A Algorithm Details

A.1 Pick-up Strategies for the Online Adaptation

In our training methodology, we have designed different pick-up strategies for handling different importance indicators, recognizing their unique properties and implications for the learning process.

For the TD-error indicator, a dynamic approach is employed due to the variable nature of its importance, which can change as the critic function updates during training. To manage this, a heap structure is used to maintain a buffer of 50,000 samples. This buffer is continuously updated to reflect the current importance of each sample, ensuring that even those with initially low importance are retained for potential future relevance. When it comes time to update the diffuser, a selection of 5,000 high-importance samples is drawn from this buffer, aligning the update process with the most pertinent data at that moment.

In contrast, the strategy for reward importance takes into account its static characteristic – the importance of these indicators does not change throughout the training. Here, a smaller buffer of 5,000 samples is employed, updated similarly based on importance. However, to counter the risk of high-importance samples perpetually dominating the buffer, a rotation system is implemented. After each update, some samples, particularly the older ones, are dropped. This system ensures that each sample contributes to the diffuser update only a limited number of times, thus maintaining a fresh and current dataset for training, reflective of the latest environmental interactions.

These tailored strategies highlight a nuanced understanding of how different indicators behave and affect the learning process, ensuring that both dynamic and static aspects of the training data are optimally utilized for updating the diffuser.

A.2 State-level Trajectory Generation

In the state-level generation, we directly generate trajectories consisting of states and actions, $\{s_t, a_t, s_{t+1}, a_{t+1}, \dots\}$. For the architecture of the diffusion model used in the state-level generation, we directly refer to the architecture used in Lu et al. (2023), while we omit the generation of the reward. Meanwhile, we extend the size of the network used from 24 to 128 to support the larger result of a trajectory.

A.3 Renderer and Recognizer

In the image-level generation, a renderer and a recognizer are employed for image rendering and key point detection tasks, respectively. The renderer, based on previous work (Lu et al., 2022; Hansen et al., 2022; Seo et al., 2022), captures observations in an environment where an AI agent operates. The images rendered, set at a resolution of 84×84, serve as input of the generation model.

The recognizer is a neural network consisting of two components. The first component utilizes a CNN to extract features from the images. In the second component, separate 5-layer MLP networks are used to determine the 3D positions of specific key points, which are selected based on the requirements of different tasks. This two-component design of the recognizer allows for efficient processing of the images, first by identifying relevant features through CNN and then pinpointing key points' positions using MLP networks, so that we can calculate the proprioceptive state.

B Details of Experiment setup

B.1 D4RL

We basically consider 3 different environments from D4RL (Fu et al., 2020). We use the original offline dataset from D4RL (Fu et al., 2020).

AntMaze. This domain steps up the complexity by replacing the 2D ball in Maze2D with an 8-DoF "Ant" quadruped robot, adding a layer of morphological complexity. It is a navigation domain that closely resembles real-world robotic navigation tasks. We followed the design of a sparse 0-

Task Name	Subtasks
`Multitask`	`evaluation task`: Push
`Sequencetask`	Sweep, Sweep Into, Coffee Push, Box Close, Push Wall, Peg Insert Side, Basketball, Soccer,

Table 1: **Meta-World task setting.** We select push task as the evaluation task for the multitask environment. We arrange all the 8 *medium* to form the Sequencetask environment.

1 reward system in this environment, activated only upon reaching the goal, to test the stitching challenge under more complex conditions.

Locomotion. Comprising tasks like Hopper, HalfCheetah, and Walker2d, the Locomotion domain is a staple in offline deep RL benchmarks. We used the same datasets in D4RL (Fu et al., 2020) for consistency with previous studies, and also experimented with a variety of datasets to observe the effects of different data qualities.

Kitchen. This environment involves controlling a 9-DoF Franka robot in a kitchen setting, interacting with everyday household items like a microwave, kettle, cabinets, an overhead light, and an oven. Each task aims to achieve a specific goal configuration, like opening the microwave and sliding cabinet door, placing the kettle on the burner, and turning on the overhead light. The Kitchen domain serves as a benchmark for multitask behavior in a realistic, non-navigation setting. Here, the stitching challenge is amplified due to the complexity of the trajectories through the state space, compelling algorithms to generalize to unseen states rather than rely solely on training trajectories.

B.2 Meta-World

The Meta-World benchmark (Yu et al., 2019) is a comprehensive suite designed for evaluating and advancing reinforcement learning and multi-task learning algorithms. It features 50 distinct robotic manipulation tasks, offering a diverse and challenging environment for testing the ability of algorithms to generalize and quickly acquire new skills. By providing a broad range of tasks, Meta-World seeks to address the limitations of existing benchmarks that focus on narrow task distributions. This benchmark is instrumental in fostering research that moves towards meaningful generalization, enabling algorithms to effectively learn multiple tasks and adapt to entirely new behaviors, which is crucial for the practical application of reinforcement learning in dynamic real-world scenarios. Following previous work (Hansen et al., 2022), we selected a total of 15 tasks from Meta-World based on their difficulty according to one previous work (Seo et al., 2022), which categorizes tasks into *easy*, *medium*, *hard*, and *very hard* categories. Same as Hansen et al. (2022), we discard *easy* tasks and select all tasks from the remaining 3 categories. Our approach differs from previous studies (Hansen et al., 2022; Seo et al., 2022) in that we solely utilize proprioceptive state information as input, rather than RGB frames or combined state information. This choice allows for easier application to general reinforcement learning methods. Following the previous settings, we use a sparse-reward signal that only provides a reward of 1 when the current task is solved and 0 otherwise. For the success criteria, we follow the original setting in Meta-World (Yu et al., 2019). Table 1 offers a detailed setting of our multitask and sequence task.

Training Dataset. The dataset is acquired by training an RL agent with SAC on each single task. For the multi-task setting, the entire dataset is combined by all 14 single tasks. For the sequence task, the entire dataset is combined by all 8 single tasks, without any trajectory change.

C Additional Experimental Results

This section includes additional experimental results, which are ablation studies and detailed results that help to better understand the properties of different methods and components.

C.1 Offline Experiments

In this section, we present experiments designed to evaluate the performance of our ATraDiff model in an offline setting.

Comparison with transition-level generation SynthER. Our objective is to verify whether ATraDiff can match or surpass the results achieved in previous research, specifically referencing the work SynthER (Lu et al., 2023). To ensure a fair and accurate comparison, we meticulously replicated the experimental settings used in Lu et al. (2023).

Our testing ground is the D4RL Locomotion benchmark, as detailed in (Fu et al., 2020). We also extend the original dataset to $5M$ samples, following the settings in Lu et al. (2023).

The findings, as outlined in Table 2, are quite revealing. They indicate that our ATraDiff method generally outperforms SynthER across various datasets. This performance improvement is particularly pronounced in high-quality datasets, underscoring the effectiveness of our approach in generating helpful trajectories and the benefits of trajectory-level generation over transition-level generation.

Learned reward predictor. Furthermore, we introduce a reward predictor to avoiding accessing the ground-truth reward function. Following previous work (Konyushkova et al., 2020), we use a supervised learning from demonstrations, by minimizing the loss:

$$L_{\text{sup}}(D_0) = \mathbb{E}_{s_t \sim D_0}[r_t - R(s_t, a_t, s_t')]^2,$$

where D_0 represent the demonstration dataset with reward label, (s_t, a_t, s_t') is the state with reward label r_t, and $R(s_t, a_t, s_t')$ is the learned reward function.

Results shown in Table 3 illustrate that with the reward predictor, our method could still improve the performance of offline RL methods, but slightly worse than the result with ground-truth reward.

Task Name	TD3+BC	TD3+BC+SynthER	TD3+BC+ours	IQL	IQL+SynthER	IQL+ours
halfcheetah-random	11.3	12.2	12.5	15.2	17.2	17.1
halfcheetah-medium	48.1	49.9	52.3	48.3	49.6	53.1
halfcheetah-replay	44.8	45.9	46.5	43.5	46.7	49.2
halfcheetah-expert	90.8	87.2	93.6	94.6	93.3	95.2
hopper-random	8.6	14.6	15.2	7.2	7.7	8.1
hopper-medium	60.4	63.4	65.7	62.8	72.0	72.4
hopper-replay	64.4	53.4	64.7	84.6	103.2	103.6
hopper-expert	101.1	105.4	111.2	106.2	90.8	113.6
walker-random	0.6	2.3	2.1	4.1	4.2	4.3
walker-medium	82.7	84.8	87.5	84.0	84.7	89.1
walker-replay	85.6	90.5	86.3	82.6	83.3	85.4
walker-expert	110.0	110.2	111.2	111.7	111.4	111.7

Table 2: Results of the offline experiments on the D4RL Locomotion benchmark (Fu et al., 2020). We show that ATraDiff outperforms SynthER on almost all the tasks and dataset.

Task Name	TD3+BC	TD3+BC+ours	TD3+BC+ours_w/ _learned_reward	IQL	IQL+ours	IQL+ours_w/ _learned_reward
halfcheetah-random	11.3	12.5	11.7	15.2	17.1	16.8
halfcheetah-medium	48.1	52.3	49.3	48.3	53.1	50.2
halfcheetah-replay	44.8	46.5	45.3	43.5	49.2	45.6
halfcheetah-expert	90.8	93.6	91.9	94.6	95.2	94.5
hopper-random	8.6	15.2	14.8	7.2	8.1	8.3
hopper-medium	60.4	65.7	64.5	62.8	72.4	72.2
hopper-replay	64.4	64.7	65.2	84.6	103.6	97.0
hopper-expert	101.1	111.2	108.7	106.2	113.6	108.3
walker-random	0.6	2.1	3.4	4.1	4.4	4.0
walker-medium	82.7	87.5	85.0	84.0	89.1	85.3
walker-replay	85.6	86.3	87.5	82.6	85.4	84.9
walker-expert	110.0	111.2	111.1	111.7	111.7	111.6

Table 3: Ablation study on the reward predictor. We show that with the learned reward predictor, our ATraDiff can still improve the performance of the offline RL methods.

C.2 Ablation Studies

Effect of different task prompts. Here we show that different task prompts have noticeable impacts on performance. We conduct experiments on the Meta-World environments (Yu et al., 2019) to test the effect of different design choices for the task prompts, focusing on how to represent the task. We include three different strategies, language task prompt, one-hot task prompt, and no prompt, where the one-hot prompt simply uses a one-hot vector to represent different tasks. The result shown in Figure 9 demonstrates that different task prompts indeed affect the performance.

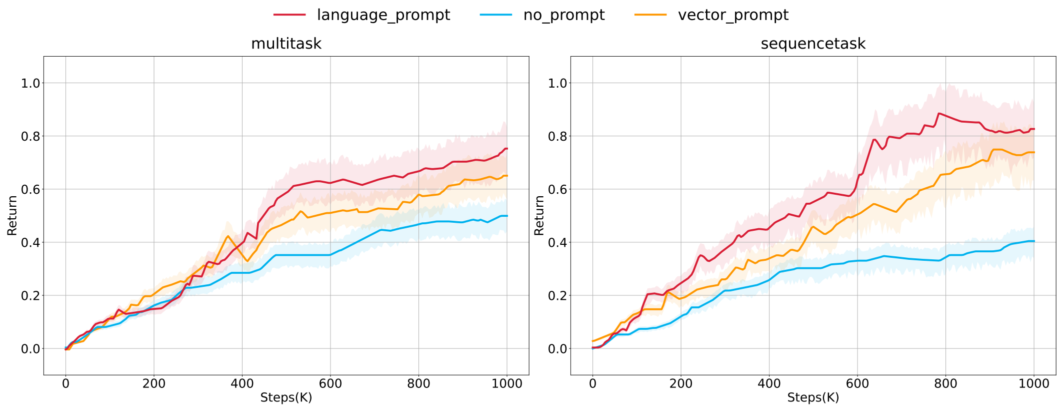


Figure 9: Ablation study on the design choices of task prompt. Different types of task prompts lead to varied behavior and performance.

Effect of the random dropping strategy. Here we show that the random dropping strategy is very important in the online adaptation process with the total reward indicator. In our online adaptation phase, when using the total reward importance indicator, the importance of any trajectory will never change during the whole training process. Hence, trajectories with high importance might always be used to update the generator, which inspired us to introduce a random dropping strategy. The ablation study conducted on the D4RL Locomotion Environments (Fu et al., 2020) shown in Figire 10 illustrates that the random dropping strategy can significantly improve the performance.

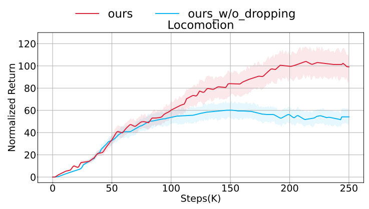


Figure 10: Ablation study on the random dropping strategy. The strategy can indeed improve the performance.

C.3 Full Results

Here we present the full results for D4RL Locomotion with all 4 different offline datasets. Figure 11 shows that our ATraDiff is comparable to or better than the original RL baseline method with low-quality data (like random data), and the performance gap between our ATraDiff and the baseline becomes much more pronounced with medium/high-quality data (like medium-expert data).

C.4 Training Time

The experiments are conducted on a single NVIDIA RTX 4090TI GPU. The training time of ATraDiff is listed in Table 4.

	Walker2D	Hopper	Halfcheetal
SAC	6.5h	7h	7h
SAC with ATraDiff	15h	17h	16.5h
RDEQ	6h	6.5h	6.5h
RDEQ with ATraDiff	14.5h	16.5h	16h

Table 4: Time cost (hours) of training SAC and RDEQ with and without ATraDiff.

C.5 Online Experiments for SynthER

We now include comparisons between our method and SynthER (Lu et al., 2023) in the online setting. We have conducted experiments on the Meta-World Environments. This experiment is

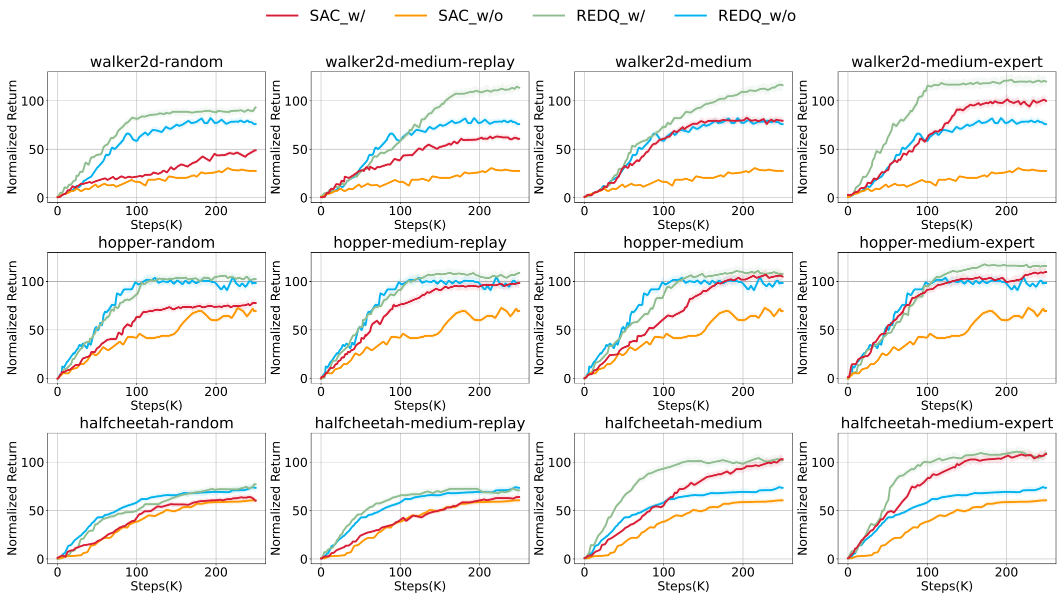


Figure 11: Learning curves on the D4RL Locomotion benchmark. ATraDiff (denoted as ‘w/’) is comparable to or better than the original RL baseline method with low-quality data, and the performance gap between ATraDiff and the baseline becomes much more pronounced with medium/high-quality data.

crucial to demonstrate the adaptability and effectiveness of our approach in dynamic and complex settings. For a fair comparison, we treat SynthER same as our method, combining it with RL methods using the method in Section 4.2. The results shown in Figure 12 indicate a significant performance advantage of our method over SynthER in these online environments.

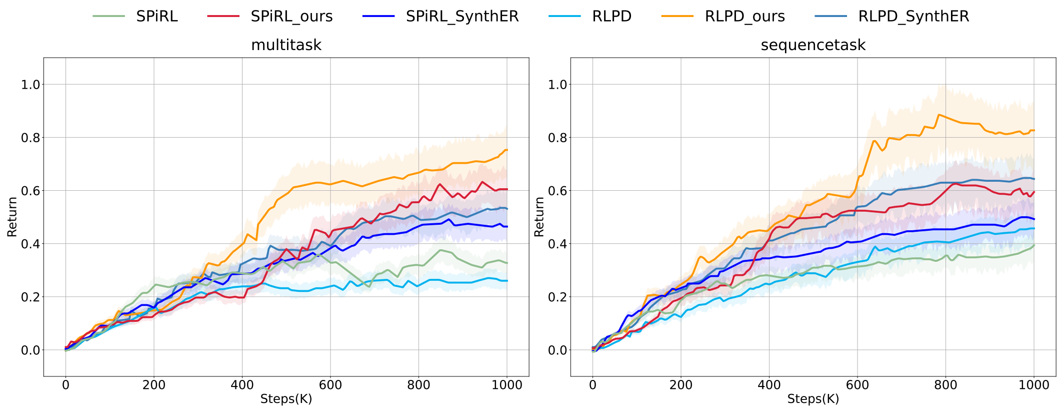


Figure 12: Experiment on the Meta-World benchmark. ATraDiff performs significantly better than SynthER on complicated environments.

C.6 Reward Predictor Under Online Settings

To verify that the learned reward predictor in Section C.1 also works under the online reinforcement learning environments, we conduct experiments that compare SAC and SAC combined with our method with a learned reward predictor. The experiments are done in 3 D4RL environments, with the medium-expert datasets. The results in Figure 13 show that our method with the reward predictor also improves the performance of RL methods under the online setting.

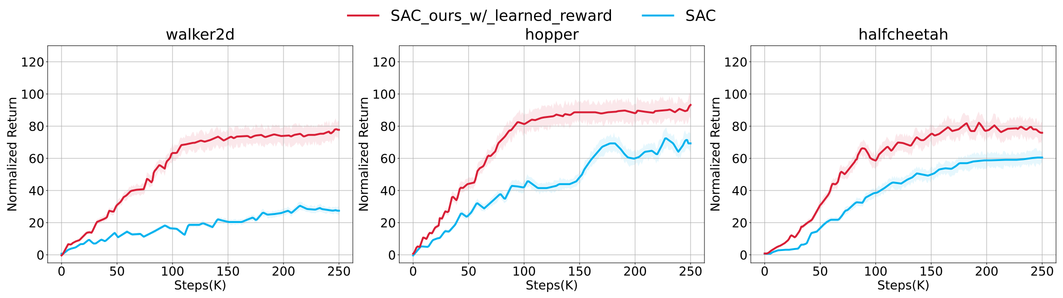
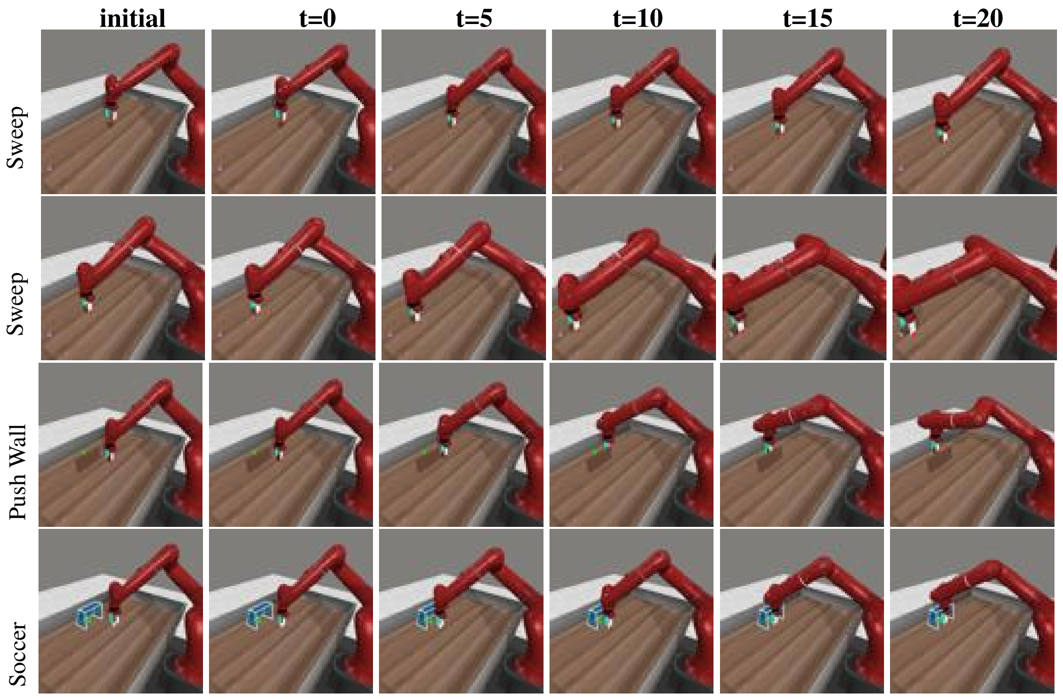


Figure 13: Experiments on the learned reward predictor in online reinforcement learning. Our method with the reward predictor still significantly improves the performance of online RL methods.

D Visualizations

In this section, we show some representative visualizations of our generated image trajectories. We present a variety of images generated under different conditions, including varying initial states and a range of tasks. The labels in the left of the figures represent the task. The results demonstrate the robustness and effectiveness of our image generation method, consistently performing well across diverse scenarios and producing high-quality and temporally coherent image trajectories.


initial
t=0
t=5
t=10
t=15
t=20
Sweep
Sweep
Push Wall
Soccer