Discrete Diffusion Reward Guidance Methods for Offline Reinforcement Learning

Matthew Coleman [1] **Olga Russakovsky** [1] **Christine Allen-Blanchette** [1] **Ye Zhu** [1,2]
[1] Princeton University [2] Illinois Institute of Technology
msc5.coleman@gmail.com, {olgarus, ca15, yezhu}@princeton.edu

Abstract

As reinforcement learning challenges involve larger amounts of data in different forms, new techniques will be required in order to generate high-quality plans with only a compact representation of the original information. While novel diffusion generative policies have provided a way to model complex action distributions directly in the original, high-dimensional feature space, they suffer from slow inference speed and have not yet been applied with reduced dimension or to discrete tasks. In this work, we propose three diffusion-guidance techniques with a reduced representation of the state provided by quantile discretization: a gradient-based approach, a stochastic beam search approach, and a Q-learning approach. Our findings indicate that the gradient-based and beam search approaches are capable of improving scores on an offline reinforcement learning task by a significant margin.

1 Introduction

The field of reinforcement learning (RL) represents the difficult challenge of training an autonomous agent to achieve some predefined objective by interacting with the environment. While many works approach this problem from a purely theoretical decision-making perspective, recent results [22, 10, 6] provide a basis for practical improvement, indicating that the use of generative modeling to improve the sample quality of actions can have a great effect on the ultimate behavior in the environment.

While it is attractive to apply diffusion-based generative models [15, 16, 8] to reinforcement learning tasks as a new, more expressive policy class, such approaches [9, 22, 6] still suffer from major drawbacks, requiring great computational effort and long inference times in exchange for policy improvement. Without major strides in improving diffusion policy efficiency, their use in tasks that require real-time inference, such as robotics, will remain intractable.

Although one approach to improve efficiency is by directly reformulating the diffusion process [18], such approaches do not scale with the dimensionality of state information, which is crucial for applications like robotics which often require detailed observations that may come from video [21, 2, 20]. Alternatively, a diffusion policy formulation that makes use of reduced state information could allow for more flexible conditioning and ultimately greater efficiency in plan generation. Moreover, data-efficient diffusion models [5, 14] are capable of reducing the computational effort required for training and inference, but in simplifying the data representation by encoding and quantization, these models make the RL task of policy optimization more difficult by obscuring the correlation between representation and state.

To that end, we propose in this work three methods for driving a discrete diffusion model process towards optimal trajectory generation. These methods include a gradient-based optimization technique that emulates Gaussian perturbation for categorical transition distributions, a value-guided sampling approach that resembles beam search [13], and finally, a Q-learning approach that modifies the variational diffusion training process. We evaluate these approaches on the offline reinforcement

ICML 2023 Workshop: Sampling and Optimization in Discrete Space (SODS 2023).

learning task using the D4RL dataset [3], and find that the gradient-based approach and beam search attain measurably superior performance compared to a baseline (behavior-cloning) diffusion policy.

2 Background

Offline Reinforcement Learning. Reinforcement learning [19] is typically defined as a Markov decision process (MDP) defined by $M = \{\mathcal{S}, \mathcal{A}, p, r, \gamma\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, p denotes the environment dynamics $p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$, r is a scalar reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and γ is a constant discount factor that describes how quickly rewards decay over time. The goal is to find a policy $\pi(\boldsymbol{a}_t \mid \boldsymbol{s}_t)$ that maximizes the discounted sum of rewards, also called return [22], i.e.:

$$R(t) = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{1}$$

Given a dataset of experiences $\mathcal{D} = \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})\}$ collected by a policy π_d, the task of offline reinforcement learning [3] is to learn a superior policy π without interacting in the environment at all during training. At test time, the new policy π is evaluated in the original live setting.

Diffusion Generative Models. Diffusion-based generative models [15, 16] (including Denoising Diffusion Probabilistic Models (DDPMs) [8], score-based models [17], and others) represent powerful techniques to generate meaningful data from noise by learning to reverse a predefined forward corruption process [8, 24, 23]. While these approaches generally feature Gaussian transitions; Discrete Diffusion Denoising Probabilistic Models (D3PMs) [1] provide an alternative formulation that makes use of categorical distributions instead, allowing for a greater variety of noising processes overall. The forward noising process is defined as:

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \text{Cat}(\boldsymbol{x}_t; \boldsymbol{p} = \boldsymbol{x}_{t-1}\boldsymbol{Q}_t), \tag{2}$$

where $\text{Cat}(\boldsymbol{x}; \boldsymbol{p})$ denotes a categorical distribution over $\boldsymbol{x}$ given by weights $\boldsymbol{p}$, and $\boldsymbol{x}$ is represented as a one-hot row vector of dimension K. The noising process is defined by the transition matrices $\boldsymbol{Q}_t$.

3 Methodology

3.1 Problem Formulation

Notation. Throughout this work we will refer to diffusion samples $\{\boldsymbol{x}^i\}$ or $\{\boldsymbol{\tau}^i\}$ using superscripts $i \in \{0 \dots N\}$ and reinforcement learning states $\{\boldsymbol{s}_t\}$, for example, using subscripts $t \in \{0 \dots T\}$.

Trajectory Optimization. The goal of this work will be to generate discrete state-action sequences or trajectories $\boldsymbol{\tau} = (\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}, \dots, \boldsymbol{s}_T, \boldsymbol{a}_T)$ that are optimal in the RL sense, *i.e.*, they maximize the expected return (Eqn. 1). Along with the stipulation that states and actions are represented as discrete vectors (reflecting a reduced form of the original continuous observations), we will focus on offline reinforcement learning as defined in section 2.

3.2 Preliminary

Diffuser [10] is a model-based diffusion policy that takes advantage of the superior generative quality of DDPMs for the task of offline reinforcement learning. After learning this distribution of trajectories from an offline dataset of interactions with an inferior policy, Diffuser uses a gradient optimization method to sample from a modified distribution of optimal trajectories, $p(\boldsymbol{\tau} \mid \mathcal{O}_{0:T})$. The optimality random variable $\mathcal{O}_t$ [12] is defined as a Bernoulli random variable with likelihood that is a function of the rewards for a state-action pair at time t:

$$p(\mathcal{O}_t = 1 \mid \boldsymbol{s}_t, \boldsymbol{a}_t) = \exp(r(\boldsymbol{s}_t, \boldsymbol{a}_t)). \tag{3}$$

Following from the methodology for perturbing the reverse kernel of a diffusion model using gradient guidance [15], the authors formulate the perturbed, optimal distribution of trajectories as:

$$\begin{aligned} \tilde{p}(\boldsymbol{\tau}) &= p(\boldsymbol{\tau} \mid \mathcal{O}_{0:T}) = p(\mathcal{O}_{0:T} \mid \boldsymbol{\tau}) p(\boldsymbol{\tau}) \\ p_\theta(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^i, \mathcal{O}_{0:T}) &\approx \mathcal{N}(\boldsymbol{\tau}^i, \mu + \Sigma \nabla_{\boldsymbol{\tau}^i} \mathcal{J}_\phi(\boldsymbol{\tau}^i), \Sigma), \end{aligned} \tag{4}$$

where $\mathcal{J}_\phi(\boldsymbol{\tau}^i)$ is a learned state-value function predictor from noisy samples $\boldsymbol{\tau}^i$, which is trained using an unbiased estimate of the value function from the offline dataset.

3.3 Method 1: Gradient-Based

Current Challenge. While Gaussian diffusion-based models such as Diffuser have the advantage of being able to directly use objective function gradients for guidance in the reverse process, the perturbed kernel $p(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^i, \mathcal{O}_{0:T})$ is not directly applicable to discrete noising processes.

Proposed Solution. Following the same intuition originally supplied in [15], we derive (in Appendix A.1) an approximation to the perturbed Gaussian kernel in Eqn. 4 that is comparable for discrete diffusion transitions. The resulting perturbed kernel is given by:

$$\tilde{p}(\boldsymbol{\tau}) \approx \log p(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^i) + \left(\nabla_{\boldsymbol{\tau}^i} \mathcal{J}_\phi(\boldsymbol{\tau}^i) - (\boldsymbol{\tau}^i)^T \nabla_{\boldsymbol{\tau}^i} \mathcal{J}_\phi(\boldsymbol{\tau}^i)\right) \tag{5}$$

This expression provides a way to compute all dimensions of the distribution in a single step. Following Diffuser's approach, the objective function $\mathcal{J}_\phi(\boldsymbol{\tau})$ is learned on the offline dataset simply by using the mean squared error between predictions and the ground-truth value function estimates.

3.4 Method 2: Beam Search

Current Challenge. Although gradient guidance in the reverse process can achieve some success in generating optimal plans as evidenced by Diffuser's results, there are several significant shortcomings that may prevent it from attaining the same results as a comparable model-free method.

One drawback is that the value prediction model is trained using noisy states $\boldsymbol{\tau}^i$, and is therefore also sensitive to the noising process in addition to the trajectory information. In general, to guide the diffusion process towards RL optimality during inference there must be some prediction of the value function based on each diffusion state, but for a discrete noising process where the gradient guidance must rely on an approximation in any case, it may be more effective to apply guidance directly.

Proposed Solution. Thus, an even more straightforward way to perturb the diffusion process towards generating optimal plans is to simply sample selectively from the learned distribution. A similar approach has previously been used to optimize a diffusion policy [6], however, this approach does not account for diffusion generation processes in which the value function prediction does not monotonically increase. Instead, we propose to use the deterministic algorithm beam search for selective sampling at each diffusion step, and to use a learned value function $\mathcal{J}_\phi(\boldsymbol{\tau}^i)$ as a score function to guide sample selection.

Specifically, at each step of the reverse process, we draw M samples from each of K diffusion kernels generated by the learned diffusion model p_θ. The K best samples (with respect to $\mathcal{J}_\phi(\boldsymbol{\tau})$) across all K beams and M samples are selected to continue the search. A more detailed explanation as well as a figure and algorithm depicting an example beam search step is provided in Appendix A.2.

The beam search algorithm provides a way to broaden the search breadth while also accounting for cases in which the noisy sample value prediction does not increase monotonically. Additionally, while gradient-based methods are fundamentally different for continuous and categorical distributions, sampling methods, however, will work for any transition distribution formulation.

3.5 Method 3: Q-Learning

Current Challenge. Another drawback of the previous value-function prediction methods in the current formulation is that the value function is not necessarily optimal, owing to the fact that all the action and reward information used to train it comes from the inferior offline policy. Thus, gradient guidance during inference will fail to induce new behavior that does not already exist in the offline dataset, since even in the best case when $\mathcal{J}_\phi$ is perfectly trained, it will still predict a lower value than may be possible. In the worst case, this detail could severely limit the potential policy results; however, in the best case it may discourage distribution shift and generation of unrealistic behavior.

Additionally, the gradient perturbation methodology (for both Gaussian and categorical processes) makes no guarantees about the optimality of the policy, and instead only guarantees that the generated trajectories move *towards* optimality over the course of the reverse diffusion process. In addition to this, it induces hyperparameters that can influence results.

Proposed Solution. To that end, we propose add an additional term to the D3PM variational lower-bound loss and train a state-value function $Q_\phi(\boldsymbol{s}_t, \boldsymbol{a}_t)$ simultaneously with the diffusion model, following in a similar procedure to [22], although applied here to a model-based discrete diffusion

policy. In contrast to the previous methods, this approach modifies the learning procedure.

$$\mathcal{L}_\theta = \mathcal{L}_{\text{vlb}} - \alpha \cdot \mathbb{E}_{\boldsymbol{s}_t \sim \mathcal{D}, \hat{\boldsymbol{a}}_t \sim p_\theta} \left[Q_\phi(\boldsymbol{s}_t, \hat{\boldsymbol{a}}_t) \right], \tag{6}$$

where Q_ϕ is trained using the double-Q learning trick [7]. The hyperparameter α is used to balance the loss contribution of the variational lower-bound term with the environment score, although for these experiments, $\alpha = 1.0$.

4 Experiments

Experimental Setup. We carried out a small series of experiments in the offline reinforcement learning task, using the "walker2d-medium" partition of the D4RL [3] dataset, which is a standard benchmark for offline control. To reduce the dimensionality of state information, we binned continuous samples by 100th quantiles along each dimension, essentially creating a codebook with a vocabulary size of 100. This technique has previously been used to discretize MuJoCo observations for the offline task in a sequence modeling approach [11].

Furthermore, we trained several D3PM models using discretized observations to generate trajectories as sequences of states and actions following from the policy optimization designs outlined in sections 3.3, 3.4, and 3.5. The best results (averaged over 15 episodes in the live environment) are collected in Table 1.

Model	Method	Score
Discrete Diffuser (Quantile Discretization)	Baseline (No Guidance)	44.7 ± 11.9
	Method 1. Gradient-based	$\mathbf{59.1 \pm 13.0}$
	Method 2. Beam Search (K=4, M=4)	$\mathbf{57.5 \pm 15.5}$
	Method 3. Q-Learning	39.7 ± 15.3
Diffuser	Baseline (No Guidance)	73.8 ± 17.3
	Gradient-based	79.6 ± 0.5

Table 1: Quantile-Discretized Policy Results. Normalized scores of each trajectory optimization method reported on the "Medium" expertise partition of the Walker dataset. The gradient-based and beam search methods improve over the baseline results by a significant margin. Each reported score is averaged over 15 episodes carried out in the environment.

Results. These results show a baseline score computed for the two generative reinforcement learning polices, i.e. Diffuser [10] and Discrete Diffuser, as well as the various policy optimization strategies outlined in previous sections. The gradient-based policy optimization method used by Diffuser is unchanged; however, Diffuser's baseline (no guidance) results are not reported in their publication, so those scores have been re-tested from an official public repository using pre-trained models.

The results from this Table show that Diffuser benefits from a small but meaningful improvement in mean score of 5.9 points over its own baseline with the kernel-perturbation guidance applied, but this conclusion largely hinges on the fact that the standard deviation decreases from 17.3 to 1.5. In contrast, Discrete Diffuser, although it attains a mean score increase (for the perturbation method) of 14.4, is not able to decrease the standard deviation by the same amount and as a result represents a policy that sometimes performs nearly the same as baseline.

Compared to model-free methods such as Diffusion-QL [22] which achieves a score of 87 ± 0.9 on this same task, Diffuser and Discrete Diffuser both attain much lower improvements. Also, comparing Diffuser and Discrete Diffuser reveals that the discrete model performed worse across all experiments (by about 25 points), implying that there may be additional problems with the discrete variant.

5 Conclusion and Discussion

In conclusion, these findings finding indicate that the two methods, gradient perturbation and beam search may each be successful in guiding the policy generation towards optimal, but more experimentation is needed to find out exactly how successful. Although diffusion policies produce promising behavior and generate high-quality action samples, their development must be tempered with the appropriate regard for the practicality and applicability in the real world, where efficiency constraints are often non-negligible. The main difficulty of balancing data-efficiency for reduced-state diffusion methods with the added policy optimization difficulty still remains relevant going forward.

References

[1] Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[2] Dos Reis, D. H., Welfer, D., De Souza Leite Cuadros, M. A., and Gamarra, D. F. T. Mobile robot navigation using an object recognition software with rgbd images and the yolo algorithm. *Applied Artificial Intelligence*, 33(14):1290–1305, 2019.

[3] Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[4] Grathwohl, W., Swersky, K., Hashemi, M., Duvenaud, D., and Maddison, C. Oops i took a gradient: Scalable sampling for discrete distributions. In *International Conference on Machine Learning*, pp. 3831–3841. PMLR, 2021.

[5] Gu, S., Chen, D., Bao, J., Wen, F., Zhang, B., Chen, D., Yuan, L., and Guo, B. Vector quantized diffusion model for text-to-image synthesis. 2022 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10686–10696, 2021.

[6] 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.

[7] Hasselt, H. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

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

[9] Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[10] Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[11] Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[12] Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[13] Reddy, D. R. et al. Speech understanding systems: A summary of results of the five-year research effort. *Department of Computer Science. Camegie-Mell University, Pittsburgh, PA*, 17: 138, 1977.

[14] 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.

[15] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

[16] Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

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

[18] Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[19] Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

[20] Tran, M. Q. and Ly, N. Q. Mobile robot planner with low-cost cameras using deep reinforcement learning. In *2020 7th NAFOSTED Conference on Information and Computer Science (NICS)*, pp. 54–59. IEEE, 2020.

[21] Wang, X. V., Pinter, J. S., Liu, Z., and Wang, L. A machine learning-based image processing approach for robotic assembly system. *Procedia CIRP*, 104:906–911, 2021.

[22] Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[23] Zhu, Y., Wu, Y., Deng, Z., Russakovsky, O., and Yan, Y. Boundary guided mixing trajectory for semantic control with diffusion models. *arXiv preprint arXiv:2302.08357*, 2023.

[24] Zhu, Y., Wu, Y., Olszewski, K., Ren, J., Tulyakov, S., and Yan, Y. Discrete contrastive diffusion for cross-modal music and image generation. In *ICLR*, 2023.

In Appendix A we provide more detailed explanations of the three proposed guidance methods. In Appendix B we include implemention details.

A Proposed Method Details

A.1 Gradient-Based Perturbation Derivation

In this section we derive the gradient-based kernel introduced in Section 3.3 in detail.

First, the goal of sampling from a distribution of trajectories that are optimal in the reinforcement learning sense can be approximated using a product of distributions [10, 12]:

$$\tilde{p}(\boldsymbol{\tau}) = p(\boldsymbol{\tau} \mid \mathcal{O}_{\boldsymbol{\tau}}) \propto p(\boldsymbol{\tau})\, p(\mathcal{O}_{\boldsymbol{\tau}} \mid \boldsymbol{\tau}). \tag{7}$$

The perturbed kernel used to generate this distribution must obey the equilibrium condition of the diffusion process [15].

$$p(\boldsymbol{\tau}^{i-1}) = \int p(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i})\, p(\boldsymbol{\tau}^{i})\, d\boldsymbol{\tau}^{i} \tag{8}$$

$$\tilde{p}(\boldsymbol{\tau}^{i-1}) = \int \tilde{p}(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i})\, \tilde{p}(\boldsymbol{\tau}^{i})\, d\boldsymbol{\tau}^{i} \tag{9}$$

$$p(\boldsymbol{\tau}^{i-1})\, p(\mathcal{O}_{\boldsymbol{\tau}^{i-1}} \mid \boldsymbol{\tau}^{i-1}) = \int \tilde{p}(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i})\, p(\boldsymbol{\tau}^{i})\, p(\mathcal{O}_{\boldsymbol{\tau}^{i}} \mid \boldsymbol{\tau}^{i})\, d\boldsymbol{\tau}^{i} \tag{10}$$

$$p(\boldsymbol{\tau}^{i-1}) = \int \tilde{p}(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i})\, p(\boldsymbol{\tau}^{i})\, \frac{p(\mathcal{O}_{\boldsymbol{\tau}^{i}} \mid \boldsymbol{\tau}^{i})}{p(\mathcal{O}_{\boldsymbol{\tau}^{i-1}} \mid \boldsymbol{\tau}^{i-1})}\, d\boldsymbol{\tau}^{i} \tag{11}$$

Thus, the following must be true in order for the equilibrium condition to be satisfied:

$$\tilde{p}(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i})\, \frac{p(\mathcal{O} \mid \boldsymbol{\tau}^{i})}{p(\mathcal{O} \mid \boldsymbol{\tau}^{i-1})} = p(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i}) \tag{12}$$

$$\tilde{p}(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i}) = p(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i})\, \frac{p(\mathcal{O} \mid \boldsymbol{\tau}^{i-1})}{p(\mathcal{O} \mid \boldsymbol{\tau}^{i})} \tag{13}$$

Then, using the definition of the optimality condition (Eqn. 3) and taking each timestep of the generated trajectory as independent:

$$\log \tilde{p}(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i}) \tag{14}$$

$$= \log p(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i}) + \log p(\mathcal{O} \mid \boldsymbol{\tau}^{i-1}) - \log p(\mathcal{O} \mid \boldsymbol{\tau}^{i}) \tag{15}$$

$$= \log p(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i}) + \textstyle\sum_{\boldsymbol{s}_t, \boldsymbol{a}_t \in \boldsymbol{\tau}^{i-1}} r(\boldsymbol{s}_t, \boldsymbol{a}_t) - \sum_{\boldsymbol{s}_t, \boldsymbol{a}_t \in \boldsymbol{\tau}^{i}} r(\boldsymbol{s}_t, \boldsymbol{a}_t) \tag{16}$$

$$= \log p(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i}) + (\mathcal{J}(\boldsymbol{\tau}^{i-1}) - \mathcal{J}(\boldsymbol{\tau}^{i})). \tag{17}$$

While it is possible (albeit generally intractable for most data) to evaluate the contribution of the objective function $\mathcal{J}(\boldsymbol{\tau})$ on each dimension of the trajectory individually over the diffusion process, a practical alternative is to instead approximate the difference $\mathcal{J}(\boldsymbol{\tau}^{i-1}) - \mathcal{J}(\boldsymbol{\tau}^{i})$ using a gradient-based estimate [4] of the true distribution given by k-bits flipping likelihood:

$$\mathcal{J}_\phi(\boldsymbol{\tau}^{i-1}) - \mathcal{J}_\phi(\boldsymbol{\tau}^{i}) \approx \nabla_{\boldsymbol{\tau}^{i}} \mathcal{J}_\phi(\boldsymbol{\tau}^{i}) - (\boldsymbol{\tau}^{i})^T \nabla_{\boldsymbol{\tau}^{i}} \mathcal{J}_\phi(\boldsymbol{\tau}^{i}). \tag{18}$$

Finally, the entire distribution can be computed using:

$$\tilde{p}(\boldsymbol{\tau}) \approx \log p(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i}) + \left(\nabla_{\boldsymbol{\tau}^{i}} \mathcal{J}_\phi(\boldsymbol{\tau}^{i}) - (\boldsymbol{\tau}^{i})^T \nabla_{\boldsymbol{\tau}^{i}} \mathcal{J}_\phi(\boldsymbol{\tau}^{i})\right). \tag{19}$$

A.2 Beam Search

This section describes the application of the beam search algorithm to the reverse diffusion process introduced in Section 3.4 in more detail.

Specifically, at each step of the reverse process, we draw M samples from each of K diffusion kernels generated by the learned diffusion model p_θ. The K best samples (with respect to $\mathcal{J}_\phi(\boldsymbol{\tau})$) across all K beams and M samples are selected to continue the search. Figure 1 depicts an example step of the beam search procedure using a diagram.

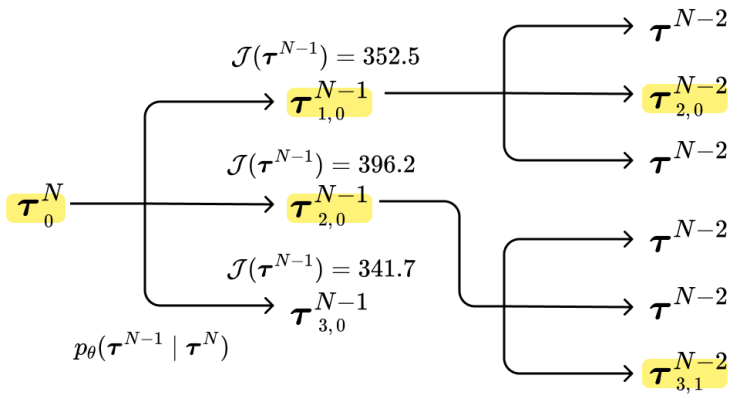

Figure 1: One step of beam search algorithm. $M = 3$ samples are taken from a single kernel and judged according to the objective function $\mathcal{J}(\boldsymbol{\tau})$. The top $K = 2$ samples (highlighted in yellow) are then forwarded through the diffusion model and the sequence is repeated.

With $K = M = \infty$, this algorithm is guaranteed to find the optimal generated trajectory, since it will cover all possibilities of the reverse diffusion chain. Thus, as K and M increase, it is expected that the value of generated trajectories will increase. Note, however, that this method does modify the learned distribution of trajectories from the offline dataset $p_\theta(\boldsymbol{\tau})$, since, over the entire process, the selective sampling procedure iteratively favors samples that are likely with respect to $\mathcal{J}_\phi(\boldsymbol{\tau})$ rather than $p_\theta(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^i)$. This can cause the opposite of the intended effect if the distribution strays too far from the dynamics of the world since any generated plans that are dynamically infeasible will result in failure.

Algorithm A.2 describes the beam-search procedure in more detail.

Algorithm 1 Beam Search Algorithm. Note, subscripts k and m denote beams and samples respectively, rather than environment timesteps.

$\boldsymbol{\tau}_k^N \leftarrow \boldsymbol{\tau}^N$
for $i \leftarrow N$ to 0 **do**
  for $k \leftarrow 0$ to K **do**
    for $m \leftarrow 0$ to M **do**
      $\boldsymbol{\tau}_{m,k}^{i-1} \sim p_\theta(\boldsymbol{\tau}_k^{i-1} \mid \boldsymbol{\tau}_k^i)$
  $\boldsymbol{\tau}_k^{i-1} \leftarrow \arg\max_{\boldsymbol{\tau}_{m,k}^{i-1}} \mathcal{J}_\phi(\boldsymbol{\tau})$
return $\arg\max_{\boldsymbol{\tau}_k^0} \mathcal{J}_\phi(\boldsymbol{\tau})$

As a visual aide, Figure 2 depicts the benefits of using beam-search over a simple maximization at each step, since the value in the generative process can improve non-monotonically.

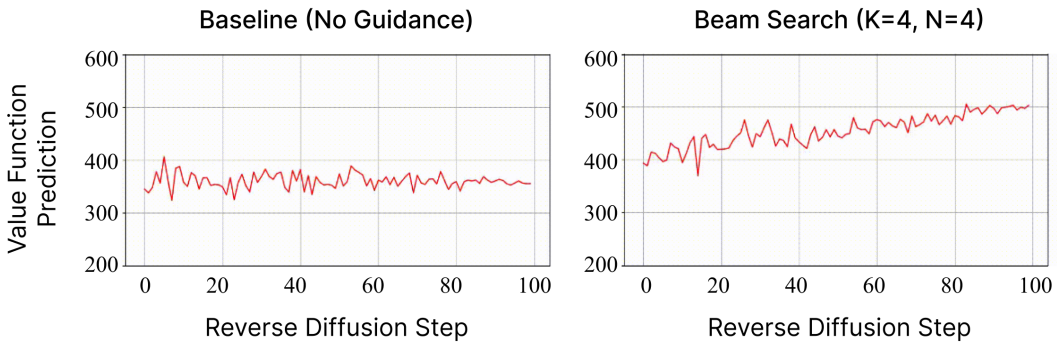

Figure 2: **Beam Search Trajectory Improvement.** These plots compare the value function prediction over the reverse diffusion process, showing that beam search improves the predicted trajectory value non-monotonically.

B Implementation Details

All reported scores result from diffusion model implemented with a 16-layer transformer architecture with 100 diffusion steps. The objective function predictors $\mathcal{J}_\phi$ and Q_ϕ have the same architecture, but with only 2 layers. The beam search experiments consist of $K = 4$ beams and $M = 4$ samples at each step. All models were trained for approximately 24 hours using a single Nvidia RTX 3090 GPU at a time.