $K \in \mathbb{N}$, Trajectories per iteration $m \in \mathbb{N}$, Batch size $B \in \mathbb{N}$, Exploration level $\sigma \in \mathbb{R}_{>0}$, Scaling parameter $\eta \in \mathbb{R}_{>0}$, Clipping parameter $\epsilon \in \mathbb{R}_{>0}$.

for N iterations **do**
  Fix current policy $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}$.
  Sample m trajectories under the current policy:

$$\mathrm{d}\mathbf{x}_t = \left(\frac{\dot{\alpha}_t}{\alpha_t}\mathbf{x}_t + \sigma^2(t)\mathbf{a}_t\right)\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \quad \mathbf{a}_t = s_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_t, t).$$

  Compute returns:

$$R_t = r(\mathbf{x}_1) - \frac{1}{2\lambda}\int_t^1 \sigma^2(s)\|s_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_s, s) - s^{\mathrm{pre}}(\mathbf{x}_s, s)\|^2\,\mathrm{d}s.$$

  Initialize dataset $\mathcal{D}_V = \{(t, \mathbf{x}_t, R_t)\}_{t\in[0,1]}$ with all trajectories.
  repeat K **times**
    Sample $\mathcal{B} \subset \mathcal{D}_V$ with batch-size B.
    Compute loss:

$$\mathcal{L}(\boldsymbol{\phi}) = \frac{1}{B}\sum_{(t,\mathbf{x}_t,R_t)\in\mathcal{B}} (V_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - R_t)^2.$$

    Make an optimization step with $\nabla\mathcal{L}(\boldsymbol{\phi})$.
  end repeat
  Sample exploration noise $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ independently for each timestep and trajectory.
  Compute pseudo-samples and advantages:

$$\tilde{\mathbf{a}}_t = \mathbf{a}_t + \sigma\boldsymbol{\epsilon}_t$$
$$q_t = \frac{1}{\eta}\big(V_{\boldsymbol{\phi}}\big(\mathbf{x}_t + \eta\sigma^2(t)\boldsymbol{\epsilon}_t, t\big) - V_{\boldsymbol{\phi}}(\mathbf{x}_t, t)\big).$$

  Initialize dataset $\mathcal{D}_\pi = \{(t, \mathbf{x}_t, \tilde{\mathbf{a}}_t, q_t)\}_{t\in[0,1]}$ with all trajectories.
  repeat K **times**
    Sample $\mathcal{B} \subset \mathcal{D}_\pi$ with batch-size B.
    Compute likelihood ratio:

$$\rho_t^{\boldsymbol{\theta}} = \frac{\pi_{\boldsymbol{\theta}}(\tilde{\mathbf{a}}_t \mid \mathbf{x}_t, t)}{\pi_{\bar{\boldsymbol{\theta}}}(\tilde{\mathbf{a}}_t \mid \mathbf{x}_t, t)}, \quad \pi_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{x}, t) = \mathcal{N}(\mathbf{a}; s_{\boldsymbol{\theta}}(\mathbf{x}, t), \sigma\mathbf{I}).$$

    Compute loss:

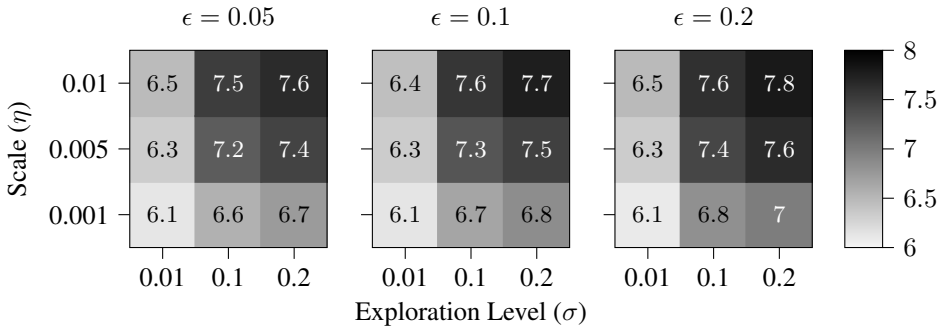
$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{B}\sum_{(t,\mathbf{x}_t,\tilde{\mathbf{a}}_t,q_t)\in\mathcal{B}} \min\big\{\rho_t^{\boldsymbol{\theta}} q_t, \mathrm{clip}\big(\rho_t^{\boldsymbol{\theta}}, 1-\epsilon, 1+\epsilon\big)q_t\big\}.$$

    Make an optimization step with $\nabla\mathcal{L}(\boldsymbol{\theta})$.
  end repeat
end for

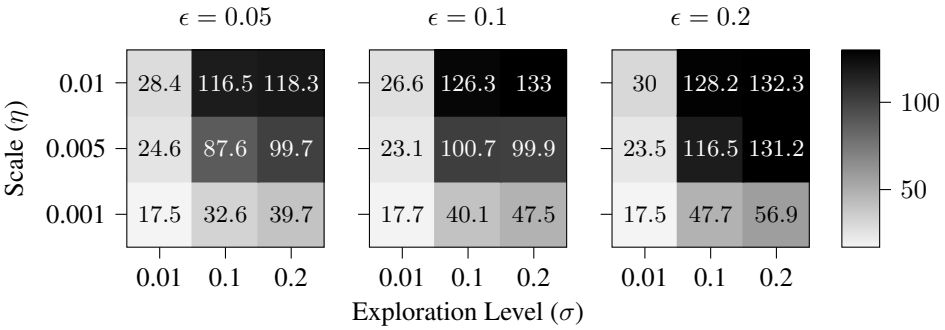
We set $K = \lceil m/B \rceil$ such that each point is seen once. For the actor and critic, we use learning rates 3×10^{-5} and 1×10^{-6}, respectively. For data collection, we standardize the number of trajectories and batch size using $m = 512$, $B = 128$, and $N = 250$. This configuration processes 128K trajectories during training, consistent with other methods in this work.

E.1 HYPERPARAMETER ABLATIONS

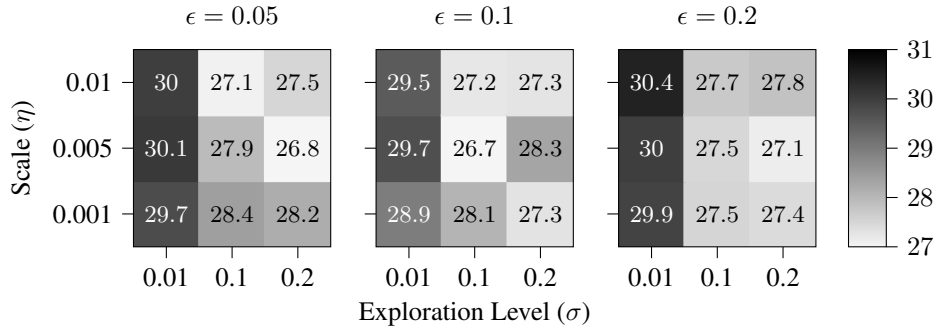
To ensure a fair comparison, we conduct comprehensive hyperparameter optimization for CT-PPO through an extensive grid search over the clipping parameter ϵ, exploration level σ, and scale η. Specifically, we conducted a grid search on $(\epsilon, \sigma, \eta) \in \{0.05, 0.1, 0.2\} \times \{0.01, 0.1, 0.2\} \times \{0.001, 0.005, 0.01\}$. While this additional tuning effort could be considered part of CT-PPO's computational overhead, it ensures optimal performance for our evaluation. In contrast, both VM and AM do not have any hyperparameter search cost.



(a) Average reward.



(b) FID to the base model.



(c) Vendi score (diversity).

Figure 25: Base model: CIFAR. Reward: Incompression ($\lambda = 100$).

F FLOW MODELS

For completeness, in this section we provide an overview of flow matching (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023; Liu et al., 2023) and how diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015; Song et al., 2020) can be viewed as an instance of it. Further, we show how to sample flow matching models through an SDE with equivalent time marginals. Lastly, we show how to sample flow matching or diffusion models from a common perspective through the score function.

F.1 FLOW MATCHING

Given a source distribution p and a target distribution q, the flow matching framework aims to solve the flow matching problem:

> *Find the velocity field* $v : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ *generating marginal distributions* p_t*, where* $p_0 = p$ *and* $p_1 = q$*.*

The flow matching framework solves this problem by the following steps:

1. Identify a known source distribution p and unknown target distribution q, of which we have finite samples.
2. Define a probability path p_t that interpolates $p_0 = p$ and $p_1 = q$.
3. Learn the velocity field by a neural network $v_{\boldsymbol{\theta}}$.
4. Sample the learned model by solving an ODE:

$$\mathrm{d}\mathbf{x}_t = v_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\,\mathrm{d}t. \tag{56}$$

In general, we could use a coupled data distribution $\mathbf{x}_0, \mathbf{x}_1 \sim p_{0,1}$, however, we will only be considering the case where $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Further, q is unknown, but we do assume that we have access to a dataset of samples from this distribution. *E.g.*, we might want to model a distribution of images and take the 32 ×32 CIFAR-10 dataset (Krizhevsky et al., 2009) as samples from this distribution.

Next, we need to define a probability path $\{p_t\}_{t\in[0,1]}$ that interpolates between $p_0 = p$ and $p_1 = q$. This is done by a conditional strategy, which involves defining $p_{t|1}$. We can then construct the marginal probability path by:

$$p_t(\mathbf{x}) = \int p_{t|1}(\mathbf{x} \mid \mathbf{x}_1) q(\mathbf{x}_1)\,\mathrm{d}\mathbf{x}_1. \tag{57}$$

We will consider an affine parameterization of the conditional probability path:

$$p_{t|1}(\mathbf{x} \mid \mathbf{x}_1) = \mathcal{N}\big(\mathbf{x}; \alpha_t \mathbf{x}_1, \beta_t^2 \mathbf{I}_d\big), \tag{58}$$

where $\alpha_t, \beta_t : [0,1] \to [0,1]$ are smooth functions satisfying $\alpha_0 = \beta_1 = 0$, $\alpha_1 = \beta_0 = 1$, and $\dot{\alpha}_t > 0 > \dot{\beta}_t$ for $t \in (0,1)$. (The dot-notation denotes the time-derivative.) We can sample from this distribution as follows:

$$\mathbf{x}_{t|1} = \alpha_t \mathbf{x}_1 + \beta_t \mathbf{x}_0, \quad \mathbf{x}_0 \sim p_0. \tag{59}$$

Commonly, the optimal transport schedule is used where $\alpha_t = t$ and $\beta_t = 1 - t$.

Differentiating w.r.t. t gives the associated marginal velocity field:

$$v(\mathbf{x}, t) = \mathbb{E}[\dot{\alpha}_t \mathbf{x}_1 + \dot{\beta}_t \mathbf{x}_0 \mid \mathbf{x}_t = \mathbf{x}]. \tag{60}$$

Thus, we can train using the flow matching loss:

$$\begin{gathered} \mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{t,\mathbf{x}_t}\big[\|v_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - v(\mathbf{x}_t, t)\|^2\big], \\ t \sim \mathcal{U}([0,1]), \mathbf{x}_t \sim p_t. \end{gathered} \tag{61}$$

However, this is (almost always) intractable, because we do not know the velocity field yet and we cannot sample p_t. In order to alleviate these issues, we can drastically simplify the loss by conditioning on the target sample $\mathbf{x}_1$:

$$\begin{gathered} \mathcal{L}_{\mathrm{CFM}}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{t,\mathbf{x}_1,\mathbf{x}_t}\big[\|v_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - v(\mathbf{x}_t, t \mid \mathbf{x}_1)\|^2\big], \\ t \sim \mathcal{U}([0,1]), \mathbf{x}_1 \sim p_1, \mathbf{x}_t \sim p_{t|1}(\cdot \mid \mathbf{x}_1), \end{gathered} \tag{62}$$

Algorithm 3 Flow Matching training.

Require: Untrained velocity model $v_{\boldsymbol{\theta}}$, Schedule $\alpha_t, \beta_t : [0,1] \to [0,1]$, Source distribution p, and target distribution q.

1: **while** not converged **do**
2: Sample $\mathbf{x}_0 \sim p$, $\mathbf{x}_1 \sim q$, and $t \sim \mathcal{U}([0,1])$.
3: Compute $\mathbf{x}_t = \alpha_t \mathbf{x}_1 + \beta_t \mathbf{x}_0$ and $\mathbf{v}_t = \dot{\alpha}_t \mathbf{x}_1 + \dot{\beta}_t \mathbf{x}_0$.
4: Compute the loss $\mathcal{L}_{\text{CFM}} = \|v_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \mathbf{v}_t\|^2$.
5: Do an optimization step with $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{CFM}}$.
6: **end while**
7: **return** $v_{\boldsymbol{\theta}}$

where the conditional velocity is

$$v(\mathbf{x}, t \mid \mathbf{x}_1) = \dot{\alpha}_t \mathbf{x}_1 + \dot{\beta}_t \mathbf{x}_0, \quad \mathbf{x}_t = \mathbf{x} \tag{63}$$

$$= \dot{\alpha}_t \mathbf{x}_1 + \frac{\dot{\beta}_t}{\beta_t}(\mathbf{x} - \alpha_t \mathbf{x}_1) \tag{64}$$

$$= \left(\dot{\alpha}_t - \frac{\alpha_t \dot{\beta}_t}{\beta_t}\right)\mathbf{x}_1 + \frac{\dot{\beta}_t}{\beta_t}\mathbf{x}. \tag{65}$$

Amazingly, these two loss functions have the same gradient w.r.t. the parameters (Lipman et al., 2023):

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{FM}} = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{CFM}}. \tag{66}$$

This justifies applying gradient-based optimization methods on the conditional loss, which is tractable, because it will lead to the same parameter updates. See Algorithm 3 for the training algorithm. Refer to (Lipman et al., 2024) for an in-depth treatment of flow matching models.

Lastly, instead of sampling from a deterministic ODE, we can also consider sampling from a family of SDEs:

$$\mathrm{d}\mathbf{x}_t = \left(v(\mathbf{x}_t, t) + \frac{\sigma^2(t)}{2\eta_t}(v(\mathbf{x}_t, t) - \kappa_t \mathbf{x}_t)\right)\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \tag{67}$$

where B_t is a Brownian motion, $\sigma : [0,1] \to \mathbb{R}^{d \times d}$ is an arbitrary state-independent diffusion coefficient, and

$$\eta_t \triangleq \beta_t\left(\frac{\dot{\alpha}_t}{\alpha_t}\beta_t - \dot{\beta}_t\right), \quad \kappa_t \triangleq \frac{\dot{\alpha}_t}{\alpha_t}. \tag{68}$$

It can be shown that the generative processes in Equation (56) and Equation (67) have equivalent time marginals (Maoutsa et al., 2020). In the memoryless noise schedule, we have $\sigma(t) = \sqrt{2\eta_t}$ (Domingo-Enrich et al., 2025).

F.2 Diffusion Models

Diffusion models take a (slightly) different perspective than flow matching. They view sampling as the reversal of a data destruction (or noising) process. For this, we must first define the noising process:

$$\mathbf{x}_{t+1} = \sqrt{\gamma_t}\mathbf{x}_t + \sqrt{1 - \gamma_t}\boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n), \tag{69}$$

where γ_t follows some schedule from 0 to T such that $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$.[1] As such, starting from $\mathbf{x}_0 = \mathbf{x} \sim q$, the data gets progressively more like Gaussian noise. Using Gaussian arithmetic, the above process can be computed in a closed form:

$$\mathbf{x}_t = \sqrt{\bar{\gamma}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\gamma}_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n), \tag{70}$$

where $\bar{\gamma}_t = \prod_{s=0}^{t-1} \gamma_s$. The denoising process from time T to 0 can be computed as follows:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\gamma_t}}\left(\mathbf{x}_t - \frac{1 - \gamma_t}{\sqrt{1 - \bar{\gamma}_t}}\boldsymbol{\epsilon}_t\right) + \sigma_t \mathbf{z}, \quad \mathbf{z} \in \mathcal{N}(\mathbf{0}, \mathbf{I}_d). \tag{71}$$

Here the only unknown is $\boldsymbol{\epsilon}_t$, so we will train a network to approximate it; see Algorithm 4

[1] Generally, $(\gamma_t, \bar{\gamma}_t)$ are denoted by $(\alpha_t, \bar{\alpha}_t)$. This notation is used here to avoid confusion with flow matching schedules.

Algorithm 4 Diffusion model training.

Require: Untrained epsilon model $\epsilon_{\boldsymbol{\theta}}$, Schedule $\{\gamma_t\}_{t=0}^{T}$, Target distribution q
1: **while** not converged **do**
2: Sample $\mathbf{x}_0 \sim q$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$ and $t \sim \mathcal{U}([T])$.
3: Compute $\mathbf{x}_t = \sqrt{\bar{\gamma}_t}\mathbf{x}_0 + \sqrt{1-\bar{\gamma}_t}\boldsymbol{\epsilon}$.
4: Compute the loss $\mathcal{L}_{\mathrm{DM}} = \|\epsilon_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \boldsymbol{\epsilon}\|^2$.
5: Do an optimization step with $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{DM}}$.
6: **end while**
7: **return** $\epsilon_{\boldsymbol{\theta}}$

F.3 Diffusion Models as an Instance of Flow Matching

We can sample a diffusion model with the DDIM schedule through the following SDE (Domingo-Enrich et al., 2025):

$$\mathrm{d}\mathbf{x}_t = \left(\frac{\dot{\bar{\gamma}}_t}{2\bar{\gamma}_t}\mathbf{x}_t - \left(\frac{\dot{\bar{\gamma}}_t}{2\bar{\gamma}_t} + \frac{\sigma^2(t)}{2}\right)\frac{\epsilon(\mathbf{x}_t, t)}{\sqrt{1-\bar{\gamma}_t}}\right)\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t. \tag{72}$$

In order to consolidate diffusion models and flow matching models into a common framework where $p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, we will be working with the score function:

$$s(\mathbf{x}, t) \triangleq \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \tag{73}$$

$$s(\mathbf{x}, t) = \frac{1}{\eta_t}(v(\mathbf{x}, t) - \kappa_t \mathbf{x}) \tag{74}$$

$$s(\mathbf{x}, t) = -\frac{\epsilon(\mathbf{x}, t)}{\sqrt{1-\bar{\gamma}_t}}. \tag{75}$$

We can now sample either a diffusion model or flow matching model by converting their parametrization to the score function and sampling the following SDE:

$$\mathrm{d}\mathbf{x}_t = \left(\kappa_t \mathbf{x}_t + \left(\frac{\sigma^2(t)}{2} + \eta_t\right)s(\mathbf{x}_t, t)\right)\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \tag{76}$$

In the case of diffusion models, we have

$$\alpha_t = \sqrt{\bar{\gamma}_t}, \quad \beta_t = \sqrt{1-\bar{\gamma}_t} \tag{77}$$

with associated time derivatives:

$$\dot{\alpha}_t = \frac{\dot{\bar{\gamma}}_t}{2\sqrt{\bar{\gamma}_t}}, \quad \dot{\beta}_t = -\frac{\dot{\bar{\gamma}}_t}{2\sqrt{1-\bar{\gamma}_t}}. \tag{78}$$

We will use the convention of flow matching models. Generally, in diffusion models, we have that time is discrete from 0–K and decreases when sampling. Thus, we have the following conversion between the two conventions:

$$\bar{\gamma}_t = \bar{\gamma}\lfloor K(1-t)\rfloor \tag{79}$$

$$\dot{\bar{\gamma}}_t = K \cdot (\bar{\gamma}\lfloor K(1-t) - 1\rfloor - \bar{\gamma}\lfloor K(1-t)\rfloor). \tag{80}$$

One can easily verify that this is equivalent to sampling from DDIM.