Learning Multimodal Behaviors from Scratch with Diffusion Policy Gradient

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep reinforcement learning (RL) algorithms typically parameterize the policy as a deep network that outputs either a deterministic action or a stochastic one modeled as a Gaussian distribution, hence restricting learning to a single behavioral mode. Meanwhile, diffusion models emerged as a powerful framework for multimodal learning. However, the use of diffusion policies in online RL is hindered by the intractability of policy likelihood approximation, as well as the greedy objective of RL methods that can easily skew the policy to a single mode. This paper presents Deep Diffusion Policy Gradient (DDiffPG), a novel actor-critic algorithm that learns *from scratch* multimodal policies parameterized as diffusion models while discovering and maintaining versatile behaviors. DDiffPG explores and discovers multiple modes through off-the-shelf unsupervised clustering combined with novelty-based intrinsic motivation. DDiffPG forms a multimodal training batch and utilizes mode-specific Q-learning to mitigate the inherent greediness of the RL objective, ensuring the improvement of the diffusion policy across all modes. Our approach further allows the policy to be conditioned on mode-specific embeddings to explicitly control the learned modes. Empirical studies validate DDiffPG's capability to master multimodal behaviors in complex, high-dimensional continuous control tasks with sparse rewards, also showcasing proof-of-concept dynamic online replanning when navigating mazes with unseen obstacles.

1 Introduction

Reinforcement learning (RL) for continuous control has experienced significant advancements during the last decade, reshaping its applicability to domains like game playing [65, 27, 4], robotics [26, 33], and autonomous driving [72, 12]. However, most RL algorithms choose to parameterize policies as deep neural networks with deterministic outputs [47, 21] or Gaussian distributions [62, 25], limiting learning to a single behavior mode. Moreover, the standard exploration-exploitation schemes can easily make a policy greedy towards one mode, in which the algorithm keeps exploiting to maximize its objective. The aforementioned issues hinder the possibility of training agents that successfully solve a task while showcasing versatility of behaviors—a property intuitive to intelligent systems like humans, who can exhibit resourcefulness for completing a task, even amidst unprecedented events.

Learning a multimodal policy has several practical applications. First, such a policy that learns many solutions to a task is useful when acting in non-stationary environments. Imagine that a routine path from the office to home is unexpectedly blocked; one needs to choose an alternate route. A policy that encompasses multiple solutions can better navigate such changing conditions, offering flexibility for replanning or serving as prior for hierarchical RL. Second, while searching for diverse solutions, a multimodal policy continues exploring even after finding a viable solution, thereby facilitating agents to escape local minima. Additionally, multimodal policies hold great promise for continual learning

Submitted to the 18th European Workshop on Reinforcement Learning (EWRL 2025). Do not distribute.

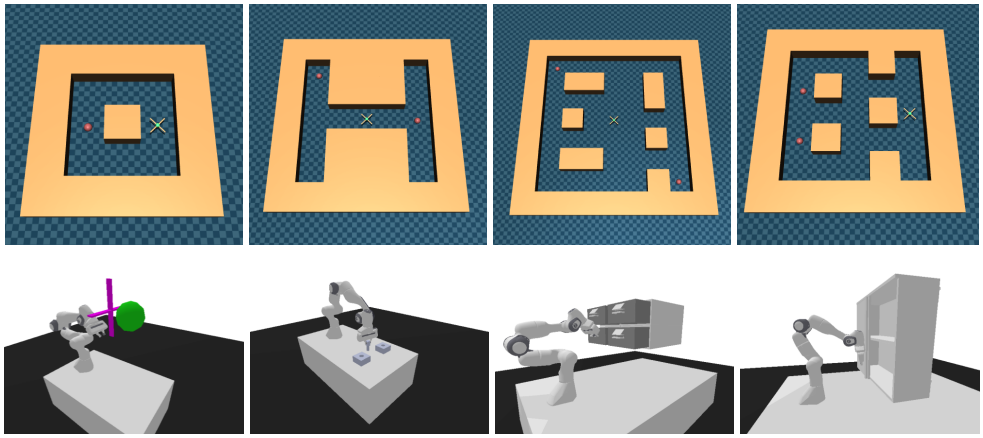


Figure 1: We design (Top) four AntMaze tasks; *AntMaze-v1*, *AntMaze-v2*, *AntMaze-v3*, *AntMaze-v4*, and (Below) four robotic tasks *Reach*, *Peg-in-hole*, *Drawer-close*, and *Cabinet-open* that have a high degree of multimodality.

scenarios; they can effectively parameterize complex action distributions when new skills or solutions are introduced, potentially mitigating the issue of catastrophic forgetting [59].

Recently, Diffusion Models [30, 68], a novel class of generative models acclaimed for impressive image generation results [38, 60], have been proposed as a powerful parameterization for policy learning. They have been extensively applied in the areas of learning from demonstrations [15, 58], offline RL [73, 71], and learning for trajectory optimization [34, 46]. One advantage of training diffusion policies *offline* is its ability to model multimodal datasets, which usually comprise trajectories stemming from suboptimal policies or various human demonstrations. However, only a few studies explored these models for *online* RL, which focus on the formulation of the training objectives for policy optimization via diffusion and showcase improved sample efficiency [55, 74, 17]. Notably, none of them studied the inherent multimodality within diffusion policies, nor have they explicitly tackled the challenge of exploration for discovering and learning multiple behavioral modes online.

In this paper, we introduce *Deep Diffusion Policy Gradient* (DDiffPG), a novel actor-critic algorithm for training multimodal policies parameterized as diffusion models from scratch. Specifically, we aim to learn a policy that is capable of employing various strategies to accomplish a task. Unlike existing multimodal methods that condition on latent variables, DDiffPG emphasizes the explicit discovery, preservation, and improvement of behavioral modes, and for that, we decouple exploration and exploitation. For exploration, we apply novelty-based intrinsic motivation and utilize an unsupervised hierarchical clustering approach to discover modes, being a priori agnostic to the possible number of modes. For exploitation, we introduce mode-specific Q-functions to ensure independent improvements across modes and construct multimodal data batches to preserve the multimodal action distribution. We empirically evaluate our method on high-dimensional and continuous control tasks with sparse rewards, comparing to SoTA baselines, verifying DDiffPG's capbility to master multimodal behaviors. Additionally, we demonstrate the usability of our multimodal policies in an online replanning application in non-stationary mazes with unseen obstacles.

Our paper makes **four contributions**. First, we introduce **diffusion policy gradient**, a novel way to train diffusion models following the RL objective. Second, we present DDiffPG, a **new actor-critic algorithm for training diffusion policies from scratch** while discovering and preserving multimodal behaviors. Third, we achieve **explicit mode control** by policy conditioning on a mode-specific latent embedding during training, shown to be beneficial for online replanning. Finally, we design a series of **new challenging robot navigation and manipulation tasks** with a high degree of multimodality, serving as a testbed for multimodal policy learning, as shown in Fig. 1.

2 Related Work

Policy learning via reinforcement learning for continuous control has exploded thanks to improved algorithms learning complex skills with *online* RL [25, 5], while significant effort has been made to provide improved methods for *offline* RL to take advantage of demonstrations and fixed datasets [40,

44]. However, the policy parameterization in both settings considers policies that can only model a single behavioral mode, e.g., by having a deterministic output [66, 5] or by learning a Gaussian distribution. Recently, the use of transformer models enabled learning for control through offline data as sequence modelling [14, 57, 31], which, through language conditioning [7, 23], can handle multi-goal behavior generation; though, it is unclear if these models can explicitly exhibit behavior diversity per goal. Similarly, goal-conditioned RL methods condition policies to various goals, but each goal-conditioned policy suffers from the single-mode modeling, prevalent in typical deep RL algorithms [2, 52, 10, 53].

Diffusion policy as universal skill representation Diffusion models [30] have recently emerged as a promising parameterization for robust and multimodal robot learning. Early works used diffusion models for trajectory optimization [34, 69, 46], and as a policy for behavioral cloning [15]. Diffusion policy has soon shown its power as a universal skill representation [76] that allows learning complex landscapes of multimodal behaviors [11, 28], goal-conditioned [58, 61] or language-conditioned ones [24, 13]. Further, the expressivity of diffusion models has proven beneficial for offline RL [73, 71]. Nevertheless, the approaches for online RL with diffusion policy optimization are scarce [55, 74, 17], while these works do not study multimodality preservation within diffusion models, nor have they explicitly tackled the challenge of learning multimodal policies online. In this work, we present a new framework that takes advantage of diffusion policy as a universal skill representation model and introduces an algorithm that allows multimodal policy learning from scratch.

Unsupervised skill discovery Multimodal behavior learning has been addressed through skill discovery approaches [9, 42, 32, 54, 36], either from offline data [64] or through unsupervised RL [19, 41], which usually exploit variational inference for mode/skill discovery [43] with policy conditioning on latent vectors, or by forming rewards or goals to be achieved by different low-level policies [39, 49]. Hierarchical learning methods, e.g., relying on options learning [3, 1], usually depend on state-specific mode discovery that conditions a low-level policy or triggers different skills [45, 35]; but the policy usually reaches each goal greedily without exhibiting versatile behaviors.

3 Diffusion Policy Gradient

It is not straightforward to apply training approaches of popular deep RL methods to train a diffusion policy online. First, it is practically intractable to approximate the policy likelihood [37]. Second, directly backprogating Q-values to the diffusion policy, like Q-learning methods [25, 47], is not viable—the Markov chain may lead to vanishing gradients [55].

To solve these issues, we first present diffusion policy gradient, a novel approach to training diffusion policies with the RL objective that is also suitable for learning multimodal behaviors. Similarly to DPG [66], we compute $\nabla_a Q(s, a)$ given a state s and action a. However, we choose not to directly use the action gradient to optimize the policy, as this leads to vanishing gradients and instability. We rather obtain a target action a^{target} via $a^{target} \leftarrow a + \eta \nabla_a Q(s, a)$, η being a suitable learning rate. We can, then, train the diffusion policy based on the transitions in the datasets $\mathcal{D}$ using a behavioral cloning (BC) objective

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{t\sim[1,T],(a_0^{target},s)\sim\mathcal{D},\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \|\epsilon - \epsilon_\theta(a_t^{target}, s, t)\|, \tag{1}$$

where $t \sim [1, T]$ refer to the diffusion step, and ϵ is the diffusion noise — see Appx. A for a brief introduction to Diffusion Models.

Implementation For every data collection, we store the transition $(s, a, a^{target}, r, s')$ in the buffer $\mathcal{D}$, where a^{target} is initialized as a. For every policy update, we sample the state-action pair (s, a^{target}), obtain a new a^{target} via gradient ascent, update the policy with equation 1 and replace the new a^{target} in the buffer. Thus, our policy update does not change the training objective of the diffusion model and avoids the vanishing-gradient problem. Note that the action a^{target} that is used to update the policy is not inferred from the current policy, which provides an opportunity to control the behaviors of the policy in an off-policy fashion and lay the foundation for learning multimodal behaviors later.

Interpretation The conventional diffusion model is trained using a dataset with supervised labels [30]. In *offline* decision-making, a diffusion policy predicts actions given states with the dataset providing both the state and the corresponding ground-truth action as the target (label). In contrast, our *online* setting does not provide a predefined ground-truth action but requires the discovery of good

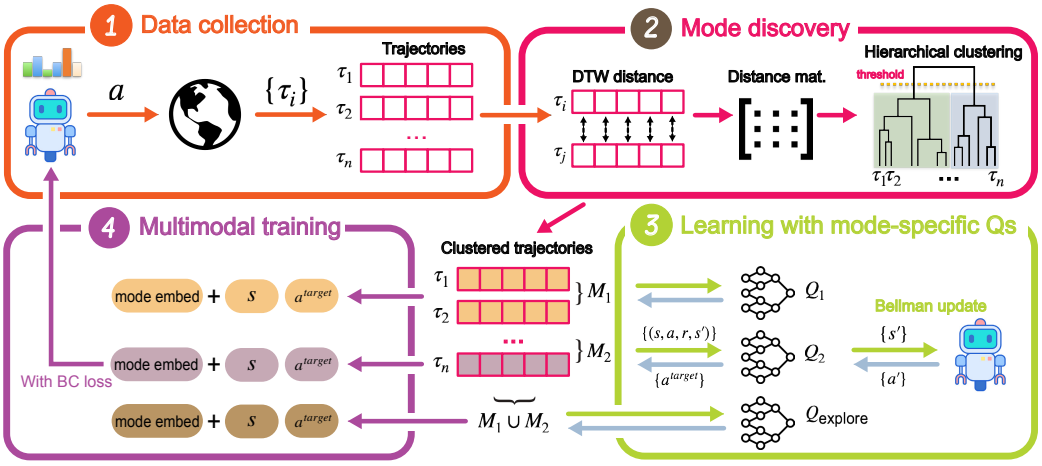


Figure 2: Overview of DDiffPG: (1) the agent interacts with the environment and collects a set of trajectories $\{\tau_i\}$. (2) Given a set of goal-reached trajectories, a DTW distance matrix is computed and used for hierarchical clustering to discover modes. (3) Each mode is associated with a set of trajectories, which is used exclusively to train mode-specific Q-functions and an exploration-specific $Q_{\texttt{explore}}$. (4) A multimodal batch is constructed by concatenating (s, a^{target}) pairs sampled from every mode and used for the diffusion policy update.

actions. To adapt without altering the supervised training framework of conventional diffusion models, we generate a^{target} and consider it as the target. The action a^{target}, derived through gradient ascent, represents an improved choice based on the current Q-function. During training, we additionally store a^{target} in the replay buffer and continuously update it based on its preceding values, ensuring a continuity of learning. Intuitively, we are chasing the target by replacing it with the newly computed a^{target} in this *online* RL setting rather than learning a static target as in the *offline* setting.

Relation to related works Our formulation is most closely related to DIPO [74] but has two key differences. First, while DIPO also performs gradient ascent on the action, it replaces a^{target} with the original a from the buffer rather than retaining an additional a^{target}. Therefore, the transition no longer aligns with the current MDP dynamics and reward function due to the replacement of the original a. Given that DIPO is an off-policy algorithm, the reuse of these replaced transitions for training the Q-function could be problematic, as the agent is training values of actions that have not been actually played out in the environment and their true outcome (reward and next state) are unknown. Second, DIPO uses different batches for Q-learning and policy updates, which does not guarantee coherent updates. *Q-score matching* (QSM) [55] also computes $\nabla_a Q(s, a)$ and matches vector fields of $\nabla_a \log \pi(a|s)$ and $\nabla_a Q(s, a)$. Therefore, QSM is optimizing on the score level while ours focuses on the action level.

4 Learning Multimodal Behaviors from Scratch

We consider the problem of learning multimodal behaviors with online continuous RL, i.e., in the absence of initial demonstrations. In this section, we introduce *Deep Diffusion Policy Gradient* (DDiffPG), which builds off of three main ideas. First, we want to explicitly discover behavior modes and master them. Different modes should act differently at certain states and then diverge, therefore being distinguishable from trajectories/sequences of states. Second, we must prevent mode collapse once modes have been discovered, a common issue where the RL policy favors the mode with higher Q-values due to its inherent greediness. Finally, we expect the diffusion policy to capture and control the multimodal actions during evaluation. Fig. 2 provides an overview of the proposed method, and the pseudocode is available in Alg. 1.

4.1 Unsupervised Mode Discovery

Novelty-based Exploration In multimodal learning, it is necessary to explore diverse behaviors (i.e., modes). Effective exploration is important especially when considering challenging high-dimensional continuous control tasks with sparse rewards (cf. Fig. 1). We adopt a simple yet effective approach,

namely using the difference between states' novelty as intrinsic motivation [75] to prompt exploration beyond already visited state-space regions. Following [8], we define the following intrinsic reward $r^{\text{intr}}(s, a, s') = \max(\text{novelty}(s') - \alpha \cdot \text{novelty}(s), 0)$, where novelty is parameterized as a Random Network Distillation (RND) [8] and α is a scaling factor. Note that while this intrinsic reward has been effectively verified for discrete state spaces, here we extend this exploration method to continuous domains, demonstrating its effectiveness for training diffusion policies.

Hierarchical Trajectory Clustering Our method explicitly identifies modes and masters them, unlike existing latent-conditioned approaches [32]. Given a collection of goal-reached trajectories, each consisting of a sequence of state-action pairs, we categorize them into clusters and consider each a behavior mode. In practice, we use an unsupervised hierarchical clustering approach [51]: as shown in Fig. 2(2), in the beginning, each trajectory is considered as a single cluster; then clusters within a small distance are progressively merged, continuing until a singular, unified cluster is formed. Differently from clustering approaches like K-means [48] that require a predefined number of clusters (modes), we determine clusters using a distance threshold. Fortunately, given that the distance between different modes is usually large, hierarchical clustering is not as hyperparameter-sensitive as K-means. We show the clustering performance in Fig. 3 and the robust clustering threshold in Tab. 2. For distance metric, we utilize Dynamic Time Warping (DTW) [50] with task-prior information, e.g., robot positions [32]. An advantage is its applicability to variant-length trajectories, which removes the burden of padding or stitching trajectories. Note that our method is agnostic to the particular clustering approach used, and it can be adapted to different (learning) approaches [63, 16]. For example, in Fig. 9(a), we provide an alternative clustering approach by training a VQ-VAE [70] to learn trajectory representations and using the codevectors for clustering, which learns meaningful representations suitable for our approach.

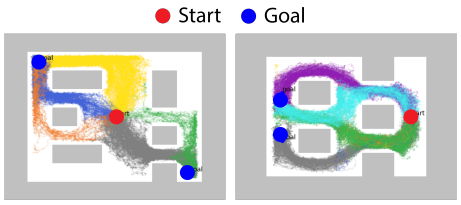


Figure 3: Hierarchical clustering on *AntMaze-v3* (Left) and *AntMaze-v4* (Right). Each color represents a mode.

4.2 Mode Learning with Mode-specific Q-functions

To master multi-modes and improve them all together, we train a different Q-function per mode and construct multimodal data batches for policy learning.

Learning mode-specific Q-functions In RL, the objective to maximize expected return can skew the policy, leading to single-mode collapse. Let us consider the *AntMaze-v1* in Fig. 1; there are two viable paths to reach the goal. If the goal position is reached via the top path, the success bonus will propagate through the TD updates, meaning that the Q-values for the top path will increase and guide the policy to the top. Even though a simple diffusion RL [74, 73] method might initially explore both sides altogether, it will eventually end with a unimodal behavior determined by which side was explored first (cf. Fig. 11b & Fig. 12b). To address this issue, we propose training a mode-specific Q-function per discovered mode, allowing for parallel policy improvements across all modes. As shown in Fig. 2(3), trajectories that, for example, are categorized into two modes M_1 and M_2, will have two Q-functions Q_1 and Q_2, respectively. One needs to notice that such mode-specific Q-functions may capture suboptimal modes, which achieve the goal but require more steps hence yielding lower discounted returns. However, these suboptimal solutions may prove valuable in practical scenarios where the optimal action is infeasible, e.g., the routine path problem discussed in Sec. 1 and Sec. 5.5.

We additionally train a Q-function dedicated to exploration, which can be considered as an exploratory mode. This Q-function is trained exclusively with the intrinsic rewards and transitions from all trajectories, e.g., $M_1 \cup M_2$ in Fig. 2(3), regardless of which behavioral mode they represent. Such decoupling of exploration-exploitation on the Q-function level ensures that we continue exploring even after certain modes are well-learned, since our intrinsic reward only considers state novelty.

Constructing a multimodal batch The diffusion model's multimodality stems from the underlying multimodal distribution of the data. An intuitive strategy to obtain multimodality is to construct a multimodal training batch and feed it to the policy — each batch contains data from different modes. While the mode clustering is on the goal-reached trajectory level, it is essential to include data from

unsuccessful trajectories as well. Practically, this is achieved by computing the distance between an unsuccessful trajectory and N goal-reached trajectories randomly sampled from each cluster. The cluster with the smallest average distance is then designated as the final cluster for the unsuccessful trajectory (lines 8-14 in Alg. 2).

Re-clustering We perform re-clustering over trajectories at every F iterations (see Tab. E). During this process, for each newly formed cluster, we assess its overlap with clusters identified in the previous clustering iteration. The new cluster then inherits Q-functions and a^{target} from the preceding cluster that has the most overlap with. This ensures a continuity of learning and adaptation across successive re-clustering stages. The pseudocode is in Alg. 2, and implementation details are in Appx.C.

4.3 Mode Control via Latent Embeddings

Controlling the learned multimodal policy to exhibit specific behaviors rather than random generation is beneficial, especially in non-stationary test environments where certain modes may become nonviable. To achieve this, we propose conditioning the diffusion policy on mode-specific embeddings during training. As shown in Fig. 2(4), our method generates a unique latent embedding for each mode, which is then incorporated into the state information. Throughout the training process, we selectively include or mask (zero-out) these embeddings with a probability of p. By providing specific embeddings, we can, therefore, explicitly control the execution of desired modes or, alternatively, mask them to enable random mode selection. This technique affords several benefits:

- In non-stationary environments, planning approaches can empower the agent to navigate around undesirable modes through controlled mode selection, improving their success rate. We demonstrate an application for online replanning in Section 5.5.
- Our method learns multiple modes, some of which may be suboptimal, e.g., longer paths in the navigation problem. Given our knowledge of each mode's trajectories, we can estimate the expected return of each mode and select the one with the highest return to optimize performance.
- Since the exploratory mode has its unique embedding, we can control the exploration-exploitation tradeoff during training by adjusting the proportion of exploratory mode used in action generation at the data collection phase. Note that we exclude the exploratory mode to eliminate noisy exploratory behaviors at test time, leading to an increased success rate.

5 Experiments

In this section, we present a comprehensive evaluation of our method against SoTA baselines. First, we verify that DDiffPG can learn multimodal behaviors and discuss the performance compared to baselines. We then highlight the advantages of learning a multimodal policy in encouraging exploration and overcoming local minima. Finally, we provide ablations on important hyperparameters and showcase a practical application of replanning with such a multimodal policy. We run each experiment with five random seeds and plot their mean and standard error.

5.1 Setup

Tasks We evaluate our method on four AntMaze tasks [20] and four robotic control tasks [22], as shown in Fig. 1. Note that all tasks (1) are high-dimensional and continuous control tasks, e.g., in all *AntMaze* versions, the objective is to control the leg joints of an *ant* to reach the goal position; (2) contain multiple possible solutions, either with multiple goals or multiple ways that solve the task, e.g., in the *Reach* task, the robot arm can bypass the obstacle from four different directions; (3) are trained with sparse rewards, alleviating the need for engineering and reward shaping. The environment description is in Appx. D.

Baselines We consider the following baselines: (1) **DIPO** [74], which we have adapted to include the additional target action in replay buffer to ensure consistency in MDP dynamics and the reward function, as detailed in Section 3; (2) **Diffusion-QL** [73] and (3) **Consistency-AC** [17], which use diffusion model and consistency model for policy parameterization; (4) **Reparameterized Policy Gradient (RPG)** [32], which uses a model-based approach with multimodal policy parameterization; (5) **TD3** [21]; (6) **SAC** [25]. For fair comparisons, we use double Q-learning [29] and distributional RL [6] for all baselines. Additionally, all baselines except RPG use the same intrinsic rewards as

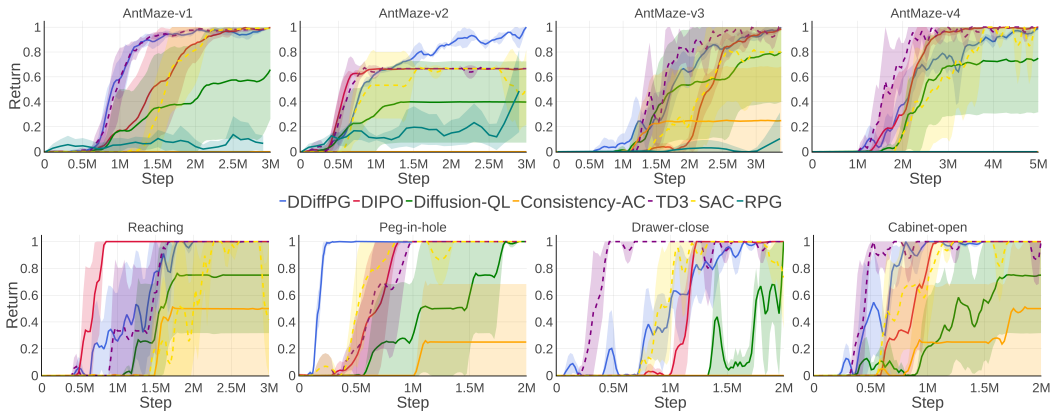


Figure 4: Performance of DDiffPG and baseline methods in the four AntMaze and robotic manipulation environments.

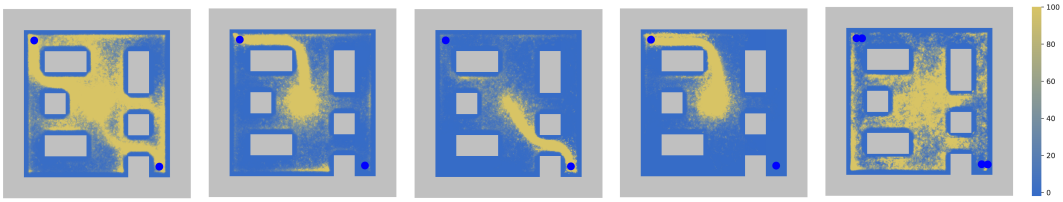
ours, while RPG has a similar built-in RND-based intrinsic reward as introduced in their paper. Hyperparameters are available in Tab. E.

5.2 DDiffPG Masters Multimodal Behaviors

We investigate whether DDiffPG can learn multimodal behaviors from scratch. We first perform observational evaluations and count the number of different modes over 20 episodes. As shown in Tab. 3 and Tab. 4, DDiffPG demonstrates consistent exploration and acquisition of multiple behaviors. For instance, in *AntMaze-v3*, multiple paths exist within the maze, but not all are the same length; DDiffPG is capable of learning and freely executing all these paths, including the suboptimal one — note that we call suboptimal a path with lesser discounted return than the shortest optimal one. However, assuming a goal-reaching success indicator all paths are successful. Nonetheless, the suboptimal issue can be mitigated given our ability to control the agent's behavior through the mode embeddings, as we discuss in Section 5.5. In *Cabinet-open*, the agent can move the arm to either layer and subsequently pull the door open. This contrasts with other methods, which fail to exhibit multimodal behavior. We observe that even policies parameterized as diffusion-based models, namely DIPO, Diffusion-QL, and Consistency-AC, can quickly collapse to a single mode and thereafter follow the greedy solution. This verifies the significance of our proposed method in capturing multimodality.

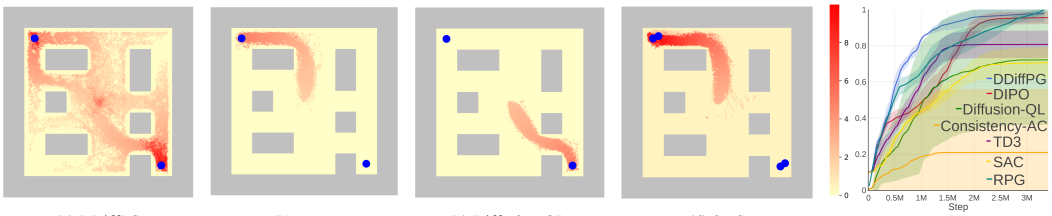
In Fig. 4, we see that DDiffPG has comparable performance to the baselines on all eight tasks while acquiring multimodal behaviors. In the AntMaze tasks, the sample efficiency of DDiffPG, DIPO, TD3 and SAC are similar: in *AntMaze-v1* and *AntMaze-v3*, TD3 and DDiffPG surpass the performance of others; in *AntMaze-v2* and *AntMaze-v4*, TD3 and DIPO are the most sample-efficient. In the manipulation tasks, a similar pattern emerges: DDiffPG leads *Peg-in-hole*, DIPO excels in *Reach*, and TD3 leads both *Drawer-close* and *Cabinet-open*. DDiffPG generally demonstrates lower sample efficiency than baselines in tasks that pose significant exploration challenges — this is expected since our method strives to discover multiple solutions. For example, in *AntMaze-v2*, the route to the top-left goal is more extended, and in *Reach*, the robotic dynamics make it difficult to explore the bottom paths. For simple exploration tasks, DDiffPG can achieve similar or even superior performance, as DDiffPG simultaneously explores the environment from multiple directions and the design of mode-specific Q-function effectively narrows the scope, facilitating faster convergence.

Diffusion-QL and Consistency-AC tend to lag behind as shown in Fig. 4. Both methods optimize the diffusion policy by backpropagating directly through the diffusion model, and we observed that their actor gradient may remain zero throughout the training in some seeds, resulting in a high variance (shadow area). In contrast, our diffusion policy gradient approach, which turns the training objective into minimizing the MSE loss w.r.t. the action target, demonstrates significantly greater stability. For RPG, its performance illustrates that policy learning remains challenging despite demonstrating good exploration. One potential reason is that VAEs condition the policy on a latent variable, which offers



(a) DDiffPG (b) DIPO (c) Diffusion-QL (d) SAC (e) RPG

Figure 5: Exploration maps of DDiffPG and baselines in *AntMaze-v3*.



(a) DDiffPG (b) DIPO (c) Diffusion-QL (d) SAC State Coverage

Figure 6: (a)-(d) Q-value maps of baselines and DDiffPG and (Right) the state coverage in *AntMaze-v3*.

a pathway to multimodality but sometimes leads to non-existing modes. This consideration led us to adopt a more straightforward yet effective clustering approach for explicit mode discovery.

5.3 Seeking of Multimodality Encourages Exploration and Overcomes Local Minima

Observation 1. *DDiffPG encourages exploration.*

We demonstrate the potential of DDiffPG in exploration using the exploration density maps and state coverage rates in the AntMaze environments. We discretize the maze and track the cell visitation. To avoid the dominance of high-density areas such as the starting positions, we set a max density threshold of 100, which means the cell has been visited at least 100 times. As shown in Fig. 5, in selected results for *AntMaze-v3*, DDiffPG explores multiple paths to the two separate goal positions, contrasting sharply with baselines that typically discover only a single path. For state coverage in Fig. 6, we measure the binary coverage of each cell and find that DDiffPG achieves a much higher coverage rate than baselines except RPG, **verifying that the continuous exploration capability of DDiffPG helps exploration**. While RPG achieves good exploration, it fails to solve the task as shown in Fig. 4. The maps for all AntMaze tasks and baselines are available in Appx. F.

Observation 2. *DDiffPG can overcome local minima.*

We showcase that DDiffPG effectively overcomes local minima when learning a multimodal policy. The key intuition is that unlike other methods that explore the first solution and collapse into it, DDiffPG continuously explores and seeks different solutions, enabling it to escape suboptimal local minima. As illustrated in Fig.1, we present two tasks, each posing distinct local minima challenges.

In *AntMaze-v1*, an ant must circumvent a central obstacle to reach its goal, with two possible routes: over the top or beneath the bottom. The optimal path depends on the ant's randomized starting position since there is only one shortest path. As shown in Fig. 11, DDiffPG discovers both routes and learns to select the shortest one based on the starting location, while baselines struggle to adjust their paths adaptively. In *AntMaze-v2*, the ant faces two goals: the top-left goal offers a higher reward, while the right-hand goal is easier to reach. Baseline models often get trapped going for the easier, lower-reward goal. However, we plot the Q-value density maps in Fig. 13, and we find that DDiffPG locates both goals and learns to go either one of them, effectively overcoming the local minima and reaching a higher cumulative return. On the other hand, RPG also explores the top-left goal and overcomes the local minima. However, it cannot consistently solve the task.

5.4 Ablation Studies

We investigate the impact of the number of diffusion steps, batch size, action gradient learning rate, and number of Updates-To-Data (UTD) ratio. These hyperparameters are of particular interest given

the diffusion policy and our learning procedure. For batch sizes in Fig. 7(a), we observe that larger batches enhance sample efficiency. Due to the mixed multimodal batch, a larger batch-size helps smooth the learning. In Fig. 7(b), we find that the number of diffusion steps appears to have minimal impact on performance, with 5 steps being sufficient to acquire and maintain multimodal behaviors, aligning with the findings of other diffusion policy learning methods [73, 17].

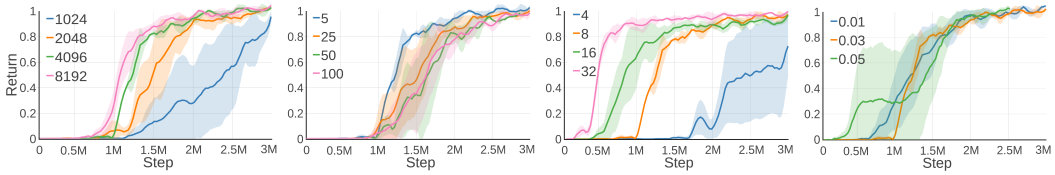


Figure 7: Abaltion studies for key parameters of DDiffPG: (a) batch size, (b) diffusion steps, (c) update-to-data (UTD) ratio, and (d) action gradient learning rate.

In Fig. 7(c), the UTD ratio indicates that more frequent updates can accelerate learning; however, a very large number would lead to increased computational demands and wall-clock time. In terms of the action-gradient learning rate (Fig.7(d)), while a higher rate initially aids learning, it may introduce increased variance due to the larger step sizes in updating target actions. An adaptive approach to the learning rate might be beneficial. Finally, our computational time analysis in Fig.8 reveals that our DDiffPG method is approximately five times slower than both TD3 and SAC, primarily due to the overhead associated with updating target actions. However, we trust that continuous developments on diffusion models will allow much faster training and inference in the future [18].

5.5 Online Replanning with a Multimodal Policy

We present a practical application of a trained multimodal policy in online replanning, particularly in nonstationary environments. We replicate the routine path problem described in Sec. 1 in our maze experiments by introducing random obstacles that obstruct certain paths. As discussed in Sec. 4.3, our policy can selectively execute different modes by conditioning the mode's embedding. To capitalize on this feature, we developed a proof-of-concept planner, which iteratively tests different modes until a successful route is found. Specifically, if the ant remains stationary at a location for 10 consecutive steps, the planner initiates an alternative mode. As shown in Tab. 3, while baseline methods tend to fail, becoming stuck in front of newly introduced obstacles until the end of an episode, our proof-of-concept planner demonstrates a significantly higher success rate. This demonstrates the adaptability and efficacy of our multimodal policy in performing in nonstationary environments.

6 Conclusions, Limitations & Future Work

In this work, we presented a novel algorithm, DDiffPG, for learning multimodal behaviors from scratch. We parameterized the policy as diffusion model and proposed diffusion policy gradient to enable training diffusion models with RL objectives in online settings. Unlike existing methods that rely on latent conditioning to retain multimodality, we emphasized explicitly discovering, preserving, and improving multimodal behaviors. First, we employed a novelty-based intrinsic motivation to explore different modes and used an unsupervised hierarchical clustering approach over trajectories to identify them. Nevertheless, the RL objective can skew the policy toward a single mode. We addressed the issue by introducing mode-specific Q-functions to optimize each mode, providing the diffusion policy with a multimodal batch to train on. We further achieved explicit mode control by conditioning the policy on a mode-specific latent embedding, which was shown to be useful for online replanning. Our evaluation demonstrated the algorithm's effectiveness in learning multimodal behaviors in complex control scenarios, such as AntMazes and robotic tasks, and showcased the potential of the multimodal policy to encourage exploration and overcome local minima.

References

[1] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[3] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[4] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *ArXiv*, abs/2206.11795, 2022.

[5] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, TB Dhruva, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. In *International Conference on Learning Representations*, 2018.

[6] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, 2017.

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[8] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. *ArXiv*, abs/1810.12894, 2018.

[9] Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020.

[10] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*, pages 1430–1440. PMLR, 2021.

[11] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

[12] Li Chen, Peng Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *ArXiv*, abs/2306.16927, 2023.

[13] Lili Chen, Shikhar Bahl, and Deepak Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. *ArXiv*, abs/2312.04549, 2023.

[14] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[15] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[16] Shripad Vilasrao Deshmukh, Arpan Dasgupta, Balaji Krishnamurthy, Nan Jiang, Chirag Agarwal, Georgios Theocharous, and Jayakumar Subramanian. Explaining rl decisions with trajectories. In *The Twelfth International Conference on Learning Representations*, 2023.

[17] Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2023.

[18] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

[19] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2018.

[20] Justin Fu, Aviral Kumar, Ofir Nachum, G. Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *ArXiv*, abs/2004.07219, 2020.

[21] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[22] Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*, 2021.

[23] Huy Ha, Pete Florence, and Shuran Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Conference on Robot Learning*, pages 3766–3777. PMLR, 2023.

[24] Huy Ha, Peter R. Florence, and Shuran Song. Scaling up and distilling down: Language-guided robot skill acquisition. *ArXiv*, abs/2307.14535, 2023.

[25] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[26] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[27] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *ArXiv*, abs/2010.02193, 2020.

[28] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[29] H. V. Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI Conference on Artificial Intelligence*, 2015.

[30] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[31] Shengchao Hu, Li Shen, Ya Zhang, Yixin Chen, and Dacheng Tao. On transforming reinforcement learning with transformers: The development trajectory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[32] Zhiao Huang, Litian Liang, Z. Ling, Xuanlin Li, Chuang Gan, and Hao Su. Reparameterized policy learning for multimodal trajectory optimization. In *International Conference on Machine Learning*, 2023.

[33] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[34] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022.

[35] Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. Robot learning of mobile manipulation with reachability behavior priors. *IEEE Robotics and Automation Letters*, 7(3):8399–8406, 2022.

[36] Pierre-Alexandre Kamienny, Jean Tarbouriech, Alessandro Lazaric, and Ludovic Denoyer. Direct then diffuse: Incremental unsupervised skill discovery for state covering and goal reaching. *ArXiv*, abs/2110.14457, 2021.

[37] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[38] Ayush Karn, Shubham Kumar, Sonu K Kushwaha, and Rahul Katarya. Image synthesis using gans and diffusion models. *2023 IEEE International Conference on Contemporary Computing and Communications (InC4)*, 1:1–6, 2023.

[39] Seongun Kim, Kyowoon Lee, and Jaesik Choi. Variational curriculum reinforcement learning for unsupervised discovery of skills. In *International Conference on Machine Learning*, 2023.

[40] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[41] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[42] Sang-Hyun Lee and Seung-Woo Seo. Unsupervised skill discovery for learning shared structures across changing environments. In *International Conference on Machine Learning*, pages 19185–19199. PMLR, 2023.

[43] Sang-Hyun Lee and Seung-Woo Seo. Unsupervised skill discovery for learning shared structures across changing environments. In *International Conference on Machine Learning*, 2023.

[44] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[45] Chengshu Li, Fei Xia, Roberto Martin-Martin, and Silvio Savarese. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning*, pages 603–616. PMLR, 2020.

[46] Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, 2023.

[47] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[48] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Mathematics*, 1967.

[49] Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *ArXiv*, abs/2110.09514, 2021.

[50] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[51] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[52] Soroush Nasiriany, Vitchyr H. Pong, Ashvin Nair, Alexander Khazatsky, Glen Berseth, and Sergey Levine. Disco rl: Distribution-conditioned reinforcement learning for general-purpose policies. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6635–6641, 2021.

[53] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[54] Seohong Park, Kimin Lee, Youngwoon Lee, and P. Abbeel. Controllability-aware unsupervised skill discovery. In *International Conference on Machine Learning*, 2023.

[55] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *ArXiv*, abs/2312.11752, 2023.

[56] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[57] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions on Machine Learning Research*, 2022.

[58] Moritz Reuss, Maximilian Xiling Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *ArXiv*, abs/2304.02532, 2023.

[59] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Neural Information Processing Systems*, 2018.

[60] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2021.

[61] Vaibhav Saxena, Yotto Koga, and Danfei Xu. Constrained-context conditional diffusion models for imitation learning. *ArXiv*, abs/2311.01419, 2023.

[62] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[63] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. In *International Conference on Machine Learning*, pages 9443–9454. PMLR, 2021.

[64] Tanmay Shankar and Abhinav Gupta. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pages 8624–8633. PMLR, 2020.

[65] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, L. Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[66] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

[67] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[68] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[69] Julen Urain, Niklas Funk, Jan Peters, and Georgia Chalvatzaki. Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5923–5930. IEEE, 2023.

[70] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[71] Siddarth Venkatraman, Shivesh Khaitan, Ravi Tej Akella, John Dolan, Jeff Schneider, and Glen Berseth. Reasoning with latent diffusion in offline reinforcement learning. *ArXiv*, abs/2309.06599, 2023.

[72] Sen Wang, Daoyuan Jia, and Xinshuo Weng. Deep reinforcement learning for autonomous driving. *ArXiv*, abs/1811.11329, 2018.

[73] Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *ArXiv*, abs/2208.06193, 2022.

[74] Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *ArXiv*, abs/2305.13122, 2023.

[75] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E. Gonzalez, and Yuandong Tian. Noveld: A simple yet effective exploration criterion. In *Neural Information Processing Systems*, 2021.

[76] Zhengbang Zhu, Hanye Zhao, Haoran He, Yichao Zhong, Shenyu Zhang, Yong Yu, and Weinan Zhang. Diffusion models for reinforcement learning: A survey. *ArXiv*, abs/2311.01223, 2023.

A Preliminaries

Diffusion Model Diffusion models [30, 67, 68] is a class of generative models that deploys a stochastic denoising process for learning to generate samples from a probability distribution $p(x)$ by mapping Gaussian noise to the target distribution through an iterative process, assuming $p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$, where $\mathbf{x}_0, \dots, \mathbf{x}_T$ are latent variables of the same dimensionality as the data $\mathbf{x}_0 \sim p(\mathbf{x}_0)$. A diffusion model approximates the posterior $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ through a *forward diffusion process*, i.e., a fixed Markov chain, which adds gradually Gaussian noise to the data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ according to a variance schedule $\beta_1, \dots, \beta_T$, defined as $q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x_t}|\mathbf{x_{t-1}})$, with $q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$. Diffusion models learn to sample from the target distribution $p(\mathbf{x}_T)$ by sampling noise from a Gaussian $p(\mathbf{x}_T) \sim \mathbf{0}, \mathbf{I}$ and iteratively denoising the noise to generate in-distribution samples, through a *reverse diffusion process* $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, defined as $p(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, with $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t))$. The reverse diffusion process is optimized by minimizing a surrogate loss-function [30] $\mathcal{L}(\theta) = \mathbb{E}_{t\sim[1,T], \mathbf{x}_0\sim q(\mathbf{x}_0), \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|$. After training, we sample the diffusion model by $\mathbf{x}_T \sim p(\mathbf{x}_T)$ and run the reversed diffusion chain to go from $t = T$ to $t = 0$.

Markov Decision Process A Markov Decision Process (MDP) is the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$ [56], where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition kernel, and $\gamma \in [0, 1)$ is the discount factor. We define a policy $\pi \in \Pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as the probability distribution of the event of executing an action a in a state s. A policy π induces a value function corresponding to the expected cumulative discounted reward collected by the agent when executing action a in state s, and following policy π thereafter: $Q^\pi(s, a) \triangleq \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{i+k+1} | s_i = s, a_i = a, \pi\right]$, where r_{i+1} is the reward obtained after the i-th transition. Solving an MDP means finding the optimal policy π^*, i.e., the one maximizing the expected discounted return. In this paper, we are particularly interested in tasks where there are multiple goals or there is only one goal but with multiple solutions to it.

B Pseudoalgorithm

Algorithm 1 Deep Diffusion Policy Gradient

Input: initial policy parameters θ, initial Q-function parameters ϕ, replay buffer $\mathcal{D}$, diffusion iteration N
for each iteration **do**
  for $t = 1, \cdots, T$ **do**
    Observe state s_t and sample action $a_t^0 \sim \pi_\theta(a_t|s_t)$ via reverse diffusion process
    Execute action $a_t = a_t^0 + \epsilon$, where $\epsilon \sim \mathcal{N}$
    Initialize $a_t^{target} = a_t$
    Store $(s_t, a_t, a_t^{target}, r_t, s_{t+1})$ in $\mathcal{D}$
  end for
  for $g = 1, \cdots, G$ **do**
    Sample M batch $B_i = \{(s, a, a^{target}, r, s')\}$ from replay buffer $\mathcal{D}$, where M is the number of modes discovered
    Get $Q_{\phi_i}, i = 1, 2, ..., M$ from Algo. 2
    for each mode **do**
      Update Q_{ϕ_i} with Bellman Equation on B_i
      for $k = 1, \cdots, K$ **do**
        Compute target action a^{target} by one step of gradient ascent following $a^{target} \leftarrow a^{target} + \eta \nabla_a Q_{\phi_i}(s, a^{target})$
      end for
      Replace a^{target} in $\mathcal{D}$
    end for
    Concatenate $\{(s, a^{target})_i\}|_{i=1}^{M}$ and update diffusion policy π_θ following (1)
  end for
end for

Algorithm 2 Mode Discovery via Clustering

Input: goal-reached trajectories $\{\tau^s\}$, unsuccessful trajectories $\{\tau^u\}$, previous cluster C_{old}
Compute distance matrix D of $\{\tau^s\}$ with DTW metric [50]
Obtain cluster C via hierarchical clustering with matrix D
for c in C **do**
    Find the cluster c_{old} with the largest overlap between c and C_{old}
    Assign Q_ϕ and a^{target} from c_{old} to c
end for
for τ^u in $\{\tau^u\}$ **do**
    for c in C **do**
        Sample N trajectories from cluster c
        Compute average distance between τ^u and $\{\tau^s_n\}|_{n=1}^N$
    end for
    Find the cluster c with the smallest average distance
    Add τ^u to c
end for

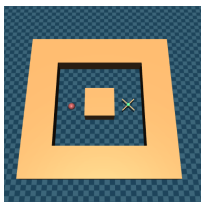
C Implementation details

Our approach includes a high-performance implementation of the proposed algorithm. First, each trajectory is assigned a unique identifier (ID), where clusters group many such IDs representing distinct modes. All trajectory elements, including states, actions, target actions, and rewards, are tagged with this ID. This allows storing all trajectories in one replay buffer, enabling efficient batch sampling. Second, computing distances between trajectories can be computationally demanding. To address this, we implement a hashmap for storing these distances, keyed by the trajectory IDs. This strategy ensures that distance computation between any two specific trajectories is performed only once.

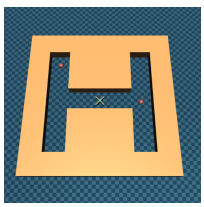
D Environment details

The *AntMaze* environments are implemented based on the D4RL benchmark [20]. The *AntMaze* is a navigation task, in which the agent controls the movement of a complex 8-DOF "Ant" quadruped robot. The objective is to reach goal positions represented by the red ball(s). The robotic manipulation environments with Franka are based on [22]. The agent controls a 7-DOF Franka arm in joint space for different manipulation tasks.

For the above environments, we designed to contain multiple possible solutions to show the multimodality, either with multiple goals or multiple ways that solve the task. Note that the goal position is static and invisible to the agent, meaning that the agent has to explore the goal first. We use a **sparse reward 0-1 reward for all environments**, which is activated upon reaching the goal. We describe each environment and its multimodal solutions as follows:



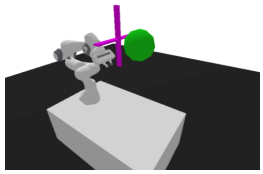
AntMaze-v1: it contains one goal in the maze. The ant is expected to bypass a central obstacle to reach the goal position, with two possible routes, either over the top or beneath the bottom of the obstacle. The optimal path varies depending on the ant's randomized starting position, as there is only one shortest path. The episode length is 500.



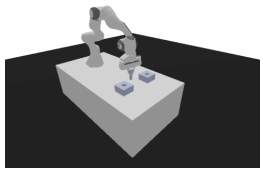
AntMaze-v2: it contains two goals in the maze, with the top-left goal offering a higher reward than the right-hand goal. Due to the higher reward, the optimal path is to reach the top-left goal, however, the ant may get trapped in the right goal as it is much easier to explore. The episode length is 500.

AntMaze-v3: it contains two goals in the maze, each accessible via multiple routes. For the **goal in the top-left**, three routes are viable: (1) go upwards to the end and then turn left; (2) move diagonally towards the top-left until encountering the left border, then head upwards, and (3) move towards the bottom-left, circumvent the left obstacle, and then go upwards. The distances of the first two paths are comparable, whereas the third is much longer. For the **goal in the right-bottom**, there are two comparable routes: (1) moving right and then downwards or (2) going downwards and then right. Possible solutions are shown in the visualization of clusetering performance in Fig. 3. The episode length is 700.

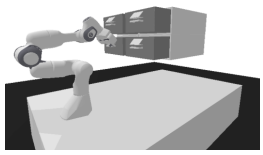
AntMaze-v4: it contains two goals in the maze, each accessible through two routes. For the top goal, the agent can bypass the obstacle by going up, then has the option to either go up or down to reach it. The routes to the bottom goal is symmetrical. Possible solutions are shown in Fig. 3. The episode length is 700.



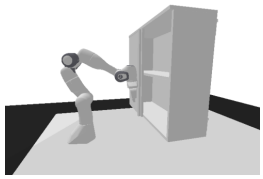
Reach: the agent controls the Franka arm to reach the red ball, navigating around a fixed cross-shaped obstacle that lies in the path. Despite the presence of a single goal position, the agent can bypass the obstacle in four distinct ways, offering multiple solutions to the task. The episode length is 100.



Peg-in-hole: the agent controls the Franka arm to perform a peg-insertion task. With two holes available on the desk, the agent can successfully complete the task by inserting the peg into either hole. The episode length is 100.



Drawer-close: the agent controls the Franka arm to close drawers. There are four drawers on the desk, and the agent can close either drawer to finish the task. The episode length is 100.



Cabinet-open: the agent controls the Franka arm to open a cabinet. The cabinet has two layers therefore the agent can move the arm to either layer and subsequently pull the door open to finish the task. The episode length is 100.

E Hyperparameters

Here, we list the hyperparameters used for all baselines and tasks.

Table 1: Hyperparameter setup for all tasks. For RPG, we use the default hyperparameters for sparse reward.

Hyperparameter	DDiffPG/DIPO/Diffusion-QL/Consistency-AC/TD3/SAC
Num. Environments	256
Critic Learning Rate	5×10^{-4}
Actor Learning Rate	3×10^{-4}
Action Learning Rate	3×10^{-2} (DDiffPG/DIPO)
Alpha (α) Learning Rate	5×10^{-3} (SAC)
V_min (distributional RL)	0
V_max (distributional RL)	5
Num. Atoms (distributional RL)	51
Optimizer	Adam
Target Update Rate (τ)	5×10^{-2}
Batch Size	4,096
UTD ratio	8
Discount Factor (γ)	0.99
Gradient Clipping	1.0
Replay Buffer Size	$2,000$ trajectories $\approx 1 \times 10^6$ (DDiffPG) 1×10^6 (baselines)
Reclustering Frequency	100 (DDiffPG)
Mode Embedding Dim.	5 (DDiffPG)

Table 2: Clustering threshold. The default threshold is set to $0.7 \max(Z[:, : 2])$ corresponding with MATLAB(TM) behavior, where Z is the linkage matrix.

	Clustering threshold
AntMaze-v1	50
AntMaze-v2	70
AntMaze-v3	70
AntMaze-v4	50
Reach	default
Peg-in-hole	default
Drawer-close	default
Cabinet-open	default

F Additional experimental results.

Table 3: Number of modes discovered, success rate (S.R.), and episode length (E.L.) for AntMazes and the maze with randomly initialized obstacles, averaged over 20 random seeds per case.

		DDiffPG	RPG	TD3	SAC	DIPO	Diff-QL	Con-AC
	#modes	2	1	1	1	1	1	0
AntMaze-v1	S.R.	1.0	0.2	1.0	0.98	0.98	0.63	0.0
	E.L.	75.7	450.1	59.2	89.1	75.3	241.7	500
	#modes	2	1.5	1	1	1	1	0
AntMaze-v2	S.R.	1.0	0.5	0.75	0.53	0.75	0.59	0.0
	E.L.	66.8	259.4	35.6	222.3	34.6	225.7	500
	#modes	4.3	1	1	1	1	1	1
AntMaze-v3	S.R.	0.98	0.1	1.0	0.8	1.0	0.77	0.25
	E.L.	142.5	642.1	83.4	207.8	99.3	226.6	545.1
	#modes	3.8	0	1	1	1	1	0
AntMaze-v4	S.R.	1.0	0.0	1.0	1.0	1.0	0.75	0.0
	E.L.	151.1	700	88.1	92.8	86.9	249.5	700
	#modes	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
Randomized	S.R.	1.0	0.0	0.5	0.45	0.5	0.45	0.1
	E.L.	162.3	700	310.2	342.1	293.5	421.4	582.3

Table 4: Number of modes, success rate (S.R.), and episode length (E.L.) for robotic tasks, averaged over 20 random seeds.

		DDiffPG	TD3	SAC	DIPO	Diff-QL	Con-AC
	#modes	2.8	1	1	1	1	1
Reach	S.R.	1.0	1.0	0.95	1.0	0.75	0.5
	E.L.	23.8	18.0	20.5	18.6	40.5	60.5
	#modes	2	1	1	1	1	1
Peg-in-hole	S.R.	1.0	1.0	1.0	1.0	1.0	0.2
	E.L.	5.9	4.7	4.7	5.1	4.92	80.91
	#modes	3.5	1	1	1	1	1
Drawer-close	S.R.	1.0	1.0	1.0	1.0	0.8	0.2
	E.L.	23.6	22.0	24.7	22.8	34.5	80.74
	#modes	2	1	1	1	1	1
Cabinet-open	S.R.	1.0	0.98	1.0	1.0	0.75	0.5
	E.L.	21.1	14.3	24.3	19.6	42.1	59.5

We provide an evaluation of computational time compared with baselines. We use NVIDIA GeForce RTX 4090 for all experiments. However, given the intense research landscape in diffusion models, we hope to make the training and inference more time-efficient in our future work.

- For data collection, DDiffPG needs more wall-clock time than others, which is due to the trajectory processing, clustering, etc. We note that DIPO has a similar wall-clock time with the time of TD3 and SAC, implying that the impact of inference speed of diffusion model is not significant.
- For policy updates, DDiffPG and DDiffPG (v) require less computational time. This is because TD3 and SAC need to estimate the Q-value during policy updates, while DDiffPG and DIPO only need to minimize the MSE loss.
- For critic update, DDiffPG requires more wall-clock time due to multiple Q-functions.
- For target action update, only DDiffPG and DIPO needs to compute the target action. DDiffPG requires more wall-clock time because it has multimodal batches and needs to compute the target action for each sub-batch.

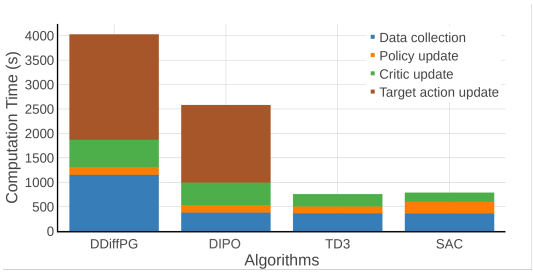
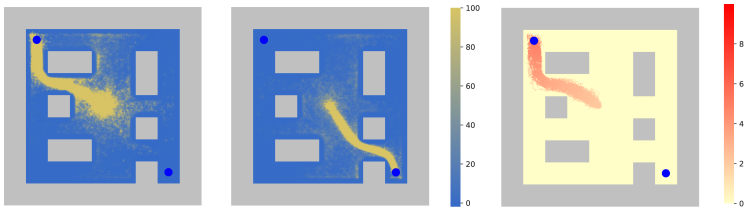


Figure 8: Comparison on wall-clock time.

We implement an alternative clustering approach on the high-dimensional state space by training a vector-quantized variational autoencoder (VQ-VAE) to learn trajectory representations. We then performed clustering using the codevectors. The visualized cluster performance and projected embedding space verify that VQ-VAE learns meaningful representations and can be used in our approach, as an alternative clustering approach. We would like to point that any unsupervised clustering method can be coupled with our approach.

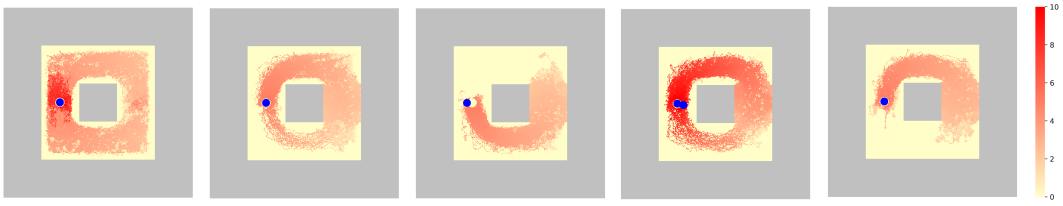
((a)) Clustering performance of VQ-VAE. ((b)) Projected embedding space.

Figure 9: Comparison of VQ-VAE clustering and projected embedding space.



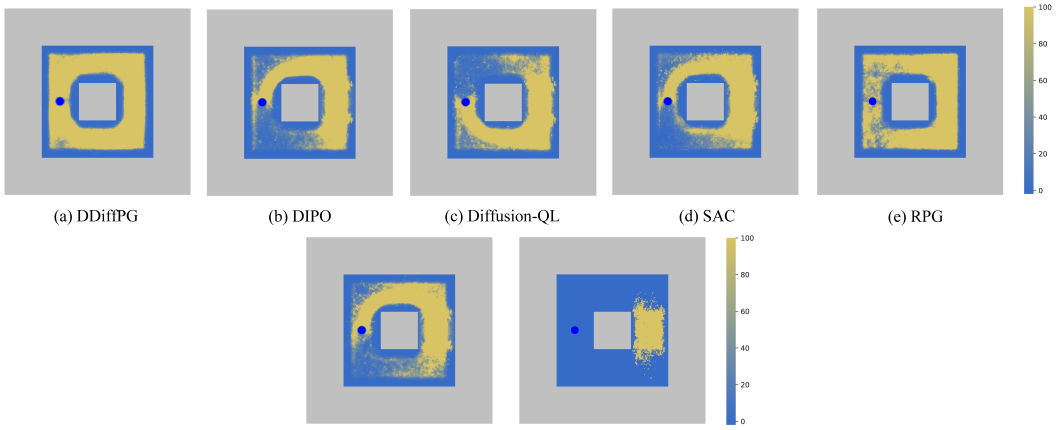
(a) TD3-explore (b) Consistency-AC-explore (c) TD3-Q

Figure 10: The rest of exploration maps and density maps in *Antmaze-v3*.



(a) DDiffPG (b) DIPO (c) Diffusion-QL (d) SAC (e) TD3

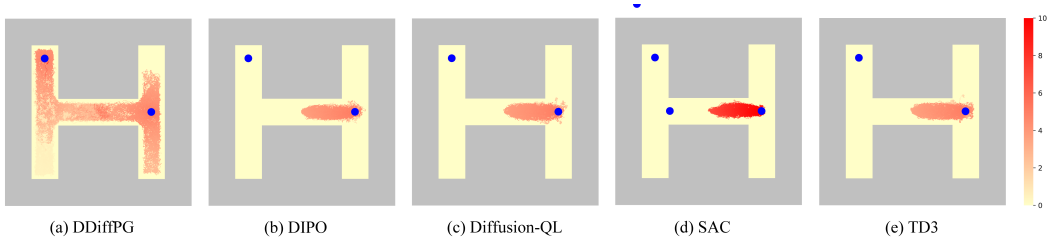
Figure 11: (a)-(d) Q-value maps of baselines and DDiffPG in *Antmaze-v1*.



(a) DDiffPG (b) DIPO (c) Diffusion-QL (d) SAC (e) RPG

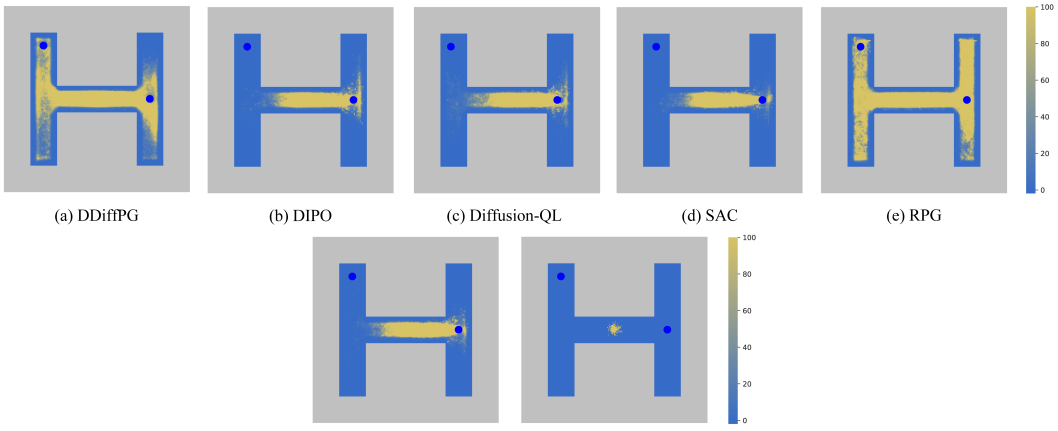
(f) TD3 (g) Consistency-AC

Figure 12: Exploration maps of DDiffPG and baselines in *AntMaze-v1*.



(a) DDiffPG (b) DIPO (c) Diffusion-QL (d) SAC (e) TD3

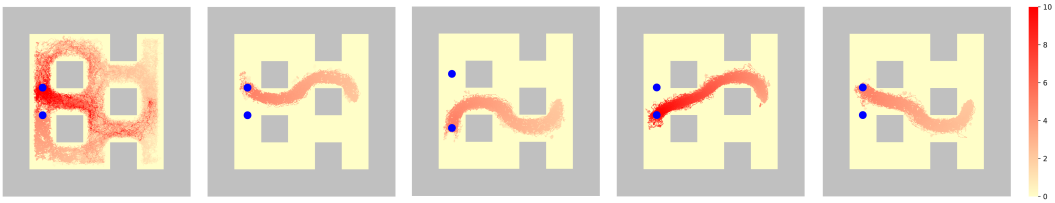
Figure 13: (a)-(d) Q-value maps of baselines and DDiffPG in *Antmaze-v2*.



(a) DDiffPG (b) DIPO (c) Diffusion-QL (d) SAC (e) RPG

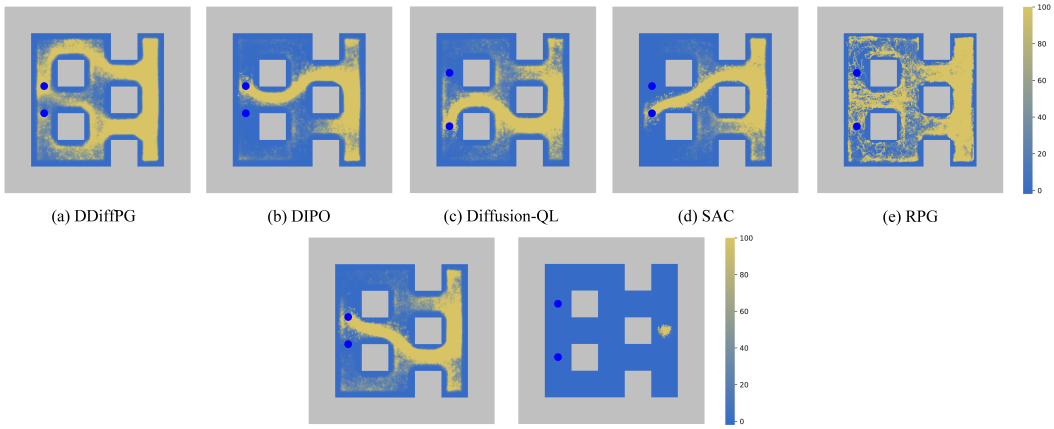
(f) TD3 (g) Consistency-AC

Figure 14: Exploration maps of DDiffPG and baselines in *AntMaze-v2*.



(a) DDiffPG (b) DIPO (c) Diffusion-QL (d) SAC (e) TD3

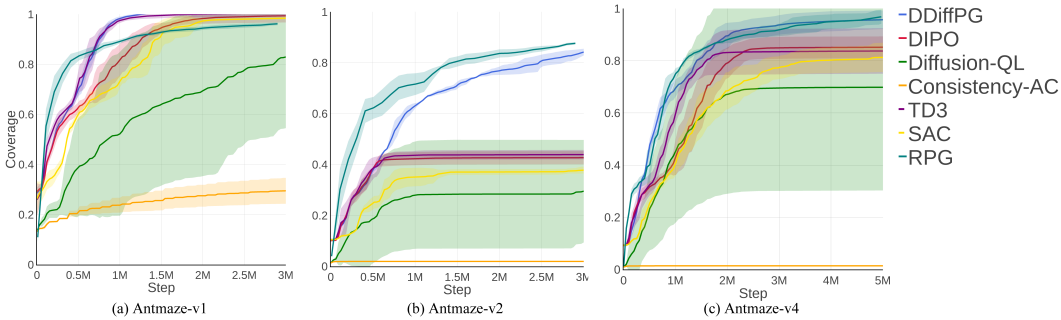
Figure 15: (a)-(d) Q-value maps of baselines and DDiffPG in *Antmaze-v4*.



(a) DDiffPG (b) DIPO (c) Diffusion-QL (d) SAC (e) RPG

(f) TD3 (g) Consistency-AC

Figure 16: Exploration maps of DDiffPG and baselines in *AntMaze-v4*.



(a) Antmaze-v1 (b) Antmaze-v2 (c) Antmaze-v4

Figure 17: State coverage in *Antmaze-v1*, *Antmaze-v2*, and *Antmaze-v4*