Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

When Life Gives You BC, Make Q-functions: Extracting Q-values from Behavior Cloning for On-Robot Reinforcement Learning

Lakshita Dodeja[1,2], Ondrej Biza[1], Shivam Vats[2], Stephen Hart[1], Stefanie Tellex[1,2], Robin Walters[3], Karl Schmeckpeper[1], and Thomas Weng[1]

[1]Robotics and AI Institute, Cambridge, MA, USA
[2]Brown University, Providence, RI, USA
[3]Northeastern University, Boston, MA, USA

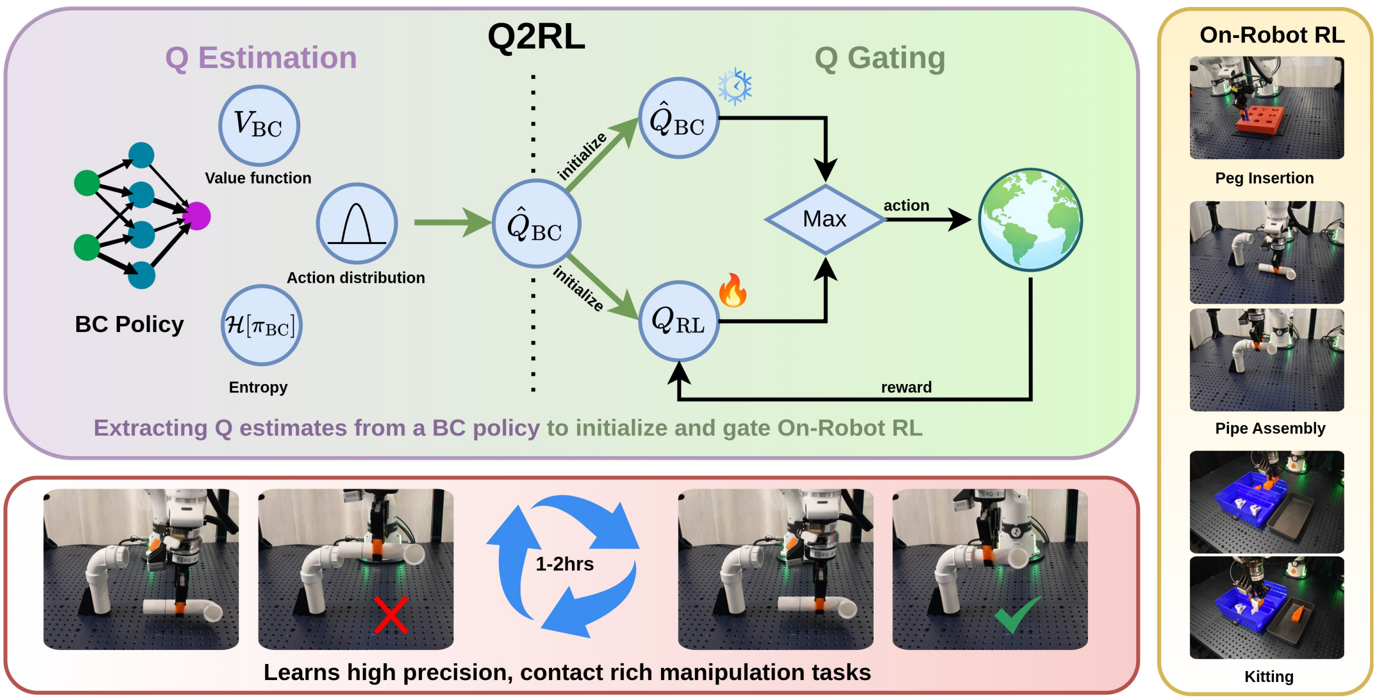


Fig. 1: *Q2RL* consists of Q-Estimation and Q-Gating. Q-Estimation extracts a Q-function from a BC Policy. Then, during RL policy training, Q-Gating selects and executes the BC or RL action with highest respective Q-value, updating the RL policy on the collected interactions. *Q2RL* learns contact-rich manipulation tasks in 1-2 hours of online interaction.

***Abstract*—Behavior Cloning (BC) has emerged as a highly effective paradigm for robot learning. However, BC lacks a self-guided mechanism for online improvement after demonstrations have been collected. Existing offline-to-online learning methods often cause policies to replace previously learned good actions due to a distribution mismatch between offline data and online learning. In this work, we propose *Q2RL*, Q-Estimation and Q-Gating from BC for Reinforcement Learning, an algorithm for efficient offline-to-online learning. Our method consists of two parts: (1) *Q-Estimation* extracts a Q-function from a BC policy using a few interaction steps with the environment, followed by online RL with (2) *Q-Gating*, which switches between BC and RL policy actions based on their respective Q-values to collect samples for RL policy training. Across manipulation tasks from D4RL and robomimic benchmarks, *Q2RL* outperforms SOTA offline-to-online learning baselines on success rate and time to convergence. *Q2RL* is efficient enough to be applied in an on-robot RL setting, learning robust policies for contact-rich and high precision manipulation tasks such as pipe assembly and kitting, in 1-2 hours of online interaction, achieving success rates of up to 100% and up to 3.75x improvement against the original BC policy. Code and video are available at https://q2rl.rai-inst.com/.**

I. Introduction

Behavior Cloning (BC) has been highly successful in robot learning due to its simple supervised training objective, which directly models an action distribution conditioned on observed states. However, BC does not have a mechanism for self-guided online improvement. When the training data for a BC policy is insufficient for the scope of the task, and/or changes in the environment induce even subtle distribution shifts, BC policy performance can collapse due to compounding errors from covariate shift [30].

Several research directions have been proposed to address the shortcomings of behavior cloning. Interactive imitation learning approaches finetune BC policies online, but require expert annotators [15]. Offline reinforcement learning (RL) learns a value function offline [1, 16] to guide online learning, but require pretraining on large datasets of positive and negative offline samples [18] for good performance. Offline-to-online learning approaches have also been proposed, but they often result in unlearning good BC actions [27, 39]. Another challenge faced by offline-to-online methods is the tendency to produce unsafe behaviors due to training an RL policy from scratch. In the on-robot RL setting in which online RL is trained directly on the robot, executing random exploratory actions is unsafe, causing significant wear and tear on the robot, workspace, and manipulated objects.

In this paper, our objective is to combine the benefits of pre-training a BC policy with the iterative improvement capability of online RL. We aim to keep good BC policy behavior, and when BC performs poorly, post-train with RL to find better actions. We propose Q-Estimation and Q-Gating from BC for Reinforcement Learning (*Q2RL*), an algorithm for efficient offline-to-online learning. In the *Q-Estimation* phase, we estimate the Q-function of the pretrained BC policy from initial online rollouts, then use it to initialize off-policy reinforcement learning. We are able to derive these estimated Q-values with just the action selection probability and entropy of the BC policy, *without any additional information of the underlying policy class or training distribution*. To ensure a stable transition from offline pretraining to online RL, we propose *Q-Gating*, which takes the Q-function from the Q-Estimation phase and uses it to guide online reinforcement learning. Two copies of the Q-function are used: the initially extracted Q function is kept frozen to preserve good BC actions, and a learnable RL Q-function is optimized to search for potentially better actions. At each state, *Q2RL* compares these Q-estimates to selectively execute BC or RL actions, ensuring that the RL policy is able to initialize learning from good BC actions and explore online to improve beyond the BC policy.

We provide extensive experiments in simulation and real-world on-robot RL to validate the method. *Q2RL* outperforms SOTA offline RL and offline-to-online methods across D4RL and robomimic benchmark tasks, improving the success rates from 50-60% for the original BC policy up to 80-100% for most environments. Ablation experiments show that both Q-Estimation and Q-Gating are valuable components of the algorithm. Our real-world experiments show that *Q2RL* learns contact-rich, high-precision manipulation tasks in 1-2 hours of online interaction, outperforming baselines especially as tasks increase in difficulty, and improving the success rate compared to the original BC policies by up to 100% and up to 3.75x improvement. Furthermore, our real-world experiments demonstrate that *Q2RL* can recover from distribution shifts in the environment.

Our contributions are as follows:

- “Q-Estimation”, an algorithm for estimating a Q-function from a behavior cloning policy and a few online rollouts. Q-Estimation can be applied to any policy class that provides action likelihoods and entropy.
- “Q-Gating”, an algorithm that guides online reinforcement learning via a gating mechanism to select between BC and RL actions.
- Experiments demonstrating that *Q2RL*, which combines Q-Estimation and Q-Gating, outperforms baselines on D4RL and robomimic benchmarks.
- Real-world, on-robot reinforcement learning experiments with *Q2RL* that surpass the original BC policy’s performance on contact-rich manipulation tasks and adapt to task distribution shifts unseen by the BC policy, in 1-2 hours of environment interaction.

II. Related Work

We review existing approaches to offline learning such as Behavior Cloning and offline RL, as well as offline-to-online approaches that combine offline learning with online RL. Then, we review on-robot reinforcement learning, which require sample-efficient techniques for real-world learning.

A. *Offline Learning*

Offline learning in robotics is broadly divided into Imitation Learning based approaches [38, 25] and Offline Reinforcement Learning approaches [16, 18]. Imitation learning approaches that train policies via supervised learning (i.e., Behavior Cloning) learn a direct mapping from states to an action distribution using demonstration data. Offline RL methods learn value functions that estimate expected returns from data generated by a behavior policy. Offline RL techniques like CQL [16] penalize out-of-distribution actions to address dataset shift, but this can introduce pessimistic bias in value estimates. IQ-Learn [10] learns soft Q-functions directly from demonstrations, enabling policy extraction through RL objectives. In contrast, our approach does not learn a new Q-function from demonstrations, but instead estimates the Q-values of an existing black-box BC policy to guide online improvement. Furthermore, offline RL methods are more suited to learn from large and potentially suboptimal datasets [9], and not necessarily from robotics datasets where successful demonstrations are easier to curate [28, 5]. BC approaches can further be divided into policy classes that explicitly parameterize the action distribution, such as Gaussian or Gaussian mixture heads paired with any network backbone [26, 32], and policy classes that implicitly represent the action distribution using generative denoising processes such as diffusion [6, 35] or flow-based models [19, 13]. *Q2RL* only requires access to the action selection probabilities and the entropy, which is easier to obtain for policy classes that explicitly parameterize the action distribution.

B. *Offline-to-Online Learning*

Offline learning can provide strong initial performance from readily available datasets, but policies trained purely offline cannot improve further without additional supervision or online interaction [15, 31]. Calibrated Q-Learning (CalQL) [27] aims to enable a smooth transition from offline to online RL by calibrating offline Q-values against a reference policy, but can struggle with limited offline data. WSRL [39] mitigates this by adding a warm-up phase that seeds the online replay buffer with rollouts from the offline policy. In contrast, other methods bypass offline RL by first pretraining an IL policy, then updating the behavior of the agent by adding corrective residual terms to the output action [33, 14, 37, 7]. These approaches require task-specific tuning of the residual action scale to ensure that the residual policy does not deviate from the BC policy, and can fail to optimize large action parameterizations. *Q2RL* does not use a residual formulation and allows the RL policy to learn completely different actions from the original BC policy, while still relying on the BC policy for guidance. Imitation Bootstrapped RL (IBRL) [12], most closely related to our approach, also combines a pretrained IL/BC policy with an RL policy and selects between their actions using a randomly initialized critic. A key benefit of *Q2RL* is that it *estimates the Q-function of the BC policy* from early online interaction, and uses it in conjunction with an RL Q-function to evaluate actions during online, off-policy reinforcement learning. In our evaluations, we show that *Q2RL* outperforms IBRL in simulated and real-world experiments, and that *Q2RL* does not require seeding the online replay buffer with high quality demonstrations, unlike IBRL.

C. On-Robot Reinforcement Learning

Online reinforcement learning on a real robot is a challenging but promising post-training strategy for achieving robust robotic policies. Since random exploration is costly on hardware, practical methods must begin with reasonable initial performance and adapt sample-efficiently. SERL [21] introduces a system designed specifically for on-robot RL. It provides a software suite to run online RL algorithms [11, 36, 2] using async actor and learner processes [34]. Extensions like HIL-SERL [23] incorporate human interventions to further improve efficiency, while other approaches like GCR [3] use reward shaping to guide learning. *Q2RL* uses the async framework for on-robot RL from SERL, but leverages readily available BC policies to provide strong initial behavior, without additional reward engineering or human intervention.

III. Problem Formulation

Assume we are given a behavior cloning policy π_{BC}, pretrained on a dataset $\mathcal{D}$ of successful human demonstrations for a task $\mathcal{M}_{\mathrm{train}} \in \mathfrak{M}$. Here $\mathfrak{M}$ denotes a class of tasks that share the same success condition; in the case of task distribution shift, $\mathcal{M}_{\mathrm{test}}$ and $\mathcal{M}_{\mathrm{train}}$ are drawn from task class $\mathfrak{M}$. The dataset consists of M trajectories of observation-action tuples:

$$\mathcal{D} = \{\tau_j\}_{j=1}^{M}\,, \qquad \tau_j = \{(o_{j,t}, a_{j,t})\}_{t=1}^{T_j}\,, \tag{1}$$

where T_j is the length of trajectory τ_j, and $o_{j,t}, a_{j,t}$ are the observation and action, respectively.

Assume the BC policy π_{BC} performs sub-optimally at test time (i.e. failing to achieve the success condition), because the amount of training data or number of training epochs were insufficient, or due to distribution shifts induced by changes in environment conditions such as lighting variations, the presence of visual distractors, modified robot hardware or controllers, or other changes to environment dynamics.

Our objective is to improve performance on task $\mathcal{M}_{\mathrm{test}}$ beyond the original BC policy's performance. To achieve this, we assume access to the test environment for online interactions and a sparse reward signal. For this online phase, we formulate the task as a Markov Decision Process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \rho_0, \gamma\}$, where $\mathcal{S} \in \mathbb{R}^n$ and $\mathcal{A} \in \mathbb{R}^m$ represent states and actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, ρ_0 is the probability distribution over initial states, and $\gamma \in [0, 1)$ is a discount factor.

A. Background

Our proposed approach will involve estimating a Q-function, or the expected return from taking an action $a \in \mathcal{A}$ from state $s \in \mathcal{S}$, then following a policy π thereafter:

$$Q_\pi(s,a) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a\right]. \tag{2}$$

The value function for policies is defined as the expectation over Q-values:

$$V(s) = \mathbb{E}_{a\sim\pi}\left[Q(s,a)\right]. \tag{3}$$

Although $Q_\pi(s,a)$ is a standard action-value function, it can also be interpreted as defining an energy-based model over actions. In particular, treating $E(s,a) = -Q(s,a)$ as an energy yields a Boltzmann policy

$$\pi(a \mid s) \propto \exp\left(\frac{1}{\alpha} Q(s,a)\right), \tag{4}$$

where α is a temperature parameter.

IV. Q2RL: Q-Estimation and Q-Gating from Behavior Cloning for Reinforcement Learning

In this section, we present *Q2RL*, an algorithm for estimating Q-values from a BC policy to guide online reinforcement learning. *Q2RL* consists of two phases: Q-Estimation (Sec. IV-A) and Q-Gating (Sec. IV-B). See Alg. 1 for a high-level description and Fig. 1 for a visual overview.

Algorithm 1 Q2RL

Given: behavior cloning policy π_{BC}

Roll out π_{BC} for N environment steps
Estimate $\hat{Q}_{\mathrm{BC}}$ using Eq. 6 ▷ Q-Estimation
Initialize Q_{RL} with $\hat{Q}_{\mathrm{BC}}$, Initialize π_{RL}

for $t = 0 \ldots T$ environment steps **do**
    $a_{\mathrm{BC}} \sim \pi_{\mathrm{BC}}(s_t)$
    $a_{\mathrm{RL}} \sim \pi_{\mathrm{RL}}(s_t)$
    Select a_t using Eq. 10 ▷ Q Gating

    Observe next state s_{t+1} and reward r_t
    Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{B}$
    Sample a minibatch $\{(s, a, r, s')\} \in \mathcal{B}$
    Update Q_{RL}, π_{RL} using off-policy RL
end for

A. Q-Estimation from a Behavior Cloning Policy

In the first phase of our method, we estimate the Q-function of a pretrained BC policy π_{BC}. We first derive an analytic formula for the Q-function in terms of the value function, policy likelihoods, and entropy. We assume the action distribution of the human demonstrations $\mathcal{D}$ used to train π_{BC} is approximately a Boltzmann distribution, which has been widely used to model human actions [17, 20, 24, 4]. Assuming the BC policy's action distribution $\pi_{\mathrm{BC}}(a \mid s)$ can be expressed as a Boltzmann distribution with respect to Q_{BC} gives:

$$\pi_{\mathrm{BC}}(a \mid s) \approx \frac{\exp\left(Q_{\mathrm{BC}}(s,a)/\alpha\right)}{\int \exp\left(Q_{\mathrm{BC}}(s,a')/\alpha\right) da'}. \tag{5}$$

Using Eq. 3 and Eq. 5, we derive the equation for the Q-function Q_{BC} using the value function, log probability of the action, and entropy:

$$\hat{Q}_{\mathrm{BC}} = V_{\mathrm{BC}}(s) + \alpha \log \pi_{\mathrm{BC}}(a \mid s) + \alpha \mathcal{H}[\pi_{\mathrm{BC}}(\cdot \mid s)]. \tag{6}$$

Appendix A provides a complete derivation for Eq. 6.

Next, we detail how each term in Eq. 6 is computed or approximated. First, we must estimate the value function V_{BC}. We collect rollouts of the BC policy for a few initial online interaction episodes $\{\tau_i\}_{i=1}^{N}$ and calculate the Monte Carlo returns under the online reward function:

$$\hat{V}_{BC}(s_t^{(i)}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=t}^{\tau_i}\gamma^k r_{t+k}^{(i)}, \text{ for } t \in \{1, \ldots, \tau_i\}. \tag{7}$$

These Monte Carlo returns are used to train a value estimator.

The remaining terms of Eq. 6 are derived from the BC policy π_{BC}. Our method is agnostic to the BC policy class, as long as it provides the log probabilities of actions and entropy. We provide the remaining components of Eq. 6 using Gaussian policies as an illustrative case. The log probability of the actions $\log \pi(a \mid s)$ for Gaussian policies has a closed-form expression:

$$\log \pi(a \mid s) = \frac{1}{2}\sum_{i=1}^{d}\left[\frac{(a_i - \mu_i(s))^2}{\sigma_i^2(s)} + \log(2\pi\sigma_i^2(s))\right], \tag{8}$$

where (μ, σ) is Gaussian distribution output by the policy. Finally, the entropy $\mathcal{H}[\pi(\cdot \mid s)]$ for Gaussian policies also has a closed-form expression:

$$\mathcal{H}[\pi(\cdot \mid s)] = \frac{1}{2}\log\left(2\pi e\,\sigma^2(s)\right). \tag{9}$$

See Appendix A for a similar treatment for Gaussian Mixture Models (GMMs).

B. Q-Gating for Online Reinforcement Learning

We now describe Q-Gating, our approach to using $\hat{Q}_{\mathrm{BC}}$ estimated from Eq. 6 to guide online reinforcement learning. We first initialize two separate Q-functions, both from $\hat{Q}_{\mathrm{BC}}$: $\hat{Q}_{\mathrm{BC}}$ and Q_{RL}. $\hat{Q}_{\mathrm{BC}}$ is kept frozen to serve as a stable reference for the BC policy. Q_{RL} initialization involves an initial supervised training period to match $\hat{Q}_{\mathrm{BC}}$ on the same BC rollouts used for Q-Estimation. Q_{RL} then serves as the critic for the RL policy, and is updated through online interaction. During online training, we sample both the BC policy and RL policy for actions a_{BC} and a_{RL}, respectively. We evaluate a_{BC} using $\hat{Q}_{\mathrm{BC}}$ and a_{RL} using Q_{RL} to get their estimated Q-values. The action corresponding to the greater of these two values is executed:

$$a = \begin{cases} a_{\mathrm{BC}} \sim \pi_{\mathrm{BC}}(s), & \hat{Q}_{BC}(s, a_{\mathrm{BC}}) > Q_{RL}(s, a_{\mathrm{RL}}) \\ a_{\mathrm{RL}} \sim \pi_{\mathrm{RL}}(s_t), & \text{otherwise.} \end{cases} \tag{10}$$

We propose this Q-Gating mechanism to preserve good BC action proposals while still enabling policy improvement through RL. $\hat{Q}_{\mathrm{BC}}$ represents the (estimated) value of taking an action in the current state, then following the BC policy thereafter. We evaluate BC actions through a frozen $\hat{Q}_{\mathrm{BC}}$, so that BC actions with high value under $\hat{Q}_{\mathrm{BC}}$ are likely to be executed. On the other hand, Q_{RL} is only initialized as $\hat{Q}_{\mathrm{BC}}$, and trains on samples with either BC or RL actions. Q_{RL} therefore improves its estimate of both BC and RL actions, enabling the reinforcement learning policy to surpass BC performance through online exploration and interaction. Note that our approach does not require access to the training data for π_{BC}; we rely on our Q-estimate $\hat{Q}_{\mathrm{BC}}$ to bootstrap online reinforcement learning.

C. Implementation Details

We assume a temperature of $\alpha = 1$ for the BC policy's Boltzmann parameterization in Eq. 6. *Q2RL* can be applied to any behavior cloning policy that provides log probabilities of actions and entropy; in this work, we demonstrate the approach using Gaussian policies and GMMs. Similarly, though *Q2RL* is agnostic to the choice of off-policy RL algorithm, we use SAC optimized for real-world learning [11, 39].

To prevent the RL policy from deviating too far from the BC state-action distribution, we also weight the RL actor loss with an auxiliary BC loss [29]. While this regularization term can occasionally prevent the policy from reaching perfect success rates, our experiments in the following section show that training with BC regularization still outperforms baselines and plays a crucial role in producing safe, smooth behaviors on real robotic systems.

V. Experiments

Our experiments are designed to answer two key questions: (i) does *Q2RL* improve performance beyond the BC policy faster and better than baseline approaches? (ii) Does *Q2RL* use high-value actions from the BC policy during online RL? To provide a comprehensive evaluation, we conduct experiments across multiple simulation environments using BC policies parameterized by Gaussian and GMM-RNN architectures, and compare against offline-to-online RL and BC-to-RL baselines. Our evaluation spans both state-based and image-based observation settings. Finally, we validate the practicality of our approach through real-world RL experiments, assessing whether *Q2RL* can support online learning on physical hardware.

Fig. 2: Tasks from D4RL (`Kitchen-Complete`, `Adroit-Pen`, `Adroit-Door`) and robomimic (`Lift`, `Can`, `Square`).

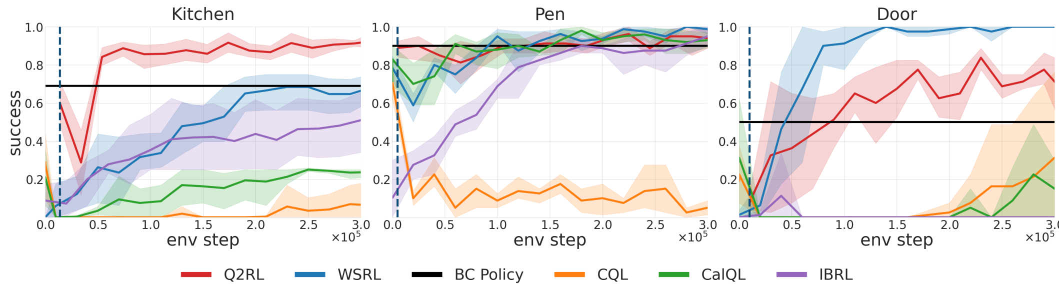


Fig. 3: Results on D4RL. Plots begin when a method starts online interaction. The dotted blue line signifies the end of the Q-Estimation phase and the start of online RL with Q-Gating for our method. Results are reported over 20 trajectories and 5 seeds, with 95% confidence intervals.

A. Simulation Environments

We test *Q2RL* on tasks from the D4RL [8] and robomimic [26] benchmarks. All tasks are visualized in Fig 2.

1) ***D4RL:*** We run experiments on two Adroit manipulation tasks: *Pen* and *Door*. The Pen task involves dexterous in-hand manipulation, requiring the robotic hand to reorient a pen to a target configuration, while Door consists of a standard door-opening task. In addition, we evaluate performance on the *Kitchen* environment, which features a Franka robot interacting with multiple objects to achieve a multi-stage goal configuration. For these experiments, we use the offline Kitchen-Complete dataset, which contains 19 successful demonstrations which are representative of standard robot learning setups where we only have access to a few successful trajectories. We conduct all D4RL experiments without access to offline training data during the online RL phase.

2) ***robomimic:*** We run experiments on three widely used robomimic tasks, *Lift*, *Can*, and *Square*, all featuring a Franka robotic arm. In the Lift task, the robot must lift a cube from a randomly sampled position on the table. The Can task requires the robot to grasp a can from one side of the table and place it into the correct bin among four target bins located on the opposite side. The Square task involves picking up a square nut from the table and inserting it onto a square peg. For robomimic tasks, we test both with and without access to offline training data during the online RL phase.

B. Baselines

We compare *Q2RL* with the following baselines:

1) ***CQL:*** Conservative Q-Learning [16] is an offline RL method that trains exclusively on an offline dataset by prioritizing in-distribution actions while penalizing out-of-distribution actions. This induces a pessimistic bias in the learned Q-function, which often requires a recalibration phase during online interaction.

2) ***CalQL:*** Calibrated Q-Learning [27] addresses the pessimism of Q-value estimates in CQL by calibrating the critic loss with respect to the value of a reference policy. However, despite this calibration, transitioning to online training can still lead to unlearning of its offline pretraining.

3) ***WSRL:*** Warm-Start RL [39] performs for offline-to-online RL without explicit data retention. To mitigate unlearning during the transition from offline to online training, WSRL introduces a warm-up phase in which the online replay buffer is seeded with rollouts generated by the offline pretrained policy. However, a key limitation of WSRL is its dependence on the quality of the offline pretraining data.

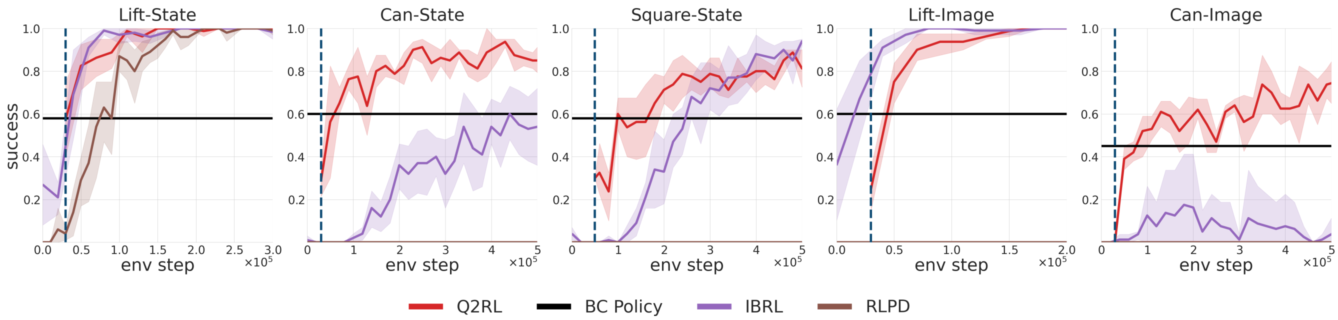
4) ***RLPD:*** RL with Prior Demonstrations [2] incorporates offline data directly into online learning by uniformly sampling from both offline and online replay buffers during actor and critic updates.

5) ***IBRL:*** Imitation Bootstrapped RL [12] uses a pre-trained BC policy to bootstrap Q-value estimates and action proposals for RL. IBRL randomly initializes the Q-function and uses demonstration data to seed the online replay buffer.

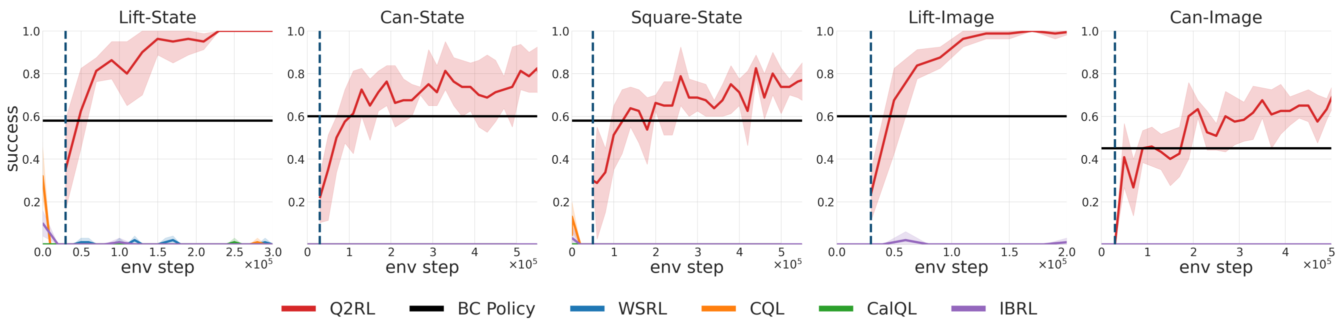
Additional details on simulation experiments can be found in Appendix B.

C. Results on D4RL

We present the results for Kitchen, Door and Pen tasks from D4RL in Fig 3. Offline RL methods perform poorly on the **Kitchen** task because the offline Kitchen-Complete dataset only contains successful demonstrations. Even with online RL after offline pre-training, these methods fail to improve over BC for the Kitchen task. *Q2RL*, on the other hand, starts



(a) Results with data seeded in the online replay buffer.



(b) Results without data seeded in the online replay buffer.

Fig. 4: Robomimic Results. The plot for each method begins at the start of online interaction. The dotted blue line signifies the end of the Q-Estimation phase and the start of online RL with Q-Gating for our method. Results are reported over 20 trajectories and 5 seeds, with 95% confidence intervals.

with better initial performance than the baselines and improves beyond the BC policy, which we attribute to Q-Estimation and online RL with Q-Gating.

In the **Pen** task, the offline data is able to provide sufficient pretraining for both BC and offline RL methods. We observe that *Q2RL* starts with near-optimal performance and maintains it throughout online learning. WSRL and CalQL are also able to reach high performance with online interaction on Pen, whereas CQL experiences a drop in performance, likely due to its conservative Q-value estimates. IBRL is also eventually able to reach high performance.

For the **Door** task, WSRL converges to a higher success rate than the baselines and *Q2RL*. However, from the rollout videos we observe that the behavior of the WSRL policy diverged from the original human demonstrations, and that it has learned to exploit the simulator through online RL, resulting in actions that would not be feasible in the real world. (Fig. 15, Appendix C). *Q2RL* uses Q-Estimation and Q-Gating to remain close to the BC policy, resulting in actions that can be realistically executed in the real world.

D. Robomimic Results

Next, we evaluate *Q2RL* on both state-based and image-based robomimic tasks. We use two cameras for image-based tasks, the agentview of the workspace and wrist camera. We implement *Q2RL* and all baselines with a learnable CNN encoder for the images to keep the implementation standard and comparisons fair. As these tasks are more challenging than the D4RL benchmarks, we consider two experimental settings: one in which training data is available in the online replay buffer, and another in which the policy does not have access to any offline training data during online learning.

1) Access to BC Training Data: Results for the robomimic tasks with access to training data during online learning are shown in Fig. 4a. When training data is available in the online replay buffer, IBRL performs competitively. IBRL randomly initializes the critic for off-policy RL and has no mechanism to preserve the initial performance of the BC policy, causing online interaction to begin with limited or no meaningful performance. In contrast, *Q2RL* begins online interaction with non-trivial initial performance rather than starting from near-zero success, which is particularly important for real-world deployment where unsafe or unproductive exploration is costly. Moreover, by restricting learning to targeted policy improvements, *Q2RL* converges to high-performing policies more rapidly. We also observe that for Can-Image, a harder high-dimensional task, IBRL is unable to reach base policy performance. We note that in the original IBRL paper, IBRL is able to converge for the Can-Image task, but we hypothesize that this is due to other optimizations used, such as specialized ViT encoders. *Q2RL* on the other hand is able to outperform the base policy with only the standard set of encoders. RLPD succeeds only on simpler tasks such as Lift and struggles to converge on the more challenging Can and Square tasks.

2) No Access to BC Training Data: BC policies are often provided by third parties without the data or infrastructure used to train them. Due to such occurrences, we evaluated our approach in cases where pre-training data was not accessible. Results for the robomimic tasks without access to training data during online learning are shown in Fig. 4b. Offline RL and offline-to-online RL methods are unable to learn for harder robotic tasks with limited offline datasets. Without access to data, IBRL is also unable to learn any meaningful performance. Early in training, the randomly initialized RL policy can propose arbitrary actions that may receive spuriously high Q-values from an untrained critic. To mitigate this issue, IBRL relies on seeding the replay buffer with training data, so that the critic learns to assign higher value to BC action proposals than to random RL actions. In the absence of a seeded replay buffer, IBRL loses this mechanism to stabilize initial Q-values, leading to unreliable action selection. *Q2RL* is able to continually improve performance even in the absence of training data. We attribute *Q2RL*'s strong performance without seeding to the effectiveness of the proposed Q-Estimation step. Q-Estimation computes Q_{BC} using BC policy rollouts *only*; it does *not* require expert demonstrations, nor a diverse offline dataset. *Q2RL* therefore has broad applicability in both limited access, memory-constrained settings where expert data is unavailable, as well as in unconstrained settings where expert data can seed online RL. We also conduct a thorough evaluation for seeding the replay buffer with different amounts of data in Appendix C.

E. Analysis of BC Actions Used During Online Training

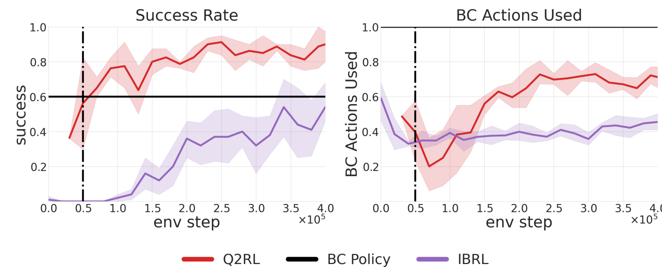
Next, we investigate the relationship between success rate and the ratio of BC vs. RL actions used during online training. Fig. 5 shows the success rate and the fraction of BC actions selected by *Q2RL* vs. IBRL on Can-State and Square-State tasks. The vertical dotted line serves as a visual reference indicating when *Q2RL*'s online performance approximately matches the original BC policy performance.

We observe that *Q2RL* rapidly recovers the original BC performance within only a few online interaction steps, demonstrating that it identified and exploited high-value actions of the BC policy. This performance is achieved while using BC actions less than half of the time, indicating that the RL policy also learned to produce effective actions. With continued training, as Q_{RL} estimates improve, Q-Gating more reliably identifies states where each policy has strong performance, so that Q2RL relies on BC for simple motions and reserves RL for difficult, contact-rich segments.

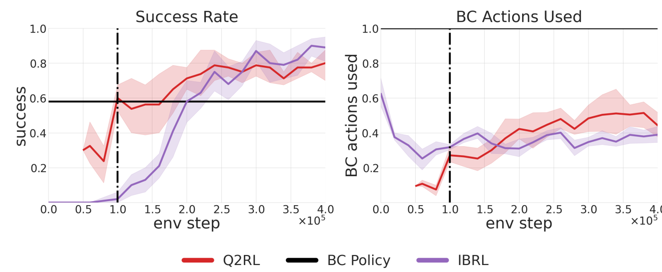
In contrast, IBRL selects a similar fraction of BC actions, but requires substantially more interaction to reach the same performance. Further, the ratio of BC actions used by IBRL during training stays relatively flat, indicating that it trades off between BC and RL actions less efficiently than *Q2RL*.

F. Ablations

We ablate Q-Initialization and Q-Gating in Fig. 6. Note that Q-Initialization, or initializing Q_{RL} with $\hat{Q}_{BC}$, is distinct



(a) Can-State



(b) Square-State

Fig. 5: BC vs. RL Actions Used During Online Training. *Q2RL* optimizes BC vs. RL action selection via Q-Gating, and achieves higher success rate earlier than IBRL.

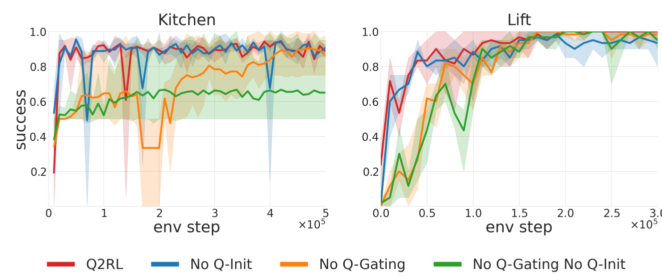


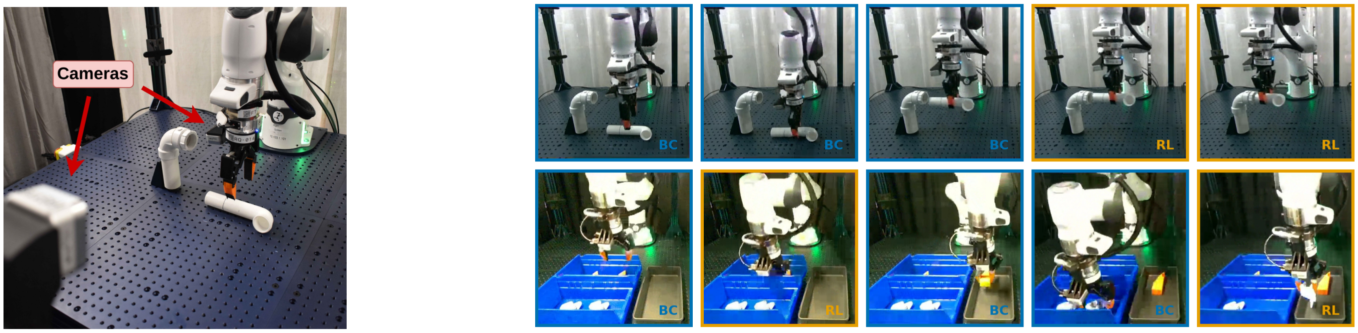
Fig. 6: Q-Initialization and Q-Gating Ablations.

from Q-Estimation, which is the algorithm for estimating $\hat{Q}_{BC}$ (Sec. IV-A). In the No Q-Gating condition, BC and RL action proposals are evaluated using the same Q-function during online RL. We see that Q-Gating is critical for achieving high performance. When Q-Gating is enabled, Q-Initialization yields performance comparable to a randomly initialized critic; however, we retain this initialization because it provides a safer starting distribution that remains closer to the BC policy, which is preferable for on-robot RL. See Appendix C for additional ablation studies.

G. Real World RL Experiments

1) Experiment Setup: We run all experiments on a 7-DOF Franka Panda Robot with a Robotiq 2F-85 gripper. For all tasks, the robot receives the end effector pose, gripper width, and 84×84 RGB images from a workspace camera and wrist camera (see Fig. 7a). The robot outputs delta pose actions executed by a Cartesian impedance controller. We evaluate on the following tasks (see Fig. 1):

Peg Insertion. This insertion task from FMB [22] has been used in recent on-robot RL methods [21, 39]. Success occurs when the peg is fully inserted into the board.



(a) On-Robot RL Setup.

(b) Trained *Q2RL* policies for pipe assembly and kitting tasks.

Fig. 7: Real World Experiments. (a) A Franka Panda arm with a Robotiq 2F-85 gripper, with RGB images from wrist and workspace Intel RealSense D405s. (b) We analyze usage of BC (**blue**) vs. RL (**yellow**) actions in representative trajectories executed by trained *Q2RL* policies. For pipe assembly, *Q2RL* relies on BC for grasping and initial alignment, then switches to RL for high-precision, contact-rich insertion. For kitting, the BC policy was trained in a task setting with only one object present in each bin; *Q2RL* learns to complete the task with multiple objects in each bin. In the modified setting, *Q2RL* uses BC actions for behaviors common to both task settings (e.g. moving between bins), and uses RL for part grasping and placement.

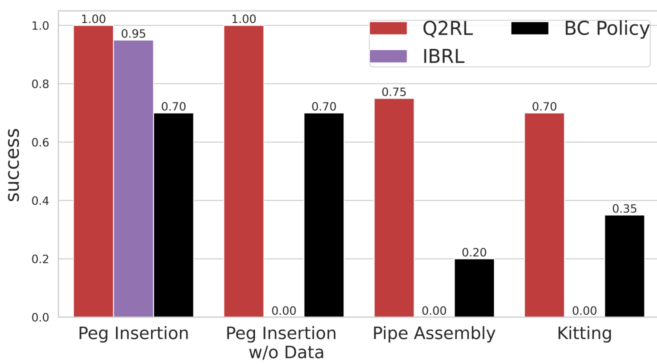


Fig. 8: On-Robot RL Results. Task success is reported over 20 trials per method. *Q2RL* significantly improves BC policy performance on every task. IBRL fails when (1) demonstrations are not seeded in the replay buffer (w/o Data), and (2) when tasks involve large action spaces and long horizon manipulation (even with seeding).

Plumbing Pipe Assembly. This task involves both grasping and low tolerance insertion. Insertion requires rotating the gripper to align the pipe with the fixture. Success occurs when the pipe is fully inserted into the fixture.

Kitting with Task Distribution Shift. In this task, two parts must be retrieved from their respective bins and placed in a kitting tray. As the longest horizon task, this task requires multiple picks and places. Success occurs when the robot has placed both parts in the kitting tray. The BC policy for this task was only trained on demonstrations with a single object per bin. To evaluate robustness to task distribution shift, the test task setting has two objects per bin, placed in new locations.

GMM-RNN BC policies were trained for each task on 50 to 100 demos collected by a human operator using a 3D Connexion Spacemouse. We compare *Q2RL* with IBRL for offline-to-online improvement with the pre-trained BC policies. During online learning, a termination signal and sparse reward signal were provided by a human operator when the task's success conditions were met. Environment resets were also handled by a human operator. Additional details on the real experiment setup can be found in Appendix D.

2) Real World Results: For the Kitting task, all results are for the modified setting (two parts in each bin instead of one). The best checkpoint after up to 2.5 hours of online training is used for each method, representing no more than 170k gradient steps and 80k environment steps (much fewer than in simulation). From Fig. 8, ***Q2RL* significantly improves performance compared to the original BC policy, achieving a 1.4x improvement and 100% success on the Peg Insertion task, 3.75x improvement on Pipe Assembly, and 2x improvement on Modified Kitting.** While IBRL performed on par with *Q2RL* for the Peg Insertion task, it achieved zero success when demonstration samples were not seeded in the online replay buffer. Even with data seeding, when tasks involved large action spaces or were long horizon, IBRL had zero success, failing to recover BC performance. We attribute IBRL's poor performance to its use of a single, randomly initialized Q-function to select BC vs. RL actions, which hinders it from accurately rating BC actions, especially for long horizon, sparse reward tasks. In contrast, we initialize Q_{RL} from $\hat{Q}_{\mathrm{BC}}$, and our proposed Q-Gating procedure uses both a frozen $\hat{Q}_{\mathrm{BC}}$ and trainable Q_{RL} to score respective actions more accurately.

Safety. During the course of training IBRL for Peg Insertion, we noted two safety violations, in which the Franka exerted excessive force against the board and caused the robot to fault. These safety violations occurred despite careful tuning of controller gains during preliminary testing that was subsequently kept fixed for all reported runs. *Q2RL* experienced no such safety violations, and we qualitatively observed that it produced relatively smoother and safer actions early in training: we attribute this behavior to a combination of the auxiliary BC loss, Q-Gating, and Q_{RL} initialization with $\hat{Q}_{\mathrm{BC}}$.

Qualitative Results. Fig. 7b shows frames from representative rollouts of trained *Q2RL* policies, annotated to indicate BC vs. RL action selection. We observe that *Q2RL* selects BC actions to perform behaviors supported by the BC policy's training data, e.g. initial pipe grasping and alignment for the Pipe Assembly task, before switching to RL actions for high-precision, contact-rich insertion. With Modified Kitting, BC actions were selected for moving between bins, with RL actions handling parts of the task that had shifted from their original setting, e.g. grasping parts from new locations. Other cases of switching between BC and RL actions corresponded to recovery behaviors, such as insertion re-alignment and re-grasping. Not all rollouts or trained *Q2RL* policies exhibit these behaviors exactly as described here, but we generally find meaningful switching between BC and RL actions. Please refer to the website for videos of annotated rollouts, and Appendix D for additional details on results.

VI. Conclusion

We presented *Q2RL*, an algorithm that improves the performance of a behavior cloning policy through online reinforcement learning. *Q2RL* estimates a Q-function from the BC policy, then uses it to initialize and guide online RL. Our approach does not require access to the BC policy's original training data, and is compatible with any BC policy class that provides action likelihoods and entropy. In on-robot reinforcement learning experiments, *Q2RL* successfully improves performance on high precision, contact-rich manipulation tasks within a few hours of online interaction.

One limitation of our work is that *Q2RL* requires the BC policy to provide action likelihoods and entropy; as future work, we aim to extend *Q2RL* to other policy classes that do not provide these by default, such as diffusion and flow matching policies.

Acknowledgments

This work was supported in part by ONR grant N00014-22-1-2592 and ONR grant "Long-Term Autonomy for Ground and Aquatic Robotics" N00014-24-1-2784. The authors thank Justin Rojas, Will Heitman, and Steve Proulx for assistance with real robot experiments and hardware, Victoria Coleman and Colin Kohler for real robot infrastructure support, and Eric Rosen for helpful paper feedback.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pages 104–114. PMLR, 2020.

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

[3] Ondrej Biza, Thomas Weng, Lingfeng Sun, Karl Schmeckpeper, Tarik Kelestemur, Yecheng Jason Ma, Robert Platt, Jan-Willem van de Meent, and Lawson L.S. Wong. On-robot reinforcement learning with goal-contrastive rewards. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4797–4805, 2025. doi: 10.1109/ICRA55743.2025.11128466.

[4] Andreea Bobu, Andrea Bajcsy, Jaime F Fisac, and Anca D Dragan. Learning under misspecified objective spaces. In *Conference on Robot Learning*, pages 796–805. PMLR, 2018.

[5] Lawrence Yunliang Chen, Simeon Adebola, and Ken Goldberg. Berkeley UR5 demonstration dataset. https://sites.google.com/view/berkeley-ur5/home.

[6] 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.

[7] Lakshita Dodeja, Karl Schmeckpeper, Shivam Vats, Thomas Weng, Mingxi Jia, George Konidaris, and Stefanie Tellex. Accelerating residual reinforcement learning with uncertainty estimation. *arXiv preprint arXiv:2506.17564*, 2025.

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

[9] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708*, 2019.

[10] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34:4028–4039, 2021.

[11] 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*, pages 1861–1870. Pmlr, 2018.

[12] Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024. doi: 10.15607/RSS.2024.XX.056.

[13] Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Siddharth Ancha. Streaming flow policy: Simplifying diffusion / flow-matching policies by treating action trajectories as flow trajectories. *arXiv preprint arXiv:2505.21851*, 2025.

[14] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[15] Michael Kelly, Chelsea Sidrane, Katherine Rose Driggs-Campbell, and Mykel J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In

International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019, pages 8077–8083. IEEE, 2019.

[16] 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.

[17] Cassidy Laidlaw and Anca Dragan. The boltzmann policy distribution: Accounting for systematic suboptimality in human models. In *ICLR*, 2022.

[18] 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.

[19] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[20] R Duncan Luce et al. *Individual choice behavior*, volume 4. Wiley New York, 1959.

[21] Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16961–16969. IEEE, 2024.

[22] Jianlan Luo, Charles Xu, Fangchen Liu, Liam Tan, Zipeng Lin, Jeffrey Wu, Pieter Abbeel, and Sergey Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *The International Journal of Robotics Research*, 44(4):592–606, 2025.

[23] Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10 (105):eads5033, 2025.

[24] Jim Mainprice, Rafi Hayne, and Dmitry Berenson. Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 885–892, 2015. doi: 10.1109/ICRA.2015.7139282.

[25] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv preprint arXiv:2003.06085*, 2020.

[26] 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. *arXiv preprint arXiv:2108.03298*, 2021.

[27] 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.

[28] Soroush Nasiriany, Tian Gao, Ajay Mandlekar, and Yuke Zhu. Learning and retrieval from prior data for skill-based imitation learning. *arXiv preprint arXiv:2210.11435*, 2022.

[29] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[30] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[31] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[32] Ziyad Sheebaelhamd, Michael Tschannen, Michael Muehlebach, and Claire Vernade. Quantization-free autoregressive action transformer. *arXiv preprint arXiv:2503.14259*, 2025.

[33] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[34] You Liang Tan. Agentlace, framework for distributed agent policy, May 2024. URL https://github.com/youliangtan/agentlace.

[35] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[36] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International conference on learning representations*, 2021.

[37] Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

[38] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5628–5635. Ieee, 2018.

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