CRITIC-FREE PRETRAINING FOR EFFICIENT ONLINE REINFORCEMENT LEARNING FINE-TUNING

Daoyi Li* **Yixian Zhang*** **Chao Yu†** **Wenbo Ding†** **Yu Wang†**
Tsinghua University

*Equal contribution. †Corresponding authors.

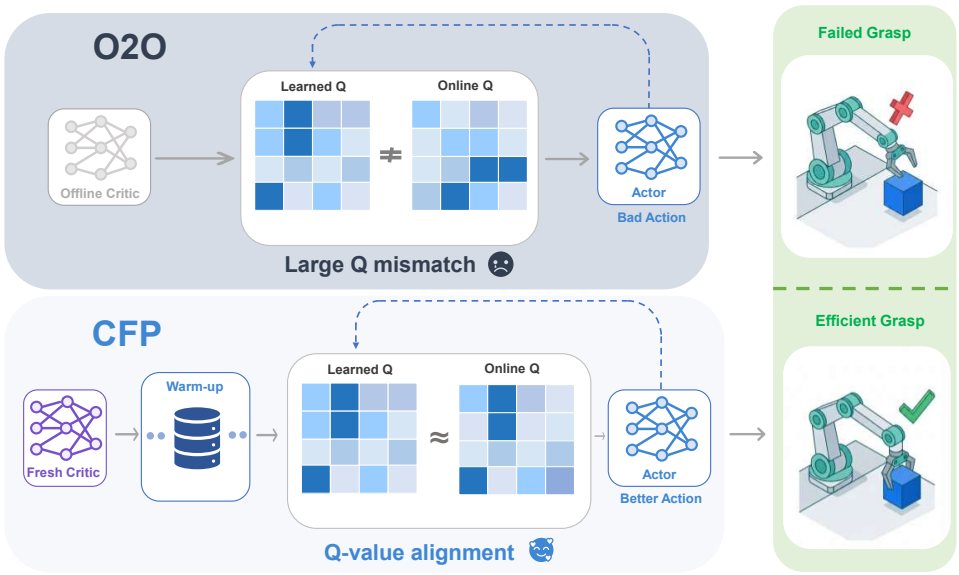


Figure 1: Overview of conventional offline-to-online fine-tuning and CFP. Conventional O2O directly reuses the offline-trained critic, which can become misaligned with the rapidly changing online policy. CFP instead retains the pretrained actor, restores a fresh critic, and calibrates it using a short warm-up before fine-tuning.

ABSTRACT

Offline-to-online (O2O) reinforcement learning aims to leverage policies pretrained on static datasets while improving them through online interaction. However, directly reusing an offline-trained critic can hinder online fine-tuning: as the policy and data distribution change rapidly, value estimates inherited from offline training may become misaligned with the online environment, leading to inaccurate policy improvement and inefficient exploration. To address this problem, we introduce **C**ritic-**F**ree **P**retraining: an efficient paradigm that completely abandons the approach of offline critic training, allowing a freshly initialized critic to adapt without inheriting biased estimates. CFP is compatible with various mainstream O2O algorithms and consistently matches or improves upon conventional O2O algorithms across a diverse set of tasks, with particularly pronounced gains on several challenging tasks.

1 INTRODUCTION

Reinforcement learning (RL) provides a general framework for learning decision-making policies through interactions with an environment (Sutton & Barto, 2018; Haarnoja et al., 2018a;b; Kaelbling

et al., 1996; Li, 2017; Arulkumaran et al., 2017). However, learning from scratch remains highly sample-inefficient in complex environments. At the beginning of training, an online agent rarely visits task-relevant regions. The agent must therefore improve its policy using low-quality experience, while the quality of future experience itself depends on the current policy. This coupling between policy learning and data collection creates a fundamental problem: without a meaningful policy, the agent cannot efficiently collect useful data, and without useful data, it cannot learn a meaningful policy.

The introduction of an offline phase significantly alleviates the pressure of exploration (Levine et al., 2020). Offline training on a dataset greatly enhances the agent's training efficiency during the online phase (Fujimoto & Gu, 2021). However, the O2O paradigm faces a critical issue: the transition from the dataset to the real-world environment often encounters significant discrepancies in data distribution, whereas the inherited critic reflects the offline behavior and data support. Its value estimates may consequently become misaligned with the evolving online policy, leading to inefficient policy updates, and even degradation of the useful behavior acquired during offline pretraining. This represents a major challenge for O2O approaches (Kumar et al., 2020; Nakamoto et al., 2023; Kostrikov et al., 2022).

In this work, we revisit the role of offline stage in O2O RL. Our insight is that the goal of offline training is not to produce a generalizable policy or value network, but rather to encourage the agent to operate in regions where it is more likely to collect effective trajectories, thereby improving sampling efficiency. This suggests that policy pretraining and critic pretraining, which are conventionally performed together, should be decoupled. Based on this insight, we propose **Critic-Free Pretraining** (CFP). Drawing inspiration from the concept of imitation learning, we utilize the dataset to train a behavior-cloning (BC) actor. Before online interaction begins, CFP retains the pretrained actor and introduces a fresh critic. A short warm-up phase then calibrates the critic. The pretrained policy provides efficient data collection, while the fresh critic avoids inheriting value bias tied to the offline distribution.

Our main contributions are listed below:

- **A critic-free offline training paradigm.** We demonstrate that training the critic during the offline phase can actually impair subsequent online fine-tuning performance. By completely eliminating offline critic training, we substantially reduce computational costs and memory requirements, improving training efficiency while providing novel insights for scaling large-scale reinforcement learning systems.
- **An effective yet simple implementation of CFP.** We conduct extensive experiments on the implementation of CFP and identify an execution approach that balances computational efficiency with experimental performance. Building on this, we designed a toy example to simulate the training workflows of O2O and CFP, intuitively demonstrating the effectiveness of CFP through visualization. This work also offers a new methodology for future research into efficient reinforcement learning.
- **Strong and broadly applicable online fine-tuning performance.** CFP achieves comparable and outstanding performance across multiple O2O algorithms and diverse benchmarks. The superior performance across different algorithms and task domains validates the effectiveness and generality of our approach.

2 Preliminaries

2.1 Reinforcement Learning

We consider reinforcement learning in a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho)$. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces; $P(s'|s, a)$ represents the transition dynamics; $r(s, a)$ is the reward function; $\gamma \in [0, 1)$ is the discount factor; and ρ is the initial state distribution. RL aims to find an policy $\pi^*(a|s)$ that maximizes the expected discounted reward:

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{1}$$

2.2 FLOW MATCHING

Flow Matching is a simple framework to train continuous normalizing flows (CNFs) (Lipman et al., 2023; 2024; Liu et al., 2023; McAllister et al., 2026; Yang et al., 2026). In contrast to denoising diffusion models (Wang et al., 2023; Ren et al., 2025; Chi et al., 2025; Kang et al., 2023; Wagenmaker et al., 2025), which aim at solving Stochastic Differential Equations (SDEs), Flow Matching model learns a time-dependent vector field $v_\theta(t, x) : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, where d is the distribution dimension, that characterizes the velocity of points moving from a noise distribution to the data distribution (Park et al., 2025b). Generation is then performed by solving the ODE:

$$\frac{dx_t}{dt} = v_\theta(t, x_t), \qquad x_0 \sim \rho_0. \tag{2}$$

2.3 FLOW-BASED POLICIES

Flow-based policy train the network via Flow Matching. The objective is to minimize the loss function:

$$L_{Flow}(\theta) = \mathbb{E}_{\substack{x_0\sim\mathcal{N}(0,I_d),\\(x_1=a,s)\sim\mathcal{D},\\t\sim\text{Unif}([0,1])}}[\|v_\theta(t, s, x_t) - (x_1 - x_0)\|_2^2] \tag{3}$$

where $\mathcal{D}$ denotes the static dataset. In inference, policies generate the action depend on the learned velocity field by computing the flow:

$$x_{t_{i+1}} = x_{t_i} + (t_{i+1} - t_i)v_\theta(t_i, s, x_{t_i}), \qquad x_{t_0} \sim \mathcal{N}(0, I). \tag{4}$$

In actual implementation, generating each action requires K-times computations of the vector field, which can lead to vanishing or exploding gradient issues during neural network training. Consequently, single-step policies $\mu_w(s, z)$ have emerged as a crucial solution. Rather than learning the velocity field directly, a single-step network learns to map to the final actions produced by K-step network. This allows actions that previously required multiple computations to be obtained in a single pass, significantly enhancing efficiency and enhancing training stability (Park et al., 2025b; Zhang et al., 2026a;b; Li et al., 2025; Li & Levine, 2026; Li et al., 2026; Doo et al., 2026). The overall loss function of flow-based policies is as below:

$$L_{Total}(\theta, \omega) = L_{Flow}(\theta) + L_\pi(\omega) \tag{5}$$

where

$$L_\pi(\omega) = \alpha\mathbb{E}_{s\sim\mathcal{D},z\sim\mathcal{N}(0,I^d)}[\|\mu_\omega(s, z) - \mu_\theta(s, z)\|_2^2] + \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\omega(\cdot|s)}[-Q_\phi(s, a)] \tag{6}$$

3 EFFICIENT ONLINE FINE-TUNING VIA CFP

We present our main methodology in this section. Our method entirely omits critic training during the offline phase, resulting in a critic-free policy-pretraining strategy. At the beginning of online fine-tuning, we initialize a fresh critic from scratch and perform a short warm-up stage on the offline dataset.

3.1 CRITIC-FREE PRETRAINING

Unlike previous O2O methods that update both the Actor and Critic simultaneously in the offline phase, we forgo training the latter in this phase. This also means that the actor's update in the offline phase does not receive gradients provided by the critic; that is, the actor's update formula is:

$$L_\pi(\omega) = \alpha\mathbb{E}_{s\sim\mathcal{D},z\sim\mathcal{N}(0,I^d)}[\|\mu_\omega(s, z) - \mu_\theta(s, z)\|_2^2] \tag{7}$$

we reach a BC actor loss function in flow policies. BC is a widely-used algorithm that reaches outstanding performance in O2O situations and robot manipulation (Bhatt et al., 2026; Torabi et al., 2018; Foster et al., 2024; Sasaki & Yamashina, 2021).

3.2 WHY A FRESH CRITIC FACILITATES ONLINE FINE-TUNING

The ability of the critic to accurately describe the true Q-values is considered key to the algorithm's success. We offer a deeper insight: since an offline critic can hinder an agent's online fine-tuning capabilities, we can achieve our objective by simply initializing the critic, rather than expending effort on calibrating it. The rationale behind this idea is that a freshly initialized critic is initially inaccurate, but it does not inherit the systematic value bias induced by prolonged fitting to the offline distribution. However, such a fresh critic may fail to provide proper guidance to the actor and could even disrupt the pretrained offline policy, leading to instability during online training. To address this, we introduce a brief warm-up phase in which the critic is trained on offline data to acquire a preliminary ability to estimate Q-values. We present the comparison between O2O and CFP in Alg.1 and Alg.2.

Algorithm 1 Traditional O2O

Init: Critic Q_ϕ, Policy π_θ, Buffer $\mathcal{B} \leftarrow \mathcal{D}$
▷ Offline Stage
for $\ell = 1 \dots L_{\text{off}}$ **do**
  Sample batch $\{(s, a, r, s')\} \sim \mathcal{B}$
  Update π_θ via offline actor loss
  Update Q_ϕ via TD loss
end for
▷ Online Stage
for $\ell = 1 \dots L_{\text{on}}$ **do**
  Collect $(s, a, r, s') \sim \pi_\theta$, append to $\mathcal{B}$
  Sample batch $\{(s, a, r, s')\} \sim \mathcal{B}$
  Update π_θ via online actor loss
  Update Q_ϕ via TD loss
end for

Algorithm 2 CFP (Ours)

Init: Policy π_θ, Buffer $\mathcal{B} \leftarrow \mathcal{D}$
▷ Offline Stage (Critic-Free)
for $\ell = 1 \dots L_{\text{off}}$ **do**
  Sample batch $\{(s, a, r, s')\} \sim \mathcal{B}$
  Update π_θ via BC Flow loss (Eq. 7)
end for
▷ **Warm-up Stage**
Init fresh critic Q_ϕ from scratch
for $\ell = 1 \dots L_{\text{warm-up}}$ **do**
  Sample batch $\{(s, a, r, s')\} \sim \mathcal{B}$
  Update π_θ via online actor loss
  Update Q_ϕ via TD loss
▷ Online Stage
for $\ell = 1 \dots L_{\text{on}}$ **do**
  Collect $(s, a, r, s') \sim \pi_\theta$, append to $\mathcal{B}$
  Sample batch $\{(s, a, r, s')\} \sim \mathcal{B}$
  Update π_θ via online actor loss
  Update Q_ϕ via TD loss
end for

Even though flow algorithms do not contain explicit penalties in loss functions, we assume that the inherent limitations of offline training are the implicit cause of the critic's pessimism. Since:

$$Q_\beta(s, a) \leq Q^*(s, a) \tag{8}$$

where $Q^*(s, a)$ is the optimal Q-value in the real environment. However, the problem extends far beyond this natural bound. In an offline dataset generated by behavior policies, the learned value function Q_β is theoretically bounded by the sub-optimality of the dataset. Modern offline datasets provide complex and sparse-reward tasks to test the effectiveness of algorithms. These datasets are overwhelmingly dominated by suboptimal, random, or failed trajectories, with high-quality successful demonstrations being exceptionally rare (Park et al., 2025a). Due to the bootstrapping nature of Temporal Difference learning, the low returns from these abundant suboptimal trajectories propagate backward throughout the state-action space, results in accumulated pessimism of the critic.

As illustrated in Figure 2, we compared the Q-values calculated by the critic for state-action pairs in a batch under O2O and CFP. We find that during the online phase of CFP, the critic's Q-value improves rapidly as fine-tuning progressed; in contrast, the Q-value in the O2O consistently remains at a lower level.

Furthermore, we identified a potential mismatch between the critic's Q evaluations and the true Q values; this discrepancy prevents the critic from effectively guiding the actor toward superior actions. Given the complexity of the benchmark environments, we could not demonstrate this deviation through visualization; instead, we designed a toy example to illustrate our insight. As shown in Box 3.2, we construct a tabular MDP with 16 states arranged on a 4×4 grid. The environment contains a negative terminal state, a positive terminal state, and several hazardous states. We deliberately

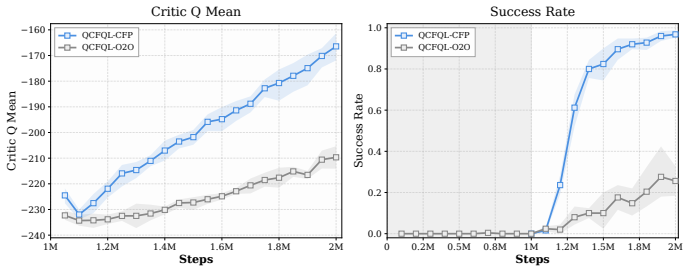
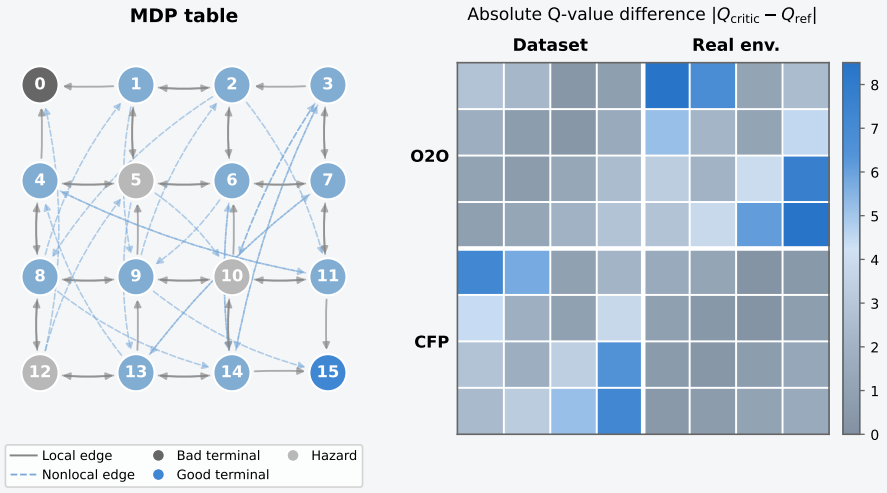


Figure 2: Critic Q-mean estimates and success rates on Cube Triple task 4. Curves and shaded regions show the mean and 95% confidence interval across five seeds, respectively.

induce a mismatch between the dataset Q distribution and the ground-truth Q distribution. We then separately simulate the training processes of O2O and CFP and evaluate the training performance of their respective critics.

Toy Example: Critic Mismatch under Distribution Shift



Schematic diagrams illustrating the MDP and Q-mismatch. The left panel represents the underlying transition dynamics within the MDP process. In the right panel, colors closer to blue indicate a greater degree of mismatch.

We evaluate each critic using the root mean squared error (RMSE) between the learned Q-values and the exact online Q-function $Q^{\tilde{\pi}_{\text{on}}}$ over all valid state transitions. As shown in Figure 3, the CFP critic exhibits a Q-value structure that more closely resembles $Q^{\tilde{\pi}_{\text{on}}}$ than the O2O critic. Moreover, CFP maintains a lower RMSE throughout online fine-tuning than O2O. These results suggest that reusing the offline critic retains considerable bias, whereas initializing critic enables more accurate estimation of the online-policy value function. For more details, see Appendix D.

4 Related Work

4.1 Offline-to-Online RL

The primary objective of the O2O RL is to leverage offline pre-training on static datasets to enable more efficient fine-tuning during the subsequent online stage. This approach mitigates the fundamental challenges where training online RL from-scratch often fails to acquire effective policies in complex environments, thereby significantly reducing both computational overhead and time costs (Levine et al., 2020; Xie et al., 2021; Guo et al., 2023; Yu & Zhang, 2023; Zhang et al., 2023; 2026b). However, O2O RL faces the problem of the distribution shift between the dataset and the

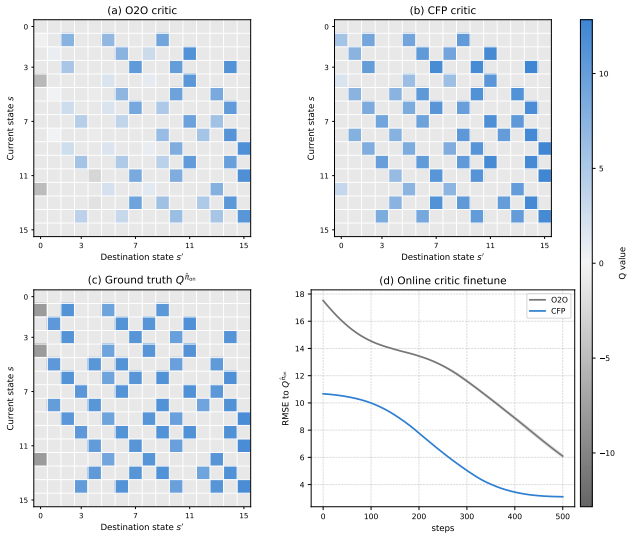


Figure 3: Critic calibration in the toy MDP. CFP produces Q value estimates closer to the ground truth $Q^{\pi_{\text{on}}}$ and achieves substantially lower RMSE than O2O during online fine-tuning.

environment, which causes catastrophic performance drop in online stage. To confront this problem, a few solutions are introduced. In this section, we list some of them that are related to our work.

Pessimism and Conservatism Early algorithms mainly focus on eliminating exploration errors. By augmenting the Bellman objective with a conservative regularizer, CQL (Kumar et al., 2020) lowers the estimated values of actions that are poorly supported by the offline dataset, thereby mitigating overestimation caused by distribution shift. Cal-QL (Nakamoto et al., 2023) constrains the learned conservative values to remain above the value of a reference policy, thereby placing the Q-values on a reasonable scale for subsequent online fine-tuning. Some algorithms reach the same goal by explicitly restrict the learned policy from deviating too far from dataset distribution. For instance, BCQ (Fujimoto et al., 2019) uses a conditional generative model and a perturbation model to restrict action selection to actions that are likely under, or close to, the dataset distribution. TD3+BC (Fujimoto & Gu, 2021) explicitly augments the actor objective with a behavior-cloning term, whereas AWAC (Nair et al., 2020) performs an advantage-weighted maximum-likelihood policy update, implicitly constraining the learned policy toward actions supported by the replay distribution.

Replay Design and Critic Calibration Beyond conservatism, recent offline-to-online RL methods improve fine-tuning through replay design and explicit value calibration. For example, RLPD (Ball et al., 2023) trains an off-policy online RL agent with minibatches drawn jointly from fixed offline data and newly collected online experience, together with high update-to-data ratios and critic regularization for stable and sample-efficient learning. WSRL (Zhou et al., 2025) introduces the warm-up stage in which only online data are used to calibrate the pretrained critic network. OPT (Shin et al., 2025) introduces a newly initialized critic and an additional online pre-training phase, during which the new critic is trained with both the offline dataset and a small set of online transitions. During fine-tuning, policy improvement is guided by a scheduled weighted combination of the offline-pretrained critic and the online-pretrained critic.

5 Experiments

To verify the effectiveness of CFP, we conduct extensive experiments on manipulation tasks under different environments. We further conducted ablation experiments, providing solid evidence for the implementation of the warm-up stage.

5.1 SETTINGS

We conduct our experiments on 8 sparse reward domains, including 5 domains from OGBench (Park et al., 2025a): `Cube-Double/Triple/Quadruple`, `Puzzle-4×4`, `Scene`, and 3 tasks in Robomimic (Mandlekar et al., 2021): `Lift`, `Can` and `Square`. We use default **play** dataset for each OGBench domain. In particular, for `Cube Quadruple`, we use 100M-size dataset released by the OGBench authors (Park et al., 2025a). In addition, we use the sparse-reward formulation for `Scene`. We use default **Multi-Human (MH)** dataset. For more details about the benchmark and dataset, see Appendix A.

5.2 BASELINES

We select 4 representative flow-based algorithms as our baselines, including **(1) FQL** (Park et al., 2025b), **(2) QC** (Ghasemipour et al., 2021; Li et al., 2026), **(3) QCFQL** (Li et al., 2026), **(4) QCFQL-nstep** (Li et al., 2026; Zhang et al., 2026b). We further implement CFP paradigm on all of them and compare the online performance among them. For more details, see Appendix B.

5.3 MAIN RESULT

Figure 4 reports the offline-to-online performance across five OGBench manipulation domains, with each curve aggregating results over five tasks. Overall, incorporating CFP consistently improves or preserves the performance of the underlying offline RL algorithms. The advantage of CFP is especially pronounced on Cube Triple, where all the CFP variants appear to outperform their corresponding O2O baselines. In particular, QCFQL-CFP and QCFQL-nstep-CFP exhibit remarkably similar learning dynamics across the five domains. Both variants rapidly improve after online fine-tuning begins and achieve nearly identical final performance on Cube Double, Cube Triple, Cube Quadruple, and Scene.

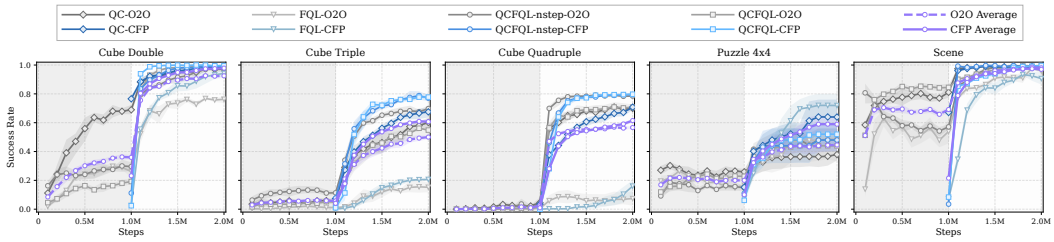


Figure 4: Performance across five tasks in each OGBench environment. Curves and shaded regions represent the mean and 95% confidence interval across five random seeds, respectively.

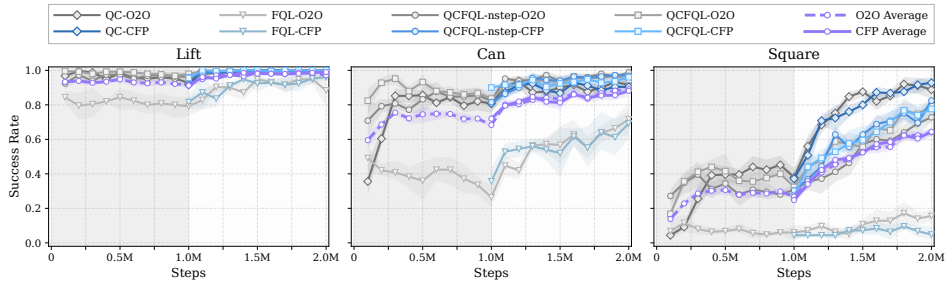


Figure 5: Performance on each Robomimic environment. Curves and shaded regions represent the mean and 95% confidence interval across five random seeds, respectively.

We also observe that the benefit of CFP depends on the underlying base algorithm. FQL-CFP achieves the largest improvement on Puzzle 4x4, reaching a final success rate of approximately 0.7, compared with less than 0.4 for FQL-O2O. Notably, FQL-CFP outperforms all QCFQL- and QC-based variants on this domain, despite being less competitive on Cube Triple and Cube Quadruple.

This indicates that CFP can complement the inductive bias of FQL particularly well in environments requiring combinatorial manipulation, rather than producing a uniform improvement across all domains. QC-CFP, by contrast, yields more moderate but consistent gains, with its clearest improvements appearing on Cube Triple and Puzzle 4x4, while remaining comparable to QC-O2O on Cube Double, Cube Quadruple, and Scene. Taken together, these results suggest that CFP is broadly compatible with different offline RL backbones, while the magnitude of its benefit is jointly determined by the base algorithm and the structural characteristics of the downstream environment.

Figure 5 reports the offline-to-online performance on three Robomimic manipulation domains. Overall, the CFP variants achieve performance comparable to their corresponding O2O baselines across most algorithms and environments. The main exception is QCFQL-nstep-CFP on Square, which exhibits a substantial improvement over QCFQL-nstep-O2O and achieves a higher final success rate. For full results, see Appendix C.2.

5.4 Ablation Study

In our ablation studies, we aim to answer the following questions:

- **How should the warm-up stage be designed?**
- **How sensitive is CFP to the number of warm-up steps?**

5.4.1 Should we allow the fresh critic to guide actor?

Since the fresh critic in CFP is initially untrained, a natural concern is that the gradients propagated through the Q-loss, $L_Q = \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\omega(\cdot|s)}[-Q_\phi(s,a)]$, may be uninformative or even harmful to the pretrained actor. This consideration suggests that, during the warm-up stage, the actor should continue to be optimized solely with the behavior cloning objective. Our experiments, however, lead to the opposite conclusion. We find that, despite the short duration of the warm-up stage—only 0.5% of the total training steps—the critic converges rapidly and already exerts a substantial influence on the actor. To investigate this effect, we compare online fine-tuning performance with and without incorporating the Q-function loss into the actor update during warm-up[1].

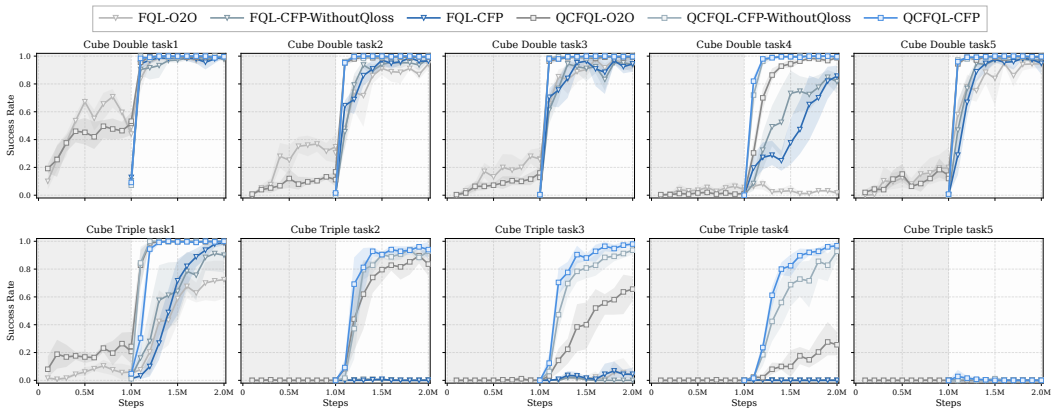


Figure 6: Ablation of the Q-loss during warm-up on OGBench. Curves and shaded regions represent the mean and 95% confidence interval across five random seeds, respectively.

Figure 6 suggests that the Q-loss backpropagating through flow networks during warm-up generally improves the subsequent online fine-tuning performance, with the clearest gains appearing on the more challenging Cube Triple tasks and Cube Double task 4.

[1]Among the algorithms considered in this paper, only QC does not apply backpropagation; therefore, the discussion in this section is restricted to the other three algorithms.

5.4.2 DOES THE NUMBER OF WARM-UP STEPS MATTER?

We view the number of steps during warm-up stage as a hyperparameter. We sweep this hyperparameter from 0 to 0.1M to see whether CFP is sensitive to it.

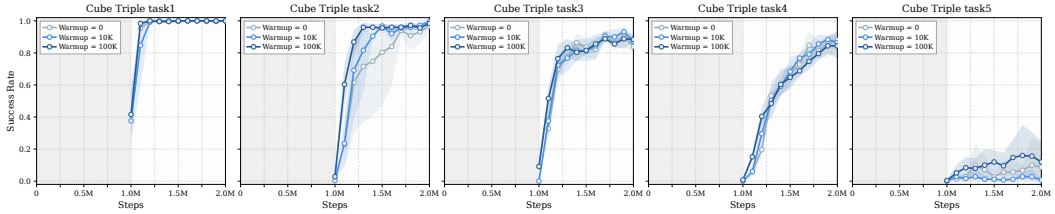


Figure 7: Ablation of the number of steps during warm-up on OGBench. The applied algorithm is QCFQL-nstep-CFP. Curves and shaded regions represent the mean and 95% confidence interval across five random seeds, respectively.

Figure 7 suggests that CFP is relatively insensitive to the length of warm-up. Warm-up-free CFP may experience unstable training during online fine-tuning. Hence, to balance stability and computation efficiency, we choose 10K as our main hyperparameter. We process our training on NVIDIA A800 GPU. In the absence of other concurrent workloads, a 10K-step warm-up requires only approximately 30 seconds, making CFP computationally efficient. For more results of ablation study, see Appendix C.3.

6 CONCLUSION AND DISCUSSION

We re-examined the training process of the Actor-Critic framework within the O2O paradigm. Our research reveals that the asymmetric training of the two components in this framework is key to achieving efficient online fine-tuning. Specifically, while the offline phase provides the actor with a sampling prior—thereby enhancing sampling efficiency during the online phase—it simultaneously introduces potential pessimism and distribution-shift-induced mismatches for the critic. We compared the mean critic outputs between O2O and CFP and designed an MDP to validate our inferences. These findings offer new insights into the role of datasets in RL training process.

However, CFP does not consistently outperform conventional O2O training on Robomimic. This limitation suggests that discarding the offline critic is most beneficial when inherited value estimates exhibit substantial distribution-induced mismatch. Developing diagnostics that predict when such mismatch occurs, and designing more effective critic-initialization and warm-up strategies, are promising directions for future work.

REFERENCES

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE signal processing magazine*, 34(6):26–38, 2017.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

Dwait Bhatt, Shih-Chieh Chou, and Nikolay Atanasov. Rainbow-demorl: Combining improvements in demonstration-augmented reinforcement learning. *arXiv preprint arXiv:2603.27400*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

JaeHyeok Doo, Byeongguk Jeon, Seonghyeon Ye, Kimin Lee, and Minjoon Seo. Q-flow: Stable and expressive reinforcement learning with flow-based policy. In *International Conference on Machine Learning*, 2026.

Dylan J Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning. *Advances in Neural Information Processing Systems*, 37:120602–120666, 2024.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062, 2019.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pp. 3682–3691. PMLR, 2021.

Siyuan Guo, Lixin Zou, Hechang Chen, Bohao Qu, Haotian Chi, Philip S Yu, and Yi Chang. Sample efficient offline-to-online reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, 36(3):1299–1310, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018b. URL http://arxiv.org/abs/1812.05905.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

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

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vd4eNAdtO6.

Qiyang Li, Seohong Park, and Sergey Levine. Decoupled q-chunking, 2025. URL https://arxiv.org/abs/2512.10926.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. *Advances in Neural Information Processing Systems*, 38:55518–55553, 2026.

Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *Conference on Learning Representations(ICLR)*, 2023.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *Conference on Learning Representations(ICLR)*, 2023.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eoEmoKoQpJ.

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

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning(ICML)*, 2025b.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

Fumihiro Sasaki and Ryota Yamashina. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=zrT3HcsWSAt.

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 *International Conference on Machine Learning (ICML) 2025*, 2025.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018. ISBN 978-0-262-03924-6.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4950–4957, 2018.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *Conference on Robot Learning*, 2025.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in neural information processing systems*, 34:27395–27407, 2021.

Shunpeng Yang, Ben Liu, and Hua Chen. Policyflow: Policy optimization with continuous normalizing flow in reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=YETCQLcKtn.

Zishun Yu and Xinhua Zhang. Actor-critic alignment for offline-to-online reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40452–40474. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/yu23k.html.

Haichao Zhang, Wei Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

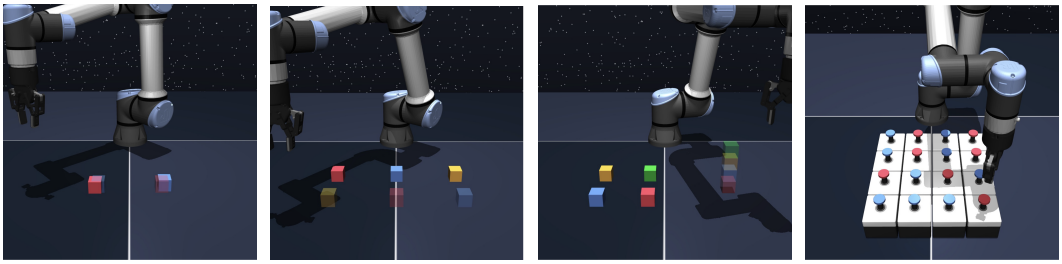
Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026a. URL https://openreview.net/forum?id=ACagRwCCqu.

Yixian Zhang, Shu'ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. SAC flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=zZvWj4JrYj.

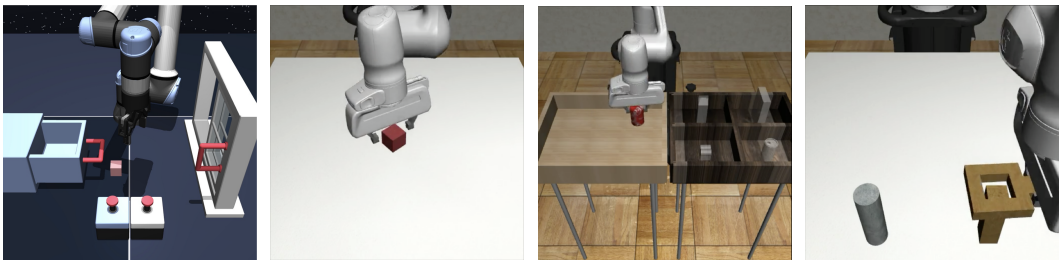
Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. In *International Conference on Learning Representations(ICLR)*, 2025. URL https://openreview.net/forum?id=HN0CYZbAPw.

A Environment

In this section, we present the domain we use in our experiments in detail. Table 1 summarizes some information of each domain, including **Reward Type, Dataset Size, Episode Length and Action Dimension.**



(a) Cube Double (b) Cube Triple (c) Cube Quadruple (d) Puzzle 4×4



(e) Scene (f) Lift (g) Can (h) Square

Figure 8: Visualizations of the environments. Subfigures (a-e) show the OGBench domains. Subfigures (f-h) depict the manipulation tasks from Robomimic.

A.1 OGBench

We choose 5 OGBench manipulation domains and use the single-task version of it. We use the standard **play** version of the dataset, in which trajectories were collected using a non-Markovian expert policy. For `Cube Double/Triple/Quadruple`, the agents need to manipulate an arm and learn a pick-and-place skill to place colored cube blocks to the right locations. Particularly, `Cube Quadruple` is an extremely challenging domain in which the agent must process four cubes

in sequence. It is almost impossible to learn an effective policy under standard dataset. Hence, we use the 100M-size dataset provided officially. `Puzzle 4×4` requires solving the "Lights Out" puzzle with a robot arm. For `Scene-sparse`, we use the sparse-reward mode of the reward function such that the agent receives a reward of 0 only upon completing the task, and -1 in all other cases.

Table 1: Specifications of the environments used in our experiments.

Environment	Reward Type	Dataset Size	Episode Length	Action Dimension
OGBench				
cube-double	Sparse	1M	500	5
cube-triple	Sparse	3M	1000	5
cube-quadruple	Sparse	100M	1000	5
puzzle-4x4	Sparse	1M	500	5
scene	Sparse	1M	750	5
Robomimic				
lift	Sparse	31,127	500	7
can	Sparse	62,756	500	7
square	Sparse	80,731	500	7

A.2 Robomimic

We select 3 challenging domain from Robomimic, including `Lift, Can and Square`. We conduct our experiments on Multi-Human datasets. These datasets were collected by 6 operators, including 300 successful trajectories. `Lift` requires the robot arm to pick up a cube. `Can` requires the robot arm to pick up a can and place it into a container. `Square` requires the robot arm to pick up a square sleeve and fit it onto a square post.

B Implementation

In this section, we present the details of our implementation of CFP on all the baselines.

Flow Q-Learning. FQL is a strong baseline for offline reinforcement learning. Its main innovation is to exploit the expressive power of a multi-step flow network to approximate the behavior distribution in the offline dataset, while using a Q-loss to propagate gradients from the critic to the actor. Directly backpropagating through the multi-step flow generation process, however, may lead to unstable gradients. To avoid this issue, FQL introduces a one-step noise-conditioned policy that directly distills the actions generated by the multi-step behavior flow. Although the one-step policy may sacrifice some of the expressive capacity of the original multi-step flow, it enables more stable policy optimization.

More concretely, FQL consists of the following three networks:

1. a behavior flow policy $\mu_\theta(s, z)$, which generates an action from a state s and Gaussian noise $z \sim \mathcal{N}(0, I^d)$ through multi-step flow integration;
2. a one-step noise-conditioned policy $\mu_\omega(s, z)$, which distills the output of the behavior flow policy; and
3. a critic $Q_\phi(s, a)$, which estimates the expected return of a state–action pair.

The one-step policy is optimized by combining flow-policy distillation with Q-value maximization:

$$L_\pi(\omega) = \alpha \mathbb{E}_{\substack{s\sim\mathcal{D},\\ z\sim\mathcal{N}(0,I^d)}} \left[\|\mu_\omega(s, z) - \mu_\theta(s, z)\|_2^2 \right] + \mathbb{E}_{\substack{s\sim\mathcal{D},\\ a\sim\pi_\omega(\cdot|s)}} \left[-Q_\phi(s, a) \right], \tag{9}$$

where α controls the strength of behavior regularization. The first term distills the output of the multi-step behavior flow into the one-step policy, whereas the second term propagates the critic gradient to the one-step actor.

The critic is trained with the temporal-difference objective

$$L_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\phi(s,a) - \left[r + \gamma Q_{\bar{\phi}}(s', a')\right]\right)^2\right], \tag{10}$$

where $a' \sim \pi_\omega(\cdot \mid s')$ and $Q_{\bar{\phi}}$ denotes the target critic.

QCFQL. QCFQL extends FQL to temporally extended action spaces by jointly generating and evaluating an action chunk $\mathbf{a}_t = (a_t, \ldots, a_{t+h-1})$. The QCFQL actor objective is

$$L_\pi(\omega) = \alpha\mathbb{E}_{\substack{s_t\sim\mathcal{D},\\ z\sim\mathcal{N}(0,I^{hd})}}\left[\|\mu_\omega(s_t,z) - \mu_\theta(s_t,z)\|_2^2\right] + \mathbb{E}_{\substack{s_t\sim\mathcal{D},\\ \mathbf{a}_t\sim\pi_\omega(\cdot|s_t)}}\left[-Q_\phi(s_t,\mathbf{a}_t)\right]. \tag{11}$$

The QCFQL critic is trained using an h-step temporal-difference target:

$$L_Q(\phi) = \mathbb{E}\left[\left(Q_\phi(s_t,\mathbf{a}_t) - \left[r_t^{(h)} + \gamma^h Q_{\bar{\phi}}(s_{t+h},\mathbf{a}_{t+h})\right]\right)^2\right], \tag{12}$$

where

$$r_t^{(h)} = \sum_{j=0}^{h-1}\gamma^j r_{t+j}, \qquad \mathbf{a}_{t+h} \sim \pi_\omega(\cdot \mid s_{t+h}). \tag{13}$$

QCFQL-nstep. QCFQL-nstep is a similar implementation of QCFQL. Instead of distilling the multi-step network with single-step network, we use direct backpropagation. The QCFQL-nstep actor objective is

$$L_\pi(\theta) = \alpha\mathbb{E}_{\substack{(s_t,\mathbf{a}_t)\sim\mathcal{D},\\ z\sim\mathcal{N}(0,I^{hd})}}\left[\|\mu_\theta(s_t,z) - \mathbf{a}_t\|_2^2\right] + \mathbb{E}_{\substack{s_t\sim\mathcal{D},\\ \mathbf{a}_t\sim\pi_\theta(\cdot|s_t)}}\left[-Q_\phi(s_t,\mathbf{a}_t)\right]. \tag{14}$$

The remaining components of QCFQL-nstep are identical to those of QCFQL.

QC. QC uses the same flow-matching behavior policy μ_θ as FQL, but does not introduce an additional one-step noise-conditioned actor. Instead, QC implicitly defines its policy through best-of-N candidate selection. Specifically, for a given state s, we first sample N independent Gaussian noise vectors,

$$z^1, z^2, \ldots, z^N \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I^{hd}), \tag{15}$$

and generate N candidate actions using the multi-step behavior flow:

$$\mathbf{a}_t^i = \mu_\theta(s_t, z^i), \qquad i \in \{1,\ldots,N\}. \tag{16}$$

The policy then selects the candidate with the largest estimated Q-value:

$$i^* = \arg\max_{i\in\{1,\ldots,N\}} Q_\phi(s_t,\mathbf{a}^i), \qquad \pi_{\text{QC}}(s) = \mathbf{a}^{i^*}. \tag{17}$$

Unlike FQL and QCFQL, QC does not backpropagate the critic gradient. The selected best-of-N action is directly used to construct the temporal-difference target. Accordingly, the critic is optimized by

$$L_Q(\phi) = \mathbb{E}_{(s,\mathbf{a}_t,r,s')\sim\mathcal{D}}\left[\left(Q_\phi(s_t,\mathbf{a_t}) - \left[r_t^{(h)} + \gamma^h Q_{\bar{\phi}}\left(s_{t+h}, \mathbf{a}_{t+h}^{i^*}\right)\right]\right)^2\right], \tag{18}$$

where $\mathbf{a}_{\mathbf{t+h}}^{\mathbf{i^*}} \sim \pi_{QC}(\cdot|s_{t+h})$.

Next, we demonstrate how we incorporate CFP into the original algorithm.

FQL-CFP. Since the implementations of all the algorithms are largely similar, we primarily describe how FQL-CFP is implemented.

The objective of FQL-CFP during the offline stage is:

$$L_\pi(\omega) = \alpha \mathbb{E}_{\substack{s\sim\mathcal{D},\\ z\sim\mathcal{N}(0,I^d)}} \left[\|\mu_\omega(s,z) - \mu_\theta(s,z)\|_2^2 \right]. \tag{19}$$

CFP does not present critic updates during the offline stage.

During the warm-up phase, the actor and the initialized critic undergo brief updates and calibration using the offline dataset, with the following update objective:

$$L_\pi(\omega) = \alpha \mathbb{E}_{\substack{s\sim\mathcal{D},\\ z\sim\mathcal{N}(0,I^d)}} \left[\|\mu_\omega(s,z) - \mu_\theta(s,z)\|_2^2 \right] - \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi_\omega(\cdot|s)}[Q_\phi(s,a)]. \tag{20}$$

The critic is trained with the temporal-difference objective

$$L_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[\left(Q_\phi(s,a) - \left[r + \gamma Q_{\bar{\phi}}(s',a') \right] \right)^2 \right], \tag{21}$$

where $a' \sim \pi_\omega(\cdot \mid s')$ and terminal transitions are masked in the bootstrap target. During the online phase, the agent samples trajectories in the real environment and stores them in the batch. The objectives are:

$$L_\pi(\omega) = \alpha \mathbb{E}_{\substack{s\sim\mathcal{B},\\ z\sim\mathcal{N}(0,I^d)}} \left[\|\mu_\omega(s,z) - \mu_\theta(s,z)\|_2^2 \right] - \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi_\omega(\cdot|s)}[Q_\phi(s,a)], \tag{22}$$

$$L_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}} \left[\left(Q_\phi(s,a) - \left[r + \gamma Q_{\bar{\phi}}(s',a') \right] \right)^2 \right]. \tag{23}$$

Since the objectives for the warm-up and online phases are identical—differing only in the source of the updates (the warm-up phase relies entirely on the dataset, whereas the online phase utilizes batches containing online data)—no further distinction will be made between the two.

QCFQL-CFP. The update method is very similar to that of FQL-CFP; QCFQL simply incorporates an action chunk. Its update objective during the offline stage is

$$L_\pi(\omega) = \alpha \mathbb{E}_{\substack{s_t\sim\mathcal{D},\\ z\sim\mathcal{N}(0,I^{hd})}} \left[\|\mu_\omega(s_t,z) - \mu_\theta(s_t,z)\|_2^2 \right] \tag{24}$$

For both warm-up and online stage, the targets are:

$$L_\pi(\omega) = \alpha \mathbb{E}_{\substack{s_t\sim\mathcal{D},\\ z\sim\mathcal{N}(0,I^{hd})}} \left[\|\mu_\omega(s_t,z) - \mu_\theta(s_t,z)\|_2^2 \right] + \mathbb{E}_{\substack{s_t\sim\mathcal{D},\\ \mathbf{a}_t\sim\pi_\omega(\cdot|s_t)}} \left[-Q_\phi(s_t,\mathbf{a}_t) \right], \tag{25}$$

$$L_Q(\phi) = \mathbb{E}\left[\left(Q_\phi(s_t,\mathbf{a}_t) - \left[r_t^{(h)} + \gamma^h Q_{\bar{\phi}}(s_{t+h},\mathbf{a}_{t+h}) \right] \right)^2 \right]. \tag{26}$$

QCFQL-nstep-CFP. QCFQL-nstep-CFP has a similar implementation. The QCFQL-nstep actor objective during offline phase is:

$$L_\pi(\theta) = \alpha \mathbb{E}_{\substack{(s_t,\mathbf{a}_t)\sim\mathcal{D},\\ z\sim\mathcal{N}(0,I^{hd})}} \left[\|\mu_\theta(s_t,z) - \mathbf{a}_t\|_2^2 \right] \tag{27}$$

For warm-up and online updates, the objectives are:

$$L_\pi(\theta) = \alpha \mathbb{E}_{\substack{(s_t,\mathbf{a}_t)\sim\mathcal{D},\\ z\sim\mathcal{N}(0,I^{hd})}} \left[\|\mu_\theta(s_t,z) - \mathbf{a}_t\|_2^2 \right] + \mathbb{E}_{\substack{s_t\sim\mathcal{D},\\ \mathbf{a}_t\sim\pi_\theta(\cdot|s_t)}} \left[-Q_\phi(s_t,\mathbf{a}_t) \right], \tag{28}$$

$$L_Q(\phi) = \mathbb{E}\left[\left(Q_\phi(s_t,\mathbf{a}_t) - \left[r_t^{(h)} + \gamma^h Q_{\bar{\phi}}(s_{t+h},\mathbf{a}_{t+h}) \right] \right)^2 \right]. \tag{29}$$

QC-CFP. QC naturally shares the same training approach as CFP—specifically, using only the actor network to approximate the velocity field. So its training objective is:

$$L_\pi(\theta) = \mathbb{E}_{\substack{x_0\sim\mathcal{N}(0,I_{hd}),\\(x_1=a,s)\sim\mathcal{D},\\t\sim\text{Unif}([0,1])}}[\|v_\theta(t,s,x_t) - (x_1 - x_0)\|_2^2]. \tag{30}$$

For warm-up and online training, the objectives are:

$$L_\pi(\theta) = \mathbb{E}_{\substack{x_0\sim\mathcal{N}(0,I_{hd}),\\(x_1=a,s)\sim\mathcal{D},\\t\sim\text{Unif}([0,1])}}[\|v_\theta(t,s,x_t) - (x_1 - x_0)\|_2^2], \tag{31}$$

$$L_Q(\phi) = \mathbb{E}_{(s,\mathbf{a}_t,r,s')\sim\mathcal{D}}\left[\left(Q_\phi(s_t,\mathbf{a_t}) - \left[r_t^{(h)} + \gamma^h Q_{\bar{\phi}}\left(s_{t+h}, \mathbf{a}_{t+h}^{i^*}\right)\right]\right)^2\right], \tag{32}$$

where $\mathbf{a}_{\mathbf{t+h}}^{\mathbf{i^*}}$ follows the Best-of-N sampling.

C Experiment

C.1 Hyperparameters

Unless otherwise specified, we use the same hyperparameters across all environments and algorithms. The common hyperparameters are summarized in Table 2.

Table 2: Common hyperparameters used in our experiments.

Parameter	**Value**
Batch size (M)	256
Discount factor (γ)	0.99
Optimizer	Adam
Learning rate	3×10^{-4}
Target network update rate (τ)	5×10^{-3}
Critic ensemble size	2
Number of flow integration steps (T)	4 for QCFQL-nstep; 10 otherwise
Action chunk length (h)	5
Number of offline training steps	1×10^6
Number of online environment steps	1×10^6
Network width	512
Network depth	4 hidden layers
Number of best-of-N candidates (QC)	32

For specific hyperparameters of α value for backpropagation algorithms and N for Best-of-N algorithms, see Table 3. For hyperparameters of warm-up steps, see Table 4.

Table 3: Task-specific hyperparameters for all O2O and CFP variants.

Environment	Task	QC (N)		FQL (α)		QCFQL-nstep (α)		QCFQL (α)	
		O2O	CFP	O2O	CFP	O2O	CFP	O2O	CFP
Cube Double	Task 1	32	32	300	300	100	100	100	100
	Task 2	32	32	300	300	100	100	100	100
	Task 3	32	32	300	300	100	100	100	100
	Task 4	32	32	300	300	100	100	100	100
	Task 5	32	32	300	300	100	100	100	100
Cube Triple	Task 1	32	32	300	300	100	100	100	100
	Task 2	32	32	300	300	100	100	100	100
	Task 3	32	32	300	300	100	100	100	100
	Task 4	32	32	300	300	100	100	100	100
	Task 5	32	32	300	300	100	100	100	100
Cube Quadruple	Task 1	32	32	300	300	100	100	100	100
	Task 2	32	32	300	300	100	100	100	100
	Task 3	32	32	300	300	100	100	100	100
	Task 4	32	32	300	300	100	100	100	100
	Task 5	32	32	300	300	100	100	100	100
Puzzle 4×4	Task 1	64	64	1000	1000	1000	1000	1000	1000
	Task 2	64	64	1000	1000	1000	1000	1000	1000
	Task 3	64	64	1000	1000	1000	100	1000	100
	Task 4	64	64	1000	1000	1000	1000	1000	1000
	Task 5	64	64	1000	1000	1000	1000	1000	1000
Scene	Task 1	32	32	300	300	300	300	300	300
	Task 2	32	32	300	300	300	300	300	300
	Task 3	32	32	300	300	300	300	300	300
	Task 4	32	32	300	300	300	300	300	300
	Task 5	32	32	300	300	300	300	300	300
Lift	–	16	16	10000	10000	10000	10000	10000	10000
Can	–	16	16	10000	10000	10000	10000	10000	10000
Square	–	16	16	10000	10000	10000	10000	10000	10000

Table 4: Task-specific critic warm-up steps for the CFP variants.

Environment	Task	QC-CFP	FQL-CFP	QCFQL-nstep-CFP	QCFQL-CFP
Cube Double	Task 1	10000	10000	10000	10000
	Task 2	10000	10000	10000	10000
	Task 3	10000	10000	10000	10000
	Task 4	10000	10000	10000	10000
	Task 5	10000	10000	10000	10000
Cube Triple	Task 1	10000	10000	10000	10000
	Task 2	10000	10000	10000	10000
	Task 3	10000	10000	10000	10000
	Task 4	10000	10000	10000	10000
	Task 5	10000	10000	0.1M	10000
Cube Quadruple	Task 1	0.2M	0.2M	0.2M	0.2M
	Task 2	0.2M	0.2M	0.2M	0.2M
	Task 3	0.2M	0.2M	0.2M	0.2M
	Task 4	0.2M	0.2M	0.2M	0.2M
	Task 5	0.2M	0.2M	0.2M	0.2M
Puzzle 4×4	Task 1	10000	10000	10000	10000
	Task 2	10000	10000	10000	10000
	Task 3	50000	50000	50000	50000
	Task 4	10000	50000	10000	10000
	Task 5	10000	50000	10000	10000
Scene	Task 1	10000	30000	10000	10000
	Task 2	10000	30000	10000	10000
	Task 3	10000	30000	10000	10000
	Task 4	10000	30000	10000	10000
	Task 5	10000	30000	10000	10000
Lift	Task 1	10000	5000	10000	10000
Can	Task 1	10000	5000	10000	10000
Square	Task 1	10000	5000	10000	5000

C.2 Full Results

OGBench. Figure 9 shows the full results on OGBench.

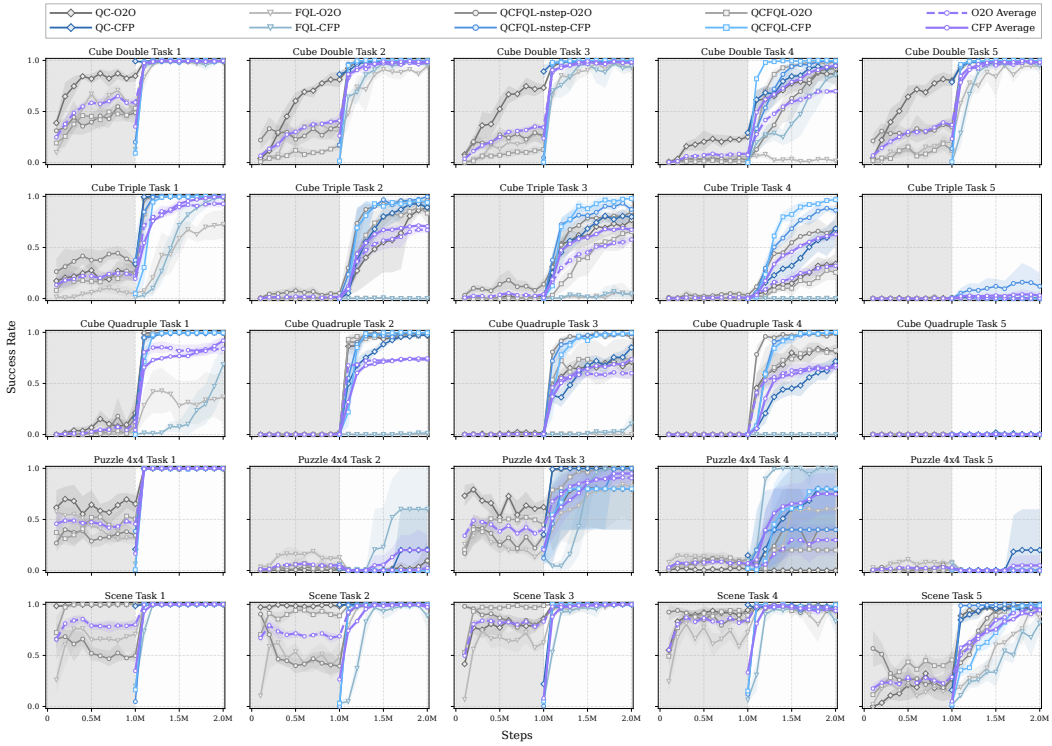


Figure 9: Full OGBench results by tasks. Curves and shaded regions represent the mean and 95% confidence interval across five random seeds, respectively.

Robomimic. Figure 5 shows the full results on Robomimic.

Table 5 shows the detailed performance comparison between O2O and CFP. Almost all CFP variants reach performance comparable to or even better than those of the O2O variants.

Table 5: Detailed mean performance comparison (%) by task. All highlighted cells correspond to CFP. **Blue-shaded** cells indicate that CFP significantly outperforms O2O; **light-blue-shaded** cells indicate minor variations or no significant difference; and **gray-shaded** cells indicate that O2O significantly outperforms CFP.

Environment	Task	QC		FQL		QCFQL-nstep		QCFQL	
		O2O	CFP	O2O	CFP	O2O	CFP	O2O	CFP
Cube Double	Task 1	84.8→100	99.2→100	48.8→100	12.8→100	48.8→100	20.0→100	53.2→100	9.2→100
	Task 2	81.2→100	86.4→100	30.4→97.2	0→100	36.0→100	17.2→100	16.8→100	1.6→100
	Task 3	73.2→100	89.2→100	25.6→98.8	0→100	26.0→100	4.8→100	12.8→100	0.4→100
	Task 4	25.2→90.0	29.2→96.4	5.6→9.2	0→89.2	2.4→94.4	0.4→99.6	1.2→100	0→100
	Task 5	80.4→100	78.8→100	20.0→97.6	0→100	34.8→100	13.6→100	12.0→100	0.8→100
Cube Triple	Task 1	25.2→100	34.0→100	5.2→78.8	1.2→99.2	33.2→100	37.6→100	24.4→100	4.8→100
	Task 2	0.8→91.2	0→96.0	0→1.2	0→1.2	5.2→97.6	0.4→100	0→92.0	0→98.4
	Task 3	2.0→80.4	2.8→84.0	0.4→8.4	0→13.2	14.0→87.6	0→95.6	0→68.8	0→99.6
	Task 4	0→38.0	0→68.4	0→0.4	0→0.4	3.2→75.6	0→91.6	0→30.8	0→98.4
	Task 5	0→2.4	0→1.6	0→0	0→0	1.2→2.4	0.4→25.6	0→0	0→4.0
Cube Quadruple	Task 1	21.2→100	16.0→100	2.4→48.8	0→68.4	16.8→100	1.6→100	8.4→100	0→100
	Task 2	0→99.6	0.4→98.8	0→0.4	0→2.8	2.4→100	0→100	0→99.6	0→100
	Task 3	1.6→76.8	1.2→86.4	0→2.0	0→10.8	0→100	0.4→100	0→80.0	0→100
	Task 4	0→89.6	0→71.6	0→0	0→0	0→99.6	0→100	0→86.8	0→100
	Task 5	0→0	0→2.4	0→0	0→0	0→0	0→0	0→0	0→0
Puzzle 4x4	Task 1	65.2→100	20.8→100	32.0→100	6.4→100	34.4→100	38.4→100	50.4→100	0.8→100
	Task 2	0.8→10.0	3.2→22.8	12.8→12.8	2.8→62.0	0.8→0.8	1.6→1.6	6.4→6.8	5.2→5.2
	Task 3	62.0→90.0	35.2→100	20.4→84.8	13.2→100	34.4→100	12.0→100	44.8→100	21.6→82.0
	Task 4	1.2→1.2	14.8→85.2	9.6→64.4	9.2→100	4.0→42.0	2.0→40.4	12.0→29.2	2.0→80.4
	Task 5	0.4→0.4	2.0→22.4	8.0→8.0	7.2→7.2	0.4→0.4	1.2→1.2	0.4→0.4	1.2→1.2
Scene	Task 1	100→100	98.4→100	70.8→100	20.0→100	49.6→100	4.8→100	100→100	16.4→100
	Task 2	98.0→100	99.6→100	48.4→100	3.2→100	39.6→100	0→100	88.8→100	3.2→100
	Task 3	86.0→100	22.0→100	62.4→100	4.4→100	85.6→100	0.4→100	98.8→100	5.2→100
	Task 4	94.4→100	99.6→100	72.4→100	6.4→100	83.6→100	12.0→100	89.2→100	15.2→100
	Task 5	27.2→99.6	16.0→99.6	15.6→96.0	1.6→85.2	27.2→100	0.4→100	45.6→100	2.0→100
Lift	–	92.8→100	91.6→100	79.2→98.8	81.6→98.4	96.4→100	96.0→100	98.0→100	95.6→100
Can	–	80.8→97.2	80.8→96.8	26.4→77.6	35.6→76.4	82.4→99.6	81.6→99.6	83.6→98.8	90.0→97.6
Square	–	38.0→94.4	37.2→94.0	6.0→23.2	4.4→12.4	30.8→73.2	30.4→82.4	32.4→82.8	27.2→80.4

C.3 Ablation Study

We present more results of the ablation study here. Figure 10 further confirms our conclusion that CFP is insensitive to the steps of warm-up stage.

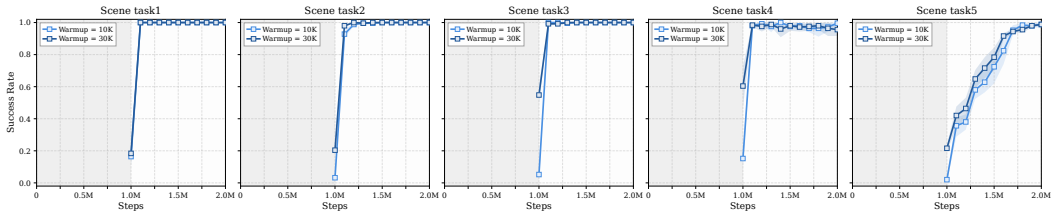


Figure 10: Ablation of the number of steps during warm-up on OGBench. The applied algorithm is QCFQL-CFP. Curves and shaded regions represent the mean and 95% confidence interval across five random seeds, respectively.

We also investigate CFP's sensitivity to warm-up steps in the absence of Q-loss during the warm-up phase. Figure 11 shows that CFP without Q-loss during warm-up stage is more sensitive to the length of warm-up steps. Overall, the longer the warm-up period, the worse the performance of online fine-tuning of CFP. This result is significantly different from the conclusion we make in section 5.4.2. We examined the corresponding mean Q-values and find that the Q mean assigned by the critic decreased as the warm-up length increased. We attribute this phenomenon to the interplay within the actor-critic architecture. We have demonstrated that a fresh critic can provide guidance even during the warm-up phase, enabling the actor to output higher-value actions; conversely, since the target action in the critic's update formula is provided by the actor, higher-quality actions from the actor result in higher action ratings from the critic. Without the inclusion of Q-loss, the actor remains confined to the low-quality actions present in the dataset, producing progressively lower critic estimates of state-action pairs. Our research further confirms that offline datasets—characterized by low environmental coverage and limited trajectory success rates—exacerbate the critic's pessimism, thereby hindering generalization during the online phase.

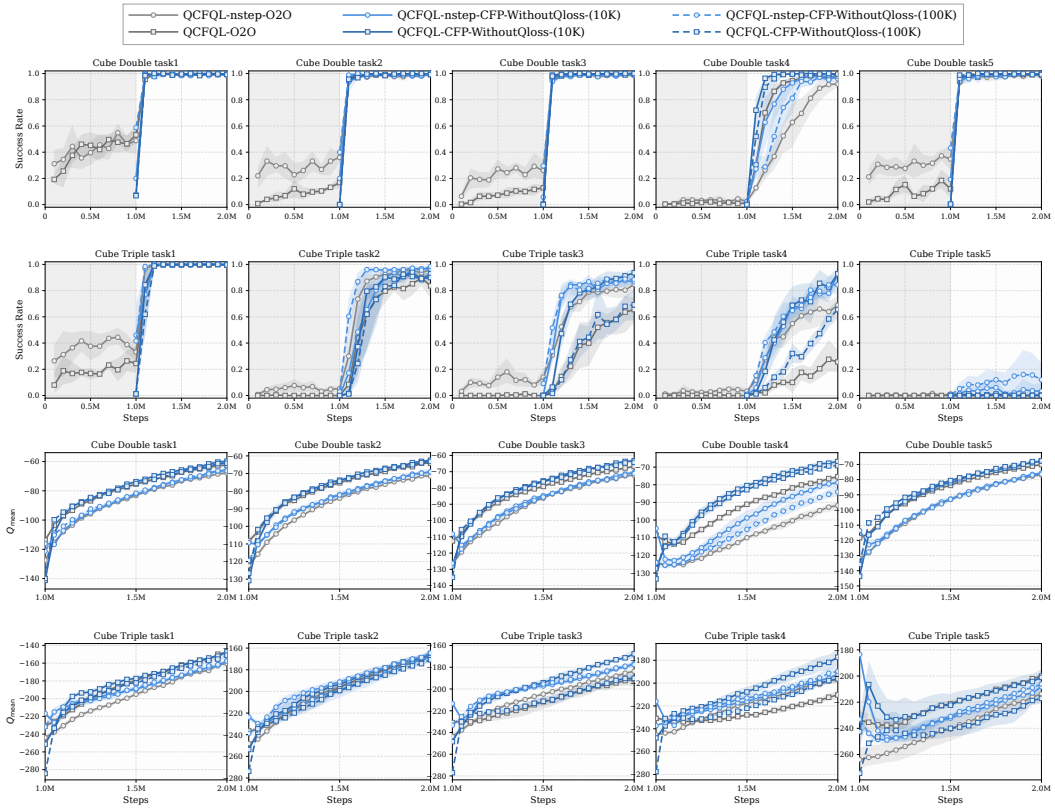


Figure 11: Ablation study of the sensitivity of CFP (without Q-loss) to warm-up steps.

D TOY EXAMPLE

In this section, we introduce the toy example we implement in Section 3.2 in detail.

We designed an MDP table and simulated the complete O2O phase as well as the CFP implementation during the actual training process. In our experiments, we rigorously controlled the variables associated with O2O and CFP, ensuring that the sole difference between the two was whether the critic was initialized during the online phase.

D.1 ENVIRONMENT

The environment is a finite MDP with 16 states. For visualization, the states are placed on a 4×4 canvas,

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15, \end{array} \tag{33}$$

but these coordinates do not define the transition dynamics. The dynamics are instead specified by an irregular directed graph. An action at state s selects one of its valid destination states $s' \in \mathcal{N}^+(s)$; therefore the action-value function is represented as a matrix $Q(s, s')$. The graph includes both local edges and nonlocal directed shortcuts, and the existence of $s \to s'$ does not imply the existence of $s' \to s$. Table 6 provides specific information. The figure in toy example 3.2 also demonstrates the same dynamics in a visual way.

Table 6: Possible transitions of the sixteen-state directed MDP.

s	$\mathcal{N}^+(s)$	s	$\mathcal{N}^+(s)$
0	$\varnothing$	8	$\{1, 4, 9, 12, 14\}$
1	$\{0, 2, 5, 9\}$	9	$\{2, 5, 8, 10, 15\}$
2	$\{1, 6, 8, 11\}$	10	$\{3, 6, 9, 11, 14\}$
3	$\{2, 7, 10, 14\}$	11	$\{4, 7, 10, 15\}$
4	$\{0, 5, 8, 11\}$	12	$\{0, 5, 8, 13\}$
5	$\{1, 4, 6, 10, 13\}$	13	$\{4, 7, 9, 12, 14\}$
6	$\{2, 5, 7, 9, 14\}$	14	$\{3, 6, 10, 13, 15\}$
7	$\{3, 6, 11, 13\}$	15	$\varnothing$

State 0 is a bad terminal with reward -8, whereas state 15 is a good terminal with reward $+12$. Every other transition has a base reward of -0.05. Entering one of the hazard states $\mathcal{H} = \{5, 10, 12\}$ incurs an additional penalty of -1.25. Several directed shortcuts also have edge-specific costs, listed in Table 7. Thus, for a nonterminal destination,

$$r(s, s') = -0.05 - 1.25\,\mathbb{I}[s' \in \mathcal{H}] + c(s, s'), \tag{34}$$

while transitions into states 0 and 15 receive their terminal rewards. The discount factor is $\gamma = 0.97$.

Table 7: Edge-specific costs $c(s, s')$.

Edge	Cost	Edge	Cost	Edge	Cost
$1 \to 9$	-0.70	$2 \to 11$	$+0.15$	$3 \to 14$	-0.40
$4 \to 11$	-0.20	$5 \to 13$	-0.60	$6 \to 14$	$+0.10$
$7 \to 13$	-0.30	$8 \to 1$	-0.50	$9 \to 2$	-0.40
$10 \to 3$	-0.20	$11 \to 4$	-0.60	$12 \to 5$	-0.25
$13 \to 4$	-0.35	$14 \to 3$	-0.45		

To simulate the inability of the offline dataset to provide optimal or suboptimal action coverage in complex environments, we designed the offline dataset to tend to reach the bad terminal. Specif-

ically, for every nonterminal state, we compute the shortest directed distance from each valid successor to state 0. A probability mass of 0.985 is distributed among the successors with minimum distance to state 0, while the remaining mass is distributed uniformly over all valid successors. This policy generates an offline dataset whose dominant trajectories terminate at state 0.

Furthermore, to simulate a policy that more closely resembles the one operating in the real world environment—rather than the dataset policy—after the actor has undergone fine-tuning during the online phase, we implemented an online sampling strategy. Specifically, the prescribed online behavior is an ϵ-greedy policy with respect to the exact optimal Q-function of the finite MDP, with $\epsilon = 0.05$. It therefore predominantly follows high-value routes toward the positive terminal state 15. This policy is used only to generate state-action samples that model the rapid actor-distribution shift observed after a small number of online Q-loss updates.

For clarity, the implementation uses three distinct datasets:

1. **Offline dataset** $\mathcal{D}_{\text{off}}$: generated by the policy attracted to terminal state 0; used to fit the offline flow and the offline critic.
2. **Post-shift actor dataset** $\mathcal{D}_{\text{actor}}$: generated by the prescribed near-optimal online behavior.
3. **Online critic replay** $\mathcal{D}_{\text{replay}}$: generated by

$$\pi_{\text{replay}} = 0.70\,\pi_{\text{near-optimal}} + 0.30\,\pi_{\text{uniform}}, \tag{35}$$

   where π_{uniform} is uniform over valid outgoing edges.

By decoupling the online actor and the online critic (regardless of whether offline pre-training is employed), we ensure that the actor remains high-performing and reliable during the online phase; this prevents low-quality offline data from impairing the actor's exploration capabilities—and, consequently, the training of the online critic. Similarly, by incorporating a uniform policy into the online critic's training data, we simulate the presence of low-quality data that might otherwise affect the critic's training.

Each dataset contains 35,000 episodes. An episode begins from a uniformly sampled nonterminal state and ends at a terminal state or after 80 transitions. Truncated transitions are treated as terminal in the TD mask.

D.2 Actor

Each discrete destination state is embedded using its normalized two-dimensional canvas coordinate in $[-1, 1]^2$. Given a state s, Gaussian noise x_0, and flow time t, the flow network predicts a velocity field. For a dataset action x_1, Flow Matching minimizes

$$x_t = (1 - t)x_0 + tx_1, \tag{36}$$

$$L_{\text{Flow}} = \mathbb{E}\Big[\|v_\theta(s, x_t, t) - (x_1 - x_0)\|_2^2\Big]. \tag{37}$$

The flow network contains two hidden layers of width 128. At inference time, we use 10 Euler integration steps, clip the continuous action to $[-1, 1]^2$, and project it to the closest valid destination of the current state.

We first train an offline flow $\hat{\pi}_{\text{off}}$ on $\mathcal{D}_{\text{off}}$. Starting from this fitted flow, we then perform 500 additional flow updates on $\mathcal{D}_{\text{actor}}$ and freeze the resulting post-shift policy $\hat{\pi}_{\text{on}}$.

D.3 Critic

The critic concatenates a one-hot state vector with the two-dimensional destination embedding, applies a single hidden layer of width 48 with a $\tanh$ activation, and outputs one scalar Q-value. The critic minimizes

$$L_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}, a'\sim\hat{\pi}(\cdot|s')}\left[\left(Q_\phi(s, a) - \left[r + \gamma Q_{\bar{\phi}}(s', a')\right]\right)^2\right] \tag{38}$$

The target parameters are updated with $\tau = 0.02$.

D.4 IMPLEMENTATION

The complete training pipeline is as follows.

1. Initialize one flow network and fit it to $\mathcal{D}_{\text{off}}$, obtaining $\hat{\pi}_{\text{off}}$.
2. Initialize a small critic once and save its random parameters ϕ_{init}.
3. Train that critic for 500 TD updates on $\mathcal{D}_{\text{off}}$ using $\hat{\pi}_{\text{off}}$ in the TD target, obtaining ϕ_{off}.
4. Adapt the flow policy on $\mathcal{D}_{\text{actor}}$ and freeze the resulting post-shift policy $\hat{\pi}_{\text{on}}$.
5. Draw one common sequence of online-replay minibatches and one common sequence of Gaussian policy-noise keys.
6. Run O2O from ϕ_{off} and CFP from the saved ϕ_{init}. Then perform 500 TD updates on the same replay using the same frozen $\hat{\pi}_{\text{on}}$, minibatches, and policy noises.

D.5 HYPERPARAMETERS

Table 8 summarizes the hyperparameters used in our toy example.

Table 8: Hyperparameters of the toy example.

Hyperparameter	Value	Hyperparameter	Value
Batch size	256	Discount factor γ	0.97
Offline episodes	35,000	Online actor episodes	35,000
Online replay episodes	35,000	Maximum episode length	80
Offline-policy greed	0.985	Online-policy ϵ	0.05
Uniform replay mixture	0.30	Flow steps	10
Actor hidden widths	$[128, 128]$	Actor learning rate	3×10^{-4}
Critic hidden width	48	Critic learning rate	10^{-3}
Offline updates	500	Online updates	500
Target update τ	0.02	RMSE recording interval	10

D.6 RESULTS AND INTERPRETATION

We calculated the target Q-values for the offline and online phases based on offline and online strategies, respectively. We also evaluate the Q-values produced by the O2O and CFP critics. The left side of the figure 12 displays the theoretical target Q-values for the offline and online settings, as well as the Q-values output by the O2O and CFP critics, respectively. The right side shows heatmaps representing the absolute differences between pairs of the plots on the left; areas appearing more gray indicate greater similarity, while darker blue areas indicate larger differences. The O2O critic output is significantly closer to the target value of the offline dataset than the CFP critic output. Conversely, the CFP critic output is closer to the online target Q-value.

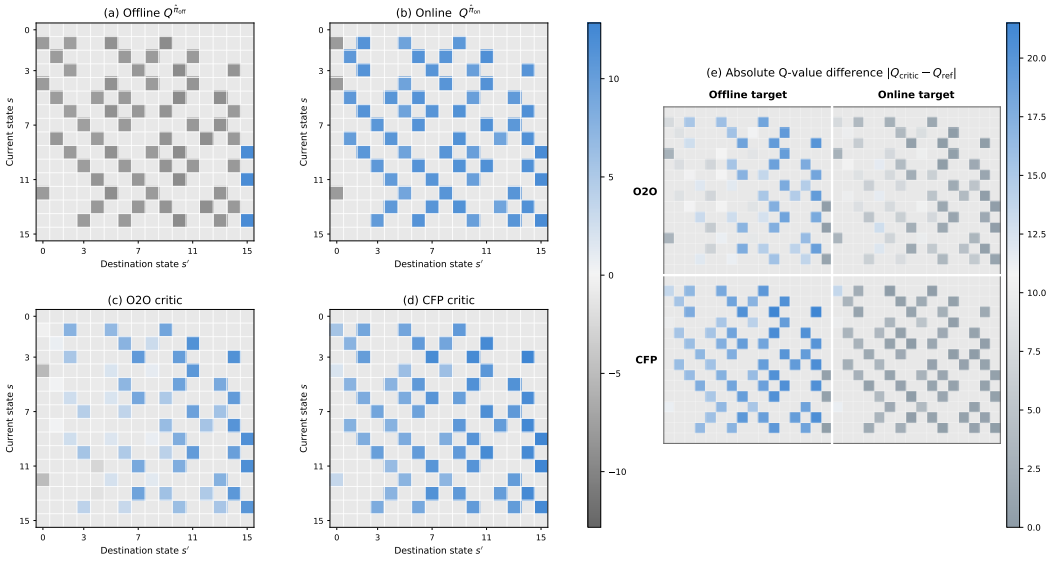


Figure 12: Results of toy example.