FLOW MATCHING LOSS FOR OFFLINE FLOW POLICY REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow models have demonstrated substantial promise for robotics control tasks. Usually, they are trained using behavior cloning, and a key challenge in training these policies is adapting traditional offline RL algorithms to flow policies. In particular, modern offline RL algorithms include a deterministic Q-gradient, which requires backpropagating through the iterative decoding process. This step undermines one of the key advantages of flow policies, which is that every decoding step is trained using a strong supervised signal derived directly from the final output. We propose the following solution: rather than target the action in the offline dataset, we first use Q-guided inference-time policy optimization to identify a better action, and then directly train the flow policy to target that action using the standard flow matching objective. We implement this idea in the context of Reversal Q-learning (RQL), a state-of-the-art offline RL algorithm for flow policies, to obtain Q-gradient RQL (QGRQL). Across multiple tasks from the OGBench locomotion and manipulation environments, we show that QGRQL outperforms existing state-of-the-art methods. Our results demonstrate the promise of better aligning offline RL algorithms with flow matching objectives.

1 INTRODUCTION

Recent progress in offline reinforcement learning (RL) has been driven by methods that adapt advances in generative modeling to reinforcement learning, such as diffusion (Chi et al., 2025) and flow policies (Hu et al., 2024). Such expressive models have been shown to learn complex trajectories from offline datasets (Park et al., 2025b; Oberai et al., 2026; Agrawalla et al., 2026).

The key challenge training flow policies in offline RL is that effective offline RL algorithms are incompatible with the standard flow matching loss. In the supervised learning setting, flow models sample random noise and iteratively refines to produce an output; then, the flow matching loss encourages the refinement on each iteration to target the labeled output. Intuitively, this strategy succeeds because it provides strong supervision across the entire iterative decoding process.

The simplest offline RL algorithm is behavioral cloning (BC), which asks the flow policy to mimic the behavioral data; this strategy is essentially supervised learning and can directly be adapted to training flow policies. However, offline RL aims to learn when the behavioral data is sub-optimal, in which case mimicking the behavioral data produces correspondingly low-quality policies.

Instead, most offline RL algorithms train the policy to optimize the Q-function (learned in parallel from the same behavioral dataset), which quantifies the long-term value of different actions. They typically still include a behavioral cloning loss to regularize the policy towards the behavioral data, but the Q-function term is the one that is hard to optimize.

Several strategies have been proposed for adapting these losses to training flow policies. The most naïve strategy is to simply backpropagate through the iterative decoding process (known as backpropagation through time, or BPTT). However, this strategy performs poorly since information needs to propagate all the way from the Q-function through the decoding process, which can lead to issues such as vanishing or exploding gradients. One alternative is to train a one-step policy instead of performing multi-step decoding (Park et al., 2025b; Espinosa-Dice et al., 2025); however, this strategy results in substantially less expressive policies than multi-step flow policies.

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

Recent work has instead considered incorporating the decoding process into the Markov decision process (MDP) defining the task (Oberai et al., 2026), and learning a Q-function and policy for the extended MDP. This strategy can mitigate the issue since learning a Q-function can enable generalization across state-action pairs; however, Q-learning still takes time to propagate information backwards through the decoding process.

Broadly speaking, all of these existing strategies aim to adapt offline RL algorithms to flow policies rather than trying to adapt flow matching losses to offline RL, which loses the benefits of strong supervision signal enjoyed by flow matching. We propose a new algorithm based on the latter strategy. At a high level, our algorithm directly optimizes the Q-function in action space, and then uses the flow matching loss to target this optimized action.

To realize this strategy, we leverage on-the-fly policy extraction (OPEX) (Park et al., 2024), an inference-time algorithm that performs gradient ascent on the Q-function to directly optimize the action. Our basic idea is to use OPEX to optimize the action at each training step (using the behavioral action as initialization), and then uses the optimized action as the target in the standard flow matching loss to update the flow policy. To ensure the action does not go too far from the behavioral action (which could cause the Q-function to generalize poorly), we modify OPEX to use projected gradient descent, with projection onto an L_2 ball around the behavioral action.

Finally, we integrate this algorithm into the Reversal Q-Learning (RQL) algorithm (Oberai et al., 2026), a state-of-the-art offline RL algorithm that trains flow policies. Our modified algorithm, which we call Q-Gradient Reversal Q-Learning (QGRQL), outperforms RQL on several locomotion and manipulation tasks in the OGBench dataset (Park et al., 2025a), with especially strong performance on several long-horizon tasks.

In summary, our contributions are:

- We show that even in early training iterations, Q-functions can be used to identify more effective actions compared to the flow policy, suggesting that the existing policy learning procedures lag Q-learning.
- Motivated by this experiment, we propose a novel algorithm for training flow policies; on each step, our algorithm uses OPEX to compute actions with high Q-values, and then uses the selected actions as targets in the flow-matching loss to update the policy.
- We integrate our algorithm with RQL to obtain Q-Gradient RQL (QGRQL).
- We perform experiments on OGBench (Park et al., 2025a), showing that training on OPEX-optimized dataset action targets outperforms state-of-the art offline RL algorithms for training flow policies.

2 Related Work

Backpropagation Through Time (BPTT) Several previous flow policy learning algorithms focus on value function maximization during training while regularizing it with a flow matching term (often referred to as a behavior-cloning term) such as Diffusion-QL(Wang et al., 2023), Consistency-AC (Ding & Jin, 2024), BAM (Li & Levine, 2026), FBRAC (Park et al., 2025b), SRDP (Ada et al., 2024), and EQL (Zhang et al., 2024). However, as previous research (Park et al., 2025b; Li & Levine, 2026; Oberai et al., 2026) have demonstrated, backpropagating through time is often unstable and underperforms approaches that sidestep BPTT entirely.

Regression-Based Approaches To avoid backpropagating through time, several methods choose to avoid using gradient information from value functions to train the policy. Instead, they take inspiration from approaches prevalent in offline Gaussian policy learning that attempt to identify and upweight the highest-quality actions present in the offline dataset (Hong et al., 2023a;b). Consequently, several flow and diffusion policy algorithms filter or reweight trajectories based on advantage weights prior to flow matching (Physical Intelligence et al., 2025; Zhang et al., 2025). Alternatively, they may also simply rejection sample from a behavioral policy and select the highest-value actions, as in IDQL (Hansen-Estruch et al., 2023), IFQL (Park et al., 2025b), and SfBC (Chen et al., 2023). However, weighted approaches underutilize value functions, upper-bounding performance benefits to the actions present in the dataset and underperform on datasets with few effective sam-

ples, while rejection sampling becomes prohibitively expensive at scale (Park et al., 2025b; 2024; Fu et al., 2022).

Distillation and Mean-Flow Based Approaches Approaches like FQL (Park et al., 2025b) and SORL (Espinosa-Dice et al., 2025) train a full flow policy using flow matching and then distil into a one-step policy, while mean-flow based approaches like MeanFlowQL (Wang et al., 2026) directly train a one-step mean-flow based policy using a similar objective by utilizing average velocities. These methods perform value function maximization during the distillation into a one-step, where due to the one-step nature of the policy. However, due to their reliance on a one-step final policy to avoid backpropagating through time, such approaches sacrifice policy expressivity, thus leading to limited performance benefits over existing approaches.

Improving Value Function Expressivity Alternative approaches for circumventing the BPTT problem have focused on improving the expressivity of value functions to provide better gradients during training. For example, floq (Agrawalla et al., 2026) optimizes FQL's performance by training a flow model to serve as the Q function. On the other hand, approaches like RQL (Oberai et al., 2026) and Q-Flow (Doo et al., 2026) train their value functions to learn to reward intermediate values produced by flow integration steps using policy-generated data. However, despite retaining expressive full-flow policies, these approaches yield limited benefits on longer-horizon tasks, where the policy-generated data is insufficient to train a Q-function that accurately rewards intermediate steps.

3 Preliminaries

3.1 Problem Setting

In the offline setting, we assume a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \mu, p)$ (Sutton & Barto, 2018) and set up an expanded MDP $\widetilde{\mathcal{M}} = (\widetilde{\mathcal{S}}, \mathcal{A}, \widetilde{r}, \widetilde{\mu}, \widetilde{p})$ for our flow policies, similar to RQL (Oberai et al., 2026). For the MDP $\mathcal{M}$, $\mathcal{S}$ is a state space, $\mathcal{A} = \mathbb{R}^d$ is an action space, $r(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $\mu(s) \in \Delta(\mathcal{S})$ is an initial state distribution, and $p(s' \mid s,a) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a transition dynamics kernel with the set of probability distributions over a space $\mathcal{X}$ denoted as $\Delta(\mathcal{X})$.

For the expanded MDP $\widetilde{\mathcal{M}}$, state space $\widetilde{\mathcal{S}} = \mathcal{S} \times \mathbb{R}^d \times \{0, 1, \ldots, F-1\}$ consists of elements corresponding to the tuples (s, x, f) of a state $s \in \mathcal{S}$, a partially generated action $x \in \mathbb{R}^d$, and discretized flow time $f \in \{0, 1, \ldots, F-1\}$. The transition dynamics kernel $\widetilde{p}((s', x', f') \mid (s, x, f), a) : \widetilde{\mathcal{S}} \times \mathbb{R}^d \to \Delta(\widetilde{\mathcal{S}})$ sets (s', x', f') deterministically to $(s, x + a, f + 1)$ if $f < F - 1$. Otherwise, we sample s' from $p(\cdot \mid s, x + a)$, x' from $\mathcal{N}(0, I_d)$, and set f' deterministically to 0. In the initial state distribution $\widetilde{\mu}(s, x, f)$, we sample s from $\mu(\cdot)$, x from $\mathcal{N}(0, I_d)$, and set f to 0. The reward function $\widetilde{r}((s, x, f), a) : \widetilde{\mathcal{S}} \times \mathbb{R}^d \to \Delta(\widetilde{\mathcal{S}})$ has a similar conditional structure: it is set to 0 if $f < F - 1$ and $r(s, x + a)$ otherwise.

Then, given an offline learning dataset $\mathcal{D} = \{\tau^{(n)}\}_{n \in \{1,2,\ldots,N\}}$ comprising trajectories in the original MDP $\tau = (s_0, a_0, r_0, s_1, \ldots, s_T)$, the goal of the offline reinforcement learning problem is defined as finding the optimal policy $\pi(a \mid s) : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the average discounted return $\mathbb{E}_{\tau \sim p^\pi(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, where we denote the discount factor as $\gamma \in (0, 1)$ and

$$p^\pi(\tau) = \mu(s_0)\, \pi(a_0 \mid s_0)\, p(s_1 \mid s_0, a_0) \ldots \pi(a_{T-1} \mid s_{T-1})\, p(s_T \mid s_{T-1}, a_{T-1}) \tag{1}$$

We note that in RQL, the expanded MDP setup only discounts rewards from $\mathcal{M}$ i.e. the expanded MDP uses discount factor $\widetilde{\gamma}$ defined as $\widetilde{\gamma} = 1$ if $f < F - 1$ and $\widetilde{\gamma} = \gamma$ otherwise.

3.2 Flow Policies

Flow matching (Albergo & Vanden-Eijnden, 2023; Liu et al., 2023; Lipman et al., 2023) is an alternative to denoising diffusion for training generative models that relies on deterministic ordinary differential equations (ODEs) instead of stochastic differential equations (SDEs). Following RQL's convention (Oberai et al., 2026), we denote the time-dependent velocity field as $v(s, x, f) : \mathcal{S} \times \mathbb{R}^d \times [0, F] \to \mathbb{R}^d$, where f denotes the flow model timestep and F denotes the largest timestep (in practice, the number of Euler method steps used to solve the flow ODE). The

velocity field can then be used to define a flow $\psi(s, x, f) : \mathcal{S} \times \mathbb{R}^d \times [0, F] \to \mathbb{R}^d$, defined as the unique solution to the following ODE:

$$\frac{\mathrm{d}}{\mathrm{d}f}\psi(s, x, f) = v(s, \psi(s, x, f), f). \tag{2}$$

The simplest flow-matching objective is the behavior cloning objective (Park et al., 2025b), which models the action distributions in dataset $\mathcal{D}$ by minimizing the following flow matching loss:

$$\mathcal{L}^{\mathrm{BC}}(v) = \mathbb{E}_{\substack{(s,a)\sim\mathcal{D},\\ x^0\sim\mathcal{N}(0,I_d),\\ f\sim\mathcal{U}(0,F)}}\left[\left\|v(s, x^f, f) - \frac{1}{F}(a - x^0)\right\|_2^2\right], \tag{3}$$

where $\mathcal{U}$ is a uniform distribution and

$$x^f = (1 - f/F)x^0 + (f/F)a. \tag{4}$$

3.3 Reversal Q-Learning (RQL)

Reversal Q-Learning (RQL) (Oberai et al., 2026) bypasses the need for backpropgating through time by converting tuples (s, a, r, s') in the original MDP $\mathcal{M}$ from the offline dataset $\mathcal{D}$ into a dataset $\widetilde{\mathcal{D}}$ comprising tuples $((s, x^f, f), a, \widetilde{r}((s, x^f, f), a), (s', x^{f'}, f'))$ in the expanded MDP $\widetilde{\mathcal{M}}$. Then, it trains a value function $V(s, x, f) : \mathcal{S} \times \mathbb{R}^d \times 0, \ldots, F - 1$, which is parametrized by the flow step f and can thus estimate values for intermediate generations. The algorithm then uses V to update the policy loss.

Dataset Conversion: Given a tuple $(s, a, r, s') \sim \mathcal{D}$, RQL computes the intermediate tuples (s, x^f, f) using the reverse flow $\theta(s, x, f) : \mathcal{S} \times \mathbb{R}^d \times [0, F] \to \mathbb{R}^d$ by solving the following "reversed" ODE:

$$\frac{\mathrm{d}}{\mathrm{d}f}\theta(s, x, f) = -v(s, \theta(s, x, f), f). \tag{5}$$

This ODE can be solved using the Euler method to compute

$$x^{f-1} \leftarrow x^f - v(s, x^f, f) \tag{6}$$

for $f = F - 1, \ldots, 1$ and $x^F = a$. Once the tuples (s, x^f, f) have been computed, the associated rewards $\widetilde{r}((s, x^f, f), a)$ are calculated using the definition of $\widetilde{r}((s, x, f), a)$ in Section 3.1.

Value Learning and Policy Extraction: To avoid querying the learned value function on out-of-distribution queries, RQL uses the following expectile regression loss (similar to a value-only variant of IQL (Kostrikov et al., 2022)) to train the value function $V(s, x, f)$.

$$\mathcal{L}(V) = \mathbb{E}_{\widetilde{\tau}}\left[\ell_2^\kappa\big(V(s, x^f, f) - (r + \gamma V(s', x'^0, 0))\big)\right] \tag{7}$$

where flow trajectories

$$\widetilde{\tau} = ((s, x^0, 0), a^0, 0, (s, x^1, 1), a^1, 0, \ldots, (s, x^{F-1}, F - 1), a^{F-1}, r, (s', x'^0, 0))$$

are sampled from $\widetilde{\mathcal{D}}$ with $a^f = x^{f+1} - x^f$, f is sampled uniformly from $\{0, 1, \ldots, F - 1\}$, and $\ell_2^\kappa(x) = |\kappa - \mathbb{I}(x > 0)|x^2$ is the expectile loss with an expectile κ. For policy extraction, RQL uses a DDPG-style loss which uses the flow-matching loss (3) as a behavioral regularizer with dataset action a.

$$\mathcal{L}(v) = -\mathbb{E}[V(s, x^f + v(s, x^f, f), f + 1)] + \alpha\mathcal{L}^{\mathrm{BC}}(v) \tag{8}$$

4 Methodology

For our algorithm, our goal is to develop a method for better identifying high-value actions during training. We first describe the observed gap between value function and policy *quality* in Section 4.1, and then describe on-the-fly policy extraction in Section 4.2, and present QGRQL, our practical algorithm in Section 4.3.

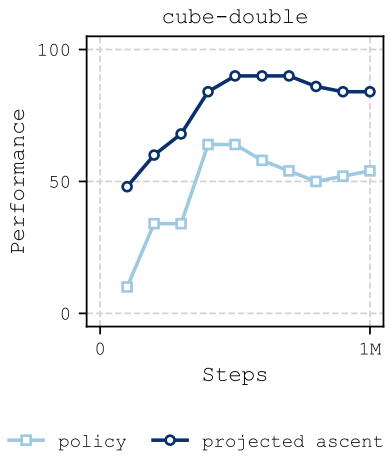

Figure 1: Performance of value-function-optimized RQL policy (*projected ascent*) compared to standard RQL (*policy*).

4.1 Does Reversal Q-Learning Learn Good Value Functions Quickly?

Prior research has shown that in offline learning with Gaussian policies, an algorithm's final performance is extremely sensitive to the policy extraction method used, with drastic performance differences observed across methods when using an identical value function (Park et al., 2024). With flow and diffusion policies, research into prior implicit Q-learning-based variants has shown that on many environments, implicit Q-learning approaches relying on rejection-sampling based policy extraction (which selects the highest-value actions from a behavior policy) either outperform a policy trained with a regularized value maximization loss, as in the case of IDQL and DQL (Hansen-Estruch et al., 2023), or perform better during the initial training iterations, as in the comparison between IFQL and FQL (Park et al., 2025b, Figure 12).

This gap suggests that value learning, especially in initial iterations, Q-function quality improves faster than and leads policy quality during early iterations. Thus, Q-functions may initially underutilized due to sampling poor-quality actions from the offline dataset. Thus, even if policies trained with value maximization are able to achieve better performance by the end of training, they train on lower-quality actions for a significant number of their initial training iterations *despite* possessing a value function that is already capable of identifying much better-quality actions. We will refer to this lag as the *value-learning-policy-extraction lag*.

While having a high-quality value function has been exploited in previous work in offline-to-online settings through action optimization of policy-generated actions (Mark et al., 2025; Dong et al., 2026), their motivation stems from adapting inference-time policy-generated action optimization techniques to online learning. Here, having a strong value function in the initial iterations of online learning is leveraged to overcome poor action choices contributed to the replay buffer by the policy. This approach, however, has not yet been explored in offline learning, primarily because it is difficult to measure if the value function has learned to outperform the policies used to create the offline dataset.

To demonstrate the existence of this gap in Reversal Q-Learning (RQL), in Figure 1, we examine the impact of optimizing policy-generated actions (similar to PA-RL (Mark et al., 2025)) with projected gradient ascent on the Q-function gradient for a few representative tasks. Our initial experiments demonstrate the existence of this value-learning-policy-extraction lag in early iterations. Consequently, we believe that the value function is capable of optimizing actions well before the policy achieves its peak performance. This motivates us to try optimizing the dataset action targets in a similar manner by utilizing the value function gradients.

4.2 Regularized Action Target Optimization

We now introduce our approach for optimizing the dataset action targets using the Q-function gradient. We consider the action optimization technique using the On-the-fly Policy Extraction (OPEX) optimization technique from Park et al. (2024):

$$a^* \triangleq a + \beta \cdot \nabla_a Q(s, a) \tag{9}$$

In OPEX, the action a is sampled from the trained policy at inference time. In our approach, we utilize our value function (trained as in RQL) for optimization by parametrizing the above equation using the flow timestep. Starting from the dataset action $a_{\text{data}} = a_0$, at each iteration t, we identify an optimized action by performing projected gradient ascent onto an L_2 ball of radius R for k iterations (as shown in Figure 2), where k is a hyperparameter of our method. This regularization ensures that the optimized actions do not deviate too far from the original dataset action, leading to a potential drift in reward. We compute a_t as shown below:

$$a_t = a_{\text{data}} + \Delta \cdot \min\left\{1, \frac{R}{\|\Delta\|}\right\} \tag{10}$$

where $\Delta = \left(a_{t-1} + \beta \cdot \nabla_x V(s, x, f)\,\big|_{x=a_{t-1}}\right) - a_{\text{data}}$, and $1 \leq t \leq k$. Our final optimized action is $a_{\text{OPEX}} = a_k$. A natural strategy would be to use the flow matching loss directly with a_{OPEX}; our algorithm takes this general approach but integrates more tightly with RQL.

4.3 Q-Gradient Reversal Q-Learning (QGRQL)

While training on optimized action targets, we need to ensure that value function learning itself remains unaffected during training, for two primary reasons. Firstly, training the value function on optimized actions is likely to degrade performance as the objective would utilize expectile losses calculated over actions the value function has already identified as high value actions. Secondly, although we regularize the actions during optimization, without access to the original reward functions in the offline setting, we may end up with actions which are misaligned with the rewards assigned to those trajectories in the offline dataset.

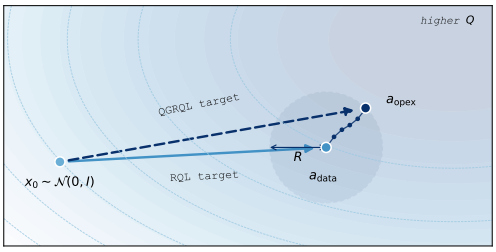

Figure 2: OPEX-action optimization in QGRQL.

Consequently, when implementing action optimization in practice, we design a separate set of flow trajectories, $\widetilde{\tau}'$ with respect to the optimized action a_{OPEX}, defined as follows:

$$\widetilde{\tau}' = ((s, x^0, 0), a^0_{\text{OPEX}}, 0, (s, x^1_{\text{OPEX}}, 1), a^1_{\text{OPEX}}, 0, \ldots, (s, x^{F-1}_{\text{OPEX}}, F-1), a^{F-1}_{\text{OPEX}}, r, (s', x'^0, 0)) \tag{11}$$

where $x^f_{\text{OPEX}} = (1 - f/F)x^0 + (f/F)a_{\text{OPEX}}$ for $0 < f \leq F - 1$ (as in (4)), and we define $a^f_{\text{OPEX}} = x^{f+1}_{\text{OPEX}} - x^f_{\text{OPEX}}$. Thus, we define the intermediate steps with respect to the optimized action a_{OPEX} unlike RQL, which defines it with respect to the dataset action a. The revised policy loss function is then defined as:

$$\begin{aligned}\mathcal{L}(v) = &- \mathbb{E}_{\widetilde{\tau}'}[V(s, x^f_{\text{OPEX}} + v(s, x^f_{\text{OPEX}}, f), f+1)] \\ &+ \alpha \cdot \mathbb{E}_{\substack{(s,a)\sim\mathcal{D},\\ x^0\sim\mathcal{N}(0,I_d),\\ f\sim\mathcal{U}(0,F)}}\left[\left\| v(s, x^f_{\text{OPEX}}, f) - \frac{1}{F}(a_{\text{OPEX}} - x^0)\right\|^2_2\right]\end{aligned} \tag{12}$$

Thus, the policy loss is calculated with a different set of intermediate values x^f_{OPEX} and flow-matched to the optimized action a_{OPEX}, while the value function loss $\mathcal{L}(V)$ is calculated as in RQL (with respect to the reverse-flow generated actions). We present our complete algorithm in Algorithm 1.

Algorithm 1 Q-Gradient Reversal Q-Learning (QGRQL)

Initialize value function $V(s, x^f, f)$, flow policy $v(s, x^f, f)$
while true **do**
  Sample $(s, a, r, s') \sim \mathcal{D}$
  for $t = 1$ to k **do**
    Compute a_t using (10)
  $a_{\text{OPEX}} = a_k$
  Compute $\widetilde{\tau}$ using (6) on (s, a, r, s')
  Train V by minimizing $\mathcal{L}(V)$ with respect to $\widetilde{\tau}$ (7)
  Compute $\widetilde{\tau}'$ using (11)
  Train v by minimizing $\mathcal{L}(v)$ as in (12)

5 EXPERIMENTS

In this section, we describe our empirical experiments on QG-RQL's performance on a diverse set of locomotion and manipulation tasks. We describe our experimental setup in Section 5.1. Our experimental results are presented and discussed in Section 5.2, and we ablate our projection radius R as well as the iteration in which we start optimizing actions in Section 5.3.

5.1 EXPERIMENTAL SETUP

Tasks. We evaluate on 50 challenging tasks from the OGBench dataset (Park et al., 2025a). We evaluate on tasks reported in (Oberai et al., 2026), which include both locomotion environments like `antmaze` and `humanoidmaze`, as well as robot manipulation environments like `puzzle`, `scene`, and `cube`. We consider multiple levels of difficulty for all environments except `scene` for an exact comparison to the set of environments evaluated in RQL (Oberai et al., 2026) and other prior work (Li & Levine, 2026). Keeping with RQL's convention, we examine performance on the 100M-size task datasets for `cube-quadruple` and `puzzle-4x4`, and action-chunk our manipulation environments with time horizon $h = 5$ (and keep $h = 1$ for locomotion environments).

Hyperparameters. The L_2 ball radius for our algorithm is set to $R = 0.2$ for non-100M environments and $R = 0.05$ for `cube-quadruple-100M` and `puzzle-4x4-100M`, which we discuss further in Section 5.3. Finally, we set the number of gradient-ascent steps to $k = 10$ across all our environments. We report peak performance on all tasks, averaged across three random seeds.

Baselines. We compare to two state-of-the art offline flow policy learning algorithms, Reversal Q-Learning (RQL) (Oberai et al., 2026) and floq (Agrawalla et al., 2026), which utilize two different strategies for avoiding backpropagating through time (BPTT). RQL performs MDP expansion and utilizes the reverse flow to generate intermediate actions. floq improves the expressivity of the Q-function by using a flow model architecture. It then uses the resulting flow Q-function in FQL (Park et al., 2025b), which distils the trained policy into a one-step policy, thereby avoiding BPTT.

5.2 RESULTS

We present our full results on all tasks in Table 2, an aggregated summary in Table 1, and training curves from a few representative tasks in Figure 3. Our results show that QGRQL achieves the best average performance across all evaluated approaches. While our average improvement over RQL is modest, we have a sizeable improvement in performance over RQL on environments with long-horizon tasks and larger offline datasets, like `cube-double`, `scene-sparse`, `puzzle-4x4-100M`, and `cube-quadruple-100M`,. We also outperform floq across most environments, although it yields a stronger performance on `cube-double`. Our results demonstrate that Q-gradient-based action target optimization yields comparable or better performance on most tasks, with strong peformance gains on longer horizon and robot manipulation tasks. This suggests that poor policy performance on such tasks in prior work may be caused due to poor policy extraction, as the performance gains suggest that RQL algorithm does learn a strong value function.

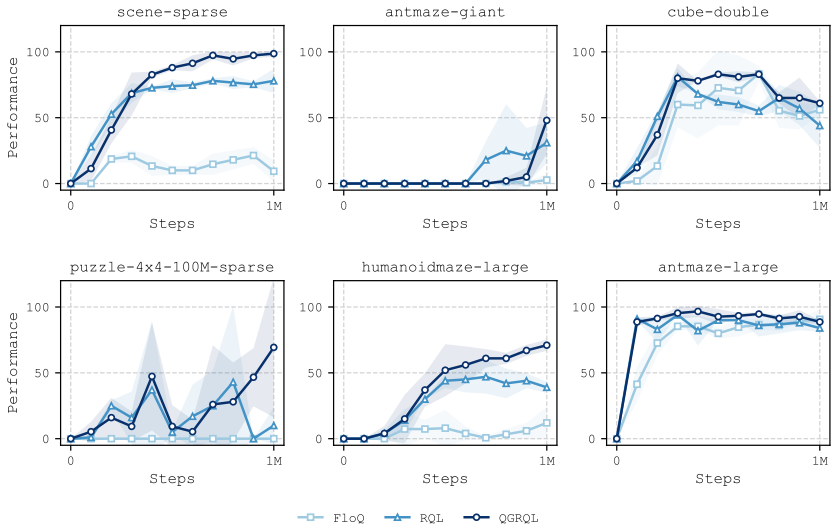

Figure 3: Performance of floq, RQL, and QGRQL over 1M training steps on one representative task per environment: `scene-sparse` (task 5), `antmaze-giant` (task 3), `cube-double` (task 1), `puzzle-4x4-100M` (task 5), `humanoidmaze-large` (task 5), and `antmaze-large` (task 1). Curves are averaged over three seeds.

Environment	Task	floq	RQL	QGRQL (Ours)
cube	cube-double	**63.2 ± 2.8**	37.1 ± 4.9	50.5 ± 2.7
	cube-triple*	5.3 ± 2.3	**7.7 ± 1.8**	6.0 ± 1.7
	cube-quadruple-100M*	0.0 ± 0.0	58.8 ± 0.4	**62.7 ± 2.3**
puzzle	puzzle-3x3-sparse*	4.0 ± 0.4	**100.0 ± 0.0**	**100.0 ± 0.0**
	puzzle-4x4-100M-sparse*	1.5 ± 0.2	66.4 ± 7.2	**74.5 ± 7.7**
scene	scene-sparse*	70.4 ± 3.9	94.3 ± 1.8	**99.6 ± 0.4**
antmaze	antmaze-large	89.2 ± 2.9	93.1 ± 1.3	**94.0 ± 1.8**
	antmaze-giant	**43.9 ± 6.2**	39.6 ± 0.4	43.2 ± 2.5
humanoidmaze	humanoidmaze-medium	82.1 ± 4.8	**97.6 ± 2.5**	95.7 ± 1.3
	humanoidmaze-large	18.9 ± 6.9	46.4 ± 1.1	**46.7 ± 5.5**
Mean		37.9	64.1	**67.3**

Table 1: **OGBench Offline RL results.** Success rate (%) aggregated over each environment's five tasks, reported as mean ± standard deviation over three seeds; floq is trained for 1M gradient steps to match RQL's update budget, which is shorter than the schedule used in its own evaluation. Environments marked * are not reported by floq; for `cube-triple` and `cube-quadruple-100M` we adopt its cube-double configuration, and for the sparse-reward variants of `scene`, `puzzle-3x3`, and `puzzle-4x4-100M` we retrain with the hyperparameters it reports for the corresponding dense tasks. For per-task results see Table 2.

5.3 Ablation Studies

In this section, we describe two ablations of QGRQL and their impacts upon our results. We perform these ablations on the default task of `cube-double`, `antmaze-large`, `cube-quadruple-100M` and with a single seed. All other hyper parameters are held at the values used in Section 5.3.

L_2-projection ball radius (R): In this ablation study, we examine the variance in QGRQL's performance under different radii for the L_2-projection ball during projected gradient ascent while optimizing the dataset actions. Lower values of R restrict the optimized action to be closer to the

original dataset action. We evaluate the selected tasks for four values of R: 0.05, 0.1, 0.2, and 0.4 and present results in Figure 4. Our experiments suggest that lower values of R lead to better performance on the datasets with more samples (the 100M environments), while relatively larger values perform better on datasets with fewer samples.

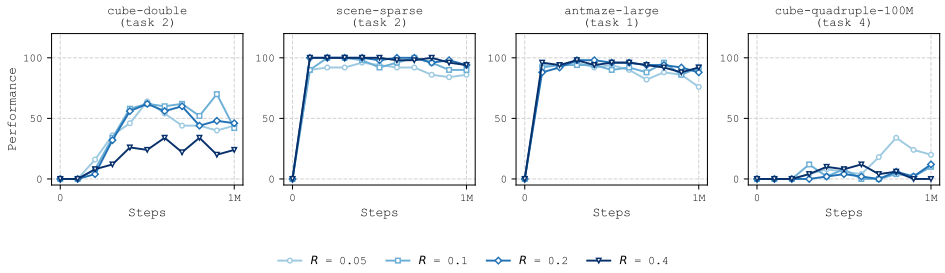

Figure 4: Impact of R on policy performance for representative tasks from four environments.

Delayed Action Target Optimization: In QGRQL, we initialize the action target optimization from the first iteration of training. However, it is possible that the algorithm may not have learned an accurate value function at the start of optimization. In this ablation, we study the impact of delaying starting value-gradient based action-target optimization until `start` training iterations have passed i.e. we train using dataset actions for the first `start` training iterations. We present our results for `start`$= 0$ (QGRQL default), $100000, 250000, 500000$ in Figure 5. Our results show that performance generally degrades when the action target optimization is introduced later. This suggests that the policy struggles to recover after initially training on poorer-quality trajectories from the dataset, even when exposed to optimized action targets from a more mature value function.

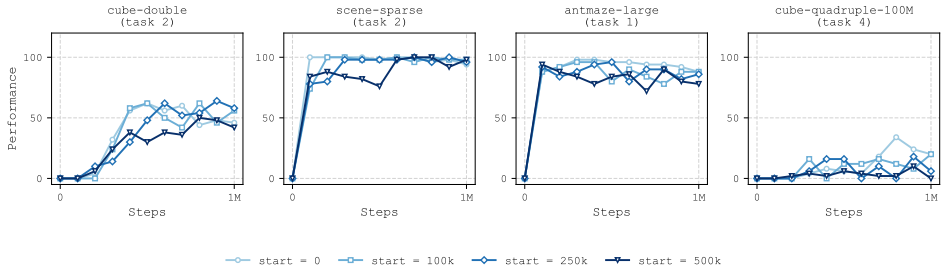

Figure 5: Impact of delaying starting optimization on policy performance for representative tasks from four environments.

6 Conclusion

In this paper, we present QGRQL, an offline algorithm for training flow policies that uses the learned value function to directly optimize dataset action targets while avoiding backpropagation through time. We show that existing policy learning procedures lag value learning, thereby underutilizing the value fuctions. Using our action target optimization procedure, we leverage value function gradients and outperform existing state-of-the art offline algorithms for flow policy learning, with strong performance gains on challenging robot manipulation and longer-horizon tasks.

Limitations Despite outperforming state-of-the-art offline algorithms for training flow policies, QGRQL's performance is highly dependent on the value function's ability to quickly learn to reward high-value actions using the offline dataset. At the same time, QGRQL exhibits sensitivity both to L_2 projection ball radius R as well as the iteration number in which action target optimization is deployed. Finally, although we evaluate QGRQL on a wide variety of robotics tasks in the OGBench dataset, our evaluations are restricted to robotics control tasks in a simulated environment, and leave the impact of action target optimization on more challenging tasks and real-world environments as an open question for future study.

AI USE STATEMENT

LLMs were used to generate and explain the code for executing some of the experiments, which was reviewed and checked by the authors.

REFERENCES

Suzan Ece Ada, Erhan Oztop, and Emre Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters*, 9(4):3116–3123, 2024. doi: 10.1109/LRA.2024.3363530.

Bhavya Agrawalla, Michal Nauman, Khush Agrawal, and Aviral Kumar. floq: Training critics via flow-matching for scaling compute in value-based rl. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 54569–54595, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/59d15ab03bd84330d44313eb203c3bb6-Paper-Conference.pdf.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=li7qeBbCR1t.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=42zs3qa2kpy.

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. doi: 10.1177/02783649241273668. URL https://doi.org/10.1177/02783649241273668.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 53047–53066, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e9e1a0abc1a5b19a4aeb80dab19c82ae-Paper-Conference.pdf.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 131379–131394, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/d56a652ad743308b24b6c4d0e6411dd7-Paper-Conference.pdf.

JaeHyeok Doo, Byeongguk Jeon, Seonghyeon Ye, Kimin Lee, and Minjoon Seo. Q-flow: Stable and expressive reinforcement learning with flow-based policy. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=oZqOS1N6Ag.

Nicolas Espinosa-Dice, Yiyi Zhang, Yiding Chen, Bradley Guo, Owen Oertell, Gokul Swamy, Kianté Brantley, and Wen Sun. Scaling offline rl via efficient and expressive shortcut models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 2376–2414. Curran Associates, Inc., 2025. doi: 10.52202/085713-0085. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/03715f9a330c916da8144d1f8dad4554-Paper-Conference.pdf.

Yuwei Fu, Di Wu, and Benoit Boulet. A closer look at offline rl agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 8591–8604. Curran Associates, Inc., 2022. doi: 10.52202/068431-0625. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/3908cadfcc99db12001eafb1207353e9-Paper-Conference.pdf.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies, 2023. URL https://arxiv.org/abs/2304.10573.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Zhang-Wei Hong, Pulkit Agrawal, Remi Tachet des Combes, and Romain Laroche. Harnessing mixed offline reinforcement learning datasets via trajectory weighting. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023a. URL https://openreview.net/forum?id=OhUAblg27z.

Zhang-Wei Hong, Aviral Kumar, Sathwik Karnik, Abhishek Bhandwaldar, Akash Srivastava, Joni Pajarinen, Romain Laroche, Abhishek Gupta, and Pulkit Agrawal. Beyond uniform sampling: Offline reinforcement learning with imbalanced datasets. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 4985–5009. Curran Associates, Inc., 2023b. doi: 10.52202/075280-0221. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/0ff3502bb29570b219967278db150a50-Paper-Conference.pdf.

Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 138836–138858. Curran Associates, Inc., 2024. doi: 10.52202/079017-4405. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/fa9cfdb49f7de3dee54007f84c0745b1-Paper-Conference.pdf.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

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.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy-agnostic RL: Offline RL and online RL fine-tuning of any class and backbone. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025. URL https://openreview.net/forum?id=VGOFxdc6AD.

Aditya Oberai, Seohong Park, and Sergey Levine. Reversal q-learning. *arXiv preprint arXiv:2606.17551*, 2026.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 79029–79056. Curran Associates, Inc., 2024. doi: 10.52202/079017-2509. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/8ffb4e3118280a66b192b6f06e0e2596-Paper-Conference.pdf.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/ecd92623ac899357312aaa8915853699-Paper-Conference.pdf.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48104–48127. PMLR, 13–19 Jul 2025b. URL https://proceedings.mlr.press/v267/park25f.html.

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

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Zeyuan Wang, Da Li, Yulin Chen, Ye Shi, Liang Bai, Tianyuan Yu, and Yanwei Fu. One-step generative policies with q-learning: a reformulation of meanflow. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i31.39885. URL https://doi.org/10.1609/aaai.v40i31.39885.

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.

Ruoqi Zhang, Ziwei Luo, Jens Sjölund, Thomas B. Schön, and Per Mattsson. Entropy-regularized diffusion policy with q-ensembles for offline reinforcement learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 98871–98897. Curran Associates, Inc., 2024. doi: 10.52202/079017-3138. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b319c0e8092ba726cb22e718d7d9a95e-Paper-Conference.pdf.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 17943–17970, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/2d9c6cdb4cfe93869c090fea7375044b-Paper-Conference.pdf.

A FULL RESULT TABLE

		floq	RQL	QGRQL
antmaze-large	task1	[92,98] **94**	[94,94] **94**	[98,100] **99**
	task2	[68,86] **77**	[90,96] **93**	[90,98] **93**
	task3	[94,96] **95**	[96,98] **97**	[98,100] **99**
	task4	[76,96] **87**	[92,94] **93**	[84,88] **87**
	task5	[90,98] **93**	[82,92] **88**	[88,94] **92**
	agg. (5 tasks)	[87,92] **89**	[92,94] **93**	[92,96] **94**
antmaze-giant	task1	[66,76] **70**	[12,38] **25**	[0,42] **14**
	task2	[86,90] **88**	[6,14] **10**	[2,50] **32**
	task3	[0,8] **3**	[20,50] **39**	[30,66] **46**
	task4	[6,34] **21**	[46,60] **54**	[54,62] **57**
	task5	[0,82] **37**	[60,74] **69**	[64,70] **67**
	agg. (5 tasks)	[38,50] **44**	[39,40] **40**	[41,46] **43**
humanoidmaze-medium	task1	[72,98] **88**	[96,100] **99**	[98,100] **99**
	task2	[100,100] **100**	[100,100] **100**	[100,100] **100**
	task3	[98,100] **99**	[100,100] **100**	[100,100] **100**
	task4	[0,40] **24**	[74,98] **89**	[74,86] **79**
	task5	[98,100] **99**	[100,100] **100**	[100,100] **100**
	agg. (5 tasks)	[79,88] **82**	[95,100] **98**	[95,97] **96**
humanoidmaze-large	task1	[14,46] **31**	[72,86] **78**	[78,80] **79**
	task2	[0,0] **0**	[4,6] **5**	[4,16] **10**
	task3	[24,36] **31**	[32,48] **41**	[32,48] **37**
	task4	[8,18] **13**	[48,64] **53**	[12,54] **33**
	task5	[10,26] **20**	[52,56] **55**	[68,78] **73**
	agg. (5 tasks)	[11,24] **19**	[46,48] **46**	[40,51] **47**
scene-sparse	task1	[68,100] **79**	[100,100] **100**	[100,100] **100**
	task2	[80,84] **82**	[88,94] **90**	[100,100] **100**
	task3	[84,86] **85**	[100,100] **100**	[100,100] **100**
	task4	[80,82] **81**	[100,100] **100**	[98,100] **99**
	task5	[24,26] **25**	[78,88] **81**	[96,100] **99**
	agg. (5 tasks)	[67,75] **70**	[93,96] **94**	[99,100] **100**
puzzle-3x3-sparse	task1	[8,10] **9**	[100,100] **100**	[100,100] **100**
	task2	[2,4] **3**	[100,100] **100**	[100,100] **100**
	task3	[2,4] **3**	[100,100] **100**	[100,100] **100**
	task4	[2,2] **2**	[100,100] **100**	[100,100] **100**
	task5	[2,4] **3**	[100,100] **100**	[100,100] **100**
	agg. (5 tasks)	[4,4] **4**	[100,100] **100**	[100,100] **100**
puzzle-4x4-100M-sparse	task1	[0,2] **1**	[54,100] **84**	[98,100] **99**
	task2	[0,2] **1**	[36,76] **58**	[6,20] **12**
	task3	[2,4] **3**	[86,100] **95**	[90,100] **96**
	task4	[2,2] **2**	[6,62] **38**	[50,98] **74**
	task5	[0,0] **0**	[28,84] **57**	[74,100] **91**
	agg. (5 tasks)	[1,2] **1**	[59,73] **66**	[66,82] **75**
cube-double	task1	[82,100] **89**	[66,86] **76**	[84,88] **87**
	task2	[60,80] **71**	[34,62] **51**	[56,74] **65**
	task3	[50,74] **65**	[22,28] **24**	[48,64] **56**
	task4	[14,30] **21**	[16,18] **17**	[30,32] **31**
	task5	[64,80] **70**	[14,20] **17**	[10,20] **14**
	agg. (5 tasks)	[60,66] **63**	[32,41] **37**	[49,54] **51**
cube-triple	task1	[16,34] **24**	[10,24] **18**	[8,24] **18**
	task2	[0,0] **0**	[2,6] **3**	[2,4] **3**
	task3	[0,6] **3**	[4,6] **5**	[2,4] **3**
	task4	[0,0] **0**	[0,4] **2**	[0,2] **1**
	task5	[0,0] **0**	[8,14] **11**	[4,6] **5**
	agg. (5 tasks)	[4,8] **5**	[6,10] **8**	[4,7] **6**
cube-quadruple-100M	task1	[0,0] **0**	[82,96] **91**	[90,100] **95**
	task2	[0,0] **0**	[88,100] **95**	[92,100] **97**
	task3	[0,0] **0**	[76,84] **79**	[80,94] **87**
	task4	[0,0] **0**	[16,36] **29**	[34,38] **35**
	task5	[0,0] **0**	[0,0] **0**	[0,0] **0**
	agg. (5 tasks)	[0,0] **0**	[58,59] **59**	[61,65] **63**
all		[39,36] **38**	[63,65] **64**	[67,68] **67**

Table 2: Performance on 50 OGBench locomotion and manipulation tasks. Success rate (%), mean over three seeds with 95% bootstrap confidence intervals, best checkpoint over 1M gradient steps. QGRQL outperforms RQL overall and most on environments with headroom to improve such as `cube-double`, `puzzle-4x4-100M-sparse`, and `scene-sparse`.

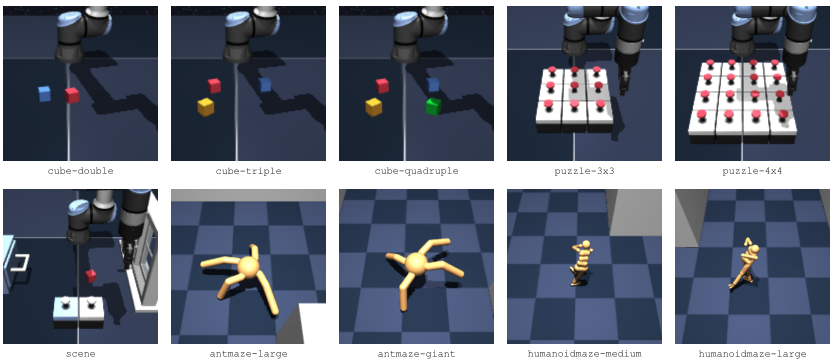

Figure 6: The ten OGBench environments used in our experiments.

B Experimental Details

We evaluate all methods on the official OGBench environments and datasets (Figure 6).

B.1 Hyperparameters Inherited from RQL

We fix the following hyperparameters for a fair comparison, following RQL (Oberai et al., 2026), unless otherwise mentioned.

Hyperparameter	**Value**
Gradient steps	1M
Optimizer	Adam (Kingma & Ba, 2015)
Learning rate	0.0003
Batch size	256
MLP size	[512, 512, 512, 512]
Nonlinearity	GELU (Hendrycks & Gimpel, 2016)
Target network update rate	0.005
Flow steps F	10
Discount factor γ	0.99 (default), 0.995 (`humanoidmaze`, `antmaze-giant`)
Action chunking size h	1 (locomotion), 5 (manipulation)
Ensemble size K	10
Critic target pessimism coefficient ρ	0.5 (default), 0 (`humanoidmaze`)

Table 3: Common hyperparameters, shared by RQL and QGRQL.

B.2 QGRQL Hyperparameters

QGRQL introduces the following additional hyperparameters.

Hyperparameter	**Value**
Gradient ascent steps k	10
Ascent step size β	0.1
L_2 ball radius R	0.2 (default), 0.05 (`cube-quadruple-100M`, `puzzle-4x4-100M`)

Table 4: QGRQL-specific hyperparameters.