Policy-Guided Diffusion

Matthew T. Jackson **Michael T. Matthews** **Cong Lu**
Jakob N. Foerster **Shimon Whiteson**
University of Oxford
jackson@robots.ox.ac.uk

Abstract

Model-free methods for offline reinforcement learning typically suffer from value overestimation, resulting from generalization to out-of-sample state-action pairs. On the other hand, model-based methods must handle in compounding errors in transition dynamics, as the policy is rolled out using the learned model. As a solution, we propose policy-guided diffusion (PGD). Our method generates entire trajectories using a diffusion model, with an additional policy guidance term that biases samples towards the policy being trained. Evaluating PGD on the Adroit manipulation environment, we show that guidance dramatically increases trajectory likelihood under the target policy, without increasing model error. When training offline RL agents on purely synthetic data, our early results show that guidance leads to improvements in performance across datasets. We believe this approach is a step towards the training of offline agents on predominantly synthetic experience, minimizing the principal drawbacks of offline RL.

1 Introduction

Despite the success of reinforcement learning (RL) as a field, it is known to be notoriously sample-inefficient, limiting its practicality on environments with expensive or dangerous data collection. One paradigm for overcoming this issue is offline RL [Levine et al., 2020], in which a pre-collected or "offline" dataset of experience is used to optimize a policy.

Naïvely applying off-policy RL to an offline dataset may result in catastrophic value overestimation when out-of-distribution state-action pairs are encountered [Kostrikov et al., 2021], with most existing methods aiming to explicitly counteract this [Kumar et al., 2020, An et al., 2021, Fujimoto et al., 2019]. An alternative, model-based approach is to learn a world model from the dataset and then rollout synthetic on-policy trajectories [Yu et al., 2020, Kidambi et al., 2020, Rigter et al., 2022, Lu et al., 2022]. However, errors in the world model compound over many timesteps, again leading to value overestimation and sub-optimal policies.

As an alternative to these approaches, we propose policy-guided diffusion (PGD): a novel technique for generating on-policy trajectories from an offline dataset, without the issue of compounding model error. PGD leverages the sampling quality of diffusion models to generate trajectories from the behavior distribution. We then augment the denoising process of this pre-trained diffusion model with guidance from the policy being trained, biasing the sample distribution towards the target policy. Crucially, PGD models the joint distribution over trajectories, meaning it generates entire trajectories all at once, thereby sidestepping the issue of compounding model error.

We demonstrate that policy guidance dramatically increases the likelihood of generated trajectories under the target distribution, without increasing the model dynamics error. Early results on D4RL [Fu et al., 2020] show promise for using guided synthetic data to learn effective policies, especially on datasets that are small or contain sub-optimal trajectories.

NeurIPS 2023 Workshop on Robot Learning: Pretraining, Fine-Tuning, and Generalization with Large Scale Models, New Orleans

2 Background

2.1 Offline Reinforcement Learning

We adopt the standard reinforcement learning (RL) formulation where an agent acts in a Markov Decision Process (MDP, Sutton and Barto [2018]). An MDP is defined as a tuple $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and γ is the discount factor. The goal of RL is to learn a policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ which is trained to maximize expected return, defined as $\mathbb{E}_{\pi,\mathcal{T}}[\sum_{t=0}^{H} \gamma^t r_t]$ where H is the horizon and r_t is the reward at timestep t. In the *offline RL setting* [Levine et al., 2020], the agent is unable to interact with the environment and instead is given a fixed dataset of trajectories $\mathcal{D} = \{\tau_i\}_{i=1,\dots,N}$ gathered by some behavior policy π_β. The core problem introduced is the distribution shift between the behavior policy π_β and the policy being optimized π, called the target policy. This can lead to catastrophic value overestimation at unobserved actions, a problem termed the out-of-sample issue [Kostrikov et al., 2021].

2.1.1 The Out-of-Sample Issue in Model-Based Offline Reinforcement Learning

Many issues of offline RL methods result from training on the behavior distribution over trajectories $p_\beta(\tau)$. Conversely, it is possible to learn a single-step world model $\mathcal{M}$ from $\mathcal{D}$ [Yu et al., 2020, Kidambi et al., 2020, Rigter et al., 2022, Lu et al., 2022]. By rolling out the target policy using $\mathcal{M}$, we generate trajectories $\tau \sim p(\tau)$ from the trajectory distribution of the target policy, or *target distribution*. While this would seem to overcome the problem of training on out-of-distribution state-action pairs, this technique simply pushes the generalization issue into the world model. In particular, the RL policies are prone to exploiting errors in the world model, which can compound over the course of an episode, again causing value overestimation.

2.2 Diffusion Models

In order to generate synthetic data, we consider diffusion models [Sohl-Dickstein et al., 2015, Ho et al., 2020], a class of generative model that allows one to sample from a distribution $p(\boldsymbol{x})$ by iteratively reversing a forward noising process. Karras et al. [2022] present an ODE formulation of diffusion which, given a noise schedule $\sigma(t)$, mutates data according to

$$d\boldsymbol{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p\left(\boldsymbol{x}; \sigma(t)\right) dt, \tag{1}$$

where the dot denotes a time derivative and $\nabla_{\boldsymbol{x}} \log p\left(\boldsymbol{x}; \sigma(t)\right)$ is the score function [Hyvärinen and Dayan, 2005], which points towards areas of high data density. Intuitively, infinitesimal forward or backward steps of this ODE either nudge a sample away or towards the data.

2.2.1 Classifier-Guided Diffusion

Our method is designed to guide the data-generating process towards on-policy trajectories, rather than the marginal behavior distribution $p_\beta(\boldsymbol{x})$. For this, we take inspiration from classifier guidance [Dhariwal and Nichol, 2021], which leverages a differentiable classifier to augment the score function of a pre-trained diffusion model towards a class-conditional distribution $p(\boldsymbol{x}|y)$. Concretely, this adds a classifier gradient to the score function, giving

$$\nabla_{\boldsymbol{x}} \log p_\lambda \left(\boldsymbol{x}|y; \sigma(t)\right) = \nabla_{\boldsymbol{x}} \log p\left(\boldsymbol{x}; \sigma(t)\right) + \lambda \nabla_{\boldsymbol{x}} \log p_\theta\left(y|\boldsymbol{x}; \sigma(t)\right), \tag{2}$$

where $\nabla_{\boldsymbol{x}} \log p_\theta\left(y|\boldsymbol{x}; \sigma(t)\right)$ is the gradient of the classifier and λ is the guidance weight.

3 Policy-Guided Diffusion

As described in Section 2.1, one of the main challenges in offline RL is distribution shift between the behavior policy π_β and the target policy π. As a solution, we propose PGD, a method that augments the denoising process of an offline diffusion model, increasing the probability of sampled trajectories under the target distribution.

We model the behavior distribution by learning a score function $\nabla_\tau \log p_\beta(\tau)$ from the offline dataset $\mathcal{D}$. Inspired by classifier-guided diffusion (Section 2.2.1), we guide the diffusion process using the

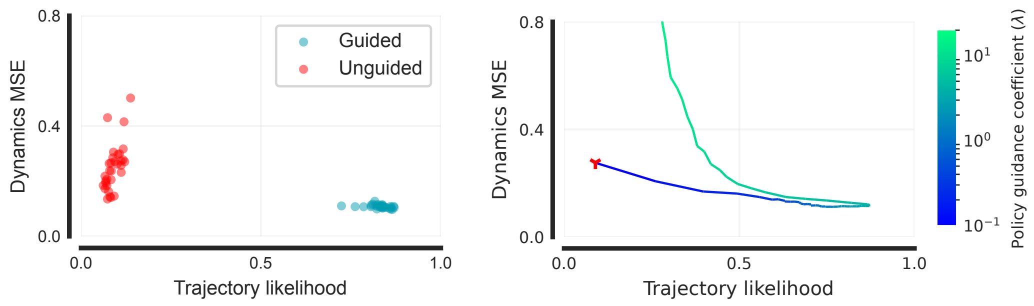

Figure 1: Dynamics mean squared error v.s. trajectory likelihood for a set of synthetic trajectories generated from a diffusion model trained on the `pen-cloned-v1` offline dataset. The guidance policy was trained using PPO on the online environment. **Left:** Synthetic trajectory statistics, for guided ($\lambda = 2.0$) and unguided ($\lambda = 0.0$) models. **Right:** Mean trajectory statistics over a range of guidance coefficients, with the red marker denoting no guidance.

gradient of the action distribution under the target policy, conditioned on the denoised states. This results in a denoising step of the form

$$\mathcal{F}(\tau|\pi, \lambda) = \nabla_\tau \log p_\beta(\tau) + \lambda \sum_{t \in |\tau|} \nabla_\tau \log \pi(a_t|s_t), \tag{3}$$

where (s_t, a_t) is a state-action pair from the trajectory τ. The action gradient $\nabla_\tau \log \pi(a_t|s_t)$ gives the gradient of the action a_t with respect to the policy $\pi(\cdot|s_t)$, meaning no guidance is directly applied to states. However, both states and actions are denoised by the score function, resulting in second-order guidance of states in response to action guidance. We note that, unlike classifier guidance, $\mathcal{F}(\tau|\pi, \lambda)$ does not model the score function for the conditional $p(\tau|\pi)$.

Intuitively, the guidance term nudges the actions in the trajectory towards those that the target policy would choose given the noised states. By tuning the guidance coefficient λ, PGD provides flexibility in balancing target and behavior probabilities. We assume that trajectories with high behavior probability will have lower dynamics error due to increased proportion in the offline dataset, making this tuning a proxy for trading-off target probability with dynamics error. Furthermore, PGD avoids the compounding error found when rolling out policies in learned world models by modeling the joint trajectory distribution.

4 Experiments

4.1 Experimental Setup

We investigate the performance of PGD on the Pen task from the Adroit environment [Rajeswaran et al., 2017] using the `pen-human-v1` ("Human", a small dataset of human trajectories), `pen-expert-v1` ("Expert", a large dataset of expert trajectories) and `pen-cloned-v1` ("Cloned", a large dataset of behavior-cloned trajectories) offline datasets from D4RL [Fu et al., 2020]. In order to train an expert agent (Section 4.2), we train with PPO [Schulman et al., 2017] directly on the environment, whilst we use IQL [Kostrikov et al., 2021] for policy optimization on synthetic data (Section 4.3). Hyperparameters for diffusion and RL are given in the supplemental material.

4.2 Trajectory Analysis

In order to analyze PGD, we define two metrics of synthetic trajectories $\tau = (s_0, a_0, ..., s_T, a_T)$:

- **Trajectory Likelihood** — The mean of the action probabilities $\frac{1}{T-1}(\sum_{t=0}^{T-1} \pi(a_t|s_t))$ for a target policy. This is a proxy objective for $p(\tau|\pi)$, intended to measure how on-policy a trajectory is.
- **Dynamics Error** — Forward dynamics model mean squared error, which is defined by $\frac{1}{T-1}\sum_{t=0}^{T-1}(\mathcal{T}(s_t, a_t) - s_{t+1})^2$ (further discussion in the supplementary material).

In Figure 1, we observe that, with a tuned guidance coefficient, guided synthetic trajectories are both further on-policy and exhibit lower dynamics error when compared to unguided trajectories. This occurs despite the Cloned behavior policy achieving significantly lower return than the expert PPO policy, demonstrating the generalization capability of PGD when guided by out-of-distribution target policies. The reduction in dynamics error is particularly interesting, as one might expect the guidance term to induce high model error. We hypothesize this is due to expert trajectories in the Pen environment being smoother and having lower state-space coverage, making them easier to model. This behavior is in contrast to offline model-based RL methods that use a forward dynamics model, where target policy rollouts can lead to large compounding model errors (Section 5). As the policy guidance coefficient increases beyond a threshold value (around 2.0 in our experiments) the generated trajectories become less on-policy and dynamics error rapidly increases. We hypothesize that this threshold value is where the magnitude of the guidance term starts to destabilize the diffusion process.

4.3 Benchmark Evaluation

We train an IQL agent [Kostrikov et al., 2021] on purely synthetic data from PGD, with $\lambda = 0$ and $\lambda = 2$ (Table 1). These results show an improvement on all datasets, but lack statistical significance. Notably, the largest performance increase is found on the smallest dataset (Human), reflecting the potential of guidance in the low-data domain. We believe that these results, along with the analysis presented in Figure 1, are an early indication of the potential of agents trained with purely synthetic data from PGD.

Table 1: Performance of IQL agents on Pen, trained on synthetic data with and without policy guidance (mean $\pm$ standard error over 5 seeds).

Dataset	**Return**	
	$\lambda = 0.0$	$\lambda = 2.0$
Human	2232 ± 230	2448 ± 130
Cloned	1649 ± 113	1763 ± 89
Expert	3113 ± 134	3166 ± 146

5 Related Work

Model-Based Offline Reinforcement Learning Model-based methods in offline RL [Yu et al., 2020, Kidambi et al., 2020, Rigter et al., 2022, Lu et al., 2022] are designed to augment the offline buffer with additional on-policy samples in order to mitigate distribution shift. This is typically done by rolling out a policy in a learned world model [Janner et al., 2019] and applying a suitable pessimism term in order to account for dynamics model errors. While these methods share the same overall motivation as our paper, the empirical realization is quite different. In particular, forward dynamics models are liable to compounding errors over long horizons, resulting in model exploitation, whereas our trajectories are generated in a single step.

Model-Free Offline Reinforcement Learning Model-free methods in offline RL typically tackle the out-of-sample issue by applying conservatism to the value function or by constraining the policy to remain close to the data. For example, CQL [Kumar et al., 2020] and EDAC [An et al., 2021] both aim to minimize the values of out-of-distribution actions. Meanwhile, BCQ [Fujimoto et al., 2019] ensures that actions used in value targets are in-distribution with the behavioral policy using constrained optimization. We take the opposite approach in this paper—by enabling our diffusion model to generate on-policy samples without diverging from the behavior distribution, we reduce the need for conservatism. SynthER [Lu et al., 2023] proposed the use of unguided diffusion models to upsample the offline dataset with synthetic transitions from the behavior policy, to then be operated on by model-free offline RL methods. Whilst this improves performance, the use of unguided diffusion results in the same issue of distributional shift.

6 Conclusion

In conclusion, we present policy-guided diffusion, a method for generating on-policy trajectories from an offline dataset, whilst avoiding the issue of compounding model error experienced by model-based offline RL methods. We intend to further explore this method by experimenting with noise-resistant policies for guidance, improving multi-step value estimation with guided synthetic data, and leveraging large-scale offline datasets for increasingly complex environments.

Acknowledgments and Disclosure of Funding

Matthew Jackson is funded by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems, and Amazon Web Services. Michael Matthews is funded by the EPSRC Doctoral Training Partnership.

References

G. An, S. Moon, J.-H. Kim, and H. O. Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

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

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

S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/fujimoto19a.html`.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf`.

A. Hyvärinen and P. Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

C. Lu, P. Ball, J. Parker-Holder, M. Osborne, and S. J. Roberts. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=zz9hXVhf40`.

C. Lu, P. J. Ball, and J. Parker-Holder. Synthetic experience replay. *arXiv preprint arXiv:2303.06614*, 2023.

A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

M. Rigter, B. Lacerda, and N. Hawes. RAMBO-RL: Robust adversarial model-based offline reinforcement learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=nrksGSRT7kX`.

O. Ronneberger, P. Fischer, and T. Brox. 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*, pages 234–241. Springer, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/sohl-dickstein15.html`.

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

T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/a322852ce0df73e204b7e67cbbef0d0a-Paper.pdf`.

A Model Hyperparameters

A.1 EDM

For EDM we used the default hyperparameters from Lu et al. [2023] (Table 2). We observed instability from policy guidance early in the diffusion process, as a result of trajectory noise. We avoided this by implementing a warm-up period, during which no guidance was applied.

Table 2: EDM hyperparameters

Hyperparameter	**Value**
Diffusion timesteps	400
Guidance warm-up	64
S_{churn}	80
S_{noise}	1.003
S_{tmax}	50
S_{tmin}	0.05
σ_{max}	80
σ_{min}	0.002

A.2 Diffusion model

For the diffusion model we used a U-Net architecture [Ronneberger et al., 2015] with the hyperparameters outlined in Table 3.

Table 3: U-Net hyperparameters

Hyperparameter	**Value**
# of epochs	512
Batch size	16
Learning rate	0.001
# of blocks	2
# of features	1024
Optimizer	Adam
Trajectory length	128

A.3 IQL

For IQL we used the hyperparameters outlined in Table 4.

Table 4: IQL hyperparameters

Hyperparameter	**Value**
β	3
τ	0.7
γ	0.99
Learning rate	0.0003
Batch size	512
Train steps	50000

B Approximating Simulation Error in Adroit

Since our diffusion model generates observations, we were unable to exactly recover the underlying states to be fed into the simulator in order to calculate the forward dynamics error. Of the 64 scalars that make up the state space in `adroit-pen`, 24 are given to us in the observation (positions of joints). The corresponding 8 velocities (each 3 dimensions) of these joints can be calculated using the finite difference between subsequent observations, divided by the api frequency (200 Hz). A further 4 scalars can be calculated by converting the Euler angles of the desired orientation into a Quaternion. This leaves 12 scalars (2 positions and 2 velocities) unaccounted for that we cannot approximate, which we simply set to 0. While clearly not a perfect representation of the true state, we believe that this approximation is close enough to be a useful approximation.