Thesis for Bachelor's Degree

ANCHOR: KL-Anchored Adaptation for Stable Offline-to-Online Reinforcement Learning

Geonwoo Cho

Department of Electrical Engineering and Computer Science,
College of Information and Computing

Gwangju Institute of Science and Technology

2026

학사학위논문

ANCHOR: 안정적인 오프라인-투-온라인 강화학습을 위한 KL 앵커링 기반 적응

조건우

정보컴퓨팅대학

전기전자컴퓨터공학과

광주과학기술원

2026

ANCHOR: KL-Anchored Adaptation for Stable Offline-to-Online Reinforcement Learning

Advisor: Sundong Kim

by

Geonwoo Cho

Department of Electrical Engineering and Computer Science

College of Information and Computing

Gwangju Institute of Science and Technology

A thesis submitted to the faculty of the Gwangju Institute of Science and Technology in partial fulfillment of the requirements for the degree of Bachelor of Science in the Department of Electrical Engineering and Computer Science Concentration

Gwangju, Republic of Korea

June 15, 2026

Approved by

Professor Sundong Kim

Committee Chair

ANCHOR: KL-Anchored Adaptation for Stable Offline-to-Online Reinforcement Learning

Geonwoo Cho

Accepted in partial fulfillment of the requirements for

the degree of Bachelor of Science

June 15, 2026

Committee Chair _____

Sundong Kim

Committee Member _____

Euiseok Hwang

Dedicated to my family.

BS/EC 20195171 Geonwoo Cho(조건우). ANCHOR: KL-Anchored Adaptation for Stable Offline-to-Online Reinforcement Learning (ANCHOR: 안정적인 오프라인-투-온라인 강화학습을 위한 KL 앵커링 기반 적응). Department of Electrical Engineering and Computer Science, College of Information and Computing . 2026. 55p. Prof. Sundong Kim.

Abstract

This paper studies offline-to-online reinforcement learning in the setting where the original offline replay buffer is unavailable during online fine-tuning. We analyze two competing failure modes: early catastrophic collapse when adaptation begins, and suppressed asymptotic performance when conservative regularization remains too strong. To address this tension, we propose ANCHOR, which keeps the actor close to the offline-pretrained policy through KL anchoring during the vulnerable early phase and then anneals the constraint while retaining a small residual critic regularizer. Across FrankaKitchen, AntMaze, and Adroit tasks, this design improves early stability while maintaining strong final performance. The paper also includes a theoretical interpretation of why anchoring helps preserve critic reliability under local occupancy shift, together with empirical ablations that clarify when the method is most effective.

BS/EC 20195171 조건우. ANCHOR: 안정적인 오프라인-투--온라인 강화학습을 위한 KL 앵커링 기반 적응. 전기전자컴퓨터공학과. 2026. 55p. 지도교수: 김선동.

국 문 요 약

본 논문은 온라인 미세조정 과정에서 기존 오프라인 리플레이 버퍼를 사용할 수 없는 오프라인-투--온라인 강화학습 문제를 다룬다. 이 설정에서는 온라인 적응이 시작될 때 성능이 급격히 하락하는 초기 붕괴 문제와, 오프라인 강화학습에서 사용된 보수적 정규화가 지나치게 강하게 유지될 때 최종 성능이 제한되는 문제가 동시에 발생할 수 있다. 본 논문은 이러한 상충 관계를 해결하기 위해 ANCHOR를 제안한다. ANCHOR는 온라인 적응 초기의 불안정한 단계에서 KL 앵커링을 통해 액터가 오프라인 사전학습 정책에서 급격히 벗어나지 않도록 제약하고, 학습이 진행됨에 따라 이 제약을 점진적으로 완화한다. 동시에 크리틱에는 작은 잔여 정규화 항을 유지하여 과도한 비관성을 줄이면서도 학습 안정성을 보완한다. FrankaKitchen, AntMaze, Adroit 과제에서 ANCHOR는 초기 안정성을 향상시키면서도 높은 최종 성능을 유지한다.

Contents

List of Tables

List of Figures

Chapter 1
Introduction

The pretrain-finetune paradigm has driven much of the recent success in modern machine learning across diverse domains such as natural language processing and computer vision [1, 2]. Inspired by these advances, reinforcement learning (RL) has adopted a similar paradigm: an agent is first pretrained on historical offline data and then fine-tuned through online interaction with the target environment [3, 4]. This setting, known as offline-to-online RL, allows the agent to effectively leverage prior knowledge while adapting to the target environment with fewer costly or risky online interactions.

Prior work in this direction has shown that modifying the training objective or algorithmic structure during the online phase leads to better performance than naively deploying offline RL algorithms without modification [5–7]. However, catastrophic failure often occurs, i.e., there is an early collapse in performance when the agent first transitions from offline to online. This raises a central question: how can we adapt a pretrained agent online while avoiding early collapse and maintaining strong asymptotic performance? This question is particularly critical in safety-sensitive applications such as healthcare and robotics [8, 9]. In such settings, catastrophic failure is not merely a transient performance drop; it can lead to unsafe or harmful decisions during deployment.

We study this question on robot-control benchmarks, including FrankaKitchen and AntMaze from D4RL [10] and Adroit dexterous manipulation tasks [11]. Controlled experiments show that removing offline replay can induce catastrophic failure, while retaining the conservative regularization inherited from offline RL can reduce final online performance. We further find that common stabilization strategies, including UTD-ratio tuning [7] and warmup-based online data collection [6], do not reliably prevent early collapse. We also show that the critic-side mechanisms we test can be insufficient on their own.

Motivated by these findings, we propose **ANCHOR**, a simple KL-anchored adaptation method for offline-to-online fine-tuning without offline replay. ANCHOR regularizes the online actor toward the offline-pretrained policy on online replay states during the vulnerable early phase, and gradually removes this constraint through KL annealing to enable online improvement. For the critic, ANCHOR keeps only a small residual conservative regularization term, avoiding excessive pessimism while retaining mild protection against overestimation.

We provide theoretical and empirical support for this design. Our analysis interprets KL anchoring as a mechanism for preserving critic reliability across nearby occupancy distributions, and gives a two-phase regret bound explaining when anchoring should be maintained or relaxed depending on pretrained policy quality. Empirically, ANCHOR substantially reduces catastrophic failure while achieving strong asymptotic performance, with the best average final success rate across the six tasks. Ablations show that KL annealing improves over both keeping the KL constraint fixed and removing it from the start, that the residual conservative regularization controls the stability-adaptation balance, and that ANCHOR is most beneficial when the pretrained policy contains useful behavior to preserve.

Chapter 2
Preliminaries

2.1 Conservative Q-Learning (CQL)

In offline RL, an agent is trained on a fixed dataset $\mathcal{D}$ collected by a behavior policy, without interacting with the environment. A key challenge is that standard Q-learning objectives can assign erroneously high values to actions not present in the dataset, leading to poor policy performance when deployed online. CQL [12] addresses this issue by adding a regularization term to the standard temporal-difference (TD) loss, $\mathcal{L}_{\text{TD}}$, that penalizes Q-values of actions sampled from the policy relative to those from the dataset:

$$\mathcal{L}_{\text{CQL}} = \mathcal{L}_{\text{TD}} + \alpha \left(\mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi(\cdot|s)}[Q_\theta(s,a)] - \mathbb{E}_{(s,a)\sim\mathcal{D}}[Q_\theta(s,a)] \right), \tag{2.1}$$

where $\alpha > 0$ controls the penalty strength. This discourages overly optimistic Q-values for out-of-distribution actions. We adopt CQL as the backbone of our method.

2.2 Offline-to-Online Reinforcement Learning

Offline-to-online (O2O) RL considers the setting in which an agent is first pretrained on a fixed offline dataset and then further adapted through online interaction with the target environment [5–7, 13]. We focus on the setting where the original offline trajectories are not available during online adaptation. In this case, the agent must reuse knowledge encoded in the pretrained model while learning from an online data distribution.

This setting presents two simultaneous requirements: the agent should avoid early performance degradation during the transition, while still improving beyond the offline solution through online interaction. We evaluate these two aspects using two metrics. *Catastrophic failure* measures the drop from the initial online performance to the

minimum success rate observed within the first 100K online steps; we use *early stability* to refer to low catastrophic failure. *Asymptotic performance* measures the average success rate over the final 50K online steps, corresponding to the 350K-400K window, and captures the late-stage plateau. Together, these metrics assess whether an O2O method can maintain early stability while enabling further online improvement. Further justification for these evaluation windows is provided in Appendix F.

Chapter 3
Diagnosing Failure Modes in Replay-Free Offline-to-Online RL

3.1 How Do Offline Data and Conservative Regularization Shape Online Adaptation?

We first examine the individual roles of offline data access and conservative regularization during the offline-to-online transition. To isolate their effects, we conduct controlled CQL experiments. Unless otherwise noted, all reported experiments are run with five random seeds, and we report the mean with 95% confidence intervals shown as shaded regions. Step 0 denotes the beginning of online fine-tuning. In Figure 3.1, "with offline replay" mixes offline data into online fine-tuning batches, while "without offline replay" uses only newly collected online data. Similarly, "with α" retains the CQL conservative regularizer during online fine-tuning, whereas "without α" removes it.

Our first observation is that conservative regularization can limit asymptotic online performance. To isolate the effect of the CQL regularizer, we remove offline replay in both settings and compare two variants: one that retains the conservative regularization coefficient α, and one that removes it. As shown in Figure 3.1(a), removing α achieves comparable or better final performance across all four tasks. This suggests that the pessimism induced by CQL, while useful for offline pretraining, can become restrictive once the policy must improve beyond the support of the offline dataset.

Our second observation is that removing offline data access can induce catastrophic early degradation. To isolate this effect, we compare standard CQL fine-tuning with and without offline replay. Figure 3.1(b) shows that removing offline replay often causes a sharp performance drop at the beginning of online fine-tuning. The main exception is

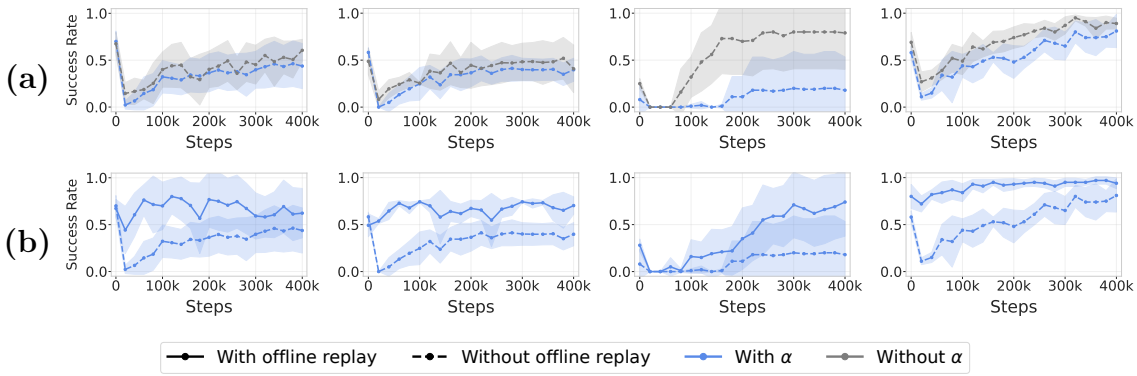


Figure 3.1: **Controlled analysis of offline data and conservative regularization.** Success rates of CQL during online fine-tuning across four tasks. (a) Keeping the conservative regularizer α during online fine-tuning can reduce asymptotic performance. (b) Removing offline replay can induce early performance degradation.

door-binary, where the initial success rate after offline pretraining is already low, making the drop less pronounced. This indicates that offline replay serves as an important stabilizing mechanism during the early online phase.

Together, these results identify two distinct factors that shape offline-to-online adaptation: offline replay stabilizes the early online phase, while strong conservative regularization can restrict asymptotic improvement. When offline replay is unavailable, we therefore need an adaptation mechanism that prevents early catastrophic failure without maintaining overly conservative regularization throughout online fine-tuning.

3.2 Can Existing Stabilization Strategies Prevent Catastrophic Failure?

Section 3.1 shows that offline-to-online fine-tuning can become unstable without offline replay. We therefore test whether existing stabilization strategies can mitigate catastrophic failure in this setting.

We first show that standard stabilization strategies fail to consistently resolve early collapse across tasks. We consider the update-to-data (UTD) ratio, which increases the number of critic updates per online interaction and has been suggested to alleviate catastrophic failure [7]. As shown in the top row of Figure 3.2, varying the UTD ratio does not consistently reduce early collapse across tasks. We next evaluate a warmup phase, where the agent collects online trajectories with the offline-pretrained policy

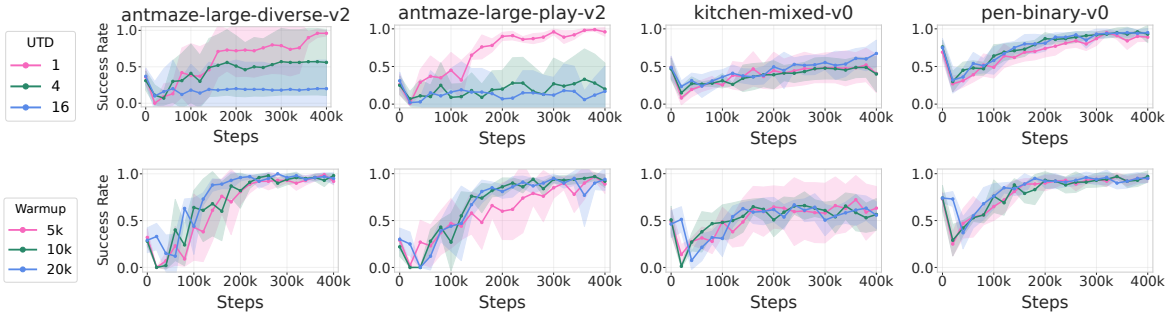


Figure 3.2: **Common stabilization strategies do not prevent catastrophic failure.** Effects of varying standard online fine-tuning choices across four tasks. *Top*: UTD ratio; *Bottom*: warmup length. Increasing UTD, or extending warmup does not reliably prevent early performance collapse.

before parameter updates, following Zhou et al. [6]. As shown in the bottom row of Figure 3.2, varying the warmup length also fails to reliably prevent catastrophic failure.

We further evaluate batch size, actor learning rate, critic learning rate, critic hidden dimension, actor hidden dimension, LayerNorm in the critic, LayerNorm in the actor, and the duration of the offline pretraining phase. Figure 3.3 shows that none of these choices consistently reduces catastrophic failure across the four tasks, reinforcing our conclusion that such collapses are not reliably resolved by existing stabilization strategies.

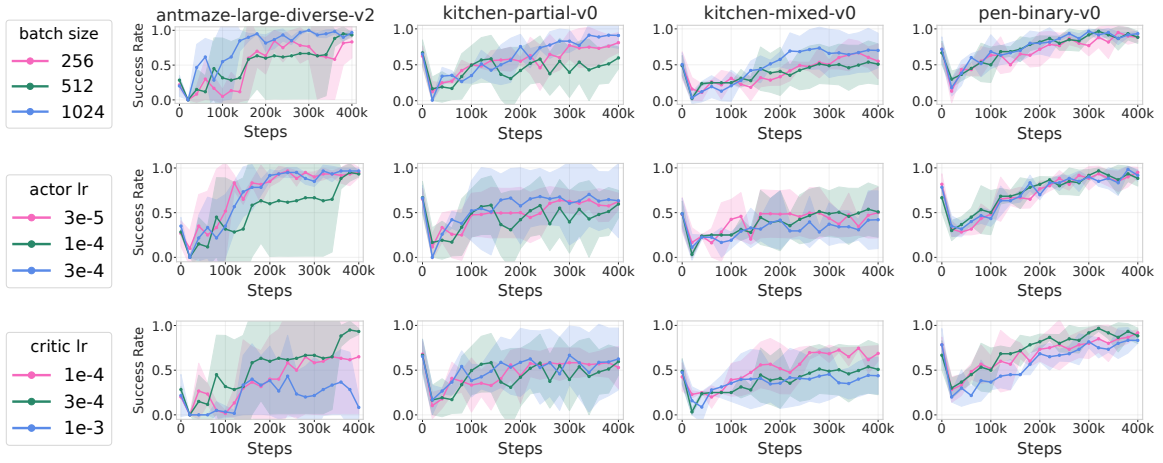


Figure 3.3: **Hyperparameter and architectural tweaks do not consistently mitigate catastrophic failure.** From top to bottom, the plots correspond to: batch size, actor learning rate, critic learning rate, critic hidden dimension, actor hidden dimension, LayerNorm in the critic, LayerNorm in the actor, and the duration of the offline pretraining phase. Across all settings and tasks, none of these factors consistently mitigates catastrophic failure.

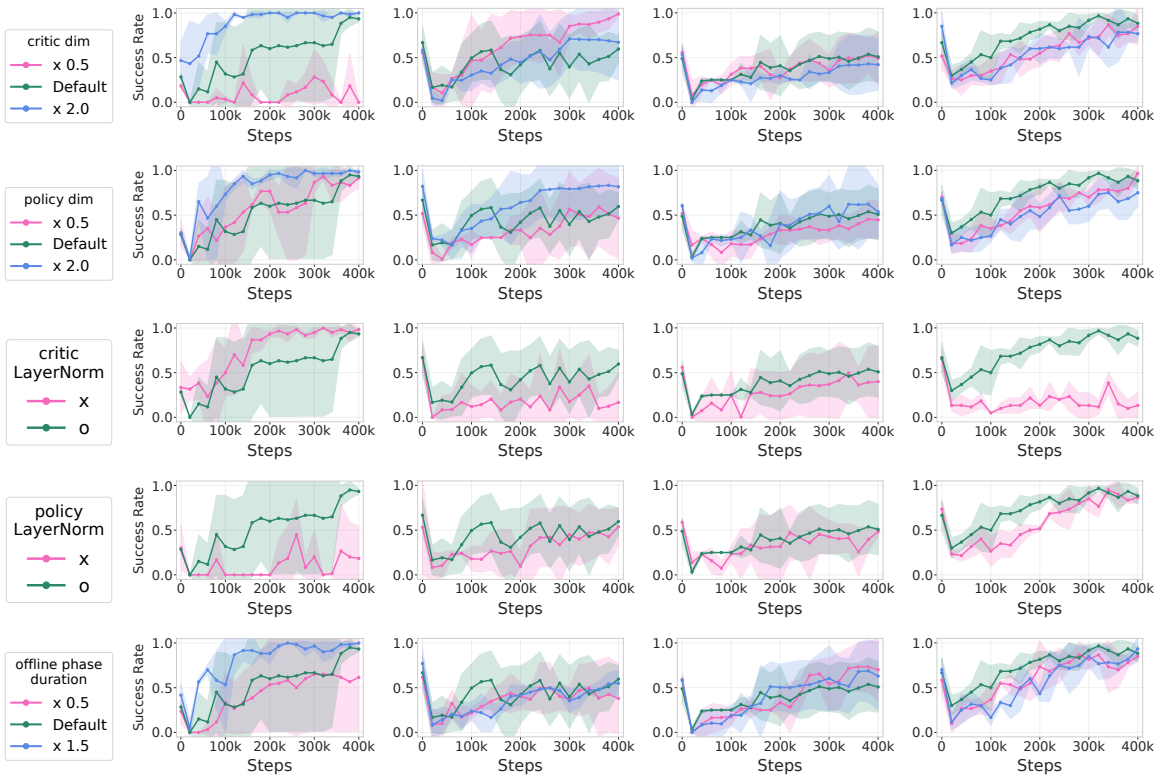


Figure 3.3: **Hyperparameter and architectural tweaks do not consistently mitigate catastrophic failure.** (continued)

3.3 Can Critic-Side Stabilization Prevent Catastrophic Failure?

We next test whether online calibration of the conservatively pretrained Q-function can stabilize fine-tuning. Cal-QL provides a natural testbed for this question, as it modifies the CQL conservative penalty to calibrate the Q-function using an online evidence-based reference value [5]. The coefficient $\alpha_{\text{Cal-QL}}$ controls the strength of this calibrated conservative regularization. As shown in Figure 3.4, tuning $\alpha_{\text{Cal-QL}}$ can reduce early collapse in antmaze-large-diverse and antmaze-large-play, but the effect is not consistent across tasks or hyperparameter values. Moreover, Appendix F shows that ANCHOR achieves lower catastrophic failure than Cal-QL on average, suggesting that there remains room for improvement.

The limited effectiveness of Cal-QL raises the question of whether critic-side stabilization alone is sufficient. Motivated by the hypothesis of Zhou et al. [6] that rapid critic shift contributes to early collapse, we test this question by explicitly penalizing

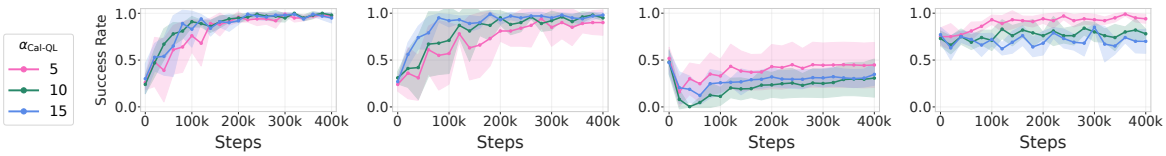


Figure 3.4: **Cal-QL does not consistently prevent catastrophic failure.** Tuning $\alpha_{\text{Cal-QL}}$ does not reliably prevent early performance collapse.

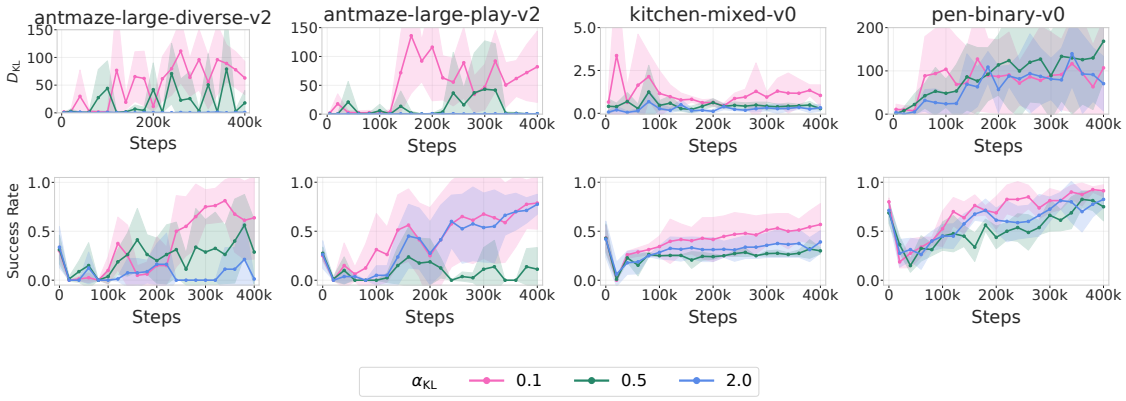


Figure 3.5: **Critic-shift regularization does not prevent catastrophic failure.** Effects of varying the critic-shift penalty strength α_{KL} across four tasks. *Top*: critic distribution shift during online fine-tuning, measured by $D_{\text{KL}}\big(\text{softmax}(Q_{\text{off}}) \,\|\, \text{softmax}(Q_\theta)\big)$. *Bottom*: corresponding success rates. Increasing α_{KL} slows the critic shift, but does not reliably prevent early performance collapse.

deviation from the offline-pretrained critic:

$$\min_\theta \; \mathcal{L}_{\text{TD}}(\theta) + \alpha_{\text{KL}}\, \mathbb{E}_{(s,a)\sim\mathcal{B}} \left[D_{\text{KL}} \left(\text{softmax}\big(Q_{\text{offline}}(s,\cdot)\big) \,\|\, \text{softmax}\big(Q_{\text{online}}(s,\cdot)\big) \right) \right],$$

where $\mathcal{B}$ is the online replay buffer, D_{KL} denotes the KL divergence, Q_{offline} is the offline-pretrained critic, and Q_{online} is the critic being updated during online fine-tuning. The coefficient $\alpha_{\text{KL}} > 0$ controls the strength of this critic-shift penalty. As shown in Figure 3.5, increasing α_{KL} slows critic distribution shift, yet the corresponding success curves show that catastrophic failure persists.

In response to the limited effect of critic-side stabilization, we further examine whether Q-driven diagnostics can explain or predict early collapse. We evaluate four tasks: antmaze-large-diverse, kitchen-partial, kitchen-mixed, and pen-binary. Since catastrophic failure typically occurs early in online fine-tuning, we report all metrics over the first 30K online steps. Due to computational constraints, this analysis is run

with three random seeds.

For a minibatch of states s, let $a_t \sim \pi_t(\cdot|s)$ denote an action sampled from the current online policy and let $a_{\text{off}} \sim \pi_{\text{off}}(\cdot|s)$ denote an action sampled from the offline-pretrained policy. Let Q_t denote the critic at online step t, and let Q_{final} denote the critic checkpoint after 400K online steps for the same random seed. We define

$$\Delta_t(s) = Q_t(s, a_t) - Q_t(s, a_{\text{off}}), \qquad \Delta_{\text{final}}(s) = Q_{\text{final}}(s, a_t) - Q_{\text{final}}(s, a_{\text{off}}).$$

Intuitively, Δ_t measures the current critic's relative preference between the online-policy action and the offline-policy action, while Δ_{final} provides a post-hoc reference after online fine-tuning has stabilized.

Spurious Q-optimism ratio and mismatch magnitude. We first examine two preference-based Q diagnostics. Let $\mathcal{B}_t$ denote the minibatch of states sampled from the online replay buffer at step t, and let $\text{sgn}(\cdot)$ denote the sign function. The first metric is the spurious Q-optimism ratio:

$$\text{SQOR}(t) = \frac{1}{|\mathcal{B}_t|} \sum_{s \in \mathcal{B}_t} \mathbf{1}\left[\Delta_t(s) > 0 \ \wedge \ \Delta_{\text{final}}(s) < 0\right].$$

SQOR measures the fraction of batch states where the current critic favors the current-policy action, while the final critic favors the offline-policy action. The second metric measures the magnitude of preference mismatch. Define the set of mismatched states within the current online batch as

$$\mathcal{M}_t = \{s \in \mathcal{B}_t : \text{sgn}(\Delta_t(s)) \neq \text{sgn}(\Delta_{\text{final}}(s))\}.$$

We define the spurious Q-optimism gap as

$$\text{SQOG}(t) = \frac{1}{|\mathcal{M}_t|} \sum_{s \in \mathcal{M}_t} \left|\Delta_t(s) - \Delta_{\text{final}}(s)\right|,$$

with the value set to zero when $\mathcal{M}_t$ is empty.

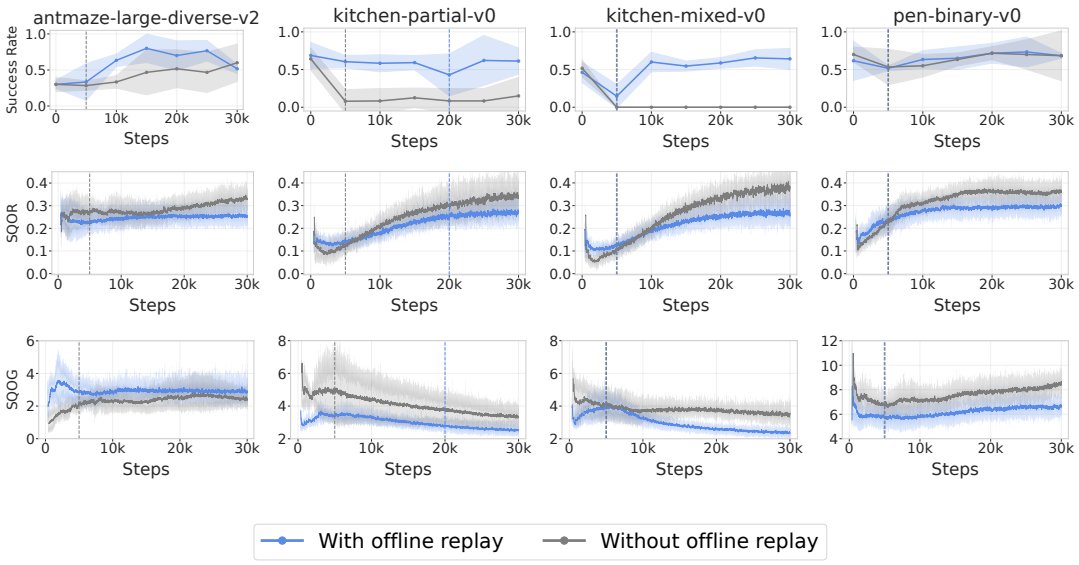


Figure 3.6: **SQOR and SQOG do not reliably predict catastrophic failure.** We compare success rate, the spurious Q-optimism ratio (SQOR), and the spurious Q-optimism gap (SQOG) across four tasks. Both metrics show task-dependent and inconsistent alignment with early performance collapse. The vertical dashed line marks the timestep at which the largest performance drop occurs.

As shown in Figure 3.6, neither SQOR nor SQOG consistently tracks catastrophic failure across tasks and stress settings. In some cases, the metrics increase without a corresponding collapse; in others, early performance drops are not preceded by a clear rise in either metric. This suggests that these preference-based Q diagnostics alone do not provide a reliable explanation for catastrophic failure.

Q-update volatility. We also examine whether abrupt critic updates explain early collapse. We define volatility as the square root of the bias-corrected exponential moving average of squared one-step Q-updates [14]:

$$\text{Volatility}_t = \sqrt{\frac{m_t}{1-\beta^t}}, \qquad m_t = \beta m_{t-1} + (1-\beta)\mathbb{E}_{(s,a)\sim\mathcal{B}_t}\left[(\Delta Q_t(s,a))^2\right],$$

where $\Delta Q_t(s,a) = Q_{t+1}(s,a) - Q_t(s,a)$ and $\beta = 0.9$. This metric measures the aggregate magnitude of critic changes over minibatch state-action pairs.

Figure 3.7 shows that Q-update volatility also fails to consistently align with catastrophic failure. High volatility does not always lead to collapse, and severe collapse can

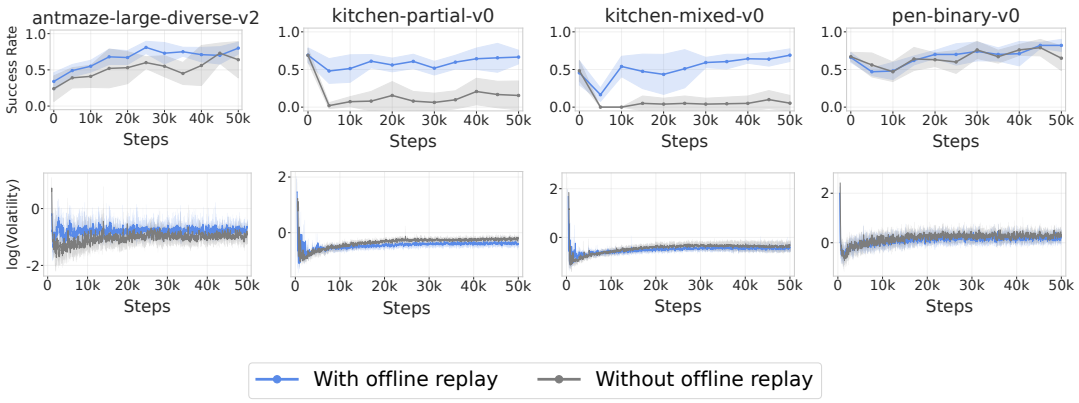


Figure 3.7: **Q-update volatility does not reliably predict catastrophic failure.** We compare success rate and critic-update volatility across four tasks. Volatility shows weak and inconsistent correspondence with early performance collapse.

occur without a distinctive volatility pattern. Together with the critic-shift regularization results above, these analyses do not rule out critic instability as a contributing factor. Rather, they suggest that critic-side stabilization alone may be insufficient to explain or prevent catastrophic failure, motivating our focus on policy-side stabilization through KL anchoring.

Chapter 4
Method

Building on Sections 3.1 and 3.2, we propose ANCHOR, a KL-anchored adaptation scheme that mitigates catastrophic failure while preserving high asymptotic performance during offline-to-online adaptation. We also provide a theoretical interpretation of ANCHOR.

4.1 ANCHOR: KL-anchored checkpoint adaptation

Let π_{off} denote the actor obtained after offline pretraining, and let π_t and Q_t denote the online actor and critic at online step t. ANCHOR regularizes the actor toward π_{off} during the early adaptation phase and gradually relaxes this constraint. Concretely, we use the actor objective

$$\mathcal{J}_{\pi,t}^{\text{ANCHOR}}(\phi) = \mathbb{E}_{s\sim\mathcal{D}_{\text{on}},\, a\sim\pi_\phi(\cdot|s)}\left[Q_t(s,a)\right] - \beta_t\, \mathbb{E}_{s\sim\mathcal{D}_{\text{on}}}\left[D_{\text{KL}}\big(\pi_\phi(\cdot|s)\,\|\,\pi_{\text{off}}(\cdot|s)\big)\right], \quad (4.1)$$

where $\beta_t \geq 0$ is the KL regularization weight.

The critic is updated with the usual online TD loss together with a conservative CQL term whose coefficient is fixed to α_{on} during online fine-tuning:

$$\mathcal{L}_Q^{\text{ANCHOR}}(\theta) = \mathcal{L}_{\text{TD}}(\theta;\mathcal{D}_{\text{on}}) + \alpha_{\text{on}}\Big(\mathbb{E}_{s\sim\mathcal{D}_{\text{on}},\, a\sim\pi_t(\cdot|s)}[Q_\theta(s,a)] - \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{on}}}[Q_\theta(s,a)]\Big). \quad (4.2)$$

Here, $\alpha_{\text{on}} \geq 0$ denotes the residual conservative regularization strength used throughout online fine-tuning, which is set smaller than the coefficient used during offline pretraining. This design avoids excessive conservatism that may limit online improvement, while retaining a small amount of critic regularization to prevent abrupt critic distribution shift and stabilize adaptation.

For intuition, consider a fixed state s and an unconstrained policy class over a

continuous action space $\mathcal{A}$. Assuming policies are absolutely continuous with respect to π_{off} and the normalizer is finite, the state-wise objective in Equation 4.1 admits the closed-form solution [15]

$$\pi_t^\star(a|s) = \frac{\pi_{\text{off}}(a|s)\exp(Q_t(s,a)/\beta_t)}{Z_t(s)}, \qquad Z_t(s) = \int_{\mathcal{A}} \pi_{\text{off}}(a'|s)\exp(Q_t(s,a')/\beta_t)\ da'. \tag{4.3}$$

Thus, β_t controls the strength of anchoring: as $\beta_t \to 0$, the ideal policy concentrates on high-Q_t actions, while large β_t keeps it close to π_{off}. In our continuous-control experiments, we do not compute this nonparametric solution explicitly; instead, we optimize the actor using Equation 4.1.

In practice, ANCHOR uses a linear KL annealing schedule:

$$\beta_t = \beta_0 \cdot \max\left(1 - \frac{t}{T_{\text{KL}}},\, 0\right),$$

where β_0 is the initial KL regularization weight and T_{KL} is the KL annealing interval. Thus, β_t decreases linearly from β_0 to 0 during the first T_{KL} online steps. After $t \geq T_{\text{KL}}$, the KL anchoring term is completely removed and the actor is optimized using the standard online objective. Exact hyperparameter settings are reported in Appendix C. We detail the difference from trust-region methods such as TRPO [16] in Appendix A.

4.2 Theoretical Interpretation

We provide a formal interpretation of ANCHOR through two supporting lemmas. The first lemma shows that KL anchoring can control the distribution-shift component of critic instability. The second lemma gives a two-phase regret bound that quantifies the cost of using KL anchoring during the early online phase before switching to the unanchored online learner.

Assumption 1 (Local occupancy control around the offline policy)**.** *There exists a constant $C_{\text{occ}} > 0$ such that, for every policy π in the anchored phase,*

$$\|d^\pi - d^{\pi_{\text{off}}}\|_1 \leq C_{\text{occ}}\, \text{TV}_{d^{\pi_{\text{off}}}}(\pi, \pi_{\text{off}}).$$

Assumption 1 is a local smoothness condition on the policy-to-occupancy map around π_{off}: within the anchored region, small policy deviations lead to proportionally controlled occupancy shifts. It excludes pathological cases in which a change on states with negligible offline visitation produces a large downstream change in the trajectory distribution. Since ANCHOR applies KL anchoring precisely during the early phase to keep the policy close to π_{off} on sampled states, the lemma is intended to characterize this local anchored regime rather than arbitrary policies far from the offline reference.

Notation. We consider an infinite-horizon discounted MDP with discount factor $\gamma \in (0, 1)$, finite action set $\mathcal{A}$, reward function $r(s, a)$, and normalized discounted state-action occupancy

$$d^{\pi}(s, a) := (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k \Pr_{\pi}(s_k = s, a_k = a).$$

With this normalization, d^{π} is a probability distribution over (s, a) and

$$J(\pi) = \frac{1}{1 - \gamma} \mathbb{E}_{(s,a)\sim d^{\pi}}[r(s, a)].$$

We also write $A^{\mu}(s, a) := Q^{\mu}(s, a) - V^{\mu}(s)$.

Performance difference lemma. We will repeatedly use the standard identity

$$J(\pi) - J(\mu) = \frac{1}{1 - \gamma} \mathbb{E}_{(s,a)\sim d^{\pi}}[A^{\mu}(s, a)]\,. \tag{4.4}$$

This is the classical performance difference lemma.

Lemma 4.1 (Critic error difference under KL-occupancy control)**.** *Let $\widehat{Q}_t$ denote the learned critic at online step t, and let Q^{π} denote the true action-value function of an arbitrary target policy π. Define*

$$e_t(s, a) = |\widehat{Q}_t(s, a) - Q^{\pi}(s, a)|.$$

If $|\widehat{Q}_t(s,a)| \le Q_{\max}$ *and* $|Q^\pi(s,a)| \le Q_{\max}$*, then for any two occupancy measures* d *and* d'*,*

$$|\mathbb{E}_d[e_t] - \mathbb{E}_{d'}[e_t]| \le 2Q_{\max}\|d - d'\|_1.$$

If the online actor additionally satisfies a KL constraint around the offline policy,

$$D_{d^{\pi_{\text{off}}}}(\pi_t \| \pi_{\text{off}}) \le \delta_t,$$

and the occupancy shift is locally controlled by policy divergence, then

$$|\mathbb{E}_{d^{\pi_t}}[e_t] - \mathbb{E}_{d^{\pi_{\text{off}}}}[e_t]| = O(\sqrt{\delta_t}).$$

Proof of Lemma 4.1. By the boundedness assumptions and the triangle inequality,

$$e_t(s,a) = |\widehat{Q}_t(s,a) - Q^\pi(s,a)| \le |\widehat{Q}_t(s,a)| + |Q^\pi(s,a)| \le 2Q_{\max}.$$

Thus, for any two occupancy measures d and d', we have

$$\mathbb{E}_d[e_t] - \mathbb{E}_{d'}[e_t] = \sum_{s,a} \big(d(s,a) - d'(s,a)\big) e_t(s,a).$$

Taking absolute values and applying Hölder's inequality,

$$|\mathbb{E}_d[e_t] - \mathbb{E}_{d'}[e_t]| \le \|e_t\|_\infty \|d - d'\|_1 \le 2Q_{\max}\|d - d'\|_1.$$

This proves the first bound. Now suppose

$$D_{d^{\pi_{\text{off}}}}(\pi_t \| \pi_{\text{off}}) \le \delta_t.$$

By Pinsker's inequality, for each state s,

$$\text{TV}(\pi_t(\cdot|s), \pi_{\text{off}}(\cdot|s)) \le \sqrt{\frac{1}{2} D_{\text{KL}}(\pi_t(\cdot|s) \| \pi_{\text{off}}(\cdot|s))}.$$

Taking expectation over $s \sim d^{\pi_{\text{off}}}$ and applying Jensen's inequality,

$$\mathrm{TV}_{d^{\pi_{\text{off}}}}(\pi_t, \pi_{\text{off}}) \le \sqrt{\frac{1}{2} D_{d^{\pi_{\text{off}}}}(\pi_t \| \pi_{\text{off}})} \le \sqrt{\delta_t/2}.$$

Combining this with Assumption 1 gives

$$\|d^{\pi_t} - d^{\pi_{\text{off}}}\|_1 \le C_{\text{occ}} \sqrt{\delta_t/2}.$$

Finally, applying the first bound with $d = d^{\pi_t}$ and $d' = d^{\pi_{\text{off}}}$ yields the stated result. □

Lemma 4.1 is a distribution-transfer bound for critic error. It does not assume that the critic error is small; instead, it shows that if the critic is accurate under the offline-policy occupancy, then its error under the current online-policy occupancy remains controlled when π_t stays close to π_{off}. Thus, if the pretrained critic is reliable on the offline data distribution and this reliability is preserved during early online updates, KL anchoring prevents the online policy from moving too quickly into regions where critic error may be much larger.

In practice, ANCHOR computes the KL penalty on online replay states rather than under the offline-policy occupancy used in the lemma. Although these distributions differ, the replay-buffer KL serves the same local role: it constrains policy updates on states encountered during fine-tuning and thereby limits abrupt policy drift.

Lemma 4.2 (Two-phase regret decomposition)**.** *Over $T = T_1 + T_2$ steps, suppose phase I uses KL anchoring around π_{off} with budgets δ_t. In phase II, suppose the actor-side KL anchor is removed, and the resulting online learner has regret $\mathrm{Reg}_{\text{base}}(T_2)$. Then under bounded advantage and local occupancy control,*

$$\mathrm{Reg}_T \le T_1 \Delta_{\text{off}} + C \sum_{t=1}^{T_1} \sqrt{\delta_t} + \mathrm{Reg}_{\text{base}}(T_2),$$

where $\Delta_{\text{off}} = J(\pi^*) - J(\pi_{\text{off}})$ and C is a problem-dependent constant.

Proof of Lemma 4.2. By definition,

$$\mathrm{Reg}_T = \sum_{t=1}^{T} \big(J(\pi^*) - J(\pi_t)\big) = \sum_{t=1}^{T_1} \big(J(\pi^*) - J(\pi_t)\big) + \sum_{t=T_1+1}^{T} \big(J(\pi^*) - J(\pi_t)\big).$$

For phase I, add and subtract $J(\pi_{\mathrm{off}})$:

$$J(\pi^*) - J(\pi_t) = \Delta_{\mathrm{off}} + \big(J(\pi_{\mathrm{off}}) - J(\pi_t)\big).$$

Thus it remains to bound $|J(\pi_t) - J(\pi_{\mathrm{off}})|$.

By the performance difference lemma (4.4),

$$J(\pi_t) - J(\pi_{\mathrm{off}}) = \frac{1}{1-\gamma} \mathbb{E}_{(s,a)\sim d^{\pi_t}} \left[A^{\pi_{\mathrm{off}}}(s,a) \right].$$

Also,

$$\mathbb{E}_{(s,a)\sim d^{\pi_{\mathrm{off}}}} [A^{\pi_{\mathrm{off}}}(s,a)] = 0,$$

because

$$\sum_a \pi_{\mathrm{off}}(a|s) A^{\pi_{\mathrm{off}}}(s,a) = 0 \qquad \text{for every state } s.$$

Therefore,

$$J(\pi_t) - J(\pi_{\mathrm{off}}) = \frac{1}{1-\gamma} \sum_{s,a} \big(d^{\pi_t}(s,a) - d^{\pi_{\mathrm{off}}}(s,a)\big) A^{\pi_{\mathrm{off}}}(s,a).$$

Using $|A^{\pi_{\mathrm{off}}}(s,a)| \le A_{\max}$,

$$|J(\pi_t) - J(\pi_{\mathrm{off}})| \le \frac{A_{\max}}{1-\gamma} \, \|d^{\pi_t} - d^{\pi_{\mathrm{off}}}\|_1.$$

By Assumption 1 and Pinsker,

$$\|d^{\pi_t} - d^{\pi_{\mathrm{off}}}\|_1 \le C_{\mathrm{occ}} \, \mathrm{TV}_{d^{\pi_{\mathrm{off}}}}(\pi_t, \pi_{\mathrm{off}}) \le C_{\mathrm{occ}} \sqrt{\delta_t/2}.$$

Hence

$$J(\pi^*) - J(\pi_t) \le \Delta_{\text{off}} + \frac{A_{\max} C_{\text{occ}}}{1-\gamma} \sqrt{\delta_t/2}.$$

Summing over $t = 1, \ldots, T_1$ yields

$$\sum_{t=1}^{T_1} \big(J(\pi^*) - J(\pi_t)\big) \le T_1 \Delta_{\text{off}} + \frac{A_{\max} C_{\text{occ}}}{1-\gamma} \sum_{t=1}^{T_1} \sqrt{\delta_t/2}.$$

For phase II, apply the assumed regret guarantee of the base learner:

$$\sum_{t=T_1+1}^{T} \big(J(\pi^*) - J(\pi_t)\big) \le \text{Reg}_{\text{base}}(T_2).$$

Combining the two parts proves the stated decomposition. □

Lemma 4.2 separates the regret into the cost of staying near the offline policy, the cost of controlled policy deviation during the anchored phase, and the regret after the anchor is removed. This decomposition clarifies how the usefulness of KL anchoring depends on the quality of the pretrained policy. When π_{off} is weak, Δ_{off} is large, so remaining close to the offline policy for too long can incur substantial regret; in this case, annealing the KL coefficient is important for allowing the actor to move beyond the pretrained solution. When π_{off} is strong, Δ_{off} is small, so the cost of staying near the offline policy is limited, and stronger or longer anchoring can be justified if it prevents unstable online updates.

Overall, the analysis supports the central design of ANCHOR: KL anchoring limits early distribution shift and helps preserve critic reliability near the pretrained policy, while annealing removes this constraint when online improvement beyond the pretrained solution is needed. The results should be viewed as an interpretation of the stabilizing role of anchoring, rather than a convergence guarantee.

Chapter 5
Experiments

5.1 Experimental Setup

Environments and tasks. We evaluate ANCHOR on three widely used offline-to-online RL benchmarks: FrankaKitchen and AntMaze from D4RL [10], and Adroit dexterous manipulation tasks [11]. Our main benchmark consists of six tasks: kitchen-mixed, kitchen-partial, antmaze-large-diverse, antmaze-large-play, door-binary, and pen-binary. All offline pretraining datasets match those used in WSRL [6]. Further environment details and task visualizations are provided in Appendices C and E.

Training procedure. We first perform offline pretraining for 1M steps in AntMaze, 250K steps in FrankaKitchen, and 40K steps in Adroit, followed by 400K steps of online fine-tuning. All methods are trained without offline data replay and use the same online interaction budget.

Baselines. We compare ANCHOR against WSRL [6], PORL [7], Cal-QL [5], CQL [12], IQL [17], and SAC [18]. Detailed descriptions of the baselines are provided in Appendix B.

For baseline reproduction, we largely follow the experimental setup of WSRL [6]. Any modifications made to stabilize training are described in Appendix C.3. ANCHOR is implemented on top of CQL, with modifications to the online training objective. Details of the ANCHOR implementation and hyperparameters are provided in Appendix C.

5.2 ANCHOR Improves Both Early Stability and Final Performance

Figure 5.1 summarizes the main benchmark results. Across the six-task suite, ANCHOR achieves the best average final performance while substantially reducing catastrophic failure. In door-binary, where the offline-pretrained policy starts from relatively low performance, the benefit of preventing catastrophic failure is less pronounced;

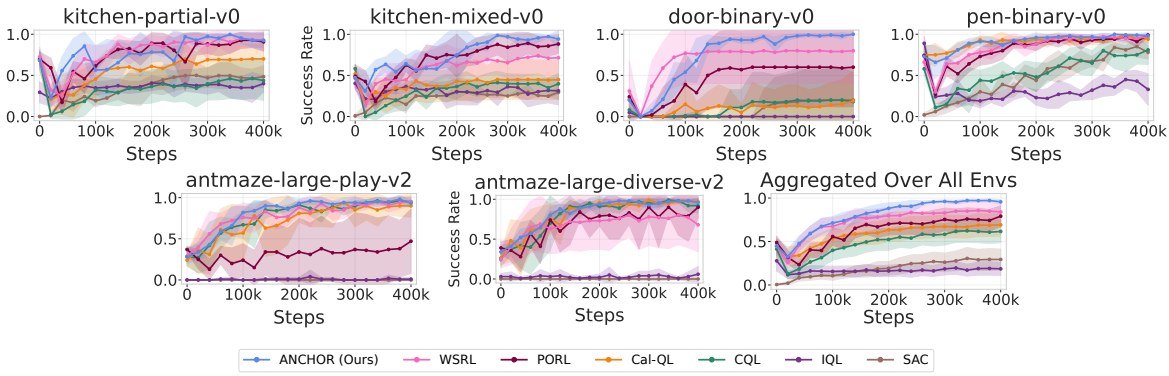


Figure 5.1: **ANCHOR improves both early stability and final performance.** Existing methods do not consistently achieve both low catastrophic failure and strong asymptotic performance across tasks. In contrast, ANCHOR reduces early performance collapse while attaining strong final performance across tasks. The aggregate curve reports the mean success rate averaged over the six tasks.

nevertheless, ANCHOR achieves the best asymptotic performance.

Existing baselines often fail to achieve both early stability and strong final performance. Cal-QL, which retains stronger conservative regularization, can reduce early collapse but often adapts more slowly, whereas more aggressive online adaptation methods, including WSRL and PORL, can suffer larger early performance drops. Full numerical results, reported as mean ± standard error, are provided in Appendix F.

5.3 Additional Environments and Failure Cases

We further evaluate ANCHOR in three additional environments that highlight challenging regimes for offline-to-online adaptation: kitchen-complete, relocate-binary, and antmaze-ultra-diverse. The first two environments have near-zero initial success after offline pretraining, leaving little useful pretrained behavior to preserve. In contrast, antmaze-ultra-diverse is a larger and more complex maze than antmaze-large-diverse, where sparse rewards and severe exploration mismatch make online improvement difficult. The offline pretraining duration follows the domain-specific setup used in the main experiments: 250K steps for kitchen-complete, 40K steps for relocate-binary, and 1M steps for antmaze-ultra-diverse.

For all three environments, we set the residual conservative regularization coefficient

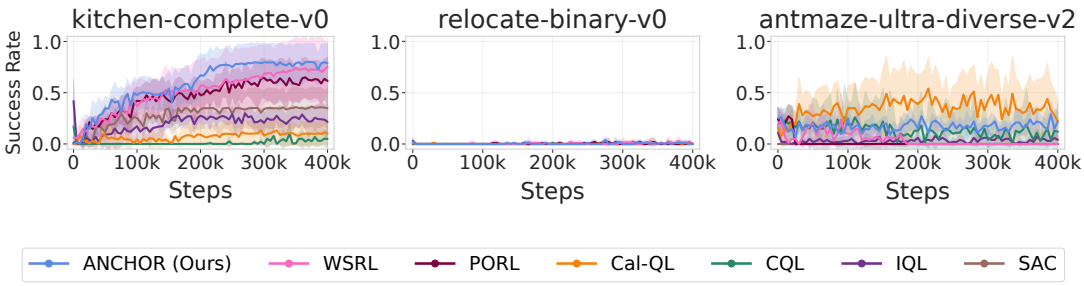


Figure 5.2: **Additional environments reveal limitations of KL-anchored adaptation.** In kitchen-complete, ANCHOR remains competitive despite near-zero initial success. In relocate-binary, all methods fail to make meaningful progress. In antmaze-ultra-diverse, most methods do not improve over the offline-pretrained policy, suggesting that harder sparse-reward exploration requires mechanisms beyond KL anchoring alone.

α_{on} to zero during online fine-tuning. Since kitchen-complete and relocate-binary start with near-zero success rates, we set the initial KL weight to $\beta_0 = 0$, reflecting the fact that there is little useful pretrained behavior to preserve. In kitchen-complete, ANCHOR attains the highest mean final performance and matches WSRL and PORL when confidence intervals are considered, while outperforming the remaining baselines. This behavior is consistent with Lemma 4.2: when the pretrained policy is weak, keeping the actor close to the offline policy can be costly, and the KL constraint should be relaxed or removed to enable online improvement. In relocate-binary, however, all methods remain near zero success throughout online fine-tuning, suggesting that this environment requires additional exploration or adaptation mechanisms beyond the scope of ANCHOR.

For antmaze-ultra-diverse, we use $\beta_0 = 1.0$ and set the KL annealing interval to $T_{\text{KL}} = 400\text{K}$, covering the full online fine-tuning phase. Despite this longer anchoring schedule, ANCHOR's final asymptotic performance remains below its initial offline-pretrained performance. Except for Cal-QL, none of the methods achieve meaningful improvement over the offline-pretrained success rate. This suggests that in extremely challenging sparse-reward environments, preserving the pretrained policy alone may be insufficient; the agent may also require exploration or online adaptation mechanisms.

To further test whether stronger preservation of offline-pretrained behavior helps

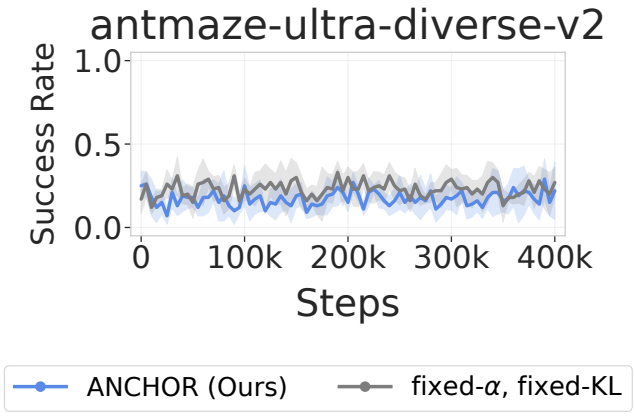


Figure 5.3: **Stronger preservation is insufficient in antmaze-ultra-diverse.** Keeping both the conservative coefficient and KL coefficient fixed yields a higher mean success rate than ANCHOR during online fine-tuning, but does not significantly improve asymptotic performance.

in antmaze-ultra-diverse, we evaluate an additional variant that keeps the conservative coefficient at its offline-pretraining value, $\alpha = 1.0$, and keeps the KL coefficient fixed at $\beta_t = 1.0$ without annealing. As shown in Figure 5.3, this variant achieves a higher mean success rate than ANCHOR across much of online fine-tuning, but its asymptotic performance does not improve significantly. These results indicate that, in extremely challenging exploration settings, simply retaining conservative regularization or maintaining a fixed KL anchor is not sufficient. Developing mechanisms for online exploration and adaptation in such environments remains an important direction for future work.

5.4 Applying KL Anchoring to IQL

We additionally test whether the actor-side KL anchoring used in ANCHOR can be applied to another offline RL backbone. Specifically, we apply KL anchoring with annealing to IQL. Since IQL does not use an explicit CQL-style conservative regularizer, we only add the actor-side KL term toward the offline-pretrained policy and anneal its coefficient during online fine-tuning. We conduct experiments on kitchen-mixed and pen-binary, where IQL exhibits nontrivial pretrained behavior with an initial success rate above 0.4. For both tasks, we use $\beta_0 = 5.0$ and $T_{\mathrm{KL}} = 320\mathrm{K}$.

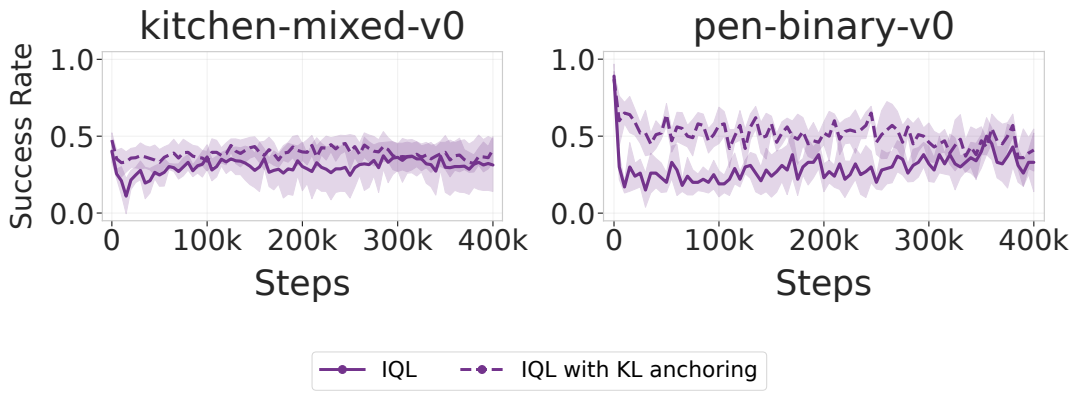


Figure 5.4: **Applying KL anchoring to IQL.** Adding KL anchoring with annealing to IQL reduces the initial performance drop, but does not improve asymptotic performance over the IQL baseline.

Figure 5.4 shows that KL anchoring can also stabilize IQL during the early online phase. Compared to naive online fine-tuning of IQL, the KL-anchored variant reduces the initial performance drop, indicating that actor-side anchoring is not specific to the CQL backbone. However, the asymptotic performance does not improve substantially over the IQL baseline. We hypothesize that this is because IQL retains its offline inductive bias during online fine-tuning, which can limit its ability to adapt to the online data distribution even when the actor is stabilized by KL anchoring. Developing backbone-specific mechanisms that remove or relax such offline biases while preserving early stability is an interesting direction for future work.

5.5 Analysis of ANCHOR

Annealing the KL constraint balances early anchoring and later adaptation. We first ask whether the KL constraint should be kept throughout online fine-tuning, removed from the beginning, or annealed over time. To answer this, we compare ANCHOR against two alternatives: fixed-KL, which keeps the initial KL weight throughout online fine-tuning without annealing, and no-KL, which does not use KL regularization.

As shown in Figure 5.5, ANCHOR most consistently achieves both low catastrophic failure and strong final performance across the three tasks. The fixed-KL variant

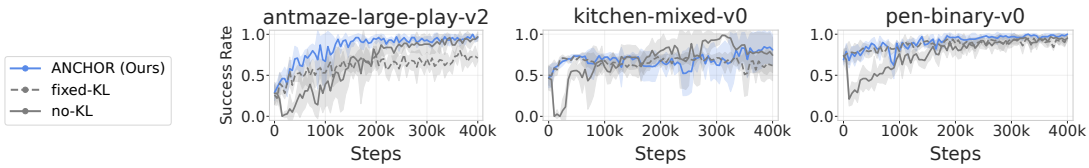


Figure 5.5: **KL annealing improves early stability while preserving final performance.** ANCHOR is compared with the fixed-KL and no-KL variants. The fixed-KL variant improves early stability but can limit final improvement, while the no-KL variant adapts aggressively but causes early collapse.

demonstrates the benefit of anchoring: keeping the actor close to the pretrained policy can suppress early collapse. However, because this constraint is never annealed, it can remain overly restrictive and reduce asymptotic performance, especially when the offline-pretrained policy is not already strong, as in antmaze-large-play and kitchen-mixed. Conversely, the no-KL variant removes this restriction entirely and allows more aggressive online adaptation, but it exhibits large catastrophic failure across all three tasks because the policy can drift away from the pretrained behavior too abruptly. These patterns justify the annealing design of ANCHOR: KL annealing provides a middle path by using strong anchoring when online fine-tuning is most vulnerable, and then relaxing the constraint as the online policy and critic adapt.

The benefit of ANCHOR depends on the quality of the pretrained policy. We next examine when ANCHOR is most beneficial by varying the quality of the offline-pretrained policy. We compare ANCHOR with the no-KL variant across pretrained checkpoints with different initial success rates.

Figure 5.6 shows that the effect of ANCHOR depends on the quality of the offline-pretrained policy. In both pen-binary and kitchen-mixed, when the pretrained policy starts with a low initial success rate, the difference in catastrophic failure between ANCHOR and the no-KL variant is less pronounced. In contrast, when the pretrained policy starts from a stronger initial success rate, the gap becomes much clearer: ANCHOR substantially reduces catastrophic failure relative to the no-KL variant.

Importantly, even in the low-initial-success regime, ANCHOR remains a reasonable design choice. Compared to the no-KL variant, it still achieves strong asymptotic performance in pen-binary and comparable asymptotic performance in kitchen-mixed.

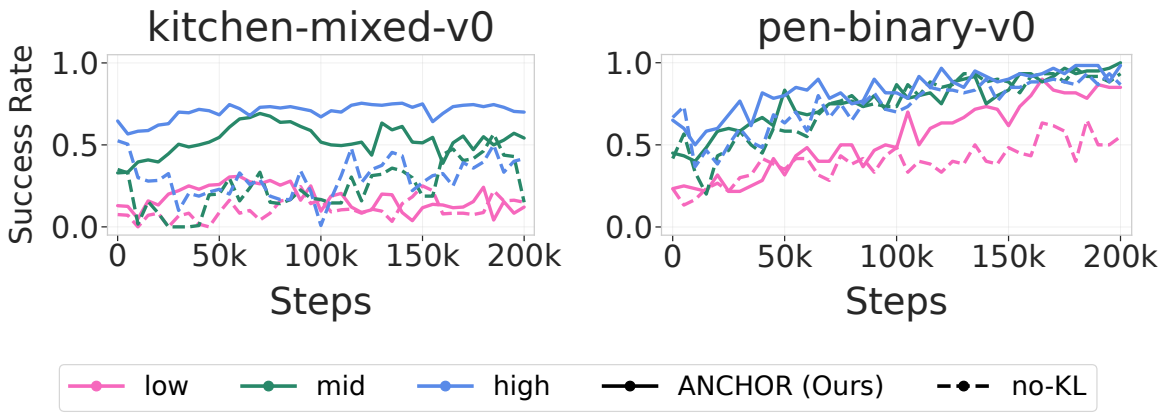


Figure 5.6: **Benefit of ANCHOR depends on pretrained policy quality.** ANCHOR reduces catastrophic failure more clearly when the offline-pretrained policy starts from a higher initial success rate. Curves with the same color correspond to the same pretrained policy. Results are averaged over three seeds.

Taken together, these results indicate that the main value of ANCHOR lies in protecting useful pretrained behavior when it exists, while not substantially hindering final online performance even when the pretrained policy is less reliable. Results with full confidence intervals are provided in Appendix F.

How sensitive is ANCHOR to its hyperparameters? We finally study the sensitivity of ANCHOR to three hyperparameters: the initial KL weight β_0, the KL annealing interval T_{KL}, and the residual online conservative regularization strength α_{on}.

Figure 5.7 shows that ANCHOR is robust to moderate changes in its KL-related hyperparameters. Although different choices of β_0 and T_{KL} can affect learning stability, all tested variants continue to reduce catastrophic failure across the three tasks and achieve similar asymptotic performance within confidence intervals. Thus, ANCHOR's gains are not driven by a single carefully tuned configuration, but reflect a robust effect of KL-anchored adaptation.

The residual conservative weight α_{on} controls a different aspect of the method. As shown in Figure 5.7, we compare three choices: removing the conservative penalty, using an intermediate residual value between zero and the offline-pretraining weight, and retaining the full offline-pretraining weight. Retaining the full offline conservative weight leads to the lowest asymptotic performance in antmaze-large-diverse and kitchen-

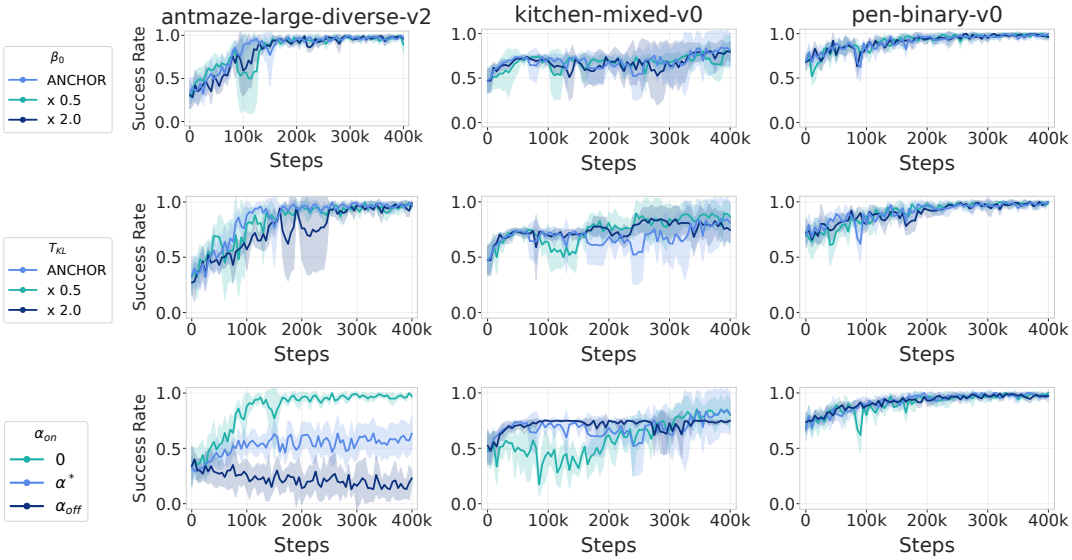


Figure 5.7: **Hyperparameter sensitivity of ANCHOR.** We vary the initial KL weight β_0 (top), the KL annealing interval T_{KL} (middle), and the residual online conservative weight α_{on} (bottom). Here, α^* denotes an intermediate value between 0 and α_{off}, where α_{off} is the conservative regularization weight used during offline pretraining. ANCHOR remains stable under moderate changes to KL-related hyperparameters, while α_{on} controls the balance between early stability and final improvement.

mixed, consistent with the analysis in Section 3. Removing the conservative penalty entirely achieves strong final performance across tasks, but causes larger catastrophic failure in kitchen-mixed. These results support the design choice of ANCHOR: reducing conservative regularization can improve online adaptability, while retaining a small residual penalty can help stabilize the early online phase depending on the task.

Figure 5.8 further evaluates the sensitivity of ANCHOR to batch size and UTD ratio. In kitchen-mixed and pen-binary, the tested variants exhibit similar early stability and asymptotic performance. In antmaze-large-diverse, however, the larger batch size of 1024 and the larger UTD ratio of 16 lead to greater learning instability and weaker prevention of catastrophic failure. Compared with the baselines in Figure 5.1, these variants still achieve competitive catastrophic failure and asymptotic performance, indicating that ANCHOR is not highly sensitive to batch size or UTD ratio, although overly aggressive update configurations can reduce early stability.

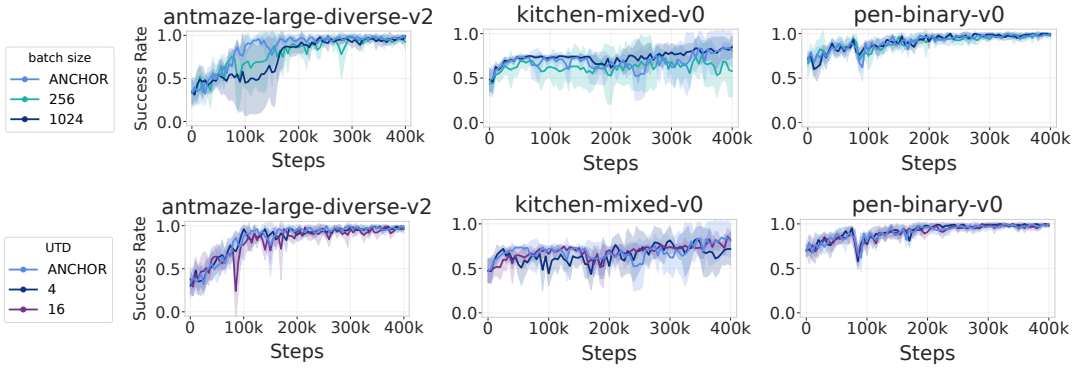


Figure 5.8: **Sensitivity to batch size and UTD ratio.** We vary the batch size (top) and UTD ratio (bottom) across three tasks. ANCHOR is generally robust to these choices, although larger batch size and larger UTD can increase learning instability.

A Larger Initial KL Weight Yields Stronger Early Anchoring

We also check whether increasing the initial KL coefficient reduces the measured deviation between the online actor and the offline-pretrained policy. This is not automatic in practice, since the KL term is optimized jointly with the online RL objective and may not be minimized effectively under online updates. We compare two initial KL weights, $\beta_0 = 0.5$ and $\beta_0 = 2.0$. Figure 5.9 reports the KL loss during online fine-tuning.

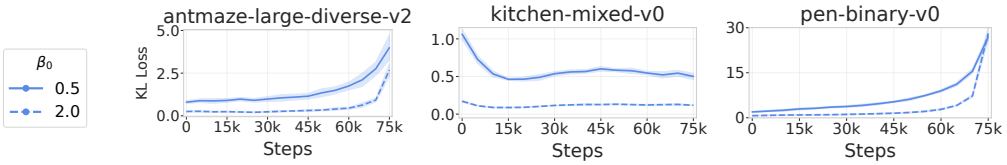


Figure 5.9: **Effect of the initial KL weight.** A larger initial KL weight ($\beta_0 = 2.0$) produces lower KL loss early in online fine-tuning, indicating that the KL penalty more strongly constrains deviation from the offline policy. As the KL coefficient is annealed, the gap between $\beta_0 = 0.5$ and $\beta_0 = 2.0$ gradually narrows.

The larger initial KL weight consistently produces lower KL loss during the early phase of online adaptation. This confirms that β_0 controls the strength of actor anchoring in practice: increasing β_0 keeps the online actor closer to the offline-pretrained policy despite simultaneous optimization of the RL objective. As training proceeds and the KL coefficient is annealed, the KL losses under the two settings become similar. This behavior is consistent with the design of ANCHOR, where KL anchoring is strongest early in online fine-tuning and gradually weakens as online improvement takes over.

Chapter 6
Conclusion

Offline-to-online adaptation without offline replay requires balancing early stability with online improvement. ANCHOR addresses this by anchoring the actor to the offline-pretrained policy during the vulnerable early phase, annealing this constraint as online learning progresses, and retaining only a small residual conservative penalty for the critic. Across FrankaKitchen, AntMaze, and Adroit tasks, this design reduces catastrophic failure while maintaining asymptotic performance.

Our analysis and ablations clarify the operating conditions of ANCHOR. The theoretical results suggest that anchoring is most useful when the pretrained policy induces nearby occupancy distributions under which the critic remains reliable, while the regret bound supports relaxing the anchor for further online improvement. Empirically, the ablations show that ANCHOR is not simply a fixed constraint: its benefit comes from using the pretrained policy as a temporary behavioral prior, while allowing both the actor and critic to adapt as online data accumulates. They also show that the residual conservative penalty should be kept small enough to avoid limiting final performance, but can still help stabilize the early online phase in some tasks.

These findings suggest that offline-to-online RL methods should control how strongly and how long they rely on the pretrained policy during deployment-time adaptation. Future work could develop adaptive anchoring schedules that adjust the KL constraint based on online signals. Another important direction is to better disentangle the roles of actor drift and critic error in catastrophic failure, which may lead to more targeted stabilization mechanisms. Finally, settings where the pretrained policy provides little useful guidance may require combining anchoring with stronger exploration mechanisms that enable online improvement without reintroducing early collapse. Additional limitations are discussed in Appendix D.

References

[1] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.

[2] Asifullah Khan, Anabia Sohail, Mustansar Fiaz, Mehdi Hassan, Tariq Habib Afridi, Sibghat Ullah Marwat, Farzeen Munir, Safdar Ali, Hannan Naseem, Muhammad Zaigham Zaheer, et al. A survey of the self supervised learning mechanisms for vision transformers. *arXiv preprint arXiv:2408.17059*, 2024.

[3] 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.

[4] Qin-Wen Luo, Ming-Kun Xie, Yewen Wang, and Sheng-Jun Huang. Optimistic critic reconstruction and constrained fine-tuning for general offline-to-online rl. *Advances in Neural Information Processing Systems*, 37:108167–108207, 2024.

[5] 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. In *Advances in Neural Information Processing Systems*, volume 36, pages 62244–62269, 2023.

[6] Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[7] Wei Xiao, Jiacheng Liu, Zifeng Zhuang, Runze Suo, Shangke Lyu, and Donglin Wang. Efficient online rl fine tuning with offline pre-trained policy only. *arXiv preprint arXiv:2505.16856*, 2025.

[8] Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. Reinforcement learning in robotic applications: A comprehensive survey. *Artificial Intelligence Review*, 55(2): 945–990, 2022. doi: 10.1007/s10462-021-10067-x.

[9] Siqi Liu, Kay Choong See, Kee Yuan Ngiam, Leo Anthony Celi, Xudong Sun, and Mengling Feng. Reinforcement learning for clinical decision support in critical care: Comprehensive review. *Journal of Medical Internet Research*, 22(7):e18477, 2020. doi: 10.2196/18477.

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

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

[12] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative

q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191, 2020.

[13] Xiaoyu Wen, Xudong Yu, Rui Yang, Haoyuan Chen, Chenjia Bai, and Zhen Wang. Towards Robust Offline-to-Online Reinforcement Learning via Uncertainty and Smoothness. *Journal of Artificial Intelligence Research*, 81:481–509, 2024. doi: 10.1613/jair.1.16457. URL `https://jair.org/index.php/jair/article/view/16457`.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Dylan J. Foster, Zakaria Mhammedi, and Dhruv Rohatgi. Is a good foundation necessary for efficient reinforcement learning? the computational role of the base model in exploration. In *Proceedings of the Thirty Eighth Conference on Learning Theory*, Proceedings of Machine Learning Research. PMLR, 2025.

[16] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2015.

[17] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1861–1870. PMLR, 2018.

[19] Xiaocong Chen, Siyu Wang, Julian McAuley, Dietmar Jannach, and Lina Yao. On the opportunities and challenges of offline reinforcement learning for recommender systems. *ACM Transactions on Information Systems*, 42(6):1–26, 2024.

[20] Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(8):10237–10257, 2023.

[21] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 20132–20145, 2021.

[22] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[23] Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:11592–11620, 2023.

[24] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in Neural Information Processing Systems*, 34:7436–7447, 2021.

[25] Alexander Nikulin, Vladislav Kurenkov, Denis Tarasov, and Sergey Kolesnikov. Anti-exploration by random network distillation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 26228–26244. PMLR, 2023.

[26] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21810–21823, 2020.

[27] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[28] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in Neural Information Processing Systems*, 34:28954–28967, 2021.

[29] Rohan Chitnis, Yingchen Xu, Bobak Hashemi, Lucas Lehnert, Urun Dogan, Zheqing Zhu, and Olivier Delalleau. Iql-td-mpc: Implicit q-learning for hierarchical model predictive control. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9154–9160. IEEE, 2024.

[30] Kwanyoung Park and Youngwoon Lee. Model-based offline reinforcement learning with lower expectile q-learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[31] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in Neural Information Processing Systems*, 34:4933–4946, 2021.

[32] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

[33] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[34] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020.

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

[36] Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[37] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *Proceedings of the*

International Conference on Machine Learning (ICML), pages 5774–5783. PMLR, 2021.

[38] Hao Hu, Yiqin Yang, Jianing Ye, Chengjie Wu, Ziqing Mai, Yujing Hu, Tangjie Lv, Changjie Fan, Qianchuan Zhao, and Chongjie Zhang. Bayesian design principles for offline-to-online reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, ICML'24. JMLR.org, 2024.

[39] Yongjae Shin, Jeonghye Kim, Whiyoung Jung, Sunghoon Hong, Deunsol Yoon, Youngsoo Jang, Geonhyeong Kim, Jongseong Chae, Youngchul Sung, Kanghoon Lee, et al. Online pre-training for offline-to-online reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025. URL `https://openreview.net/forum?id=5anr1H5vR5`.

[40] 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*, pages 1702–1712. PMLR, 2022.

[41] Kai Zhao, Jianye Hao, Yi Ma, Jinyi Liu, Yan Zheng, and Zhaopeng Meng. Enoto: Improving offline-to-online reinforcement learning with q-ensembles. page 2609–2611, 2023.

[42] Jiaheng Feng, Mingxiao Feng, Haolin Song, Wengang Zhou, and Houqiang Li. Suf: Stabilized unconstrained fine-tuning for offline-to-online reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11961–11969, 2024.

[43] Siyuan Guo, Yanchao Sun, Jifeng Hu, Sili Huang, Hechang Chen, Haiyin Piao, Lichao Sun, and Yi Chang. A simple unified uncertainty-guided framework for offline-to-online reinforcement learning. *arXiv preprint arXiv:2306.07541*, 2023.

[44] 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 *Proceedings of the International Conference on Machine Learning (ICML)*, pages 34556–34583. PMLR, 2023.

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

[46] Zishun Yu and Xinhua Zhang. Actor-Critic Alignment for Offline-to-Online Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 40452–40474, 2023.

[47] Whitney K Newey and James L Powell. Asymmetric least squares estimation and testing. *Econometrica: Journal of the Econometric Society*, pages 819–847, 1987.

[48] 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.

[49] Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. In *Proceedings of the Inter-*

national Conference on Learning Representations (ICLR), 2021. arXiv preprint arXiv:2101.05982.

[50] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[51] Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1577–1594. PMLR, 2023.

Appendix A
Related Works

Offline RL. Online reinforcement learning requires interaction with the environment, which can be expensive, unsafe, or impractical in domains such as robotics, healthcare, and recommender systems [8, 9, 19]. Offline RL addresses this limitation by learning policies from a static dataset collected by a behavior policy [20]. However, because the dataset provides limited coverage of the state-action space, offline-trained policies can suffer when they encounter unseen or poorly covered regions at deployment [20].

A large body of work has been developed to address this challenge, including regularization-based methods [12, 21–23], uncertainty-aware methods [24, 25], model-based methods [26–30], one-step methods [22, 31, 32], weighted-regression methods [33, 34], and in-sample maximization methods [17, 35, 36]. Among these approaches, we build on Conservative Q-Learning (CQL) [12], which directly mitigates value overestimation by penalizing high Q-values for actions outside the dataset support. We also include Implicit Q-Learning (IQL) [17] as a baseline, as it improves over the behavior policy through value-based policy extraction without explicitly evaluating out-of-distribution actions.

Offline-to-Online RL. Offline-to-online RL studies how to further improve an offline-pretrained agent through online interaction. Although offline RL methods such as CQL [12], IQL [17], and related approaches [23, 37] can be fine-tuned online, simply continuing the offline objective often yields limited improvement [5]. This has motivated methods that explicitly adapt the offline-trained agent to the online learning regime.

Existing approaches include relaxing excessive conservatism in value estimates [4, 5, 38], introducing an intermediate adaptation phase between offline pretraining and online fine-tuning [6, 7, 39], using multiple Q-functions [40, 41], tuning the update-to-data (UTD) ratio [7, 42], and incorporating uncertainty estimation [13, 43]. Another line of

work focuses on leveraging pretrained policies directly, rather than updating both the actor and critic [7, 44, 45].

A key challenge in this transition is catastrophic failure: performance can drop sharply at the start of online fine-tuning due to distribution shift and unstable value learning [5, 13]. Prior work has addressed this issue through uncertainty estimation [13], calibration penalties [5], and actor-critic alignment [46]. In contrast, ANCHOR stabilizes the early online phase by anchoring the actor to the offline-pretrained policy with a KL penalty. The KL coefficient is linearly annealed to zero, allowing the method to move from conservative checkpoint adaptation to unconstrained online improvement. For the critic, ANCHOR immediately reduces the CQL coefficient to a small residual value α_{on} and keeps it fixed during online fine-tuning, avoiding excessive conservatism while retaining mild regularization against overestimation. Thus, ANCHOR addresses catastrophic failure with a simple adaptation schedule while preserving strong asymptotic performance, without requiring additional uncertainty models or complex auxiliary modules.

Trust-region and KL-regularized policy optimization. Our actor-side KL regularization is related in form to trust-region methods such as TRPO [16], which constrain the KL divergence between consecutive policies to stabilize online policy updates. However, ANCHOR differs in both purpose and reference distribution. TRPO imposes a local trust region around the previous policy at each update to support stable on-policy improvement. In contrast, ANCHOR regularizes the online actor toward a fixed offline-pretrained policy during the early phase of offline-to-online adaptation. This anchor is not intended to enforce a monotonic policy improvement guarantee; rather, it preserves pretrained behavior when offline replay is unavailable and prevents abrupt policy drift into regions where the critic may be unreliable. Moreover, ANCHOR anneals the KL coefficient to zero, explicitly removing the constraint once the online actor and critic have adapted. Thus, while both approaches use KL-based control of policy updates, ANCHOR uses KL regularization as a temporary checkpoint-adaptation mechanism for offline-to-online RL.

Appendix B
Details on Baseline Algorithms

Soft Actor-Critic (SAC) [18]. SAC extends standard actor-critic methods by incorporating entropy maximization into the RL objective. Instead of maximizing only the expected return, SAC also maximizes the entropy of the policy to encourage exploration and prevent premature convergence to deterministic policies. The resulting objective is to maximize the expected sum of rewards and entropies:

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right)\right],$$

where $\mathcal{H}(\pi(\cdot|s_t)) = -\mathbb{E}_{a_t\sim\pi}[\log \pi(a_t|s_t)]$ denotes the entropy of the policy at state s_t, and $\alpha > 0$ is the temperature parameter that balances reward maximization and entropy.

Conservative Q-Learning (CQL) [12]. CQL learns a Q-function by explicitly regularizing Q-values to mitigate overestimation issues common in offline RL [20]:

$$\mathcal{L}(\theta) = \alpha \underbrace{\left(\mathbb{E}_{s\sim\mathcal{D}, a\sim\pi}[Q_\theta(s,a)] - \mathbb{E}_{s,a\sim\mathcal{D}}[Q_\theta(s,a)]\right)}_{\text{Conservative regularizer}} + \mathcal{L}_{\text{TD}}(\theta)$$

Here, π represents the current policy, $\mathcal{D}$ is the offline dataset, and α controls the intensity of the conservative regularization. The term $\mathcal{L}_{TD}$ is the temporal difference loss used in Q-learning methods, while the conservative regularization penalizes Q-values for state-action pairs not present in the offline dataset $\mathcal{D}$.

Implicit Q-learning (IQL) [17]. The IQL algorithm learns a state-value function $V_{\theta_V} : \mathcal{S} \to \mathbb{R}$ and an action-value function $Q_{\theta_Q} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ by minimizing:

$$\mathcal{L}_V(\theta_V) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\ell_\kappa^2\left(V_{\theta_V}(s) - Q_{\bar{\theta}_Q}(s,a)\right)\right],$$

$$\mathcal{L}_Q(\theta_Q) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_{\theta_Q}(s,a) - r - \gamma V_{\theta_V}(s')\right)^2\right],$$

where the expectile loss is defined as $\ell_\kappa^2(x) = |\kappa - \mathbf{1}[x < 0]|x^2$ [47], and $\bar{\theta}_Q$ represents the target network parameters. Unlike standard MSE loss, expectile loss asymmetrically weights positive and negative errors, with $\kappa > 0.5$ emphasizing higher Q-values. By doing so, IQL evaluates only actions from the dataset, which enables policy improvement without querying out-of-distribution actions, thereby inducing implicit conservatism.

Calibrated Q-Learning (Cal-QL) [5]. Cal-QL adjusts CQL by calibrating the learned Q-function relative to a reference policy to avoid initial degradation caused by overly pessimistic Q-values during fine-tuning. The conservative regularization term in CQL is altered by substituting $\mathbb{E}_{s\sim\mathcal{D},a\sim\pi}[Q_\theta(s,a)]$ with $\mathbb{E}_{s\sim\mathcal{D},a\sim\pi}[\max(Q_\theta(s,a), V^\mu(s))]$, where $V^\mu(s)$ represents the value function of the reference policy μ, which can be estimated via Monte-Carlo return. This calibration prevents the conservative penalty from further suppressing actions whose estimated values are already below the reference value, thereby mitigating the overly pessimistic Q-values that can hinder online fine-tuning.

Policy-Only Reinforcement Learning Fine-Tuning (PORL) [7]. PORL focuses on the challenge of fine-tuning online RL using only a pretrained policy, without relying on pretrained Q-functions or offline datasets. This approach is especially beneficial when pretrained Q-functions are either unreliable due to pessimism or unavailable, such as in imitation learning scenarios. PORL begins by training the randomly initialized Q-function at the beginning of the online fine-tuning phase. Training data is collected based on the online interaction of the pretrained policy using an epsilon-greedy exploration strategy. During this initial sampling phase, the Q-function is trained via temporal difference learning with a high UTD ratio. After this pre-sampling period, PORL transitions to standard SAC fine-tuning, updating both the policy and the Q-function.

Warm-start RL (WSRL) [6]. WSRL addresses the distribution shift issue from offline to online data in fine-tuning without offline data by introducing a warmup phase. This phase utilizes a frozen pretrained policy to collect online rollouts. The data collected during warmup bridges the distribution mismatch and helps recalibrate

the offline Q-function to the online distribution, allowing the method to adapt in the online environment quickly. Once the warmup phase is complete, WSRL proceeds with conventional online RL using SAC, employing a high UTD ratio.

Appendix C
Implementation Details

C.1 Details on Environments

AntMaze. The AntMaze environment, part of the D4RL benchmark suite [10], tasks an 8-DoF ant quadruped robot with navigating through a large and complex maze to reach a designated goal position. The agent receives a binary reward of $+1$ only upon successfully reaching the goal. The observation space is 29-dimensional, including the robot's position, orientation, and velocity, while the action space is a continuous 8-dimensional vector, normalized to the range $[-1, 1]$. We use three variants: antmaze-large-diverse-v2, which contains trajectories collected by commanding the agent to random goals from random start positions; antmaze-large-play-v2, which contains trajectories directed to a specific location; and antmaze-ultra-diverse-v2, which uses a larger and more complex maze with the same diverse data collection strategy. The two large-maze variants share a maximum episode length of 1000 steps, while antmaze-ultra-diverse-v2 allows up to 700 steps.

FrankaKitchen. The FrankaKitchen environment contained in D4RL [10] requires controlling a 9-DoF Franka Panda robotic arm to manipulate various kitchen appliances and configure the environment into a predefined target state. Each task consists of four subtasks, and the agent receives a reward between 0 and 4 based on the number of successfully completed subtasks. The observation space is 60-dimensional, encompassing joint positions and object states, and the action space is a 9-dimensional continuous vector normalized to $[-1, 1]$. We use three benchmark environments from D4RL: kitchen-partial-v0, kitchen-mixed-v0, and kitchen-complete-v0. kitchen-partial-v0 includes a mix of complete and incomplete demonstrations, where task elements involve operating the microwave, kettle, light switch, and slide cabinet. kitchen-mixed-v0 contains only incomplete demonstrations, with task components including the microwave, kettle, bottom

burner, and light switch. kitchen-complete-v0 contains only complete demonstrations covering the same four subtasks as kitchen-partial-v0: microwave, kettle, light switch, and slide cabinet. All three environments use a maximum episode length of 280 steps.

Adroit. The Adroit suite [48] evaluates dexterous manipulation using a high-DoF robotic hand. Specifically, we use three tasks from the D4RL benchmark: pen-binary-v0, door-binary-v0, and relocate-binary-v0. In pen-binary-v0, a 24-DoF shadow hand must reorient a pen to match a target pose, while in door-binary-v0, a 28-DoF hand must grasp and rotate a door handle to open it. In relocate-binary-v0, a 30-DoF hand must pick up an object and relocate it to a target position. All three environments have sparse binary rewards: a reward of $+1$ is given only upon successful task completion. The observation space is 45-dimensional for the pen task, 39-dimensional for the door and relocate tasks, consisting of hand joint angles and object poses. The action space is continuous, 24-dimensional for the pen task, 28-dimensional for the door task, and 30-dimensional for the relocate task, each normalized to $[-1, 1]$. The maximum episode lengths are 100 steps for the pen task and 200 steps for the door and relocate tasks.

C.2 Network Architectures

ANCHOR uses the same network architecture as the CQL baseline in the WSRL codebase [6]. The agent follows an actor-critic framework. The actor takes an observation as input and outputs the mean and log standard deviation of a Gaussian action distribution. The critic takes the concatenated observation-action pair as input and uses an ensemble of ten Q-functions, each producing a scalar Q-value. For stability, two Q-functions are randomly subsampled during target computation, and the minimum of the two values is used. Both the actor and critic are implemented as multilayer perceptrons (MLPs) with ReLU activations. A learnable temperature parameter controls the entropy regularization. The hidden dimensions for each domain are reported in Table C.1.

C.3 Reproducing Baselines

All baseline methods are reproduced under a unified experimental setup. For WSRL, Cal-QL, CQL, IQL, and SAC, we use the implementations and hyperparameters provided in the WSRL codebase [6], with several stability-oriented modifications applied consistently across methods. Specifically, (i) we increase the batch size from 256 to 512; (ii) we use an ensemble of ten Q-functions and compute target values using the minimum over two critics randomly subsampled at each update, following REDQ [49]; and (iii) we apply layer normalization [50] to both the policy and critic networks to mitigate uncontrolled extrapolation effects [51]. Since no public implementation of PORL was available, we reimplement it following Xiao et al. [7] and use the authors' reported settings.

C.4 Hyperparameters

Table C.1 reports the hyperparameters used to train ANCHOR. For hyperparameters not specific to ANCHOR, we follow the default CQL settings used by Zhou et al. [6], except for the UTD ratio in door-binary, where we use the WSRL setting. For ANCHOR-specific hyperparameters, we tune $\alpha_{\text{on}} \in \{0.0, 1.0\}$, $T_{\text{KL}} \in \{40\text{K}, 80\text{K}, 160\text{K}\}$, and $\beta_0 \in \{0.2, 0.4, 1.0\}$.

C.5 Online Training Time Comparison

Table C.2 compares the online training time of ANCHOR and the baselines. ANCHOR has a similar average training time to WSRL and PORL, despite the additional actor-side KL regularization. While it is slower than lightweight baselines such as IQL and SAC, these methods suffer from substantially weaker asymptotic performance and/or larger catastrophic failure in our main results. Considering the gains of ANCHOR in both early stability and final performance, its computational overhead is moderate and comparable to other competitive offline-to-online methods.

Table C.1: **Hyperparameters of ANCHOR.**

	AntMaze		**Adroit**		**Kitchen**	
	large diverse	large play	door	pen	partial	mixed
ANCHOR						
α_{on}	0.0	0.0	0.0	0.0	1.0	1.0
T_{KL}	80,000	80,000	40,000	80,000	160,000	160,000
β_0	0.4	0.2	0.2	1.0	1.0	1.0
Optimization						
Actor Learning Rate	1×10^{-4}		1×10^{-4}		1×10^{-4}	
Critic Learning Rate	3×10^{-4}		3×10^{-4}		3×10^{-4}	
Batch Size	512		512		512	
UTD	1	1	16	1	1	1
Offline Steps	1,000,000		40,000		250,000	
α_{off}	5.0		1.0		5.0	
Warmup Steps	5,000		5,000		5,000	
Architecture						
Critic Network	[256, 256, 256, 256]		[512, 512, 512]		[512, 512, 512]	
Actor Network	[256, 256]		[512, 512]		[512, 512, 512]	
Activations	ReLU		ReLU		ReLU	
Q-ensemble	10		10		10	

Table C.2: **Online training time (in decimal hours) comparison across baselines.**

Task	**ANCHOR (Ours)**	WSRL	PORL	Cal-QL	CQL	IQL	SAC
antmaze-large-diverse-v2	2.75 ± 0.02	2.73 ± 0.01	3.35 ± 0.07	3.44 ± 0.01	2.82 ± 0.02	2.08 ± 0.02	1.53 ± 0.0
antmaze-large-play-v2	2.75 ± 0.01	3.48 ± 0.08	3.54 ± 0.06	3.26 ± 0.03	2.34 ± 0.03	2.09 ± 0.03	1.52 ± 0.0
kitchen-partial-v0	4.31 ± 0.02	3.37 ± 0.0	3.41 ± 0.03	4.35 ± 0.01	3.28 ± 0.05	1.41 ± 0.01	1.38 ± 0.0
kitchen-mixed-v0	3.88 ± 0.03	3.61 ± 0.01	3.51 ± 0.13	4.36 ± 0.0	3.33 ± 0.02	1.46 ± 0.02	1.36 ± 0.0
door-binary-v0	3.07 ± 0.01	2.9 ± 0.01	2.93 ± 0.01	3.55 ± 0.01	2.43 ± 0.01	1.28 ± 0.02	0.78 ± 0.01
pen-binary-v0	2.33 ± 0.02	2.77 ± 0.01	2.79 ± 0.01	3.51 ± 0.01	2.29 ± 0.01	1.13 ± 0.1	0.67 ± 0.0
Average	3.18	3.14	3.25	3.75	2.75	1.58	1.21

C.6 Experimental Setup and Reproducibility

All experiments were conducted on a compute node equipped with 8 NVIDIA A100 GPUs (80GB each), an AMD EPYC 7543 32-Core CPU, and 885 GB of RAM. The software environment was based on Python 3.10.18, PyTorch 2.7.0, JAX 0.4.25 with CUDA 12.2.

Appendix D
Limitations

While ANCHOR provides a simple and effective recipe for offline-to-online adaptation without offline replay, several limitations remain.

Scope of evaluation. Our experiments focus on single-agent continuous-control benchmarks, including AntMaze, FrankaKitchen, and Adroit. We do not evaluate ANCHOR under hard safety constraints, non-stationary dynamics, partial observability, or multi-agent settings. Therefore, its effectiveness in these regimes is not guaranteed.

Dependence on pretrained policy quality. ANCHOR is most effective when the offline-pretrained policy contains useful behavior to preserve. When the pretrained policy has near-zero success, as in relocate-binary, KL anchoring provides limited benefit. This suggests that ANCHOR is better suited for stabilizing useful pretrained behavior than for solving tasks where the offline policy provides little meaningful guidance.

Limited exploration in hard sparse-reward environments. ANCHOR controls policy drift during online fine-tuning, but it does not introduce a new exploration mechanism. In challenging sparse-reward environments such as antmaze-ultra-diverse, most methods fail to improve substantially over the offline-pretrained policy. Addressing such settings may require combining KL anchoring with stronger exploration or task-specific adaptation mechanisms.

Hand-designed KL schedules. ANCHOR uses a manually specified initial KL weight and annealing interval. Although our ablations show robustness to moderate changes in these hyperparameters, selecting the schedule still requires some environment-level tuning. Future work could develop adaptive KL schedules based on online uncertainty, critic reliability, or measured policy drift.

Backbone dependence. Most of our experiments instantiate ANCHOR on top of CQL. We also test actor-side KL anchoring with IQL and find that it reduces early

catastrophic failure, but does not improve asymptotic performance. This suggests that different offline RL backbones may require backbone-specific mechanisms for relaxing offline inductive biases during online adaptation.

Statistical coverage. Unless otherwise noted, experiments are run with five random seeds due to computational constraints, and some auxiliary analyses use fewer seeds. Although confidence intervals are reported, rare failure modes may still be under-sampled.

Appendix E

Visualizations

(a) **antmaze-large-diverse-v2:** 8-DoF ant navigating through a maze to reach the goal.

(b) **antmaze-large-play-v2:** Same evaluation as diverse-v2, different training data.

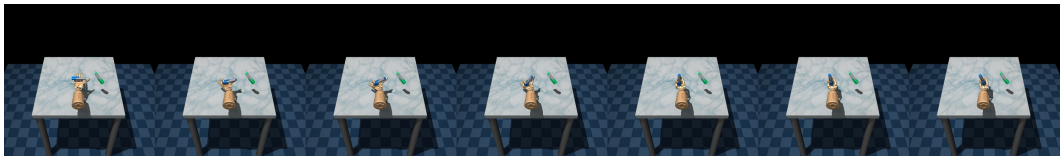
(c) **antmaze-ultra-diverse-v2:** 8-DoF ant navigating through a larger maze than large-diverse-v2.

(d) **kitchen-partial-v0:** Completing subtasks: microwave, kettle, light switch, and slide cabinet.

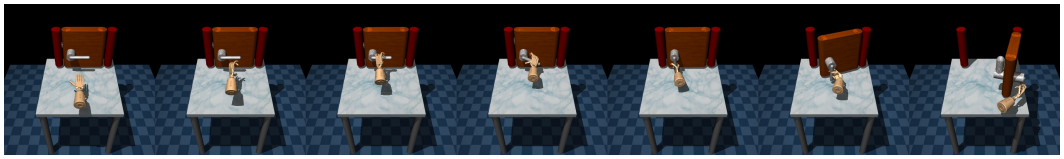
(e) **kitchen-mixed-v0:** Completing subtasks: microwave, kettle, bottom burner, and light switch.

Figure E.1: **Representative episodes across nine tasks.** Each row shows temporally ordered frames from a trajectory, demonstrating navigation (antmaze-ultra-diverse-v2, antmaze-large-diverse-v2, antmaze-large-play-v2), kitchen manipulation (kitchen-partial-v0, kitchen-mixed-v0, kitchen-complete-v0), and dexterous control (pen-binary-v0, door-binary-v0, relocate-binary-v0).

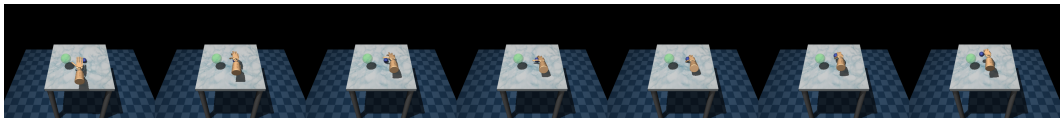
(f) **kitchen-complete-v0:** Completing subtasks: microwave, kettle, light switch, and slide cabinet.



(g) **pen-binary-v0:** 24-DoF shadow hand reorienting a pen to target pose.



(h) **door-binary-v0:** 28-DoF hand opening a door by rotating the handle.



(i) **relocate-binary-v0:** 30-DoF shadow hand relocating an object to a target location.

Figure E.1: **Representative episodes across nine tasks.** (continued)

Appendix F
Numerical Results

F.1 Raw Performance and Catastrophic Failure Values

Table F.1: **Performance at 400k online steps. Bold** indicates the best performance per task, and underline indicates the second best.

Task	**ANCHOR (Ours)**	WSRL	PORL	Cal-QL	CQL	IQL	SAC
antmaze-large-diverse-v2	**0.97** ± 0.01	0.68 ± 0.17	0.9 ± 0.07	**0.97** ± 0.01	0.92 ± 0.05	0.06 ± 0.05	0.0 ± 0.0
antmaze-large-play-v2	**0.96** ± 0.02	0.95 ± 0.02	0.47 ± 0.2	0.9 ± 0.07	0.95 ± 0.03	0.01 ± 0.01	0.0 ± 0.0
kitchen-partial-v0	**1.0** ± 0.0	0.93 ± 0.06	0.91 ± 0.06	0.7 ± 0.09	0.44 ± 0.12	0.4 ± 0.05	0.49 ± 0.07
kitchen-mixed-v0	0.81 ± 0.09	0.72 ± 0.11	**0.88** ± 0.06	0.45 ± 0.12	0.4 ± 0.06	0.31 ± 0.09	0.3 ± 0.05
door-binary-v0	**1.0** ± 0.0	0.8 ± 0.2	0.6 ± 0.24	0.19 ± 0.19	0.18 ± 0.18	0.0 ± 0.0	0.2 ± 0.2
pen-binary-v0	**1.0** ± 0.0	0.97 ± 0.01	0.99 ± 0.01	0.94 ± 0.02	0.81 ± 0.09	0.33 ± 0.1	0.78 ± 0.04
Average	**0.86** ± 0.04	0.72 ± 0.07	0.68 ± 0.07	0.62 ± 0.06	0.55 ± 0.07	0.16 ± 0.03	0.25 ± 0.06

Table F.2: **Performance at 350k online steps. Bold** indicates the best performance per task, and underline indicates the second best.

Task	**ANCHOR (Ours)**	WSRL	PORL	Cal-QL	CQL	IQL	SAC
antmaze-large-diverse-v2	**0.97** ± 0.01	0.73 ± 0.18	0.84 ± 0.07	0.95 ± 0.03	0.96 ± 0.02	0.03 ± 0.03	0.0 ± 0.0
antmaze-large-play-v2	0.96 ± 0.02	**0.98** ± 0.01	0.37 ± 0.23	0.89 ± 0.07	0.95 ± 0.03	0.01 ± 0.01	0.0 ± 0.0
kitchen-partial-v0	**1.0** ± 0.0	0.95 ± 0.05	0.9 ± 0.06	0.7 ± 0.09	0.44 ± 0.12	0.39 ± 0.05	0.47 ± 0.06
kitchen-mixed-v0	0.83 ± 0.1	0.7 ± 0.12	**0.88** ± 0.06	0.45 ± 0.12	0.4 ± 0.06	0.3 ± 0.08	0.25 ± 0.03
door-binary-v0	**0.97** ± 0.01	0.8 ± 0.2	0.6 ± 0.24	0.17 ± 0.17	0.19 ± 0.19	0.0 ± 0.0	0.2 ± 0.2
pen-binary-v0	0.94 ± 0.02	**0.96** ± 0.02	0.95 ± 0.02	**0.96** ± 0.01	0.72 ± 0.05	0.4 ± 0.05	0.84 ± 0.02
Average	**0.83** ± 0.05	0.72 ± 0.07	0.65 ± 0.07	0.63 ± 0.06	0.54 ± 0.06	0.16 ± 0.03	0.25 ± 0.06

Tables F.1 and F.2 report success rates (mean ± SE) after 400k and at 350k online fine-tuning, respectively, averaged over five seeds. Success rates for all baselines stabilize by 350k steps, validating the 350k-400k window as a reliable proxy for asymptotic performance. Offline-to-online methods consistently outperform offline-only approaches, and ANCHOR achieves the highest average success rate.

Table F.3 reports success rates (mean ± SE) at the end of the offline pretraining phase, i.e., the start of online fine-tuning, averaged over five seeds. Despite identical offline training procedures, end-of-offline performance shows small variations across

Table F.3: **Performance after the offline phase.**

Task	**ANCHOR (Ours)**	WSRL	PORL	Cal-QL	CQL	IQL
antmaze-large-diverse-v2	0.33 ± 0.09	0.24 ± 0.05	0.39 ± 0.04	0.26 ± 0.04	0.26 ± 0.05	0.03 ± 0.02
antmaze-large-play-v2	0.29 ± 0.05	0.29 ± 0.02	0.37 ± 0.02	0.24 ± 0.04	0.28 ± 0.05	0.0 ± 0.0
kitchen-partial-v0	0.7 ± 0.03	0.73 ± 0.06	0.7 ± 0.04	0.68 ± 0.05	0.7 ± 0.06	0.3 ± 0.06
kitchen-mixed-v0	0.47 ± 0.06	0.47 ± 0.06	0.49 ± 0.07	0.52 ± 0.06	0.58 ± 0.03	0.4 ± 0.04
door-binary-v0	0.26 ± 0.04	0.31 ± 0.19	0.24 ± 0.03	0.21 ± 0.07	0.08 ± 0.08	0.05 ± 0.04
pen-binary-v0	0.69 ± 0.03	0.66 ± 0.05	0.74 ± 0.03	0.75 ± 0.05	0.58 ± 0.03	0.89 ± 0.04
Average	0.42 ± 0.04	0.41 ± 0.04	0.45 ± 0.04	0.4 ± 0.04	0.39 ± 0.04	0.25 ± 0.05

ANCHOR, WSRL, PORL, Cal-QL, and CQL. These differences reflect evaluation stochasticity rather than algorithmic effects.

Table F.4: **Catastrophic failure (0-400K window). Bold** indicates the lowest catastrophic failure, and underline indicates the second lowest.

Task	**ANCHOR (Ours)**	WSRL	PORL	Cal-QL	CQL	IQL
antmaze-large-diverse-v2	0.12 ± 0.07	**0.07** ± 0.04	0.39 ± 0.04	0.17 ± 0.07	0.21 ± 0.04	0.03 ± 0.02
antmaze-large-play-v2	**0.11** ± 0.09	0.2 ± 0.06	0.37 ± 0.02	0.16 ± 0.05	0.16 ± 0.07	0.0 ± 0.0
kitchen-partial-v0	**0.15** ± 0.08	0.68 ± 0.09	0.7 ± 0.04	0.68 ± 0.05	0.7 ± 0.06	0.19 ± 0.04
kitchen-mixed-v0	**0.22** ± 0.11	0.42 ± 0.09	0.48 ± 0.07	0.52 ± 0.06	0.58 ± 0.03	0.35 ± 0.04
door-binary-v0	0.26 ± 0.04	0.31 ± 0.19	0.24 ± 0.03	0.21 ± 0.07	0.08 ± 0.08	0.05 ± 0.04
pen-binary-v0	**0.14** ± 0.04	0.52 ± 0.04	0.66 ± 0.04	0.33 ± 0.08	0.58 ± 0.03	0.87 ± 0.03
Average	**0.16** ± 0.03	0.34 ± 0.05	0.44 ± 0.03	0.31 ± 0.04	0.36 ± 0.04	0.23 ± 0.05

Table F.5: **Catastrophic failure (0-100K window). Bold** indicates the lowest catastrophic failure, and underline indicates the second lowest.

Task	**ANCHOR (Ours)**	WSRL	PORL	Cal-QL	CQL	IQL
antmaze-large-diverse-v2	0.1 ± 0.08	**0.07** ± 0.04	0.39 ± 0.04	0.17 ± 0.07	0.21 ± 0.04	0.03 ± 0.02
antmaze-large-play-v2	**0.11** ± 0.09	0.2 ± 0.06	0.37 ± 0.02	0.16 ± 0.05	0.16 ± 0.07	0.0 ± 0.0
kitchen-partial-v0	**0.06** ± 0.02	0.68 ± 0.09	0.7 ± 0.04	0.68 ± 0.05	0.7 ± 0.06	0.18 ± 0.04
kitchen-mixed-v0	**0.03** ± 0.01	0.42 ± 0.09	0.48 ± 0.07	0.52 ± 0.06	0.58 ± 0.03	0.33 ± 0.03
door-binary-v0	0.26 ± 0.04	0.31 ± 0.19	0.24 ± 0.03	0.21 ± 0.07	0.08 ± 0.08	0.05 ± 0.04
pen-binary-v0	**0.14** ± 0.04	0.52 ± 0.04	0.66 ± 0.04	0.33 ± 0.08	0.58 ± 0.03	0.83 ± 0.04
Average	**0.12** ± 0.02	0.34 ± 0.05	0.44 ± 0.03	0.31 ± 0.04	0.36 ± 0.04	0.22 ± 0.05

Tables F.4 and F.5 report catastrophic failure over the 0-400K and 0-100K windows, respectively. All values are averaged over five seeds and reported as mean ± SE. The

estimates are similar across the two windows, indicating that most failures occur early in the offline-to-online transition. ANCHOR achieves the lowest average catastrophic failure. For clarity, we do not highlight IQL, whose low failure mainly reflects weak post-offline performance rather than stable adaptation, or door-binary, where all methods reach zero success during online fine-tuning.

F.2 Pretrained Policy Quality Analysis with Confidence Intervals

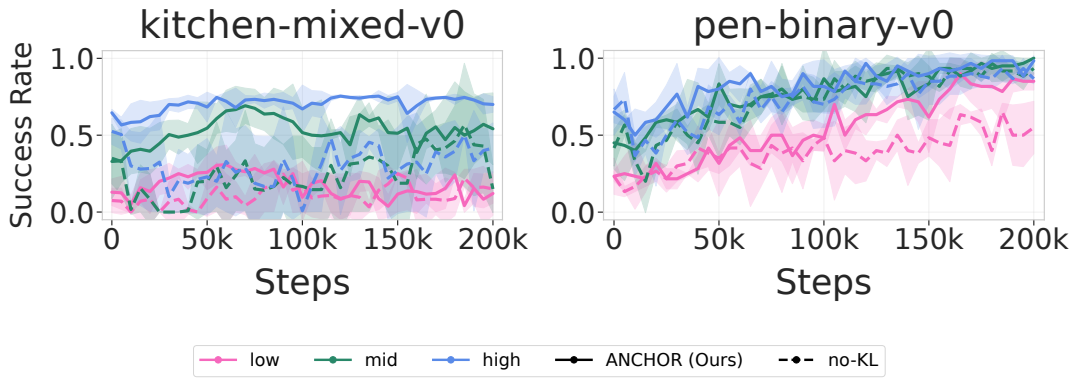


Figure F.1: **Benefit of ANCHOR depends on pretrained policy quality.** ANCHOR reduces catastrophic failure more clearly when the offline-pretrained policy starts from a higher initial success rate. Curves with the same color correspond to the same pretrained policy. Results are averaged over three seeds.

Figure F.1 reports the pretrained-policy-quality analysis from Section 5.5 with full confidence intervals. The results are consistent with the interpretation of Figure 5.6: when the pretrained policy starts from a higher initial success rate, the gap between ANCHOR and the no-KL variant becomes clearer. This shows that ANCHOR reduces catastrophic failure more effectively when there is useful pretrained behavior to preserve.

Acknowledgements

I would like to thank my advisor, collaborators, and family for their guidance and support throughout this work.