DO YOU REALLY NEED TO PRETRAIN Q-FUNCTIONS FOR ONLINE RL FINE-TUNING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Pre-training followed by fine-tuning has become the dominant recipe for learning performant policies, and in value-based reinforcement learning (RL) this raises a natural question: given a pretrained policy, should the Q-function be pretrained on offline data too? Conventional wisdom suggests it should, but recent results show that online RL with a randomly-initialized Q-function can result in highly performant and reliable policies without needing to pretrain the Q-function. In this paper, we systematically study whether pretraining the Q-function actually helps when fine-tuning on top of a pretrained base policy. We find, surprisingly, that naive Q-function pretraining often provides little benefit over random initialization. We show this stems from a fundamental mismatch: the Q-function learned during pretraining targets the pretrained policy's Q-function, not the Q-function that online fine-tuning converges to, and this gap persists even after offline value maximization. Motivated by this finding, we propose Initialization via Policy Ensemble (IPE), a simple method that trains multiple diverse policies and uses their pooled rollouts to bootstrap the Q-function learning in online RL. Across a suite of challenging continuous control benchmarks, IPE yields an average 1.26x improvement in fine-tuning performance over naive Q-function pre-training.

1 INTRODUCTION

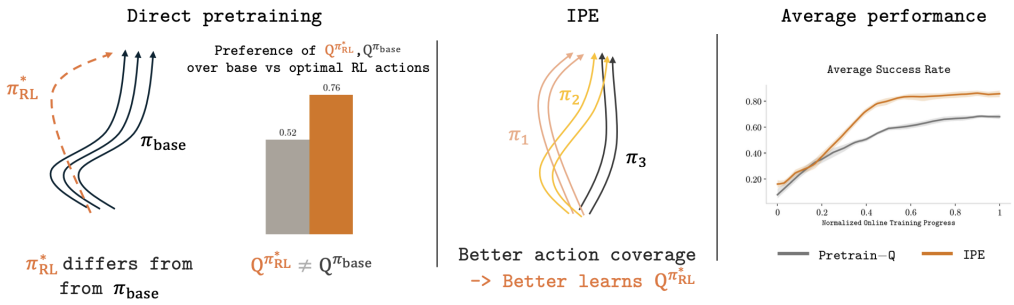

Figure 1: **IPE.** Left: naively pretraining does not help RL fine-tuning performance as $Q^{\pi_{\text{base}}} \neq Q^{\pi^*_{\text{RL}}}$. Mid: IPE enables online RL fine-tuning to better learn $Q^{\pi^*_{\text{RL}}}$ through diverse action coverage. Right: IPE results in an average of 26% improvement over directly pretraining the Q-function.

Pretraining followed by using reinforcement learning (RL) to finetune has become the predominant approach of learning reliable, performant policies in a range of settings from language modeling (Ouyang et al., 2022) to robotics (Intelligence et al., 2025; Dong et al., 2026a). Given access to strong pretrained policies, a central question is how to best finetune them with RL. In value-based RL, in addition to the policy, the Q-function, which tells the policy which actions are good in a state, is also crucial for downstream performance. Given that the policy is pretrained on offline data, intuitively it makes sense to also pretrain the Q-function on offline data. However, recent works have shown strong results by not initializing the Q-function and only starting with a pretrained policy for online training (Dong et al., 2026a). In this work, we put this intuition to question and ask: when given a pretrained policy, should you pretrain a Q-function for online RL finetuning?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Conventionally, offline-to-online RL (Vecerik et al., 2018; Nair et al., 2021; Nakamoto et al., 2024; Ball et al., 2023; Dong et al., 2025b) works have explored jointly pretraining a Q-function and a policy, followed by online RL finetuning. However, this requires the policy to be trained jointly with the Q-function using the specific offline RL algorithm's objective, as opposed to a general pretrained policy such as with supervised learning. Adapting an existing pretrained policy to fit this joint offline RL training procedure can be challenging and costly, requiring changes to the policy's training pipeline, and its effectiveness at scale remains largely untested. Instead, we focus on the setting of only having access to a pretrained policy and data, where the task is to perform online RL finetuning and obtain the highest performance. Since the Q-function directly shapes which actions the policy is pushed toward during online RL, its initialization can substantially affect both learning speed and final performance. This suggests great care needs to be taken to examine the effects of pretraining Q-functions for downstream performance.

In this work, we answer these questions through a systematic empirical analysis to understand how Q-function pretraining affects downstream performance, and to see if there is a way to make pretraining more effective. We start by analyzing what happens when you pretrain a Q-function for online RL fine-tuning and make the surprising finding that naively pretraining the Q-function before online RL often does not result in performance improvements. We posit this is because the Q-function learned from pretraining, $Q^{\pi_{\text{base}}}$, is fundamentally different from the optimal Q-function RL fine-tuning learns, $Q^{\pi^*_{\text{RL}}}$, as evidenced by their different preferences for actions, and this results in a shift in the value landscape from pretraining to finetuning. Furthermore, we show that this difference requires online adaptation to overcome and cannot be mitigated through offline value maximization of the pretrained policy. Based on these observations, we propose a simple method to boost the performance of online RL finetuning, which we call Initialization via Policy Ensemble (IPE). IPE trains multiple different policies on the same action distribution of the pretrained policy and collects rollouts with those policies to allow the Q-function to distinguish actions that the policy generates, while increasing the diversity of the data around the policy data distribution.

Our main contribution is an extensive analysis of Q-function pretraining for finetuning policies with online RL. Contrary to conventional beliefs of pretraining as much as possible, we find that directly pretraining the Q-function when starting from a pretrained policy often does not help performance of RL finetuning. On top of that, we identify the reason where Q-function pretraining is not helpful – the offline Q-function is different than the Q-function learned by online RL. Building on these insights, we propose IPE for learning better Q-functions for online RL finetuning. Evaluating on complex continuous control task benchmarks, IPE results in an average of 26% improvement over naively pretraining the Q-function for online RL finetuning.

2 Related Work

Offline-to-Online RL. Offline-to-online RL studies the problem of learning a policy online with interactions after training it offline. Popular approaches for offline-to-online RL involve balancing exploration (Yang et al., 2023; Hu et al., 2023; Zhang et al., 2023; Mark et al., 2023), calibrating values (Nakamoto et al., 2024), or maintaining offline data for online fine-tuning (Vecerik et al., 2018; Nair et al., 2021; Ball et al., 2023; Dong et al., 2025a;b), with the goal of improving policy performance after offline RL. Offline RL methods have also been directly applied to online fine-tuning (Fujimoto et al., 2019; Fujimoto & Gu, 2021; Hansen-Estruch et al., 2023; Park et al., 2025b; Dong et al., 2026b;c). Our work differs from this line of research in that we focus on the problem of fine-tuning with online RL on top of a pretrained policy, such as a Vision-Language-Action policy trained on a diverse offline dataset using supervised learning (Team et al., 2025; Intelligence et al.), rather than performing the entire offline RL procedure which can be challenging.

RL with Pretrained Policies. A separate line of work leverages pretrained policies as a starting point for downstream reinforcement learning, rather than relying on performing offline RL, such as jump-starting RL from a suboptimal or pretrained policy (Uchendu et al., 2023), RL without retaining offline data (Zhou et al., 2025), residual policy learning that trains a corrective policy on top of a fixed pretrained one (Johannink et al., 2018; Silver et al., 2019), and training BC policies for more efficient RL fine-tuning (Wagenmaker et al., 2025). Other works use on-policy methods to fine-tune pretrained policies policy (Schulman et al., 2017; Schaal, 1996; Rajeswaran et al., 2018; Ren et al., 2024). We focus on the off-policy case for increased sample-efficiency. More recently, this paradigm has been

applied to fine-tuning large-scale VLA policies with online RL, where the VLA itself serves as the pretrained policy that is improved through online interaction with RL (Chen et al., 2025; Dong et al., 2026a). Our work builds on this direction, and specifically investigates how pretraining Q-functions affect the performance of RL fine-tuning.

3 Problem Formulation

We consider a Markov decision process (MDP) specified by the tuple $\{\mathcal{S}, \mathcal{A}, \rho, r, \gamma, T\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $\rho(s)$ the distribution over initial states, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, $\gamma \in [0, 1)$ the discount factor, and $T(s'|s, a)$ the state-transition probability. The goal of RL is to find a policy π that maximizes the expected discounted return $\mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$.

In this paper, we study reinforcement learning finetuning on top of a pretrained base policy π_{base}. This could, for example, be a pretrained Vision-Language-Action (VLA) model trained with supervised learning on offline data, though the experiments in this paper focus on smaller scale policies (also trained with supervised imitation) to enable extensive analysis. Concretely, we assume access to a fixed offline dataset $\mathcal{D}_{\text{offline}} = \{(s, a, r, s')\}$ of previously collected expert transitions, which is used to pretrain π_{base} and, optionally, a Q-function Q_ϕ, which is the focus of study in this work. During online finetuning, the agent collects tuples (s_t, a_t, r_t, s_{t+1}) through environment interaction; these are appended to a replay buffer $\mathcal{D}$ (initialized from $\mathcal{D}_{\text{offline}}$) and used to continue updating both the policy and the Q-function toward higher returns. Our central question is how the pretraining of Q_ϕ affects the sample efficiency and final performance of the resulting online RL fine-tuning.

We focus on off-policy RL, where the Q-function estimates the discounted return of the policy given a state and action, and is trained with TD learning:

$$\mathcal{L}(\phi) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left[\left(r_t + \gamma Q_{\phi'}(s_{t+1}, \tilde{a}^*_{t+1}) - Q_\phi(s_t, a_t) \right)^2 \right], \tag{1}$$

where $Q_{\phi'}$ is a target network and $\tilde{a}^*_{t+1}$ is the next action selected by the RL policy.

EXPO. We use EXPO (Dong et al., 2025b), a recently proposed, sample-efficient and stable RL algorithm, which has also been shown to be effective for finetuning pretrained policies such as VLA models (Dong et al., 2026a), as a general off-policy RL algorithm to analyze Q-function pretraining. EXPO maintains two parameterized policies: a base flow policy—in our case the base policy π_{base}, trained with a supervised loss—and a lightweight edit policy π_{edit} trained to maximize the Q-function:

$$\mathcal{L}(\pi_{\text{edit}}) = -\mathbb{E}_{(s_t, a_t) \sim \mathcal{D},\ \hat{a}_t \sim \pi_{\text{edit}}} \left[Q_\phi(s_t,\ a_t + \hat{a}_t) - \alpha \log \pi_{\text{edit}}(\hat{a}_t \mid s_t, a_t) \right]. \tag{2}$$

The edit policy predicts an edit $\hat{a}_t$, constrained to $[-\beta, \beta]$, that is added to the base action a_t to produce an edited action $\tilde{a}_t = a_t + \hat{a}_t$. This avoids backpropagating gradients through the base policy while still grounding TD updates in near-optimal actions. The final inference and TD-backup policy is an on-the-fly (OTF) policy that selects the value-maximizing candidate across N sampled base and edited actions:

$$\tilde{a}^* = \underset{a \in \bigcup_{i=1}^{N} \{a_i, \tilde{a}_i\}}{\arg\max} \; Q_\phi(s, a), \tag{3}$$

with Q_ϕ trained via the TD objective above.

4 Understanding the Case of Q-Function Pretraining

The predominant paradigm in machine learning pretrains a model on offline data before fine-tuning it with RL, and pretraining is typically expected to help. In online RL finetuning, this intuition would suggest that pretraining the Q-function on offline data, in addition to pretraining the policy, should give online fine-tuning a head start. In this section, we test this intuition directly. We compare online RL performance with and without Q-function pretraining, both initialized from an identical pretrained policy, and then dig into why Q-function pretraining fails to help (Section 4.2) even under several natural attempts to fix it.

Setup. We evaluate on six challenging manipulation tasks drawn from Robomimic and OGBench. The Robomimic tasks (`lift`, `can`, `square`) involve controlling a 7-DoF robot arm on tasks such

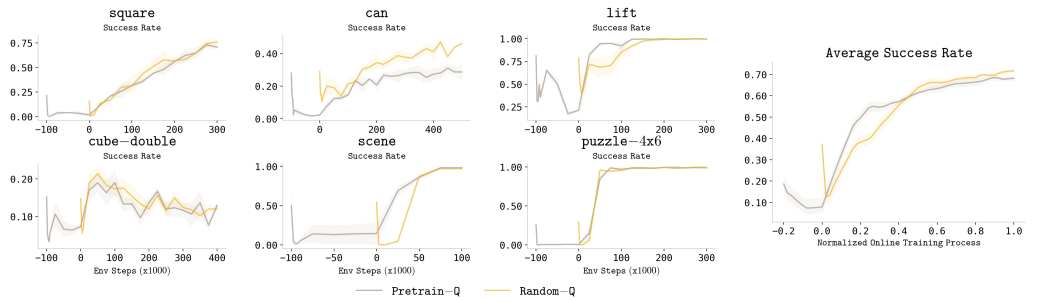

Figure 2: **Pretraining the Q-function does not improve performance for downstream fine-tuning.** Success rate over steps of online fine-tuning for a pretrained Q-function vs a randomly initialized Q-function, both sharing the same pretrained policy.

as picking up a block, picking up and placing a can, and inserting a tool onto a square peg, and provide a sparse reward that only indicates task completion. The OGBench tasks provide a reward between $-n_{\text{task}}$ and 0 depending on how many subtasks are complete: `cube` requires pick-and-place of colored blocks, `scene` requires long-horizon reasoning over interactions with multiple objects, and `puzzle` requires solving combinatorial "Lights Out" puzzles with the robot arm, testing generalization beyond the training distribution.

Our experiments use update-to-data ratio 1 to focus on how Q-function pretraining affects downstream performance. We refer additional results on higher UTD and more algorithms to Appendix Section A. We report additional environment details in Appendix Section B.3 and hyperparameters in Appendix Section B.2.

4.1 Does Pretraining the Q-Function Help Fine-Tuning?

We first ask the most direct question: starting from the same pretrained policy π_{base}, does online fine-tuning improve if the Q-function is also pretrained via offline TD learning with $\mathcal{D}_{\text{offline}}$, compared to initializing the Q-function from scratch and learning it entirely online? We pretrain the policy using supervised learning as a stable loss for pretraining. Figure 2 compares these two pipelines across all six tasks.

Contrary to the intuition that pretraining should help, we find that pretraining the Q-function provides little to no improvement in downstream fine-tuning performance, and in several tasks it actually hurts online learning relative to fine-tuning on top of a randomly-initialized Q-function. Even in the cases that show a modest acceleration early in online training, this advantage disappears as fine-tuning progresses, with both pipelines converging to similar performance. This is surprising: instead of reducing the burden of online TD learning, pretraining the Q-function did not improve performance. We investigate why this happens below.

> Takeaway 1: Naive Q-function pretraining does not accelerate fine-tuning. Despite pretraining on offline data, the Q-function performs no better — and sometimes worse — than a randomly-initialized Q-function once online fine-tuning begins.

4.2 Why is Q-Function Pretraining Not Effective?

To understand why pretraining the Q-function fails to translate into a fine-tuning advantage, we examine how well the pretrained Q-function actually estimates the value of good actions. We introduce a *preference accuracy* metric: the fraction of state-action comparisons in which the learned Q-function prefers an action from π^* over an action from π_{base}, where π^* is an estimate of the optimal policy (see Appendix B.1 for details). A well-calibrated Q-function should generally prefer π^*'s actions.

Why does pretraining not help? We hypothesize that the central issue is a mismatch between $Q^{\pi_{\text{base}}}$, the value function the offline data supports, and $Q^{\pi^*_{\text{RL}}}$, the value function that fine-tuning is

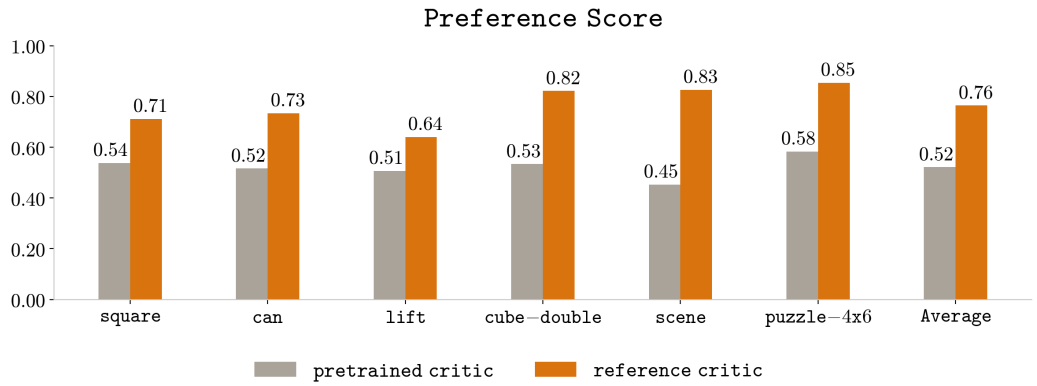

Figure 3: **Preference of $Q^{\pi_{\text{base}}}$ and $Q^{\pi^*_{\text{RL}}}$.** $Q^{\pi_{\text{base}}}$ is not the same as $Q^{\pi^*_{\text{RL}}}$ even for a near-optimal base policy, but the Q-functions have preferences close to each other.

learning. To isolate this gap from confounds due to a weak base policy, we train a base policy to near-100% success rate and pretrain its Q-function to convergence — as close to $Q^{\pi_{\text{base}}}$ for an optimal distribution of the base policy as offline pretraining can get. We emphasize that, in practice, it is rare for a base data distribution to be optimal enough to support training a policy offline to 100% performance, which is precisely why we construct this setting to characterize in the limit what the pretrained Q-function converges to. We then use the preference accuracy metric to compare how the learned $Q^{\pi_{\text{base}}}$ and $Q^{\pi^*_{\text{RL}}}$ each rank actions sampled from the base policy π_{base} against actions sampled from π^*_{RL}, where π^*_{RL} is a ground truth RL policy trained to 100% performance. We present the results in Figure 3.

Two observations emerge. First, $Q^{\pi_{\text{base}}} \neq Q^{\pi^*_{\text{RL}}}$: even with a near-optimal base policy, the offline-pretrained critic does not coincide with the critic that online fine-tuning ultimately aims to learn, so $Q^{\pi^*_{\text{RL}}}$ must be learned through online interaction regardless of the optimality of the base distribution. Second, despite this mismatch, the two value functions are not wildly different as the preference of the two Q-functions are quite close, suggesting a nontrivial fraction of optimal actions learned by RL belongs within the distribution of π_{base}.

Takeaway 2: The pretrained critic targets the wrong value function. $Q^{\pi_{\text{base}}}$ is not the same as $Q^{\pi^*_{\text{RL}}}$ — the Q-function learned through pretraining is different from the optimal Q-function learned with online RL finetuning — but the two are also not entirely unrelated, since a meaningful share of actions from the pretrained policy remain optimal after fine-tuning.

Does offline value maximization close the gap? Since $Q^{\pi_{\text{base}}} \neq Q^{\pi^*_{\text{RL}}}$, a natural fix is to perform value maximization *during* pretraining, rather than simply fitting $Q^{\pi_{\text{base}}}$, in the hope of pushing the pretrained critic closer to $Q^{\pi^*_{\text{RL}}}$ before online fine-tuning even starts. We test three such approaches: (1) learning the edit policy from EXPO, (2) distilling a separate one-step policy that maximizes Q-value on top of the pretrained policy (max Q + BC loss) similar to Park et al. (2025b), and (3) applying action gradients directly to the pretrained policy alongside value maximization. We see in Figure 4 and Figure 5 across all three, neither final performance nor preference accuracy improves over naive Q-pretraining, indicating that offline value maximization alone is not sufficient to bridge the gap identified above and online data is needed.

Takeaway 3: Fixing pretraining post hoc through offline value maximization is insufficient. Offline value-maximization techniques do not close the gap between $Q^{\pi_{\text{base}}}$ and $Q^{\pi^*_{\text{RL}}}$; more data helps incrementally but does not resolve the underlying mismatch.

Taken together, these results indicate that naively pretraining the Q-function on offline data is not an effective way to obtain better performance for fine-tuning: pretraining converges to the wrong target and offline value maximization during pretraining is not enough to fix this. Instead, the Q-function

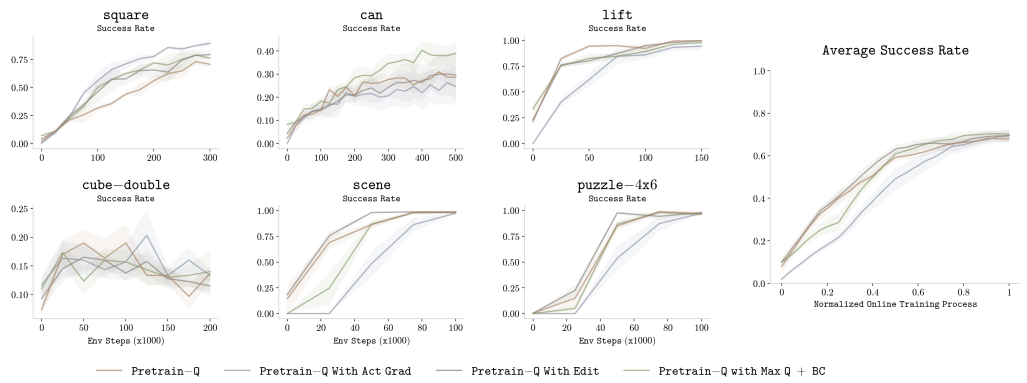

Figure 4: **Performance of online RL after offline value maximization.** Performing offline value maximization does not improve the performance over directly pretraining, suggesting online data and learning is necessary.

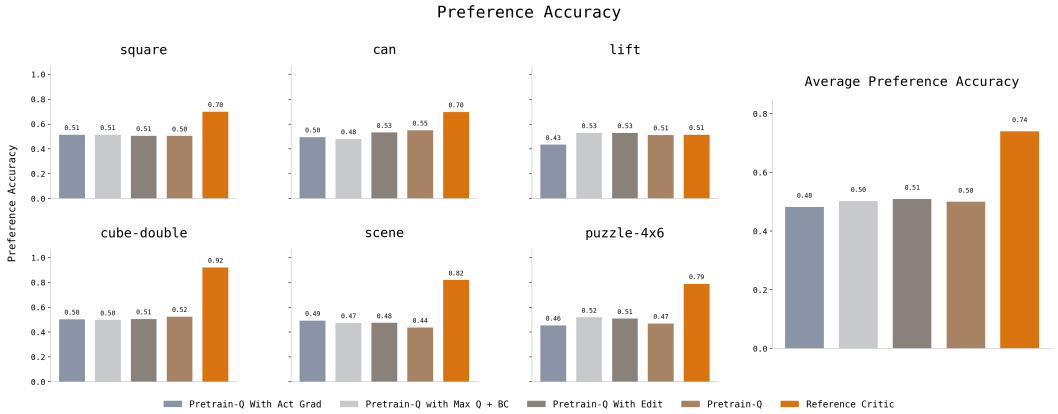

Figure 5: **Preference score analysis of offline value maximization.** The preference of actions after performing offline value maximization stays similar to pretraining without value maximization.

online fine-tuning learns must be acquired through online interaction. This motivates a different way of using the offline data to improve performance for online RL fine-tuning, which we introduce next.

5 IPE: Initialization via Policy Ensemble

So far, we have seen that naively pretraining the Q-function on offline data does not meaningfully improve fine-tuning performance, because pretraining converges to $Q^{\pi_{\text{base}}}$ which is different to the $Q^{\pi^*_{\text{RL}}}$ that online fine-tuning actually needs (Section 4.2). This raises the question: is there a way to make better use of the offline data during pretraining, one that enables online RL finetuning to better learn $Q^{\pi^*_{\text{RL}}}$?

Key idea. The offline dataset encodes a prior over what a near-optimal policy would predict, and Section 4.2 showed that this prior is not far from $Q^{\pi^*_{\text{RL}}}$. The failure of direct pretraining is not that the data is uninformative, but that pretraining directly on π_{base} overly constrains the Q-function to $Q^{\pi_{\text{base}}}$. This problem is compounded by a second issue: a single trained policy, even an expressive one such as a diffusion or flow-matching policy, tends to produce a narrow action distribution. When the Q-function is pretrained against actions from only one such policy, it receives almost no coverage over alternative actions at a given state, and the resulting $Q^{\pi_{\text{base}}}$ collapses toward $V^{\pi_{\text{base}}}$ rather than capturing meaningful action-value structure (Section 6.1).

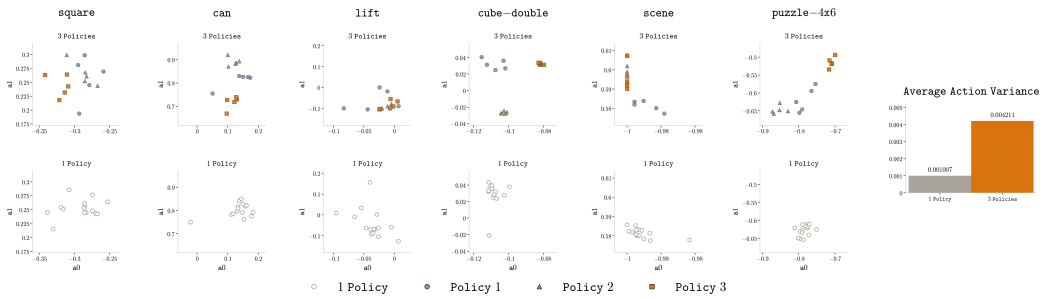

Figure 6: **Actions from a single policy vs multiple policies.** Actions sampled from one pretrained policy are significantly less diverse than actions pooled across three different policies trained on the same action distribution, the latter giving the Q-function meaningful coverage to learn from.

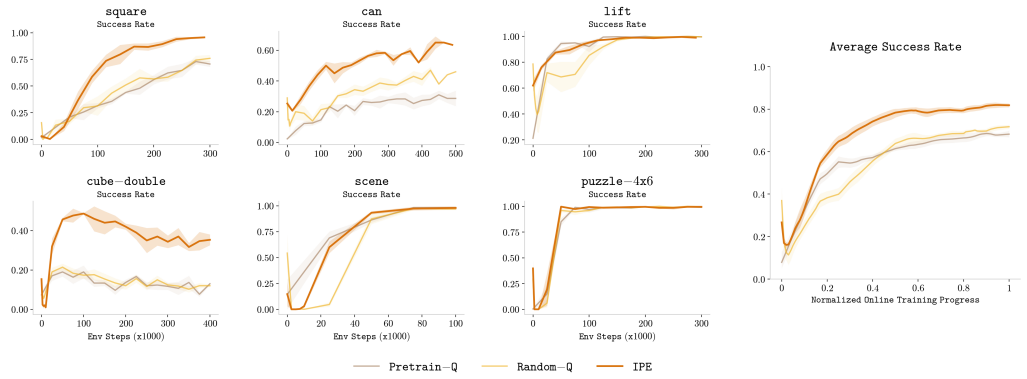

Figure 7: **IPE.** Through using multiple policies to collect diverse rollouts, IPE is able to better learn $Q^{\pi^*_{\text{RL}}}$ from RL finetuning.

Our key idea is to break this narrowness by initializing the Q-function on rollouts from a *set of different policies*, all trained on the same underlying data distribution as π_{base}. Because each policy is a different function approximator (or trained with different seeds/subsets of data), their rollouts are different from one another even though they target the same behavior distribution, giving the Q-function meaningful coverage over alternative actions at each state. This diversified coverage can allow the Q-function to not be constrained to $Q^{\pi_{\text{base}}}$ and get closer to $Q^{\pi^*_{\text{RL}}}$ during RL finetuning.

Algorithm 1 IPE

Require: Offline data $\mathcal{D}_{\text{offline}}$, base policy π_{base}

1: Train N policies $\{\pi_i\}_{i=1}^{N}$ via supervised learning on the π_{base} action distribution
2: Initialize replay buffer $\mathcal{D} \leftarrow \emptyset$
3: **for** each policy $\pi \in \{\pi_{\text{base}}, \pi_1, \ldots, \pi_N\}$ **do**
4: Roll out π in the environment to collect transitions $\{(s, a, r, s')\}$
5: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, r, s')\}$
6: **return** replay buffer $\mathcal{D}$ for online RL

IPE algorithm. IPE instantiates this idea in three steps: (1) obtain/pretrain a base policy π_{base} on the offline dataset; (2) train a set of N additional policies on the same action distribution that π_{base} was trained on — this can use the identical dataset or different subsets thereof; (3) roll out each of the $N+1$ policies in the environment and use the resulting transitions to bootstrap the replay buffer. At $N = 0$, this recovers the warm up phase for online RL. As we show in Section 6.3, larger N typically yields higher performance. We summarize IPE in Algorithm 1.

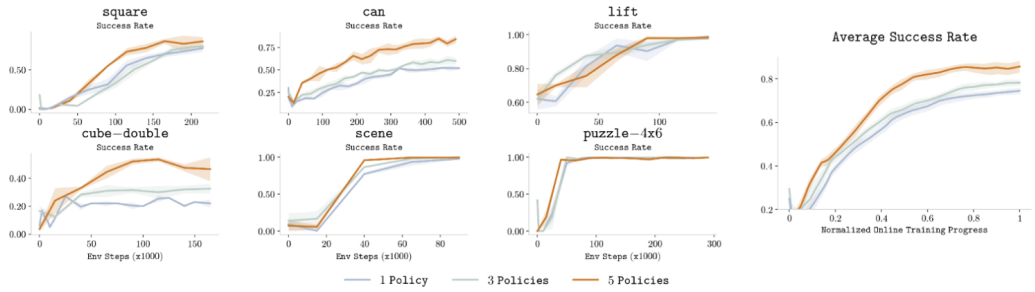

Figure 9: **Scaling the number of policies for IPE.** Increasing the number of policies used to collect rollouts results in better performance.

6 Experiments

Our experiments study how well IPE enables effective fine-tuning on top of a pretrained base policy. Concretely, we investigate the following questions: (1) Does a single pretrained policy in fact produce narrow, low-diversity action rollouts? (2) Does IPE improve over both naive Q-pretraining and no pretraining at all? and (3) How does performance scale with the number of policies N used for IPE?

6.1 Does a Single Policy Produce Narrow Action Coverage?

We first test the premise behind IPE: that a single pretrained policy yields insufficiently diverse actions for the Q-function to learn beyond $V^{\pi_{\text{base}}}$. Figure 6 visualizes the actions sampled from a single pretrained policy alongside actions sampled from three independently-trained policies on the same data. We find that actions from the single policy cluster tightly, exhibiting substantially less diversity than the pooled actions from three separate policies, which spread more broadly over the action space despite all three being trained on identical data. This confirms that a single policy's rollouts provide limited action coverage for Q-function pretraining. Further, we show the variance across actions of Q-functions pretrained on data from 1 policy vs 3 policies in Figure 8. The Q-function trained on rollouts from 1 policy exhibits very low variance indicating that it is deterministic. Pretraining the Q-function against rollouts from multiple policies, rather than a single policy, gives the critic coverage over alternative actions and helps it escape the $Q^{\pi_{\text{base}}} \approx V^{\pi_{\text{base}}}$ collapse.

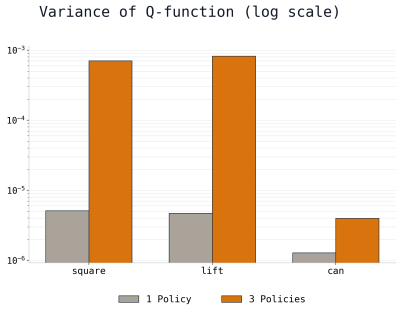

Figure 8: **Variance of Q-functions.** The Q-function trained on data from 1 policy exhibits very low variance, where the Q-function is essentially deterministic.

6.2 Does IPE Improve Over Pretraining and No Pretraining?

Next, we compare IPE against the two approaches from Section 4.1: fine-tuning with a naively-pretrained Q-function, and fine-tuning with a randomly-initialized Q-function, all of which of trained using the same finetuning procedure on the same pretrained policy, to evaluate the effectiveness of IPE in improving online RL fine-tuning performance. Figure 7 reports success rate over the course of fine-tuning across our six tasks, along with the preference accuracy metric from Section 4.2. We find that IPE outperforms both baselines in fine-tuning to higher success rates, indicating its effectiveness to enable RL fine-tuning to better learn $Q^{\pi^*_{\text{RL}}}$ and obtain higher performance.

6.3 Scaling with the Number of Policies

Finally, we study how IPE's performance scales with the number of policies N used to collect pretraining data. To do so, we vary N across a range of values while holding the total number of pretraining trajectories fixed, so that increasing N corresponds to collecting data from a more diverse

set of behavior policies rather than simply having more data. The fine-tuning performance is shown in Figure 9. We observe a general trend of improving fine-tuning performance as N increases, with $N = 5$ having the best performance in our experiments. This suggest the broader coverage provided by IPE that enables the policy prior to have a wider range of behaviors and state-action combinations than any single policy would provide on its own is a simple and effective method for improving downstream fine-tuning performance.

7 Discussion

In this work, we present a systematic study of Q-function pretraining for reinforcement learning finetuning on top of pretrained policies, and IPE, a simple method for learning more effective Q-functions for online RL finetuning. Contrary to the conventional intuition that more pretraining is always beneficial, we find that naively pretraining the Q-function on offline data *does not* translate into improved downstream finetuning performance. We trace this behavior to a fundamental mismatch between the Q-function learned during offline pretraining and the one required for online RL. We show that IPE improves significantly over naive Q-function pretraining. Despite these results, IPE has limitations. Our analysis focuses specifically on finetuning with off-policy RL. While we expect the underlying mismatch we identify to generalize, examining similar trends for on-policy methods remains an important direction for future work. Additionally, IPE's benefit comes from collecting rollouts across multiple policies, which introduces additional data-collection overhead relative to pretraining a Q-function on a single fixed policy's data. We leave the problem of reducing this overhead as future work.

AI use statement

We did not use generative AI tools for developing the core research ideas, formulating the IPE algorithm, interpreting experimental results, or making research claims. More generally, we did not use AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. We used generative AI tools to assist writing and clarity, and assist code development such as generating and refining code components and improving code readability. All AI-assisted content and code were carefully reviewed and verified by the authors. We take responsibility for the final content of this work.

Ethics statement

This work investigates reinforcement learning methods and does not involve human subjects, personally identifiable information, or private human data. The experiments are conducted with RoboMimic and OGBench, publicly available robotic manipulation benchmarks following their respective usage guidelines.

Reproducibility statement

We provide detailed descriptions of the proposed IPE algorithm and analysis in the main paper and supplementary material. We present the algorithm details in the paper, and hyperparameters, and implementation details are provided in Appendix B. We evaluate our method on RoboMimic and OGBench that are public.

References

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data, 2023. URL https://arxiv.org/abs/2302.02948.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy, 2025. URL https://arxiv.org/abs/2502.05450.

Perry Dong, Alec M. Lessing, Annie S. Chen, and Chelsea Finn. Reinforcement learning via implicit imitation guidance, 2025a. URL https://arxiv.org/abs/2506.07505.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies, 2025b. URL https://arxiv.org/abs/2507.07986.

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models, 2026a. URL https://arxiv.org/abs/2605.25477.

Perry Dong, Kuo-Han Hung, Alexander Swerdlow, Dorsa Sadigh, and Chelsea Finn. Tql: Scaling q-functions with transformers by preventing attention collapse, 2026b. URL https://arxiv.org/abs/2602.01439.

Perry Dong, Chongyi Zheng, Chelsea Finn, Dorsa Sadigh, and Benjamin Eysenbach. Value flows, 2026c. URL https://arxiv.org/abs/2510.07650.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning, 2021. URL https://arxiv.org/abs/2106.06860.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration, 2019. URL https://arxiv.org/abs/1812.02900.

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, 2023. URL https://arxiv.org/abs/2304.10573.

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. *arXiv preprint arXiv:2311.02198*, 2023.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control, 2018. URL https://arxiv.org/abs/1812.03201.

Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation, 2018. URL https://arxiv.org/abs/1811.02790.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation, 2021. URL https://arxiv.org/abs/2108.03298.

Max Sobol Mark, Archit Sharma, Fahim Tajwar, Rafael Rafailov, Sergey Levine, and Chelsea Finn. Offline retraining for online rl: Decoupled policy learning to mitigate exploration bias. *arXiv preprint arXiv:2310.08558*, 2023.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets, 2021. URL https://arxiv.org/abs/2006.09359.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning, 2024. URL https://arxiv.org/abs/2303.05479.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl, 2025a. URL https://arxiv.org/abs/2410.20092.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning, 2025b. URL https://arxiv.org/abs/2502.02538.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2018. URL https://arxiv.org/abs/1709.10087.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization, 2024. URL https://arxiv.org/abs/2409.00588.

Stefan Schaal. Learning from demonstration. In M.C. Mozer, M. Jordan, and T. Petsche (eds.), *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. URL https://proceedings.neurips.cc/paper_files/paper/1996/file/68d13cf26c4b4f4f932e3eff990093ba-Paper.pdf.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning, 2019. URL https://arxiv.org/abs/1812.06298.

Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, Michael Bloesch, Konstantinos Bousmalis, Philemon Brakel, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Christine Chan, Oscar Chang, London Chappellet-Volpini, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, Adrian Collister, David B. D'Ambrosio, Sudeep Dasari, Todor Davchev, Meet Kirankumar Dave, Coline Devin, Norman Di Palo, Tianli Ding, Carl Doersch, Adil Dostmohamed, Yilun Du, Debidatta Dwibedi, Sathish Thoppay Egambaram, Michael Elabd, Tom Erez, Xiaolin Fang, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Ruiqi Gao, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Oliver Groth, Agrim Gupta, Roland Hafner, Steven Hansen, Leonard Hasenclever, Sam Haves, Nicolas Heess, Brandon Hernaez, Alex Hofer, Jasmine Hsu, Lu Huang, Sandy H. Huang, Atil Iscen, Mithun George Jacob, Deepali Jain, Sally Jesmonth, Abhishek Jindal, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Matija Kecman, J. Chase Kew, Donnie Kim, Frank Kim, Junkyung Kim, Thomas Kipf, Sean Kirmani, Ksenia Konyushkova, Li Yang Ku, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Tuan Anh Le, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Guy Lever, Jacky Liang, Li-Heng Lin, Fangchen Liu, Shangbang Long, Caden Lu, Sharath Maddineni, Anirudha Majumdar, Kevis-Kokitsi Maninis, Andrew Marmon, Sergio Martinez, Assaf Hurwitz Michaely, Niko Milonopoulos, Joss Moore, Robert Moreno, Michael Neunert, Francesco Nori, Joy Ortiz,

Kenneth Oslund, Carolina Parada, Emilio Parisotto, Amaris Paryag, Acorn Pooley, Thomas Power, Alessio Quaglino, Haroon Qureshi, Rajkumar Vasudeva Raju, Helen Ran, Dushyant Rao, Kanishka Rao, Isaac Reid, David Rendleman, Krista Reymann, Miguel Rivas, Francesco Romano, Yulia Rubanova, Peter Pastor Sampedro, Pannag R Sanketi, Dhruv Shah, Mohit Sharma, Kathryn Shea, Mohit Shridhar, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Rachel Sterneck, Ian Storz, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Saran Tunyasuvunakool, Jake Varley, Grace Vesom, Giulia Vezzani, Maria Bauza Villalonga, Oriol Vinyals, René Wagner, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Chengda Wu, Markus Wulfmeier, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Jimmy Yan, Sherry Yang, Skye Yang, Yuxiang Yang, Hiu Hong Yu, Wenhao Yu, Wentao Yuan, Yuan Yuan, Jingwei Zhang, Tingnan Zhang, Zhiyuan Zhang, Allan Zhou, Guangyao Zhou, and Yuxiang Zhou. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer, 2025. URL https://arxiv.org/abs/2510.03342.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 34556–34583. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/uchendu23a.html.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards, 2018. URL https://arxiv.org/abs/1707.08817.

Andrew Wagenmaker, Perry Dong, Raymond Tsao, Chelsea Finn, and Sergey Levine. Posterior behavioral cloning: Pretraining bc policies for efficient rl finetuning, 2025. URL https://arxiv.org/abs/2512.16911.

Hanlin Yang, Chao Yu, Siji Chen, et al. Hybrid policy optimization from imperfect demonstrations. *Advances in Neural Information Processing Systems*, 36:4653–4663, 2023.

Haichao Zhang, We Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning, 2023. URL https://arxiv.org/abs/2302.00935.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data, 2025. URL https://arxiv.org/abs/2412.07762.

A Additional Experiments

To further examine the effect of pretraining the Q-function on downstream performance, we report results for EXPO and RLPD with a UTD ratio of 20. The setup mirrors the experiments in Figure 2: we are given a pretrained policy, and the Q-function is pretrained on top of it. In the non-pretrained variant, the Q-function is instead initialized randomly and trained solely online. Across all settings, we find that pretraining the Q-function does not translate into a meaningful improvement in downstream performance for either method, and in some cases yields slightly lower performance. As discussed in Section 4.2, the Q-function learned during pretraining differs fundamentally from the one learned by online RL; as a result, pretraining on offline data fails to help compared to a randomly initialized Q-function.

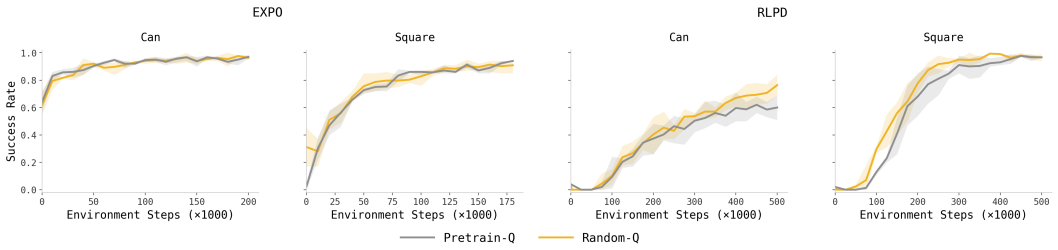

Figure 10: **Evaluation success rates of pretraining versus no pretraining of Q-functions.** Pretraining Q-functions for both EXPO with UTD 20 and RLPD with UTD 20 do not result in improvement in performance compared to a randomly initialized Q-function.

B Experiment Details

B.1 Preference Accuracy

We use a *preference accuracy* metric to evaluate whether a learned critic recovers action preferences identified by a strong reference critic. The reference critics are trained RL agents that successfully solve the task to 100% or close to 100% performance, which we can view as $Q^{\pi^*_{\text{RL}}}$. To reduce variance from any single trained critic and recover the distribution of $Q^{\pi^*_{\text{RL}}}$, we train 10 reference critics and average over them. For each state s_i, we compare two actions: an action a_i^{base} from the base policy and a reference action a_i^{ref}. The preference accuracy evaluates how often a critic prefers a reference action a_i^{ref} over a base action a_i^{base}. A Q-function that is calibrated should generally prefer a_i^{ref} over a_i^{base}.

B.2 Hyperparameters

This section provides additional details on our training procedure, pretraining datasets, and actor-generated rollout configurations across all six environments. Each row reports (N_{add}), the number of additional generated rollouts, and (p_f), the fraction of failed trajectories among those rollouts. Within each environment, the dataset size and (N_{add}) are held fixed across settings.

Table 1: Policy generated rollouts configurations.

Setting	square (ph)	can (mh)	lift (ph)	cube-double	scene	puzzle-4x6
Pretraining demonstrations	100	10	10	40	50	10
N_{add}, p_f	200, 0.3	20, 0.1	50, 0.1	80, 0.1	100, 0.1	30, 0.1

Table 2: Shared EXPO model and optimization hyperparameters.

Hyperparameter	Value
Base actor optimizer	AdamW
Edit actor optimizer	Adam
Critic optimizer	Adam
Temperature optimizer	Adam
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Adam coefficients $(\beta_1, \beta_2, \epsilon)$	$(0.9, 0.999, 10^{-8})$
Discount factor γ	0.99
Critic target update rate	0.005
Actor target update rate	0.001
Update-to-data ratio	1
Batch size	256
Offline-to-online batch ratio	0.5
Critic hidden dimensions	$(256, 256, 256)$
Number of Q-functions	10
Number of min Qs	2
Critic layer normalization	True
Diffusion denoising steps	10
Diffusion noise schedule	Variance preserving
Diffusion time-embedding dimension	128
Diffusion residual blocks	3
Diffusion actor layer normalization	True
Number of action samples	8
Edit action scale	0.1

Table 3: Additional hyperparameters used for the Max Q + BC and Action-gradient Q-pretraining variants.

Variant	Hyperparameter	Value
Max Q + BC	Offline pretraining updates	100,000
Max Q + BC	Distillation coefficient α_{FQL}	10.0
Q Action gradient	Offline pretraining updates	100,000
Q Action gradient	Q-gradient coefficient	0.05
Q Action gradient	Gradient clipping threshold	1.0

B.3 Environments

Our evaluations use a set of tasks from OGBench (Park et al., 2025a) and Robomimic (Mandlekar et al., 2021). OGBench is a large-scale benchmark designed for goal-conditioned reinforcement learning (RL), where the agent must learn to reach any state from any other state in the dataset using as few steps as possible. In this work, we focus on a diverse set of manipulation tasks that evaluate the agent's object manipulation, sequential reasoning, and combinatorial generalization capabilities. Each task is compatible with the standard reward-maximizing RL algorithms and its reward ranges from $-n_{\text{tasks}}$ to 0, depending on the number of completed subtasks. We use the following OGBench datasets for each domain:

- `cube-double-singletask-task2-v0`
- `scene-singletask-task2-v0`
- `puzzle-4x6-singletask-task2-v0`

The `cube-double` family of tasks involves complex pick-and-place manipulation of multiple colored cube blocks, where the robot arm must arrange cubes into designated target configurations. The `scene` tasks require long-horizon reasoning and interaction with diverse objects in unstructured environments, where the agent must sequentially combine learned manipulation primitives, such as opening and closing drawers or windows. The `puzzle` task tests the generalization capabilities of the agent, requiring it to achieve a desired color configuration of a puzzle. All experiments follow the official evaluation protocols and metrics defined by OGBench (Park et al., 2025a). For each setting, we report the mean and standard deviation of the success rate over three random seeds.

For Robomimic, we evaluate on the `lift-ph`, `square-ph`, and `can-mh` tasks, which require lifting a block, transferring a can into a target bin, and inserting a nut onto a square peg, respectively. The offline datasets are collected through RoboTurk (Mandlekar et al., 2018), a remote teleoperation platform, and consist entirely of successful demonstrations. The PH datasets contain demonstrations collected by a single experienced teleoperator, whereas the MH datasets consist of demonstrations collected by six teleoperators with varying levels of expertise (two experienced, two intermediate, and two novice operators).