Diffusion Tree Sampling: Scalable inference-time alignment of diffusion models

Vineet Jain [1 2] **Kusha Sareen** [1 2] **Mohammad Pedramfar** [1 2] **Siamak Ravanbakhsh** [1 2]

Abstract

Adapting pretrained diffusion models to new objectives at inference remains an open problem in generative modeling. Existing steering methods yield biased estimates at high noise levels due to inaccurate value estimation, and discard information from prior computations, leading to inefficient use of compute. We address these limitations by casting inference-time alignment as a search problem that reuses past computations. Inspired by Monte Carlo Tree Search, we propose Diffusion Tree Sampling (DTS), a novel inference-time alignment approach that *samples* from the reward-aligned target density by propagating terminal rewards back through the diffusion chain and iteratively refining value estimates with each additional generation. DTS recovers exact samples from the reward-weighted target, while its greedy variant, Diffusion Tree Search (DTS*), efficiently finds high-reward samples. On MNIST and CIFAR-10 class-conditional generation, DTS matches best-performing baseline with $5\times$ less compute. In text-to-image and language completion tasks, DTS* matches best-of-N with $2\times$ less compute. Our method is an *anytime algorithm*, which turns additional compute budget into better samples, providing a scalable framework for inference-time diffusion alignment.

1. Introduction

Diffusion models have become a dominant framework for generative modeling, achieving state-of-the-art results in image synthesis, molecular conformer generation, and text generation (Ho et al., 2020; Hoogeboom et al., 2022; Sahoo et al., 2024). Yet guiding a pretrained diffusion model to satisfy new objectives at inference—without costly retraining—remains an open challenge, as existing steering approaches struggle to adapt dynamically to user-defined rewards (Uehara et al., 2025).

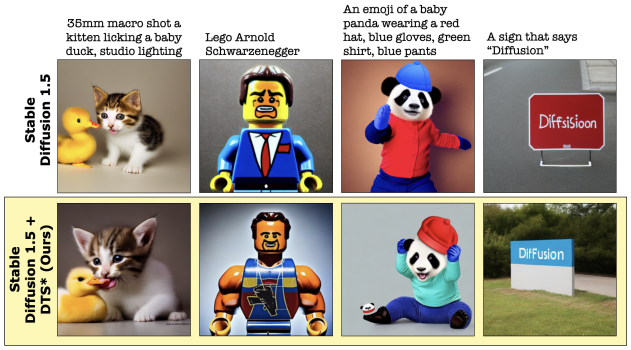

Figure 1: Sample text-image pairs using Stable Diffusion v1.5 (Rombach et al., 2022) and ImageReward (Xu et al., 2023) as the guiding function, with generated samples picked at random for each method and prompt.

Most alignment tasks can be cast as drawing samples from the diffusion prior weighted by exponentiated rewards. However, since rewards are only observed at the final denoising step, inference-time alignment must guide intermediate states based on unseen terminal feedback. This delayed-reward setting poses a credit-assignment problem (Minsky, 1961): *how can we estimate the value of noisy states using rewards only observed at the end?*

Prior methods approximate terminal values in different ways: gradient-based guidance perturbs denoising via reward gradients (Dhariwal & Nichol, 2021), sequential Monte Carlo (SMC) maintains particle sets and resamples based on predicted rewards (Wu et al., 2023a), and recent search schemes perform local greedy trajectory optimization (Li et al., 2024). All rely on intermediate reward proxies, which biases decision-making at high noise levels.

Beyond value estimation, effective inference-time alignment should also leverage information from past rollouts to refine future samples. This raises two core questions: (i) how to obtain low-bias, low-variance value estimates under delayed feedback; and (ii) how to systematically reuse computations across rollouts in a scalable sampling process?

Monte Carlo Tree Search (MCTS), a classical reinforcement learning algorithm, addresses both of these issues for sequential decision-making (Browne et al., 2012). We observe that during denoising, the model acts can be viewed as a de-

[*]Equal contribution [1]School of Computer Science, McGill University, Montréal, Canada [2]Mila - Quebec AI Institute, Montréal, Canada. Correspondence to: Vineet Jain <jain.vineet@mila.quebec>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

terministic policy, while the Gaussian reverse step serves as a stochastic environment transition. This suggests applying tree search to inference-time diffusion alignment.

We introduce Diffusion Tree Sampling (DTS), a novel inference-time alignment method that casts the denoising process as a finite-horizon tree, where similar to MCTS, rollouts are used to continuously improve value estimates for intermediate noisy states. For applications requiring optimization rather than sampling, we propose a search variant Diffusion Tree Search (DTS*), that identifies the modes within high-volume regions of the target density.

Our contributions can be summarized as follows:

- We formulate inference-time alignment of diffusion models as a tree search problem for sampling from the reward-aligned distribution or optimizing for high reward samples.
- We develop a general tree-based algorithm that yields asymptotically exact samples from the target distribution in the limit of infinite rollouts.
- We demonstrate that DTS significantly reduces bias and variance in value estimation compared to common approximations used by many existing methods.
- We show that DTS and DTS* scale more favorably compared to leading baselines and match their performance with up to $5\times$ less compute on class-conditional image generation, and $2\times$ less compute on text-to-image alignment and language completion tasks.

2. Background and Problem Setting

Diffusion models (Ho et al., 2020; Song et al., 2021) define a generative process via a Markov chain that gradually adds noise to data $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$,

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

so that at $t = T$, $\mathbf{x}_T \sim \mathcal{N}(0, I)$. A learned reverse process then denoises samples sequentially with the following transition and join density:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1};\, \mu_\theta(\mathbf{x}_t, t),\, \sigma_t^2 I\right),$$
$$p_\theta(\mathbf{x}_0) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t).$$

To steer a pretrained model toward desirable outputs, we assume a reward function $r(\mathbf{x})$ and define the target density

$$\pi^*(\mathbf{x}) \;\propto\; p_\theta(\mathbf{x}) \exp\big(r(\mathbf{x})\big), \tag{1}$$

from which we wish either to sample or to identify high-reward modes. Since $r(\mathbf{x})$ is only observed at the end of the denoising chain, guiding intermediate steps requires estimating, for each noisy state $\mathbf{x}_t$, the expected terminal reward under the reverse model.

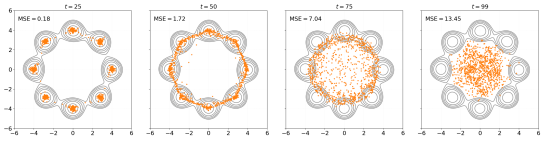

Figure 2: One-step prediction using Tweedie's formula for different time steps, along with average mean squared error with the ground truth data samples. Close to $t = 0$, the predictions are fairly accurate, but towards the maximum timestep $T = 99$, they devolve into random predictions.

We can view this as a finite-horizon Markov decision process, with the denoising network acting like a policy. The corresponding *soft value function*

$$V_t(\mathbf{x}_t) = \log \mathbb{E}_{p_\theta(\mathbf{x}_{0:t-1}|\mathbf{x}_t)}\Big[\exp\big(r(\mathbf{x}_0)\big)\Big]$$
$$V_t(\mathbf{x}_t) = \frac{1}{\lambda} \log \mathbb{E}_{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\left[\exp\left(\lambda V_{t-1}(\mathbf{x}_{t-1})\right)\right], \tag{2}$$

satisfies a soft-Bellman recursion analogous to RL, and the Boltzmann density corresponding to this soft value defines the optimal sampling policy at any timestep t (Ziebart et al., 2008; Haarnoja et al., 2017).

Existing inference-time alignment methods approximate $V_t(\mathbf{x}_t)$ in two stages: first by Jensen's inequality,

$$V_t(\mathbf{x}_t) \approx \mathbb{E}_{p_\theta(\mathbf{x}_{0:t-1}|\mathbf{x}_t)}[\, r(\mathbf{x}_0)\,],$$

and then by a one-step proxy $\mathbb{E}[\, r(\mathbf{x}_0)\,] \approx r\big(\hat{\mathbf{x}}_0\big)$, where

$$\hat{\mathbf{x}}_0 = \tfrac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t + (1-\bar{\alpha}_t)\,\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\Big),$$

where $\hat{\mathbf{x}}_0$ is the obtained using Tweedie's formula (Efron, 2011; Chung et al., 2023). In practice, this single-step estimate is accurate at low noise but degrades to near-random at high t, biasing guidance (see Figure 2).

Moreover, gradient guidance (Chung et al., 2023; Song et al., 2023), SMC (Wu et al., 2023a; Kim et al., 2025), or search methods (Li et al., 2024; 2025; Yoon et al., 2025) treat each sample run independently, discarding all intermediate evaluations. Consequently, they can only scale by increasing particle or rollout counts, without any mechanism to reuse past computations to correct value-estimate errors or improve sample quality over time.

3. Diffusion Tree Sampling and Search

The pitfalls above suggest two complementary desiderata for an effective inference–time sampler:

(D1) Use information from low-noise timesteps, where the reward signal is reliable, to *refine decisions made at high-noise timesteps*.

(D2) Reuse information from previous trajectories so that *additional compute improves sample quality* – a property characteristic of an anytime algorithm.

3.1. Denoising tree

The Markov property of the reverse diffusion chain naturally induces a finite horizon tree in $\mathbb{R}^d$, where d is the dimensionality of the space over which we are diffusing. Here, the nodes at depth t represent noisy states $\mathbf{x}_t$ and the edges represent a denoising step. Each node $\mathbf{x}_t$ can be stochastically denoised into multiple children $\mathbf{x}_{t-1} \sim p_\theta(\cdot \mid \mathbf{x}_t)$.

This framing allows us to track information across multiple denoising trajectories, including estimates of the soft value function, which helps with global credit assignment. The tree structure also gives us the flexibility to sample from the target density or search for the highest reward sample with minimal changes to the underlying algorithm.

3.2. Tree-based sampling

Similar to MCTS, we construct a tree $\mathcal{T}$, where nodes represent states $\mathbf{x}_t$ and edges represent transitions $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ following the base diffusion model. Each node maintains the current state and timestep $(\mathbf{x}_t, t)$, an estimate of the soft value function $\hat{v}(\mathbf{x}_t)$, and the visit count $N(\mathbf{x}_t)$. Since we do not have a fixed starting state, we introduce a dummy state as the root $\mathbf{x}_{T+1}$ that transitions to the prior in our diffusion model – i.e., $p(\mathbf{x}_T \mid \mathbf{x}_{T+1}) = \mathcal{N}(\mathbf{0}, I)$. Additionally, we use $\mathcal{C}(\mathbf{x}_t)$ to denote the set of children of node $\mathbf{x}_t$.

The goal is to expand this tree while improving value estimates as we expand it, so that it can be used for approximate sampling from the target distribution at any time during the construction. The resulting tree sampling process provably samples from the target distribution π^* in the limit of infinite rollouts. The tree-building procedure of DTS repeats the following steps iteratively:

1. **Selection.** Starting from the root $\mathbf{x}_0$, sample a child $\mathbf{x}_{t-1} \in \mathcal{C}(\mathbf{x}_t)$ from Boltzmann distribution of the values $\propto \exp\left(\lambda\, \hat{v}(\mathbf{x}_{t-1})\right)$ recursively until either an unexpanded node is reached or $t = 0$.
2. **Expansion.** If we reach a node $\mathbf{x}_t$ such that the number of children is less than the maximum allowed value and $t > 0$, we create a new child node $\mathbf{x}_{t-1} \sim p_\theta(\cdot \mid \mathbf{x}_t)$ and initialize $\hat{v}(\mathbf{x}_{t-1}) = 0,\ N(\mathbf{x}_{t-1}) = 1$.
3. **Rollout.** From the newly created node, we perform a rollout till terminal states $\mathbf{x}_0$ by recursively sampling from $p_\theta(\cdot \mid \mathbf{x}_{t'})$ for $t' = t-1, \ldots, 0$. An important distinction from traditional MCTS is that we add the rollout path to $\mathcal{T}$.
4. **Backup.** Evaluate the terminal node using the reward function $\hat{v}(\mathbf{x}_0) = r(\mathbf{x}_0)$ and use soft Bellman equation (Equation (2)) to update parent node values using the children node values recursively for $t = 0, \ldots, T$. The visit counts for all nodes in the path are also updated.

Each traversal from the root to the backup of the value function constitutes one tree-building iteration. For sampling from $\mathcal{T}$, we simply start from the root and perform selection steps until we reach a terminal node. A formal algorithm is provided in Appendix F and design choices in Appendix G. We now show that DTS is asymptotically consistent, i.e., it samples from the correct distribution in the limit of infinite iterations. The proof is provided in Appendix E.

Proposition 3.1 (Asymptotic consistency)**.** *Let r be bounded and $\lambda > 0$, then* DTS *produces a sequence of terminal states whose empirical distribution converges to the optimal policy π^* as the number of tree iterations $M \to \infty$.*

4. Experiments

4.1. Class-conditional posterior sampling

We evaluate DTS on the task of sampling from a class-conditioned posterior distribution $p(\mathbf{x} \mid c) \propto p_\theta(\mathbf{x}) p(c \mid \mathbf{x})$ where $p_\theta(\mathbf{x})$ is a pretrained unconditional diffusion model and $p(c \mid \mathbf{x})$ is a classifier. This would correspond to setting $r(\mathbf{x}) = \log p(c \mid \mathbf{x})$ in Equation (1).

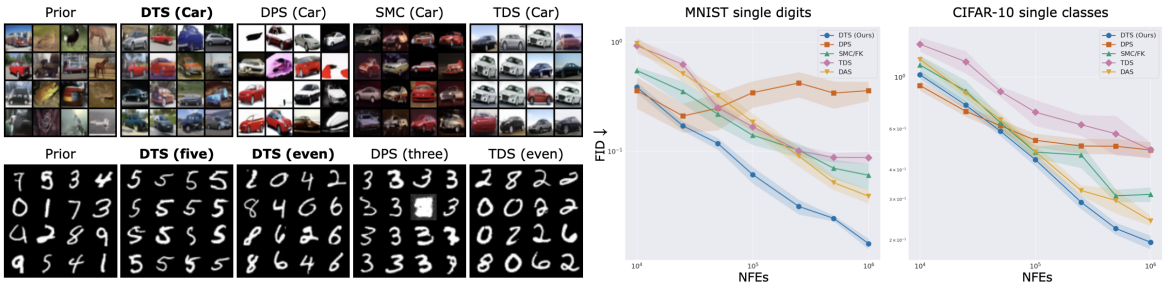

Figure 3: Samples generated from the CIFAR-10 (upper left) and MNIST (lower left) base diffusion models, and posterior samples using different methods at 10^6 NFEs. Gradient-based guidance like DPS can be unstable leading to samples that lie outside the support of the prior. SMC-based methods struggle to accurately sample from multi-modal distributions – for MNIST even digits, TDS oversamples from the digit two and undersamples from the digit four, and for CIFAR-10 car, both SMC and TDS suffer from mode collapse. (Right) FID versus number of function evaluations for different methods on MNIST single digit and CIFAR-10 single class, averaged over all 10 classes. All methods were evaluated with 5000 generated samples per class.

Setting. We use MNIST and CIFAR-10 datasets and train prior unconditional diffusion model from scratch on MNIST and use an off-the-shelf model[1] for CIFAR-10. For MNIST, we consider two settings: sampling from individual digits, and sampling from even/odd digits. The latter is a multi-modal posterior with $r(\mathbf{x}) = \max_{\{i=0,2,4,6,8\}} \log p(c = i \mid \mathbf{x})$ for even digits and similarly for odd. For CIFAR-10, we sample from individual classes.

We compare the performance of DTS with DPS (Chung et al., 2023), SMC/FK (Singhal et al., 2025), TDS (Wu et al., 2023a) and DAS (Kim et al., 2025). DPS, TDS, and DAS use reward gradients, while SMC/FK is derivative-free. We report Fréchet Inception Distance (FID) – that compare generated samples with ground truth samples from the dataset. Additional metrics are in Appendix J.1.

Results. Figure 3 shows that DTS achieves very low FID across different NFEs and has better scaling properties with more compute compared to some of the baselines. We provide additional results in Appendix J.1 and Table 2. In all three settings, DTS achieves the lowest FID and CMMD by a considerable margin, indicating it closely matches the true posterior. TDS and SMC in particular show characteristics of mode collapse with very high average rewards and low diversity, whereas DPS often generates samples outside the support of the base model.

4.2. Text-to-image generation

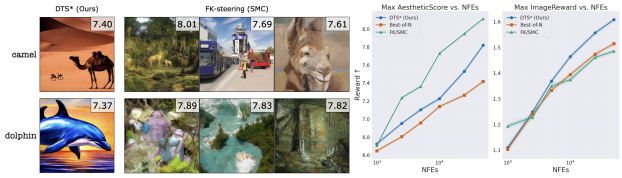

Figure 4: Samples generated using SD-v1.5 for simple animal prompts and Aesthetic Score as the reward at $50k$ NFEs. (Left) For each prompt, DTS* faithfully matches the prompt while achieving high reward, whereas SMC samples score higher but visibly over-optimize. Numbers in the corner show aesthetic scores. (Right) Maximum aesthetic score vs. compute (NFEs) per prompt, averaged over 45 common animal prompts, and maximum ImageReward vs. compute (NFEs) per prompts, averaged over 200 prompts from DrawBench. Additional samples in Appendix J.2.

Setting. We use Stable Diffusion v1.5 (Rombach et al., 2022), a latent diffusion model, as the prior over 512×512 images $\mathbf{x} \sim p_\theta(\mathbf{x} \mid \mathbf{y})$ where $\mathbf{y}$ denotes the text prompt. We evaluate on two different settings: (a) DrawBench (Saharia et al., 2022), comprising of 200 prompts, with ImageReward (Xu et al., 2023) $r(\mathbf{x}, \mathbf{y})$ that encodes prompt accuracy and human preferences; and (b) 45 common animals as prompts following Black et al. (2024), with the LAION aesthetics predictor (Schuhmann, 2022) $r(\mathbf{x})$ that encodes aesthetic quality of an image but does not check for prompt accuracy.

Results. A strong baseline for high-reward generation is *best-of-N*, which draws N samples from the base model and keeps the one with the highest reward. SMC has also been applied to this problem (Singhal et al., 2025), but (a) as discussed in Section 2, it relies on inaccurate value estimates, and (b) Section 4.1 shows that it often collapses onto narrow modes. Because DTS* backs up *soft* values, each node aggregates *posterior mass* rather than peak density. Consequently, a reward spike that lies in a vanishing-probability region of the prior contributes negligibly to the value estimates (cf. Section 2). In our experiments, DTS* delivers higher-reward samples than best-of-N *without* reward over-optimization, as demonstrated in Figures 1 and 4 and scales more favorably with compute.

4.3. Text generation

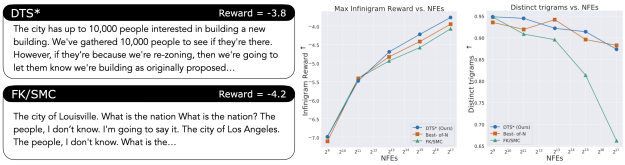

Figure 5: Samples generated by MDLM using infinigram perplexity as reward. (Left) Typical samples generated by DTS* and FK/SMC for the prompt *"The city"* at $131k$ NFEs. FK/SMC seems more prone to reward-hacking by producing repetitive outputs. (Right) Max infinigram reward and distinct trigrams vs. compute (NFEs). DTS* obtains the highest reward while avoiding reward-hacking like FK/SMC. Additional samples in Appendix J.3.

Setting. We evaluate DTS* on text generation using MDLM (Sahoo et al., 2024), a discrete diffusion language model. We generate three text completions of length 64 for each of 15 prompts introduced by Han et al. (2023). As a reward function, we use the perplexity assigned by Infini-gram (Liu et al., 2025), which is more robust against reward hacking than alternative approaches (Singhal et al., 2025). We also report diversity by computing the number of distinct trigrams in each generated sequence. For decoding, we find that using DTS* with max-backup ($\lambda \to \infty$) yields the best performance. We compare our method against two baselines: FK/SMC (Singhal et al., 2025) and best-of-N.

Results. As shown in Figure 5, DTS* consistently achieves the highest rewards as the number of function evaluations (NFEs) increases. Notably, reward functions in text domains are particularly susceptible to over-optimization, often resulting in repetitive outputs, observed with FK/SMC. By contrast, DTS* produces outputs that have both high rewards and high diversity.

[1] https://huggingface.co/google/ddpm-cifar10-32

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J. B., Jaakkola, T. S., and Agrawal, P. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

Bansal, A., Chu, H.-M., Schwarzschild, A., Sengupta, S., Goldblum, M., Geiping, J., and Goldstein, T. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 843–852, 2023.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

Black, K., Janner, M., Du, Y., Kostrikov, I., and Levine, S. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Cardoso, G., Le Corff, S., Moulines, E., et al. Monte carlo guided denoising diffusion models for bayesian linear inverse problems. In *The Twelfth International Conference on Learning Representations*, 2024.

Chen, B., Martí Monsó, D., Du, Y., Simchowitz, M., Tedrake, R., and Sitzmann, V. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

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

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023.

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

Couëtoux, A., Hoock, J.-B., Sokolovska, N., Teytaud, O., and Bonnard, N. Continuous upper confidence trees. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, pp. 433–445. Springer, 2011.

Del Moral, P., Doucet, A., and Jasra, A. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Dou, Z. and Song, Y. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

Efron, B. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

Fan, Y., Watkins, O., Du, Y., Liu, H., Ryu, M., Boutilier, C., Abbeel, P., Ghavamzadeh, M., Lee, K., and Lee, K. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pp. 1352–1361. PMLR, 2017.

Han, X., Kumar, S., and Tsvetkov, Y. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control, 2023. URL https://arxiv.org/abs/2210.17432.

Hansen-Estruch, P., Kostrikov, I., Janner, M., Kuba, J. G., and Levine, S. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, Y., Murata, N., Lai, C.-H., Takida, Y., Uesaka, T., Kim, D., Liao, W.-H., Mitsufuji, Y., Kolter, J. Z., Salakhutdinov, R., et al. Manifold preserving guided diffusion. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Ho, J. and Salimans, T. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pp. 8867–8887. PMLR, 2022.

Jain, V. and Ravanbakhsh, S. Learning to reach goals via diffusion. In *International Conference on Machine Learning*, pp. 21170–21195. PMLR, 2024.

Jain, V., Akhound-Sadegh, T., and Ravanbakhsh, S. Sampling from energy-based policies using diffusion. *arXiv preprint arXiv:2410.01312*, 2024.

Janner, M., Du, Y., Tenenbaum, J., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.

Jayasumana, S., Ramalingam, S., Veit, A., Glasner, D., Chakrabarti, A., and Kumar, S. Rethinking fid: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9307–9315, 2024.

Kang, B., Ma, X., Du, C., Pang, T., and Yan, S. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36: 67195–67212, 2023.

Kim, S., Kim, M., and Park, D. Test-time alignment of diffusion models without reward over-optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Kozakowski, P., Pacek, M., and Miloś, P. Planning and learning using adaptive entropy tree search. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Lee, K., Liu, H., Ryu, M., Watkins, O., Du, Y., Boutilier, C., Abbeel, P., Ghavamzadeh, M., and Gu, S. S. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

Li, X., Zhao, Y., Wang, C., Scalia, G., Eraslan, G., Nair, S., Biancalani, T., Ji, S., Regev, A., Levine, S., et al. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024.

Li, X., Uehara, M., Su, X., Scalia, G., Biancalani, T., Regev, A., Levine, S., and Ji, S. Dynamic search for inference-time alignment in diffusion models. *arXiv preprint arXiv:2503.02039*, 2025.

Liu, J., Min, S., Zettlemoyer, L., Choi, Y., and Hajishirzi, H. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens, 2025. URL https://arxiv.org/abs/2401.17377.

Lu, C., Chen, H., Chen, J., Su, H., Li, C., and Zhu, J. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pp. 22825–22855. PMLR, 2023.

Ma, N., Tong, S., Jia, H., Hu, H., Su, Y.-C., Zhang, M., Yang, X., Li, Y., Jaakkola, T., Jia, X., et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.

Minsky, M. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

Morozov, N., Tiapkin, D., Samsonov, S., Naumov, A., and Vetrov, D. Improving gflownets with monte carlo tree search. *arXiv preprint arXiv:2406.13655*, 2024.

Painter, M., Baioumy, M., Hawes, N., and Lacerda, B. Monte carlo tree search with boltzmann exploration. *Advances in Neural Information Processing Systems*, 36: 78181–78192, 2023.

Prabhudesai, M., Goyal, A., Pathak, D., and Fragkiadaki, K. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739*, 2023.

Psenka, M., Escontrela, A., Abbeel, P., and Ma, Y. Learning a diffusion model policy from rewards via q-score matching. In *International Conference on Machine Learning*, pp. 41163–41182. PMLR, 2024.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36: 53728–53741, 2023.

Ren, A. Z., Lidard, J., Ankile, L. L., Simeonov, A., Agrawal, P., Majumdar, A., Burchfiel, B., Dai, H., and Simchowitz, M. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

Reuss, M., Li, M., Jia, X., and Lioutikov, R. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

Robert, C. P., Doucet, A., and Godsill, S. J. Marginal map estimation using markov chain monte carlo. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 3, pp. 1753–1756. IEEE, 1999.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. 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.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35: 36479–36494, 2022.

Sahoo, S. S., Arriola, M., Schiff, Y., Gokaslan, A., Marroquin, E., Chiu, J. T., Rush, A., and Kuleshov, V. Simple and effective masked diffusion language models, 2024. URL https://arxiv.org/abs/2406.07524.

Schuhmann, C. Laion aesthetics, 2022. URL https://laion.ai/blog/laion-aesthetics/.

Singhal, R., Horvitz, Z., Teehan, R., Ren, M., Yu, Z., McKeown, K., and Ranganath, R. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *The Eighth International Conference on Learning Representations*, 2020.

Song, J., Zhang, Q., Yin, H., Mardani, M., Liu, M.-Y., Kautz, J., Chen, Y., and Vahdat, A. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pp. 32483–32498. PMLR, 2023.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Trippe, B. L., Yim, J., Tischer, D., Baker, D., Broderick, T., Barzilay, R., and Jaakkola, T. S. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *The Eleventh International Conference on Learning Representations*, 2023.

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

Uehara, M., Zhao, Y., Wang, C., Li, X., Regev, A., Levine, S., and Biancalani, T. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review. *arXiv preprint arXiv:2501.09685*, 2025.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Venkatraman, S., Jain, M., Scimeca, L., Kim, M., Sendera, M., Hasan, M., Rowe, L., Mittal, S., Lemos, P., Bengio, E., et al. Amortizing intractable inference in diffusion models for vision, language, and control. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Wu, L., Trippe, B., Naesseth, C., Blei, D., and Cunningham, J. P. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023a.

Wu, X., Sun, K., Zhu, F., Zhao, R., and Li, H. Human preference score: Better aligning text-to-image models with human preference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2096–2105, 2023b.

Xiao, C., Huang, R., Mei, J., Schuurmans, D., and Müller, M. Maximum entropy monte-carlo planning. *Advances in Neural Information Processing Systems*, 32, 2019.

Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., and Dong, Y. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36: 15903–15935, 2023.

Yang, L., Huang, Z., Lei, F., Zhong, Y., Yang, Y., Fang, C., Wen, S., Zhou, B., and Lin, Z. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

Yoon, J., Cho, H., Baek, D., Bengio, Y., and Ahn, S. Monte carlo tree diffusion for system 2 planning. *arXiv preprint arXiv:2502.07202*, 2025.

Zhang, T., Pan, J.-S., Feng, R., and Wu, T. T-scend: Test-time scalable mcts-enhanced diffusion model. *arXiv preprint arXiv:2502.01989*, 2025.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A. Related work

Gradient-based guidance. One way to sample from the optimal policy π^* is to use the first-order Taylor expansion of V_{t-1} around the pretrained mean $\mu_\theta(\mathbf{x}_t, t)$. This yields the gradient-based denoising step $\tilde{\mathbf{x}}_{t-1} \sim \mathcal{N}\left(\mu_\theta(\mathbf{x}_t, t) + \lambda\, \sigma_t^2\, \nabla_{\mathbf{x}} V_{t-1}(\mathbf{x}_{t-1}), \sigma_t^2\, I\right)$. This can be considered a form of classifier guidance (Dhariwal & Nichol, 2021) and is used in many proposed inference-time steering methods (Chung et al., 2023; Bansal et al., 2023; He et al., 2024). The gradient approximation can be improved by using Monte Carlo samples for estimation (Song et al., 2023).

Sequential Monte Carlo. Particle-based methods are another very popular approach, where a population of samples is maintained to approximately sample from the desired distribution. Sequential Monte Carlo (SMC) (Del Moral et al., 2006) uses potential functions, which usually approximate the soft value function, to assign weights to particles and resample them at every step. Different variations of SMC have been proposed for diffusion model alignment (Wu et al., 2023a; Trippe et al., 2023; Cardoso et al., 2024; Dou & Song, 2024; Kim et al., 2025). Classical SMC guarantees exact sampling in the limit of infinite particles and exact value estimation. In practice, however, the repeated sampling procedure can reduce diversity due to weight variance and inaccurate value estimates.

Search-based methods. Recently, there has been a growing interest in using search-based methods to align diffusion models (Ma et al., 2025). Most of these methods propose doing a local search (Li et al., 2024; 2025) by obtaining multiple denoising candidates at each step and selecting the best one based on their value. More recently, tree search has been combined with best-of-N (Zhang et al., 2025), and an MCTS-based approach (Yoon et al., 2025) has been applied in the specific context of diffusion forcing (Chen et al., 2024) over sequences for planning. However, these methods either do not use an explicit backup mechanism, resulting in a limited local search(Li et al., 2024; 2025; Zhang et al., 2025; Ma et al., 2025), or they rely on inaccurate value estimates (Yoon et al., 2025). DTS , on the other hand, performs global credit assignment using all trajectories for asymptotically exact sampling.

B. Illustrative experiments in 2D

In this section, we perform experiments on simple 2D settings to answer the following questions:

- Does DTS sample accurately from the target distribution?
- Does reward backup in DTS result in more accurate value estimates (desideratum D1)?
- Does sample quality of DTS improve with more inference-time compute (desideratum D2)?

Setting. We compare against several inference-time steering methods including some which were originally proposed for posterior sampling in inverse problems, and adapt them to the reward-guidance setting: (1) *DPS-RG* (our reward-guided version of DPS (Chung et al., 2023)) = gradient-based guidance only, (2) *SMC* (Singhal et al., 2025), (3) *TDS-RG* (reward-guided version of TDS (Wu et al., 2023a)) = SMC + gradient guidance, (4) *DAS* (Kim et al., 2025) = SMC + gradient guidance + tempering. We also implement a version of SMC, which we call *SMC-Rollout*, where the values are estimated via one full DDIM (Song et al., 2020) rollout. For fair comparison, we benchmark all methods with respect to number of function evaluations (NFEs) of the diffusion model.

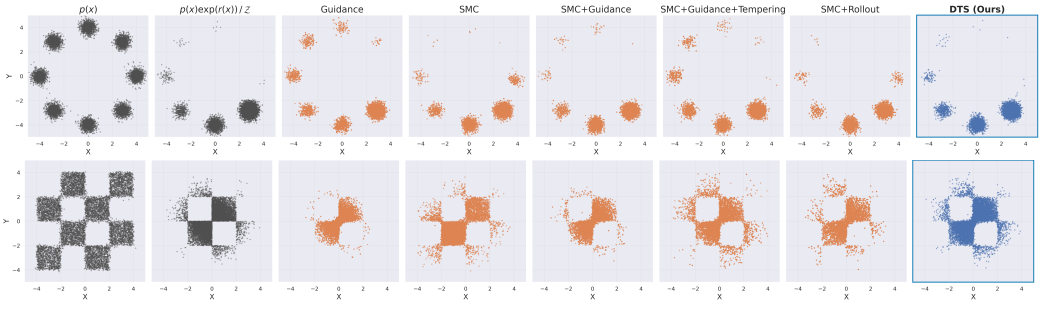

Figure 6: Samples from the prior $p(\mathbf{x}_0)$, target $p(\mathbf{x}_0)\exp(r(\mathbf{x}_0))\,/\,\mathcal{Z}$ and different sampling methods at 10^6 NFEs. (Top) The prior is an equal-weighted mixture of Gaussians, and the reward function distributes mass unevenly. (Bottom) The prior has support on alternate square regions in a checkerboard pattern, and the reward function $r(x, y) = -0.5(x^2 + y^2)$ is negative distance from the origin.

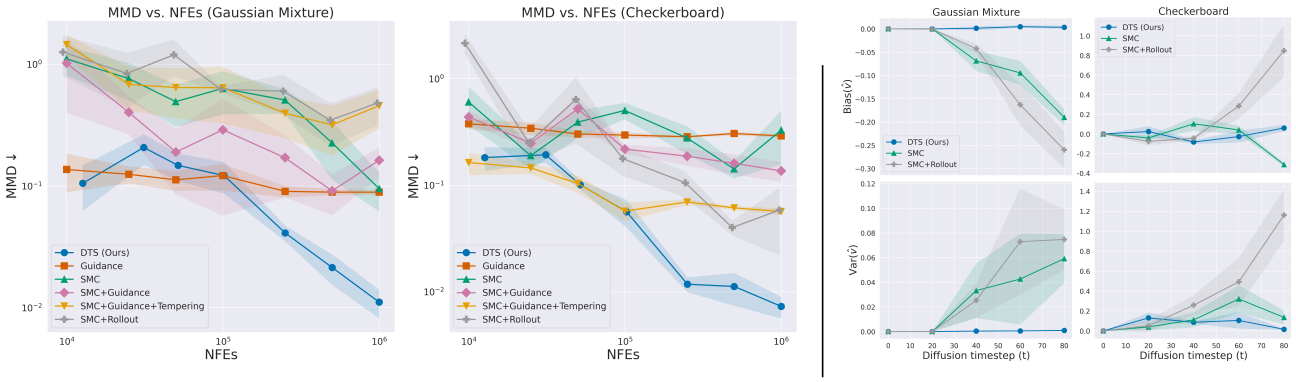

Figure 7: Maximum mean discrepancy (MMD) between generated samples and target ground truth samples as a function of number of function evaluations of the prior diffusion model (left). Bias and variance of value estimates for different approaches (right).

Results. Figure 6 plots the samples obtained using different methods for two different settings. In both cases, DTS approximates the ground truth target density more accurately, with other methods distributing mass inaccurately to different areas of the support. In particular, gradient-based methods like DPS-RG and TDS-RG suffer from instability and require gradient clipping to stabilize denoising steps. We present the plots of maximum mean discrepancy (MMD), which is a kernel-based statistical test used to determine whether two distributions are the same. From Figure 7, we observe that DTS scales more efficiently compared to other methods as the number of function evaluations (NFEs) of the pre-trained diffusion model increases. This empirically validates that the sample quality of DTS improves with more compute, satisfying desideratum D2.

Bias-variance analysis of value estimates. We estimate ground truth soft value estimates at different timesteps by performing 1000 rollouts from noisy states using the base model and then taking log-sum-exp of the rewards. We then compute the relative mean squared error with the value estimates obtained using different approximations and decompose it into bias and variance. Figure 7 shows this for different diffusion timesteps using DTS , Tweedie's formula (SMC + variants), and a single full DDIM rollout (SMC-Rollout). We see that backing up reward information greatly reduces the bias and variance, especially for higher timesteps.

C. Sequential Monte Carlo for diffusion sampling

Many existing methods for inference-time diffusion alignment (Wu et al., 2023a; Trippe et al., 2023; Cardoso et al., 2024; Dou & Song, 2024; Kim et al., 2025) apply sequential Monte Carlo (SMC) (Del Moral et al., 2006) to the reverse diffusion chain. SMC maintains a population of K particles to approximately sample from a sequence of intermediate targets $\{\pi_t(\mathbf{x}_{t:T})\}_{t=T}^{0}$, culminating in the desired $\pi^*(\mathbf{x}_0) \propto p_\theta(\mathbf{x}_0)\exp(\lambda r(\mathbf{x}_0))$. In diffusion alignment, one usually sets

$$\pi_t(\mathbf{x}_{t:T}) \propto p(\mathbf{x}_T) \prod_{s=t+1}^{T} p_\theta(\mathbf{x}_{s-1} \mid \mathbf{x}_s) \exp\left(\lambda\, \hat{v}_t(\mathbf{x}_t)\right), \tag{3}$$

where $\hat{v}_t$ is a *potential* approximating the soft value V_t. Each SMC iteration for $t = T, T-1, \ldots, 0$ has three steps:

1. **Propagation.** Sample particles $\tilde{\mathbf{x}}_{t-1}^{(k)} \sim q_t(\cdot \mid \mathbf{x}_t^{(k)})$, for $k = 1, \ldots, K$ where q_t is the proposal distribution, often set to be the diffusion transition $p_\theta(\cdot \mid \mathbf{x}_t)$.
2. **Weighting.** Assign importance weights

$$w_{t-1}^{(k)} = \underbrace{\frac{p_\theta(\tilde{\mathbf{x}}_{t-1}^{(k)} \mid \mathbf{x}_t^{(k)})}{q_t(\tilde{\mathbf{x}}_{t-1}^{(k)} \mid \mathbf{x}_t^{(k)})}}_{\text{importance ratio}} \times \exp\left(\lambda\, \hat{v}_{t-1}(\tilde{\mathbf{x}}_{t-1}^{(k)})\right). \tag{4}$$

The first factor corrects for using a proposal and the second tilts weights toward high estimated value.

3. **Resampling.** Resample $\{\tilde{\mathbf{x}}_{t-1}^{(k)}\}_{k=1}^{K}$ proportional to $\{w_{t-1}^{(k)}\}_{k=1}^{K}$ to obtain an equally weighted particle set $\{\mathbf{x}_{t-1}^{(k)}\}_{k=1}^{K}$ for the next iteration.

Classical SMC guarantees that, as $K \to \infty$ and if the potentials are exact, the empirical measure $\sum_k w_0^{(k)} \delta\left(\mathbf{x}_0^{(k)}\right)$ converges to the target distribution π^*, where $\delta(x)$ is the Dirac delta at x. In practice, however, this repeated sampling procedure can reduce the diversity of samples, especially when the weights have high variance. This results in an *effective sample size* which is much lower than K.

Another major issue when applying SMC to diffusion models is that estimating the soft value function V_t is not straightforward and errors in the approximation can lead to inaccurate sampling. The next subsection discusses the *value-estimation problems* in more detail.

D. Connection with Generative Flow Networks

Diffusion Tree Sampling can be viewed as an on-the-fly, non-parametric realization of the ideas behind Generative Flow Network (GFlowNet) (Bengio et al., 2021). Both frameworks ultimately seek to sample from an unnormalised density:

$$\pi^*(x) = \frac{1}{\mathcal{Z}} f(x), \quad \mathcal{Z} = \int f(x)\, dx,$$

but they do so with different machinery and at different points in the learning–inference pipeline. GFlowNets define a probability over complete paths $\tau = (\mathbf{s}_0 \to \cdots \to \mathbf{s}_T = x)$ through

$$P_\theta(\tau) = \prod_{t=1}^{T} P_\theta(\mathbf{s}_t \mid \mathbf{s}_{t-1}),$$

and train the parameters θ so that the *forward flow* leaving every non-terminal state equals the *backward flow* entering it plus injected terminal reward $r(\mathbf{x}) = \log f(\mathbf{x})$. In log form, this constraint is a soft Bellman equation with $F(\mathbf{s})$ the learned log-flow function:

$$F(\mathbf{s}) = \frac{1}{\lambda} \log \sum_{s' \in \text{Child}(s)} P_\theta(\mathbf{s}' \mid \mathbf{s}) \, \exp\left(\lambda F(\mathbf{s}')\right).$$

DTS satisfies the same soft Bellman recursion (cf. Equation (2)), but does so *without* learning parameters. During tree construction, DTS estimates the soft value V_t by Monte-Carlo log-sum-exp backups; selection then samples children proportionally to $\exp(\lambda \hat{v}_{t-1})$, where $\hat{v}_{t-1}$ is the estimated soft value. Repeated roll-outs make the empirical terminal distribution converge to the reward-tilted posterior $\pi^*(\mathbf{x}_0) \propto p_\theta(\mathbf{x}_0) \exp\left(\lambda r(\mathbf{x}_0)\right)$, just as a perfectly trained GFlowNet would.

The key differences between DTS and GFlowNets are summarized below.

- **Proposal.** DTS uses a *fixed*, pretrained diffusion p_θ as a proposal, whereas GFlowNets learn the forward policy P_θ.
- **Learning vs. search.** DTS performs pure inference without updating any parameters, whereas GFlowNets learn the parameters of the sampler to amortize future sampling.
- **Computational regime.** DTS excels when one has a strong prior and large *inference* budget for new rewards; GFlowNets shine when the reward is fixed and repeated queries amortize the *training* cost.

Because DTS is a search procedure, it is ideal for adapting a pretrained diffusion model to different unseen reward functions without retraining. GFlowNets, in contrast, learn a fast parametric sampler for a single reward.

E. Theoretical proofs

E.1. Derivation of soft Bellman equation and optimal policy

We derive the recursive relation satisfied by the soft value function in Section 2 as well as the optimal policy for completeness.

Soft value function. This recursive relation is analogous to the soft Bellman equation in maximum entropy RL (Ziebart et al., 2008; Haarnoja et al., 2017). Starting from the definition of $V_t(\mathbf{x}_t)$:

$$\begin{aligned}
V_t(\mathbf{x}_t) &= \frac{1}{\lambda} \log \mathbb{E}_{p_\theta(\mathbf{x}_{0:t-1}|\mathbf{x}_t)} \left[\exp\left(\lambda r(\mathbf{x}_0)\right)\right] \\
&= \frac{1}{\lambda} \log \int p(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}|\mathbf{x}_t) \exp\left(\lambda r(\mathbf{x}_0)\right) d\mathbf{x}_0 d\mathbf{x}_1 \ldots d\mathbf{x}_{t-1} \\
&= \frac{1}{\lambda} \log \int p(\mathbf{x}_0, \mathbf{x}_1, \ldots \mathbf{x}_{t-2}|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{x}_t) \exp\left(\lambda r(\mathbf{x}_0)\right) d\mathbf{x}_0 d\mathbf{x}_1 \ldots d\mathbf{x}_{t-1} \\
&= \frac{1}{\lambda} \log \int p(\mathbf{x}_{t-1}|\mathbf{x}_t) \underbrace{\left(\int p(\mathbf{x}_0, \mathbf{x}_1, \ldots \mathbf{x}_{t-2}|\mathbf{x}_{t-1}) \exp\left(\lambda r(\mathbf{x}_0)\right) d\mathbf{x}_0 d\mathbf{x}_1 \ldots d\mathbf{x}_{t-2} \right)}_{= \exp(\lambda V(\mathbf{x}_{t-1}))} d\mathbf{x}_{t-1} \\
&= \frac{1}{\lambda} \log \int p(\mathbf{x}_{t-1}|\mathbf{x}_t) \exp\left(\lambda V(\mathbf{x}_{t-1})\right) d\mathbf{x}_{t-1} = \frac{1}{\lambda} \log \mathbb{E}_{p(\mathbf{x}_{t-1}|\mathbf{x}_t)} \left[\exp\left(\lambda V(\mathbf{x}_{t-1})\right)\right].
\end{aligned}$$

The above relation combined with the terminal condition $V_0(\mathbf{x}_0) = r(\mathbf{x}_0)$ gives Equation (2).

Optimal policy. The joint target density over the full chain $(\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t)$ is given by:

$$\pi^*(\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t) = \frac{1}{\mathcal{Z}}\, p_\theta(\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t)\, \exp\left(\lambda r(\mathbf{x}_0)\right),$$

where $\mathcal{Z}$ represent the normalization constant of this joint density.

The marginal joint density of $(\mathbf{x}_t, \mathbf{x}_{t-1})$ under π^* is:

$$\begin{aligned}
\pi^*(\mathbf{x}_t, \mathbf{x}_{t-1}) &= \frac{1}{\mathcal{Z}} \int p_\theta(\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t) \exp\left(\lambda r(\mathbf{x}_0)\right) d\mathbf{x}_0 \ldots d\mathbf{x}_{t-2} \\
&= \frac{1}{\mathcal{Z}}\, p_\theta(\mathbf{x}_t)\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \left(\int p_\theta(\mathbf{x}_0, \ldots, \mathbf{x}_{t-2} \mid \mathbf{x}_{t-1}) \exp\left(\lambda r(\mathbf{x}_0)\right) d\mathbf{x}_0 \ldots d\mathbf{x}_{t-2} \right) \\
&= \frac{1}{\mathcal{Z}}\, p_\theta(\mathbf{x}_t)\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \exp\left(\lambda V(\mathbf{x}_{t-1})\right)
\end{aligned}$$

Similarly, the marginal density of $\mathbf{x}_t$ under π^* is:

$$\pi^*(\mathbf{x}_t) = \frac{1}{\mathcal{Z}}\, p_\theta(\mathbf{x}_t) \exp\left(\lambda V(\mathbf{x}_t)\right)$$

By dividing these two marginals, we get the transitions under the optimal policy:

$$\begin{aligned}
\pi^*(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \frac{\pi^*(\mathbf{x}_t, \mathbf{x}_{t-1})}{\pi^*(\mathbf{x}_t)} &= \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \exp\left(\lambda V(\mathbf{x}_{t-1})\right)}{\exp\left(\lambda V(\mathbf{x}_t)\right)} \\
&= \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \exp\left(\lambda V_{t-1}(\mathbf{x}_{t-1})\right)}{\int p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \exp\left(\lambda V_{t-1}(\mathbf{x}_{t-1})\right) d\mathbf{x}_{t-1}}.
\end{aligned} \tag{5}$$

The above relation gives the optimal policy.

E.2. Proof of Proposition 3.1

Proposition E.1 (Asymptotic consistency)**.** *Let r be bounded and $\lambda > 0$, then* DTS *produces a sequence of terminal states whose empirical distribution converges to the optimal policy π^* as the number of tree iterations $M \to \infty$.*

Proof. We use $p(\cdot \mid \mathbf{x}_t)$ to denote a general proposal distribution. For application to diffusion alignment, this would correspond to transitions under the pretrained model $p_\theta(\cdot \mid \mathbf{x}_t)$. Additionally, we use $\hat{q}(\cdot \mid \mathbf{x}_t)$ to denote the transition density of DTS .

Step 1: Transition probability under DTS . Recall that under DTS , given a node $\mathbf{x}_t$, we create each child by sampling from the base model $p(\cdot \mid \mathbf{x}_t)$. During tree traversal, we select the next state $\mathbf{x}_{t-1}$ proportional to the exponentiated soft value function. Thus, the transition probability of DTS from $\mathbf{x}_t$ to $\mathbf{x}_{t-1}$ is given by:

$$\hat{q}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \frac{p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \exp\left(\lambda \hat{v}(\mathbf{x}_{t-1})\right)}{\int p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \exp\left(\lambda \hat{v}(\mathbf{x}_{t-1})\right)\, d\mathbf{x}_{t-1}} = \frac{p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \exp\left(\lambda \hat{v}(\mathbf{x}_{t-1})\right)}{\exp\left(\lambda \hat{v}(\mathbf{x}_t)\right)}, \tag{6}$$

where the second equality follows from the definition of the soft Bellman equation:

$$\hat{v}(\mathbf{x}_t) = \frac{1}{\lambda} \log \mathbb{E}_{\mathbf{x}_{t-1} \sim p(\cdot \mid \mathbf{x}_t)} \left[\exp\left(\lambda \hat{v}(\mathbf{x}_{t-1})\right)\right].$$

Step 2: Joint density of trajectory. Recall that the root node of DTS contains a dummy state $\mathbf{x}_{T+1}$ that transitions to the diffusion process prior $\hat{q}(\mathbf{x}_T \mid \mathbf{x}_{T+1}) = \mathcal{N}(0, I)$. Then, the joint density of a full trajectory $\{\mathbf{x}_T, \mathbf{x}_{T-1}, \ldots, \mathbf{x}_0\}$ under DTS is given by:

$$\begin{aligned}
\hat{q}(\mathbf{x}_T, \mathbf{x}_{T-1}, \ldots, \mathbf{x}_0) &= \prod_{t=1}^{T+1} \hat{q}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \prod_{t=1}^{T+1} \frac{p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \exp\left(\lambda \hat{v}(\mathbf{x}_{t-1})\right)}{\exp\left(\lambda \hat{v}(\mathbf{x}_t)\right)} \\
&= \frac{\exp\left(\lambda \hat{v}(\mathbf{x}_0)\right)}{\exp\left(\lambda \hat{v}(\mathbf{x}_{T+1})\right)} \prod_{t=1}^{T+1} p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \\
&= \frac{\exp\left(\lambda \hat{v}(\mathbf{x}_0)\right)}{\exp\left(\lambda \hat{v}(\mathbf{x}_{T+1})\right)} p(\mathbf{x}_T, \mathbf{x}_{T-1}, \ldots, \mathbf{x}_0).
\end{aligned}$$

Step 3: Marginalizing. Marginalizing over intermediate states $\mathbf{x}_1, \ldots, \mathbf{x}_T$, we get the distribution of terminal state $\mathbf{x}_0$:

$$\begin{aligned}
\hat{q}(\mathbf{x}_0) &= \int \hat{q}(\mathbf{x}_T, \mathbf{x}_{T-1}, \ldots, \mathbf{x}_0)\, d\mathbf{x}_T d\mathbf{x}_{T-1} \ldots d\mathbf{x}_1 \\
&= \frac{\exp\left(\lambda \hat{v}(\mathbf{x}_0)\right)}{\exp\left(\lambda \hat{v}(\mathbf{x}_{T+1})\right)} \int p(\mathbf{x}_T, \mathbf{x}_{T-1}, \ldots, \mathbf{x}_0)\, d\mathbf{x}_T d\mathbf{x}_{T-1} \ldots d\mathbf{x}_1 \\
&= \frac{\exp\left(\lambda \hat{v}(\mathbf{x}_0)\right)}{\exp\left(\lambda \hat{v}(\mathbf{x}_{T+1})\right)} p(\mathbf{x}_0).
\end{aligned}$$

By definition, the soft value function at the terminal node is $\hat{v}(\mathbf{x}_0) = r(\mathbf{x}_0)$. Plugging this and using the definition of value function from Equation (2), we have:

$$\begin{aligned}
\hat{q}(\mathbf{x}_0) &= \frac{\exp\left(\lambda r(\mathbf{x}_0)\right) p(\mathbf{x}_0)}{\int p(\mathbf{x}_T, \mathbf{x}_{T-1}, \ldots, \mathbf{x}_0 \mid \mathbf{x}_{T+1}) \exp\left(\lambda r(\mathbf{x}_0)\right)\, d\mathbf{x}_T d\mathbf{x}_{T-1} \ldots d\mathbf{x}_1 d\mathbf{x}_0} \\
&= \frac{\exp\left(\lambda r(\mathbf{x}_0)\right) p(\mathbf{x}_0)}{\int p(\mathbf{x}_0) \exp\left(\lambda r(\mathbf{x}_0)\right)\, d\mathbf{x}_0}.
\end{aligned}$$

This has the form of the target distribution in Equation (5), except that it uses the value estimates $\hat{v}$ that are calculated based on rollouts starting from each state $\mathbf{x}_t$. In the limit of infinite rollouts, these value estimates approach the true soft values, confirming that the sampling distribution $\hat{q}$ from DTS exactly matches the target distribution π^*.

Therefore, DTS is consistent, as it correctly generates samples from the desired target distribution asymptotically.

□

F. DTS and DTS* Algorithm

Algorithm 1 Diffusion Tree Sampling (DTS) and Diffusion Tree Search (DTS*)

Input: base policy p_θ, reward function r, number of iterations M, inverse temperature λ, parameters $C, \alpha, c_{\text{uct}}$
Initialize root node $\mathbf{x}_{T+1}$ with dummy value, $\hat{v}(\mathbf{x}_{T+1}) = 0$, $N(\mathbf{x}_{T+1}) = 1$
Initialize tree $\mathcal{T}$ with root node $\mathbf{x}_{T+1}$
for $m = 1, \dots, M$ **do**
  $P \leftarrow \{\mathbf{x}_{T+1}\}$
  Set $t \leftarrow T + 1$
  `// Selection`
  while $|\mathcal{C}(\mathbf{x}_t)| \geq C \cdot N(\mathbf{x}_t)^\alpha$ and $t > 0$ **do**
    [DTS] select child probabilistically: $\mathbf{x}_{t-1} \sim \frac{\exp(\lambda \hat{v}(\mathbf{x}_{t-1}))}{\sum_{\mathbf{x}' \in \mathcal{C}(\mathbf{x}_t)} \exp(\lambda \hat{v}(\mathbf{x}'))}$
    [DTS*] select child maximizing UCT: $\mathbf{x}_{t-1} = \arg\max_{\mathbf{x}' \in \mathcal{C}(\mathbf{x}_t)} \hat{v}(\mathbf{x}') + c_{\text{uct}} \sqrt{\frac{\log N(\mathbf{x}_t)}{N(\mathbf{x}')}}$
    $P \leftarrow P \cup \{\mathbf{x}_{t-1}\}$
    $t \leftarrow t - 1$
  end while
  `// Expansion: expand` $\mathbf{x}_t$ `by sampling a new child`
  if $t > 0$, and $|\mathcal{C}(\mathbf{x}_t)| < C \cdot N(\mathbf{x}_t)^\alpha$ **then**
    `// Rollout: from new node` $\mathbf{x}_{t-1}$ `sample rollout path to terminal` $\mathbf{x}_0$
    while $t > 0$ **do**
      $\mathbf{x}_{t-1} \sim p_\theta(\cdot \mid \mathbf{x}_t), \quad \hat{v}(\mathbf{x}_{t-1}) = 0, \quad N(\mathbf{x}_{t-1}) = 1$
      $P \leftarrow P \cup \{\mathbf{x}_{t-1}\}$
      $t \leftarrow t - 1$
    end while
  end if
  Evaluate terminal reward: $\hat{v}(\mathbf{x}_0) = r(\mathbf{x}_0)$
  `// Backup: update value along path` P
  for $t = 0, \dots, T$ **do**
    Soft backup: $\hat{v}(\mathbf{x}_{t+1}) \leftarrow \frac{1}{\lambda} \log \sum_{\mathbf{x}_t \in \mathcal{C}(\mathbf{x}_{t+1})} \exp(\lambda \hat{v}(\mathbf{x}_t))$
    Update visits: $N(\mathbf{x}_{t+1}) \leftarrow N(\mathbf{x}_{t+1}) + 1$
  end for
end for
return $\mathcal{T}$

G. Design choices for DTS and DTS*

The algorithm discussed above can be applied to any Markov chain. However, in this work, we apply it to the problem of inference-time alignment of diffusion models. We discuss various considerations and design choices below, with more implementation details in Appendix F.

Sampling or Search. DTS is designed to sample from the target distribution π^*, but for settings where a single high-quality sample is required, we introduce a *search* variant, DTS* . It keeps the same soft value backup but modifies the selection step by always selecting the child with the largest soft value estimate instead of Boltzmann sampling. Since DTS* uses soft values, this is different from standard MCTS – it implements a *marginal-MAP* (max–sum) inference scheme (Robert et al., 1999) over the tree. At every noise step, the algorithm selects the branch whose subtree carries the greatest mass under $\pi^\star$ and, once $t = 0$ is reached, returns the highest-density leaf inside that dominant region. As we will see in the Section 4, this volume-based selection helps avoid reward over-optimization.

Branching. Extensions of MCTS to continuous spaces commonly use *progressive widening* (Couëtoux et al., 2011) to decide the maximum number of branches $B(\mathbf{x}_t)$ allowed per node based on the number of visits: $B(\mathbf{x}_t) = C \cdot N(\mathbf{x}_t)^\alpha,\ C > 0,\ \alpha \in (0, 1)$. The high-level intuition is that nodes that are visited more often should be expanded more, since they represent more promising directions for denoising. We adopt the same strategy and during tree traversal, if we encounter a node such that $|\mathcal{C}(\mathbf{x}_t)| < B(\mathbf{x}_t)$ and $t > 0$, we will always expand.

Exploration. There is a rich literature on search methods for classical MCTS, and the most popular approach, UCT (Kocsis & Szepesvári, 2006), is an application of upper-confidence bounds (Auer et al., 2002) to trees. We employ this exploration strategy for DTS* , i.e. we choose the child $\mathbf{x}_{t-1} \in \mathcal{C}(\mathbf{x}_t)$ with the maximum value of the UCT estimate:

$$\mathrm{UCT}(\mathbf{x}_{t-1}) = \hat{v}(\mathbf{x}_{t-1}) + c_{\text{uct}} \sqrt{\frac{\log N(\mathbf{x}_t)}{N(\mathbf{x}_{t-1})}}, \qquad c_{\text{uct}} > 0. \tag{7}$$

For DTS, we do not employ explicit exploration, because, in practice we observe that sampling obviates the need for an exploration bonus or handcrafted mechanism.

Efficient implementation. The main computational cost is incurred when using the diffusion model proposal to sample new children or perform rollouts. We implement an efficient batched version of the algorithm by collecting nodes by timestep and performing one forward pass for all nodes at the same timestep. The selection and backup steps involve simple tensor operations and pointer manipulation with negligible cost. Therefore, while the control flow of our method is sequential, the practical algorithm can be parallelized. Note that once the tree has been built, sampling is near instantaneous by repeatedly selecting children without any model calls.

H. Implementation details for DTS and DTS*

H.1. Tree structure

The algorithm presented in Section 3.2 and Appendix F allows every state $\mathbf{x}_t$ along the denoising trajectory to be considered for branching. However, in practice, we only branch every few timesteps. We noticed very little difference in performance between the two cases for the same number of function evaluations, however, we expect branching at every step to outperform for a very high compute budget. We match the tree branching schedule with the resampling schedule for all baselines with SMC, similar to the setting from Singhal et al. (2025). The exact setting for each experiment is presented in Table 1, where we always branch at the root node corresponding to $t = T$.

Table 1: Branching schedule for DTS and DTS* , which is also the resampling schedule used for SMC-based methods – SMC/FK (Singhal et al., 2025), TDS (Wu et al., 2023a), DAS (Kim et al., 2025).

Domain	Total denoising steps	Branching schedule
Two-dimensional	100	$100(\text{root}), 80, 60, 40, 20$
Image pixels (MNIST, CIFAR-10)	50	$50(\text{root}), 40, 30, 20, 10$
Image latents (SD-v1.5)	50	$50(\text{root}), 40, 30, 20, 10$
Text tokens (MDLM)	256	$256(\text{root}), 216, 176, 136, 96, 56$

Apart from this, we have hyperparameters associated with progressive widening that control the maximum number of branches at any node. We used $\alpha = 0.8$ and $C = 2$ for all two-dimensional and image experiments and $\alpha = 0.7$ and $C = 2$ for text generation. There is a scope of improving the performance of DTS and DTS* further by tuning these parameters for specific tasks.

H.2. Experiment details

Illustrative 2D

Base diffusion model: The denoising network is an MLP that takes as input the 2-dimensional data $\mathbf{x}_t$ and the timestep t and outputs a 2-dimensional noise prediction. The timestep is transformed using sinusoidal embeddings (Vaswani et al., 2017). The network has four hidden layers of 128 dimension each with the sigmoid linear unit (SiLU, (Hendrycks & Gimpel, 2016)) activation. We used the linear noise schedule with $\beta_{\text{min}} = 0.001$ and $\beta_{\text{max}} = 0.07$ and the score matching objective. The optimizer used for training was Adam (Kingma, 2014) with a learning rate of 3×10^{-3}. We train the model for 500 epochs on a training set of 10000 samples.

Reward function: *Gaussian mixture:* The reward function is:

$$r(\mathbf{x}) = \log \left(\sum_{i=1}^{8} w_i \, \exp\left(-\|\mathbf{x} - \boldsymbol{\mu}_i\|^2 / 2\sigma^2\right) \right),$$

where $w_i = \exp(1.5\, i)$, $\boldsymbol{\mu}_i = 4 \left(\cos \frac{2\pi(i-1)}{8}, \, \sin \frac{2\pi(i-1)}{8} \right)$, $i = 1, \ldots, 8$, with $\sigma = 0.3$. *Checkerboard:* The reward is negative distance from the center $r(\mathbf{x}) = -0.5\|\mathbf{x}\|^2$.

Class-conditional MNIST

Base diffusion model: The denoising network is a Unet architecture (Ronneberger et al., 2015) that operates on images of size $32 \times 32 \times 1$ (upscaled from $28 \times 28 \times 1$) with block channels $\{32, 64, 128, 256\}$. We use the DDIMScheduler[a] from diffusers library with default parameters, except we set $\eta = 1.0$ so the inference process is stochastic like DDPMs (Ho et al., 2020). We use the AdamW optimizer with a learning rate of 10^{-4} for 100 epochs on the MNIST training set.

Reward function: We train a classifier $p(c \mid \mathbf{x})$ on the MNIST training set. The classifier is a convolutional neural network (LeCun et al., 2015) using two 5×5 kernels with $(16, 32)$ channels followed by 2×2 max pooling operation with ReLU activations. The features are then flattened and followed by a linear layer with 10 outputs corresponding to the classes. The network was trained using Adam optimizer with learning rate 10^{-3}. The reward function for single class generation is the log likelihood of the class $r_i(\mathbf{x}) = \log p(c = i \mid \mathbf{x})$ for $i \in \{0, 1, \ldots, 9\}$. For the even or odd generation, it is defined as $r(\mathbf{x}) = \max_{i \in \mathcal{S}} \log p(c = i \mid \mathbf{x})$, where $\mathcal{S} = \{0, 2, 4, 6, 8\}$ for even digit generation and $\mathcal{S} = \{1, 3, 5, 7, 9\}$ for odd digit generation.

[a] https://huggingface.co/docs/diffusers/en/api/schedulers/ddim

Class-conditional CIFAR-10

Base diffusion model: We used the pre-trained diffusion model `ddpm-cifar10-32`[a] from Hugging Face, which uses a Unet architecture and diffuses over $32 \times 32 \times 3$ images in pixel-space. We use the DDIMScheduler with $\eta = 1.0$ for stochastic denoising.

Reward function: We train a classifier $p(c \mid \mathbf{x})$ on the CIFAR-10 training set. The classifier uses a ResNet-18 (He et al., 2016) backbone that outputs an embedding which is average pooled, flattened, and passed to a single linear layer with 10 outputs. The network is trained using Adam optimizer with learning rate 10^{-3}. Similar to MNIST single class generation, the reward function is the log likelihood of the class $r_i(\mathbf{x}) = \log p(c = i \mid \mathbf{x})$ for $i \in \{0, 1, \ldots, 9\}$.

[a] https://huggingface.co/google/ddpm-cifar10-32

Text-to-image

Base diffusion model: We use Stable Diffusion v1.5[a] from Hugging Face, which is a latent diffusion model (Rombach et al., 2022). The diffusion process is defined over $64 \times 64 \times 4$ latent variables, which are obtained by encoding $512 \times 512 \times 3$ images using a variational autoencoder. The model uses CLIP (Radford et al., 2021) to encode text prompts into embeddings which are then used to condition the generative process via classifier-free guidance (Ho & Salimans, 2021). We use the DDIMScheduler with $\eta = 1.0$.

Reward function: We use pre-trained models as reward functions including ImageReward[b] $r(\mathbf{x}, \mathbf{y})$ that encodes prompt accuracy as well as human preferences the LAION aesthetic score predictor[c] $r(\mathbf{x})$ that encodes aesthetic quality of an image.

[a] https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5
[b] https://github.com/THUDM/ImageReward
[c] https://github.com/LAION-AI/aesthetic-predictor

Conditional text

Base diffusion model: We use MDLM[a] for our text generation experiments. This is a discrete diffusion model with 110M parameters that directly predicts the tokens. We define the diffusion process over a context length of 64 tokens with 256 sampling steps and use the standard discrete unmasking update for stochastic denoising.

Reward function: We use ∞-gram language model[b]. In particular, we use the perplexity from the `v4_dolmasample_olmo` version trained on the Dolma-v1.6-sample corpus with the OLMo tokenizer. This reward function $r(\mathbf{x})$ encodes the linguistic plausibility of a given string $\mathbf{x}$.

[a] https://huggingface.co/kuleshov-group/mdlm-owt
[b] https://infini-gram.io/

H.3. Compute

We report execution times on a single A100 GPU with 80 gigabytes of memory.

- Each 2D experiment including all methods runs in 15 minutes. Adding up the time over five seeds and two different datasets, the combined run time is approximately 2.5 GPU hours.
- The MNIST and CIFAR-10 class-conditional experiments use approximately 3 GPU hours per class including all methods. Over all 22 tasks (10 MNIST single digit + 2 MNIST even/odd + 10 CIFAR-10 classes) equals approximately 66 GPU hours.
- The text-to-image experiments using Stable Diffusion v1.5 require roughly 30 minutes per prompt across all methods. Adding up all 200 prompts from DrawBench and 45 animal prompts, reproducing all experiments requires approximately 123 GPU hours.
- The text generation experiments using MDLM requires roughly 30 minutes per prompt. Thus, generating 3 completions per prompt for the 15 prompts requires roughly 22.5 GPU hours.

I. Details of baselines

We re-implemented all baseline methods in our unified codebase since most of them use SMC as a backbone and share the same underlying infrastructure. Each implementation was validated by reproducing the quantitative results reported in its original paper. Appendix C provides a concise primer on SMC for reference. The complete source code including all baselines will be released publicly upon publication of this work.

DPS. Diffusion Posterior Sampling (Chung et al., 2023) was originally proposed for noisy inverse problems such as image super-resolution and de-blurring using the gradient of the final objective. To adapt this method for general reward functions, we make a minor modification by replacing the gradient of the inverse problem objective with the gradient of the reward function:

$$\tilde{\mathbf{x}}_{t-1} \sim \mathcal{N}\left(\mu_\theta(\mathbf{x}_t, t) \; + \; \lambda\, \sigma_t^2\, \nabla_{\mathbf{x}_t} r(\hat{\mathbf{x}}_0(\mathbf{x}_t)), \sigma_t^2\, I\right), \tag{8}$$

where $\hat{\mathbf{x}}_0$ is obtained using Tweedie's formula (cf. **??**), μ_θ is the predicted mean of the base diffusion model, and $r(\mathbf{x})$ is the reward function in the two-dimensional experiments and classifier log likelihoods $\log p(c = i \mid \mathbf{x})$ for class-conditional image experiments. The official implementation is provided here.

SMC/FK-Steering. In our paper, SMC refers to the simplest variant, FK-Steering (Singhal et al., 2025), which defines different weighting schemes and uses the pre-trained diffusion model as the proposal distribution. As per the setting in Singhal et al. (2025), we perform the resampling step at fixed intervals during denoising (given in Table 1) and use adaptive resampling to increase diversity of generated samples. Our sampling experiments (two-dimensional and class-conditional image generation) use the `'diff'` potential with $\lambda = 1.0$, whereas the search experiments (text-to-image and text generation) use the `'max'` potential with $\lambda = 10.0$. The weights for resampling are given by Equation (4) where the proposal is equal to the pre-trained diffusion transition and the value estimates are equal to:

$$\hat{v}_{t-1}^{\text{diff}}(\tilde{\mathbf{x}}_{t-1}) = r\left(\hat{\mathbf{x}}_0(\tilde{\mathbf{x}}_{t-1})\right) - r\left(\hat{\mathbf{x}}_0(\mathbf{x}_t)\right), \quad \hat{v}_T^{\text{diff}}(\mathbf{x}_T) = r\left(\hat{\mathbf{x}}_0(\mathbf{x}_T)\right).$$
$$\hat{v}_{t-1}^{\max}(\tilde{\mathbf{x}}_{t-1}) = \max\left\{r(\hat{\mathbf{x}}_0(\tilde{\mathbf{x}}_{t-1})),\, m_t^{(k)}\right\}, \quad m_t^{(k)} = \max_{s \geq t} r\left(\hat{\mathbf{x}}_0(\mathbf{x}_s^{(k)})\right).$$

We adapted the official implementation provided here.

TDS. Twisted Diffusion Sampler (Wu et al., 2023a) comprises of a "twisted" proposal which is used along with SMC to sample from the target posterior distribution. For general reward functions, the twisted proposal is the same as the one used in Equation (8) and the final weights are obtained using Equation (4) after plugging in the twisted proposal and the value estimates:

$$q_t(\tilde{\mathbf{x}}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\tilde{\mathbf{x}}_{t-1}\,;\, \mu_\theta(\mathbf{x}_t, t) \,+\, \lambda\, \sigma_t^2\, \nabla_{\mathbf{x}_t} r(\hat{\mathbf{x}}_0(\mathbf{x}_t)), \sigma_t^2\, I\right).$$
$$\hat{v}_{t-1}(\tilde{\mathbf{x}}_{t-1}) = r\left(\hat{\mathbf{x}}_0(\tilde{\mathbf{x}}_{t-1})\right) - r\left(\hat{\mathbf{x}}_0(\mathbf{x}_t)\right), \quad \hat{v}_T(\mathbf{x}_T) = r\left(\hat{\mathbf{x}}_0(\mathbf{x}_T)\right).$$

The official implementation is provided here.

DAS. Diffusion Alignment as Sampling (Kim et al., 2025) re-uses the twisted proposal of TDS but multiplies the reward term by a monotone tempering schedule $0 = \gamma_T \leq \gamma_{T-1} \leq \ldots \leq \gamma_0 = 1$ to reduce the bias from inaccurate value estimates at high noise levels. The weights are given by Equation (4) after plugging in the tempered proposal and value estimates:

$$q_t(\tilde{\mathbf{x}}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\tilde{\mathbf{x}}_{t-1}\,;\, \mu_\theta(\mathbf{x}_t, t) \,+\, \lambda\, \gamma_{t-1}\, \sigma_t^2\, \nabla_{\mathbf{x}_t} r(\hat{\mathbf{x}}_0(\mathbf{x}_t)), \sigma_t^2\, I\right).$$
$$\hat{v}_{t-1}(\tilde{\mathbf{x}}_{t-1}) = \gamma_{t-1}\, r\left(\hat{\mathbf{x}}_0(\tilde{\mathbf{x}}_{t-1})\right) - \gamma_t\, r\left(\hat{\mathbf{x}}_0(\mathbf{x}_t)\right), \quad \hat{v}_T(\mathbf{x}_T) = \gamma_T\, r\left(\hat{\mathbf{x}}_0(\mathbf{x}_T)\right).$$

The official implementation is provided here.

J. Additional results

J.1. Class-conditional image experiments

We report two distribution-based metrics – Fréchet Inception Distance (FID) and CLIP maximum mean discrepancy (CMMD) (Jayasumana et al., 2024) – that compare generated samples with ground truth samples from the dataset, in addition to average rewards and CLIP diversity (pairwise cosine distance) in Table 2. Figure 8 shows that across the three settings for most values of NFEs, DTS matches the target distribution more accurately compared to other methods (lowest FID and CMMD).

We also present random samples for each method and setting in Figures 9 to 10. We observe the same trend as noticed in Figure 3 – gradient-based guidance like DPS can be unstable leading to unnatural images, while SMC-based methods show signs of mode collapse with low average diversity and high average rewards. DTS balances both diversity and high rewards effectively by closely matching the true posterior distribution.

Table 2: Comparison of inference-time posterior sampling methods. We report the mean$\pm$std of each metric across the relevant classes and highlight $\pm 5\%$ values from the best experimental value.

Dataset →	MNIST				MNIST even/odd				CIFAR-10			
Algorithm ↓	FID (↓)	CMMD (↓)	$\mathbb{E}[\log r(\mathbf{x})]$ (↑)	Diversity (↑)	FID (↓)	CMMD (↓)	$\mathbb{E}[\log r(\mathbf{x})]$ (↑)	Diversity (↑)	FID (↓)	CMMD (↓)	$\mathbb{E}[\log r(\mathbf{x})]$ (↑)	Diversity (↑)
DPS	0.359 $\pm$0.227	0.441 $\pm$0.447	-0.323 $\pm$0.286	0.474 $\pm$0.051	0.123 $\pm$0.031	0.293 $\pm$0.139	-0.002 $\pm$0.001	0.572 $\pm$0.053	0.486 $\pm$0.121	2.609 $\pm$0.824	-0.002 $\pm$0.001	0.551 $\pm$0.024
SMC/FK	0.060 $\pm$0.051	0.177 $\pm$0.142	-0.002 $\pm$0.004	0.422 $\pm$0.040	0.027 $\pm$0.009	0.123 $\pm$0.113	-0.003 $\pm$0.003	0.583 $\pm$0.084	0.313 $\pm$0.070	1.409 $\pm$0.445	-0.102 $\pm$0.093	0.487 $\pm$0.045
TDS	0.087 $\pm$0.035	0.463 $\pm$0.260	-0.001 $\pm$0.001	0.404 $\pm$0.042	0.053 $\pm$0.010	0.250 $\pm$0.056	-0.001 $\pm$0.000	0.576 $\pm$0.124	0.487 $\pm$0.112	2.675 $\pm$0.665	-0.046 $\pm$0.055	0.469 $\pm$0.042
DAS	0.039 $\pm$0.017	0.179 $\pm$0.099	-0.016 $\pm$0.016	0.440 $\pm$0.041	0.031 $\pm$0.002	0.079 $\pm$0.011	-0.015 $\pm$0.019	0.603 $\pm$0.094	0.241 $\pm$0.037	0.822 $\pm$0.203	-0.584 $\pm$0.200	0.530 $\pm$0.023
DTS (ours)	0.014 $\pm$0.005	0.068 $\pm$0.030	-0.023 $\pm$0.006	0.452 $\pm$0.050	0.007 $\pm$0.003	0.036 $\pm$0.029	-0.010 $\pm$0.004	0.597 $\pm$0.069	0.195 $\pm$0.041	0.745 $\pm$0.201	-0.305 $\pm$0.116	0.542 $\pm$0.020

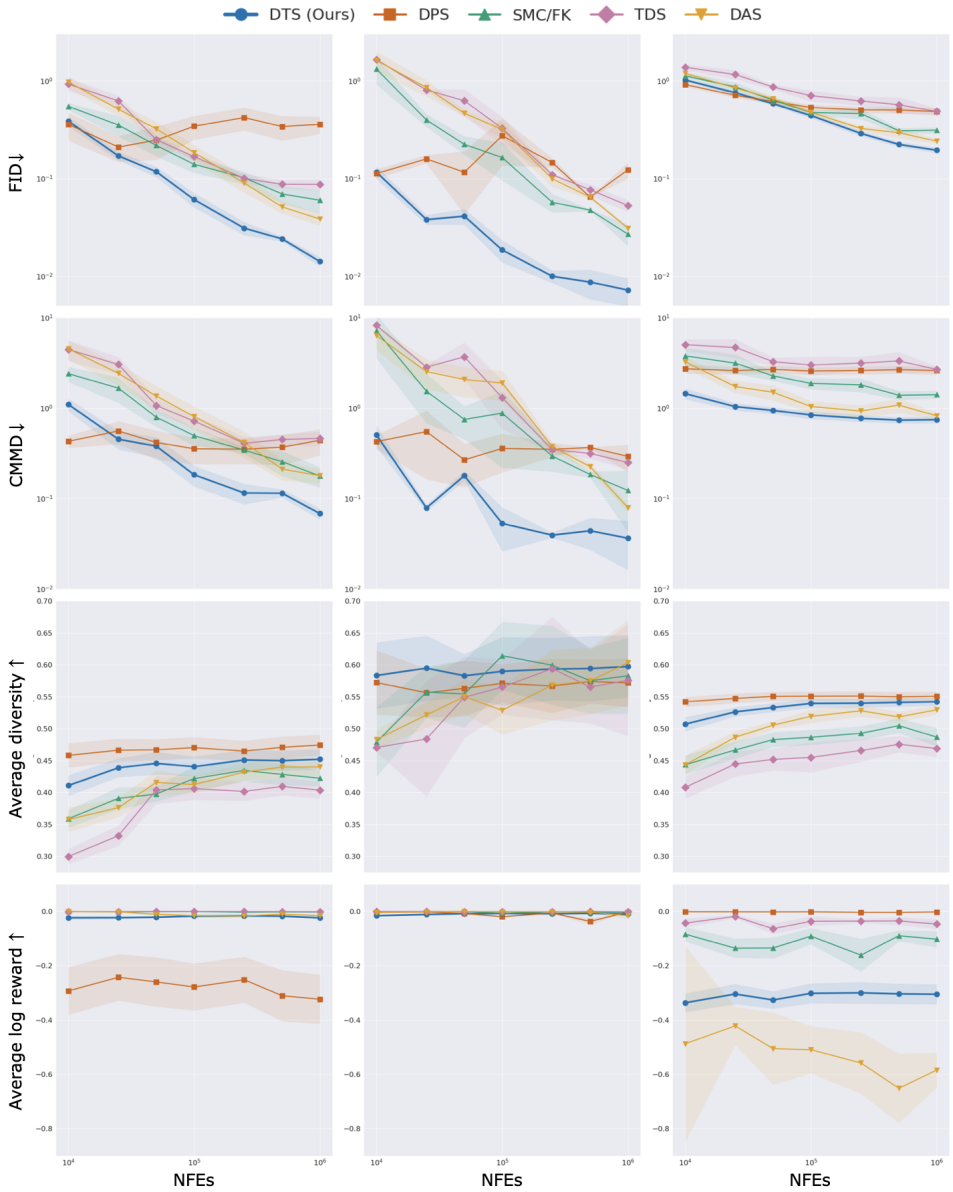

Figure 8: Various distribution level metrics versus number of function evaluations for different methods on MNIST single digit generation averaged over all 10 digits (left), MNIST odd and even digit generation (center), and CIFAR-10 single class generation averaged over all 10 digits (right). All methods were evaluated with 5000 generated samples per class. Metrics reported: FID (lower is better), CMMD (lower is better), average log rewards (higher is better), and average diversity (higher is better).

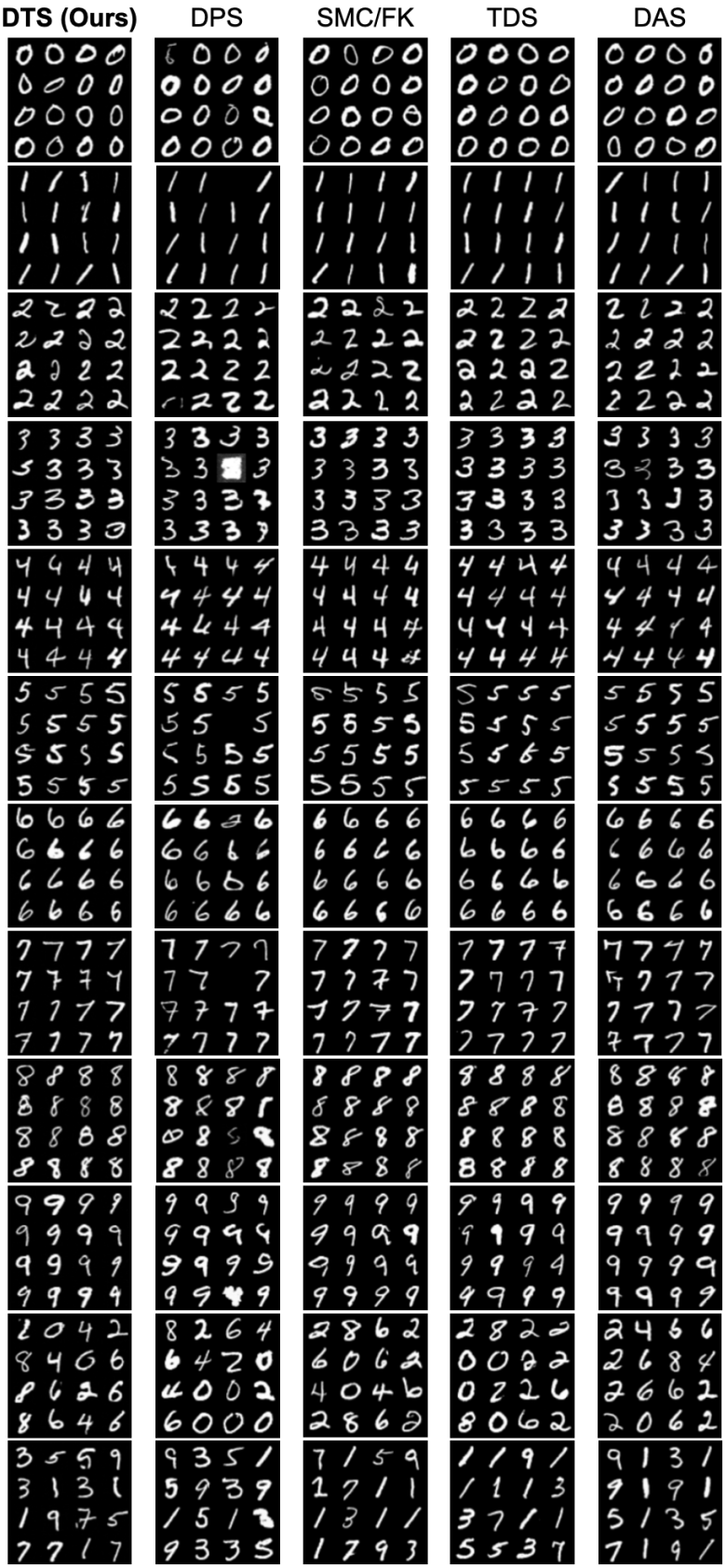

Figure 9: MNIST posterior samples generated using different methods for digits 0-9, even and odd.

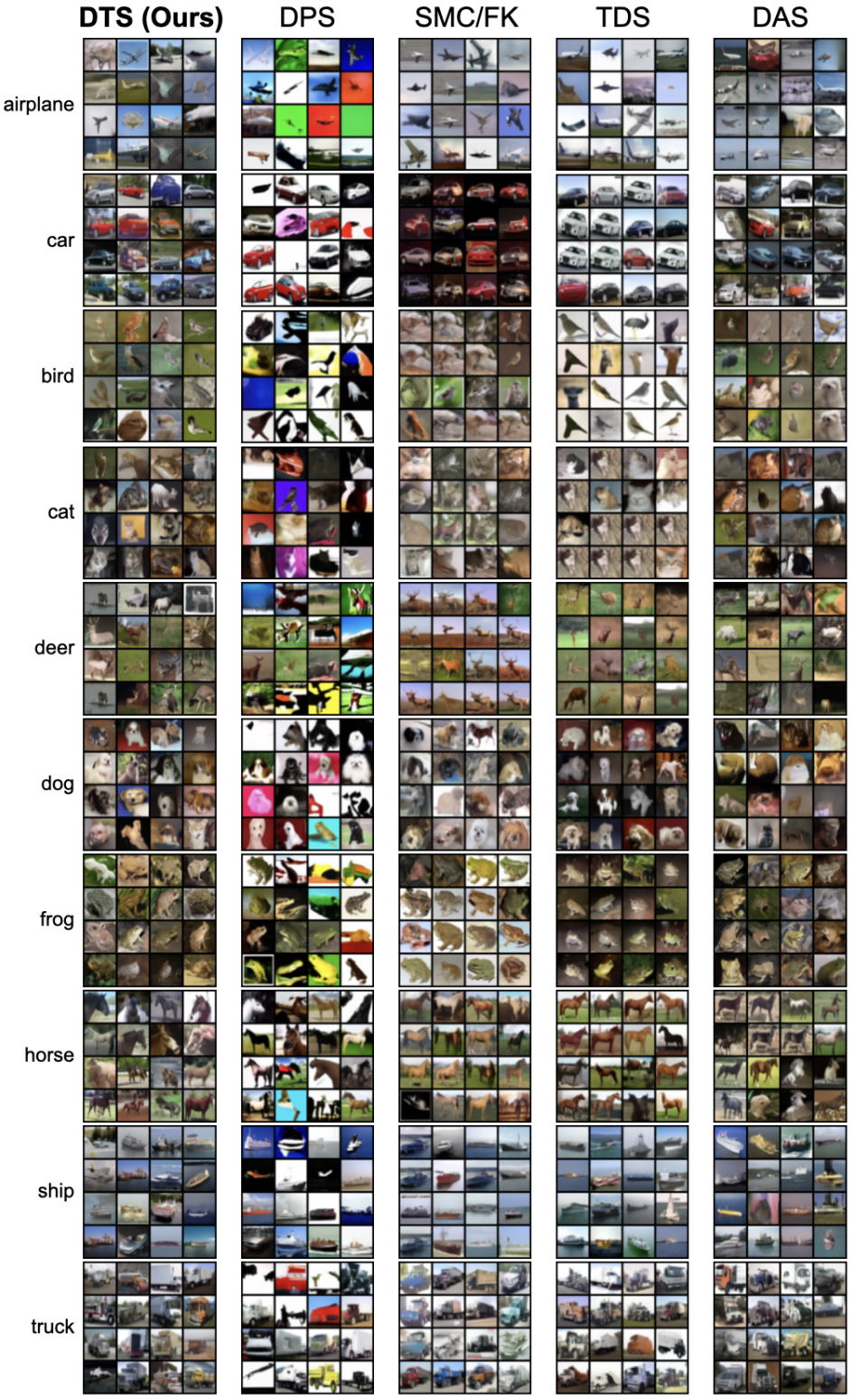

Figure 10: CIFAR-10 posterior samples generated using different methods for all classes.

J.2. Text-to-image examples

We present more samples for qualitative analysis. Figure 11 shows how samples change with increasing amount of inference-time compute, providing visual evidence for the quantitative results from Figure 4. Figures 12 to 14 shows text-image pairs testing different concepts such as artistic style, spatial arrangement and object count.

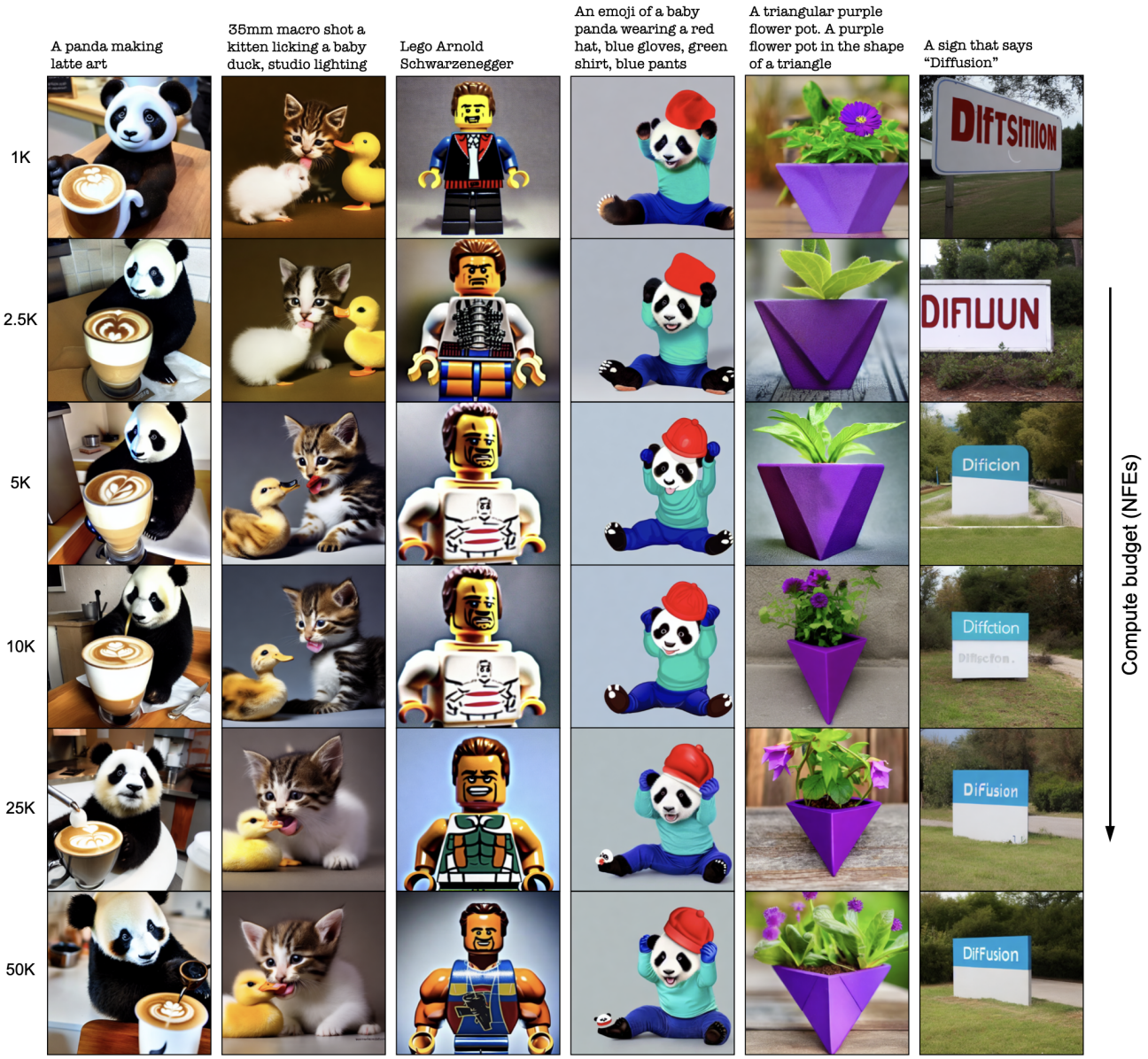

Figure 11: Text-image pairs from Figure 1 with increasing amount of inference-time compute, measured in number of function evaluations (NFEs) of the diffusion model.

Figure 12: Sample text-image pairs using Stable Diffusion v1.5 and ImageReward as the guiding function for prompts requiring a specific artistic style. Samples are picked at random for each method and prompt.

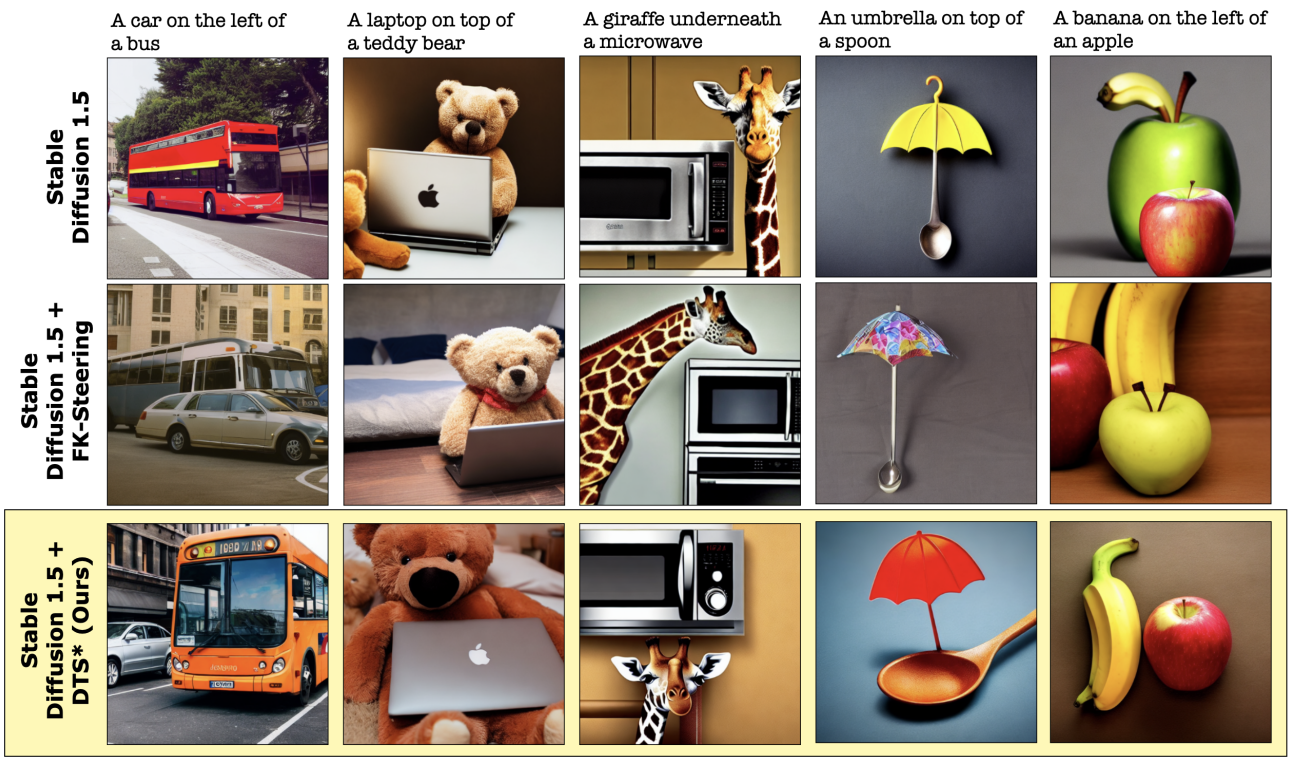

Figure 13: Sample text-image pairs using Stable Diffusion v1.5 and ImageReward as the guiding function for prompts requiring specific spatial relationships between objects. Samples are picked at random for each method and prompt.

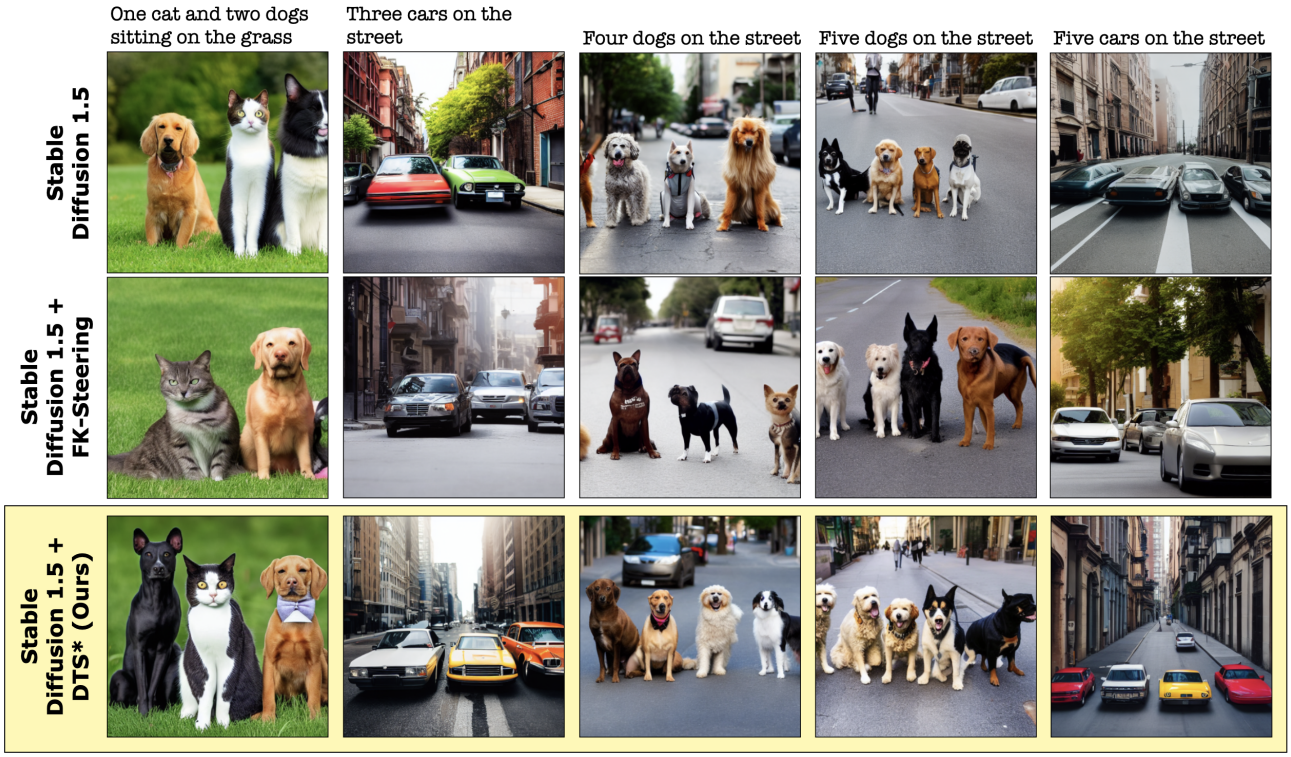

Figure 14: Sample text-image pairs using Stable Diffusion v1.5 and ImageReward as the guiding function for prompts requiring specific object counts. Samples are picked at random for each method and prompt.

J.3. Text completion examples

We present additional text completions for the base MDLM model, FK-Steering and DTS* in Figure 15.

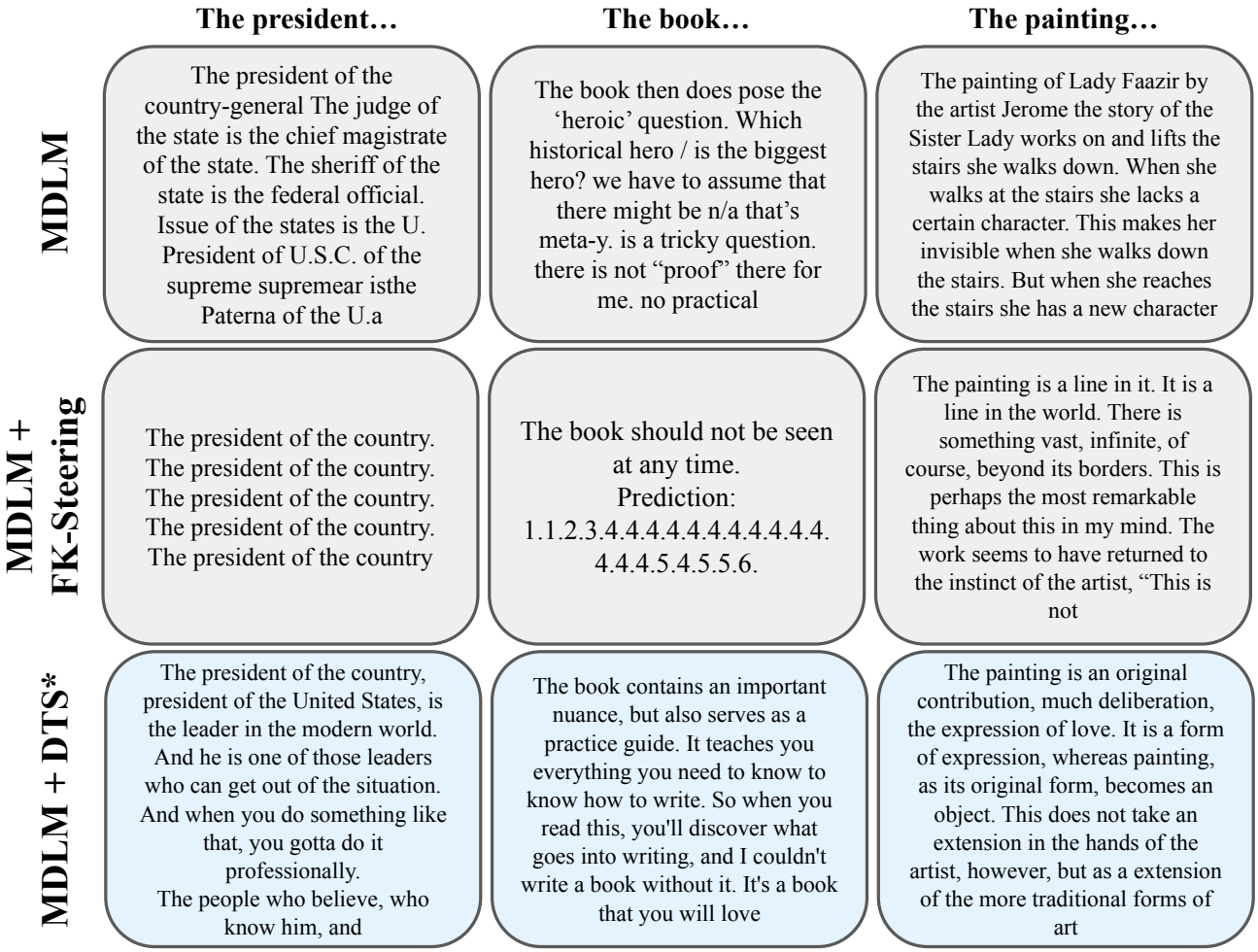

Figure 15: Sample text completions using MDLM and infini-gram perplexity as reward. Samples are picked at random for each method and prompt.

K. Extended related work

We discussed the main approaches that have been proposed for inference-time alignment of diffusion models in **??**. Below, we briefly review three tangentially related areas: fine-tuning of diffusion models, their use in reinforcement learning, and entropy-regularized variations of Monte Carlo Tree Search.

Fine-tuning of diffusion models. To sample from the target distribution π^* for a fixed, known reward function, one option is to amortize the posterior sampling problem and update the model parameters via fine-tuning. The paradigm mirrors the trajectory of large–language-model alignment (Ziegler et al., 2019; Rafailov et al., 2023; Ahmadian et al., 2024). Supervised preference finetuning trains directly on synthetic pairs scored by a reward model (Lee et al., 2023; Wu et al., 2023b). Some early methods exploit differentiable objectives to back-propagate a single scalar all the way to the noise prediction network (Clark et al., 2024; Prabhudesai et al., 2023), whereas more traditional reinforcement learning approaches cast each reverse step as an action and optimize expected reward (Black et al., 2024). To avoid over-optimization of the reward, recent works use KL regularisation (Fan et al., 2023; Uehara et al., 2024; Venkatraman et al., 2024).

Diffusion models in reinforcement learning. Since the introduction of diffusion models as powerful frameworks for generative modeling, they have become popular for sampling actions or future states in RL. The earliest successes were

in offline imitation learning, where some approaches model trajectories (Janner et al., 2022; Ajay et al., 2023) or expert policies (Chi et al., 2023) from offline datasets. Other works maximize a Q-function in addition to behavior cloning (Wang et al., 2023; Kang et al., 2023), employ an explicit actor–critic scheme (Hansen-Estruch et al., 2023), or treat the critic as an energy function to guide the denoiser (Lu et al., 2023). Some goal-conditioned extensions have also been proposed (Reuss et al., 2023; Jain & Ravanbakhsh, 2024). Recent works have explored similar ideas in the online setting (Yang et al., 2023; Psenka et al., 2024; Jain et al., 2024; Ren et al., 2024). Those methods aim to maximize return for control tasks, while we aim to draw unbiased samples from the reward-tilted distribution for any chosen reward.

Entropy-regularized MCTS. Monte-Carlo Tree-Search (MCTS) has recently been extended to soft-value objectives that incorporate an entropy bonus (Xiao et al., 2019), which uses a log-sum-exp value update and samples actions from a Boltzmann distribution, guaranteeing improved exploration at the cost of converging to the soft rather than the standard optimum. Follow-up work proposed to adapt the entropy term to a predefined value (Kozakowski et al., 2022) and decay the entropy term (Painter et al., 2023). Very recently, Morozov et al. (2024) used soft-backup MCTS to improve planning in Generative Flow Networks (Bengio et al., 2021). Our Diffusion Tree Sampling (DTS) follows the same Boltzmann selection and soft value backup pattern, it is the first to embed a *pre-trained diffusion kernel* inside the tree and to prove consistency for sampling from the KL-regularised posterior, not just selecting a single high-reward action. In this sense, DTS bridges the gap between entropy-regularized MCTS used for control and unbiased posterior sampling required for inference-time alignment of generative models.