Generative Reinforcement Learning via Denoising Diffusion

Yibin Hu
Tulane University

October 14, 2024

Abstract

Diffusion Model for RL[1] is deployed on Walker2D task to gain insights and findings.

1 Motivation

The application of generative methods in reinforcement learning (RL) aligns closely with our intuitive understanding of how agents should interact with their environments. When presented with a current state that encapsulates the environment at a specific moment, an intelligent agent naturally processes this information to predict and plan a sequence of future states and actions. This cognitive approach mirrors the core principles of generative modeling, making it a promising avenue for RL research.

Diffusion models, in particular, are well-suited for this purpose as they can effectively capture the complex dynamics of state transitions and action selections. By leveraging their capacity to model intricate relationships and uncertainties, diffusion models provide a robust framework for understanding and guiding agent behavior in various tasks. Specifically, we investigate the Walker2D task to glean insights into how these models can enhance decision-making processes in RL, ultimately contributing to the development of more adaptive and efficient agents.

2 Algorithm for Guided Generative Diffusion

Given a trajectory which is sequence of state action pair. The diffusion model[2] is trained by learning to iteratively denoise the trajectory from Gaussian noise while conditioned on initial state. A value function estimating the value a trajectory is used to steer the generation in favor high value trajectory. So the training contains two parts.

1, Train a base diffusion generative model $p_\theta(\tau|s)$, which given s generateS the trajectory that start with this state s

2, Train a value estimation function $V(\tau)$ that estimates the value of trajectory

Algorithm 1.1 Diffusion: Training of Prior Diffusion $p_\theta(\boldsymbol{\tau}|s)$

$\{\boldsymbol{\tau}\} \leftarrow SampleTrajectories()$
schedule $\{\beta_t\}$
while θ not converge **do**
  $\boldsymbol{\tau^0} \leftarrow sample(\{\boldsymbol{\tau}\})$ ▷ $shape(\boldsymbol{\tau}) = (T+1, dim(s)+dim(a))$
  $t \leftarrow Uniform(\{1, ..., T_d\})$ ▷ t is diffusion time not planning time
  $\boldsymbol{\epsilon} \leftarrow \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\boldsymbol{\tau}^t \leftarrow \sqrt{\bar{\alpha}_t}\boldsymbol{\tau}^0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ ▷ $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i, \alpha_t = 1 - \beta_t$
  $s_0^t \leftarrow s_0^0$ ▷ condition on noiseless initial state s_0^0
  $\theta \leftarrow \theta - \eta\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\boldsymbol{\tau}^t, t\right)\right\|^2$

Algorithm 1.2 Denoising: Sampling of Prior Diffusion $p_\theta(\boldsymbol{\tau}|s)$

$\boldsymbol{\tau}^{T_d} \leftarrow \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$s_0^t \leftarrow s$
for $t = T_d, ..., 1$ **do**
  $\mathbf{z} \leftarrow \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ if $t > 1$, else $\mathbf{z} = 0$
  $\boldsymbol{\tau}^{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{\tau}^t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{\tau}^t, t) \right) + \sigma_t \mathbf{z}$ ▷ $\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
  $s_0^{t-1} \leftarrow s$
return $\boldsymbol{\tau}^0$

3 Experiment

The diffusion model is deep UNet composed with residual blocks. An linear attention is used to exploit the relation within the planning horizon, we found this is important for performance, as the unweighted loss treat predictions over the horizon equally important, but in reality, state-action predictions that follows the initial state and near the start of the sequence is more important, as a smaller horizon can be used in testing, and we also note the use of long horizon in training significantly increase the difficulty for training. The value model bear the structure of the first half UNet and then process the compressed output into a value. The horizon is chosen to be 8. The result is shown in table below, where for each window size 10 runs are done and averaged to get the mean performance. Value rate is the mean value accumulated at each step.

```
Window Size 1: Mean Value = 4461.9377, V-Rate = 4.8190
Window Size 2: Mean Value = 3660.7400, V-Rate = 4.7716
Window Size 3: Mean Value = 3764.6957, V-Rate = 4.7824
Window Size 4: Mean Value = 4083.3252, V-Rate = 4.8152
Window Size 5: Mean Value = 4295.3235, V-Rate = 4.8278
Window Size 6: Mean Value = 3192.0946, V-Rate = 4.6812
Window Size 7: Mean Value = 3740.4791, V-Rate = 4.7504
```

(a) Raw

Window Size	Normalized Score	Value Rate
1	90.318	0.975
2	74.100	0.966
3	76.204	0.968
4	82.654	0.975
5	86.945	0.977
6	64.614	0.948
7	75.714	0.962
Guided 7	84.065	0.971

(b) Normalized

Figure 1: results of testing for agent trained on walker2d-expert-v2 dataset, where value is normalized to percentage of dataset's mean value by $v_{normalized} = \frac{v}{v_{data}} \times 100$, and this is slightly different from what used in original paper $v_{normalized} = \frac{v - v_{random}}{v_{max} - v_{random}} \times 100$, where v_{random} is value achieved by random action sampler and v_{max} is max value recorded on this dataset.

The figure below shows value achieved by individual runs.

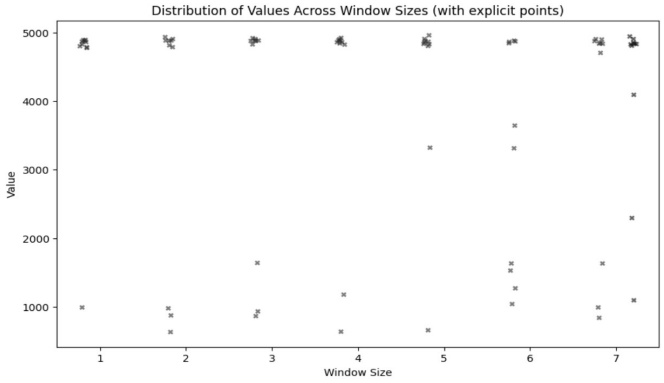

Figure 2: values of test runs, some run ends early as agent encounter out of data distribution and fall, and this happen more frequently as the window size increases

The fact that all normalized scores are below 100% (i.e., less than 1 when not converted to percentage) suggests that the diffusion model does not surpass the performance of the data it was trained on, which is expected. This aligns with the design of diffusion models, which are intended to sample from the distribution of the training data. Our results largely agree with those reported in the original paper, where scores also remained below 100%, with the exception of the Medium-Expert dataset. The higher scores in this case can be attributed to the dataset being a mixture of medium and expert data, allowing the agent to exploit the expert data and exceed the average performance of the dataset.

Dataset	Environment	BC	CQL	IQL	DT	TT	MOPO	MOReL	MBOP	Diffuser
Medium-Expert	HalfCheetah	55.2	91.6	86.7	86.8	95.0	63.3	53.3	**105.9**	88.9 ±0.3
Medium-Expert	Hopper	52.5	**105.4**	91.5	**107.6**	**110.0**	23.7	**108.7**	55.1	103.3 ±1.3
Medium-Expert	Walker2d	**107.5**	**108.8**	**109.6**	**108.1**	101.9	44.6	95.6	70.2	**106.9** ±0.2
Medium	HalfCheetah	42.6	44.0	**47.4**	42.6	**46.9**	42.3	42.1	44.6	42.8 ±0.3
Medium	Hopper	52.9	58.5	66.3	67.6	61.1	28.0	**95.4**	48.8	74.3 ±1.4
Medium	Walker2d	75.3	72.5	**78.3**	74.0	**79.0**	17.8	**77.8**	41.0	**79.6** ±0.55
Medium-Replay	HalfCheetah	36.6	45.5	44.2	36.6	41.9	**53.1**	40.2	42.3	37.7 ±0.5
Medium-Replay	Hopper	18.1	**95.0**	**94.7**	82.7	**91.5**	67.5	**93.6**	12.4	**93.6** ±0.4
Medium-Replay	Walker2d	26.0	77.2	73.9	66.6	**82.6**	39.0	49.8	9.7	70.6 ±1.6
Average		51.9	**77.6**	**77.0**	74.7	**78.9**	42.1	72.9	47.8	**77.5**

Table 2. **(Offline reinforcement learning)** The performance of Diffuser and a variety of prior algorithms on the D4RL locomotion benchmark (Fu et al., 2020). Results for Diffuser correspond to the mean and standard error over 150 planning seeds. We detail the sources for the performance of prior methods in Appendix A.3. Following Kostrikov et al. (2022), we emphasize in bold scores within 5 percent of the maximum per task ($\geq 0.95 \cdot \max$).

Figure 3: results in locomotion from original paper[1]

References

[1] Janner, M., Du, Y., Tenenbaum, J. B., Levine, S. (2022). Planning with diffusion for flexible behavior synthesis. arXiv. https://doi.org/10.48550/arXiv.2205.09991

[2] Ho, J., Jain, A., Abbeel, P. (2020). Denoising diffusion probabilistic models. arXiv. https://doi.org/10.48550/arXiv.2006.11239