Are Expressive Models Truly Necessary for Offline RL?

Guan Wang[1*], **Haoyi Niu**[1*], **Jianxiong Li**[1], **Li Jiang**[2], **Jianming Hu**[1†], **Xianyuan Zhan**[1,3†]
[1] Tsinghua University, [2] McGill Univsersity, [3] Shanghai AI Lab
imonenext@gmail.com, nhy22@mails.tsinghua.edu.cn
{hujm,zhanxianyuan}@mail.tsinghua.edu.cn

Abstract

Among various branches of offline reinforcement learning (RL) methods, goal-conditioned supervised learning (GCSL) has gained increasing popularity as it formulates the offline RL problem as a sequential modeling task, therefore bypassing the notoriously difficult credit assignment challenge of value learning in conventional RL paradigm. Sequential modeling, however, requires capturing accurate dynamics across long horizons in trajectory data to ensure reasonable policy performance. To meet this requirement, leveraging large, expressive models has become a popular choice in recent literature, which, however, comes at the cost of significantly increased computation and inference latency. Contradictory yet promising, we reveal that lightweight models as simple as shallow 2-layer MLPs, can also enjoy accurate dynamics consistency and significantly reduced sequential modeling errors against large expressive models by adopting a simple recursive planning scheme: recursively planning coarse-grained future sub-goals based on current and target information, and then executes the action with a goal-conditioned policy learned from data relabeled with these sub-goals. We term our method as **R**ecursive **S**kip-Step **P**lanning (**RSP**). Simple yet effective, RSP enjoys great efficiency improvements thanks to its lightweight structure, and substantially outperforms existing methods, reaching new SOTA performances on the D4RL benchmark, especially in multi-stage long-horizon tasks.

1 Introduction

Offline reinforcement learning (RL) has emerged as a promising approach to solving many real-world tasks using logged experiences without extra costly or unsafe interactions with environments [Fujimoto *et al.*, 2019; Levine *et al.*, 2020; Zhan *et al.*, 2022]. Unfortunately, the absence of online interaction also poses the counterfactual reasoning challenge in out-of-distribution (OOD) regions, causing catastrophic failures in the conventional RL paradigm due to extrapolation error accumulation during value function learning using temporal difference (TD) updates [Fujimoto *et al.*, 2019; Kumar *et al.*, 2019a; Li *et al.*, 2023].

To remedy this, Goal-Conditioned Supervised Learning (GCSL) has gained much attention as an alternative paradigm since it reformulates offline RL problems as sequential modeling tasks, where policy extraction can be learned in a supervised manner, thus bypassing problematic value update in conventional TD learning. However, this comes with the price of much higher requirements of accurately describing the data dynamics in sequential modeling, which naturally favors more expressive models. Recent advances in highly expressive models have yielded remarkable achievements across a diverse range of domains such as computer vision [Liu *et al.*, 2021] and natural language processing [Vaswani *et al.*, 2017], which have also been extended to offline RL to accurately capture policy distributions [Wang *et al.*, 2022b; Hu *et al.*, 2023a; Hansen-Estruch *et al.*, 2023; Ada *et al.*, 2023; Wang *et al.*, 2023; Chen *et al.*, 2023], or reduce the accumulative compounding errors in

[*]Equal contribution. Code is available at https://github.com/imoneoi/RSP_JAX.
[†]Corresponding authors.

Preprint. Under review.

sequential data modeling [Chen *et al.*, 2021; Janner *et al.*, 2021, 2022; Ajay *et al.*, 2022; Villaflor *et al.*, 2022; Jiang *et al.*, 2022; Mazoure *et al.*, 2023; Niu *et al.*, 2024]. Although they sometimes provide encouraging improvements over previous approaches, they also suffer from noticeable performance deterioration when they fail to accurately represent the behavior policy and/or environment dynamics in long-horizon tasks.

Moreover, the integration of expressive models in offline RL inevitably increases both the computational load and inference latency. This poses a significant challenge for many real-world applications that are latency-sensitive or resource-constrained. Besides, methods that employ such expressive models are notoriously data-hungry to train, making them impractical for scenarios with expensive samples [Zhan *et al.*, 2022]. In Fig. 1, we compare key metrics of recent offline RL methods, including policy performance, training time, and inference latency. The use of large expressive models in prior offline RL methods such as DT [Chen *et al.*, 2021], TAP [Zhang *et al.*, 2022], TT [Janner *et al.*, 2021], and Diffuser [Janner *et al.*, 2022] sometimes results in marginal performance gains. However, this incremental improvement is offset by the exponentially increased computational load and inference latency, particularly when compared to models that do not employ such expressive models. This prompts us to ask the question:

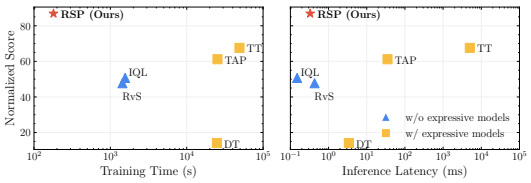

Figure 1: Comparison of performance, latency, and training time for RSP and established baselines on the D4RL AntMaze dataset. RSP significantly outperforms all baselines while completing training in just 180 seconds and maintaining an inference latency under 1ms.

Are expressive models truly necessary for offline RL?

While prior attempts introduce highly expressive models to combat the accumulated compounding error in long-sequence modeling, these methods still operate in a *fine-grained* (step-by-step) manner [Chen *et al.*, 2021; Wang *et al.*, 2022b; Janner *et al.*, 2022], which is inherently tied to the temporal structure of the environment and is susceptible to rapidly accumulated approximation errors over longer horizons [Levine *et al.*, 2020]. In this study, we provide a novel perspective to avoid step-wise accumulated compounding error in sequential modeling while only relying on a lightweight model architecture (as simple as 2-layer MLPs). The core of our method is the Recursive Skip-step Planning (RSP) scheme, which employs an iterative two-phase approach to solve tasks: it recursively plans skip-step future sub-goals conditioned on the current and target information, and then executes a goal-conditioned policy based on these coarse-grained predictions. During the recursive planning phase, RSP bypasses step-wise sub-goal prediction and focuses more on longer-horizon outcomes through recursively skipping steps. In essence, RSP can generate long-horizon sub-goals with just a few planning steps, which uses exponentially fewer steps than what is required by previous fine-grained sequence modeling methods, therefore smartly bypassing the long-horizon compounding error issue while enjoying great computational efficiency.

RSP can be practically implemented using as few as two-layer shallow MLPs for recursive skip-step dynamics models and the goal-conditioned policy. The entire learning process can be conducted in a supervised learning fashion, eliminating the need for complex stabilization tricks or delicate hyperparameter tuning in value learning. Notably, RSP not only exhibits great training efficiency but also enjoys very low inference complexity, while it achieves comparable or even superior performance against existing offline RL algorithms on D4RL benchmark [Fu *et al.*, 2020], including those equipped with expressive models. This advantage is particularly evident in complex tasks such as `AntMaze` and `Kitchen`, demonstrating the effectiveness of the long-horizon modeling capability by adopting our recursive skip-step planning scheme.

2 Related Work

Model-free Offline RL Most prior methods incorporate pessimism during training to alleviate the distributional shift problem [Levine *et al.*, 2020]. One solution is leveraging various behavior regularizations to constrain the learned policies close to the behavior policy in offline dataset [Fujimoto *et al.*, 2019; Wu *et al.*, 2019; Kumar *et al.*, 2019a; Fujimoto and Gu, 2021; Li *et al.*, 2023]. Some other works introduce pessimism during policy evaluation by assigning low confidences or low values for the value functions in out-of-distribution (OOD) regions [Kumar *et al.*, 2020; Kostrikov *et al.*, 2021a; Lyu *et al.*, 2022; An *et al.*, 2021; Bai *et al.*, 2021]. In-sample learning methods [Kostrikov

et al., 2021b; Xu *et al.*, 2023, 2022; Garg *et al.*, 2023; Xiao *et al.*, 2023] have recently emerged as an alternative, optimizing on samples exclusively from the offline dataset and thus eliminating any OOD value queries. Moreover, an alternative in-sample learning approach performs RL tasks via supervised learning by conditioning other available sources, also called Reinforcement Learning via Supervised Learning) (RvS) [Kumar *et al.*, 2019b; Schmidhuber, 2019; Emmons *et al.*, 2021; Feng *et al.*, 2022]. These approaches are super stable and easy to tune compared to other methods. Existing methods, however, rely on shallow MLPs to model their actors, performing well on simple tasks but falling short on more complex long-horizon planning tasks, such as `AntMaze` and `Kitchen` tasks. To combat this, one popular direction is to increase policy capacity by employing highly expressive models to accurately approximate the actor, which obtains certain degrees of performance gains [Wang *et al.*, 2022b; Hansen-Estruch *et al.*, 2023; Ada *et al.*, 2023; Chen *et al.*, 2023; Lu *et al.*, 2023]. However, this marginal improvement comes at the cost of extra model complexity and exponentially increased computational burden, inevitably limiting their applications.

Sequential Modeling for Offline RL. Different from traditional TD methods, this avenue treats offline RL problems as general sequence modeling tasks, with the motivation to generate a sequence of actions or trajectories that attain high rewards. In this formulation, shallow feedforward models are typically believed to suffer from a limited modeling horizon due to accumulated modeling errors caused by deficient model capacities [Janner *et al.*, 2019; Amos *et al.*, 2021]. To combat this, recent innovations in sequential modeling techniques shift towards achieving precise long-horizon modeling through the use of high-capacity sequence model architectures such as Transformer [Chen *et al.*, 2021; Janner *et al.*, 2021; Jiang *et al.*, 2022; Wang *et al.*, 2022a; Konan *et al.*, 2023; Hu *et al.*, 2023b; Wu *et al.*, 2023; Villaflor *et al.*, 2022] and diffusion models [Janner *et al.*, 2022; Ajay *et al.*, 2022; Niu *et al.*, 2024]. However, Fig. 1 manifests that the adoption of expressive models imposes a large amount of computational load and complexity. In this study, RSP adopts a recursive coarse-grained paradigm for sub-goal prediction solely with shallow 2-layer MLPs, achieving exceptional results while bypassing training inefficiency and high inference latency issues led by expressive models.

3 Preliminaries

In this paper, we consider the standard Markov Decision Process (MDP), denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \rho, r, \gamma)$, which is characterized by states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition probabilities $T(s_{t+1}|a_t, s_t)$, initial state distribution $\rho(s_0)$, reward function r and the discount factor $\gamma \in (0, 1)$. In each episode, the agent is given a reward function $r(s_t, a_t)$ and embodies with a policy $\pi(a \mid s)$ to interact with the environment. Let $\tau := (s_0, a_0, s_1, a_1, \cdots)$ denote an infinite length trajectory. We employ $\pi(\tau)$ to represent the probability of sampling trajectory τ from the policy $\pi(a \mid s)$. The primary goal is to optimize the expected cumulative discounted rewards incurred along a trajectory, where the objective and the corresponding Q-functions are expressed as:

$$\max_{\pi} \ \mathbb{E}_{\pi(\tau)}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\Big], \ Q^{\pi}(s, a) := \mathbb{E}_{\pi(\tau)}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\Big|_{a_0=a}^{s_0=s}\Big]. \tag{1}$$

We are interested in the offline setting in this work and the objective of offline RL is in line with traditional RL, but with infeasible additional online interactions. In the offline setting, a fixed dataset is given by $\mathcal{D} := \{\tau_1, \tau_2, \tau_3, \cdots\}$, which comprises multiple trajectories gathered from unknown behavior policies, denoted with $\beta(a|s)$ in this work.

3.1 RL via Goal-Conditioned Supervised Learning

Reformulating RL as GCSL, previous works [Schmidhuber, 2019; Emmons *et al.*, 2021] typically employ hindsight relabeling to perform conditional imitation learning. The core idea underlying this approach is that any trajectory can be considered a successful demonstration of reaching certain goals, i.e. future states or accumulating rewards within the same trajectory. GCSL demonstrates significant potential in the offline setting, largely due to the availability of that goal information [Kumar *et al.*, 2019b; Chen *et al.*, 2021; Emmons *et al.*, 2021; Feng *et al.*, 2022].

Algorithm 1 General Procedure of GCSL

1: **Input**: Offline dataset $\mathcal{D}$
2: RELABEL(τ), $\forall \tau \sim \mathcal{D}$ ▷ Step 1
3: Sample transitions $(s, a, e) \in \mathcal{D}$
4: Update $\pi(a \mid s, e)$ with Eq. 2 ▷ Step 2
5: **return** $\pi(a \mid s, e)$

The general procedure of GCSL is simple and straightforward, as shown in 1, consisting of two main steps. In the first step, GCSL relabels the dataset by adding an additional goal information e to each state-action pair and forms a state-action-goal tuple, i.e., (s, a, e). The goal can be the target state,

cumulative return, or language description [Lynch *et al.*, 2020; Ghosh *et al.*, 2021; Kumar *et al.*, 2019b; Chen *et al.*, 2021; Feng *et al.*, 2022], if applicable. For the state s_t, the goal e_t is randomly sampled from the future steps from the current step t along a valid trajectory in the dataset. The key idea behind this resampling is to consider any future goal e_t is reachable by taking action a_t and state s_t. In the second step, GCSL learns a goal-conditioned policy within the relabeled dataset via maximum log-likelihood:

$$\arg\max_{\pi} \mathbb{E}_{(s,a,e)\sim\mathcal{D}} \left[\log \pi_\phi(a \mid s, e)\right]. \tag{2}$$

4 Recursive Skip-Step Planning

4.1 Why Non-recursive Planning Might Struggle?

Fine-grained planning using single-step dynamics prediction, $f(s_{t+1}|s_t, g)$, often faces significant challenges due to the rapid accumulation of step-wise errors in sequential modeling, led by such a limited planning horizon. Moreover, the target goal g may provide insufficient guidance for s_{t+1} to fully capture long-horizon capabilities. For enhanced accuracy of single-step dynamics modeling, expressive architectures such as Transformers [Chen *et al.*, 2021; Janner *et al.*, 2021] and Diffusers [Janner *et al.*, 2022; Ajay *et al.*, 2022] are introduced with sophisticated mechanisms to ensure global consistency within entire sequences, while significantly increasing computational load and complexity. This has sparked a considerable amount of research aimed at reducing the complexity and improving the efficiency of expressive models. However, these efforts remain confined to the realm of *fine-grained* sequential modeling, which might be exactly the underlying source of all the challenges faced. As first principles suggest [Spencer, 1904], current solutions only treat its symptoms, potentially ignoring a more fundamental question: *why not attempt to bypass fine-grained modeling entirely?*

In contrast, coarse-grained planning alleviates these problems by employing a skip-step dynamics model, $f(s_{t+k}|s_t, g)$, to predict intermediate sub-goals that fit skip-step states. This approach extends the prediction horizon and reduces the frequency of predictions during planning, thereby minimizing the compounding errors associated with multiple single-step predictions. However, horizon selection is challenging and involves significant trade-offs. In other words, while a larger horizon k might better align with long-horizon dynamics, it should not be excessively large. When the skip-step state s_{t+k} is too distant from the current state s_t, the learning signal may become too weak to provide adequate supervision for the initial steps of downstream execution. This can lead to increased policy approximation errors, particularly in long-horizon tasks. In essence, the challenge lies in balancing the approximation error of the next-level policy $\pi(a_t|s_t, s_{t+k}, g)$ with that of the current-level sub-goal prediction $f(s_{t+k}|s, g)$.

To address this dilemma, we can treat the skip-step planning sequences $(s_t, s_{t+k}, s_{t+2k}, \dots)$ as a high-level "single-step" sequence under a coarse-grained MDP, which may still suffer from compounding errors in long-horizon planning. Since coarse-grained planning generally leads to milder sub-goal error accumulation compared to fine-grained approaches as analysed above, we can introduce a higher-level skip-step dynamics model that predicts sub-goals over a longer horizon, i.e., $f'(s_{t+k'}|s_t, g)$ where $k' > k$, and use it to predict the current sub-goal with $f(s_{t+k}|s_t, s_{t+k'}, g)$. Intuitively, replacing $f(s_{t+k}|s_t, g)$ with $f(s_{t+k}|s_t, s_{t+k'}, g)$ could offer greater stability and supervision from the target goal, thereby reducing the approximation error of s_{t+k}. However, this introduces a new balance between the approximation errors of $f(s_{t+k}|s_t, s_{t+k'}, g)$ and $f'(s_{t+k'}|s_t, g)$. Naturally, we can recursively sample sub-goal states from longer horizons to shorter ones using skip-step dynamics models to minimize the approximation error of the current-level sub-goal. In Section 4.4, we further demonstrate through experiments how the recursive process and skip-step dynamics modeling are equally crucial in reducing accumulated errors in long-horizon planning.

4.2 Coarse-grained Planning in A Recursive Way

In the RSP framework, each current state s_t in the dataset is first relabeled with a fixed-horizon future target state s_{t+k} as the sub-goal, resulting in an augmented dataset $\mathcal{D} = \{(s_t, a_t, s_{t+k}, g)\}$. This allows us to revise the optimization objective of goal-conditioned policies in Equation 2 to focus on goal-conditioned sub-goal prediction:

$$\hat{f} = \arg\max_{f} \mathbb{E}_{(s_t, s_{t+k}, g)\sim\mathcal{D}} \left[\log f(s_{t+k}|s_t, g)\right]. \tag{3}$$

As explained in the previous section, planning with a learned coarse-grained dynamics model $\hat{f}$ is insufficient for balancing the minimization of action and sub-goal prediction errors. To further reduce the sub-goal prediction error, we extend it to a recursive skip-step prediction strategy, as depicted in Fig. 2, predicting high-level sub-goals to stabilize the lower-level sub-goal predictions in sequential modeling. To standardize the recursive process, we set $K^{(n)} := t + K/2^n, n \in [0, \cdots, N-1]$ where N is the recursion depth, K is the skip step of the highest-level sub-goal, and $s_{K^{(n)}}$ are sub-goal ground truths. The highest-level sub-goal skips K steps while the lowest-level sub-goal skips $k = K/2^{N-1}$ steps from the current step. For N-level recursion, we need to relabel the dataset $\mathcal{D} = \{(s_t, a_t, g)\}$ with additional N skip-step sub-goal ground truths:

$$\mathcal{D} \leftarrow \left\{ \left(a_t, \kappa_t^{(N)} := (s_t, s_{K^{(N-1)}}, \cdots, s_{K^{(0)}}, g) \right) \right\} \tag{4}$$

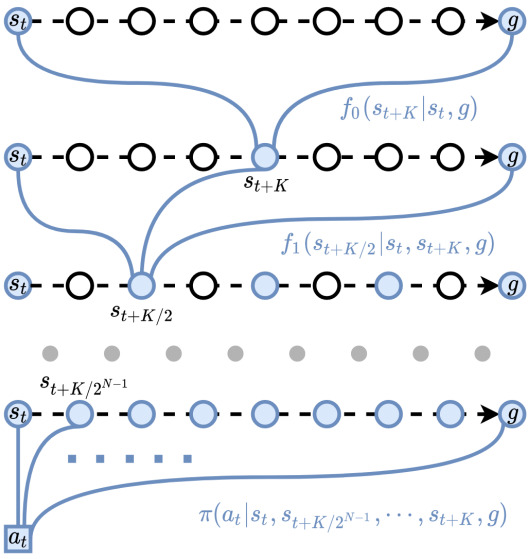

Figure 2: The illustration of Recursive Skip-Step Planning

where $\kappa_t^{(0)} = (s_t, g)$; $\forall n \in [1, \ldots, N]$, $\kappa_t^{(n)} = s_{K^{(n-1)}} \cup \kappa_t^{(n-1)}$ contains all the information required by the n-th level dynamics model to fit the sub-goal ground truth $s_{K^{(n-1)}}$. For brevity, we omit t and refer to $\kappa^{(n)}$ in the following descriptions. With all the definitions above, the highest(first)-level coarse-grained dynamics model could be denoted by $\hat{f}_0(s_{K^{(0)}}|s_t, g)$. Subsequently, we can learn more coarse-grained dynamics models to push the limits of the benefits of coarse-grained planning, as depicted in Fig. 2. In terms of the n-th dynamics model, the recursion solution naturally extends to the optimization objective in Eq. 5:

$$\max_{f_n} \mathbb{E}_{\kappa^{(n)} \sim \mathcal{D}} \left[\log f_n(s_{K^{(n-1)}}|\kappa^{(n-1)}) \right], \forall n \in [1, \ldots, N]. \tag{5}$$

where the current level sub-goal generation is conditioned on all the sub-goals predicted with higher-level dynamics for stability, which proves a significant design in our experiments.

During the evaluation process, we can formulate the recursive sub-goal planning as:

$$\widehat{s_{K^{(n-1)}}} = \arg\max_{s_{K^{(n-1)}}} \hat{f}_n(s_{K^{(n-1)}}|\widehat{\kappa^{(n-1)}}), \forall n \in [1, \cdots, N] \tag{6}$$

where $\widehat{\kappa^{(0)}} = \kappa^{(0)} = (s_t, g)$; $\forall n \in [1, \ldots, N]$, $\widehat{\kappa^{(n)}} = \widehat{s_{K^{(n-1)}}} \cup \widehat{\kappa^{(n-1)}}$. Eventually, we get the predicted sub-goals and leverage $\widehat{\kappa^{(N)}}$ for policy extraction.

4.3 Policy Extraction

Different from other planning methods, the coarse-grained planning approach employed by RSP does not explicitly integrate the corresponding policy into the learning process, necessitating a separate policy extraction step. To maintain the simplicity and efficiency of RSP, we extract the policy by maximizing the log-likelihood within a supervised learning framework. We introduce a goal-conditioned policy, distinct in that it incorporates all the sub-goals planned from prior coarse-grained dynamics models:

$$\max_{\phi} \mathbb{E}_{(a_t, \kappa^{(N)}) \sim \mathcal{D}} \left[\log \pi_\phi(a_t \mid \kappa^{(N)}) \right]. \tag{7}$$

At the evaluation stage, given the current state s_t, the action is determined by both the coarse-grained dynamics model and the goal-conditioned policy with:

$$\widehat{a_t} = \arg\max_{a_t} \widehat{\pi_\phi}(a_t|\widehat{\kappa^{(N)}})) \tag{8}$$

where $\widehat{\kappa^{(N)}} = (s_t, \widehat{s_{K^{(N-1)}}}, \cdots, \widehat{s_{K^{(0)}}}, g)$ is obtained from Eq. 6 recursively.

Algorithm 2 Recursive Skip-Step Planning (RSP)

Input: Dataset $\mathcal{D}$, Recursion depth N, Fixed planning step K
Relabel Dataset with Eq. 4 ▷ Step 1
`// Training` ▷ Step 2
for every training step **do**
  Sample transitions $(s_t, a_t, s_{K^{(0)}}, \cdots, s_{K^{(N-1)}}, g) \in \mathcal{D}$
  for $n = 1, \cdots, N$ **do**
    Update the n-th dynamics model $\hat{f}_n$ by Eq. 5
  end for
  Update π_ϕ by Eq. 7
end for
`// Evaluation` ▷ Step 3
Get initial state s, set d as False
while not d **do**
  Get action a from Eq. 8
  Roll out a and get (s', r, d)
  Set $s = s'$
end while

The final procedure in Algorithm 2 consists of three steps. Initially, it relabels the dataset $\mathcal{D}$, augmenting each transition with the future skip-state sub-goals and final goal information. Subsequently, RSP learns N coarse-grained dynamics models from Eq. 5 and a goal-conditioned policy from Eq. 7 that determines the action based on the sub-goals predicted with all the dynamics models. Finally, RSP plans the sub-goal sequence recursively with Eq. 6 and executes action from sub-goals with Eq. 8.

4.4 Discussions

In this section, we provide an in-depth explanation of why recursive skip-step planning might outperform from the perspective of model rollout prediction errors. We conduct experiments on different (lowest-level) skip-step horizons ($k = 4, 8, 16, 32, 64$) and recursion depths ($N = 1, 2, 3, 4$) in the long-horizon `Antmaze-Ultra-Diverse` task, where trajectories can extend up to 3000 steps. For example, [16,32,64] in Fig. 3 denotes $N = 3$ and $k = 16$. We plot the root mean squared errors (RMSE) between the model rollout predictions and the ground truths of skip-step state trajectories. Specifically, we obtain the next skip-step prediction s_{t+k} through the recursive approach of RSP; we iteratively query the next skip-step prediction s_{t+nk} from the last prediction $s_{t+(n-1)k}$ and finally obtain model rollout predictions of skip-step state trajectories $s_t, \cdots, s_{t+(n-1)k}, s_{t+nk}, \cdots, s_{t+1024}$; then calculate RMSE between the predictions and ground truths of those trajectories.

In Fig. 3, as k increases from 4 to 64, the accuracy of the model rollout prediction improves, aligning more closely with the ground truth over longer-horizon rollouts. This suggests that coarse-grained prediction outperforms fine-grained prediction in mitigating accumulated compounding errors as the skip-step horizon k increases, thereby enhancing long-horizon planning capabilities. Besides, we observe that the RMSE of model rollout predictions achieves a lower asymptotic value and remains closer to the ground truth over longer-horizon rollouts as the recursion depth increases, demonstrating the effectiveness of recursive planning.

Choices of Recursion Depth and Skip-Step Horizon. At first glance, Fig. 3 indicates that increasing recursion depth and skip-step horizon improves the planning performance of RSP. However, these plots assume that an oracle policy model perfectly follow the skip-step state predictions. In practice, deviations from optimal state trajectories can easily occur due to suboptimal policies, especially when the policy model $\pi(a_t|s_t, s_{t+k}, \cdots, g)$ is conditioned on a distant skip-step state prediction s_{t+k}, which offers minimal guidance for current actions. Therefore, selecting recursion depth and horizon requires to strike the delicate balance between the expressiveness of policy models and the long-horizon capability of dynamics models. Given that we use a 2-layer MLP for policy learning for simplicity in experiments, we opt for $k = 32$ and $N = 1$ in recursive skip-step dynamics models across all tasks in order to provide sufficient long-horizon planning ability while easing the burden for policy learning. Practitioners using more expressive policy models and advanced policy learning methods may consider enlarging the horizon and deepening the recursion process accordingly.

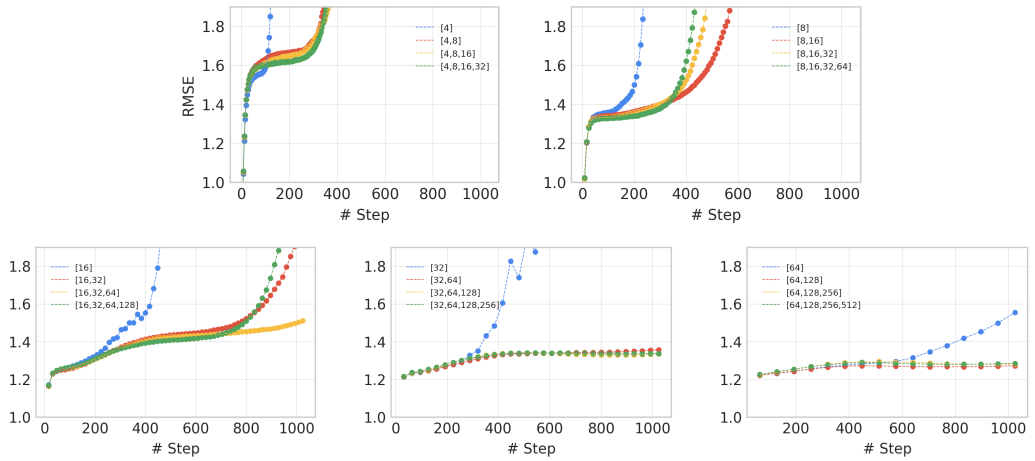

Figure 3: Root mean squared error (RMSE) of coarse and fine-grained dynamics predictors on `Antmaze-Ultra`. Averaged over 10 random seeds.

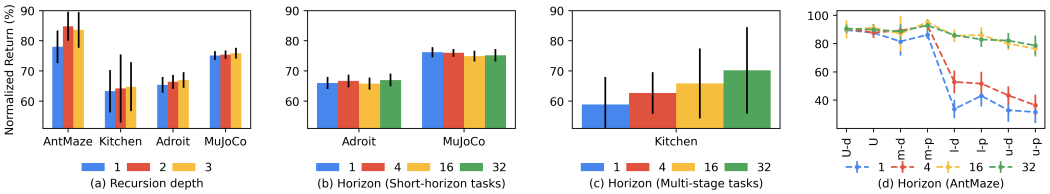

Figure 4: Effects of varying recursion depth and skip-step horizon.

5 Experiments

5.1 D4RL Benchmark Results

Evaluation Setups. We evaluate the performance of various methods using a diverse set of challenging tasks from D4RL benchmark [Fu *et al.*, 2020], including `Antmaze navigation`, `Kitchen Manipulation`, `Adroit hand manipulation`, and `Mujoco locomotion` tasks. The Antmaze navigation tasks require intricate long-horizon planning to guide an ant navigate from the starting point to the goal location, which are pretty challenging for traditional offline RL to solve. To test the planning capabilities on extremely long-horizon tasks ($>$ 2000 steps), we consider a more extensive and challenging "Ultra" Antmaze environment. Kitchen tasks involves a 9-DoF Franka robot interacting with a kitchen scene to complete a combination of household sub-tasks, which requires multi-skill long-horizon composition ability. In contrast, the Adroit hand manipulation tasks involve controlling a dexterous robot hand to complete tasks like opening a door or rotating a pen, with planning horizons typically ranging from 100 to 200 steps, which are relatively simple. Finally, the Mujoco locomotion tasks are believed as the simplest tasks among all these tasks. They only focus on controlling different robots to move forward as fast as possible, which are relatively simple since the control pattern is repetitive. To assess performance, we normalize the final scores by following the benchmark standards [Fu *et al.*, 2020], where 0 represents random policy performance and 100 means expert-level performance. Our evaluations encompass 10 random seeds, with mean and standard deviation reported. The evaluation consists of 100 episodes per seed for every task.

Baselines. We compare RSP against the following state-of-the-art (SOTA) baselines. *BC* is the simplest imitation learning approach that imitates the offline dataset; *RvS* [Emmons *et al.*, 2022] is an SOTA conditioned imitation learning method that casts offline RL as supervised learning problems; *CQL* [Kumar *et al.*, 2020] is an offline RL method that penalizes Q-values for OOD regions; *IQL* [Kostrikov *et al.*, 2021b] is an SOTA in-sample learning offline RL method; *DT* [Chen *et al.*, 2021] and *TT* [Janner *et al.*, 2021] are two methods that utilize transformer [Vaswani *et al.*, 2017] architecture, regarding offline RL problems as sequential modeling problems;*TAP* [Zhang *et al.*, 2022] is also a transformer-based method that plans in a latent space encoded by VQ-VAE [Van Den Oord *et al.*, 2017]; *Diffuser* [Janner *et al.*, 2022] builds on top of diffusion model, directly generating the entire fixed-length trajectory and executing the first step.

Table 1: Comparison of performance in AntMaze navigation, Kitchen manipulation, Mujoco locomotion tasks, and Adroit hand manipulation tasks from the D4RL dataset. AntMaze tasks require long-horizon planning to navigate from the starting point to the goal. Adroit tasks involve high-dimensional action space control of a 24-DoF robot hand. Baselines in **teal** utilize expressive models including Transformer or diffusion models. Emphasis is placed on scores within 5% of the maximum for locomotion tasks following, while maximum scores are highlighted for other tasks. $+G$ denotes goal-conditioned methods. $+Q$ denotes using Q-function as a search heuristic following [Janner *et al.*, 2021]. All results of baselines are from their original papers, and we use "-" if they do not report scores on the specific tasks. Note that only a few baseline methods report their performance on the challenging Adroit tasks, so we only compare with baselines having reported scores.

Dataset	Environment	BC	RvS $(+G)$	CQL	IQL	HIQL	DT	TT $(+Q)$	TAP $(+G)$	RSP (Ours) $(+G)$
Umaze	AntMaze	65.0	65.4	74.0	87.5	–	59.2	**100.0**	–	88.2 ± 5.7
Umaze-Diverse	AntMaze	55.0	60.9	84.0	66.2	–	53.2	–	–	**92.9** ± 3.5
Medium-Play	AntMaze	0.0	58.1	61.2	71.2	84.1	0.0	**93.3**	78.0	91.4 ± 8.8
Medium-Diverse	AntMaze	0.0	67.3	53.7	70.0	86.8	0.0	**100.0**	85.0	92.7 ± 3.4
Large-Play	AntMaze	0.0	32.4	15.8	39.6	86.1	0.0	66.7	74.0	**83.0** ± 3.7
Large-Diverse	AntMaze	0.0	32.9	14.9	47.5	88.2	0.0	60.0	82.0	**86.0** ± 3.8
Ultra-Play	AntMaze	0.0	33.2	–	8.3	39.2	0.0	20.0	22.0	**80.3** ± 4.2
Ultra-Diverse	AntMaze	0.0	31.6	–	15.6	52.9	0.0	33.3	26.0	**80.5** ± 4.1
Average		15.0	47.7	50.6	50.7	–	14.0	67.6	61.2	**86.9** ± 4.7
Partial	Kitchen	38.0	51.4	20.8	46.3	65.0	–	–	–	**76.5** ± 8.1
Mixed	Kitchen	51.5	60.3	24.2	51.0	**67.7**	–	–	–	64.0 ± 20.4
Average		44.8	55.9	22.5	48.7	66.4	–	–	–	**70.2** ± 14.3

Dataset	Environment	BC	RvS	CQL	IQL	Diffuser	DT	TT	TAP	RSP (Ours)
Medium-Expert	HalfCheetah	55.2	**92.2**	**91.6**	86.7	88.9	86.8	**95.0**	**91.8**	**92.5** ± 0.5
Medium-Expert	Hopper	52.5	101.7	**105.4**	91.5	103.3	**107.6**	**110.0**	**105.5**	**109.6** ± 0.7
Medium-Expert	Walker2d	**107.5**	**106.0**	**108.8**	**109.6**	**106.9**	**108.1**	101.9	**107.4**	**106.7** ± 1.0
Medium	HalfCheetah	42.6	41.6	44.0	**47.4**	42.8	42.6	**46.9**	45.0	42.9 ± 0.2
Medium	Hopper	52.9	60.2	58.5	66.3	**74.3**	67.6	61.1	63.4	63.4 ± 2.5
Medium	Walker2d	75.3	71.7	72.5	**78.3**	**79.6**	74.0	**79.0**	64.9	**76.6** ± 1.6
Medium-Replay	HalfCheetah	36.6	38.0	**45.5**	**44.2**	37.7	36.6	41.9	40.8	40.4 ± 0.4
Medium-Replay	Hopper	18.1	73.5	**95.0**	**94.7**	**93.6**	82.7	**91.5**	87.3	88.2 ± 5.6
Medium-Replay	Walker2d	26.0	60.6	77.2	73.9	70.6	66.6	**82.6**	66.8	66.9 ± 2.9
Average		51.9	71.7	**77.6**	**77.0**	**77.5**	74.7	**78.9**	74.8	**76.4** ± 1.7

Dataset	Environment	BC	CQL	IQL	TT	TAP	RSP (Ours)
Cloned	Pen	56.9	39.2	37.3	11.4	57.4	**67.9** ± 6.9
Cloned	Hammer	0.8	2.1	2.1	0.5	1.2	**3.6** ± 3.1
Cloned	Door	−0.1	0.4	1.6	−0.1	**11.7**	0.8 ± 0.6
Cloned	Relocate	−0.1	−0.1	−0.2	−0.1	−0.2	**0.0** ± 0.0
Expert	Pen	85.1	107.0	**133.1**	72.0	127.4	120.9 ± 4.6
Expert	Hammer	125.6	86.7	119.5	15.5	127.6	**129.7** ± 0.9
Expert	Door	34.9	101.5	**105.7**	94.1	104.8	105.0 ± 0.3
Expert	Relocate	101.3	95.0	106.1	10.3	105.8	**108.1** ± 0.4
Average		50.6	54.0	63.1	25.4	**67.0**	**67.0** ± 2.1

Main Results. We present the evaluation results on the D4RL benchmark are presented in Table 1. The results demonstrate that RSP consistently outperforms or obtains on-par performance compared to other baselines. In particular, RSP achieves unparalleled success in complex `Antmaze` navigation and `Kitchen` manipulation tasks that necessitate long-horizon and/or multi-stage planning. For instance, all baselines fail miserably on the extremely challenging `Antmaze Ultra` tasks, while RSP can obtain similar success rates as it achieves on the `Antmaze Large` tasks. This indicates that the increased difficulty resulting from longer planning horizons has a marginal impact for RSP, demonstrating the superior long-horizon planning capabilities of RSP. Apart from the long-horizon tasks, RSP also demonstrates consistently good performance on Adroit hand manipulation and Mujoco locomotion tasks. On these tasks, Table 1 shows that even with much simpler model architecture, RSP can surprisingly obtain on-par performance results compared to RL as sequential modeling methods that utilize expressive models such as TT, TAP, and Diffuser, as well as TD-based offline RL baselines such as CQL and IQL.

5.2 Training Cost and Inference Latency

To further showcase the advantages of RSP, we monitor the training time and inference latency of different methods on `Antmaze-Large-Diverse` task and illustrate the connections between training time, inference latency and the performance for different methods in Fig. 1. From Fig. 1, we can clearly observe that by leveraging expressive models for sequential modeling, one can sometimes obtain minor improvements in policy performance againts TD-based offline RL approaches. However, these marginal gains come at the expense of exponentially increased training time and inference latency. This inevitably imposes significant computational burdens, restricting the applicability of expressive models in many real-world scenarios. In contrast, RSP obtains the best performance and meanwhile enjoys fast training and minimal inference latency. In particular, RSP completes the training in approximately 180 seconds and exhibits an inference latency of under 1ms, highlighting its exceptional computational efficiency. This sheds light on its potential real-world applicability, given its impressive performance, simplicity, and efficiency.

5.3 Ablation Studies

In this section, we conduct extensive ablation studies on the critical components of RSP including the recursion depth N and the skip-step horizon of the lowest-level sub-goal for policy extraction k. The results are presented in Fig. 4.

We first fix the horizon at 8 and conduct ablation studies on different recursion depths: 1, 2, and 3. The results in Fig. 4(a) indicate that a deeper recursion process consistently improves performance. However, increasing recursion depth beyond a certain threshold can introduce compounding errors within the recursive procedure. Additionally, policy extraction error may become an uncertain factor during the execution of recursive skip-step planning, even though the model rollout prediction error of states decreases with deeper recursion as we analyze in Section 4.4. These factors might diminish the expected improvement in policy performance as recursion depth increases. For instance, a recursion depth of 3 performs worse than a depth of 2 in the Antmaze task.

Next, we fix the recursion depth at 1 and ablate on the choices of horizon lengths: 1, 4, 16, and 32 steps. A larger horizon corresponds to more coarse-grained planning. We observe that horizon length has minimal impact on short-horizon tasks, such as Adroit and MuJoCo, as shown in Fig. 4(b). However, significantly reducing the planning horizon leads to performance degradation in long-horizon tasks, such as Antmaze and Kitchen, as seen in Fig. 4(c) and (d).

6 Conclusion

In this study, we propose a novel recursive skip-step planning framework for offline RL, which leverages a set of coarse-grained dynamics models to perform recursive sub-goal prediction, and use a goal-conditioned policy to extract planned actions based on these sub-goals. Our method can smartly bypass the long-horizon compounding error issue in existing sequence modeling-based offline RL methods through hierarchical recursive planning, while still maintaining the advantage of learning in a completely supervised manner. Notably, even using lightweight MLP networks, our method can provide comparable or even better performance as compared to sequence modeling methods that use heavy architectures like Transformer or diffusion models, while providing much better training and inference efficiency. Our work highlights the need to rethink the existing design principles for offline RL algorithms: instead of relying on increasingly heavier models, maybe it's time to introduce more elegant and lightweight modeling schemes to tackle existing challenges in offline RL.

References

Suzan Ece Ada, Erhan Oztop, and Emre Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *arXiv preprint arXiv:2307.04726*, 2023. 1, 3

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022. 2, 3, 4

Brandon Amos, Samuel Stanton, Denis Yarats, and Andrew Gordon Wilson. On the model-based stochastic value gradient for continuous reinforcement learning. In *Learning for Dynamics and Control*, pages 6–20. PMLR, 2021. 3

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Proc. of NeurIPS*, 2021. 2

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhi-Hong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *Proc. of ICLR*, 2021. 2

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. 2, 3, 4, 7

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *International Conference on Learning Representations*, 2023. 1, 3

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021. 3

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *International Conference on Learning Representations*, 2022. 7

Xiaoyun Feng, Li Jiang, Xudong Yu, Haoran Xu, Xiaoyan Sun, Jie Wang, Xianyuan Zhan, and Wai Kin Victor Chan. Curriculum goal-conditioned imitation for offline reinforcement learning. *IEEE Transactions on Games*, 2022. 3, 4

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *ArXiv preprint*, 2020. 2, 7, 13

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *ArXiv preprint*, 2021. 2

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proc. of ICML*, pages 2052–2062, 2019. 1, 2

Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. *arXiv preprint arXiv:2301.02328*, 2023. 3

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *Proc. of ICLR*, 2021. 4

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. 1, 3

Jifeng Hu, Yanchao Sun, Sili Huang, SiYuan Guo, Hechang Chen, Li Shen, Lichao Sun, Yi Chang, and Dacheng Tao. Instructed diffuser with temporal condition guidance for offline reinforcement learning. *arXiv preprint arXiv:2306.04875*, 2023. 1

Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Graph decision transformer. *arXiv preprint arXiv:2303.03747*, 2023. 3

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019. 3

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Proc. of NeurIPS*, 2021. 2, 3, 4, 7, 8

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. 2, 3, 4, 7

Zhengyao Jiang, Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291*, 2022. 2, 3

Sachin G Konan, Esmaeil Seraj, and Matthew Gombolay. Contrastive decision transformers. In *Conference on Robot Learning*, pages 2159–2169. PMLR, 2023. 3

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *Proc. of ICML*, pages 5774–5783, 2021. 2

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *ArXiv preprint*, 2021. 2, 7

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Proc. of NeurIPS*, pages 11761–11771, 2019. 1, 2

Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019. 3, 4

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proc. of NeurIPS*, 2020. 2, 7

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv preprint*, 2020. 1, 2

Jianxiong Li, Xianyuan Zhan, Haoran Xu, Xiangyu Zhu, Jingjing Liu, and Ya-Qin Zhang. When data geometry meets deep function: Generalizing offline reinforcement learning. In *Proc. of ICLR*, 2023. 1, 2

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. *arXiv preprint arXiv:2304.12824*, 2023. 3

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, 2020. 4

Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2206.04745*, 2022. 2

Bogdan Mazoure, Walter Talbott, Miguel Angel Bautista, Devon Hjelm, Alexander Toshev, and Josh Susskind. Value function estimation using conditional diffusion models for control. *arXiv preprint arXiv:2306.07290*, 2023. 2

Haoyi Niu, Qimao Chen, Tenglong Liu, Jianxiong Li, Guyue Zhou, Yi ZHANG, Jianming HU, and Xianyuan Zhan. xTED: Cross-domain adaptation via diffusion-based trajectory editing. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024. 2, 3

Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019. 3

Herbert Spencer. *First principles*, volume 1. JA Hill, 1904. 4

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 7

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 7

Adam R Villaflor, Zhe Huang, Swapnil Pande, John M Dolan, and Jeff Schneider. Addressing optimism bias in sequence modeling for reinforcement learning. In *international conference on machine learning*, pages 22270–22283. PMLR, 2022. 2, 3

Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. Bootstrapped transformer for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:34748–34761, 2022. 3

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022. 1, 2, 3

Han Wang, Youfang Lin, Sheng Han, and Kai Lv. Offline reinforcement learning with diffusion-based behavior cloning term. In *International Conference on Knowledge Science, Engineering and Management*, pages 267–278. Springer, 2023. 1

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *ArXiv preprint*, 2019. 2

Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. *arXiv preprint arXiv:2307.02484*, 2023. 3

Chenjun Xiao, Han Wang, Yangchen Pan, Adam White, and Martha White. The in-sample softmax for offline reinforcement learning. *arXiv preprint arXiv:2302.14372*, 2023. 3

Haoran Xu, Li Jiang, Jianxiong Li, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. In *Proc. of NeurIPS*, 2022. 3

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. *arXiv preprint arXiv:2303.15810*, 2023. 3

Xianyuan Zhan, Haoran Xu, Yue Zhang, Xiangyu Zhu, Honglei Yin, and Yu Zheng. Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning. In *Proc. of AAAI*, pages 4680–4688, 2022. 1, 2

Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, Yuandong Tian, et al. Efficient planning in a compact latent action space. In *The Eleventh International Conference on Learning Representations*, 2022. 2, 7, 13

A Experimental Details

A.1 Training Implementations

We use feed-forward MLPs for all our policies and dynamics models, with two layer neural networks with 1024 units each and ReLU nonlinearities used for all tasks. We list the basic hyperparameters in 2. We try to keep the same series of hyperparameters for all the experiment tasks but in `Kitchen` environment we have to add dropouts and decrease the batch size for less overfitting due to the extremely limit dataset provided by D4RL [Fu *et al.*, 2020]. For the recursive dynamics models, [32] denotes "Recursion step $N = 1$" and "Horizon $k = 32$", which implies that we use one dynamics model to predict the probability of the state after 32 steps, i.e. $f(s_{t+32}|s_t, g)$, and extract actions from $\pi(a_t|s_t, s_{t+32}, g)$. Similarly, [8,16,32] denotes "Recursion Depth $N = 3$" and "Horizon $k = 8$". For consistency, we adopt [32] across all the tasks in our experiments and achieve superior performance.

Table 2: Hyperparameters.

Hyper-parameter	Value
MuJoCo & Adroit & Antmaze	
Dropout	0.0
Batch size	16384
Kitchen	
Dropout	0.1
Batch size	256
Shared	
Learning rate	1e-3
Learning rate schedule	cosine
Hidden layers	[1024,1024]
Recursive skip-step dynamics	[32]

A.2 Evaluation Setups

We evaluate the performance of various methods using a diverse set of challenging tasks from D4RL benchmark [Fu *et al.*, 2020], including `Antmaze navigation`, `Kitchen manipulation`, `Adroit hand manipulation`, and `Mujoco locomotion` tasks.

The `Antmaze navigation` tasks require intricate long-horizon planning to guide an ant navigate from the starting point to the goal location, which is pretty challenging for traditional offline RL to solve. To test the planning capabilities on extremely long-horizon tasks (2000 steps), we consider a more extensive and challenging "Ultra" Antmaze environment. To test multi-stage multi-skill planning capabilities, we consider the challenging `Kitchen manipulation` tasks require long-horizon skill composition ability for a 9-DoF Franka robot to sequentially finish four household sub-tasks in a simulated kitchen. In contrast, the `Adroit hand manipulation` tasks involve controlling a dexterous robot hand to complete single-skill tasks like opening a door or rotating a pen, with planning horizons typically ranging from 100 to 200 steps, which are relatively simple. The `Mujoco locomotion` tasks are believed the simplest among all these tasks. They only focus on controlling different robots to move forward as fast as possible, which are relatively simple since the control pattern is repetitive and short-horizon. To assess performance, we normalize the final scores by following the benchmark standards [Fu *et al.*, 2020], where 0 represents random policy performance and 100 means expert-level performance. Our evaluations encompass 10 random seeds, with mean and standard deviation reported. The evaluation process consists of 100 episodes per seed for every task. Across all the tasks mentioned, our experimental results in Section *Experiments* demonstrate that RSP excels particularly in complex, long-horizon, multi-stage tasks such as `Antmaze` and `Kitchen`.

A.3 Baseline Scores

On AntMaze-Ultra task, results of BC, DT, CQL and RvS are from our own reproduction while those of IQL, TT, TAP are from TAP paper [Zhang *et al.*, 2022]. On Adroit-Expert task, the results are from our own reproduction. Other baseline results are collected from the original papers.

The baseline repositories we use for reproduction are listed below:

BC: https://github.com/Farama-Foundation/D4RL-Evaluations.

RvS: https://github.com/scottemmons/rvs

DT: https://github.com/kzl/decision-transformer

TT: https://github.com/jannerm/trajectory-transformer

TAP: https://github.com/ZhengyaoJiang/latentplan

IQL: https://github.com/ikostrikov/implicit_q_learning

A.4 Devices

All the results are obtained on a server with 512G RAM, $4\times$ A800 PCIe 80GiB (GPU) and $2\times$ AMD EPYC 7T83 64-Core Processor (CPU). The inference latency and training time shown in Fig. 1 were recorded while each model was trained or inferred using a single GPU on the server, with no other GPUs active during the process.