EFFICIENT ONLINE REINFORCEMENT LEARNING FINE-TUNING NEED NOT RETAIN OFFLINE DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

The modern paradigm in machine learning involves pre-training models on diverse data, followed by task-specific fine-tuning. In reinforcement learning (RL), this translates to learning via offline RL on a static dataset, followed by rapid online RL fine-tuning using autonomous interaction data. Most RL fine-tuning methods require continued training on offline data for stability and performance. This is undesirable because retaining offline data is both slow and expensive for large datasets, but has been inevitable so far. In this paper, we show that retaining offline data is unnecessary as long as we use a correctly-designed online RL approach for fine-tuning offline RL initializations. We start by analyzing the role of retaining offline data in online fine-tuning. We find that continued training on offline data is mostly useful for preventing a sudden divergence from the offline RL value function at the onset of fine-tuning, caused by a distribution mismatch between the offline data and online rollouts. As a result, this leads to unlearning and forgetting the benefits of offline pre-training. Our approach, WSRL, mitigates a catastrophic forgetting of pre-trained priors by using a warmup phase that seeds the online RL run with a very small number of rollouts from the pre-trained policy to do fast online RL. The data collected during warmup helps "recalibrate" the offline Q-function to the online distribution, allowing us to completely discard offline data without risking destabilizing the online RL training. We show that WSRL is able to fine-tune without retaining any offline data, and is able to learn faster and attains higher performance than existing algorithms irrespective of whether they do or do not retain offline data.

1 INTRODUCTION

The predominant paradigm for machine learning at scale today involves pre-training models on diverse prior datasets, and then fine-tuning them on a more limited amount of domain-specific data to specialize them to particular downstream tasks (Devlin et al., 2018; Brown et al., 2020; Driess et al., 2023; Radford et al., 2021; Zhai et al., 2023; Touvron et al., 2023; Zhou et al., 2024). In the context of learning decision-making policies, this paradigm translates to pre-training on a large amount of previously-collected static experience via offline reinforcement learning (RL) (Levine et al., 2020) methods, followed by fine-tuning these initializations via online RL efficiently. Generally, this fine-tuning is done by continued training with the very same offline RL algorithm (e.g., pessimistic (Kumar et al., 2020; Cheng et al., 2022) algorithms or algorithms that apply behavioral constraints (Fujimoto & Gu, 2021; Kostrikov et al., 2021)) on a mixture of offline data and autonomously-collected online data, with minor modifications to the offline RL algorithm itself (Nakamoto et al., 2024).

While this paradigm has led to promising results (Kostrikov et al., 2021; Nakamoto et al., 2024), unlike the standard practice in machine learning, RL fine-tuning requires continued training on offline data for stability and performance reasons (Zhang et al. (2023; 2024); Section 4). Retaining offline data during fine-tuning is problematic for multiple reasons. First, as offline datasets grow in size and diversity, training on offline data becomes inefficient, and impractical to the point that practitioners might prefer to simply not use online RL for fine-tuning. Second, the need for retaining offline data perhaps defeats the point of offline RL pre-training altogether: recent results (Song et al., 2023), also corroborated by our experiments in Section 4, show that current fine-tuning approaches are not able to make good use of several strong offline RL value and/or policy initializations as better fine-tuning performance can be obtained by directly running online RL from scratch with offline data put in the

replay buffer (Ball et al., 2023). All of this questions the efficacy of current online RL fine-tuning approaches.

Our goal is to address the aforementioned shortcomings of current online fine-tuning methods and build an online RL approach that does *not* retain offline data, making it practical for users to easily fine-tune offline RL policies. To develop our approach, we first empirically analyze the importance of retaining offline data in current offline-to-online fine-tuning algorithms. We find that for both pessimistic (e.g., CQL (Kumar et al., 2020)) and behavioral constraint (e.g., IQL (Kostrikov et al., 2021)) algorithms, the offline Q-function undergoes a "recalibration" phase at the onset of online fine-tuning where its values change substantially. This recalibration phase can lead to unlearning of the offline initialization, and even divergence, when no offline data is present for training. Even methods specifically designed for fine-tuning (Nakamoto et al., 2024) still suffer from this problem with limited or no offline data. We show that the main culprit behind the unlearning is the distribution mismatch between the offline data and online training distribution, and retaining offline data attenuates the effect of this mismatch, playing an essential role in the working of current offline-to-online fine-tuning methods. *Is it possible to transition into online fine-tuning from offline RL value and policy initializations without catastrophically forgetting offline pre-training, and without retaining offline data?*

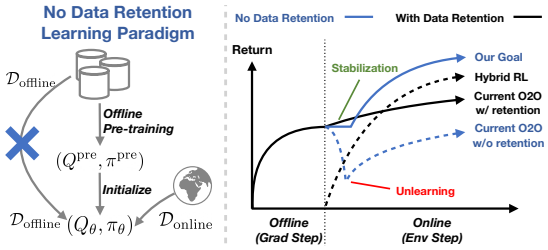


Figure 1: **No Data Retention fine-tuning** is an offline pre-training, online RL fine-tuning paradigm we focus on that mirrors the common paradigm in machine learning at scale today. An offline dataset is first used to offline pre-train a policy π^{pre} and a Q-function Q^{pre} which are then used to initialize online RL fine-tuning of the Q-function and the policy (Q_θ, π_θ). The most critical and challenging constraint in this paradigm is that the online RL fine-tuning may **not retain offline data** and only be able to use the pre-trained policy and Q-function. Naïvely applying state-of-the-art offline-to-online RL methods to paradigm fail due to **learning instability** at the beginning of the online fine-tuning that destroys the offline initializations. Our goal is to develop an online fine-tuning method that is able to leverage pre-trained initializations while **stabilizing** the online learning.

Our key insight is that seeding the online fine-tuning with even a small amount of appropriately collected online data that "simulates" offline data retention can greatly facilitate recalibration, preventing catastrophic forgetting that never recovers with continued training. Once this recalibration is over, we can run the most effective online RL approach (without pessimism or constraints) for the most efficient learning. Our approach, WSRL (**W**arm **S**tart **R**einforcement **L**earning), instantiates this idea by incorporating a warmup phase to initialize the online replay buffer with a small number of online rollouts from the pre-trained policy, and then running the best online RL method with various offline RL initializations to fine-tune. WSRL is able to learn faster and attains higher asymptotic performance than existing algorithms irrespective of whether they do or do not retain offline data. We emphasize that this is not a particularly novel or clever algorithm, and perhaps a very simple approach to fixing the problem (though to our knowledge such a warmup phase has not been previously applied for offline-to-online RL), but it is quite effective for fine-tuning offline initializations, without any complex design choices.

Our main contribution in this paper is the study of RL online fine-tuning with no offline data retention, a paradigm we call ***no-retention fine-tuning***. We provide a detailed analysis of existing offline-to-online RL methods and find that offline data is often needed during fine-tuning to mitigate the Q-value divergence due to distribution shift, but can also slow down fine-tuning. We demonstrate that if online fine-tuning is done correctly, we can use a simple method (WSRL) that does not require data retention and perform fine-tuning faster with better asymptotic performance.

2 Related Work

Offline-to-online RL. Offline-to-online RL focuses on leveraging an offline dataset to run online RL as sample-efficient as possible (Lee et al., 2022; Nair et al., 2020). Many methods developed for this setting utilize offline pre-training followed by a dedicated fine-tuning phase (Nair et al., 2020; Kostrikov et al., 2021; Agarwal et al., 2022; Hu et al., 2023; Rafailov et al., 2023; Nakamoto et al.,

2024) *on a mix of offline and online data*. Offline RL methods can also be directly used to fine-tune online by continued training while adding new online data to the offline data buffer (Kumar et al., 2020; Kostrikov et al., 2021; Tarasov et al., 2024). Most similar to the goal in our paper is Agarwal et al. (2022), which attempts to use previous RL computations as a better initialization for downstream tasks. However, this work, along with all the methods above, still require retaining all of the pre-training data in the data buffer. As we also show, these methods completely fail without the offline data in the buffer. Our work does not retain offline data. Uchendu et al. (2023) utilize pre-trained policy to guide online fine-tuning without the need of offline data retention, but do not show how to initialize the offline pre-trained Q-function. As we show in our experiments (Section 6), initializing the Q-function is crucial in achieving strong online fine-tuning performance across environments. Ji et al. (2023) and Luo et al. (2024) run offline RL and online RL concurrently on a shared replay buffer, following the idea of tandem learning (Ostrovski et al., 2021) such that online RL can benefit from the offline RL algorithms. Although the high-level motivating principle behind this line of work is also to use offline RL to boost online RL efficiency, there's no offline pre-training.

Bottlenecks in online RL fine-tuning of offline RL policies. In this work we show that offline data retention greatly stabilizes the recalibration of the Q-function at the onset of fine-tuning, which otherwise can lead to unlearning due to state-action distribution shift. Luo et al. (2023) observe that putting the offline data into the offline RL replay buffer stabilizes fine-tuning but can slow down learning. But this prior work did not attempt to study why offline data hurts fine-tuning, which our analysis aims to answer. Lee et al. (2022) identify the existence of state-action distribution shift between offline data and online rollout data, but do not explicitly analyze the negative effects of this shift in online fine-tuning. Nakamoto et al. (2024) show the poor calibration of offline pre-trained Q-function to be a key cause for instability of pessimistic algorithms during online fine-tuning with offline data retention, though this analysis is restricted to the use of pessimism and does not apply to constraint methods, and as we show in our analysis, their final CalQL approach still requires offline data to function well. Not only does our analysis not retain offline data Nakamoto et al. (2024), but we also answer why this can be problematic due to no distribution shift.

Online RL with prior data without offline pre-training. Another line of work bypasses offline RL pre-training altogether, directly using a purely online RL agent to learn on data samples from both offline data and online interaction data from scratch (Song et al., 2022; Zhou et al., 2023; Ball et al., 2023). Despite not using pre-training, this recipe can work well across the board, often outperforming offline-to-online fine-tuning methods that utilize a separate offline pre-training phase. If the most effective way to utilize prior data is to include it in the replay buffer without any pre-training at all–no matter which pre-training algorithm is used–then it perhaps indicates that we are missing some important ingredients for a truly scalable RL formula for pre-training and fine-tuning. In this paper, we show that at least a big part of the problem lies in online fine-tuning of offline RL initializations, and build an extremely simple approach to fix the problem.

Fine-tuning RL policies with no data retention. Many continual and lifelong RL methods also fine-tune policies without retaining prior experiences due to the non-stationarity assumption in the environment dynamics and task specification (Ring, 1994; Kirkpatrick et al., 2017; Huang et al., 2021; Wołczyk et al., 2021; Powers et al., 2022). Meta-RL methods (Duan et al., 2016; Rothfuss et al., 2018; Stadie et al., 2018; Rakelly et al., 2019; Arndt et al., 2020; Dorfman et al., 2021; Grigsby et al., 2023) assume access to a task/environment distribution to optimize for fast fine-tuning online. In contrast, we only consider the single-environment, single task setting where the pre-training and fine-tune are in the same environment for the same task. In the same single-environment, single task setting, many prior works study on-policy RL methods (e.g., PPO (Schulman et al., 2017)) to fine-tune pre-trained policies (Schaal, 1996; Kober & Peters, 2008; Rajeswaran et al., 2017; Gupta et al., 2019; Wołczyk et al., 2024; Ren et al., 2024). Among these, Wołczyk et al. (2024) also observe unlearning in the beginning of the fine-tuning and find that explicitly mitigating the unlearning with techniques from continual learning improves the efficiency of fine-tuning. In contrast to these prior works, our study focuses on off-policy actor-critic RL methods, that provide an elevated sample efficiency, and require different solution strategies to address this unlearning problem.

3 Problem Formulation: Fine-Tuning without Offline Data

We operate in an infinite-horizon Markov Decision Process (MDP), $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathbf{P}, r, \gamma, \rho\}$, consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition dynamics function $\mathbf{P}(s'|s,a) : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{P}(\mathcal{A})$, a reward function $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, a discount factor $\gamma \in [0, 1)$, and an initial state distri-

bution $\rho : \mathcal{P}(\mathcal{S})$. We have access to an offline RL pre-trained policy $\pi^{\text{pre}}_{\psi}(a|s) : \mathcal{S} \mapsto \mathcal{P}(\mathcal{A})$ and pre-trained Q-function $Q^{\text{pre}}_{\theta}(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ obtained by running some offline RL algorithm (e.g., CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2021), CalQL (Nakamoto et al., 2024)) on some prior data. We denote this offline dataset as $\mathcal{D}_{\text{off}}$. Our goal is to build an online fine-tuning algorithm that uses the pre-trained policy $\pi^{\text{pre}}_{\psi}(a|s)$ as a policy initialization for online RL training in the MDP $\mathcal{M}$. The goal of this online RL fine-tuning is to train $\pi_{\psi}(a|s)$ so that it maximizes the discounted return: $\eta(\pi) = \mathbb{E}_{s_{t+1}\sim\mathbf{P}(\cdot|s_t,a_t),a_t\sim\pi(\cdot|s_t),s_0\sim\rho} \sum_{t=0}^{\infty} [\gamma^t r(s_t, a_t)]$.

Problem setup. Crucially note that the online RL fine-tuning problem we study *does not* allow retaining $\mathcal{D}_{\text{off}}$. We will refer to this problem setting ***no retention online fine-tuning***. Conceptually, our problem setting is close to the standard offline-to-online fine-tuning problem setting (Nair et al., 2020; Kostrikov et al., 2021; Nakamoto et al., 2024), but no data retention is possible.

4 Understanding the Role of Offline Data in Online Fine-Tuning

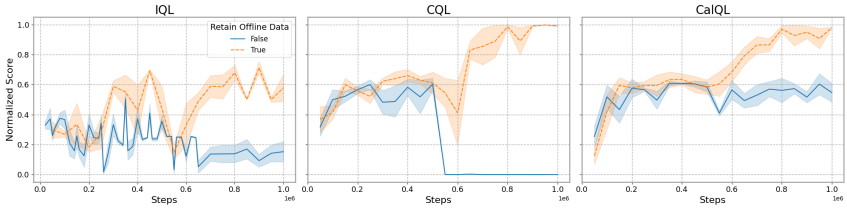


Figure 2: In no-retention fine-tuning, IQL (left), CQL (middle), and CalQL (right) all fail to fine-tune on `kitchen-partial`. In contrast, when co-training on offline data during fine-tuning, these algorithms work as intended. All agents are pre-trained for 500k steps and fine-tuned for 500k steps.

To make progress towards our goal of developing an RL algorithm that works well without any data retention, we first attempt to understand the role that retaining offline data plays in online fine-tuning of current offline RL methods. In particular, we hope to gain insights into developing new methods that serve a similar role, but do not require retaining offline data. We center our study along two axis: **(1)** we analyze the role of retaining offline data at the onset of fine-tuning, and **(2)** we analyze the effect of retaining offline data on asymptotic fine-tuning performance.

4.1 The Role of Offline Data at the Onset of Fine-Tuning

Extending observations from previous work (Nakamoto et al., 2024), we find that the online fine-tuning of offline RL algorithms fails severely if no offline data is retained. Specifically, observe in Figure 2 that offline RL algorithms IQL (Kostrikov et al., 2021) and CQL (Kumar et al., 2020) fails immediately at the onset of fine-tuning, with performance dropping down to nearly a 0% success rate on the `kitchen-partial` task from D4RL (Fu et al., 2020b). Moreover, neither CQL nor IQL is able to recover over the course of fine-tuning. CalQL (Nakamoto et al., 2024), an offline RL approach specifically designed for subsequent online fine-tuning by leveraging calibrated Q-functions, experiences performance drop initially but improves with further online training. However, it still struggles to improve beyond its pre-trained performance. To be more concrete, while all of the algorithms suffer from some sort of **"unlearning"** at the beginning of online fine-tuning (i.e., referring to the performance drop at the start of fine-tuning), some algorithms suffer from a more severe **"catastrophic forgetting"** phenomenon such that recovery becomes nearly impossible with online training. While it is reasonable to expect that there some "drop" in performance (i.e., unlearning as per the above terminology) might be inevitable since the stochastic policy must now act on new states it visit when online fine-tuning begins, forgetting is much more problematic. Figure 2 indicates a bottleneck in fine-tuning with online RL without offline data, and different offline RL initializations suffer from this challenge to different extents. Why?

Why is retaining offline data necessary for some algorithms? We compare Q-values at the onset of fine-tuning when retaining different amounts of offline data. This enables us to build a mental picture of what precisely goes wrong in online fine-tuning as offline data is gradually removed. We find in Figure 3 (b) that the average Q-values under the offline distribution begin to diverge as the amount of retained offline data decreases. This Q-value divergence in turn corresponds to a divergence in the TD-error (Figure 3 (c)), perhaps highlighting some sort of catastrophic forgetting.

Diving deeper, we find that this divergence only happens under the distribution of the offline data (on which we evaluate metrics but do not train): TD-error on online data attains similar values

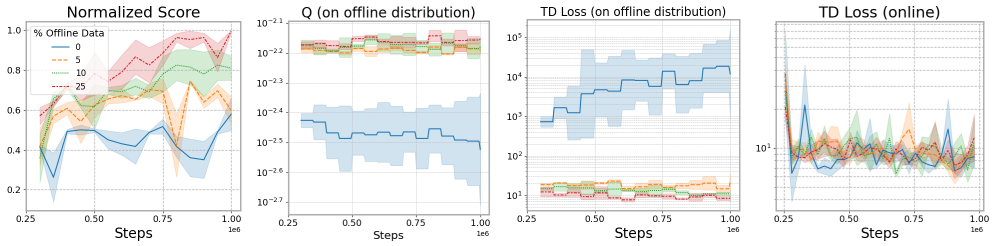


Figure 3: **Fine-tuning CalQL on `kitchen-partial` with varying amounts of offline data.** The curves represent online update batches that include 0%/5%/10%/25% offline data. Fine-tuning starts at step $250k$. (a) Shows that incorporating more offline data improves performance and is necessary for surpassing pre-trained performance. (b) Highlights how Q-values on offline transitions tend to blow up with less offline data. (c) Similarly, the critic loss on offline transitions increases significantly with reduced offline data. (d) Demonstrates that the online critic loss is optimized consistently regardless of offline data retention. Panels (c) and (d) are shown on a log scale. These trends underscore the importance of co-training sufficient offline data to maintain stability and improve performance when fine-tuning CalQL. We have similar findings with IQL and CQL.

regardless of the amount of offline data retained (Figure 3(d)); on the other hand, the TD error under the offline data distribution grows substantially as the amount of offline data decreases during fine-tuning (Figure3 (c)). Such divergence happens for all of CQL, IQL, and CalQL. We find that the divergence for CalQL is the least servere, as shown in Figure 3, which correlates with the stability and best performance of CalQL in this problem setting in Figure 2. This suggests that the problem with no data retention in current offline-to-online fine-tuning algorithms likely stems from a form of ***distribution shift*** between the online rollout data and offline data distribution: fine-tuning offline RL initializations on more on-policy data destroys how well temporal difference relations are satisfied on offline data. As we will see, this can lead to unlearning and forgetting of the pre-trained initialization.

Takeaway 1: Distribution shift between offline and online data destroys Q-function fit

Our analysis suggests that training on on-policy rollout data during fine-tuning destroys how well the model is able to fit the offline data: despite attaining similar TD-errors on the online data, TD-errors under the offline dataset distribution keep growing.

Why do Q-values diverge? Not only does the TD-error under the offline data distribution grow, but we also observe a divergence in Q-values at the onset of fine-tuning (see for e.g., Figure 4). This Q-value divergence is a manifestation of the "***recalibration***" process (Nakamoto et al., 2024) at the boundary between offline RL and online fine-tuning. Unlike the setting of Nakamoto et al. (2024), the recalibration process in no retention fine-tuning must operate entirely on limited on-policy rollouts, since we do not retain any offline data in this phase. Thus we see that despite explicit modifications to the scale of the offline Q-function initialization in CalQL (Nakamoto et al., 2024), this approach is still insufficient at preventing forgetting when offline data is not retained.

Next, we wish to intuitively understand why recalibration leads to divergent Q-values, specifically for the case of continuing to run pessimistic offline RL algorithms (e.g., CQL or CalQL) during fine-tuning. Consider running the CQL loss on the very first batch of online rollouts collected from the environment. The target values for the TD-error on these online state-action pairs will query the pre-trained offline Q-function, Q_θ^{pre} on state-action pairs that are out-of-distribution of the offline dataset. Due to the inherent nature of the conservative regularizer in CQL (and CalQL), Q-values at out-of-distribution state-action pairs are expected to take very small values. Using such small values for computing TD targets in the Bellman backup will, in turn, propagate these *underestimation* errors onto the previous state-action pair through the TD-error. Meanwhile, the conservative regularizer from CQL still continues to push down out-of-distribution Q-values, which means that at the onset of fine-tuning when the number of on-policy rollouts is small, Q-values for several actions at new states will keep getting smaller.

This mechanistic understanding of the onset of fine-tuning hints at a form of a "downward spiral" in the Q-function, until the Q-function begins to recover from its initialization by correctly backing up environment reward. But usually by this point, the policy has degraded and is no longer able to recover to its offline performance (Figure 2). We find that this unlearning followed by recovery takes substantially longer to finish and is more and more detrimental as the amount of offline data reduces,

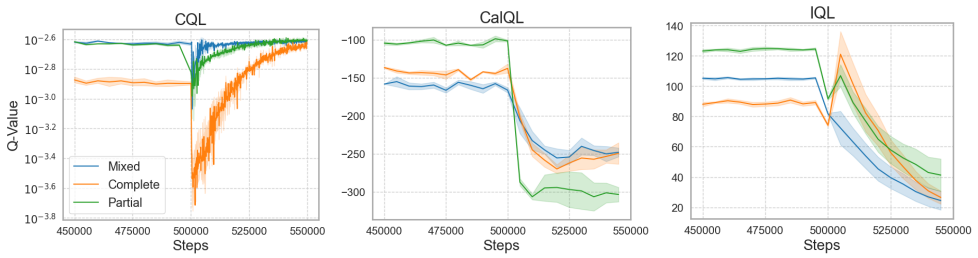


Figure 4: A downward spiral effect in CQL (left), CalQL (middle), and IQL (right) Q-functions in no-retention fine-tuning on `Kitchen-mixed`, `Kitchen-complete`, and `Kitchen-partial`: When fine-tuning starts at 500k steps, Q function goes on a downward spiral. When it eventually recovers, the policy has already unlearned (Figure 2).

as shown in Figure 3, with the most adverse effects when no offline data is present. In fact, contrary to the claims of Nakamoto et al. (2024), we find that even calibrated offline RL algorithms, such as CalQL, can suffer from this challenge (Figure 4 middle), though we did find them to be more robust than offline RL algorithms.

Takeaway 2: Re-calibration of Q-values leads to excessive underestimation

We find that Q-value recalibration at the onset of fine-tuning leads to excessive underestimation due to backups with over-pessimistic TD-targets.

4.2 The Adverse Impact of Offline Data on Asymptotic Performance

As shown above, offline data seems to play an important role in fine-tuning of current offline-to-online algorithms at the onset of fine-tuning by helping to recalibrate the Q-values and preventing Q-value divergence. But how does it affect performance in the longer term, once recalibration is over? In this section, we present results showing that continued training on offline data *hurts* final performance and sample efficiency. Specifically, we find that offline RL followed by RL fine-tuning tends to be substantially slower than online RL algorithms from scratch that simply initialize the online replay buffer with the offline data (Ball et al., 2023; Song et al., 2023) (see Figure 5 for results justifying this claim). This is quite concerning because it indicates that either offline RL pre-training provides no benefits for fine-tuning (unlike other fields of machine learning where pre-training helps substantially) or that existing RL fine-tuning approaches from various offline RL initializations are not effective enough in making the best use of an offline initialization. In the next section, we show that via a simple modification to online RL methods in the high updates-to-data (UTD) regime, we are able to make good use of initializations from several offline RL algorithms, without offline data.

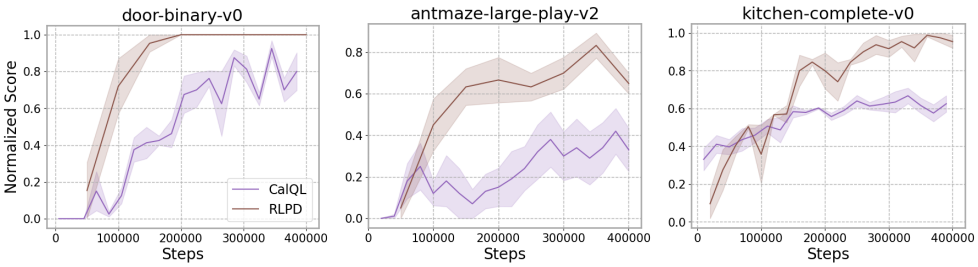


Figure 5: Retaining offline data is not efficient, and is outperformed by online RL methods like RLPD on three different environments. RLPD starts from scratch, and CalQL starts from pre-trained at step 0.

Takeaway 3: Retaining offline data hurts asymptotic performance

While retaining offline data appears to be crucial for preventing unlearning during recalibration at the onset of fine-tuning in current methods, continuing to make updates on it with an offline RL algorithm negatively impacts asymptotic performance and sample efficiency.

5 WSRL: Fast Fine-Tuning Without Offline Data Retention

In the previous section, we saw that retaining offline data in offline RL algorithms slows down online fine-tuning to the point that running online RL from scratch with offline data works better. However,

we cannot simply remove the offline data to speed up fine-tuning because offline RL algorithms will suffer from unlearning and catastrophic forgetting. How can we tackle *both* catastrophic forgetting of the offline initialization and attain asymptotic sample efficiency online?

Key idea. Perhaps one straightforward approach to address asymptotic efficiency issues is to utilize a standard online RL approach, with no pessimism or constraints for fine-tuning, unlike current offline-to-online fine-tuning approaches that still retain this offline RL tooling during fine-tuning. We can further accelerate online learning by operating in the high updates-to-data (UTD) regime (Ball et al., 2023). The remaining question is: how do we tackle catastrophic forgetting at the onset of fine-tuning that prevents further improvements online, without offline data? Our insight is that we can "simulate" continued training on offline data by collecting a small number of *warmup* transitions with a *frozen* offline RL policy at the onset of online fine-tuning. Training on these transitions via an aggressive, high updates-to-data (UTD) online RL approach, without retaining offline data can mitigate the challenges of catastrophic forgetting. Our approach, WSRL (**W**arm **S**tart **R**einforcement **L**earning) instantiates these insights into an extremely simple and practical method that enables us to obtain strong fine-tuning results without offline data.

5.1 WSRL Algorithm: Initialization, Learning, and Optimization

WSRL is an off-policy actor-critic algorithm (see pseudocode in Algorithm 1). At the onset of fine-tuning, it initializes the value function and policy with the pre-trained Q-function Q_{off} and policy π_{off} respectively. This offline initialization could come from any offline RL algorithm; Appendix G presents results of running WSRL multiple offline RL methods. Then, WSRL uses the first K steps of online fine-tuning to collect a few rollouts using the frozen offline RL policy to simulate the retention of offline data. We refer to this phase as the "warmup" phase. After warmup data collection, WSRL trains both the value and policy using standard temporal-difference (TD) updates and policy gradient (in this case, a reparameterization based policy gradient estimator).

For fine-tuning, we wish to use the best practices and techniques for most efficient online learning, following the empirical study in Ball et al. (2023). To this end, we run fine-tuning in a high updates-to-data (UTD) regime (Fu et al., 2019; Chen et al., 2021), enabling WSRL to learn rapidly online. To combat issues such as Q-value overestimation in RL (Hasselt, 2010), especially in the high UTD regime, we use an ensemble of Q functions (Chen et al., 2021) and layer normalization (Hiraoka et al., 2022) in both the actor and the critic.

5.2 Implementation Details

We experimented with CQL, IQL, and CalQL offline RL pre-training as initializations and found that all three initializations produce similar performance. Most of the results in this paper use CalQL to initialize WSRL, even though in principle other initializations could also be used. Appendix G contains ablation studies of different initializations. We choose soft actor-critic (Haarnoja et al., 2018a), with an ensemble of 10 Q-networks and layer normalization after every layer in the both the actor and the critic, as our online fine-tuning algorithm. This design is inspired by the work of Ball et al. (2023). We use $K = 5000$ warmup steps at the onset of fine-tuning for all of our experiments. Further implementation details are provided in Appendix H.

6 Experimental Evaluation

The goal of our experiments is to study how well WSRL is able to fine-tune with online RL without offline data retention. To this end, we compare WSRL to previous state-of-the-art fine-tuning methods and rapid online RL algorithms on a variety of challenging benchmarks. We also ablate the design decisions in WSRL to understand the efficacy of WSRL. We study the following questions: **(1)** Can WSRL enable efficient finetuning in the no-retention setting?; **(2)** How does WSRL compare with methods that do retain offline data?; **(3)** How critical is the warmup phase in WSRL?; **(4)** How important is it to use online RL algorithm for online fine-tuning?, and **(5)** How important is it to pretrain the policy, value function, or both?

6.1 Baselines and Prior Comparisons

While most prior methods in offline-to-online RL are not designed to handle the no-retention fine-tuning setting, there are existing methods that can be directly applied or repurposed to our setting. **JSRL** (Uchendu et al., 2023) uses a pre-trained policy as an exploration policy to roll in for some number of steps during each episode, and the online policy is trained from scratch with both the

roll in experience and the policy's own rollout experience. To improve JSRL's competitiveness, we further initialize the online policy with the pre-trained policy's weight. Offline RL methods have also been shown to be able to fine-tune online. We consider two offline methods, **CQL** (Kumar et al., 2020) and **IQL** (Kostrikov et al., 2021). To evaluate them in the no-retention fine-tuning setting, we discard the offline data and initialize the replay buffer to be empty in the beginning of fine-tuning, as opposed to the typical recipe for fine-tuning these algorithms. We also consider **CalQL** (Nakamoto et al., 2024), a variant of CQL that calibrates the Q-values during learning for efficient fine-tuning. While the typical CalQL fine-tune recipe involves sampling each update batch from both the offline dataset and the online replay buffer, we evaluate it in the no-retention setting by only sampling from the online buffer. **SO2** (Zhang et al., 2024) is an RL fine-tuning algorithm that is designed to balance the RL objective the pessimistic pre-training through high UTD and perturbed value udpates. SO2 requires initializing the replay buffer with the entire offline dataset. Finally, **RLPD** (Ball et al., 2023) is an efficient online RL algorithm that learns from scratch. One important design decision is that it does $50/50$ sampling of the offline dataset and online buffer during online RL. When applying this approach in the no-retention setting, we only sample from the online buffer, making it similar to a rapidly updating Soft Actor Critic (Haarnoja et al., 2018a) agent, which we refer to as **SAC (fast)**.

6.2 Experimental Setup

We evaluate WSRL on a variety of challenging benchmark tasks and pre-training datasets used by prior works (Nakamoto et al., 2024; Kostrikov et al., 2021; Kumar et al., 2020): **(1)** The `Antmaze` tasks from D4RL (Fu et al., 2020a) are a class of long-horizon navigation tasks that require controlling an 8-DOF Ant robot to reach a goal with a sparse reward. The agent has t learn to "stitch" experiences together from a suboptimal dataset. In addition to the original mazes from D4RL, we include `Antmaze-Ultra` (Jiang et al., 2022), a larger and more challenging maze. We only include three of the hardest Antmaze environments in Section 6, and provide results on all eight Antmazes in Appendix A. **(2)** The `Kitchen` environment is a long-horizon manipulation task to control a 9-DoF Franka robot arm to perfrom 4 sequential subtasks in a simulated kitchen. **(3)** The `Adroit` environments are a suite of dexterous manipulation tasks to control a 28-DoF five-fingered hand to manipulate a pen to desired position, open a door, and relocating a ball to desired position. The agent observes a binary reward when it succeeds. Each data has an offline dataset that provides a narrow offline dataset of 25 human demonstrations and additonal trajectories collected by a behavior cloning policy. **(4)** The `Mujoco Locomotion` environments in D4RL are dense reward settings where agents learn to control robotic joints to perform various locomotion tasks.[1]

6.3 Can WSRL enable efficient fine-tuning in no-retention fine-tuning?

Figure 6 compares WSRL with previous methods applied to the no-retention fine-tuning setting. In seven different environments, WSRL is able to significantly outperform baseline methods, fine-tuning faster to a higher asymptotic performance. Note that while WSRL experience an initial dip in policy performance early on (see Figure 16 for a zoomed-in version of the first $50k$ steps of fine-tuning), it recovers quickly and outperforms other baselines which do not recover at all. We provide a detailed analysis of this dip in Appendix D, illustrating why WSRL **unlearns** initially but does **not catastrophically forget** its priors. CQL, IQL, and CalQL completely fail to fine-tune in this setting, as observed before in Section 4 because of Q-value divergence. SAC (fast), which only updates with the online experience using an ensemble of Q functions with a high UTD, completely fails in exploration-heavy

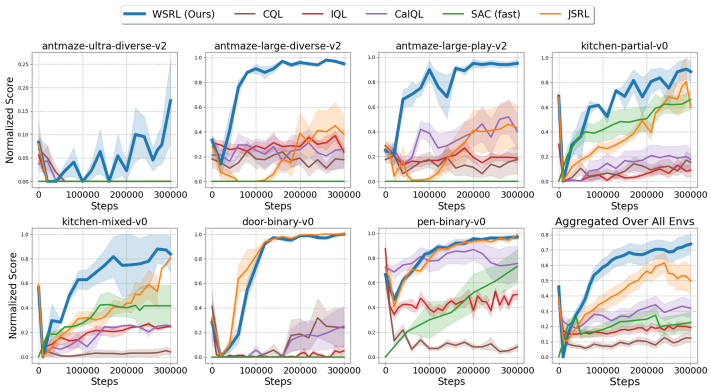


Figure 6: In no-retention fine-tuning, WSRL fine-tunes efficiently and greatly outperforms previous algorithms, which often fail catastrophically.

[1]Results in Appendix B.

environments, but can improve slowly in some environments. The most competitive baseline is JSRL with policy initialized with pre-training: WSRL is significantly better than JSRL on `Antmaze` and `Kitchen`, and roughly the same on `Adroit`. This is because JSRL actually implicitly employs a warmup period: the roll-in periods of JSRL is similar to warmup, where it uses a frozen pre-trained policy to collect online data throughout the course of fine-tuning. The improvement of WSRL over JSRL likely stems from value-initialization: on datasets where the pre-trained Q-function is good (e.g. `Antmaze`), value-initialization helps significantly. We provide a more detailed disuccsion and ablation of the impact of value-initialization in Section 6.7.

6.4 How does WSRL compare to methods that retain offline data?

In Figure 7 we compare WSRL to previous methods that do utilize offline data during fine-tuning. For example, CalQL here would sample both offline data and online data for each update batch. To make comparison fair, we also compare to a version of CalQL that uses high UTD online, so all methods use UTD=4. WSRL also outperforms the baselines that retain the offline dataset and benefits from more information during fine-tuning. Specifically, WSRL usually achieves higher asymptotic performance than CalQL and fine-tunes faster, indicating co-training on the offline data hurts performance and slows down learning, as we have shown in Section 4. WSRL also out performs RLPD, indicating that WSRL can effectively utilize the pre-trained value function and policy to do rapid online learning.

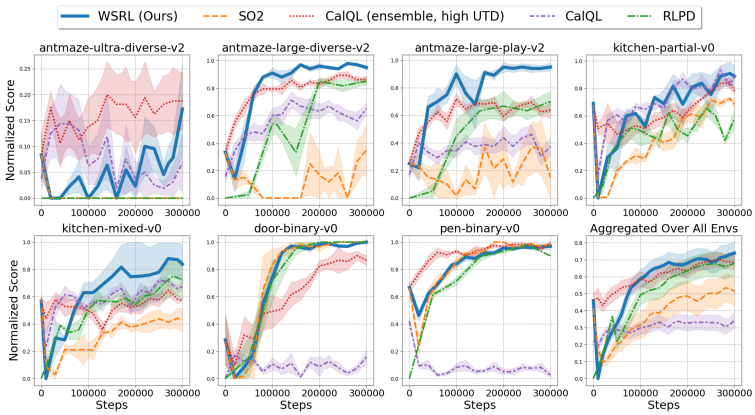


Figure 7: Compared to methods that **do retain offline data** online, WSRL, perhaps surprisingly, is still able to fine-tune faster or competitively.

6.5 How critical is the warmup phase?

We find that the warmup phase is crucial for fine-tuning with online RL. We freeze the pre-trained policy and value to collect environment interactions for $K = 5000$ steps before any online RL updates. As Figure 8 shows, WSRL without warmup does significantly worse in three different kinds of environment. We find that such warmup phase helps because it dynamically re-calibrates the scale of pre-trained Q-values (i.e. it adjust the value-scale of Q-values from potentially conservative pre-training). As Figure 9 shows, when we initialize with a warmup phase, the Q-value scale does not diverge to over pessimistic values during fine-tuning and the TD losses remain small. See more detailed discussion in Appendix F.

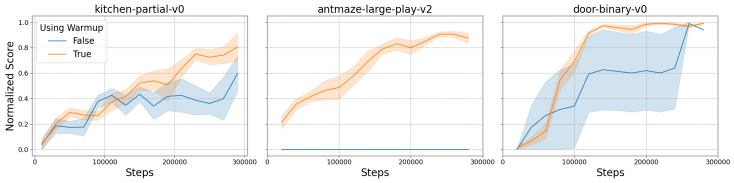


Figure 8: Warmup is critical to fast fine-tuning: When WSRL does not use the initial 5000 steps of warmup, it performs significantly worse.

6.6 How important is using online RL algorithm for fine-tuning?

Aside from using a warmup phase and initializing with the pre-trained weights, using an online RL algorithm for fine-tuning is also a critical design choice in WSRL. We ablate this decision by attempting to use an offline RL algorithm during fine-tuning. Here we choose to use CalQL, because it is more unlikely to experience Q-divergence (Section 4) as compared to CQL and IQL. We also initialize the CalQL agent with the pre-trained policy and value function, and use the same number of warmup steps online. As shown in Figure 11b, using an offline algorithm is significantly worse than using the online RL algorithm, SAC.

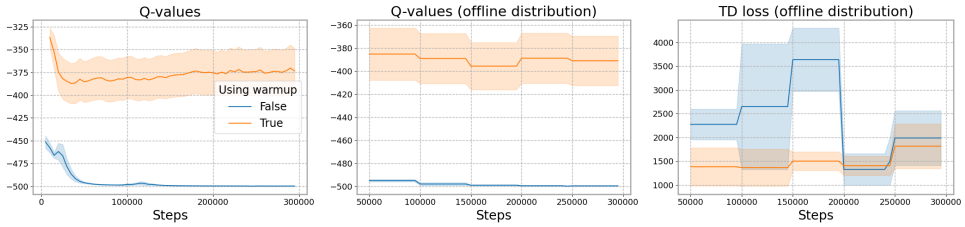


Figure 9: Warmup phase helps Q-value stabilization: (left) Q-values during fine-tuning, and warmup mitigates over-pessimistic values; (middle, right) Q-value and TD error evaluated on the offline distribution, where warmup prevents divergence. Data from WSRL on `Antmaze-large-play`.

6.7 How important is it to initialize the policy, value function, or both?

Importance of policy initializaiton. At the start of the online fine-tuning phase, WSRL initializes the online policy to the pre-trained policy. Since the pre-trained policy is already capable of meaningful interactions with the environment from pre-training, it speeds up online learning. In Figure 10, we compare WSRL's performance with and without "policy initialization", and find that initializing with the pre-trained policy is crucial for fast fine-tuning. In this ablation, we leave out `Kitchen-complete` because the pre-trained policy has 0% performance.

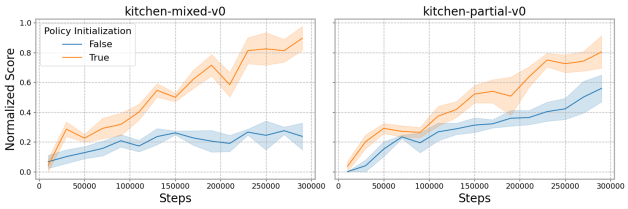
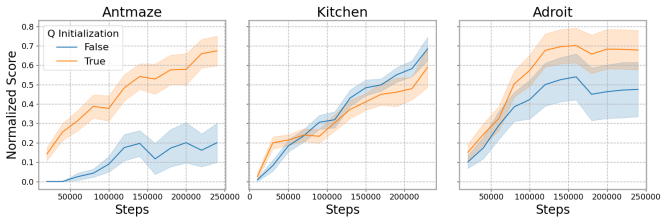
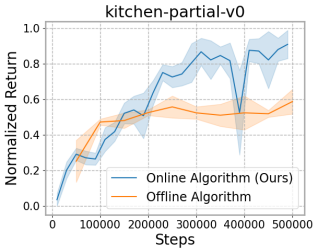


Figure 10: Importance of policy initialization in WSRL: with policy initialization, WSRL performs much better in `Kitchen`.

Benefits of Q-value initialization. In Figure 11a, we find that, while initializing the value function did not bring additional benefits in some domains, it made fine-tuning much faster in others. In particular, initializing with the Q-function was especially helpful in the `Antmaze` domains. We hypothesize that this is because the pre-training datasets in Antmazes have much broader coverage compared to those in `Adroit` and `Kitchen`, resulting in a better offline Q-function. Consequently, initializing with a more informative Q-function in Antmazes accelerates online fine-tuning.



(a) Q-function initialization is especially helpful when the pre-training dataset has high coverage (e.g. `Antmazes`). Each plot shows results averaged across different Antmaze/Kitchen/Adroit environments.



(b) Importance of using an online RL algorithm during no-retention fine-tuning. Here, SAC learns significantly faster than CalQL.

Figure 11

7 Conclusion

In this paper, we explore the possibility of fine-tuning RL agents online without retaining and co-training on any offline datasets. Such setting is important for truly scalable RL, where offline RL is used to pre-train on a diverse dataset, followed by online RL fine-tuning where keeping the offline data is expensive or impossible. We find that previous offline-to-online RL algorithms fail completely in this setting because of Q-value divergence due to distribution shift. However, if we simply use online RL algorithm for fine-tuning and allow the Q-values to stabilize through a warmup phase, we can prevent the Q-divergence. We hope that WSRL sheds light on the challenges in no-retention fine-tuning, and inspire future research on the important paradigm of no-retention RL fine-tuning.

8 Reproducibility Statement

We describe all the implementation details in Appendix H, which should enable researchers to reproduce our algorithm. We also remark that our algorithmic modifications are fairly simple and a large portion of experiments that we run for this paper include analyzing existing methods. We will share the code for the new method (WSRL) during the review process and also release the code publicly.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. *Advances in neural information processing systems*, 35:28955–28971, 2022.

Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation. In *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 2725–2731. IEEE, 2020.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd.

C. Cheng, T. Xie, N. Jiang, and A. Agarwal. Adversarially Trained Actor Critic for Offline RL. *ICML*, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta reinforcement learning–identifiability challenges and effective data collection strategies. *Advances in Neural Information Processing Systems*, 34:4607–4618, 2021.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. In *arXiv*, 2020a. URL https://arxiv.org/pdf/2004.07219.

Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep q-learning algorithms. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020b.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. *arXiv preprint arXiv:2310.09971*, 2023.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *arXiv*, 2018a. URL https://arxiv.org/pdf/1801.01290.pdf.

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

Hado van Hasselt. Double q-learning. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, 2010.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xCVJMsPv3RT.

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

Yizhou Huang, Kevin Xie, Homanga Bharadhwaj, and Florian Shkurti. Continual model-based reinforcement learning with hypernetworks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 799–805. IEEE, 2021.

Tianying Ji, Yu Luo, Fuchun Sun, Xianyuan Zhan, Jianwei Zhang, and Huazhe Xu. Seizing serendipity: Exploiting the value of past success in off-policy actor-critic. *arXiv preprint arXiv:2306.02865*, 2023.

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Advances in neural information processing systems*, 21, 2008.

Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. *arXiv preprint arXiv:2103.08050*, 2021.

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

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643*, 2020.

Yicheng Luo, Jackie Kay, Edward Grefenstette, and Marc Peter Deisenroth. Finetuning from offline reinforcement learning: Challenges, trade-offs and practical solutions. *arXiv preprint arXiv:2303.17396*, 2023.

Yu Luo, Tianying Ji, Fuchun Sun, Jianwei Zhang, Huazhe Xu, and Xianyuan Zhan. Offline-boosted actor-critic: Adaptively blending optimal historical behaviors in deep off-policy rl. *arXiv preprint arXiv:2405.18520*, 2024.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Mitsuhiko Nakamoto, Simon 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. *Advances in Neural Information Processing Systems*, 36, 2024.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pp. 16828–16847. PMLR, 2022.

Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The difficulty of passive learning in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:23283–23295, 2021.

Sam Powers, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Gupta. CORA: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents. In *Conference on Lifelong Learning Agents*, pp. 705–743. PMLR, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Rafael Rafailov, Kyle Beltran Hatch, Victor Kolev, John D Martin, Mariano Phielipp, and Chelsea Finn. Moto: Offline pre-training to online fine-tuning for model-based robot learning. In *Conference on Robot Learning*, pp. 3654–3671. PMLR, 2023.

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. *arXiv preprint arXiv:1709.10087*, 2017.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340. PMLR, 2019.

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. *arXiv preprint arXiv:2409.00588*, 2024.

Mark Bishop Ring. *Continual learning in reinforcement environments*. The University of Texas at Austin, 1994.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

Stefan Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yuda Song, Yifei Zhou, Ayush Sekhari, J Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid RL: Using both offline and online data can make RL efficient. *arXiv preprint arXiv:2210.06718*, 2022.

Yuda Song, Yifei Zhou, Ayush Sekhari, Drew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid RL: Using both offline and online data can make RL efficient. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=yyBis80iUuU.

Bradly C Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. Jump-start reinforcement learning. In *International Conference on Machine Learning*, pp. 34556–34583. PMLR, 2023.

Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Maciej Wołczyk, Bartłomiej Cupiał, Mateusz Ostaszewski, Michał Bortkiewicz, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Fine-tuning reinforcement learning models is secretly a forgetting mitigation problem. *arXiv preprint arXiv:2402.02868*, 2024.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11975–11986, 2023.

Haichao Zhang, We Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. *arXiv preprint arXiv:2302.00935*, 2023.

Yinmin Zhang, Jie Liu, Chuming Li, Yazhe Niu, Yaodong Yang, Yu Liu, and Wanli Ouyang. A perspective of q-value estimation on offline-to-online reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16908–16916, 2024.

Yifei Zhou, Ayush Sekhari, Yuda Song, and Wen Sun. Offline data enhanced on-policy policy gradient with provable guarantees. *arXiv preprint arXiv:2311.08384*, 2023.

Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models. *arXiv preprint arXiv:2407.20635*, 2024.

Appendices

A Additional Results on Antmaze Environments

In the main paper, we presented results on three of the most challenging antmaze environments. Here, in addition to the set of three antmaze environments shown in Figures 6 and 7, we provide the results of WSRL on all eight D4RL antmaze environments, together with strong baseline methods.

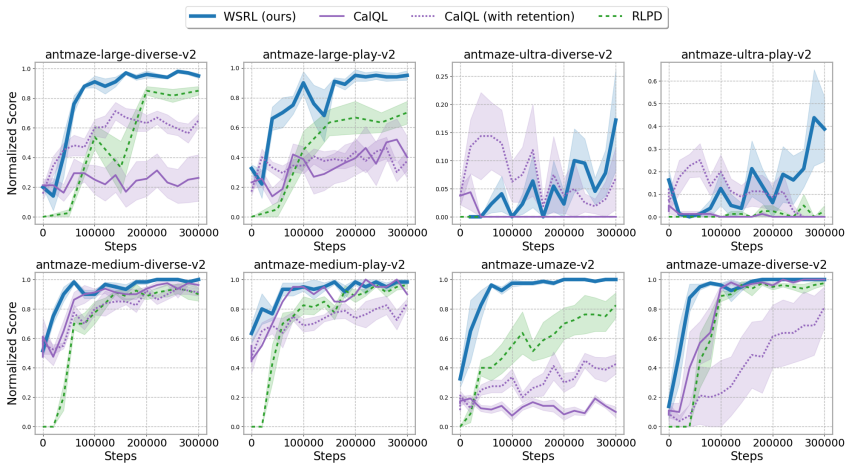


Figure 12: WSRL on all eight D4RL antmaze environments, along with RLPD and CalQL baselines. Step 0 shows the start of fine-tuning for WSRL and CalQL, and start of RLPD. Solid lines do not retain offline data, while dotted lines do.

B Results on Mujoco Locomotion Environments

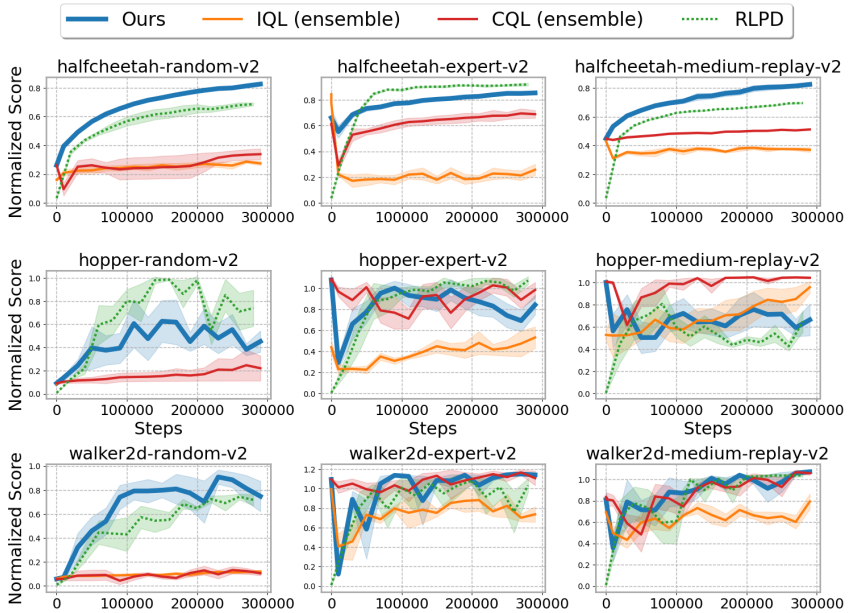


Figure 13: WSRL on nine Mujoco locomotion environments with dense rewards, along baselines. Step 0 shows the start of fine-tuning for WSRL and CalQL, and start of RLPD. Solid lines do not retain offline data, while dotted lines do.

Additionally, we also apply WSRL on nine different Mujoco locomotion domains. Specifically, we experiment with three different robot embodiments (`Halfcheetah`, `Hopper`, and `Walker`), each with three different types of datasets. The `random` datasets are collected with a random policy; the `expert` datasets are collected with a policy trained to completion with SAC; and the `medium-replay` datasets are collected with the replay buffer of a policy trained to the performance approximately 1/3 of the expert. As Figure 13 shows, WSRL outperforms or is similar to the best baseline methods.

For WSRL, the hyperparameters were exactly as those in Section 6 and listed in Appendix H with one exception: its pre-trained policy and value function are done with CQL offline training instead of CalQL. This is because these offline datasets have dense rewards and do not end in a terminal state, and therefore do not have ground-truth return-to-go to support the CalQL regularizer. For the same reason we did not include a CalQL baseline in Figure 13. Both the IQL and CQL baseline in Figure 13 uses an ensemble of 10 Q function, along with layer normalization in the Q functions.

C Ablation Studies on Warmup Phase

Impact of different warmup types. One natural question arises: why does the simple approach of warming up the replay buffer significantly boost performance during fine-tuning? One hypothesis is that seeding the replay buffer with some data helps prevent early overfitting, as much work has found in online RL (Nikishin et al., 2022). To test this hypothesis, we plot in Figure 14 in the fine-tuning performance of initializing with random actions, and compare it to initializing with pre-trained policy actions as well as not initializing the buffer atl all. It is clear that seeding the buffer with random actions significantly underperform the warmup approach, and in fact does not even provide much benefit as compared to not seeding the buffer at all. This suggests that the reason warmup phase helps is not because of preventing overfitting.

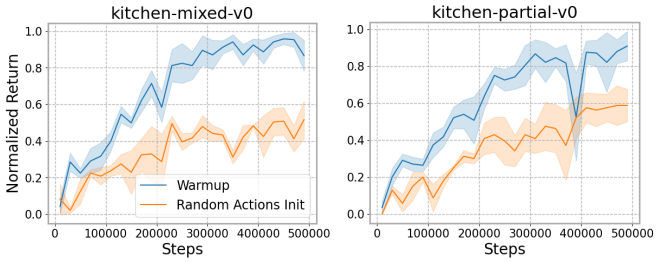


Figure 14: Comparing seeding the buffer with random actions to actions from the pre-trained policy: initializing with the pre-trained policy action works significantly better on `kitchen-mixed` (left) and `kitchen-partial` (right).

Impact of different length warmups.

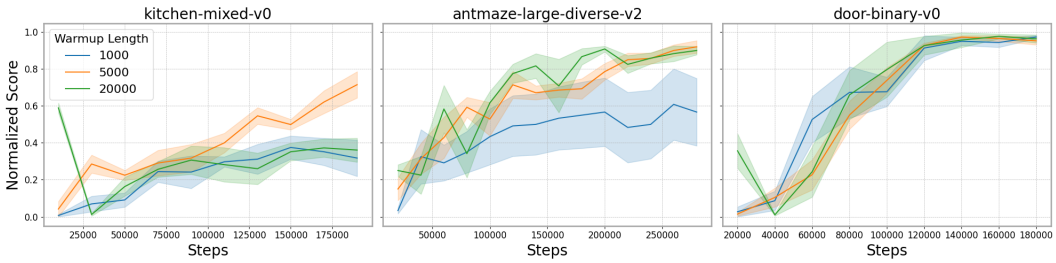


Figure 15: Impact of warmup phase of length 1k, 5k, 20k on `Kitchen-mixed` (left), `Antmaze-large-diverse` (middle), and `Door-binary` (right).

Since warmup phase seems to be critical to efficient online fine-tuning, we study whether the length of this warmup phase impacts fine-tuning performance. Figure 15 shows warmup phase of lengths 1k, 5k, and 20k on three different environments. It is clear that short warmup phase (1k) sometimes

lead to worse asymptotic performance or instability during fine-tuning. On the other hand, longer warmup phases could also hurt (e.g. on **Kitchen-mixed**) because it adds too much offline-like data into the replay buffer and slows down online improvement. In WSRL, we did not tune the lengths of the warmup phase beyond what is shown in Figure 15, and we use 5000 warmup steps for all environments.

D START OF FINE-TUNING: DID WSRL UNLEARN ITS PRIORS?

To further illustrate the behavior of offline-to-online RL agents at the start of online fine-tuning, we show in Figure 16 the performance of WSRL, along with two other algorithms, evaluated at much smaller intervals than Figure 7. We fine-tune all agents for $50,000$ steps online across six different environments, and evaluate every $1,000$ environment steps.

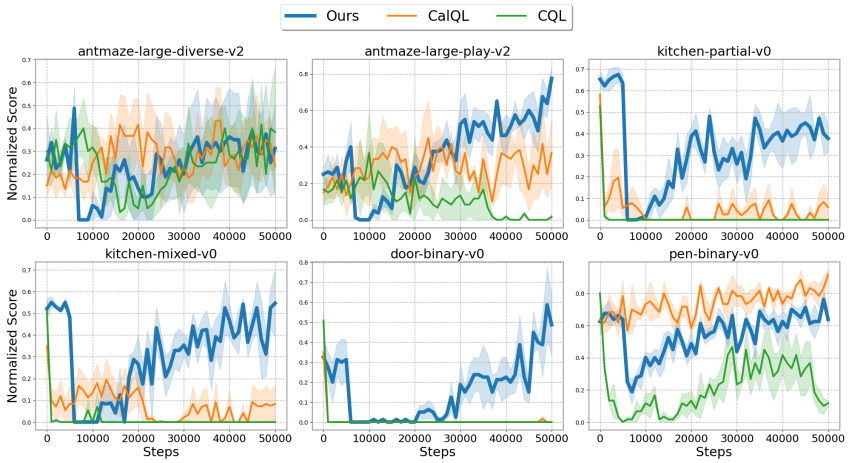


Figure 16: First $50,000$ steps of fine-tuning with denser evaluation intervals. Step 0 in the plot show the start of online fine-tuning. All agents are evaluated in the no-retention fine-tuning setting.

Figure 16 shows that WSRL experiences an initial dip in policy performance after $5,000$ steps of warmup, but recovers much faster than CQL and CalQL. We hypothesize that such a dip might be inevitable at the start of online fine-tuning in the no offline data retention setting because the policy is experiencing different states than what it was trained on and potentially states it has never seen (We analyze this with much more detail below). Moreover in some environments (e.g., binary reward environments), one might expect small fluctuations in the policy to manifest as large changes in the actual policy performance. However, such brief performance dip does not mean the policy/Q function has been catastrophically destroyed, which is evidenced by the fact that WSRL recovers faster than its peer algorithms and learns faster than online RL algorithms such as RLPD (See Figure 7). If this initial dip would have destroyed all pre-training knowledge from the policy, then we would not expect quick recovery.

In fact, in general, it is impossible to build a no data retention fine-tuning algorithm whose performance does not initially degrade as we move from offline data to online training on all environments and offline data compositions. Intuitively, this is because it violates a sort of "no free lunch" result: for example, consider a sparse reward problem where the reward function is an arbitrary non-smooth function over actions, here even a minor change in policy action results in a catastrophic change in return. Therefore just deducing whether an algorithm has lost is prior or not based on performance may not be the most informative. Instead, a more meaningful metric to measure catastrophic forgetting is to evaluate how much a fine-tuning algorithm with no data retention deviates from its pre-training, and how fast it can adjust to the online state-action distribution.

Therefore, to investigate how much the policy has deviated from its offline pre-training during the initial "performance dip", we plot the KL divergence between the offline pre-trained policy and the online fine-tuned policy ($D_{\mathrm{KL}}(\pi_{\text{offline}}||\pi_{\text{online}})$) in Figure 17. In Figure 17 (Top), we can see that D_{KL} on the offline distribution generally increases during fine-tuning for all three agents. This in-

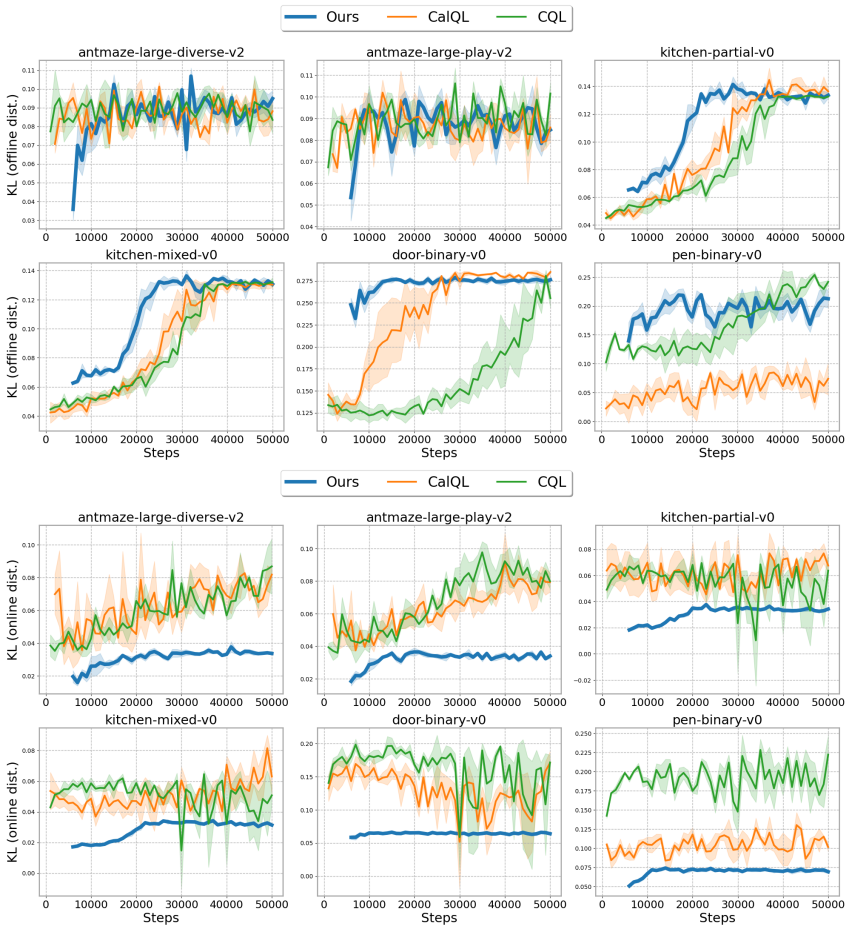


Figure 17: KL divergence $D_{KL}(\pi_{offline}||\pi_{online})$ between the pre-trained offline policy and the fine-tuned online policy at the first $50k$ steps of fine-tuning. The top plot shows the KL divergence evaluated on the offline dataset distribution; the bottom plot shows the KL divergence on the online state-and-action distribution, sampled from the replay buffer. WSRL is not plotted during the first $5,000$ steps of warmup. CQL and CalQL do not retain offline data.

crease indicates that the fine-tuned policy has deviated from the pre-trained policy on at least some parts of the dataset distribution. This is actually expected in the no-retention fine-tuning setting because of the distribution shift from offline to online. To be more specific, for example, in `Antmaze` environments, the offline dataset exhibits a very diverse state-action distribution, covering almost all the locations in the entire maze, while fine-tuning is a single-goal navigation task. In no-retention fine-tuning, the agent is incentivized to forget about parts of the offline dataset that is irrelevant to the fine-tuning task and specialize to the online task. Similarly, the `Kitchen-partial` dataset contains extra tasks not seen during fine-tuning and `Kitchen-mixed` dataset doesn't specify actions after subtask completion, both of which should be actively forgotten during no-retention fine-tuning. As such, we see an increase in D_{KL} on the offline distribution in Figure 17 where the irrelevant information should be forgotten. However, the `Adroit` datasets contain expert human demos and behavior cloning policy rollouts, and both are relevant to the online task. As such, we see minimal D_{KL} increase in the adroit environments, suggesting that the policy did not forget useful priors in pre-training. Compared to CQL and CalQL, WSRL generally has the same asymptotic value for D_{KL} but reaches convergence much faster. This suggests that WSRL actively adapts to the online distribution much quicker compared to its no-retention counterparts.

On the other hand, Figure 17 (Bottom) shows D_{KL} on the online distribution for WSRL increases slightly, but is much smaller compared to CQL and CalQL (without data retention). This indicates that WSRL's policy remains almost the same on the online distribution. This is desirable because

the pre-trained policy already has decent performance, and a capable fine-tuning algorithm should not forget that capability while adjusting slightly to unseen (but not out-of-distribution) states. This suggests that WSRL remains stable during fine-tuning and **does not destroy priors learned from offline pre-training**. In summary, due to the distribution shift from offline pre-training to no-retention fine-tuning, WSRL is forgetting experience in the offline dataset that was learned during offline pre-training but is in reality irrelevant for specializing to the online task. In contrast, it is instead specializing to the online task.

E Does Freezing the Policy at the Start of Fine-Tuning Help?

Since Appendix D shows a brief period at the start of fine-tuning where policy performance take a dip, one natural question is whether such a dip is avoidable. The most straight forward way to avoid such a drop in policy performance is to freeze the policy during initial fine-tuning. In other words, for N steps at the on set of fine-tuning, we only pass the gradients through the Q function and train the Q function, but freeze the policy. After N online steps, we start training both the policy and the Q function.

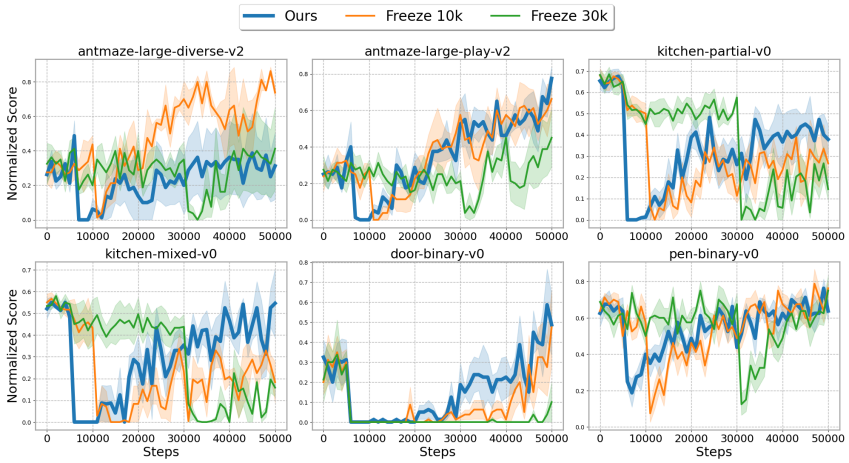


Figure 18: Freezing the policy for $N \in \{10k, 30k\}$ steps at the onstart of fine-tuning doesn't prevent the performance dip. In the plot, we show policy performance of WSRL vs. WSRL with initial policy freeze across six environments. Step 0 is the start of online fine-tuning.

Figure 18 shows the performance of WSRL after freezing the policy for $N = \{10k, 30k\}$ steps. It's obvious that even when we freeze the policy for some number of steps to let the Q-function adjust online, the policy still suffers a dip after it is unfrozen. In fact, this is somewhat an expected result because the policy needs to adjust to the OOD online state-action distribution, as well as the new online Q-function, and such adjustment process is expected to make the policy performance worse.

F Why Warm-up Prevents Q-Values Divergence

In WSRL, the policy and value function is pre-trained offline with CalQL (Nakamoto et al., 2024), and the online fine-tuning process is done with SAC (Haarnoja et al., 2018a). This change of RL algorithm could lead to miscalibration issues, where the pre-trained values are more pessimistic than ground truth values. As we have shown in Section 4, this hurts fine-tuning when it backs up a pessimistic target Q-value through the Bellman update. This particularly hurts when the Bellman target is computed on an OOD state-action pair, because OOD state-action pair have more pessimistic values than state-action paris seen in the offline dataset, by the nature of pessimistic pre-training. If there were no warm-up phase, the agent will collect OOD data into the buffer, leading to Bellman backups with pessimistic target Q values, which in turn leads to Q-divergence. This is the "downward spiral" phenomenon in Section 4. However, warmup solves this problem by putting more offline-like data into the replay buffer where Q-values are not as pessimistic, thereby preventing

the downward spiral in the online Bellman backups and uses high UTD in online RL to quickly re-calibrate the Q-values.

G Ablation Studies on Different Types of Value Initialization

WSRL is agnostic to the offline RL pre-training algorithm. Furthermore, we find that it is not crucial which specific offline RL algorithm we use to obtain the pre-trained Q values. In Figure 19, we show that the Q-values from IQL, CQL, and CalQL work just as well on three different environments, even though CQL optimizes for conservative Q-values, CalQL is less conservative, and IQL is not conservative at all. In particular, we observe that Calibrated Q-values as an initialization provides some small performance benefits on `Kitchen-mixed`. Therefore, we use calibrated Q-values from CalQL offline pre-training for our main experiments.

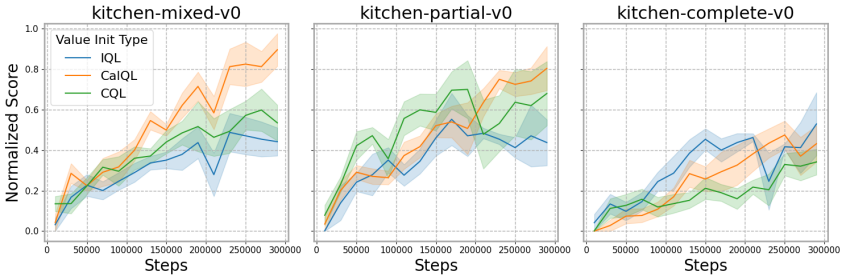


Figure 19: The offline RL algorithm used to pre-train the Q-values does not affect performance: for `Kitchen-mixed` (left), `Kitchen-partial` (middle), and `Kitchen-complete`, WSRL is able to achieve similar performances by initializing with pre-trained values from CQL, IQL, and CalQL, though CalQL initializations have small benefits on `Kitchen-partial`.

H Implementation Details

Pseudocode. See Algorithm 1

Algorithm 1 WSRL: **W**arm **S**tart **R**einforcement **L**earning

Require: Offline RL algorithm $\mathcal{A}_{\text{off}}$, Pre-training dataset D_{offline}.
  $Q_{\text{off}}, \pi_{\text{off}} \leftarrow$ train_offline($\mathcal{A}_{\text{off}}, D$) ▷ Offline RL pre-training
Require: $Q_{\text{off}}, \pi_{\text{off}}$, Online RL algorithm $\mathcal{A}_{\text{on}}$, Replay buffer $\mathcal{R} \leftarrow \phi$, warmup step K.
  $Q \leftarrow Q_{\text{off}}$ ▷ Value initialization
  $\pi \leftarrow \pi_{\text{off}}$ ▷ Policy initialization
  $\mathcal{D}_{\text{online}} \leftarrow \emptyset$
  while step $\leq$ max steps **do**
    if step $\leq K$ **then**
      $(s, a, s', r) \leftarrow$ interact(π_{off}, environment) ▷ Warmup Phase
    else
      $(s, a, s', r) \leftarrow$ interact(π, environment)
    end if
    $\mathcal{D}_{\text{online}} \leftarrow \mathcal{D}_{\text{online}} \cup \{(s, a, s', r)\}$
    if step $> K$ **then**
      $B \sim D_{\text{online}}$
      $Q, \pi \leftarrow$ train_high_utd($\mathcal{A}_{on}, B$) ▷ Online RL updates
    end if
  end while

WSRL Hyperparameters. We use 5K warmup steps ($K = 5,000$). For the online RL algorithm in WSRL, we use the online SAC (Haarnoja et al., 2018b) implementation in RLPD (Ball et al., 2023) with a UTD of 4, batch size of 256, actor learning rate of $1e-4$, critic learning rate of $3e-4$,

and actor delay of 4 (update the actor once for every four critic steps). For the AntMaze tasks, we disable the entropy backup and MinQ network in SAC. When we initialize the policy network and the Q-function network from offline RL pre-training, we keep the optimizer state of these networks.

I WSRL WITH OFFLINE DATA RETENTION

In the main paper, we have shown that WSRL can efficiently fine-tune without retaining the offline pre-training dataset. One natural question arises: can WSRL do even better if we allow offline data retention? To answer this question, we run WSRL with the online replay buffer initialized with the whole offline dataset. Figure 20 shows that on average, retaining the offline data does not give WSRL any advantages, probably because it already has the necessary knowledge in the offline policy and Q-function; WSRL is actually a bit faster later on in fine-tuning, perhaps due to the fact that it is updating the policy on more online data.

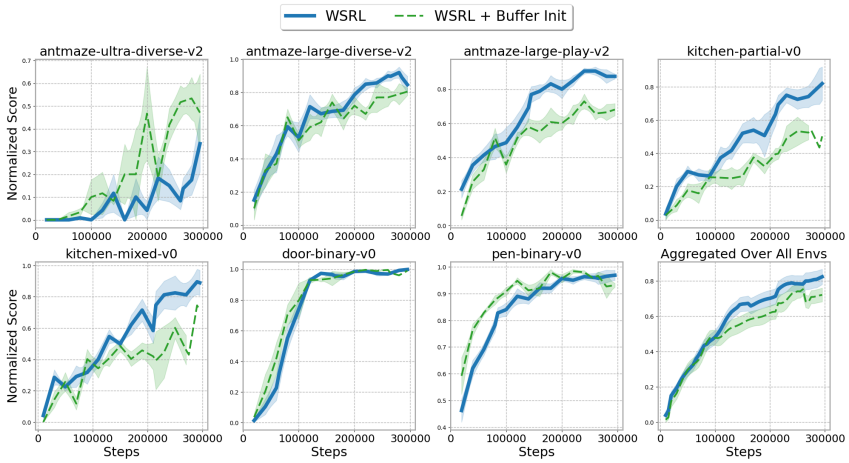


Figure 20: On average, initializing the replay buffer with the offline dataset does not give WSRL any advantage during fine-tuning, and may make it a bit slower.

J WSRL WITH VARYING LEVELS OF OFFLINE POLICY

We investigate how WSRL performs with varying levels of expertise of the offline pre-trained policy. Specifically, we consider `Kitchen-complete-v0` and `Relocate-binary-v0`, two especially hard tasks for offline RL where pre-training with CalQL leads to poor performance. In `Recolate-binary-v0`, CalQL completely fails and has pre-trained performance near 0; CQL and IQL also has pre-training performance 0, indicating that this task is inherently hard for offline RL agents. In `Kitchen-complete-v0`, CalQL (15.47%) significantly underperforms IQL (70.83%) despite our tunning efforts, which suggests there is some inherent limitation in CalQL learning a good Q-funciton in this domain. Not surprisingly, Figure 21 shows that WSRL also performs poorly: while WSRL can learn somewhat in `Kitchen-complete-v0` with a non-zero initialization, it completely fails to learn in `Adroit-binary-v0`. This is expected because when pre-training fails, initializing with the pre-trained network may not bring any useful information gain, and may actually hurt fine-tuning by reducing the network's plasticity (Nikishin et al., 2022), a known issue in online RL.

K WARMUP WITH TRANSITIONS FROM THE OFFLINE DATASET

We have shown in Section 6 that the warmup period is essential for efficient fine-tuning with WSRL. One interesting question is whether such warmup data can be collected by sampling the offline dataset, instead of online interactions with the frozen pre-trained policy as in WSRL. Therefore, we run an ablation experiment in Figure 22 where we replace the $5,000$ steps of warmup period by

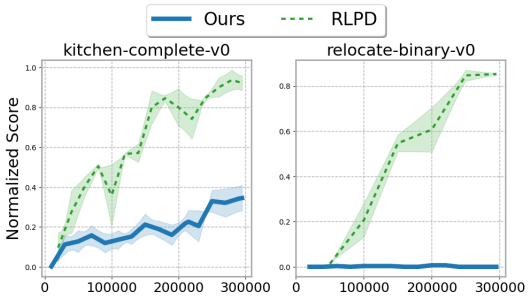


Figure 21: On environments where the pre-training completely fails, WSRL does not work well.

initializing the online replay buffer with $5,000$ random transitions sampled from the offline dataset, which we will refer to as "Dataset Warmup". As Figure 22 shows, while the two methods are similar on `Adroit` environments, WSRL is slightly better in `Kitchen` and much better on `Antmaze`. This is perhaps because the 5000 randomly sampled transitions might not be relevant to the online fine-tuning task, especially in `Antmaze` where the dataset has diverse state-action coverage (See Appendix D for a more detailed discussion). When the replay buffer is initialized with less relevant data, it is less effective at preventing Q-value divergence (Section 4) and recalibrating the online Q-function and policy. This perhaps highlights the utility of our approach: despite not having access to *any* offline data, WSRL is able to achieve similar or better performance than using transitions from the offline dataset in the no-retention fine-tuning setting.

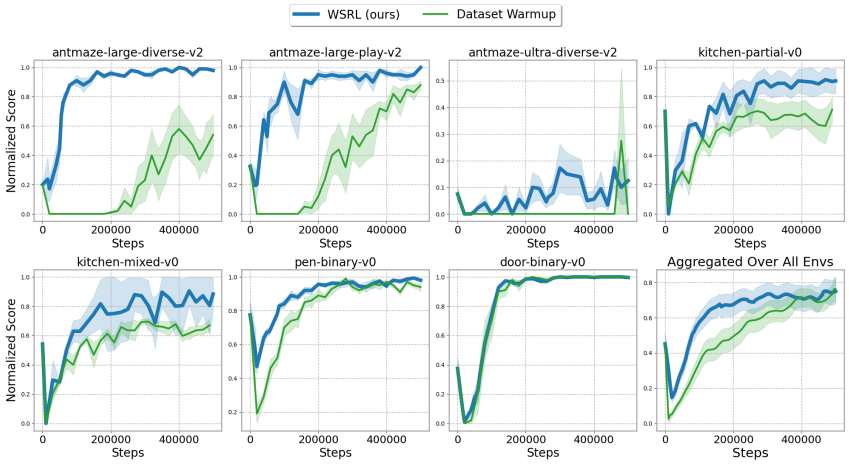


Figure 22: Overall, warming up with transitions from the offline dataset is less effective than warming up with online interactions (WSRL).