Towards Planning with Diffusion for Complex Environments

Matin Moezzi
1008701479

Xuanchi Ren
1009173403

Abstract: Model-based reinforcement learning (RL) techniques employ learning just to estimate an approximate dynamics model, leaving the remaining decision-making to traditional trajectory optimizer or policy optimization methods. While theoretically straightforward, this independent two-stage planning has a number of empirical flaws that indicate trained models might not be the best choice for the learned policy or conventional trajectory optimization. Taking advantage of the diffusion probabilistic model, some previous works tried to incorporate a trajectory optimization pipeline into model learning such that sampling from the model and using it for planning become essentially equivalent. In this work, we first perform an empirical study on injecting different types of attention-based layers into the Diffuser. By adding a global receptive field, the Diffuser performs better than the one with the only local receptive field. Then, we propose a latent-based conditional diffusion model for dynamic environments. We test our proposed approach on the MinAtar games and achieve good performance. Our code is available here.

Keywords: Model-based RL, Diffusion Probabilistic Models, Planning for RL

1 Introduction

For model-based reinforcement learning (RL), we first learn a predictive model and use this learned model to evaluate potential sequences of actions and select the best one. Though this process is conceptually simple, it suffers from severe problems. Trajectories generated by this pipeline resemble adversarial examples rather than optimal solutions. Thus, instead of following the above trajectory optimization procedure, model-based RL approaches usually inherit more from model-free algorithms like value functions and policy gradients. To address this issue, Janner et al. [1] proposes to treat trajectory optimization as a generative modeling process with diffusion models.

Diffuser by Janner et al. [1] demonstrates a great potential for using diffusion models for planning. It is mainly applied to simple scenarios such as a Maze with a local receptive field and states represented by main coordinates. However, when it comes to scenarios with multi-agents and complex environments (i.e., Atari 2600 games: Freeway and Asterix), such a local receptive field will fail to capture the history of trajectories of agents and the interaction between the ego-agent and the environment, as shown in Figure 1.

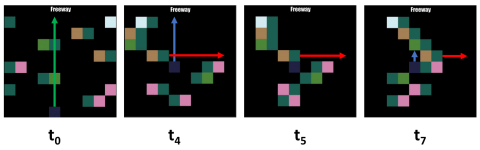


Figure 1: Scenario of Freeway environment from MinAtar's [2] Atari 2600 games. The agent-controlled blue car has to drive from the bottom to the top (green arrow at t_0). It has to wait at t_4 to avoid a crash with the car with the red arrow. It can continue at t_7. This demonstrates the necessity of global awareness of the whole environment as well as other agents.

In this project, we plan with a learned diffusion model in challenging dynamic environments containing obstacles and multi-agents, such as a high-DoF robotic hand (Adroit) or miniaturized versions of Atari 2600 games, to fully leverage the potential of Diffuser [1]. To realize this, we first replicate the result of Janner et al. [1]. Based on it, we perform an empirical analysis on combining attention layers with the current architecture to

empower the Diffuser [1] with the global receptive field. In addition, we propose a new framework based on conditional diffusion models for dynamic environments. We also adopt a latent diffusion manner to facilitate image-based states.

Main contribution. We summarize our project as follows:

- We extend the implementation of Janner et al. [1] to be able to handle various tasks from the high-DoF D4RL Adroit domain [3].
- We add a non-local receptive field to Diffuser [1] using attention-based layers so that it outperforms the original Diffuser [1] on the locomotion benchmark.
- We expand Diffuser [1] further by utilizing latent space encoding and state conditioning to enable it for dynamic, high-dimensional environments. This method performs well on the Asterix game from the MinAtar [2].

2 Related Work

Model-based RL and Planning. A lot of research has been done on model-based reinforcement learning. Their differences stem from the choice of model parameterization, which is connected to various applications of the model for policy learning. The most impressive robotic learning applications yet, it turns out, have been accomplished utilizing the most basic model parameterization, notably, linear models, where the model either operates directly over the raw state or over a feature representation of the state. Such models enable very effective policy optimization using methods from optimal control. However, unless a separate feature learning phase is implemented, they only have limited expressiveness and do not scale well to complex nonlinear dynamics or high-dimensional state spaces.

Nonparametric models like Gaussian Processes (GPs) are an alternative. As long as there is a sufficient amount of data, such models may effectively preserve uncertainty over the predictions and have limitless representation power. However, due to the curse of dimensionality, they are only truly useful in environments with few dimensions.

Diffusion Models. With the success of Diffusion Models in density estimation and sample quality, it is also introduced into the domains of images [4, 5, 6], video [7, 8, 9], and 3D [10, 11] with an underlying neural backbone as a UNet [12]. The generative process of Diffusion Models is formulated as an iterative denoising procedure [4] with lots of variants. While Diffusion Models in these domains achieve amazing performance and have been well-investigated, there are few works developing them for decision-making or reinforcement learning tasks. Janner et al. [1] make the first step with a local receptive field on some simple scenarios. In this project, we plan to extend it to complex scenarios, which are challenging and require a novel architecture.

3 Methodology

In this section, we first briefly review some basic concepts of diffusion models and the diffusion model for trajectory optimization - Diffuser [1]. Then we elaborate on how to enable Diffuser to have knowledge of the global context and then our new version of the conditional diffusion model for dynamic environments.

3.1 Background

Diffusion Models. Diffusion models are probabilistic models that learn a data distribution $p_\theta(\boldsymbol{X}_0)$ by gradually denoising a standard Gaussian distribution, in the form of $p_\theta(\boldsymbol{X}_0) = \int p_\theta(\boldsymbol{X}_{0:T})\, d\boldsymbol{X}_{1:T}$. Here, $\boldsymbol{X}_{1:T}$ are intermediate denoising results with the same shape as $\boldsymbol{X}_0 \sim q(\boldsymbol{X})$, and θ are learnable parameters of the deep denoising network, usually realized as a U-Net.

Figure 2 shows the graphical model of a typical diffusion model [4]. The joint distribution $q(\boldsymbol{X}_{1:T}|\boldsymbol{X}_0)$ is called the *forward process* or *diffusion process*, which is a fixed Markov Chain

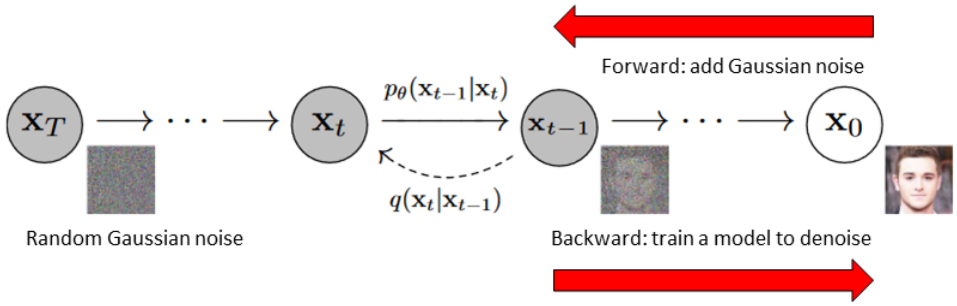


Figure 2: Forward and reverse process of a typical diffusion model. Image adopted from [4].

that gradually adds Gaussian noise to the clean data $\boldsymbol{X}_0$. The noise is controlled by a pre-defined variance schedule $\{\beta_t\}_{t=1}^T$:

$$q(\boldsymbol{X}_{1:T}|\boldsymbol{X}_0) = \prod_{t=1}^{T} q(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}) \tag{1}$$

$$\begin{aligned} q(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}) &= \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{X}_{t-1}, \beta_t \boldsymbol{I}) \\ &= \sqrt{1-\beta_t}\boldsymbol{X}_{t-1} + \beta_t \epsilon_t \\ &\text{where } \epsilon_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \end{aligned} \tag{2}$$

Thanks to the nice property of Gaussian distributions, good property of this formulation is that $\boldsymbol{X}_t$ can be sampled directly from $\boldsymbol{X}_0$ in closed form without adding the noise t times. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, we have:

$$\begin{aligned} q(\boldsymbol{X}_t|\boldsymbol{X}_0) &= \mathcal{N}(\sqrt{\bar{\alpha}_t}\boldsymbol{X}_0, (1-\bar{\alpha}_t)\boldsymbol{I}) \\ &= \sqrt{\bar{\alpha}_t}\boldsymbol{X}_0 + (1-\bar{\alpha}_t)\epsilon_t \end{aligned} \tag{3}$$

We can now train a model to reverse this process and thus generate target data from random noise $\boldsymbol{X}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The *reverse process* $p_\theta(\boldsymbol{X}_{0:T})$ is also defined as a Markov Chain with a learned Gaussian transition:

$$\begin{aligned} p_\theta(\boldsymbol{X}_{0:T}) &= p(\boldsymbol{X}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t) \\ p_\theta(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t) &= \mathcal{N}(\mu_\theta(\boldsymbol{X}_t, t), \Sigma_\theta(\boldsymbol{X}_t, t)) \end{aligned} \tag{4}$$

In practice, we do not learn the variance and usually set it to $\Sigma_t = \beta_t \boldsymbol{I}$. Also, instead of learning the mean μ_θ directly, we learn to predict the noise ϵ_t in Equation (2). See [4] for how we can compute $\boldsymbol{X}_{t-1}$ given $\boldsymbol{X}_t$ and the predicted $\hat{\epsilon}_t$.

The training process of diffusion models is thus simple given Equation (3). At each step, we sample a batch of clean data $\boldsymbol{X}_0$ from the training set, timestamps t uniformly from $\{1, ..., T\}$, and random Gaussian noise $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We then create the noisy version of data $\boldsymbol{X}_t$ by applying Equation (3). A denoising model ϵ_θ is utilized to predict the noise via $\hat{\epsilon}_t = \epsilon_\theta(\boldsymbol{X}_t, t)$. The entire network is trained end-to-end via an MSE loss:

$$\mathcal{L}_{DM} = \mathbb{E}_{\boldsymbol{X}, t, \epsilon}[||\epsilon_t - \epsilon_\theta(\boldsymbol{X}_t, t)||^2] \tag{5}$$

Planning with Diffusion. Given a reward function $r(s_t, a_t)$, our goal of planning is to find the optimal sequence of actions $a_{0:T}$ that satisfies:

$$a^*_{0:T} = \underset{a_{0:T}}{\operatorname{argmin}} \sum_{t=0}^{T} r(s_t, a_t) = \underset{a_{0:T}}{\operatorname{argmin}}\, \mathcal{J}(s_0, a_{0:T}), \tag{6}$$

where $\mathcal{J}$ is an objective that characterizes the value of trajectories. Moreover, we denote the trajectory τ as:

$$\tau = \begin{bmatrix} s_0 & s_1 & \dots & s_T \\ a_0 & a_1 & \dots & a_T \end{bmatrix} \tag{7}$$

As demonstrated in Janner et al. [1], this planning process can be approximately subsumed into a generative modeling framework, and thus it can be decomposed into two modules: (i) a diffusion model $p_\theta(\tau)$ modeling the physically realistic of the trajectories; (ii) a reward constraint $h(\tau)$ characterized by a model $\mathcal{J}_\phi$. The $p_\theta(\tau)$ is a temporal UNet composed of $1D$ convolutions and trained with the denoising diffusion process. It is capable of iteratively generating realistic τ from random sampled noise. For the $\mathcal{J}_\phi$, it is trained separately to predict the cumulative rewards of the trajectories during the diffusion process as a signal indicator.

3.2 Improving Diffuser with Non-local Receptive Field

Since Diffuser [1] focuses on the local receptive field, it might fail to deal with the global context in the environment, especially for the complex scenes. To enable the model to have knowledge of the global context, we would like to perform an empirical study of injecting attention layers into the UNet architecture of the diffusion model. Here, we consider two different types of attention layers: self-attention [13] layer and squeeze-and-excitation (SE) [14] layer.

Self-attention layer. Given n query vectors, each with dimension d_q: $\boldsymbol{Q} \in \mathbb{R}^{n \times d_q}$, an attention function $\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ maps queries $\boldsymbol{Q}$ to outputs using m key-value pairs $\boldsymbol{K} \in \mathbb{R}^{m \times d_q}, \boldsymbol{V} \in \mathbb{R}^{m \times d_v}$,

$$\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_q}})\boldsymbol{V}, \tag{8}$$

where dot product $\boldsymbol{Q}\boldsymbol{K}^T \in \mathbb{R}^{n \times m}$ measures the similarity between queries and keys. $Softmax(\cdot)$ normalizes this similarity matrix to serve as the weight of combining the m elements in $\boldsymbol{V}$. And thus, given an input array $\boldsymbol{X} \in \mathbb{R}^{n \times d_x}$, a self-attention $\mathrm{SA}(\cdot)$ layer takes $\boldsymbol{X}$ as query, key, and value:

$$\mathrm{SA}(\boldsymbol{X}) = \mathrm{Attn}(\boldsymbol{X}, \boldsymbol{X}, \boldsymbol{X}) \tag{9}$$

Squeeze-and-excitation (SE) layer. Given an input array $\boldsymbol{X} \in \mathbb{R}^{n \times d_x}$, a SE layer employ a simple gating mechanism:

$$s = F_{ex}(X) = \sigma(W_2 \delta(W_1 z)), \tag{10}$$

where δ refers to the ReLU function, $W_1 \in \mathbb{R}^{\frac{d_x}{r} \times d_x}$ and $W_2 \in \mathbb{R}^{d_x \times \frac{d_x}{r}}$. And thus, an output of a SE layer is obtained by:

$$SE(X) = sX. \tag{11}$$

To make the receptive larger, we propose to inject these attention-based layers into the UNet of the Diffuser. We add the transformer layers at 3 places: (i) at the end of each block inside the encoder, (ii) at the bottleneck (middle) part of the UNet, (iii) at the end of each block inside the decoder.

3.3 State-conditioned Diffusion for Dynamic Environment

Besides the problem of the local receptive field, the Diffuser [1] only considers the static environments without any movable agents with the unconditional diffusion model. However, when it comes to dynamic environments, their strategy will fail. Thus, we formulate the planning in the dynamic environment in an auto-regressive manner. And for each step m, we predict the next step state s_{m+1} and action a_{m+1} conditioned on current state s_m. Figure 3 demonstrates the desired generation process of our proposed framework, where the denoising network is guided by S_m to synthesize s_{m+1} and a_{m+1}.

Inspired by previous works [15] with condition diffusion model, we choose to concatenation with the current state s_m (donated as $\hat{\boldsymbol{S}}$) to guide the denoising process towards generating the next step

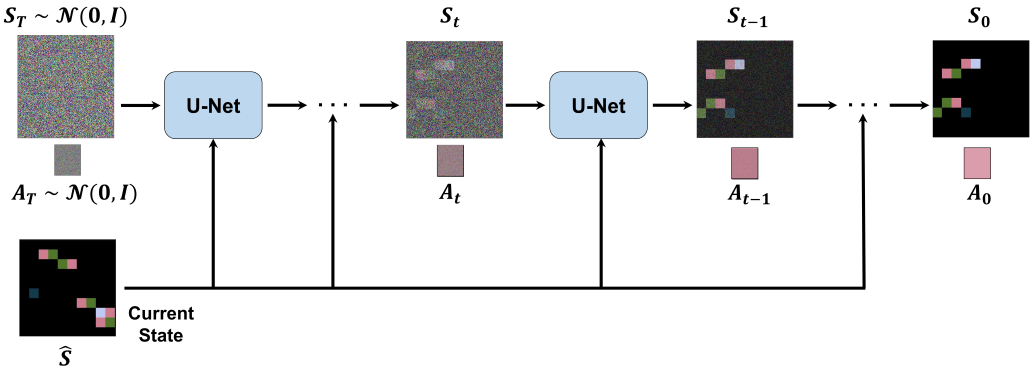


Figure 3: Overview of our condition diffusion model via iterative denoising. The input to each UNet is the concatenation of a noisy next state $\boldsymbol{S_t}$, a noisy next action $\boldsymbol{A_t}$, and the current state $\hat{\boldsymbol{S}}$. And the output is the noise ϵ_t which is used to obtain $\boldsymbol{S_{t-1}}$ and $\boldsymbol{A_{t-1}}$. Note that $\hat{\boldsymbol{S}}$ and $\boldsymbol{S}$ are first encoded by a pretrained VAE encoder, and we plot them as images for better visualization.

state s_{m+1} (donated as $\boldsymbol{S}$) and action a_{m+1} (donated as $\boldsymbol{A}$). Hence, $\boldsymbol{X}$ (see in Sec. 3.1) is equal to $[\boldsymbol{S}, \boldsymbol{A}]$ in our case.

As shown in Figure 3, we simply concatenate $\hat{\boldsymbol{S}}$, $\boldsymbol{S_t}$ and $\boldsymbol{A_t}$ as the input to the denoiser ϵ_θ, and predict the added noises ϵ and ϵ. The training loss in Equation (5) is thus as:

$$\mathcal{L} = \mathbb{E}_{(\boldsymbol{S},\boldsymbol{A},\hat{\boldsymbol{S}}),t,\epsilon}[||\epsilon_t - \epsilon_\theta(\boldsymbol{S}_t, \boldsymbol{A}_t, \hat{\boldsymbol{S}}, t)||^2] \tag{12}$$

Moreover, the Diffuser [1] only considers tasks with simple states that $1D$ arrays can represent, while the states of complex scenarios can not be trivially treated as $1D$ arrays. To address this difficulty, we propose to follow the latent diffusion model [16] to use a pretrained VAE [17] to encode the complex states into latent space. For example, for a state s represented by a 2D image, we can use a pre-trained encoder E to encode it as $e = E(s)$ to the latent space.

4 Experiments

The main purposes of our experimental evaluation are (1) to examine the performance of Diffuser [1] on dexterous hand manipulation tasks with high-dimensional action space, (2) to evaluate our Diffuser with attention-based layers by comparing it to the original diffusion planning algorithm, (3) to investigate the performance of our state-conditioned diffusion model on dynamic scenarios with multiple agents.

4.1 Experimental Setup

D4RL. We first follow diffusion to use the D4RL [3] locomotion benchmark, including **HalfCheetah** and **Walker2d** to evaluate our study of the non-local receptive field (see Subsection 3.2). We also extend the Diffuser [1] to the environment with high-dimensional action space – **Adroit**, a 24-DoF ShadowHand, to perform a dexterous manipulation task as a high-dimensional continuous action space environment.

MinAtar. The MinAtar [2] project implements simplified versions of several Atari 2600 games. Specially, we use **Asterix**. In Asterix, the agent can move freely in all four cardinal directions. Treasures and enemies spawn on the side of the environment and move vertically at a fixed speed. The agent's goal is to pick up as many treasures as possible. The episode terminates if the agent makes contact with an enemy. For each timestep, the state is represented by a $10 \times 10 \times 3$ image. Since there is no expert setting in MinAtar, we adopt a pretrained DreamerV2 [18] agent to generate the pseudo ground truth trajectories. Episodes that are unsuccessful are deleted from our training set. We use this dynamic, multi-agent environment to evaluate our state-conditioned diffusion (see Subsection 3.3).

4.2 Evaluation of hand dexterous manipulation tasks

The authors of Diffuser [1] evaluated their method only in the `HalfCheetah`, `Hooper`, and `Walker2d` environments. In this section, we have experimented with the Diffuser model in hand dexterous manipulation tasks in the Adroit environment. In order to do that, we have modified the implementation code provided by the authors and extended it to more complex environments since it has several runtime issues in environments other than the three ones mentioned. To be specific, the Diffuser's original code does not take the actions produced by a policy to render the samples that come from the dataset, the diffuser, when training and planning. Instead, it sets environments' states directly to the samples generated by the Diffuser model. In other words, the authors assumed that the environments are fully observable while some are not. This assumption causes runtime errors in environments with partial observations like the Adroit domain. For instance, in the relocate-v0 task in Adroit, sampling the Diffuser planner generates the position of the hand joints, hand palm, object, and target as an observation in each time step which has a shape of 39 in total. However, to set the state of the environment, we need the position and velocity of all hand joints, which has a shape of 72.

This incompatibility in observation size happens in rendering the samples come from three situations: (1) the dataset, (2) the Diffuser when training, and (3) the Diffuser when planning. To render trajectories in the D4RL dataset [3], we use the additional information provided by the dataset, like joints' positions and velocities, to set the environment states. To solve the rendering problem for the Diffuser planner in training, we add the joints' velocities as input and output. Accordingly, the Diffuser is trained using the full state of the environment. Moreover, to render the planned trajectories of hand dexterous manipulation tasks, we use the actions generated by the planner and take them into the environment. Then, we use the `renderer` object of the Mujoco simulator instead of setting the state of the environment directly.

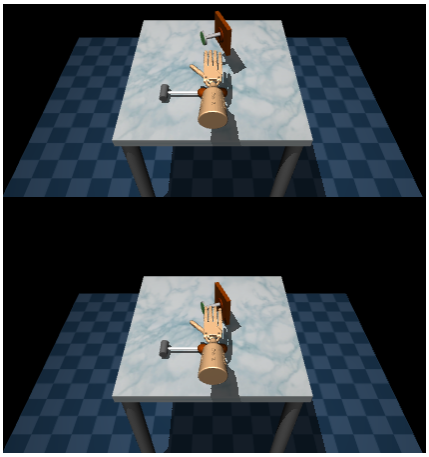
We evaluate four hand dexterous manipulation tasks in the Adroit domain: `relocate-v0, pen-v0, hammer-v0`, and `door-v0` and use the expert trajectories from the D4RL dataset [3]. Figure 4 shows the sample of hand dexterous manipulation tasks generated by the Diffuser planner. The `relocate-v0` task has been trained via 10^6 iterations and 100 epochs and sampled per 20000 steps with 20 number of diffusion steps. The `door-v0, pen-v0` and `hammer-v0` have been trained via 10^4 iterations and 100 epochs and sampled per 1000 steps. The total reward and score (normalized reward) are shown in table 1.

Dataset	Task	Total reward	Score
Expert	Relocate-v0	99.1	59.11
	Pen-v0	-11.93	43.2
	Door-v0	-129.86	-29
	Hammer-v0	9.43	27.74

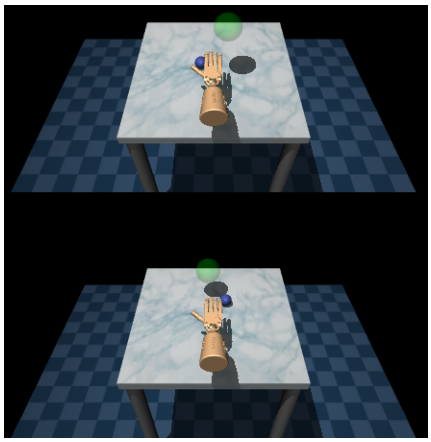
Table 1: Total reward and score of hand manipulation tasks in Adroit domain.

4.3 Evaluation of injecting non-local receptive field

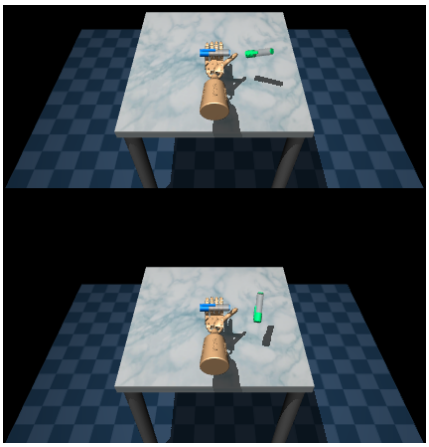
We empirically study the influence of injecting a non-local receptive field into the Diffuser. In this study, we adopt the **score** as the metric and use the Diffuser [1] as our baseline. Table 2 shows the quantitative results of our study. The SE variant achieves the best performance, and the SA variant performs poorly. We conjecture that the complexity of SA layers is high. Thus, it is not suitable for the locomotion benchmark. And the SE layers' complexity is limited by the gated function, and thus this variant outperforms the baseline.



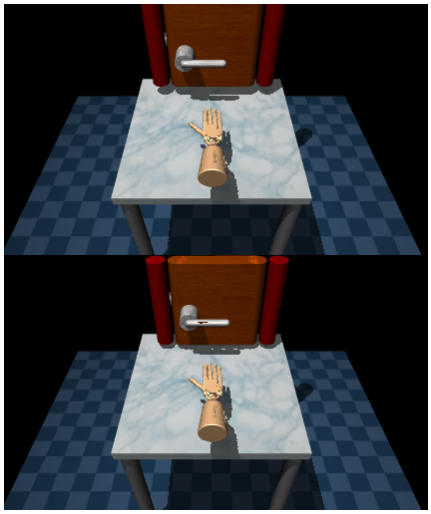
(a) `hammer-v0` task



(b) `relocate-v0` task



(c) `pen-v0` task



(d) `door-v0` task

Figure 4: Samples of Adroit tasks generated by the diffuser planner

Dataset	Environment	Diffuser	Diffuser + SA	Diffuser + SE
Medium-Expert	HalfCheetah	88.9	89.66	**91.14**
	Walker2d	106.4	106.2	**107.1**

Table 2: Quantitative comparison between Diffuser our non-local receptive field variants. Our SE version achieves the best performance. SA indicates the self-attention layers, and SE indicates the

4.4 Evaluation on dynamic environment

We then evaluate our method on the dynamic environment Asterix from the MinAtar project. We random sample 1024 $\{s_m, (s_{m+1}, a_{m+1})\}$ pairs and measure the accuracy of our predicted action $\hat{a}_{m+1}$. Since Diffuser is unable to be applied to this environment, we mainly experiment with different settings of our model.

Table 3 shows the model performance in terms of accuracy. Predicting the next state significantly improves the performance, indicating that predicting the future is helpful for planning (action prediction). In addition, removing the SE layers leads to slightly worse results, which is same for the observation from Sec.4.3.

Method	Accuracy (%) ↑
Ours (Full Model)	**72.85**
Without predicting next state	63.66
Without SE layers	67.65

Table 3: Results on Asterix. Predicting the next state and adding SE layers can improve the performance.

5 Discussion

Limitations. We extended the Diffuser [1] to be able to handle dynamic environments and evaluated it on Asterix from MinAtar [2]. In the future, we would like to further validate the viability of our method by evaluating it on additional tasks from the MinAtar project like Freeway. However, the environments in MinAtar are not extremely complex, and the results of the Adroit tasks are not acceptable. Thus, we plan to try more advanced technologies in the field of diffusion models, such as using cross-attention for the condition to apply our method to CARLA [19].

The fully-observable assumption in the original implementation code is another limitation of the Diffuser planner. As shown in table 1, the score of the `door-v0` and `hammer-v0` are significantly lower than the other environments, which implies that extending to more complex environments by modifying the input and output shape of the Diffuser is not the perfect solution.

Conclusion. We extend the implementation of Diffuser [1] to be compatible with the D4RL Adroit domain [3]. This allows us to evaluate Diffuser on multiple complex dexterous manipulation tasks with a high-dimensional, continuous action space.

In addition, we propose to extend Diffuser with attention-based layers, which allows it to better deal with environments where a global, non-local view is beneficial. We conclude that the use of squeeze-and-excitation layers increases accuracy substantially. However, self-attention layers are not particularly advantageous.

Finally, we note that Diffuser is limited to static environments. To overcome this limitation, we introduce a state-conditioned diffusion model for dynamic environments. Our experiment shows that our agent works in a dynamic environment and performs well, especially when utilizing squeeze-and-excitation layers in addition to state conditioning.

References

[1] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, 2022.

[2] K. Young and T. Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

[3] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020.

[4] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[5] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

[6] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021.

[7] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models. In *NeurIPS*, 2022.

[8] W. Harvey, S. Naderiparizi, V. Masrani, C. Weilbach, and F. Wood. Flexible diffusion modeling of long videos. In *NeurIPS*, 2022.

[9] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[10] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *CoRR*, abs/2209.14988, 2022.

[11] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin. Magic3d: High-resolution text-to-3d content creation. *arXiv preprint arXiv:2211.10440*, 2022.

[12] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *CoRR*, abs/2205.11487, 2022.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurPIS*, 2017.

[14] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[15] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. Image super-resolution via iterative refinement. *T-PAMI*, 2022.

[16] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[17] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[18] D. Hafner, T. P. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. In *ICLR*, 2021.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *CORL*, 2017.