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

Cristian Perez Jensen
ETH Zürich
`cjense@ethz.ch`

Luca Schaufelberger[*]
ETH Zürich
NCCR Catalysis
`schaluca@ethz.ch`

Riccardo De Santi[*]
ETH Zürich
ETH AI Center
NCCR Catalysis
`rdesanti@ethz.ch`

Kjell Jorner
ETH Zürich
NCCR Catalysis
`kjorner@ethz.ch`

Andreas Krause
ETH Zürich
ETH AI Center
NCCR Catalysis
`krausea@ethz.ch`

Abstract

Adapting large-scale flow and diffusion models to downstream applications is essential for making them practically useful, but current fine-tuning methods are increasingly impractical due to memory demands that scale with model size. This raises the question whether alignment with downstream rewards can be achieved with resource requirements that are independent of model complexity. We propose **V**alue **M**atching (VM), a scalable method that sidesteps fine-tuning by training a lightweight value function to guide generation. VM learns the value function through Monte Carlo estimation, entirely decoupled from base model gradients, enabling efficient adaptation. Further, VM supports non-differentiable rewards and is more stable, sample-efficient, and expressive than classifier guidance. Experiments on image and molecular generation show that VM can successfully steer the pre-trained model density to high-reward regions while requiring only a small fraction of the memory used by current fine-tuning methods based on reinforcement learning or control theory techniques.

1 Introduction

Large-scale generative foundation models have recently made remarkable progress. Among them, flow [35, 1, 37] and diffusion models [28, 47, 48] stand out for their ability to generate high-fidelity samples across a wide range of domains, including images [28], chemistry [29], biology [9], and robotics [8]. For many applications (*e.g.*, controllable image editing and drug discovery), adapting such large pre-trained models to downstream rewards is essential. Existing approaches based on reinforcement learning (RL) [55, 6, 19, 30] and stochastic optimal control (SOC) [15, 51, 50, 14] rely on backpropagating

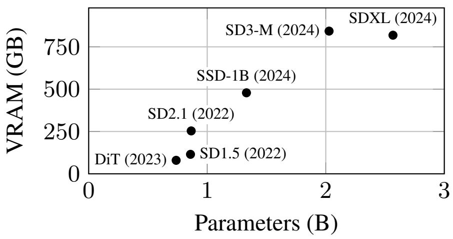


Figure 1: The recent trend toward more complex flow and diffusion generative models leads to prohibitively high fine-tuning memory (VRAM) requirements.

[*] Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Structured Probabilistic Inference & Generative Modeling.

through the full model. This makes them increasingly
memory-intensive as model sizes scale to billions of parameters, as illustrated in Fig. 1.

Despite their promising performances (*e.g.*, [15, 55]), existing RL- and SOC-based methods remain fundamentally limited by resource requirements that scale with model size. This raises a key question:

How can we leverage RL/SOC machinery to adapt flow and diffusion models to maximize non-differentiable rewards without scaling resource requirements with base model complexity?

We address this question by introducing a principled method, **V**alue **M**atching (VM), which avoids fine-tuning the base model and instead learns a lightweight value function providing reward-guided control. Crucially, this decouples adaptation from model size, enabling adaptation of large models.

Our Contributions:

- ***V**alue **M**atching* (VM), a theory-grounded method for reward adaptation of flow models; VM is an SOC approach independent of model complexity and supports non-differentiable rewards (Sec. 4).
- By adopting a control theoretic viewpoint, we show that VM can be interpreted as a generalization of classifier guidance with higher objective expressivity and practical sample complexity (Sec. 5).
- We perform experiments on image and molecular generation tasks with non-differentiable rewards, showing that VM increases the rewards of generated samples, while being more stable and sample-efficient than classifier guidance, and that small value networks suffice for adaptation (Sec. 6).

2 Background and Notation

Flow Models. Flow matching [35, 1, 37] and diffusion models [28, 47, 48] have emerged as leading approaches for generative modeling across various domains, including images [46, 41, 43, 24, 18], text [23], and molecular design [39, 29, 16, 17]. Typically, flow models are sampled through an ordinary differential equation (ODE) [35], however, we can also sample them via a stochastic differential equation (SDE) with equivalent time marginals [15]. In this work, we assume we can sample by:

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

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 coefficient, and B_t is a Brownian motion. We parameterize the above SDE as memoryless [15]:

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

with (α_t, β_t) defined through the marginal flow schedule [35] and $(\dot{\alpha}_t, \dot{\beta}_t)$ being time derivatives.

Stochastic Optimal Control. SOC [5, 20] deals with optimization problems over processes where the dynamics are modeled by an SDE. We will restrict ourselves to a quadratic cost control-affine Bolza problem [7] on a finite time horizon $[0, 1]$. In this case, we model the dynamics as:

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

Further, the cost functional J is defined as the total cost of a control $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ 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 finding the control u that minimizes J at every point $(\mathbf{x}, t)$:

$$u^\star \in \underset{u:\mathbb{R}^d\times[0,1]\to\mathbb{R}^d}{\arg\min}\; J(u; \mathbf{x}, t) \triangleq \mathbb{E}_{p^u}\left[\frac{1}{2}\int_t^1 \|u(\mathbf{x}_s, s)\|^2\,\mathrm{d}s + g(\mathbf{x}_1)\;\middle|\; \mathbf{x}_t = \mathbf{x}\right]. \tag{4}$$

From here, the value function is defined as the optimal value of the cost functional [20]:

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

A key property is that the value function can be defined through the pre-trained distribution [14]:

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

3 Problem Setting

We consider the problem of adapting a pre-trained flow or diffusion model p^{pre} so that it maximizes a reward function $r : \mathbb{R}^d \to \mathbb{R}$ in expectation, weighted by $\lambda \in \mathbb{R}_{\geq 0}$, and stays close to p^{pre} in terms of KL. Formally, we optimize over policies π with induced marginal distribution p^{π}:

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

The optimal solution $\pi^{\star}$ of Problem (7) induces the *tilted distribution* $p^{\star}(\mathbf{x}) \propto p^{\text{pre}}(\mathbf{x})\exp(\lambda r(\mathbf{x}))$ [15]. In flow models, this objective is equivalent to the control problem [15, 50]:

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

Existing fine-tuning methods (*e.g.*, [15, 55]) encounter scalability challenges, since updating the weights requires backpropagating through the base model, leading to memory costs that grow with model size. In the following section, we introduce an approach that addresses this issue.

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

Value **M**atching (VM) solves our objective in Eq. (7) by learning the value function and leveraging Pontryagin's maximum principle [44] to obtain the optimal control:

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

Proposition 1. *Under minimal regularity conditions (r needs to only be measurable and bounded), the value function is differentiable in $\mathbf{x}$ at $t < 1$.*

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

A key advantage is that the value function is learned with a lightweight network instead of fine-tuning the large base model, greatly reducing resource demands. VM trains this network by forming a Monte Carlo estimate $\hat{J}$ of $J(u; \mathbf{x}, t)$ under the current control $u = -\sigma^{\intercal}\nabla_{\mathbf{x}} V_{\boldsymbol{\theta}}$ and matching $V_{\boldsymbol{\theta}}$ to $\hat{J}$. We formalize this as Eq. (10) that uses $J(t; \mathbf{x}_{[0,1]}, u)$ as a single-sample Monte Carlo estimate of $J(u; \mathbf{x}, t)$, with trajectories sampled from the current control to ensure consistency with the expectation:

$$\begin{gathered}\mathcal{L}_{\text{VM}}(\boldsymbol{\theta}; \mathbf{x}_{[0,1]}) \triangleq \frac{1}{2}\int_0^1 w(t)\cdot|V_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - J(t; \mathbf{x}_{[0,1]}, -\sigma^{\intercal}\nabla_{\mathbf{x}} V_{\bar{\boldsymbol{\theta}}})|^2\,\mathrm{d}t, \\ J(t; \mathbf{x}_{[0,1]}, u) \triangleq \frac{1}{2}\int_t^1 \|u(\mathbf{x}_s, s)\|^2\,\mathrm{d}s - \lambda r(\mathbf{x}_1), \quad \bar{\boldsymbol{\theta}} = \texttt{stopgrad}(\boldsymbol{\theta}) \\ \mathrm{d}\mathbf{x}_t = (b(\mathbf{x}_t, t) - \sigma(t)\sigma^{\intercal}(t)\nabla_{\mathbf{x}} V_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_t, t))\,\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \quad \mathbf{x}_0 \sim p_0.\end{gathered} \tag{10}$$

Being on-policy, the algorithm trains on samples more closely aligned with inference, improving sample-efficiency. Also, it only requires zero-th order information of r, so differentiability is not needed. Finally, VM provides convergence guarantees in expectation to the true value function.

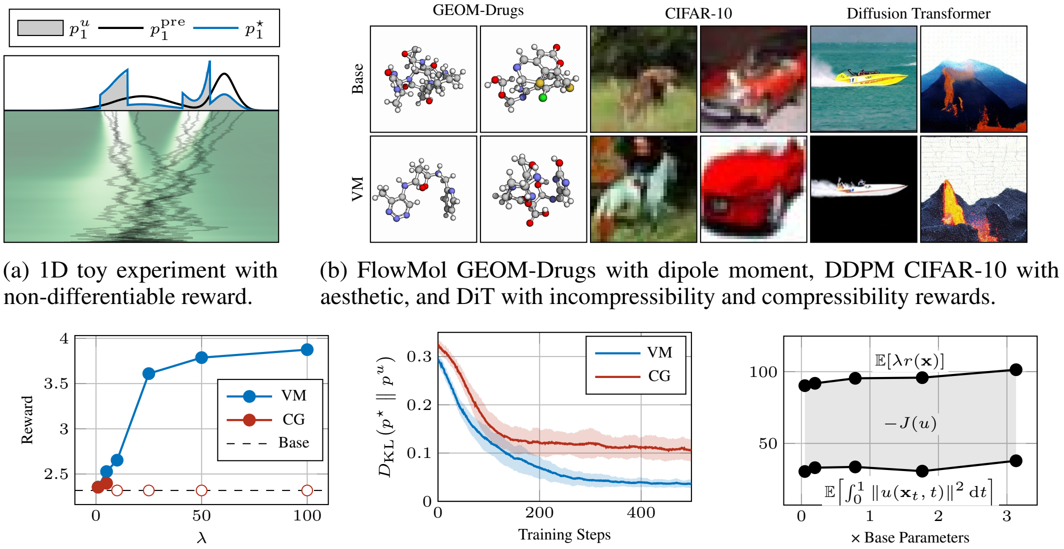
Proposition 2. *The value function V is the unique critical point of* $\mathbb{E}[\mathcal{L}_{\text{VM}}]$.

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

This result justifies gradient-based optimization of the value function approximator $V_{\boldsymbol{\theta}}$. In the next section, we will show how classifier guidance can be viewed as learning the value function.

5 Classifier Guidance From a Control Viewpoint

Classifier guidance [13] admits a control-theoretic interpretation with reward $r(\mathbf{x}) = \log p_{Y|1}(y \mid \mathbf{x})$ where y is a class label or prompt [40]. Using Eq. (6), the corresponding value function is $V(\mathbf{x}, t) =$



(a) 1D toy experiment with non-differentiable reward.

(b) FlowMol GEOM-Drugs with dipole moment, DDPM CIFAR-10 with aesthetic, and DiT with incompressibility and compressibility rewards.

(c) Reward scaling λ *vs.* reward using aesthetic reward on CIFAR-10.

(d) 1D example with KL divergence from policy to tilted distribution.

(e) Varying value network sizes on CIFAR-10 with aesthetic reward.

Figure 2: **(a)** Intermediate value function and marginals in 1D. **(b)** Promising samples from base model and VM on drug design and image generation, showing reward alignment and closeness to base model. **(c, d)** Comparison of VM with classifier guidance on large and small-scale experiments. **(e)** Larger value networks yield minimal gains despite higher cost, showing efficiency of small models.

$-\log p_{Y|t}(y \mid \mathbf{x})$ (derivation in Apx. C.5). This motivates a generalized loss based on Eq. (6):

$$\begin{aligned}\mathcal{L}_{\mathrm{CG}}(\boldsymbol{\theta}; \mathbf{x}_{[0,1]}) &\triangleq \frac{1}{2}\int_0^1 |\exp(-V_{\boldsymbol{\theta}}(\mathbf{x}_t,t)) - \exp(\lambda r(\mathbf{x}_1))|^2\,\mathrm{d}t, \\ \mathrm{d}\mathbf{x}_t &= b(\mathbf{x}_t,t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \quad \mathbf{x}_0 \sim p_0.\end{aligned} \tag{11}$$

However, this loss introduces several issues.

Instability. The exponential in Eq. (11) makes this method unstable. *E.g.*, it results in divergence to infinity of the loss when $\lambda r \gtrapprox 90$ in 32-bit precision. Applying a logarithm to the reward (*i.e.*, $\exp(\lambda r(\mathbf{x}_1)) \mapsto \exp(\lambda \log r(\mathbf{x}_1)) = r(\mathbf{x}_1)^\lambda$) does not resolve this issue because then it is exponential in λ. As a result, this loss function reduces the expressivity to small reward scalings.

Distribution Mismatch. Another issue is that $\mathcal{L}_{\mathrm{CG}}$ requires samples from the pre-trained distribution, making it off-policy. Next, we show this fact results in reduced sample-efficiency compared to VM.

6 Experimental Evaluation

We now evaluate **V**alue **M**atching (VM), aiming to showcase three primary insights: (i) verify that VM recovers the tilted distribution in an illustrative environment, (ii) demonstrate that VM scales to high-dimensional domains, and (iii) compare VM against classifier guidance, showing that VM is more sample-efficient and expressive. See Apx. F for experimental details.

Toy Environment. To verify that VM solves the objective, we apply it to a one-dimensional toy environment (Fig. 2a). It recovers the tilted distribution even with a non-differentiable reward.

Image Generation. To demonstrate VM's efficacy, we apply it to the Diffusion Transformer [41] 256×256 ImageNet [12] base model with a non-differentiable (in)compressibility reward, measured as bits per pixel after JPEG compression at quality 85. The VM versions are trained without classifier-free guidance (CFG) [27] and evaluated with CFG weight 2. Over 10K samples, VM scores 0.6 ± 0.3 and 3.1 ± 1.1 bits/pixel trained separately with compressibility and incompressibility rewards, respectively, compared to the base model's 1.9 ± 0.8 bits/pixel. The VRAM requirement for this task was 5.3 GB, as opposed to approximately 80 GB for fine-tuning the base model.

Molecular Design. For molecular design, we evaluate VM on the continuous FlowMol model [17], pre-trained on QM9 [45] and GEOM-Drugs [2]. The reward is the dipole moment, computed via `GFN2-xTB` [4] after geometry relaxation with `GFN-FF` [49]. Importantly, one way of increasing dipole moment is through fragmenting the molecule. To prevent this exploitation of the reward, we assign zero reward to fragmented samples. On GEOM-Drugs, over 10K samples, VM scores 7.5 ± 3.8 Debye with 28% fragmentation, *vs.* 6.4 ± 3.5 Debye with 31% for the base model. From this, we conclude that VM is able to increase the dipole moment without resorting to reward exploitation. The VRAM requirement for this task was $0.8\,\mathrm{GB}$ for QM9 and $5.3\,\mathrm{GB}$ for GEOM-Drugs.

Comparison with Classifier Guidance. As shown in Fig. 2c, classifier guidance is unstable for even moderate λ values, diverging when $\lambda \geq 10$. Moreover, Fig. 2d shows that VM reliably converges to better optima in small-scale experiments with small batch sizes.

Scaling the Value Network. Fig. 2e shows how the performance of VM varies with the value function's parameter count, breaking down $J(u)$ into its components. While larger models achieve higher rewards, they also incur greater running costs, resulting in similar total cost. This highlights that VM can be much cheaper than fine-tuning, as it does not require large networks.

7 Conclusion

We introduce **V**alue **M**atching (VM), a scalable method for adapting pre-trained flow models to arbitrary reward functions while preserving prior information. Leveraging insights from optimal control, VM employs a lightweight value function to guide the base model, eliminating costly fine-tuning. This approach reduces memory usage compared to fine-tuning and supports non-differentiable rewards. Further, we observe greater stability and sample efficiency than classifier guidance. Overall, VM provides a scalable, gradient-free, and resource-efficient alternative to fine-tuning.

Acknowledgments and Disclosure of Funding

This publication was made possible by the ETH AI Center doctoral fellowship to Riccardo De Santi. The project has received funding from the Swiss National Science Foundation under NCCR Catalysis (grant number 180544 and 225147) and NCCR Automation (grant agreement 51NF40 180545). This work was supported by an ETH Zurich Research Grant.

References

[1] M. S. Albergo and E. Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023.

[2] S. Axelrod and R. Gomez-Bombarelli. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):185, 2022.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] C. Bannwarth, S. Ehlert, and S. 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.

[5] R. E. Bellman and S. E. Dreyfus. *Applied dynamic programming*. Princeton university press, 2015.

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

[7] O. Bolza. *Lectures on the Calculus of Variations*, volume 14. University of Chicago Press, 1904.

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

[9] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. S. Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. In *The Eleventh International Conference on Learning Representations*, 2023.

[10] R. De Santi, M. Vlastelica, Y.-P. Hsieh, Z. Shen, N. He, and A. Krause. Flow density control: Generative optimization beyond entropy-regularized fine-tuning. In *The Exploration in AI Today Workshop at ICML 2025*, 2025.

[11] R. De Santi, M. Vlastelica, Y.-P. Hsieh, Z. Shen, N. He, and A. Krause. Provable maximum entropy manifold exploration via diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

[14] C. Domingo-Enrich, J. Han, B. Amos, J. Bruna, and R. T. Chen. Stochastic optimal control matching. *Advances in Neural Information Processing Systems*, 37:112459–112504, 2024.

[15] C. Domingo-Enrich, M. Drozdzal, B. Karrer, and R. 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.

[16] I. Dunn and D. R. Koes. Exploring discrete flow matching for 3d de novo molecule generation. *ArXiv*, pages arXiv–2411, 2024.

[17] I. Dunn and D. R. Koes. Mixed continuous and categorical flow matching for 3d de novo molecule generation. *ArXiv*, pages arXiv–2404, 2024.

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

[19] Y. Fan, O. Watkins, Y. Du, H. Liu, M. Ryu, C. Boutilier, P. Abbeel, M. Ghavamzadeh, K. Lee, and K. 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.

[20] W. H. Fleming and R. W. Rishel. *Deterministic and stochastic optimal control*, volume 1. Springer Science & Business Media, 2012.

[21] W. H. Fleming and H. M. Soner. *Controlled Markov processes and viscosity solutions*. Springer, 2006.

[22] A. Friedman. Stochastic differential equations and applications. In *Stochastic differential equations*, pages 75–148. Springer, 1975.

[23] I. Gat, T. Remez, N. Shaul, F. Kreuk, R. T. Chen, G. Synnaeve, Y. Adi, and Y. Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37:133345–133385, 2024.

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

[25] S. Gutjahr, R. D. Santi, L. Schaufelberger, K. Jorner, and A. Krause. Constrained molecular generation via sequential flow model fine-tuning. In *ICML 2025 Generative AI and Biology (GenBio) Workshop*, 2025.

[26] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

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

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

[29] E. Hoogeboom, V. G. Satorras, C. Vignac, and M. Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pages 8867–8887. PMLR, 2022.

[30] Z. Hu, F. Zhang, L. Chen, K. Kuang, J. Li, K. Gao, J. Xiao, X. Wang, and W. Zhu. Towards better alignment: Training diffusion models with reinforcement learning against sparse rewards. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23604–23614, 2025.

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

[32] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[34] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[35] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[36] Y. Lipman, M. Havasi, P. Holderrieth, N. Shaul, M. Le, B. Karrer, R. T. Chen, D. Lopez-Paz, H. Ben-Hamu, and I. Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

[37] X. Liu, C. 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.

[38] D. Maoutsa, S. Reich, and M. Opper. Interacting particle solutions of fokker–planck equations through gradient–log–density estimation. *Entropy*, 22(8):802, 2020.

[39] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

[40] 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.

[41] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[42] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[43] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024.

[44] L. S. Pontryagin. *Mathematical theory of optimal processes*. Routledge, 1962.

[45] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[46] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[47] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

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

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

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

[51] M. Uehara, Y. Zhao, K. Black, E. Hajiramezanali, G. Scalia, N. L. Diamant, A. M. Tseng, T. Biancalani, and S. Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[53] Y. Wu and K. He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[54] R. Zhang. Making convolutional networks shift-invariant again. In *International conference on machine learning*, pages 7324–7334. PMLR, 2019.

[55] H. Zhao, H. Chen, J. Zhang, D. Yao, and W. Tang. Score as action: Fine tuning diffusion generative models by continuous-time reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

Appendix Contents

A Related Work

Reward-Guided Flow Fine-Tuning for Generative Optimization. Recent work has explored RL as a way to fine-tune flow models toward objectives beyond likelihood maximization. Black et al. [6] introduced DDPO, which views the denoising process as a sequential decision problem and applies a policy gradient method to optimize arbitrary rewards, outperforming reward-weighted likelihood baselines. Fan et al. [19] extended this approach with DPOK, adding KL regularization to stabilize training and preserve the pre-trained model's fidelity. Hu et al. [30] addressed the challenge of sparse rewards, where feedback only comes at the end of the generation process, by proposing B^2-DiffuRL, which trains the model step-by-step starting from the final denoising stage and uses branch-based sampling to compare alternative trajectories, making it easier to identify which steps actually improve alignment while maintaining diversity. More recently, Zhao et al. [55] introduced score-as-action, a continuous-time RL framework that uses an actor-critic method to fine-tune diffusion models, avoiding discretization issues and supporting a wider range of solvers. In this setup, the framework first trains a critic (equivalent to the value function in our formulation) before fine-tuning the model. We argue that this second stage may not be needed, since the optimal policy can be defined directly through the value function, as demonstrated in this work.

Multiple recent works have formulated fine-tuning of flow models through the lens of SOC. Domingo-Enrich et al. [14] introduce SOCM, which learns controls via least-squares regression on a matching vector field, leveraging a path-wise reparameterization trick to achieve lower error and more stable training compared to prior iterative diffusion optimization methods. Building on this perspective, Uehara et al. [51] frame fine-tuning flow models as entropy-regularized SOC, showing that entropy regularization against the pre-trained model mitigates reward collapse by preserving sample diversity and proximity to the data distribution. Tang [50] provides a rigorous theoretical treatment of this framework, extending it beyond entropy to general f-divergences for fine-tuning. Most recently, Domingo-Enrich et al. [15] propose Adjoint Matching, which resolves a key value function bias in earlier SOC formulations by enforcing a memoryless noise schedule, thereby enabling provably unbiased fine-tuning.

Flow Adaptation Beyond Reward Maximization. Recent works show that solving the entropy-regularized fine-tuning objective can be used as an oracle for far more general adaptation processes beyond reward maximization. This includes S-MEME [11], which employs Adjoint Matching [15] as an oracle for manifold exploration through entropy maximization. Further, FDC [10] extends this framework to arbitrary distributional utilities under general divergences. Moreover, ALF^2 [25] adopts the same oracle to solve the entropy-regularized fine-tuning objective with arbitrary constraints. A limitation of using Adjoint Matching as the oracle is that it requires the reward function and constraints to be differentiable. Our method could alleviate this issue by substituting Adjoint Matching by our method as the oracle.

B Further Background on Classifier Guidance

Classifier guidance [13] guides a generative model by augmenting the base score function with the log-gradient of a separately trained classifier $p(y \mid \mathbf{x}_t)$, following Bayes' rule:

$$\nabla \log p(\mathbf{x}_t \mid y) = \nabla \log\left(\frac{p(\mathbf{x}_t)p(y \mid \mathbf{x}_t)}{p(y)}\right) \tag{12}$$

$$= \nabla \log p(\mathbf{x}_t) + \nabla \log p(y \mid \mathbf{x}_t). \tag{13}$$

To bias samples toward regions where y is more likely, a nonnegative weight $w \in \mathbb{R}_{\geq 0}$ is introduced:

$$\nabla \log p(\mathbf{x}_t) + w\nabla \log p(y \mid \mathbf{x}_t) \tag{14}$$

The parameter w governs a trade-off: larger values emphasize sample quality at the cost of diversity, whereas smaller values favor diversity but reduce quality. In practice, classifier guidance and CFG [27] are widely used to improve FID scores. In this work, we reinterpret classifier guidance from a control-theoretic perspective under a specific reward function $r(\mathbf{x}) = \log p(y \mid \mathbf{x})$.

C Proofs

C.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.

C.2 Useful Lemmas

Lemma 1 (Application of the Feynman-Kac formula [31])**.** *Let V be the value function defined in Eq. (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 [22]; 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 and Soner [21], 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 Eq. (17) is unique.*

Proof. We show this result by a comparison principle. Assume that Eq. (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:

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

C.3 Proposition 1

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

Proof. Let $t < 1$. Define $\psi(\mathbf{x}, t) \triangleq \mathbb{E}_{p^{\mathrm{pre}}}[\exp(\lambda r(\mathbf{x}_1)) \mid \mathbf{x}_t = \mathbf{x}]$. 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:

$$\begin{aligned} \|\nabla_{\mathbf{x}} \exp(\lambda r(\mathbf{y})) p_{1|t}(\mathbf{y} \mid \mathbf{x})\| &= \exp(\lambda r(\mathbf{y})) \|\nabla_{\mathbf{x}} p_{1|t}(\mathbf{y} \mid \mathbf{x})\| & (25) \\ &\leq MC(1-t)^{-\frac{d+1}{2}} \exp\left(-c \frac{\|\mathbf{y} - \mathbf{x}\|^2}{1-t}\right), & (26) \end{aligned}$$

   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 $t < 1$. □

C.4 Proposition 2

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

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 `stropgrad` 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 $\mathbb{E}[\mathcal{L}_{\mathrm{VM}}]$. Let $C : \mathbb{R}^d \times [0,1] \to \mathbb{R}$ be an arbitrary function, then:

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}\epsilon} \mathbb{E}[\mathcal{L}_{\mathrm{VM}}(W + \epsilon C; \mathbf{x}_{[0,1]})]\Big|_{\epsilon=0} & (30) \\ &= \frac{\mathrm{d}}{\mathrm{d}\epsilon} \mathbb{E}\left[\frac{1}{2}\int_0^1 w(t) \cdot \left|(W + \epsilon C)(\mathbf{x}_t, t) - J\big(t; \mathbf{x}_{[0,1]}, -\sigma^\intercal \nabla \bar{W}\big)\right|^2 \mathrm{d}t\right]\Bigg|_{\epsilon=0} & (31) \\ &= \mathbb{E}\left[\frac{1}{2}\int_0^1 w(t) \cdot \frac{\mathrm{d}}{\mathrm{d}\epsilon}\left|(W + \epsilon C)(\mathbf{x}_t, t) - J\big(t; \mathbf{x}_{[0,1]}, -\sigma^\intercal \nabla \bar{W}\big)\right|^2\Bigg|_{\epsilon=0} \mathrm{d}t\right] & (32) \\ &= \mathbb{E}\left[\int_0^1 C(\mathbf{x}_t, t) \cdot w(t) \cdot \big(W(\mathbf{x}_t, t) - J\big(t; \mathbf{x}_{[0,1]}, -\sigma^\intercal \nabla \bar{W}\big)\big) \, \mathrm{d}t\right] & (33) \end{aligned}$$

Using the tower property of expectation:

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

So, the functional derivative is:

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

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

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

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

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

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

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

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

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:

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

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:

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

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

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

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 $\mathbb{E}[\mathcal{L}_{\mathrm{VM}}]$. □

C.5 Derivation of Classifier Guidance Value Function

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

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

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

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

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

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

By marginalization:

$$= -\log p_{Y|t}(\mathbf{x}_1, y \mid \mathbf{x}). \tag{49}$$

This concludes the derivation of the statement.

D Value Matching Algorithm

We explicitly write the full algorithm in Alg. 1. To implement it practically, one needs to determine the number of steps T to discretize the SDE and integrals with.

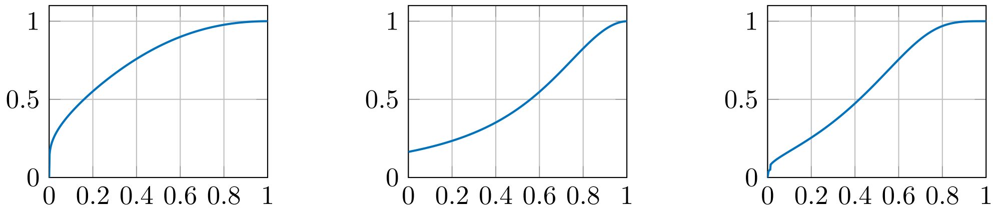
Algorithm 1 Value Matching algorithm.

```
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 $w : [0,1] \to \mathbb{R}_{>0}$.
for $N$ iterations do
    Sample $m$ trajectories under the current policy:
        $\mathrm{d}\mathbf{x}_t = \big(b(\mathbf{x}_t, t) - \sigma^2(t)\nabla V_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\big)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \quad \mathbf{x}_0 \sim p_0.$
    Compute the cost functionals:
        $J_t = \frac{1}{2}\int_t^1 \|\sigma(s)\nabla V_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_s, s)\|^2\,\mathrm{d}s - r(\mathbf{x}_1), \quad \bar{\boldsymbol{\theta}} = \texttt{stopgrad}(\boldsymbol{\theta}).$
    Compute the loss function:
        $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\int_0^1 w(t) \cdot |V_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - J_t|^2\,\mathrm{d}t.$
    Make an optimization step with $\nabla\mathcal{L}(\boldsymbol{\theta})$.
end for
```

The weighting function $w : [0,1] \to \mathbb{R}_{>0}$ is used to stabilize training; under the memoryless schedule, $\sigma(t) \to \infty$ as $t \to 0$, causing numerical issues. Empirically, we find the following weighting to perform well (see Fig. 3 for plots for the base models considered in this work):

$$w(t) = \frac{1}{\lambda^2\left(1 + \frac{1}{2}\int_t^1 \sigma^2(s)\,\mathrm{d}s\right)}. \tag{50}$$

This weighting scheme normalizes rewards by the scaling factor λ and down-weights timesteps with high future variance. For models with multiple schedulers (*e.g.*, FlowMol), weights are averaged across schedulers. Without weighting (*i.e.*, $w(t) = 1$), we find that VM almost always diverges.



(a) Optimal transport schedule.

(b) DDPM schedule.

(c) FlowMol schedule.

Figure 3: Weighting function $w(t)$ under various (α_t, β_t)-schedules, $\lambda = 1$, and the memoryless noise schedule.

E Flow Models

For completeness, in this section we provide an overview of flow matching [35, 1, 37] and how diffusion models [28, 47, 48] 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.

E.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:

$$\mathrm{d}\mathbf{x}_t = v_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\,\mathrm{d}t. \tag{51}$$

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 [34] 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:

$$p_t(\mathbf{x}) = \int p_{t|1}(\mathbf{x} \mid \mathbf{x}_1)q(\mathbf{x}_1)\,\mathrm{d}\mathbf{x}_1. \tag{52}$$

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

$$p_{t|1}(\mathbf{x} \mid \mathbf{x}_1) = \mathcal{N}\big(\mathbf{x}; \alpha_t\mathbf{x}_1, \beta_t^2\mathbf{I}_d\big), \tag{53}$$

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 $t \in (0, 1)$.[2] We can sample from this distribution as follows:

$$\mathbf{x}_{t|1} = \alpha_t\mathbf{x}_1 + \beta_t\mathbf{x}_0, \quad \mathbf{x}_0 \sim p_0. \tag{54}$$

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:

$$v(\mathbf{x}, t) = \mathbb{E}[\dot{\alpha}_t\mathbf{x}_1 + \dot{\beta}_t\mathbf{x}_0 \mid \mathbf{x}_t = \mathbf{x}]. \tag{55}$$

Thus, we can train using the flow matching loss:

$$\begin{gathered}\mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{t,\mathbf{x}_t}\big[\|v_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - v(\mathbf{x}_t, t)\|^2\big], \\ t \sim \mathcal{U}([0, 1]), \mathbf{x}_t \sim p_t.\end{gathered} \tag{56}$$

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 $\mathbf{x}_1$:

$$\begin{gathered}\mathcal{L}_{\mathrm{CFM}}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{t,\mathbf{x}_1,\mathbf{x}_t}\big[\|v_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - v(\mathbf{x}_t, t \mid \mathbf{x}_1)\|^2\big], \\ t \sim \mathcal{U}([0, 1]), \mathbf{x}_1 \sim p_1, \mathbf{x}_t \sim p_{t|1}(\cdot \mid \mathbf{x}_1),\end{gathered} \tag{57}$$

[2] The dot-notation denotes the time-derivative.

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

where the conditional velocity is

$$\begin{aligned} v(\mathbf{x}, t \mid \mathbf{x}_1) &= \dot{\alpha}_t \mathbf{x}_1 + \dot{\beta}_t \mathbf{x}_0, \quad \mathbf{x}_t = \mathbf{x} && (58) \\ &= \dot{\alpha}_t \mathbf{x}_1 + \frac{\dot{\beta}_t}{\beta_t}(\mathbf{x} - \alpha_t \mathbf{x}_1) && (59) \\ &= \left(\dot{\alpha}_t - \frac{\alpha_t \dot{\beta}_t}{\beta_t} \right) \mathbf{x}_1 + \frac{\dot{\beta}_t}{\beta_t} \mathbf{x}. && (60) \end{aligned}$$

Amazingly, these two loss functions have the same gradient w.r.t. the parameters [35]:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{FM}} = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CFM}}. \tag{61}$$

This justifies applying gradient-based optimization methods on the conditional loss, which is tractable, because it will lead to the same parameter updates. See Alg. 2 for the training algorithm. Refer to [36] 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:

$$\mathrm{d}\mathbf{x}_t = \left(v(\mathbf{x}_t, t) + \frac{\sigma^2(t)}{2\eta_t} (v(\mathbf{x}_t, t) - \kappa_t \mathbf{x}_t) \right) \mathrm{d}t + \sigma(t)\, \mathrm{d}B_t, \tag{62}$$

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

$$\eta_t \triangleq \beta_t \left(\frac{\dot{\alpha}_t}{\alpha_t} \beta_t - \dot{\beta}_t \right), \quad \kappa_t \triangleq \frac{\dot{\alpha}_t}{\alpha_t}. \tag{63}$$

It can be shown that the generative processes in Eq. (51) and Eq. (62) have equivalent time marginals [38]. In the memoryless noise schedule, we have $\sigma(t) = \sqrt{2\eta_t}$ [15].

E.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:

$$\mathbf{x}_{t+1} = \sqrt{\gamma_t} \mathbf{x}_t + \sqrt{1 - \gamma_t} \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n), \tag{64}$$

where γ_t follows some schedule from 0 to T such that $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$.[3] 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:

$$\mathbf{x}_t = \sqrt{\bar{\gamma}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\gamma}_t} \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n), \tag{65}$$

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

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\gamma_t}} \left(\mathbf{x}_t - \frac{1 - \gamma_t}{\sqrt{1 - \bar{\gamma}_t}} \boldsymbol{\epsilon}_t \right) + \sigma_t \mathbf{z}, \quad \mathbf{z} \in \mathcal{N}(\mathbf{0}, \mathbf{I}_d). \tag{66}$$

Here the only unknown is $\boldsymbol{\epsilon}_t$, so we will train a network to approximate it; see Alg. 3

[3] 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 3 Diffusion model training.

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

E.3 Diffusion Models as an Instance of Flow Matching

We can sample a diffusion model with the DDIM schedule through the following SDE [15]:

$$\mathrm{d}\mathbf{x}_t = \left(\frac{\dot{\bar{\gamma}}_t}{2\bar{\gamma}_t}\mathbf{x}_t - \left(\frac{\dot{\bar{\gamma}}_t}{2\bar{\gamma}_t} + \frac{\sigma^2(t)}{2}\right)\frac{\epsilon(\mathbf{x}_t, t)}{\sqrt{1 - \bar{\gamma}_t}}\right)\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t. \tag{67}$$

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:

$$\begin{align} s(\mathbf{x}, t) &\triangleq \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \tag{68}\\ s(\mathbf{x}, t) &= \frac{1}{\eta_t}(v(\mathbf{x}, t) - \kappa_t \mathbf{x}) \tag{69}\\ s(\mathbf{x}, t) &= -\frac{\epsilon(\mathbf{x}, t)}{\sqrt{1 - \bar{\gamma}_t}}. \tag{70}\end{align}$$

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:

$$\mathrm{d}\mathbf{x}_t = \left(\kappa_t \mathbf{x}_t + \left(\frac{\sigma^2(t)}{2} + \eta_t\right) s(\mathbf{x}_t, t)\right)\mathrm{d}t + \sigma(t)\,\mathrm{d}B_t, \tag{71}$$

In the case of diffusion models, we have

$$\alpha_t = \sqrt{\bar{\gamma}_t}, \quad \beta_t = \sqrt{1 - \bar{\gamma}_t} \tag{72}$$

with associated time derivatives:

$$\dot{\alpha}_t = \frac{\dot{\bar{\gamma}}_t}{2\sqrt{\bar{\gamma}_t}}, \quad \dot{\beta}_t = -\frac{\dot{\bar{\gamma}}_t}{2\sqrt{1 - \bar{\gamma}_t}}. \tag{73}$$

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

$$\begin{align} \bar{\gamma}_t &= \bar{\gamma}\lfloor K(1 - t)\rfloor \tag{74}\\ \dot{\bar{\gamma}}_t &= K \cdot (\bar{\gamma}\lfloor K(1 - t) - 1\rfloor - \bar{\gamma}\lfloor K(1 - t)\rfloor). \tag{75}\end{align}$$

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

F Experimental Details

All experiments use the Adam optimizer [32] with a learning rate of 1×10^{-4}, a batch size of 128, and 100 SDE discretization steps. Furthermore, we clip gradients to unit norm before every update to the parameters. Unless stated otherwise, image experiments use a 1.8M-parameter convolutional neural network (CNN) and molecular experiments a 2.5M-parameter graph neural network (GNN) to parameterize the value function approximator $V_{\boldsymbol{\theta}}$. Further, to normalize the classifier guidance 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 $\exp(2\lambda)$.

F.1 Architectures

CNN. In this work, we use a fairly standard CNN architecture for our experiments. It consists of an input convolution, three downsampling stages, followed by an adaptive average pool, and a final linear head. To embed the timesteps, we use a sinusoidal timestep embedder [52]. Each downsampling stage consists of two layers, each consisting of a convolution with a 3×3 kernel $\rightarrow$ group normalization [53] $\rightarrow$ FiLM [42] to incorporate timestep information $\rightarrow$ sigmoid linear unit [26] activation function; lastly, the input is added residually and the result is downsampled with a blur pool [54]. For the convolutional layers in the downsampling stages, the base hidden dimensionality is 64, doubling every stage.

GNN. The GNN architecture used is similar to the CNN architecture (minus the downsampling), where we replace the input convolution by a linear layer, convolutions with graph convolutions [33], and the group normalization by layer normalization [3]. We use all the node information available from the FlowMol model: atom position, atom type, and formal charge. Also, we make use of the 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. The hidden dimensionality used for each block is 256 and 6 stages are used.

G Figure Details

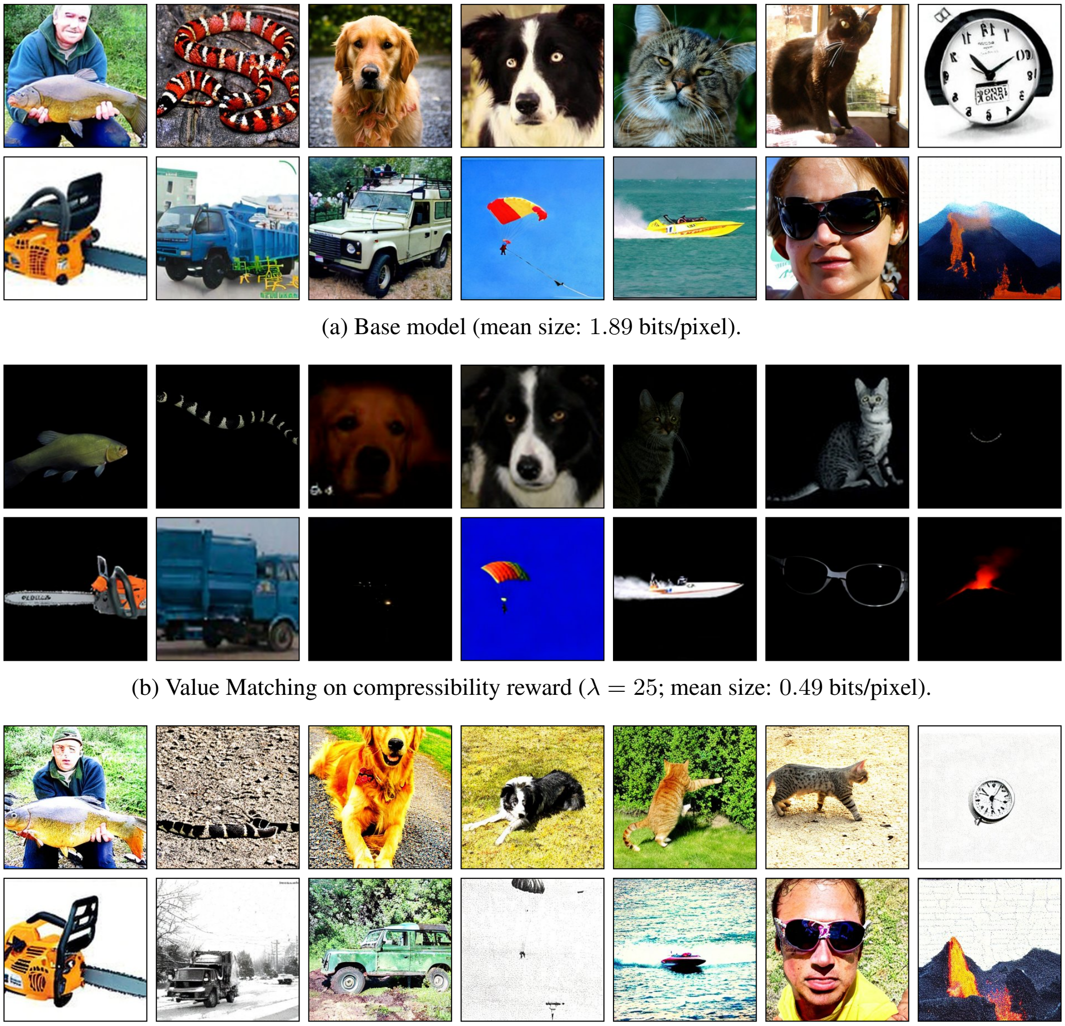
Fig. 1. The VRAM requirement is approximated by loading the model and measuring the VRAM usage for a forward pass (with gradient computation) for batch sizes 1 and 2. Then, the VRAM requirement is computed by $V_1 + 127 \times (V_2 - V_1)$. The measured base models are Diffusion Transformer (DiT) [41], Segmind Stable Diffusion (SSD) [24], Stable Diffusion 1.5 and 2.1 (SD1.5 and SD2.1) [46], Stable Diffusion XL (SDXL) [43], and Stable Diffusion 3 (SD3) Medium [18]. Keep in mind that these estimates are made in 16-bit precision, whereas we apply VM in 32-bit precision.

H 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 [15]:

$$D_{\mathrm{KL}}\big(p^u(\mathbf{x}_{[0,1]} \mid \mathbf{x}_0) \,\big\|\, p^{\mathrm{pre}}(\mathbf{x}_{[0,1]} \mid \mathbf{x}_0)\big) = \mathbb{E}_{p^u}\left[\frac{1}{2}\int_0^1 \|u(\mathbf{x}_t, t)\|^2 \,\mathrm{d}t\right] \tag{76}$$

H.1 Diffusion Transformer

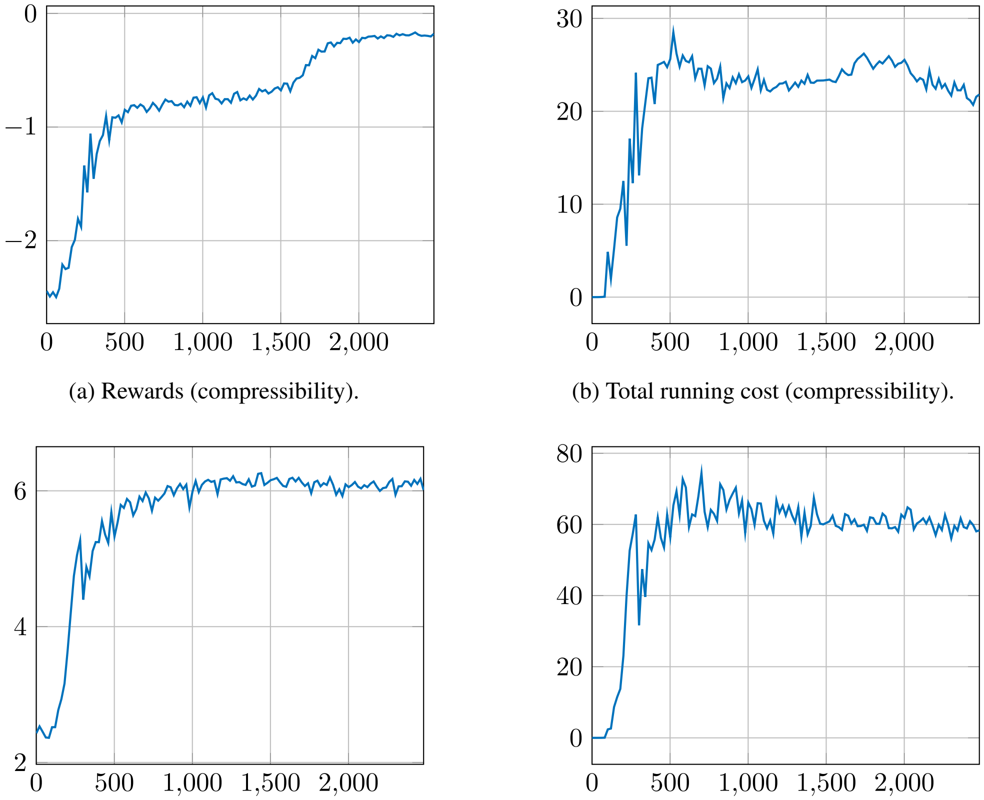


(a) Base model (mean size: 1.89 bits/pixel).

(b) Value Matching on compressibility reward ($\lambda = 25$; mean size: 0.49 bits/pixel).

(c) Value Matching on incompressibility reward ($\lambda = 25$; mean size: 3.31 bits/pixel).

Figure 4: Samples from Diffusion Transformer models generated under the same random seed. Inference with CFG weight 2, whereas training was done without CFG.



(a) Rewards (compressibility).

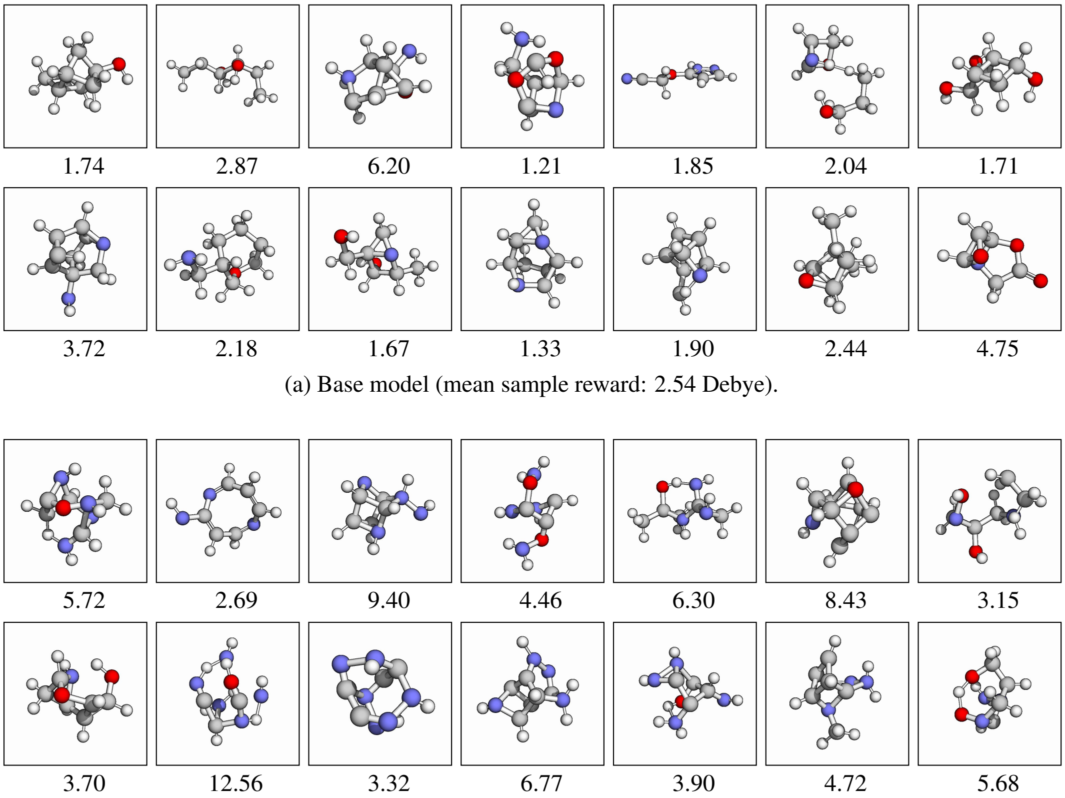
(b) Total running cost (compressibility).

(c) Rewards (incompressibility).

(d) Total running cost (incompressibility).

Figure 5: Training curves for Value Matching on the Diffusion Transformer model with compressibility and incompressibility rewards ($\lambda = 25$).

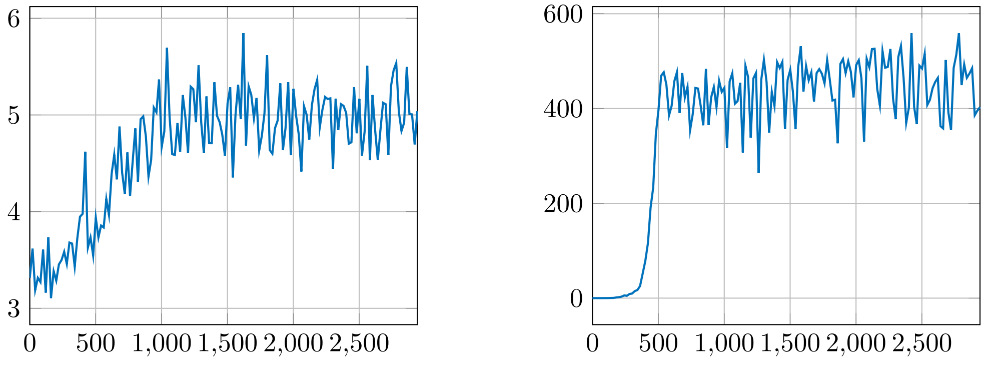
H.2 QM9



(a) Base model (mean sample reward: 2.54 Debye).

(b) Value Matching on dipole moment reward ($\lambda = 100$; mean sample reward: 5.77 Debye).

Figure 6: Samples from the continuous QM9 FlowMol base and Value Matching model under the same random seed.

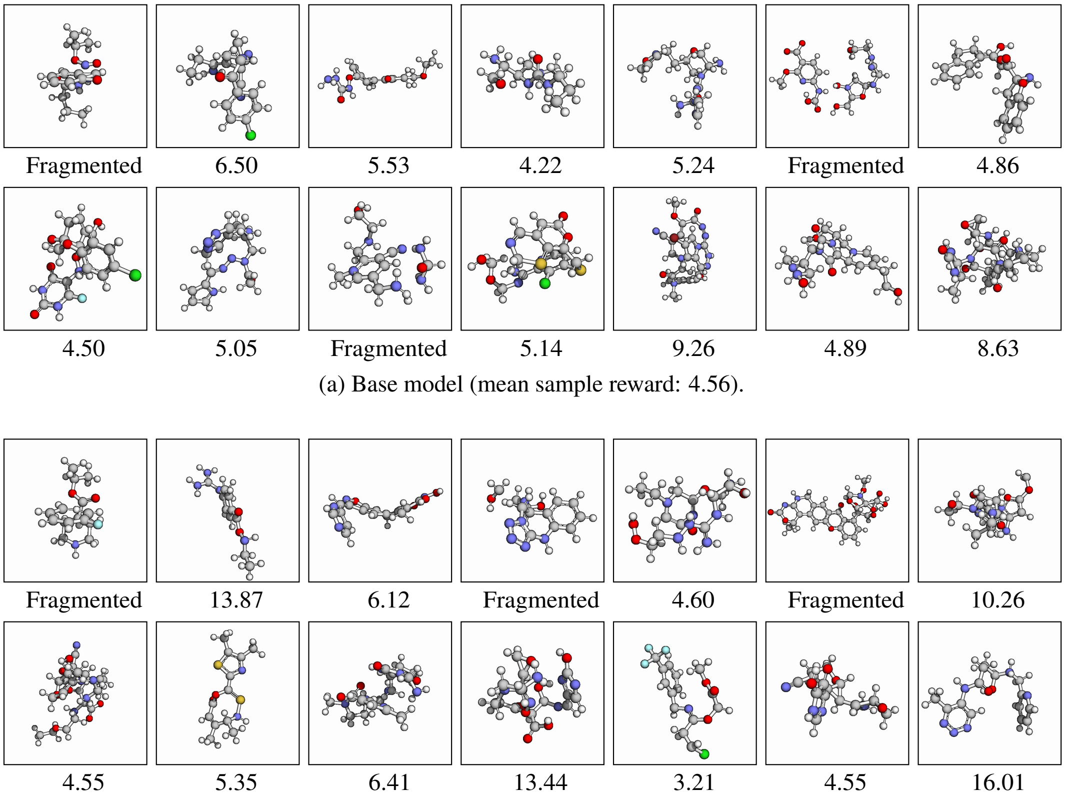


(a) Rewards.

(b) Total running cost.

Figure 7: Training curves for Value Matching on the continuous QM9 FlowMol model with dipole moment reward ($\lambda = 100$).

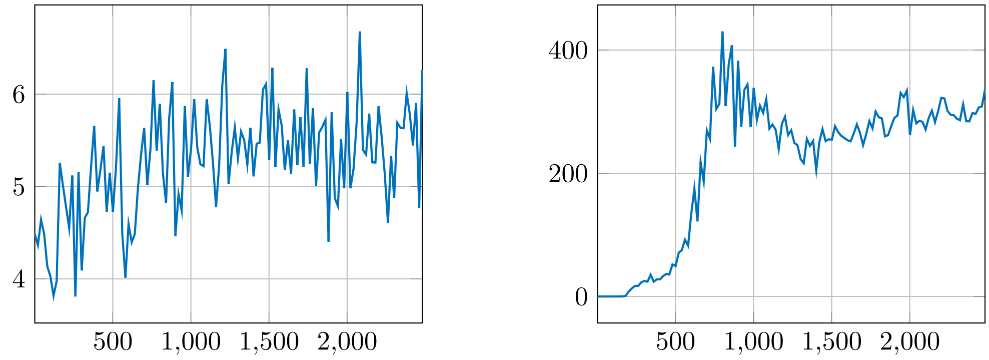
H.3 GEOM-Drugs



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

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

Figure 8: Samples from the continuous GEOM-Drugs FlowMol base and Value Matching model under the same random seed.



(a) Rewards.

(b) Total running cost.

Figure 9: Training curves for Value Matching on the continuous GEOM-Drugs FlowMol model with dipole moment reward ($\lambda = 100$).