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

Author Names Omitted for Anonymous Review

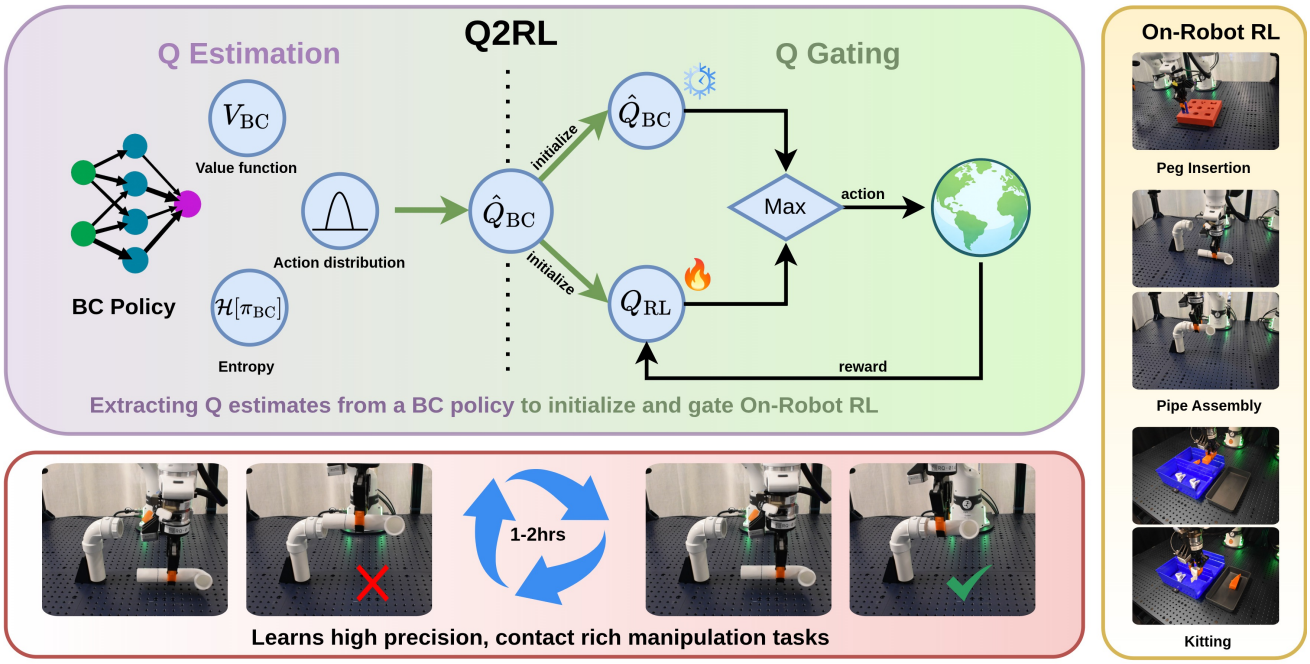


Fig. 3: *Q2RL* consists of Q-Estimation and Q-Gating. Q-Estimation extracts a Q function from a BC Policy. Then, during RL policy training, Q-Gating selects and executes the BC or RL action with highest respective Q value, updating the RL policy on the collected interactions. *Q2RL* learns contact-rich 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 ditribution 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. Video overview here.**

***Index Terms*—Robot Learning, Reinforcement Learning**

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 sufficient 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 [12].

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 use RL to explore when BC performs poorly. 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, one fixed as the initially extracted Q function to preserve good BC actions, and a learnable RL Q function to enable exploration of 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. 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 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 aQfunction 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.
- Real-world, on-robot reinforcement learning experiments with *Q2RL* that achieve performance over the original BC policy on contact-rich manipulation tasks in 1-2 hours of environment interaction, and adapt to task distribution shifts unseen by the BC policy.

I. 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 $\mathcal{M}_{\text{train}} \in \mathfrak{M}$. Here $\mathfrak{M}$ denotes a class of tasks that share the same success condition; in the case of task distribution shift, $\mathcal{M}_{\text{test}}$ and $\mathcal{M}_{\text{train}}$ 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.

Our objective is to improve performance on task $\mathcal{M}_{\text{test}}$ beyond the original BC policy's performance. 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.

II. 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. II-A) and Q-Gating (Sec. II-B). See Fig. 3 for a visual overview. We also provide a high level algorithm in the appendix.

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 have been used to model human actions in prior work [5, 6]. Assuming the BC policy's action distribution $\pi_{\text{BC}}(a \mid s)$ can be expressed as a Boltzmann distribution with respect to Q_{BC} gives:

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

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:

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

Appendix B 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:

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

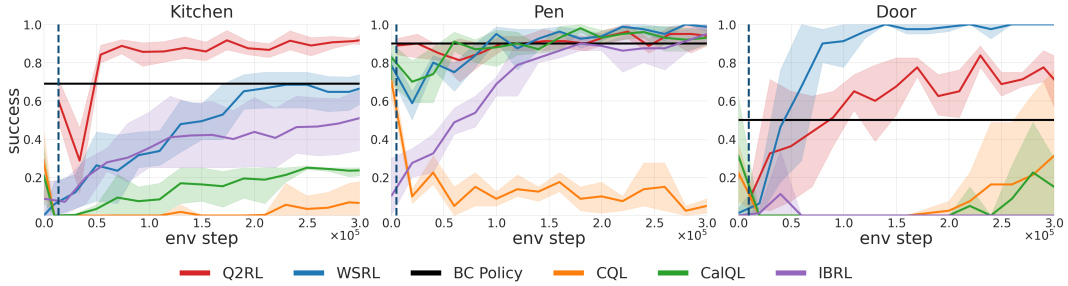


Fig. 4: Results on D4RL. Plots begin when a method starts online interaction. The dotted blue line signifies the end of the Q-Estimation phase and the start of online RL with Q-Gating for our method.

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. Nevertheless, 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:

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

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:

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

See Appendix B for a similar treatment for GMMs.

B. Q-Gating for Online Reinforcement Learning

We now describe Q-Gating, our approach to using $\hat{Q}_{\text{BC}}$ estimated from Eq. 6 to guide online reinforcement learning. We first initialize two separate Q-functions, both from $\hat{Q}_{\text{BC}}$: $\hat{Q}_{\text{BC}}$ and Q_{RL}. $\hat{Q}_{\text{BC}}$ is kept frozen to serve as a stable reference for the BC policy. Q_{RL} 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 $\hat{Q}_{\text{BC}}$ and a_{RL} using Q_{RL} to get their estimated Q values. The action corresponding to the greater of these two values is executed:

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

We propose this Q-Gating mechanism to preserve good BC action proposals while still enabling policy improvement through RL. $\hat{Q}_{\text{BC}}$ 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 $\hat{Q}_{\text{BC}}$, so that BC actions with high value under $\hat{Q}_{\text{BC}}$ are likely to be executed. On the other hand, Q_{RL} is only initialized as $\hat{Q}_{\text{BC}}$, 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 $\hat{Q}_{\text{BC}}$ to bootstrap online reinforcement learning.

C. Implementation Details

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 [11]. 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.

III. 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?

A. Simulation Environments

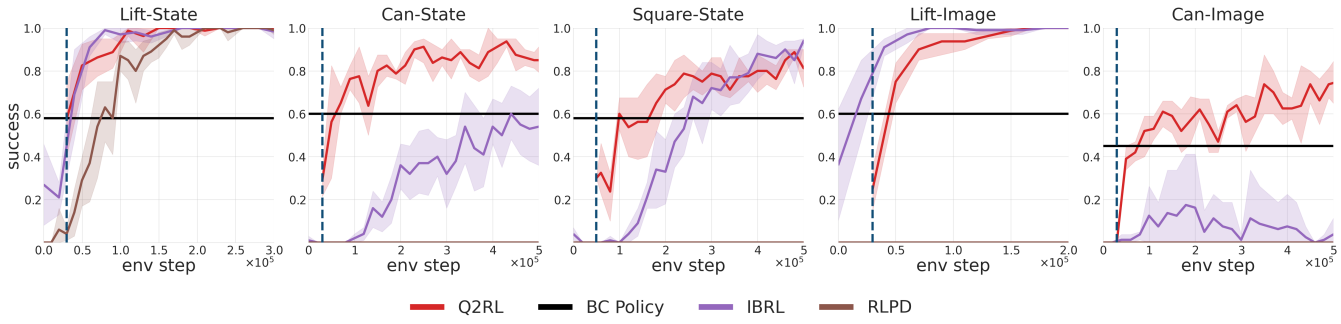
We test *Q2RL* on tasks from the D4RL [2] and robomimic [9] benchmarks. For **D4RL**, we run experiments on two Adroit manipulation tasks: *Pen* and *Door*. We also test on the *Kitchen* environment with the Kitchen-Complete dataset containing 19 successful demonstrations. For **robomimic** We run experiments on three widely used robomimic tasks, *Lift*, *Can*, and *Square*, all featuring a Franka robotic arm.

B. Baselines

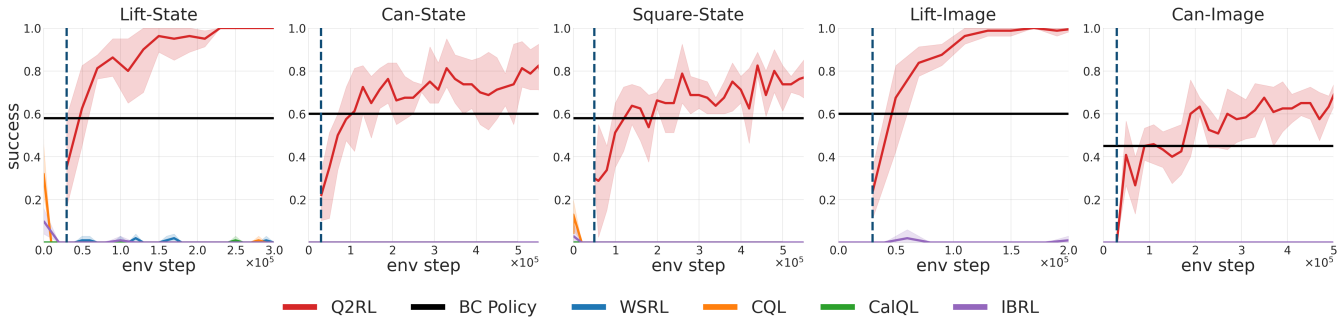
1) ***CQL:*** Conservative Q-Learning [4] is an offline RL method that trains exclusively on an offline dataset by prioritizing in-distribution actions while penalizing out-of-distribution actions.

2) ***CalQL:*** Calibrated Q-Learning [10] addresses the pessimism of Q-value estimates in CQL by calibrating the critic loss with respect to the value of a reference policy.

3) ***WSRL:*** Warm-Start RL [13] performs for offline-to-online RL without explicit data retention. 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.



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



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

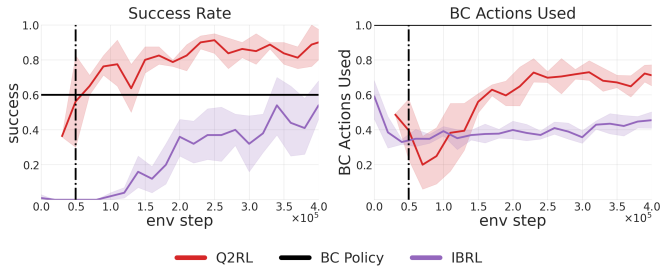
Fig. 5: Robomimic Results. The plot for each method begins at the start of online interaction. The dotted blue line signifies the end of the Q-Estimation phase and the start of online RL with Q-Gating for our method.

*4) **RLPD**:* RL with Prior Demonstrations [1] incorporates offline data directly into online learning by uniformly sampling from both offline and online replay buffers.

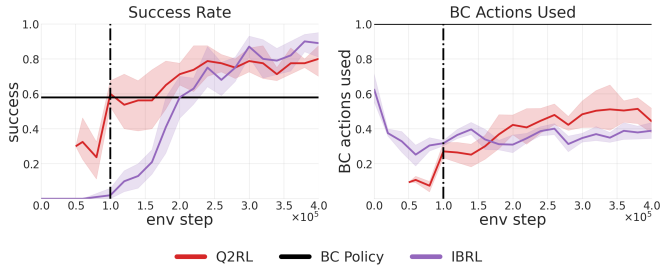
*5) **IBRL**:* Imitation Bootstrapped RL [3] uses a pre-trained BC policy to bootstrap Q value estimates and action proposals for RL. IBRL randomly initializes the Q function and uses demonstration data to seed the online replay buffer.

C. Results on D4RL

We present the results of D4RL tasks in Fig 4. For the **Kitchen** task, since the offline Kitchen-Complete dataset only contains successful demonstrations, representative of the current robot learning paradigm, the offline RL methods do not have enough data for a good pretraining. As offline RL pretraining fails, offline to online RL methods take longer time to achieve comparable performance. Whereas *Q2RL* is able to start with some initial performance provided by the BC pretraining, recover it further using our gating mechanism and continuously improve the performance with more environment interactions. 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, CalQL and IBRL are also able to reach optimal performance with online interaction, whereas CQL experiences a drop in performance, which we attribute to its conservative Q value estimates. For the **Door** task, we observe that WSRL is able to converge to a more optimal policy than the baselines while *Q2RL* is not able to converge to optimal performance. Though the Q2RL policy does not converge to optimal performance, from the rollout videos we can see that they are more true to the human demonstrations and therefore more realistic to be executed in the real world whereas the RL policy would be infeasible to be carried out in the real world.

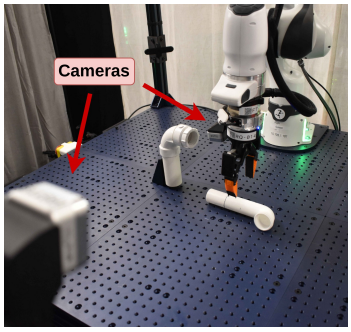


(a) Can-State

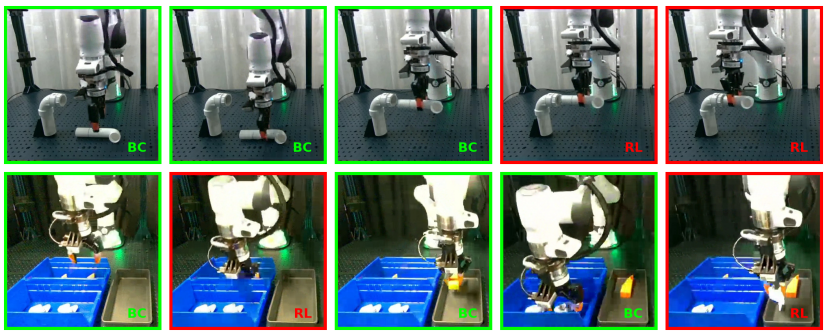


(b) Square-State

Fig. 6: 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.



(a) On-Robot RL Setup.



(b) Trained *Q2RL* policies for pipe assembly and kitting tasks. Green: BC actions, red: RL actions.

Fig. 7: Real World Experiments. (a) We use a Franka Panda arm with a Robotiq 2F-85 gripper, and RGB images from both wrist and workspace Intel RealSense D405s. (b) We analyze usage of BC vs. RL 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 actions for part grasping and placement.

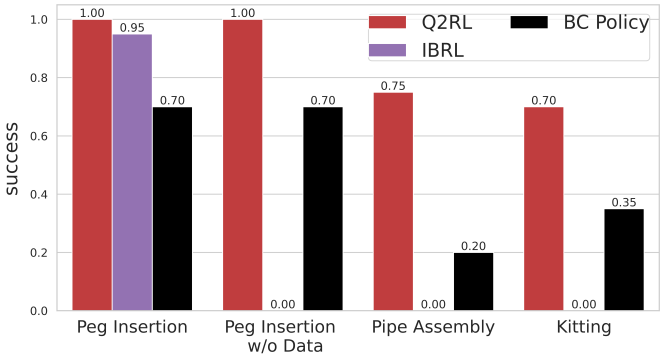


Fig. 8: On-Robot RL Results. Task success is reported over 20 trials per method. *Q2RL* significantly improves BC policy performance on every task. IBRL fails when (1) demonstrations are not seeded in the replay buffer (w/o Data), and (2) when tasks involve large action spaces and long horizon manipulation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

D. Robomimic Results

Next, we evaluate *Q2RL* on state-based and image-based robomimic tasks. We implement all methods 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 with access to training data during online learning, and another without access to the training data.

1) Access to BC Training Data: Results for the robomimic tasks with access to training data during online learning are shown in Fig. 5a. When training data is available in the online replay buffer, IBRL performs competitively and is able to converge to optimal performance. While IBRL is simple and effective, it does not explicitly 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 harder high-dimensional task, Can-Image, IBRL is unable to reach the base policy performance. Though, we note that in the IBRL paper, it is able to converge for Can-Image task, we hypothesize that this is due to the other optimization that IBRL uses for their RL policy like 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: Results for the robomimic tasks without access to training data during online learning are shown in Fig. 5b. 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-generated actions than to random RL actions. In the absence of a seeded replay buffer, IBRL has no mechanism to calibrate Q-values, leading to unreliable action selection. Our method is able to continually improve the performance even in the absence of training data. We also conduct a thorough evaluation for seeding the replay buffer with different amounts of data in Appendix E.

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. 6 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 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, the ratio of BC actions for our method increases, indicating that our Q-Gating algorithm enables higher success rates as Q_{RL} improves.

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. 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. 3):

Peg Insertion. This insertion task from FMB [8] has been used in recent on-robot RL methods [7, 13]. Success occurs when the peg is fully inserted into the board.

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 F.

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 Q_{BC}, and our proposed Q-Gating procedure uses both a frozen Q_{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 Q_{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. Appendix F contains additional details on results. **Videos of real world experiments available here**.

IV. 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 through Q-gating. In on-robot reinforcement learning experiments, *Q2RL* successfully improves performance on high precision, contact-rich manipulation tasks within a few hours of online interaction. See appendix for full related work, ablations and additional details here.

References

[1] 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.

[2] 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.

[3] Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning, 2023.

[4] 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.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.