Rainbow-DemoRL: Combining Improvements in Demonstration-Augmented Reinforcement Learning

Dwait Bhatt[1], Shih-Chieh Chou[2], and Nikolay Atanasov[1]

***Abstract*— Several approaches have been proposed to improve the sample efficiency of online reinforcement learning (RL) by leveraging demonstrations collected offline. The offline data can be used directly as transitions to optimize RL objectives, or offline policy and value functions can first be learned from the data and then used for online finetuning or to provide reference actions. While each of these strategies has shown compelling results, it is unclear which method has the most impact on sample efficiency, whether these approaches can be combined, and if there are cumulative benefits. We classify existing demonstration-augmented RL approaches into three categories and perform an extensive empirical study of their strengths, weaknesses, and combinations to isolate the contribution of each strategy and determine effective hybrid combinations for sample-efficient online RL. Our analysis reveals that directly reusing offline data and initializing with behavior cloning consistently outperform more complex offline RL pretraining methods for improving online sample efficiency.**

I. Introduction

Online reinforcement learning (RL) [1] has demonstrated impressive performance in game settings [2]–[4] and, more recently, has shown potential to advance control of physical robot systems [5]–[7]. A key benefit of online RL is that it enables an intelligent system to adapt continuously to changing conditions by learning directly from ongoing interactions. This has long been viewed as having potential to enable continuous lifelong learning in robotics [8]. However, two major challenges prevent widespread deployment of online RL on robot systems: (a) low sample-efficiency, which makes learning from scratch on physical hardware prohibitively slow and costly, and (b) the need for random exploration during learning, which can lead to unsafe behaviors when interacting with the physical world.

To address these challenges, a promising direction has been to leverage offline demonstrations to accelerate and improve online RL. Several diverse strategies have been proposed for this purpose. First, some methods use demonstration data directly within the online training loop. This can be done by pre-filling the replay buffer with expert transitions to bootstrap learning from high-quality examples [9] or by adding an auxiliary supervised loss to the RL objective to encourage the agent to mimic the expert's actions [10]. Second, another class of methods uses the data to learn an initial policy or value function via offline RL, which then provides a strong starting point for online finetuning [11], [12]. Finally, a third approach uses a pre-trained offline policy as a reference for action selection, mixing its outputs with the online policy's actions to guide exploration and improve performance [13]–[15].

While each of these strategies has shown compelling results, they are not mutually exclusive, and many hybrid approaches are possible. This creates a critical knowledge gap: it is unclear which components are complementary and how each one contributes to final performance and learning speed. The current state of demonstration-augmented RL is reminiscent of the state of deep RL in 2017 when several independent extensions of Deep Q-Networks (DQN) [2] were explored in the literature. Rainbow-DQN [16] systematically combined these extensions and performed an ablation study that clarified the individual importance of each component.

Inspired by this line of work, we conduct a large-scale empirical study of demonstration-augmented online RL with the following objectives:

- identify general classes of techniques for demonstration-augmented RL in the relevant prior work,
- measure the contribution of each strategy to performance and sample efficiency via extensive evaluation,
- propose effective hybrid combinations of strategies for sample-efficient online learning in robot manipulation.

II. Problem Statement

Consider a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, p, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ are continuous state and action spaces, $r(s_t, a_t)$ and $p(s_{t+1}|s_t, a_t)$ are the reward function and the transition probability density function, where $s_t, s_{t+1} \in \mathcal{S}$, $a_t \in \mathcal{A}$, and $\gamma \in [0, 1)$ is the discount factor. The goal in RL is to learn a stochastic policy $\pi(a|s)$ or a deterministic policy $\pi(s)$ to generate actions that maximize the value function $V^\pi(s) = \mathbb{E}_{s_{t+1}\sim p(\cdot|s_t,a_t), a_t\sim\pi(\cdot|s_t)}[\sum_t \gamma^t r(s_t, a_t)|s_0 = s]$. The action value or Q-value of the policy is given by $Q^\pi(s, a) = \mathbb{E}_{s_{t+1}\sim p(\cdot|s_t,a_t), a_t\sim\pi(\cdot|s_t)}[\sum_t \gamma^t r(s_t, a_t)|s_0 = s, a_0 = a]$. We use $\pi(a|s; \phi)$ or $\pi(s; \phi)$ to denote a stochastic/deterministic policy parameterized by a neural network with weights ϕ and $Q^\pi(s, a; \theta)$ to denote its Q-value estimate parameterized by a neural network with weights θ.

In this paper, we aim to learn a policy π_{on} and its value function Q_{on} by sampling transitions (s_t, a_t, r_t, s_{t+1}) from $\mathcal{M}$ online such that $s_t \in \mathcal{S}$, $a_t \sim \pi_{\text{on}}(\cdot|s_t)$, $r_t = r(s_t, a_t)$,

This work was supported by Technology Innovation Program 20018112 (Development of autonomous manipulation and gripping technology using imitation learning based on visual and tactile sensing) funded by the Ministry of Trade, Industry & Energy (MOTIE), Korea.

[1]Dwait Bhatt and Nikolay Atanasov are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA, USA, e-mails: {dhbhatt,natanasov}@ucsd.edu.

[2]Shih-Chieh Chou is with the Department of Mechanical and Electro-Mechanical Engineering, National Sun Yat-sen University, Taiwan.

$s_{t+1} \sim p(\cdot|s_t, a_t)$. However, in addition, we assume access to an offline dataset of transitions $\mathcal{D}_{\text{off}} := \{(s_t, a_t, r_t, s_{t+1})\}$ collected by an unknown behavior policy π_β such that $s_t \in \mathcal{S}$, $a_t \sim \pi_\beta(\cdot|s_t)$, $r_t = r(s_t, a_t)$, $s_{t+1} \sim p(\cdot|s_t, a_t)$. From $\mathcal{D}_{\text{off}}$, we can infer a policy π_{off}, its value function Q_{off}, or both. These components can be used in several ways to assist learning π_{on} and Q_{on} with few interactions in $\mathcal{M}$, as shown in Sec. IV. Our goal is to minimize the number of online interactions required to learn an optimal π_{on} by leveraging $\mathcal{D}_{\text{off}}$, π_{off} and Q_{off}.

III. Preliminaries

In this section, we introduce algorithms for training a policy and value function online with interactions in $\mathcal{M}$ as well as learning them offline using the dataset $\mathcal{D}_{\text{off}}$.

A. Online training

We consider actor-critic algorithms SAC [17] and TD3 [18] for online training due to their ability to learn from off-policy transition data, and their success in sample-efficient RL [9], [19]. The training objectives follow policy iteration [1] by alternating between optimizing the critic (policy evaluation) and actor (policy improvement) as follows:

$$L(\theta) = \mathbb{E}_{(s_t, a_t, \ldots, s_{t+h}) \sim \mathcal{D}_{\text{on}}} \left[\left(Q_{\text{on}}(s_t, a_t; \theta) - y_t \right)^2 \right] \quad (1)$$

$$L(\phi) = -\mathbb{E}_{s \sim \mathcal{D}_{\text{on}}, a \sim \pi_{\text{on}}(\cdot|s;\phi)} [Q_{\text{on}}(s, a)], \quad (2)$$

where $\mathcal{D}_{\text{on}}$ is a replay buffer which stores transitions (s_t, a_t, r_t, s_{t+1}) from online interactions with $\pi_{\text{on}}(\cdot|s;\phi)$ in $\mathcal{M}$, and y_t is the target Q-value which computes the h-step Bellman backup $\mathcal{B}^{\pi_{\text{on}}} Q_{\text{on}}$ as follows:

$$y_t := \sum_{i=0}^{h-1} \gamma^i r_{t+i} + \gamma^h \mathbb{E}_{a' \sim \pi_{\text{on}}(\cdot|s_{t+h};\phi)} Q_{\text{on}}(s_{t+h}, a'; \bar{\theta}) \quad (3)$$

where $\bar{\theta}$ are target network parameters which are exponential moving averages of the Q-network parameters θ [20]. SAC and TD3 modify the actor-critic losses in (1), (2) to accommodate entropy and deterministic policies, respectively. Both use Clipped Double Q-learning (CDQ) [18] to overcome overestimation bias [21], a situation where states with arbitrarily bad value are estimated as high-value by Q-learning.

To use samples collected online efficiently, it is common to make several gradient updates per interaction, which is referred to as operating in a high update-to-data (UTD) regime. To achieve this, we use the critic ensemble modification of CDQ from REDQ [19] which calculates target Q-values as:

$$Q_{\text{on}}(s', a'; \bar{\theta}) = \min_{i \in \mathcal{Z}} Q_{\text{on},i}(s', a'; \bar{\theta}), \quad (4)$$

where $\mathcal{Z}$ is a randomly sampled subset of 2 indices from the critic ensemble.

B. Offline training

Extracting a policy π_{off} from the offline dataset $\mathcal{D}_{\text{off}}$ is a key objective of offline training, and is typically achieved through imitation learning [22], [23]. A simple approach is Behavior Cloning (BC) [24], which directly supervises π_{off} with state-action pairs from the dataset. For a stochastic policy, BC minimizes the negative log likelihood of the data, while for a deterministic policy, the objective is to minimize the mean squared error:

$$L(\phi) = \mathbb{E}_{(s,a) \sim \mathcal{D}_{\text{off}}} [-\log \pi_{\text{off}}(a|s;\phi)] \quad \text{(stochastic)} \quad (5)$$

$$L(\phi) = \mathbb{E}_{(s,a) \sim \mathcal{D}_{\text{off}}} [(\pi_{\text{off}}(s;\phi) - a)^2] \quad \text{(deterministic)} \quad (6)$$

In addition to a policy, a value function Q_{off} may also be extracted from the dataset. This presents unique challenges compared to standard online learning. The overestimation bias of Q-learning is exacerbated in the offline setting due to the inability to correct estimation errors by observing returns for desired actions. Offline RL methods like CQL [11] and CalQL [12] counter this by learning a conservative lower bound on the true Q-function by minimizing Q-values alongside the standard Bellman error objective from (1). The CQL critic objective is

$$\begin{aligned} L(\theta) = \; & \mathbb{E}_{s \sim \mathcal{D}_{\text{off}}, a \sim \pi_{\text{off}}(\cdot|s;\phi)} [Q_{\text{off}}(s, a; \theta)] \\ & - \mathbb{E}_{(s,a) \sim \mathcal{D}_{\text{off}}} [Q_{\text{off}}(s, a; \theta)] \\ & + \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}_{\text{off}}} \left[\left(Q_{\text{off}}(s, a; \theta) - y(r, s') \right)^2 \right]. \end{aligned} \quad (7)$$

While CQL avoids Q-value overestimation, it tends to severely underestimate Q-values. This hinders online finetuning, as the Q-values first undergo a recalibration to the correct scale before effective learning proceeds. CalQL counters this issue by lower bounding the Q-value estimates $Q_{\text{off}}(s, a)$ by a Monte-Carlo estimate of the value $V^{\pi_\beta}(s)$. This can be calculated by sampling trajectories of consecutive transitions $\tau = (s_0, a_0, r_0, s_1, \cdots, s_T)$ from $\mathcal{D}_{\text{off}}$ as follows:

$$V^{\pi_\beta}(s) \approx \mathbb{E}_{\tau \sim \mathcal{D}_{\text{off}}} [G_t | s_t = s] \quad (8)$$

$$G_t = \sum_{i=0}^{T-t} \gamma^i r_{t+i}, \quad (9)$$

where T is the task horizon. In practice, since we focus on continuous control problems, no two states in $\mathcal{D}_{\text{off}}$ are identical. This leads to a single trajectory return contributing to the Monte-Carlo value estimate. CalQL uses this estimate to replace the first term from (7) with

$$\mathbb{E}_{s \sim \mathcal{D}_{\text{off}}, a \sim \pi_{\text{off}}(\cdot|s;\phi)} \left[\max(Q_{\text{off}}(s, a; \theta), V^{\pi_\beta}(s)) \right]. \quad (10)$$

CQL and CalQL both learn π_{off} with the SAC variant of (2).

IV. Rainbow DemoRL

We identify three broad approaches for leveraging $\mathcal{D}_{\text{off}}$, Q_{off}, and π_{off} to improve the sample efficiency of learning Q_{on} and π_{on} online. Strategy A uses the offline data $\mathcal{D}_{\text{off}}$ directly for online training. Strategy B first learns Q_{off} and π_{off} and uses them as initializations for Q_{on} and π_{on}. Strategy C uses π_{off} as a control prior to provide reference actions for the online policy. Fig. 1 provides an overview of the components involved in each of the strategies. Below, we discuss implementing each strategy along with considerations for hybrid combinations.

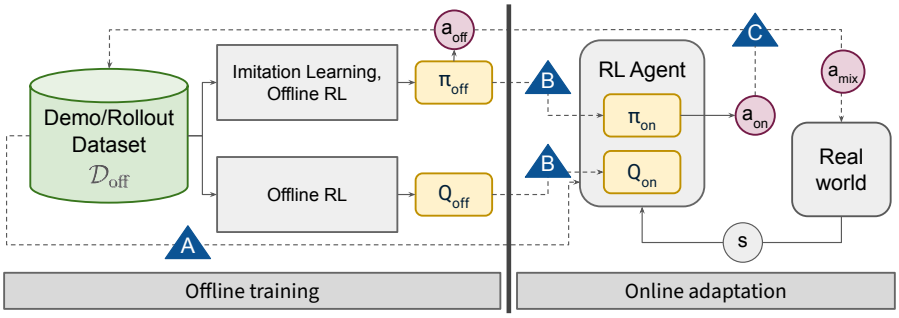


Fig. 1: Three approaches for combining offline components with online RL. Strategy A samples data from $\mathcal{D}_{\text{off}}$ along with the online RL buffer. Strategy B uses pretrained Q_{off} and π_{off} and finetunes them with online experience. Strategy C uses actions from an offline policy as reference to generate a mixed action a_{mix}.

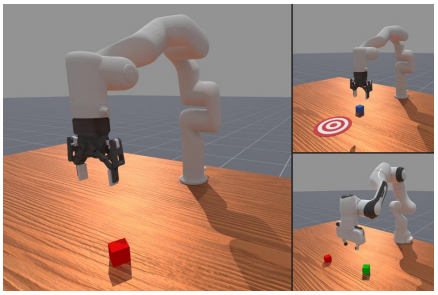


Fig. 2: Tabletop manipulation tasks with Panda and xArm6 robots in the ManiSkill simulator [25].

A. Strategy A: Using $\mathcal{D}_{off}$ directly

Off-policy actor-critic methods sample transitions from a replay buffer to update the policy and value parameters via (1) and (2). A straightforward approach to incorporate an offline dataset is to sample some transitions from $\mathcal{D}_{\text{off}}$ along with the online buffer $\mathcal{D}_{\text{on}}$. We follow the recommendation from RLPD [9] to use 50/50 sampling from the offline dataset and online buffer, along with high UTD and a critic ensemble. Methods combining these elements are referred to as "prefill" methods in our experiments.

Instead of using transitions from $\mathcal{D}_{\text{off}}$ in the RL objectives, an alternative approach is to use the offline data to optimize an auxiliary BC loss [10]. This method adds a supervised term to the RL actor loss from (2), which is the BC loss from (5) or (6) for online SAC and TD3 respectively.

B. Strategy B: Finetuning pretrained Q_{off}, π_{off}

Offline RL methods learn Q_{off} and π_{off} as discussed in Sec. III-B. We can use these as initializations for π_{on} and Q_{on}, respectively. We follow prior work on offline RL finetuning [26] which finds that dropping conservative terms and using pure online RL objectives during finetuning leads to better asymptotic performance.

We also use a simple baseline, Monte-Carlo-Q (MCQ), which learns an offline Q-function as follows:

$$L(\theta) = \mathbb{E}_{\tau\sim\mathcal{D}_{\text{off}},t\in\{0,1,\ldots,T\}}\left[(Q_{\text{off}}(s_t,a_t;\theta) - G_t)^2\right],$$

where G_t is the Monte-Carlo return as defined in (9). To ensure self-consistency of the learnt Q-function, with a small probability ϵ_b, we use the TD error from (1) to optimize the critic. In our experiments, whenever we initialize Q_{on} using MCQ, we also initialize π_{on} with a BC policy.

C. Strategy C: Using π_{off} for reference actions

Given a state s, we can generate both $a_{\text{off}} \sim \pi_{\text{off}}(\cdot|s)$ and $a_{\text{on}} \sim \pi_{\text{on}}(\cdot|s)$, and "mix" these actions in a meaningful way to help online learning. We consider the following three ideas for action mixing from IBRL [13], CHEQ [14] and Residual RL [15], [27].

1) Higher Q-value [13]: An action that maximizes the Q-values is chosen:

$$a_{\text{mix}} = \arg\max_{a\in\{a_{\text{off}},a_{\text{on}}\}} Q_{\text{on}}(s,a)$$

and used for interaction with the environment as well as to compute the Q-target in (3).

Recent work [28] finds that offline RL methods learn a good value function Q_{off} but struggle to extract a good policy. Hence, a natural hybrid method with this action mixing variant is to leverage a pretrained Q_{off} to initialize Q_{on}. This can enable effective action selection from the beginning of online training, without having to wait to first learn a good Q-function.

2) Linear interpolation [14]: Here a_{mix} is a linear interpolation of a_{off} and a_{on}, with the online action given lower weight $\lambda \in [0,1]$ if the online critic is uncertain about its value estimate:

$$a_{\text{mix}} = (1-\lambda)a_{\text{off}} + \lambda a_{\text{on}}.$$

The uncertainty is measured by the variance of the Q-value predictions across the Q-ensemble:

$$\lambda \propto \frac{1}{Var_i[Q_{\text{on},i}(s,a_{\text{on}})]}.$$

This approach also requires the mixing weight to be observable by RL to allow it to adapt its actions correctly. Hence, we use a modified state space $\mathcal{S}' \subseteq \mathbb{R}^{n+1}$ with state vectors $s' = [s^\top, \lambda]^\top$ for all CHEQ-based methods.

3) Residual RL [15]: Residual Policy Learning (RPL) considers learning a_{on} as a residual corrective action on top of a_{off}; that is, $a_{\text{mix}} = a_{\text{off}} + a_{\text{on}}$. Consistent with recent work [29], [30], the residual actor is conditioned on the offline action, and its final layer is zero-initialized to ensure $a_{\text{mix}} \approx a_{\text{off}}$ during initial interaction. The online learning objectives remain identical to the standard setting in Sec. III-A with a_{on} simply replaced by a_{mix}.

V. Experiments

We evaluate all combinations of strategies described in Sec. IV against a Soft Actor-Critic (SAC) [17] baseline

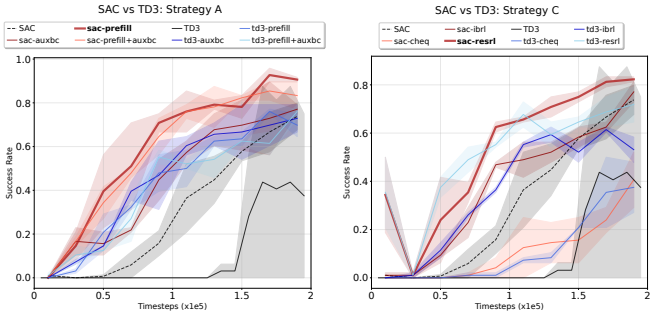


Fig. 3: Comparison of SAC vs TD3 success rates for Strategy A and C approaches.

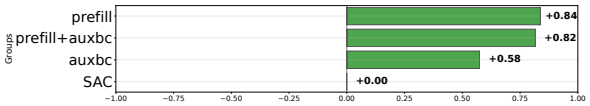
across five scenarios[1]. These scenarios consist of the PickCube, PushCube, and StackCube tasks from ManiSkill [25] on uFactory xArm6 and Franka Emika Panda robots (we exclude the StackCube-xArm6 configuration, where the SAC baseline failed to learn within a budget of 1M steps). The setup for these tasks is visualized in Fig. 2. The state s is composed of robot proprioception and distances between the end effector and relevant objects, while the actions a are joint velocities. We use dense rewards from Maniskill for each task. Our demonstrations consist of 5,000–10,000 trajectories per task, collected from the replay buffer of an RL expert trained for over 1M steps. To ensure high-quality data, we filter for episodes that achieve at least 90% of the peak return observed during the expert's training.

To evaluate both learning speed and asymptotic performance, we compute the area-under-curve (AUC) of the success rate with respect to online interaction steps. Our evaluation proceeds in three stages. First, for each environment and robot configuration, we calculate the percentage improvement of an algorithm x over the SAC baseline as $(AUC_x - AUC_{\text{SAC}})/AUC_{\text{SAC}}$, averaged across three random seeds. Second, we apply a sign-preserving normalization where positive and negative improvements are scaled to $[0, 1]$ and $[-1, 0]$ respectively, anchored by the best and worst performing algorithms in each setting. Finally, we average these normalized values across all experiment settings to produce the Sample Efficiency Improvement (SEI) score for algorithm x. This metric provides a balanced comparison of algorithmic performance that is robust to task-specific variance and prevents outliers in easier tasks from skewing the aggregate results.

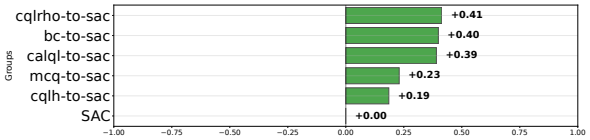
A. Base RL algorithm

We compare SAC [17] and TD3 [18] variants of single strategy methods to fix the base online RL algorithm. Methods employing strategy B naturally work better with SAC since conservative offline RL algorithms like CQL and CalQL learn a stochastic policy. We also observe that for strategies A and C, SAC variants have lower variance across random seeds and often outperform their TD3 counterparts,

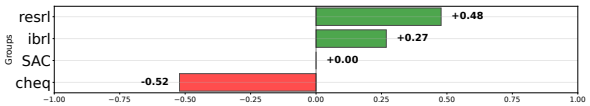
[1]A full set of results is available on our public wandb dashboard at `https://wandb.ai/ucsd_erl/rainbow-demorl-final`



(a) Comparison of SEI scores for strategy A approaches using $\mathcal{D}_{\text{off}}$ with online RL (SAC).



(b) Comparison of SEI scores for strategy B approaches finetuning pretrained policies and value functions.



(c) Comparison of SEI scores for strategy C approaches using π_{off} for reference actions.

Fig. 4: Comparing Sample Efficiency Improvement (SEI) scores for single strategy approaches.

as seen in the comparison of these algorithms for PickCube in Fig. 3. Hence, in the remainder of the experiments, we use SAC as the base RL algorithm.

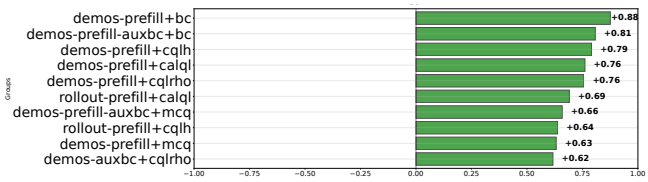
B. Single strategy

1) Only A: Fig. 4a compares the performance of strategy A variants with BC auxiliary loss [10] and prefilling the replay buffer with offline data [9]. While all variants here are more efficient than the baseline, the fastest approach is the one that uses buffer prefilling like RLPD [9], with the approach combining prefill and auxiliary BC loss following closely behind.

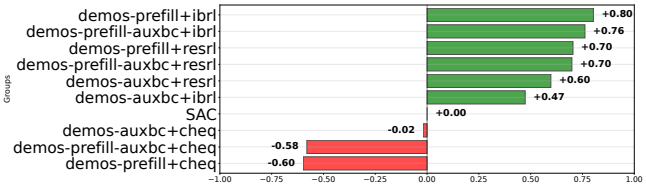
2) Only B: Fig. 4b compares the performance of strategy B variants that finetune pretrained value and policy models. We first consider the case of pretraining an offline BC policy and starting with an untrained critic. Next, we consider jointly pretrained actors and critics with offline RL methods CQL and CalQL, and finally with MCQ described in Sec. IV-B. For CQL, there are two ways of implementing the first term from (7), corresponding to CQL-H and CQL-ρ [11].

While all finetuning strategies improve online sample efficiency over SAC, pretraining with CQL-ρ emerges as the fastest algorithm in this category, with simple BC policy pretraining with an untrained critic as a close runner-up. Despite the gains from finetuning, no strategy B approach learns as fast as Strategy A methods.

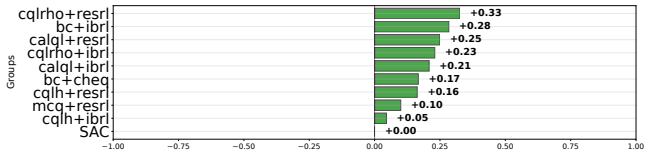
3) Only C: Fig. 4c evaluates the performance of strategy C variants, which compare the Q-values of the online and offline actions (IBRL [13]), use linear interpolation between a_{off} and a_{on} (CHEQ [14]), or learn a_{on} as a residual on top of a_{off} (RPL [15]). All methods except CHEQ lead to improvements in online sample efficiency. RPL achieves the fastest learning and typically also has the best initial performance since it starts with a zero-initialized online policy leading to



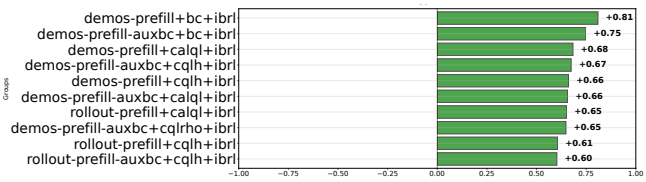
(a) Comparison of SEI scores for the top ten hybrid variants combining strategies A and B.



(b) Comparison of SEI scores for the top ten hybrid variants combining strategies A and C.



(c) Comparison of SEI scores for the top ten hybrid variants combining strategies B and C.



(d) Comparison of SEI scores for the top ten hybrid variants combining strategies A, B and C.

Fig. 5: Comparison of Sample Efficiency Improvement (SEI) scores for the top ten hybrid variants across different strategy combinations.

$a_{\text{mix}} \approx a_{\text{off}}$. Overall, strategy C methods have higher variance in performance, and their sample efficiency gains are highly dependent on the specific action mixing variant.

C. Hybrid strategies

While single-strategy methods provide significant gains, we test if integrating multiple approaches can yield further improvements in sample efficiency. Certain hybrid methods benefit from specific modifications to the individual strategies to maximize the performance. For A+B hybrids that both prefill the replay buffer and use a pretrained policy π_{off}, a key consideration is the data source used for pre-filling the buffer. WSRL [26] finds that prefilling with rollouts from π_{off} instead of the original offline dataset improves sample efficiency and final agent performance.

For B+C hybrids that use pretrained value and policy along with CHEQ for action mixing, we need to account for the updated state space $\mathcal{S}' \subseteq \mathbb{R}^{n+1}$, which necessitates larger input layers for the Q and policy networks than the ones from Q_{off} and π_{off}. We handle these differences by partially loading the pretrained weights onto Q_{on} and π_{on} for the first n neurons in the input layer, and randomly initializing weights for the new input neuron corresponding to λ.

The relative effectiveness of hybrid approaches is summarized in the following comparisons from Fig 5, which highlights the top ten variants identified for each strategy combination. A+B hybrids learn significantly faster than the baseline. However, the performance of the best A+B hybrids is similar. The combination with demonstration prefill and simple BC initialization achieves better performance than conservative offline RL methods, as seen in Fig. 5a. A+C hybrids are compared in Fig. 5b, which shows that combinations using IBRL and residual RL learn significantly faster than those using CHEQ, which at best matches the baseline performance. The fastest approach here is IBRL with a replay buffer prefilled with demonstration data. Next,

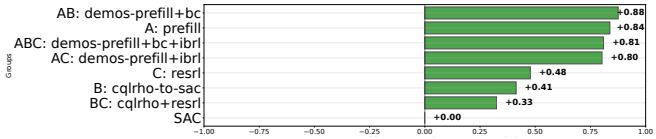


Fig. 6: Comparison of Sample Efficiency Improvement scores for the best performing variant of each strategy combination.

we see from Fig. 5c that B+C hybrids achieve only moderate sample-efficiency gains over the baseline. This indicates that strategy A of using samples from the offline dataset directly is critical to achieving sample-efficient online RL. Finally, methods combining all three strategies also improve on the baseline with the prefill and IBRL components commonly showing up among the best performers as seen in Fig. 5d.

We perform a comparison across the best methods for each strategy combination in Fig. 6. We note that the hybrid methods employing strategy A outperform the rest. Finally, an extensive comparison of all methods considered in the paper is presented in Fig. 9. We find that the A+B approach that uses a replay buffer prefilled with demonstration data along with a simple pretrained BC policy improves the sample efficiency of online RL the most. Other hybrids using buffer prefill and an auxiliary BC loss follow closely behind in performance.

D. Per-component impact

To isolate the marginal contribution of individual algorithmic choices in hybrid algorithms, we conduct a paired ablation analysis. For a given environment and robot configuration, we identify all algorithm pairs that differ solely by the presence of a specific component x. The relative performance gain yielded by x is calculated as $AUC_{\text{with } x} - AUC_{\text{without } x}$. We use the difference rather than a ratio to ensure the impact score remains robust across diverse baseline performance levels ($AUC_{\text{without } x}$) and to avoid over-weighting improve-

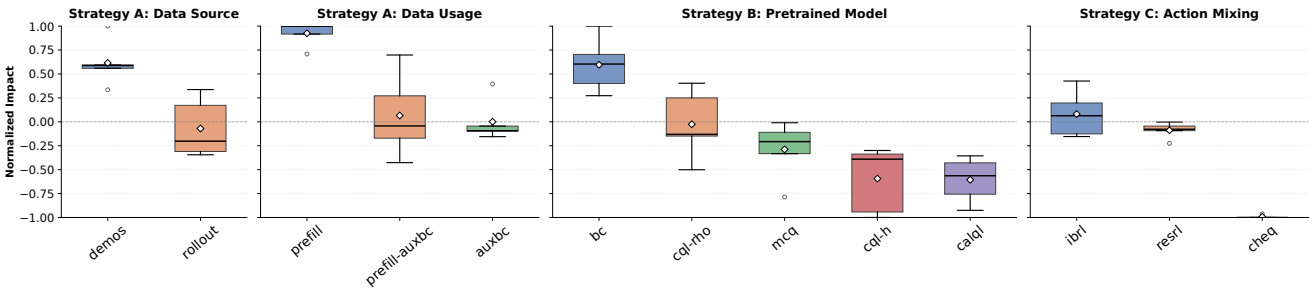


Fig. 7: Distribution of normalized impact scores for individual algorithmic components across all task and robot configurations. Scores are independently normalized to $[-1, 1]$ per configuration. Positive values indicate that adding the component improved sample efficiency compared to matched baselines, while negative values indicate a degradation. The spread of the whiskers illustrates each component's sensitivity to different task difficulties.

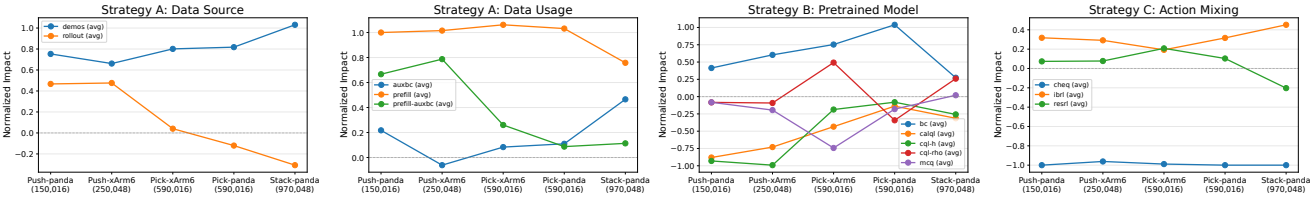


Fig. 8: Normalized impact of individual components as a function of task difficulty. Tasks are ranked from easiest to hardest, defined by the number of steps required for SAC to reach peak success (values shown in parentheses below each task). Results are grouped by strategy to highlight how the utility of each algorithmic choice shifts in more complex settings.

ments over weak ablation baselines. We average this value across all valid algorithm pairs to determine the raw impact of component x in that specific setting. To account for the inherent variance in task difficulty, these per-setting impact scores undergo the same sign-preserving normalization used for our SEI metric. Positive and negative impacts are scaled independently to $[0, 1]$ and $[-1, 0]$, anchored by the highest and lowest performing components within each setting.

To capture both the average impact and the cross-task variance of each design choice, we visualize the distribution of these normalized impact scores across all experimental settings in Fig. 7. Notably, while certain value function pretraining methods like CQL-ρ show strong standalone performance in isolation (Sec. V-B.2), simple BC policy pretraining is the only Strategy B component that consistently yields positive marginal gains across the full diversity of hybrid configurations. This suggests that the sample overhead of recalibrating a conservative offline critic often outweighs its initialization benefits when combined with other acceleration strategies. Conversely, direct data usage via buffer prefilling like RLPD proves to be a universally effective component, consistently improving sample efficiency across all environments.

Finally, we evaluate the relative impact of each component as a function of task difficulty, quantified by the number of online interaction steps required for the SAC baseline to reach its peak success rate (Fig. 8). For strategy A, we observe a strong negative correlation between task difficulty and the utility of policy rollouts as a data source. This is expected, as the behavioral quality of a pre-trained policy's rollouts naturally degrades in more complex environments. In addition, as task complexity scales, the relative impact of buffer prefilling diminishes, while that of an auxiliary BC loss increases. Among strategy B and C components, the most pronounced trends emerge from the offline RL methods CalQL and CQL-H, both of which exhibit an increasingly positive impact on sample efficiency as tasks become more challenging.

VI. Conclusion

We presented Rainbow-DemoRL, a systematic categorization and large empirical evaluation of demonstration-augmented online reinforcement learning. We organized existing methods into three core strategies of direct data usage, model fine-tuning, and reference action mixing, and evaluated both their isolated and combined impacts on online sample efficiency. Our ablation study led to the following practical recommendation for robotic manipulation using reinforcement learning: complex offline RL value pretraining often slows down online adaptation due to the sample overhead required for critic recalibration. Instead, the most performant and sample-efficient approaches employ replay buffer prefilling [9] with a simple behavior cloning policy initialization and an untrained critic. We hope the comprehensive analysis presented in this work serves as a valuable reference for researchers seeking to deploy manipulation policies that adapt rapidly during online interaction with the world.

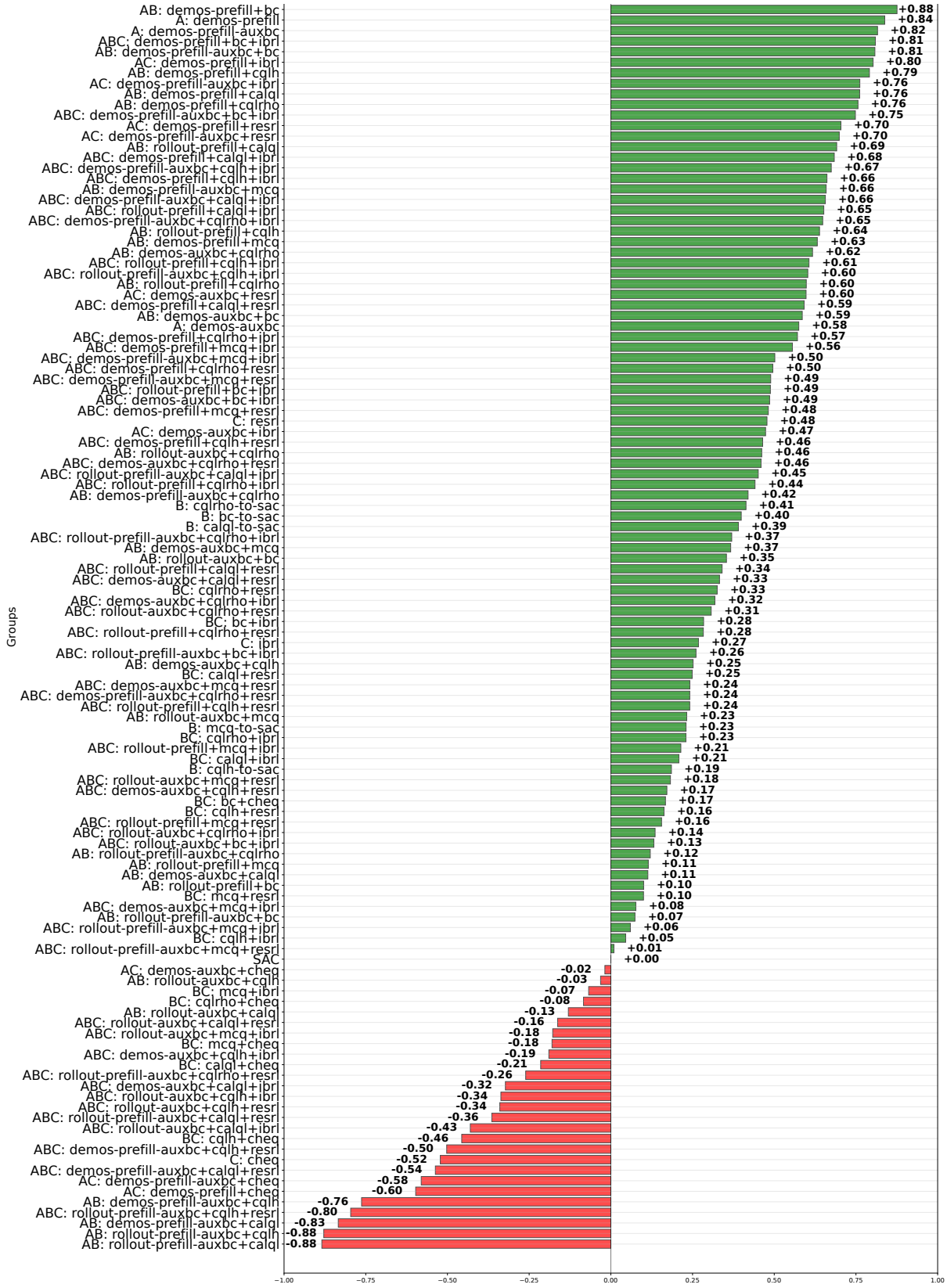


Fig. 9: Comparison of Sample Efficiency Improvement (SEI) scores for all methods, aggregated across the four evaluated task-robot configurations. Higher SEI scores indicate faster learning speed and final performance relative to the SAC baseline (SEI = 0). Component abbreviations are defined as follows: *demos/rollouts* indicates the data source; *prefill/auxbc* denotes the Strategy A direct data usage method; *bc/mcq/calql/cqlrho/cqlh* specifies the Strategy B pretraining method; and *resrl/ibrl/cheq* represents the Strategy C action mixing mechanism.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2nd ed., 2018.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[4] O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. Agapiou, M. Jaderberg, and D. Silver, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, pp. 350–354, 10 2019.

[5] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, pp. 28694–28698, 2025.

[6] J. Lee, M. Bjelonic, A. Reske, L. Wellhausen, T. Miki, and M. Hutter, "Learning robust autonomous navigation and locomotion for wheeled-legged robots," *Science Robotics*, vol. 9, no. 89, p. eadi9641, 2024.

[7] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[8] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is enough," *Artificial Intelligence*, vol. 299, p. 103535, 2021.

[9] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, "Efficient online reinforcement learning with offline data," in *International Conference on Machine Learning (ICML)*, 2023.

[10] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 20132–20145, 2021.

[11] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1179–1191, 2020.

[12] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine, "Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 62244–62269, 2023.

[13] H. Hu, S. Mirchandani, and D. Sadigh, "Imitation bootstrapped reinforcement learning," in *Robotics: Science and Systems*, 2024.

[14] E. Cramer, B. Frauenknecht, R. Sabirov, and S. Trimpe, "Contextualized hybrid ensemble Q-learning: Learning fast with control priors," in *Reinforcement Learning Conference*, 2024.

[15] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," in *arXiv:1812.06298*, 2019.

[16] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, pp. 1861–1870, 2018.

[18] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning (ICML)*, pp. 1587–1596, 2018.

[19] X. Chen, C. Wang, Z. Zhou, and K. W. Ross, "Randomized ensembled double Q-learning: Learning fast without a model," in *International Conference on Learning Representations (ICLR)*, 2021.

[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *arXiv:1509.02971*, 2015.

[21] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[22] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A survey of imitation learning: Algorithms, recent developments, and challenges," *IEEE Transactions on Cybernetics*, vol. 54, no. 12, pp. 7173–7186, 2024.

[23] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[24] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural computation*, vol. 3, no. 1, pp. 88–97, 1991.

[25] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T.-K. Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, V. N. Rajesh, Y. W. Choi, Y.-R. Chen, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su, "Demonstrating GPU Parallelized Robot Simulation and Rendering for Generalizable Embodied AI with ManiSkill3," in *Proceedings of Robotics: Science and Systems*, (LosAngeles, CA, USA), June 2025.

[26] Z. Zhou, A. Peng, Q. Li, S. Levine, and A. Kumar, "Efficient online reinforcement learning fine-tuning need not retain offline data," in *arXiv:2412.07762*, 2025.

[27] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, 2019.

[28] S. Park, K. Frans, S. Levine, and A. Kumar, "Is value learning really the main bottleneck in offline RL?," *Advances in Neural Information Processing Systems*, vol. 37, pp. 79029–79056, 2024.

[29] L. Ankile, Z. Jiang, R. Duan, G. Shi, P. Abbeel, and A. Nagabandi, "Residual off-policy rl for finetuning behavior cloning policies," 2025.

[30] K. Li, P. Li, T. Liu, Y. Li, and S. Huang, "Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6991–7003, 2025.

APPENDIX

All experiments use the hyperparameters listed in Table I.

TABLE I: Hyperparameters

Hyperparameter	**Value**
Online RL Hyperparameters	
Horizon (h)	3
Discount factor (γ)	0.8
Target smoothing (τ)	0.01
Batch size	1024
Learning rate	3×10^{-4}
Update-to-data ratio	1 / 5 (for prefill methods)
Neural Network Architecture	
MLP hidden dimension	256
Actor hidden layers	3
Critic hidden layers	3
Number of critics	2 / 5 (for prefill methods)
Algorithm-Specific Parameters	
TD3	
Target policy noise	0.1
Noise clip	0.5
Exploration noise	0.1
SAC	
Entropy coeff. (α)	0.2
CQL	
Actor learning rate	3×10^{-5}
CQL weight	5.0
MCQ	
Bootstrap epsilon (ϵ_b)	0.1
CHEQ	
Uncertainty bounds (u)	[0.15, 0.275]
Lambda bounds (λ)	[0.2, 1.0]
Residual RL	
Critic burn-in steps	20,000