

A Implementation Details

A.1 Diffusion policy

Diffusion probabilistic models [84, 65] are a type of generative model that learns the data distribution $q(x)$ from a dataset $\mathcal{D} := \{x_i\}_{0 \leq i < M}$. It represents the process of generating data as an iterative denoising procedure, denoted by $p_\theta(x_{i-1}|x_i)$ where i is an indicator of the diffusion timestep. The denoising process is the reverse of a forward diffusion process that corrupts input data by gradually adding noise and is typically denoted by $q(x_i|x_{i-1})$. The reverse process can be parameterized as Gaussian under the condition that the forward process obeys the normal distribution and the variance is small enough: $p_\theta(x_{i-1}|x_i) = \mathcal{N}(x_{i-1}|\mu_\theta(x_i, i), \Sigma_i)$, where μ_θ and Σ are the mean and covariance of the Gaussian distribution, respectively. The parameters θ of the diffusion model are optimized by minimizing the evidence lower bound of negative log-likelihood of $p_\theta(x_0)$, similar to the techniques used in variational Bayesian methods: $\theta^* = \arg \min_\theta -\mathbb{E}_{x_0}[\log p_\theta(x_0)]$. For model training, a simplified surrogate loss [65] is proposed based on the mean μ_θ of $p_\theta(x_{i-1}|x_i)$, where the mean is predicted by minimizing the Euclidean distance between the target noise and the generated noise: $\mathcal{L}_{\text{denoise}}(\theta) = \mathbb{E}_{i, x_0 \sim q, \epsilon \sim \mathcal{N}}[|\epsilon - \epsilon_\theta(x_i, i)|^2]$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Specifically, our diffusion policy is represented as Equation (4) via the reverse process of a conditional diffusion model, but the reverse sampling, which requires iteratively computing ϵ_ϕ networks N times, can become a bottleneck for the running time. To limit N to a relatively small value, with $\beta_{\min} = 0.1$ and $\beta_{\max} = 10.0$, we follow [85] to define:

$$\beta_i = 1 - \alpha_i = 1 - e^{-\beta_{\min}(\frac{1}{N}) - 0.5(\beta_{\max} - \beta_{\min})\frac{2i-1}{N^2}}, \quad (12)$$

which is a noise schedule obtained under the variance preserving SDE of [86].

A.2 Goal distributions

We train our state-goal value function and high-level policy respectively with Equation (3) and (7), using different goal-sampling distributions. For the state-goal value function (Equation (3)), we sample the goals from either random states, futures states, or the current state with probabilities of 0.3, 0.5, and 0.2, respectively, following [28]. We use $\text{Geom}(1 - \gamma)$ for the future state distribution and the uniform distribution over the offline dataset for sampling random states. For the high-level policy, we mostly follow the sampling strategy of [87]. We first sample a trajectory $(s_0, s_1, \dots, s_t, \dots, s_T)$ from the dataset D_O and a state s_t from the trajectory. we either (i) sample g uniformly from the future states s_{t_g} ($t_g > t$) in the trajectory and set the target subgoal g_{sub} to $s_{\min(t+k, t_g)}$ or (ii) sample g uniformly from the dataset and set the target subgoal to $s_{\min(t+k, T)}$.

A.3 Advantage estimates

Following [14], the advantage estimates for Equation (6) is given as:

$$\tilde{A}(s_t, s_{t+\tilde{k}}, g) = \gamma^{\tilde{k}} V_\theta(s_{t+\tilde{k}}, g) + \sum_{t'=t}^{\tilde{k}-1} r(s_{t'}, g) - V_\theta(s_t, g), \quad (13)$$

where we use the notations \tilde{k} and $\tilde{s}_{t+\tilde{k}}$ to incorporate the edge cases discussed in the previous paragraph (i.e., $\tilde{k} = \min(k, t_g - t)$ when we sample g from future states, $\tilde{k} = \min(k, T - t)$ when we sample g from random states, and $\tilde{s}_{t+\tilde{k}} = s_{\min(t+k, T)}$). Here, $s_{t'} \neq g$ and $s_t \neq \tilde{s}_{t+\tilde{k}}$ always hold except for those edge cases. Thus, the reward terms in Equation (13) are mostly constants (under our reward function $r(s, g) = 0$ (if $s = g$), -1 (otherwise)), as are the third terms (with respect to the policy inputs). As such, we practically ignore these terms for simplicity, and this simplification further enables us to subsume the discount factors in the first terms into the temperature hyperparameter β . We hence use the following simplified advantage estimates, which we empirically found to lead to almost identical performances in our experiments:

$$\tilde{A}(s, g_{\text{sub}}, g) = V_\theta(g_{\text{sub}}, g) - V_\theta(s, g), \quad (14)$$

where we use g_{sub} to represent $s_{t+\tilde{k}}$ under various conditions.

Table 1: Hyperparameters.

Hyperparameter	Value
Batch Size	1024
High-level Policy MLP Dimensions	(256, 256)
State-Goal Value MLP Dimensions	(512,512,512)
Representation MLP Dimensions	(512,512,512)
Nonlinearity	GELU [88]
Optimizer	Adam [89]
Learning Rate	0.0003
Target Network Smoothing Coefficient	0.005
AWR Temperature Parameter	1.0
IQL Expectile τ	0.7
Discount Factor γ	0.99
Diversity of Subgoals α	0.5

B Hyperparameters

We present the hyperparameters used in our experiments in Table 1, where we mostly follow the network architectures and hyperparameters used by [28, 14]. We use layer normalization [90] for all MLP layers and we use normalized 10-dimensional output features for the goal encoder of state-goal value function to make them easily predictable by the high-level policy, as discussed in Appendix A.

For the subgoal steps k , we use $k = 50$ (AntMaze-Ultra), $k = 15$ (FetchReach, FetchPickAndPlace, and SawyerReach), or $k = 25$ (others). We sample goals for high-level or flat policies from either the future states in the same trajectory (with probability 0.7) or the random states in the dataset (with probability 0.3). During training, we periodically evaluate the performance of the learned policy at every 20 episode using 50 rollouts.

C Ablation Study Results

Subgoal Steps. In order to examine the impact of subgoal step values (k) on performance, we conduct an evaluation of our method on AntMaze tasks. We employ six distinct values for $k \in \{1, 5, 15, 25, 50, 100\}$. The results, depicted in Figure 9, shed light on the relationship between k and performance outcomes. Remarkably, our method consistently demonstrates superior performance when k falls within the range of 25 to 50, which can be identified as the optimal range. Our method exhibits commendable performance even when k deviates from this range, except in cases where k is excessively small. These findings underscore the resilience and efficacy of our method across various subgoal step values.

Ablation on Subgoals and Exploration Guidance. To demonstrate how subgoals and exploration guidance contribute to efficient policy learning for goal-reaching tasks, we conduct ablation experiments where we remove each component separately. The results, as shown in the Figure 10, highlight the crucial importance of subgoal setting, as the absence of subgoals hinders the resolu-

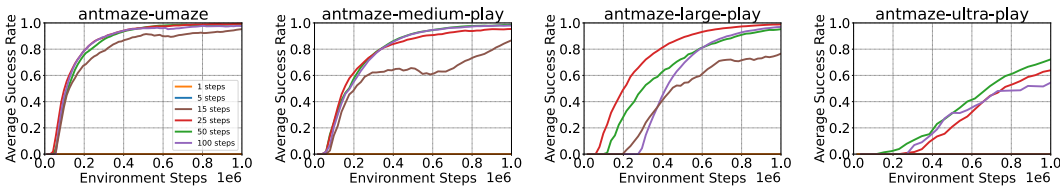


Figure 9: Ablation study of the subgoal steps k . Our method generally achieves the best performances when k is between 25 and 50. Even when k is not within this range, ours mostly maintains reasonably good performance unless k is too small (i.e., ≤ 5).

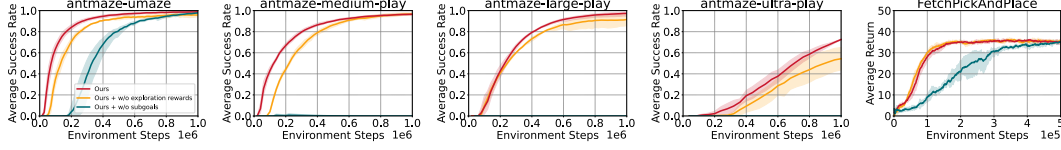


Figure 10: Ablation study on Subgoals and Exploration Guidance. The result shows that the crucial importance of subgoal setting. Additionally, incorporating exploration guidance facilitates the policy in efficiently reaching subgoals, resulting in further improvements in learning efficiency. Shaded regions denote the 95% confidence intervals across 5 random seeds.

tion of long-horizon tasks. Additionally, incorporating exploration guidance facilitates the policy in efficiently reaching subgoals, resulting in further improvements in learning efficiency. Overall, our findings indicate that including both subgoal setting and exploration guidance enables our approach to leverage the benefits of both, leading to efficient learning efficiency.

D Environments

SawyerReach environment, derived from multi-world, involves the Sawyer robot reaching a target position with its end-effector. The observation and goal spaces are both 3-dimensional Cartesian coordinates, representing the positions. The state-to-goal mapping is a simple identity function, $\phi(s) = s$, and the action space is 3-dimensional, determining the next end-effector position.

FetchReach and **FetchPickAndPlace** environments in OpenAI Gym feature a 7-DoF robotic arm with a two-finger gripper. In FetchReach, the goal is to touch a specified location, while FetchPickAndPlace involves picking up a box and transporting it to a designated spot. The state space comprises 10 dimensions, representing the gripper’s position and velocities, while the action space is 4-dimensional, indicating gripper movements and open/close status. Goals are expressed as 3D vectors for target locations.

Maze2D is a goal-conditioned planning task, which involves guiding a 2-DoF ball that can be force-actuated in the cartesian directions of x and y. Given the starting location and the target location, the policy is expected to find a feasible trajectory that reaches the target from the starting location avoiding all the obstacles.

AntMaze is a class of challenging long-horizon navigation tasks where the objective is to guide an 8-DoF Ant robot from its initial position to a specified goal location. We evaluate the performance in four different difficulty settings, including the “umaze”, “medium” and “large” maze datasets from the original D4RL benchmark. While the large mazes already pose a significant challenge for long-horizon planning, we also introduce an even larger maze “ultra” proposed by [91]. The maze in the AntMaze-Ultra task is twice the size of the largest maze in the original D4RL dataset. Each dataset consists of 999 length-1000 trajectories, in which the Ant agent navigates from an arbitrary start location to another goal location, which does not necessarily correspond to the target evaluation goal. At test time, to specify a goal g for the policy, we set the first two state dimensions (which correspond to the x-y coordinates) to the target goal given by the environment and the remaining proprioceptive state dimensions to those of the first observation in the dataset. At evaluation, the agent gets a reward of 1 when it reaches the goal.

CALVIN is another long-horizon manipulation environment features four target subtasks. We use the offline dataset provided by [92], which is based on the teleoperated demonstrations from [79]. The dataset consists of 1204 length-499 trajectories. In each trajectory, the agent achieves some of the 34 subtasks in an arbitrary order, which makes the dataset highly task-agnostic [92]. At test time, to specify a goal g for the policy, we set the proprioceptive state dimensions to those of the first observation in the dataset and the other dimensions to the target configuration. At evaluation, the agent gets a reward of 1 whenever it achieves a subtask.

586 E More Related Work

587 **Learning Efficiency.** Introducing relabeling can enhance learning efficiency. HER [81] relabels
 588 the desired goals in the buffer with achieved goals in the same trajectories. CHER [93] goes a step
 589 further by integrating curriculum learning with the curriculum relabeling method, which adaptively
 590 selects the relabeled goals from failed experiences. Drawing from the concept that any trajectory
 591 represents a successful attempt towards achieving its final state, GCSL [82], inspired by supervised
 592 imitation learning, iteratively relabels and imitates its own collected experiences. [94] filters the
 593 actions from demonstrations by Q values and adds a supervised auxiliary loss to the RL objective
 594 to improve learning efficiency. RIS [83] uses imagined subgoals to guide the policy search process.
 595 However, such methods are only useful if the data distribution is diverse enough to cover the space
 596 of desired behaviors and goals and may still face challenges in hard exploration environments.

597 F Baseline Introduction

598 F.1 Online learning baselines

599 **Online:** A standard off-policy actor-critic algorithm [69] which trains an actor network and a critic
 600 network simultaneously from scratch that does *not* make use of the prior data at all.

601 **RND:** Extends the *Online* method by incorporating Random Network Distillation [4] as a novelty
 602 bonus for exploration. given an online transition (s, a, r, s') , and RND feature networks $f_\phi(s, a)$,
 603 $\bar{f}(s, a)$, we set

$$\hat{r}(s, a) \leftarrow r + \frac{1}{L} \|f_\phi(s, a) - \bar{f}(s, a)\|_2^2 \quad (15)$$

604 and use the transition (s, a, \hat{r}, s') in the online update. The RND training is done the same way as in
 605 our method where a gradient step is taken on every new transition collected.

606 **HER:** Combines *Online* method with Hindsight Experience Replay [81] to improve data efficiency
 607 by re-labeling past data with different goals.

608 **GCSL:** Trains the policy using supervised learning, leading to stable learning progress.

609 **RIS:** This method [83] incorporates a separate high-level policy that predicts intermediate states
 610 halfway to the goal. By aligning the subgoal reaching policy with the final policy, RIS effectively
 611 regularizes the learning process and improves performance in complex tasks.

612 **ExPLORe:** This approach learns a reward model from online experience, labels the unlabeled prior
 613 data [6] with optimistic rewards, and then uses it concurrently alongside the online data for down-
 614 stream policy and critic optimization.

615 F.2 offline-online baselines

616 **AWAC:** AWAC combines sample-efficient dynamic programming with maximum likelihood policy
 617 updates, providing a simple and effective framework that is able to leverage large amounts of offline
 618 data and then quickly perform online fine-tuning of reinforcement learning policies.

619 **IDL:** Avoiding querying out-of-sample actions by converting the max operator in the Bellman opti-
 620 mal equation into expectile regression, and thus learn a better Q Estimation.

621 **CQL:** CQL imposes an additional regularizer that penalizes the learned Q-function on out-of-
 622 distribution (OOD) actions while compensating for this pessimism on actions seen within the train-
 623 ing dataset. Assuming that the value function is represented by a function, Q_θ , the training objective
 624 of CQL is given by

$$\min_{\theta} \alpha \underbrace{(\mathbb{E}_{s \sim \mathcal{D}, a \sim \pi} [Q_\theta(s, a)] - \mathbb{E}_{s, a \sim \mathcal{D}} [Q_\theta(s, a)])}_{\text{Conservative regularizer } \mathcal{R}(\theta)} + \frac{1}{2} \mathbb{E}_{s, a, s' \sim \mathcal{D}} \left[(Q_\theta(s, a) - \mathcal{B}^\pi \bar{Q}(s, a))^2 \right], \quad (16)$$

625 where $\mathcal{B}^\pi \bar{Q}(s, a)$ is the backup operator applied to a delayed target Q-network, \bar{Q} : $\mathcal{B}^\pi \bar{Q}(s, a) :=$
626 $r(s, a) + \gamma E_{a' \sim \pi(a'|s')} [\bar{Q}(s', a')]$. The second term is the standard TD error. The first term $R(\theta)$ is
627 a conservative regularizer that aims to prevent overestimation in the Q-values for OOD actions by
628 minimizing the Q-values under the policy $\pi(a|s)$, and counterbalances by maximizing the Q-values
629 of the actions in the dataset following the behavior policy π_β .

630 **Cal-QL:** This method learns a conservative value function initialization can speed up online fine-
631 tuning and harness the benefits of offline data by underestimating learned policy values while en-
632 suring calibration. Specifically, Calibrating CQL constrain the learned Q-function Q_θ^π to be larger
633 than value function V via a simple change to the CQL training objective. Cal-QL modifies the CQL
634 regularizer, $R(\theta)$ in this manner:

$$\mathbb{E}_{s \sim \mathcal{D}, a \sim \pi} [\max(Q_\theta(s, a), V(s))] - \mathbb{E}_{s, a \sim \mathcal{D}} [Q_\theta(s, a)], \quad (17)$$

635 where the changes from standard CQL are depicted in **red**.

636 **SPOT:** This work constrains the policy network in offline reinforcement learning (RL) to not only
637 be within the support set but also avoid the out-of-distribution actions effectively unlike the standard
638 behavior policy through behavior regularization.

639 **PEX:** This work introduces a policy expansion scheme. After learning the offline policy, it is in-
640 cluded as a candidate policy in the policy set, which further assists in learning the online policy. This
641 method avoids fine-tuning the offline policy, which could disrupt the learned policies, and instead
642 allows the offline policy to participate in online exploration adaptively.