Is Value Learning Really the Main Bottleneck in Offline RL?

Seohong Park[1] **Kevin Frans**[1] **Sergey Levine**[1] **Aviral Kumar**[2]
[1]University of California, Berkeley [2]Google DeepMind
seohong@berkeley.edu

Abstract

While imitation learning requires access to high-quality data, offline reinforcement learning (RL) should, in principle, perform similarly or better with substantially lower data quality. However, current results indicate that offline RL often performs worse than imitation learning, and it is often unclear what holds back the performance of offline RL. In this work, we aim to understand bottlenecks in current offline RL algorithms. While the worse performance of offline RL is typically attributed to an imperfect value function, we ask: *is the main bottleneck of offline RL indeed in learning the value function, the policy, or something else?* To answer this question, we perform a systematic empirical study of (1) value learning, (2) policy extraction, and (3) policy generalization in offline RL problems from the lens of "data-scaling" properties of each component, analyzing how these components affect performance. We make two surprising observations. First, the choice of a policy extraction algorithm affects the performance and scalability of offline RL significantly, often more so than its underlying value learning objective. For instance, widely used value-weighted regression objectives (*e.g.*, AWR) are not able to fully leverage the learned value function, and switching to behavior-regularized policy gradient objectives (*e.g.*, DDPG+BC) often leads to substantial improvements in performance and scaling behaviors. Second, the suboptimal performance of offline RL is often due to imperfect policy generalization on test-time states out of the support of the training data, rather than the policy accuracy on in-distribution states. While most current offline RL algorithms do not explicitly address this, we show that the use of suboptimal but high-coverage data or on-the-fly policy extraction techniques can be effective in addressing the policy generalization issue in practice.

1 Introduction

Data-driven approaches that convert offline datasets of past experience into policies are a predominant approach for solving control problems in several domains [9, 46, 48]. Primarily, there are two paradigms for learning policies from offline data: imitation learning and offline reinforcement learning (RL). While imitation requires access to high-quality demonstration data, offline RL loosens this requirement and can learn effective policies even from suboptimal data, which makes offline RL preferable to imitation learning in theory. However, recent results often show that tuning imitation learning by collecting more expert data often outperforms offline RL even when provided with sufficient data in practice [35, 45], and it is often unclear what holds back the performance of offline RL.

The primary difference between offline RL and imitation learning is the use of a *value function*, which is absent in imitation learning. The value function drives the learning progress of offline RL methods, enabling them to learn from suboptimal data. Value functions are typically trained via temporal-difference (TD) learning, which presents convergence [37, 52] and representational [26, 28, 53] pathologies. This has led to the conventional wisdom that the gap between offline RL and imitation is a direct consequence of poor value learning [25, 32, 35]. Following up on this conventional wisdom, much recent research in the community has been devoted towards improving the value function quality of offline RL algorithms [1, 11, 14, 18, 24, 25]. While improving value functions will definitely help improve performance, we question whether this is indeed the best way to maximally improve the performance of offline RL, or if there is still headroom to get offline RL to perform better even with current value learning techniques. More concretely, given an offline RL problem, we ask: *is*

Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2024).

the bottleneck in learning the value function, the policy, or something else? What is the best way to improve performance given the bottleneck?

We answer these questions via an empirical study. By construction, there are three potential factors that could bottleneck an offline RL algorithm: (**B1**) imperfect **value** function estimation, (**B2**) imperfect **policy** extraction guided by the learned value function, and (**B3**) imperfect policy **generalization** to states that it will visit during evaluation. While all of these contribute in some way to the performance of offline RL, we wish to identify how each of these factors interact in a given scenario and develop ways to improve them. To understand the effect of these factors, we use data size, quality, and coverage as levers for systematically controlling their impacts, and study the "data-scaling" properties, *i.e.*, how data quality, coverage, and quantity affect these three aspects of the offline RL algorithm, for three value learning methods and three policy extraction methods on diverse types of environments.

Through our analysis, we make two surprising observations, which naturally provide actionable advice for both domain-specific practitioners and future algorithm development in offline RL. **First,** even when value function learning is not perfect, we find that the choice of *policy extraction* algorithm often has a larger impact on performance, despite the policy being subordinate to the value function in theory. This is striking, given that policy extraction often tends to be an afterthought in the design of value-based offline RL algorithms. Specifically, we find that behavior-regularized policy gradient (*e.g.*, DDPG+BC [14]) almost always leads to much better performance and favorable data scaling than other widely used methods like value-weighted regression (*e.g.*, AWR [43, 44, 55]). **This means that** the policy extraction objective is often a significant bottleneck in offline RL, and with an appropriate choice of a policy extraction objective, we observe a favorable and consistent performance increase even for the same value function.

Second, we find that existing offline RL algorithms are often heavily bottlenecked by how well the policy *generalizes* on *test-time* states, rather than how optimal the policy is on the dataset state distribution. This provides a different perspective on offline RL, contrasting with the previous main focus on pessimism and behavioral regularization. However, no amount of policy training on dataset states or improvement to offline RL value learning objectives could address this in general, without additional assumptions. Nonetheless, we find that committing to using suboptimal but high-coverage data or continually adapting the policy within the course of a test-time rollout can improve this generalization bottleneck in practice. In particular, we develop two schemes for such test-time policy adaptation and find them to both be performant. **This means that** training value functions on *high-coverage* data can help improve the performance of offline RL methods at test time, and it can also be further improved if value functions are utilized in conjunction with the policy *during evaluation rollouts*.

Our main contributions are an analysis of the bottlenecks in offline RL as evaluated via data-scaling properties of various algorithmic choices. Contrary to the conventional belief that value learning is the bottleneck of offline RL algorithms, we find that the performance is often limited by the choice of policy extraction objective and the degree to which the policy generalizes at test time. This suggests that, with an appropriate policy extraction procedure (*i.e.*, regularized policy gradients and not value-weighted imitation learning) and an appropriate recipe for handling policy generalization (*e.g.*, test-time training with the value function), collecting more high-coverage data to train a value function is a universally better recipe to improve offline RL performance whenever the practitioner has access to collecting some new data for learning. These results also imply that more research should be done in developing policy learning recipes that can effectively translate improvements in value learning into performant offline RL policies.

2 Related work

Offline reinforcement learning [30, 32] aims to learn a policy solely from previously collected data. The central challenge of offline RL is to deal with the distributional shift in the state-action distributions of the dataset and the learned policy, which could lead to catastrophic value overestimation when not adequately addressed [32]. To prevent such failure, previous works in offline RL have proposed a number of techniques to estimate value functions solely from offline data, based on conservatism [8, 25], out-of-distribution penalization [14, 50, 56], in-sample maximization [16, 24, 58], uncertainty minimization [1, 18, 57], convex duality [31, 38, 47], or contrastive learning [11]. Then, these methods train policies to maximize the learned value function, which is typically done by behavior-regularized policy gradients (*e.g.*, DDPG+BC) [14, 33], weighted behavioral cloning (*e.g.*, AWR) [43, 44], or sampling-based action selection (*e.g.*, SfBC) [7, 15, 20]. Depending on the algorithm, these value learning and policy extraction stages can be either interleaved [14, 25, 39] or decoupled [5, 11, 16, 24]. While numerous methods have been proposed so far, relatively few works

have aimed to analyze and understand the practical challenges in offline RL. Instead of proposing a new algorithm, we mainly aim to understand the current bottlenecks in offline RL via a comprehensive analysis of existing techniques.

Several prior works have analyzed individual components of offline RL or imitation learning algorithms: value bootstrapping [14, 15], representation learning [26, 28, 59], data quality [4], differences between RL and behavioral cloning (BC) [27], and performance [10, 22, 34, 35, 51]. Our goal is distinct from this line of work: our goal is to analyze the bottlenecks in offline RL performance from a holistic perspective, comparing value function learning, policy extraction, and generalization. That is, our goal is not to diagnose pathologies with one of these components, but to understand how these components interact with each other, and how a practitioner could extract the most by improving one or more of them. Perhaps the closest study to ours is Fu et al. [13], which study whether representations, value accuracy, or policy accuracy can explain the performance of offline RL. They also find that combining IQL [24] with a TD3+BC-style policy extraction objective [14] improves performance. While this study makes insightful observations about the potential relationships between some metrics and performance, it is limited to D4RL locomotion tasks [12], and does not study *data-scaling* properties nor policy *generalization*, which we find to be one of the most substantial yet overlooked bottlenecks in offline RL. In contrast, we conduct a large-scale analysis on diverse environments (*e.g.*, pixel-based, goal-conditioned, manipulation) and analyze the bottlenecks in offline RL with the aim of providing actionable takeaways that can enhance the performance and scalability of offline RL.

3 Research hypothesis

Our primary goal is to understand when and how the performance of offline RL can be bottlenecked in practice. As discussed earlier, there exist three potential factors that could bottleneck an offline RL algorithm: (**B1**) imperfect **value** function estimation from data, (**B2**) imperfect **policy** extraction from the learned value function, and (**B3**) imperfect policy **generalization** on the test-time states that the policy visits at the evaluation time. We note that the bottleneck of an offline RL algorithm under a certain dataset can always be attributed to one or some of these factors, since the policy will attain optimal performance if both value learning and policy extraction are perfect, with perfect generalization to test-time states.

Our main research hypothesis in this work is that, somewhat contrary to the prior belief that the accuracy of the value function is the primary factor limiting performance of offline RL methods, **policy learning is often the main bottleneck of offline RL**. In other words, while value function accuracy is certainly important, how the policy is extracted from the value function (**B2**) and how well the policy generalizes on states that it visits at the deployment time (**B3**) are often the main factors that significantly affect both performance and scalability in many problems. To verify this hypothesis, we conduct two main analyses in this paper. In Section 4, we compare the effects of value learning and policy extraction on performance under various types of environments, datasets, and algorithms (**B1** and **B2**). In Section 5, we analyze the degree to which the policy generalizes on test-time states affects performance (**B3**).

4 Empirical analysis 1: Is it the value or the policy? (B1 and B2)

We first perform controlled experiments to identify whether imperfect value functions (**B1**) or imperfect policy extraction (**B2**) contribute more to holding back the performance of offline RL in practice. To systematically compare value learning and policy extraction, we run different algorithms while varying the *the amounts of data* for value function training and policy extraction, and draw **data-scaling matrices** to visualize the aggregate results. Increasing the amount of data provides a convenient lever to control the effect of each component, enabling us to draw conclusions about whether the value or the policy serves as a bigger bottleneck in different regimes when different amounts of training data are available, and to understand the differences between various algorithms.

To clearly dissect value learning from policy learning, in this section, we focus on offline RL methods with decoupled value and policy training phases (*e.g.*, One-step RL [5], IQL [24], CRL [11], etc.), where policy learning does not affect value learning, *i.e.*, methods that first train a value function without involving policies, and then extract a policy from the learned value function with a separate objective. While this might sound a bit restrictive, we surprisingly find that policy learning is often the main bottleneck *even in these decoupled methods*, which attempt to solve a simple, single-step optimization problem for extracting a policy given a static and stationary value function.

4.1 Analysis setup

We now introduce the value learning objectives, policy extraction objectives, and environments that we study in our analysis (see Appendix B for preliminaries).

Value learning objectives. We consider three decoupled value learning objectives that fit value functions without involving policy learning: (**1**) **implicit Q-learning** (**IQL**) [24], (**2**) **SARSA** [5], and (**3**) **contrastive RL** (**CRL**) [11]. IQL fits an optimal Q function (Q^*) by approximating the Bellman optimality operator with an expectile loss. SARSA fits a behavioral Q function (Q^β) using the Bellman evaluation operator. In goal-conditioned tasks, we employ CRL instead of SARSA, which similarly fits a behavioral Q function, but with a different contrastive learning-based objective that leads to better performance. We refer to Appendix D.1 for detailed descriptions of these value learning methods.

Policy extraction objectives. Prior works in offline RL typically use one of the following objectives to extract a policy from the value function. All of them are built upon the same principle: maximizing values while being close to the behavioral policy, to avoid the exploitation of erroneous critic values.

- (**1**) **Weighted behavioral cloning** (*e.g.*, **AWR**). Weighted behavioral cloning is one of the most widely used offline policy extraction objectives for its simplicity [24, 39, 41, 43, 44, 55]. Among weighted behavioral cloning methods, we consider advantage-weighted regression (AWR [43, 44]) in this work, which maximizes the following objective:

$$\max_{\pi} \mathcal{J}_{\mathrm{AWR}}(\pi) = \mathbb{E}_{s,a\sim\mathcal{D}}[e^{\alpha(Q(s,a)-V(s))} \log \pi(a \mid s)], \tag{1}$$

  where α is an (inverse) temperature hyperparameter. Intuitively, AWR assigns larger weights to higher-advantage transitions when cloning behaviors, which makes the policy selectively copy only good actions from the dataset.

- (**2**) **Behavior-constrained policy gradient** (*e.g.*, **DDPG+BC**). Another popular policy extraction objective is behavior-constrained policy gradient, which directly maximizes Q values while not deviating far away from the behavioral policy [1, 14, 18, 25, 56]. In this work, we consider the objective that combines deep deterministic policy gradients and behavioral cloning (DDPG+BC [14]):

$$\max_{\pi} \mathcal{J}_{\mathrm{DDPG}}(\pi) = \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s, \mu^\pi(s)) + \alpha \log \pi(a \mid s)], \tag{2}$$

  where $\mu^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[a]$ and α is a hyperparameter that controls the strength of the BC regularizer. This objective is equivalent to Q maximization regularized by the forward KL divergence.

- (**3**) **Sampling-based action selection** (*e.g.*, **SfBC**). Instead of learning an explicit policy, some previous methods implicitly define a policy as the action with the highest value among action samples from the behavioral policy [7, 15, 17, 20]. In this work, we consider the following objective that selects the $\arg\max$ action from behavioral candidates (SfBC [7]):

$$\pi(\cdot \mid s) = \operatorname*{arg\,max}_{a\in\{a_1,\ldots,a_N\}} [Q(s, a)], \tag{3}$$

  where $a_1, \ldots, a_N$ are sampled from the learned BC policy $\pi^\beta(a \mid s)$ [7, 20].

Environments and datasets. To understand how different value learning and policy extraction objectives affect performance and data scalability, we consider eight environments (Figure 9) across state- and pixel-based, robotic locomotion and manipulation, and goal-conditioned and single-task settings with varying levels of data suboptimality: (**1**) `gc-antmaze-large`, (**2**) `antmaze-large`, (**3**) `d4rl-hopper`, (**4**) `d4rl-walker2d`, (**5**) `exorl-walker`, (**6**) `exorl-cheetah`, (**7**) `kitchen`, and (**8**) `gc-roboverse`. We highlight some features of these tasks: `exorl-{walker, cheetah}` are tasks with highly suboptimal, diverse datasets collected by exploratory policies, `gc-antmaze-large` and `gc-roboverse` are goal-conditioned ('`gc-`') tasks, and `gc-roboverse` is a *pixel-based* robotic manipulation task with a $48 \times 48 \times 3$-dimensional observation space. For some tasks (*e.g.*, `gc-antmaze-large` and `kitchen`), we additionally collect data to enhance dataset sizes to depict scaling properties clearly. We refer to Appendix D.2 for the complete task descriptions.

4.2 Results

Figure 1 shows the data-scaling matrices of three policy extraction algorithms (AWR, DDPG+BC, and SfBC) and three value learning algorithms (IQL and {SARSA or CRL}) on eight environments, aggregated from a total of 7744 runs (4 seeds for each cell). In each matrix, we individually tune the hyperparameter for policy extraction (α or N) for each entry. These matrices show how performance varies with different amounts of data for the value and the policy. In our analysis, we specifically

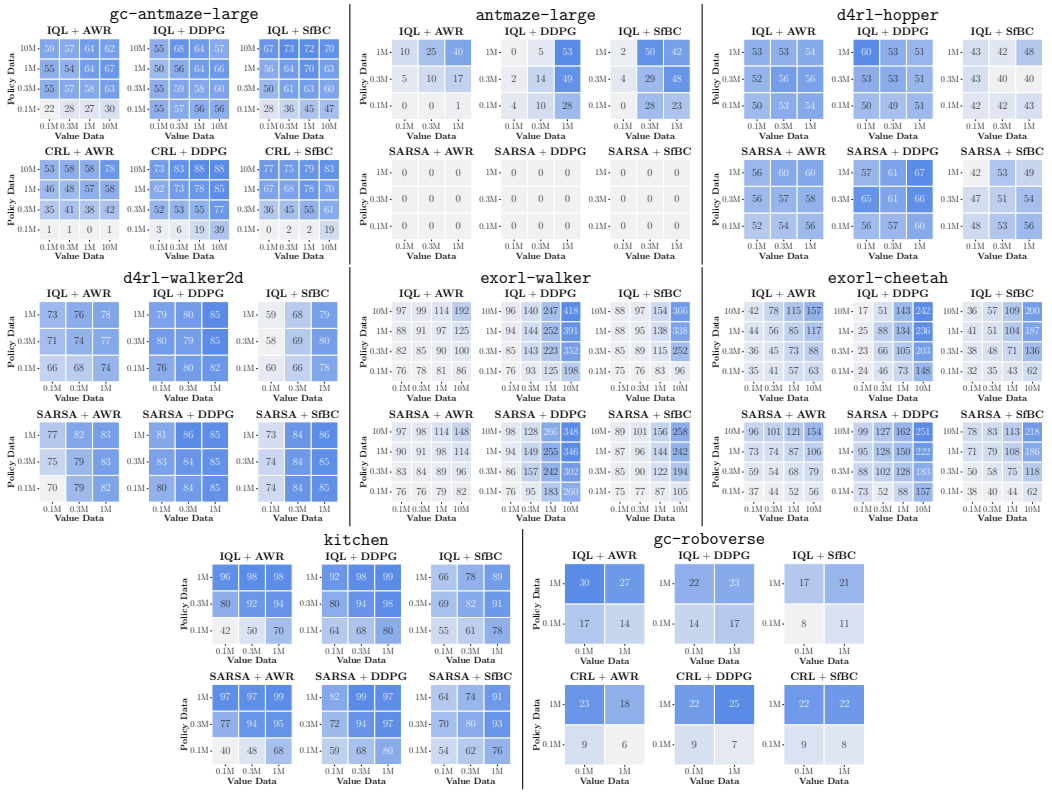


Figure 1: **Data-scaling matrices of three policy extraction methods (AWR, DDPG+BC, and SfBC) and three value learning methods (IQL and {SARSA or CRL}).** To see whether the value or the policy imposes a bigger bottleneck, we measure performance with varying amounts of data for the value and the policy. The color gradients (⇑, ↗, ⇒) of these matrices reveal how the performance of offline RL is bottlenecked in each setting.

focus on the *color gradients* of these matrices, which reveal how the performance of offline RL is bottlenecked in each setting. Note that the color gradients are mostly either vertical, horizontal, or diagonal. Vertical (⇑) color gradients (*e.g.*, IQL+AWR on `gc-antmaze-large`) indicate that the performance is most strongly affected by the amount of *policy* data, horizontal (⇒) gradients (*e.g.*, IQL+SfBC on `d4rl-walker2d`) indicate it is mostly affected by *value* data, and diagonal (↗) gradients (*e.g.*, IQL+DDPG+BC on `exorl-walker`) indicate both.

Side-by-side comparisons of the data-scaling matrices from different policy extraction methods in Figure 1 suggest that, perhaps surprisingly, **different policy extraction algorithms often lead to significantly different performance and data-scaling behaviors, even though they extract policies from the *same* value functions** (recall that the use of decoupled algorithms allows us to train a single value function, but use it for policy extraction in different ways). For example, on `exorl-walker` and `exorl-cheetah`, AWR performs remarkably poorly compared to DDPG+BC or SfBC on both value learning algorithms. Such a performance gap between policy extraction algorithms exists even when the value function is far from perfect, as can be seen in the low-data regimes in `gc-antmaze-large` and `kitchen`. In general, we find that the choice of policy extraction procedure affects performance often more than the choice of value learning objective except `antmaze-large`, where the value function must be learned from sparse-reward, suboptimal datasets with long-horizon trajectories.

Among policy extraction algorithms, we find that **DDPG+BC almost always achieves the best performance and scaling behaviors across the board**, followed by SfBC, and the performance of AWR falls significantly behind the other two extraction algorithms in many cases. Notably, the data-scaling matrices of AWR always have vertical (⇑) or diagonal (↗) color gradients, implicitly implying that it does not fully utilize the value function (see Section 4.3 for clearer evidence). In other words, a non-careful choice of the policy extraction algorithm (*e.g.*, weighted behavioral cloning) hinders the use of learned value functions, imposing an unnecessary bottleneck on the performance of offline RL.

4.3 Deep dive 1: How different are the scaling properties of AWR and DDPG+BC?

To gain further insights into the difference between value-weighted behavioral cloning (*e.g.*, AWR) and behavior-regularized policy gradients (*e.g.*, DDPG+BC), we draw data-scaling matrices with

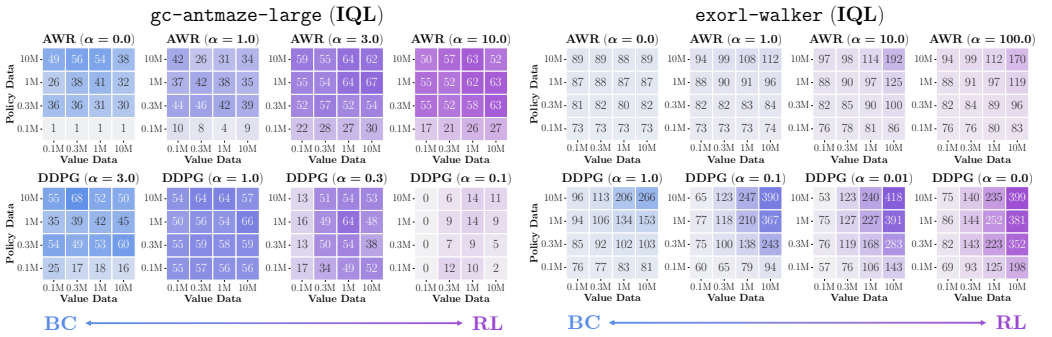


Figure 2: **Data-scaling matrices of AWR and DDPG+BC with different BC strengths (α).** In `gc-antmaze-large`, AWR is *always* policy-bounded (⇑), but DDPG+BC has *both* policy-bounded (⇑) and value-bounded (⇒) modes, depending on the value of α. Notably, an in-between value ($\alpha = 1.0$) of DDPG+BC leads to the best of both worlds (see the bottom left corner of `gc-antmaze-large` with 0.1M datasets)!

different values of α (in Equations (1) and (2)), a hyperparameter that interpolates between RL and BC. Note that $\alpha = 0$ corresponds to BC in AWR and $\alpha = \infty$ corresponds to BC in DDPG+BC. We recall that the previous results (Figure 1) use the best temperature for each matrix entry (*i.e.*, aggregated by the maximum over temperatures), but here we show the full results with individual hyperparameters.

Figure 2 highlights the results on `gc-antmaze-large` and `exorl-walker` (see Appendix E for the full results). The results on `gc-antmaze-large` show a clear difference in scaling matrices between AWR and DDPG+BC. That is, AWR is *always* policy-bounded regardless of the BC strength α (*i.e.*, vertical (⇑) color gradients), whereas DDPG+BC has two "modes": it is policy-bounded (⇑) when α is large, and value-bounded (⇒) and when α is small. Intriguingly, an in-between value of $\alpha = 1.0$ in DDPG+BC enables having the best of both worlds, significantly boosting performances across the entire matrix (note that it achieves very strong performance even with a 0.1M-sized dataset)! This difference in scaling behaviors suggests that the use of the learned value function in weighted behavioral cloning is limited. This becomes more evident in `exorl-walker` (Figure 2), where AWR fails to achieve strong performance even with a very high temperature value ($\alpha = 100$).

4.4 Deep dive 2: *Why* is DDPG+BC better than AWR?

We have so far seen several empirical results that suggest DDPG+BC should be preferred to AWR in any cases. What makes DDPG+BC so much better than AWR? There are three potential reasons.

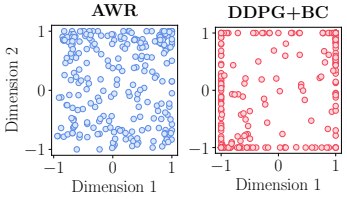


Figure 3: **AWR vs. DDPG actions.**

First, AWR only has a *mode-covering* weighted behavioral cloning term, while DDPG+BC has *both mode-seeking* first-order value maximization and *mode-covering* behavioral cloning terms. As a result, actions learned by AWR always lie within the convex hull of dataset actions, whereas DDPG+BC can "hillclimb" the learned value function, even allowing extrapolation to some degree while not deviating too far away from the mode. This not only enables a better use of the value function but yields potentially more optimal actions. To illustrate this, we plot test-time action sampled from policies learned by AWR and DDPG+BC on `exorl-walker`. Figure 3 shows that AWR actions are relatively centered around the origin, while DDPG+BC actions are more spread out and thus potentially have high optimality.

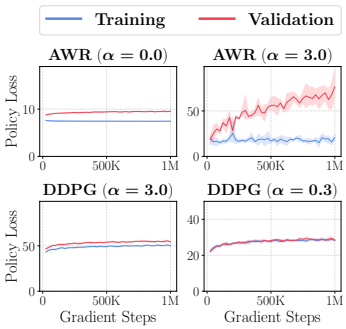


Figure 4: **AWR overfits.**

Second, value-weighted behavioral cloning uses a much smaller number of *effective* samples than behavior-regularized policy gradient methods, especially when the temperature (α) is large. This is because a small number of high-advantage transitions can potentially dominate the learning signals of AWR (*e.g.*, a single transition with a weight of e^{10} can dominate other transitions with smaller weights like e^2). As a result, AWR effectively uses only a fraction of datapoints for policy learning, being susceptible to overfitting. On the other hand, DDPG+BC is based on first-order maximization of the value function without any weighting, and thus is free from this issue. Figure 4 illustrates this, where we compare

the training and validation policy losses of AWR and DDPG+BC on `gc-antmaze-large` with the smallest 0.1M dataset (8 seeds). The results show that AWR with a large temperature ($\alpha = 3.0$) causes severe overfitting. Indeed, Figure 1 shows DDPG+BC often achieves significantly better performance than AWR in low-data regimes.

Third, AWR has a theoretical pathology in the regime with limited samples: since the coefficient in front of $\log \pi(a \mid s)$ in the AWR objective (Equation (1)) is always positive, AWR can increase the likelihood of *all* dataset actions, regardless of their optimality. If the training dataset has covered all possible actions, then the condition for normalization of the probability density function of $\pi(a \mid s)$ would have alleviated this concern, but this condition is rarely achieved in practice. Under limited data coverage, and especially when the policy network is highly expressive and dataset states are unique (*e.g.*, continuous control problems), AWR can in theory *memorize* all state-action pairs in the dataset, potentially reverting to *unweighted* behavioral cloning.

Takeaway: Policy extraction can inhibit the complete use of the value function

Do *not* use value-weighted behavior cloning (*e.g.*, AWR); always use behavior-constrained policy gradient (*e.g.*, DDPG+BC), regardless of the value learning objective. This enables better scaling of performance with more data and better use of the value function.

5 Empirical analysis 2: Policy generalization (B3)

We now turn our focus to the third hypothesis, that policy **generalization** to states that the policy visits at the evaluation time has a significant impact on performance. This is a unique bottleneck to the *offline* RL problem setting, where the agent encounters new, potentially out-of-distribution states at test time. To measure policy accuracy, we first define three key metrics quantifying a notion of *accuracy* of the policy in terms of the mean squared error (MSE) against the optimal policy:

$$(\text{Training MSE}) = \mathbb{E}_{s \sim \mathcal{D}_{\text{train}}}[(\pi(s) - \pi^*(s))^2], \tag{4}$$

$$(\text{Validation MSE}) = \mathbb{E}_{s \sim \mathcal{D}_{\text{val}}}\ [(\pi(s) - \pi^*(s))^2], \tag{5}$$

$$(\text{Evaluation MSE}) = \mathbb{E}_{s \sim p^{\pi}(s)}\ [(\pi(s) - \pi^*(s))^2], \tag{6}$$

where $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ respectively denote the training and validation datasets, π^* denotes an optimal policy, that we assume access to for evaluation and visualization purposes only. We assume that the policies $\pi, \pi^* : \mathcal{S} \to \mathcal{A}$ are deterministic for simplicity. Validation MSE measures the policy accuracy on states sampled from the *same* dataset distribution as the training distribution (*i.e.*, in-distribution MSE), while evaluation MSE measures the policy accuracy on states the agent visits at test time, which can potentially be very different from the dataset distribution (*i.e.*, out-of-distribution MSE). We note that, while these metrics might not always be perfectly indicative of policy accuracy (see Appendix A for limitations), they often serve as convenient proxies to estimate policy accuracy in many continuous-control domains in practice.

One way to measure the degree to which test-time policy generalization affects performance is to see how various policy MSE metrics evolve and correlate with performance after further training the agent on data sampled from the test-time distribution, which serves as one of the ideal distributions to improve performance. Hence, we measure the three types of MSEs in the *offline-to-online* RL setting, in which we observe how these MSEs improve over time with additional online interaction data. Specifically, we train offline-to-online IQL agents on six D4RL [12] tasks (`antmaze-{medium, large}`, `kitchen`, and `adroit-{pen, hammer, door}`), and measure the MSEs with pre-trained expert policies that approximate π^* (see Appendix D.4).

Results. Figure 5 shows the results (8 seeds with 95% confidence intervals), where we denote online training steps in red. The results show that, perhaps surprisingly, in many environments offline-to-online RL *only* improves evaluation MSEs, not training MSEs nor validation MSEs, and the performance of offline RL is most strongly (inversely) correlated with the evaluation MSE among the three metrics. What does this tell us? In a sense, online interaction data presents an "oracle" data distribution that should improve policy accuracy across the state space, at least locally around the states that the policy visits and are important for the task. However, in many environments, we see such policy improvement is *only* happening in the policy's own distribution (*i.e.*, evaluation MSE), while the other two dataset MSEs often remain completely flat. Of course, since we further train the policy on its own interaction data, the evaluation MSE naturally gets more improvements

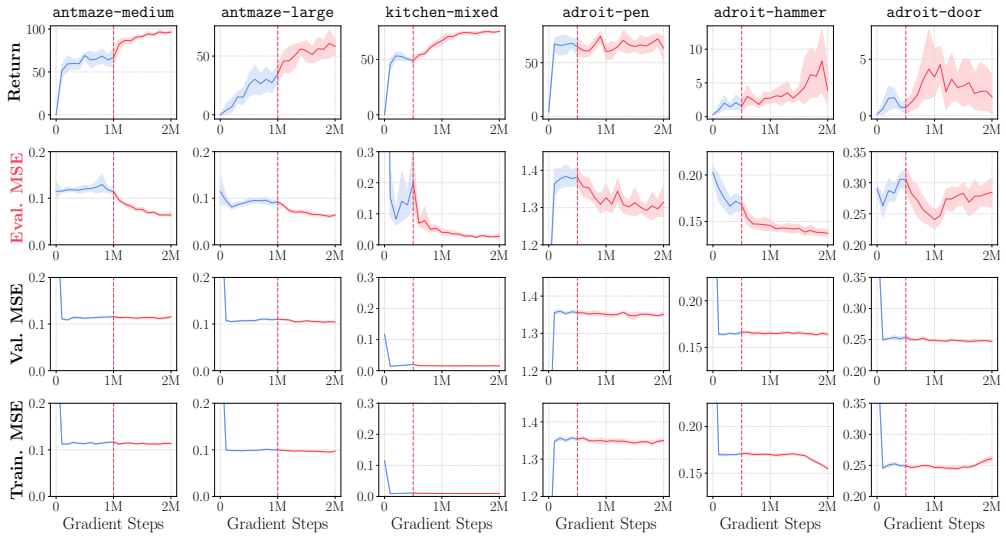


Figure 5: **How do offline RL policies get improved with additional interaction data?** In many environments, offline-to-online RL *only* improves evaluation MSEs, while validation MSEs and training MSEs often *remain completely flat* (see Section 5 for the definitions of these metrics). This suggests that current offline RL algorithms may already be great at learning an effective policy on *in-distribution* states, and the performance of offline RL is often determined by how well the policy *generalizes* on its own state distribution at test time.

than the other two metrics, but it is remarkable that (1) the dataset MSEs completely flatline in many environments and (2) the performance is very strongly correlated with the evaluation MSE. This indicates that, current offline RL methods may already be great at learning the best possible policy *within the distribution of states on the dataset*, and **the agent's performance is often mainly bottlenecked by how well it generalizes under its own state distribution at test time**. This finding somewhat contradicts prior beliefs: while algorithmic techniques in offline RL largely hillclimb on improving policy optimality on *in-distribution states* (by addressing the issue with out-of-distribution *actions*), our results suggest that modern offline RL algorithms may already saturate on this axis. Further performance differences may simply be due to the effects of a given offline RL objective on *novel states*, which very few methods explicitly control.

That said, controlling test-time generalization might also appear impossible: while offline RL methods could hillclimb on validation accuracy via a combination of techniques that address statistical errors such as regularization (*e.g.*, Dropout [49], LayerNorm [3], etc.), improving *test-time* policy accuracy requires generalization to a potentially very different *distribution*, which is theoretically impossible to guarantee without additional coverage or structural assumptions, as the test-time state distribution can be arbitrarily adversarial in the worst case. However, if we actively utilize the information available at test time or have the freedom to design offline datasets, it is possible to improve test-time policy accuracy in practice, and we discuss such solutions below (see Appendix C for further discussions).

Improve offline data *coverage*. If we have the freedom to control the data collection process, perhaps the most straightforward way to improve test-time policy accuracy is to use a dataset that has as *high coverage* as possible so that test-time states can be covered by the dataset distribution. However, at the same time, high-coverage datasets often involve exploratory actions, which may compromise the quality (optimality) of the dataset. This makes us wonder in practice: *which is more important, high coverage or high optimality?*

To answer this question, we empirically compare the data-scaling matrices on datasets collected by expert policies with different levels of action noises (σ_{data}). Figure 6 shows the results of IQL agents on `gc-antmaze-large` and `adroit-pen` (4 seeds each). The results suggest that the performance of offline RL generally improves as the dataset has better state coverage, despite the increased suboptimality. This is aligned with our findings in Figure 5, which indicate that the main challenge of offline RL is often *not* on learning an effective policy from suboptimal data, but rather learning a policy that generalizes well at test-time states. Also, the low-data regimes in `gc-antmaze-large` further support the claim made in Section 4, which says weighted behavioral cloning (*e.g.*, AWR) inhibits the complete use of the value function. In summary, our results suggest practitioners prioritize *high coverage* (particularly around the states that the optimal policy will likely visit) over high optimally when collecting datasets for offline RL.

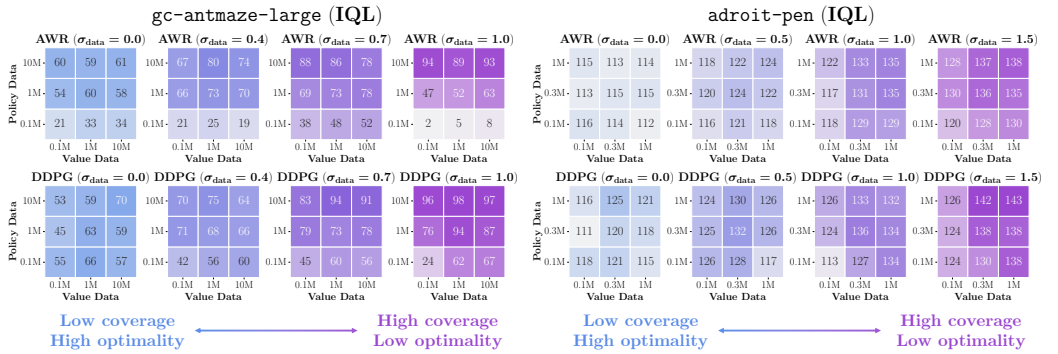


Figure 6: **Should we use high-coverage or high-optimality datasets?** The data-scaling matrices above show that *high-coverage* datasets can be much more effective than high-optimality datasets. This is because high-coverage datasets can improve *test-time policy accuracy*, one of the main bottlenecks of offline RL.

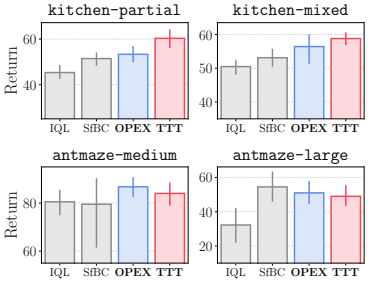


Figure 7: **OPEX and TTT.**

***Test-time* policy improvement.** If we do not have control over offline data collection, another way to improve test-time policy accuracy is to *on-the-fly* train or steer the policy guided by the learned value function on *test-time states*. Especially given that imperfect policy extraction from the value function is often a significant bottleneck in offline RL (Section 4), we propose further distilling the information in the value function into the policy by adjusting policy actions in the value gradient direction *at test time*, *i.e.*, $a \leftarrow a + \beta \cdot \nabla_a Q(s, a)$, where β is the test-time "learning rate". This way, we can further adjust policy actions on unseen states to maximize values, while not too much deviating from the learned policy. We call this **on-the-fly policy extraction** (**OPEX**). Note that OPEX requires only *a single line of additional code* at evaluation and does not change the training procedure at all. In our experiments, we also consider another variant that further updates the parameters of the policy, in particular, by continuously extracting the policy from the fixed value function on test-time states, as more rollouts are performed. We call this **test-time training** (**TTT**). We refer to Appendix D.5 for the implementation details of these test-time improvement schemes. Figure 7 compares the performances of vanilla IQL, SfBC (Equation (3), another test-time policy extraction method that does not involve gradients), and our test-time policy improvement strategies on four tasks (8 seeds each), showing that our gradient-based test-time strategies improve performance over vanilla IQL in many tasks.

Takeaway: Improving test-time policy accuracy significantly boosts performance

Test-time policy *generalization* is one of the most significant bottlenecks of offline RL. Use high-coverage datasets. Improve policy accuracy on test-time states with on-the-fly policy improvement techniques.

6 Conclusion: What does our analysis tell us?

In this work, we empirically demonstrated that, contrary to the prior belief that improving the quality of the value function is the primary bottleneck of offline RL, current offline RL methods are often heavily limited by how faithfully the policy is *extracted* from the value function and how well this policy *generalizes* on test-time states. **For practitioners,** our analysis suggests a clear empirical recipe for effective offline RL: train a value function on as *diverse* data as possible, and allow the policy to maximally utilize the value function, with the best policy extraction objective (*e.g.*, DDPG+BC) and/or potential test-time policy improvement strategies, as discussed in this paper. **For future algorithms research,** our analysis emphasizes two important open questions in offline RL: (1) What is the best way to *extract* a policy from the learned value function? (2) How can we train a policy in a way that it *generalizes* well on test-time states? The second question is particularly notable, because it suggests a diametrically opposed viewpoint to the prevailing theme of pessimism in offline RL, where only a few works have explicitly aimed to address this generalization aspect of offline RL. We believe finding effective answers to these questions would lead to significant performance gains in offline RL, substantially enhancing its applicability and scalability.

References

[1] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Neural Information Processing Systems (NeurIPS)*, 2017.

[3] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[4] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[5] David Brandfonbrener, William F. Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[6] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*, 2019.

[7] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[8] Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2022.

[9] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li,

Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and rt-x models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[10] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? In *International Conference on Learning Representations (ICLR)*, 2022.

[11] Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2022.

[12] Justin Fu, Aviral Kumar, Ofir Nachum, G. Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *ArXiv*, abs/2004.07219, 2020.

[13] Yuwei Fu, Di Wu, and Benoît Boulet. A closer look at offline rl agents. In *Neural Information Processing Systems (NeurIPS)*, 2022.

[14] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[15] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, 2019.

[16] Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. In *International Conference on Learning Representations (ICLR)*, 2023.

[17] Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning (ICML)*, 2021.

[18] Seyed Kamyar Seyed Ghasemipour, Shixiang Shane Gu, and Ofir Nachum. Why so pessimistic? estimating uncertainties for offline rl through ensembles, and why their independence matters. In *Neural Information Processing Systems (NeurIPS)*, 2022.

[19] Dibya Ghosh. dibyaghosh/jaxrl_m, 2023. URL https://github.com/dibyaghosh/jaxrl_m.

[20] 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. *ArXiv*, abs/2304.10573, 2023.

[21] Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1993.

[22] Bingyi Kang, Xiao Ma, Yi-Ren Wang, Yang Yue, and Shuicheng Yan. Improving and benchmarking offline reinforcement learning algorithms. *ArXiv*, abs/2306.00972, 2023.

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

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

[25] Aviral Kumar, Aurick Zhou, G. Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[26] Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[27] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should i run offline reinforcement learning or behavioral cloning? In *International Conference on Learning Representations (ICLR)*, 2021.

[28] Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron C. Courville, G. Tucker, and Sergey Levine. Dr3: Value-based deep reinforcement learning requires explicit regularization. In *International Conference on Learning Representations (ICLR)*, 2022.

[29] Cassidy Laidlaw, Stuart J. Russell, and Anca D. Dragan. Bridging rl theory and practice with the effective horizon. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[30] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pages 45–73. Springer, 2012.

[31] Jongmin Lee, Wonseok Jeon, Byung-Jun Lee, Joëlle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning (ICML)*, 2021.

[32] Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

[33] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[34] Cong Lu, Philip J. Ball, Tim G. J. Rudner, Jack Parker-Holder, Michael A. Osborne, and Yee Whye Teh. Challenges and opportunities in offline reinforcement learning from visual observations. *Transactions on Machine Learning Research (TMLR)*, 2023.

[35] Ajay Mandlekar, Danfei Xu, J. Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Mart'in-Mart'in. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

[37] Rémi Munos. Error bounds for approximate policy iteration. In *International Conference on Machine Learning (ICML)*, 2003.

[38] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *ArXiv*, abs/1912.02074, 2019.

[39] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *ArXiv*, abs/2006.09359, 2020.

[40] Whitney Newey and James L. Powell. Asymmetric least squares estimation and testing. *Econometrica*, 55: 819–847, 1987.

[41] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[42] Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *International Conference on Machine Learning (ICML)*, 2024.

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

[44] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *International Conference on Machine Learning (ICML)*, 2007.

[45] Rafael Rafailov, Kyle Beltran Hatch, Anikait Singh, Aviral Kumar, Laura Smith, Ilya Kostrikov, Philippe Hansen-Estruch, Victor Kolev, Philip J Ball, Jiajun Wu, et al. D5rl: Diverse datasets for data-driven deep reinforcement learning. In *Reinforcement Learning Conference (RLC)*, 2024.

[46] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions on Machine Learning Research (TMLR)*, 2022.

[47] Harshit S. Sikchi, Qinqing Zheng, Amy Zhang, and Scott Niekum. Dual rl: Unification and new methods for reinforcement and imitation learning. In *International Conference on Learning Representations (ICLR)*, 2024.

[48] Jost Tobias Springenberg, Abbas Abdolmaleki, Jingwei Zhang, Oliver Groth, Michael Bloesch, Thomas Lampe, Philemon Brakel, Sarah Bechtle, Steven Kapturowski, Roland Hafner, Nicolas Manfred Otto Heess, and Martin A. Riedmiller. Offline actor-critic reinforcement learning scales to large models. In *International Conference on Machine Learning (ICML)*, 2024.

[49] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014.

[50] Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[51] Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. Corl: Research-oriented deep offline reinforcement learning library. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[52] Ruosong Wang, Dean Phillips Foster, and Sham M. Kakade. What are the statistical limits of offline rl with linear function approximation? In *International Conference on Learning Representations (ICLR)*, 2021.

[53] Ruosong Wang, Yifan Wu, Ruslan Salakhutdinov, and Sham M. Kakade. Instabilities of offline rl with pre-trained neural representation. In *International Conference on Machine Learning (ICML)*, 2021.

[54] Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning (ICML)*, 2023.

[55] Ziyun Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott E. Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Manfred Otto Heess, and Nando de Freitas. Critic regularized regression. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[56] Yifan Wu, G. Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *ArXiv*, abs/1911.11361, 2019.

[57] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua M. Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2021.

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

[59] Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. In *International Conference on Machine Learning (ICML)*, 2021.

[60] Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, P. Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *ArXiv*, abs/2201.13425, 2022.

[61] Chongyi Zheng, Benjamin Eysenbach, Homer Walke, Patrick Yin, Kuan Fang, Ruslan Salakhutdinov, and Sergey Levine. Stabilizing contrastive rl: Techniques for offline goal reaching. *ArXiv*, abs/2306.03346, 2023.

Appendices

A Limitations

One limitation of our analysis is that the MSE metrics in Equations (4) to (6) are in some sense "proxies" to measure the accuracy of the policy. For instance, if there exist multiple optimal actions that are potentially very different from one another, or the expert policy used in practice is not sufficiently optimal, the MSE metrics might not be highly indicative of the performance or accuracy of the policy. Nonetheless, we empirically find that there is a strong correlation between the evaluation MSE metric and performance, and we believe our analysis could further be refined with potentially more sophisticated metrics (*e.g.*, by considering $\mathbb{E}[Q^*(s, a)]$ instead of $\mathbb{E}[(\pi(s) - \pi^*(s))^2]$), which we leave for future work.

B Preliminaries

We consider a Markov decision process (MDP) defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \mu, p)$. $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function, $\mu \in \Delta(\mathcal{S})$ denotes the initial state distribution, and $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition dynamics, where $\Delta(\mathcal{X})$ denotes the set of probability distributions over a set $\mathcal{X}$. We consider the offline RL problem, whose goal is to find a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the discount return $J(\pi) = \mathbb{E}_{\tau \sim p^\pi(\tau)}[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)]$, where $p^\pi(\tau) = p^\pi(s_0, a_0, s_1, a_1, \ldots, s_T) = \mu(s_0)\pi(a_0 \mid s_0)p(s_1 \mid s_0, a_0) \cdots \pi(a_T \mid s_T)$ and γ is a discount factor, solely from a static dataset $\mathcal{D} = \{\tau_i\}_{i \in \{1,2,\ldots,N\}}$ without online interactions. In some of our experiments, we consider offline *goal-conditioned* RL [2, 11, 21, 41, 54] as well, where the policy and reward function are also conditioned on a goal state g, which is sampled from a goal distribution $p_g \in \Delta\mathcal{S}$. For goal-conditioned RL, we assume a sparse goal-conditioned reward function, $r(s, g) = \mathbb{1}(s = g)$, which does not require any prior knowledge about the state space, and we assume that the episode ends upon goal-reaching [41, 42, 54].

C Policy generalization: Rethinking the role of state representations

In this section, we introduce another way to improve test-time policy accuracy from the perspective of *state representations*. Specifically, we claim that we can improve test-time policy accuracy by using a "good" representation that *naturally* enables out-of-distribution generalization. Since this might sound a bit cryptic, we first show results to illustrate this point.

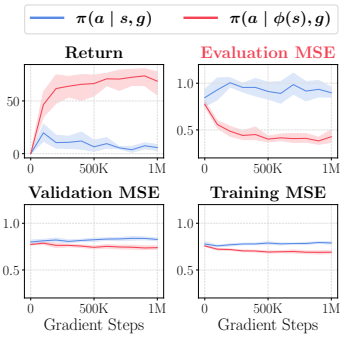


Figure 8: **A good state representation naturally enables test-time generalization, leading to substantially better performance.**

Figure 8 shows the performances of goal-conditioned BC[1] on `gc-antmaze-large` with two different *homeomorphic* representations: one with the original state representation s, and one with a different representation $\phi(s)$ with a continuous, *invertible* ϕ (specifically, ϕ transforms x-y coordinates with invertible $\tanh$ kernels; see Appendix D.6). Hence, these two representations contain the exactly same amount of information and are even topologically homeomorphic (under the standard Euclidean topology). However, they result in *very* different performances, and the MSE plots in Figure 8 indicate that this difference is due to nothing other than the better test-time, *evaluation* MSE (observe that their training and validation MSEs are nearly identical)!

This result sheds light on an important perspective of state representations: a good state representation should be able to enable *test-time generalization naturally*. While designing such a good state representation might require some knowledge or inductive biases about the task, our results suggest that using such a representation is nonetheless very important in practice, since it affects the performance of offline RL significantly by improving test-time policy generalization capability.

[1] Here, we use BC (not RL) to focus solely on state representations, obviating potential confounding factors regarding the value function.

D Experimental details

We provide the full experimental details in this section.

D.1 Value learning objectives

One-step RL (SARSA). SARSA [5] is one of the simplest offline value learning algorithms. Instead of fitting a Bellman optimal value function Q^*, SARSA aims to fit a behavioral value function Q^β with TD-learning, without querying out-of-distribution actions. Concretely, SARSA optimizes

$$\min_Q \mathcal{L}_{\text{SARSA}}(Q) = \mathbb{E}_{(s,a,s',a')\sim\mathcal{D}}[(r(s,a) + \gamma\bar{Q}(s',a') - Q(s,a))^2], \tag{7}$$

where s' and a' denote the next state and action, respectively, and $\bar{Q}$ denotes the target Q network [36]. Despite its apparent simplicity, extracting a policy by maximizing the value function learned by SARSA is often a surprisingly strong baseline [5, 29].

Implicit Q-learning (IQL). Implicit Q-learning (IQL) [24] aims to fit a Bellman optimal value function Q^* by approximating the maximum operator in the Bellman optimal equation with an in-sample expectile regression. IQL minimizes the following objectives:

$$\min_Q \mathcal{L}^Q_{\text{IQL}}(Q) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma V(s') - Q(s,a))^2], \tag{8}$$

$$\min_V \mathcal{L}^V_{\text{IQL}}(V) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[\ell^2_\tau(\bar{Q}(s,a) - V(s))], \tag{9}$$

where $\ell^2_\tau(x) = |\tau - \mathbb{1}(x < 0)|x^2$ is the expectile loss [40] with an expectile parameter τ. Intuitively, when $\tau > 0.5$, the expectile loss in Equation (9) penalizes positive errors more than negative errors, which makes V closer to the maximum value of $\bar{Q}$. In this way, IQL approximates V^* and Q^* only with in-distribution dataset actions, without referring to the erroneous values at out-of-distribution actions.

Contrastive RL (CRL). Contrastive RL (CRL) [11] is a value learning algorithm for offline goal-conditioned RL based on contrastive learning. CRL maximizes the following objective:

$$\max_f \mathcal{J}_{\text{CRL}}(f) = \mathbb{E}_{s,a\sim\mathcal{D},g\sim p^+_\mathcal{D}(\cdot|s,a),g^-\sim p^+_\mathcal{D}(\cdot)}[\log\sigma(f(s,a,g)) + \log(1-\sigma(f(s,a,g^-)))], \tag{10}$$

where σ denotes the sigmoid function and $p^+_\mathcal{D}(\cdot \mid s,a)$ denotes the geometric future state distribution of the dataset $\mathcal{D}$. Eysenbach et al. [11] show that the optimal solution of Equation (10) is given as $f^*(s,a,g) = \log(p^+_\mathcal{D}(g \mid s,a)/p^+_\mathcal{D}(g))$, which gives us the behavioral goal-conditioned Q function as $Q^\beta(s,a,g) = p^+_\mathcal{D}(g \mid s,a) = p^+_\mathcal{D}(g)e^{f^*(s,a,g)}$, where $p^+_\mathcal{D}(g)$ is a policy-independent constant.

D.2 Environments and datasets

We describe the environments and datasets we employ in our analysis in this section.

D.2.1 Data-scaling analysis

For the data-scaling analysis in Section 4, we employ the following environments and datasets (Figure 9).

- `antmaze-large` and `gc-antmaze-large` are based on the `antmaze-large-diverse-v2` environment from the D4RL suite [12], where the agent must be able to manipulate a quadrupedal robot to reach a given target goal (`antmaze-large`) or to reach any goal from any other state (`gc-antmaze-large`) in a given maze. For the dataset for `gc-antmaze-large` in our data-scaling analysis, we collect 10M transitions using a noisy expert policy that navigates through the maze. We use the same policy and noise level ($\sigma_{\text{data}} = 0.2$) as the one used to collect `antmaze-large-diverse-v2` in D4RL.
- `d4rl-hopper` and `d4rl-walker2d` are the `hopper-medium-v2` and `walker2d-medium-v2` tasks from the D4RL locomotion suite. We use the original 1M-sized datasets collected by partially trained policies [12].
- `exorl-walker` and `exorl-cheetah` are the `walker-run` and `cheetah-run` tasks from the ExORL benchmark [60]. We use the original 10M-sized datasets collected by RND agents [6].

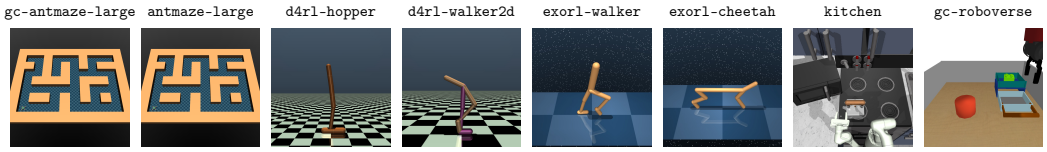


Figure 9: **Environments.**

Since the datasets are collected by purely unsupervised exploratory policies, they feature high suboptimality and high state-action diversity.

- `kitchen` is based on the `kitchen-mixed-v0` task from the D4RL suite, where the goal is to complete four manipulation tasks (*e.g.*, opening the microwave, moving the kettle) with a robot arm. Since the original dataset size is relatively small, for our data-scaling analysis, we collect a large 1M-sized dataset with a noisy, biased expert policy, where we add noises sampled from a zero-mean Gaussian distribution with a standard deviation of 0.2 in addition to a randomly initialized policy's actions to the expert policy's actions.
- `gc-roboverse` is a pixel-based goal-conditioned robotic task, where the goal is to manipulate a robot arm to rearrange objects to match a target image. The agent must be able to perform object manipulation purely from $48 \times 48 \times 3$ images. We use the 1M-sized dataset used by Park et al. [41], Zheng et al. [61].

D.2.2 Policy generalization analysis

For the policy generalization analysis in Section 5, we use the `antmaze-medium-diverse-v2`, `antmaze-large-diverse-v2`, `kitchen-partial-v0`, `kitchen-mixed-v0`, `pen-cloned-v1`, `hammer-cloned-v1`, and `door-cloned-v1` environments and datasets from the D4RL suite [12].

D.3 Data-scaling matrices

We train agents for 1M steps (500K steps for `gc-roboverse`) with each pair of value learning and policy extraction algorithms. We evaluate the performance of the agent every 100K steps with 50 rollouts, and report the performance averaged over the last 3 evaluations and over 4 seeds. In Figures 1 and 6, we individually tune the policy extraction hyperparameter (α for AWR and DDPG+BC, and N for SfBC) for each cell, and report the performance with the best hyperparameter. To save computation, we extract multiple policies with different hyperparameters from the same value function (note that this is possible because we use decoupled offline RL algorithms). To generate smaller-sized datasets from the original full dataset, we randomly shuffle trajectories in the original dataset using a fixed random seed, and take the first K trajectories such that smaller datasets are fully contained in larger datasets.

D.4 MSE metrics

We randomly split the trajectories in a dataset into a training set (95%) and a validation set (5%) in our experiments. For the expert policies π^* in the MSE metrics defined in Equations (4) to (6), we use either the original expert policies from the D4RL suite (`adroit-{pen, hammer, door}` and `gc-antmaze-large`) or policies pre-trained with offline-to-online RL until their performance saturates (`antmaze-{medium, large}` and `kitchen-mixed`). To train "global" expert policies for `antmaze-{medium, large}`, we reset the agent to arbitrary locations in the entire maze. This initial state distribution is only used to train an expert policy; we use the original initial state distribution for the other experiments.

D.5 Test-time policy improvement strategies

In Section 5, we introduce two test-time policy improvement strategies: OPEX and TTT.

On-the-fly policy extraction (OPEX). At test time, after sampling an action from the policy $a \sim \pi(\cdot \mid s)$, OPEX adjusts the action with the following formula:

$$a \leftarrow a + \beta \cdot \nabla_a Q(s, a), \tag{11}$$

where β is a hyperparameter that controls the test-time "learning rate". Intuitively, OPEX updates the action in the direction that maximally increases the learned Q function. In practice, we clip the action to be within $[-1, 1]$ after this adjustment.

Test-time training (TTT). TTT updates the parameters of (only) the offline RL policy with online interaction data to further distill the information in the fixed, learned value function into the policy. Specifically, TTT maximizes the following objective:

$$\max_{\pi} \mathcal{J}_{\mathrm{TTT}}(\pi) = \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s, \mu^{\pi}(s)) - \beta \cdot D_{\mathrm{KL}}(\pi^{\mathrm{off}} \parallel \pi)], \tag{12}$$

where $\mu^{\pi}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[a]$, π^{off} is the learned offline RL policy, and β is a hyperparameter that controls the strength of the regularizer. Equation (12) only trains π with test-time interaction data, while Q and π^{off} remain fixed. Intuitively, Equation (12) is a "parameter-updating" version of OPEX, where we adjust the parameters of the policy to maximize the learned value function, while not deviating too far away from the learned offline RL policy.

In Figure 7, for IQL, SfBC, and OPEX, we train IQL agents for 1M (`antmaze`) or 500K gradient steps (`kitchen`). For TTT, we further train the policy up to 2M gradient steps. In `antmaze`, we consider both deterministic evaluation and stochastic evaluation with a fixed standard deviation of 0.4 (which roughly matches the learned standard deviation of the BC policy), and report the best performance of them for each method.

D.6 State representation experiments

We describe the state representation ϕ used in Appendix C. An `antmaze` state consists of a 2-D x-y coordinates and 27-D proprioceptive information. We transform x and y individually with 32 $\tanh$ kernels, *i.e.*,

$$\tilde{x}_i = \tanh\left(\frac{x - x_i}{\delta_x}\right) \tag{13}$$

$$\tilde{y}_i = \tanh\left(\frac{y - y_i}{\delta_x}\right), \tag{14}$$

where $i \in \{1, 2, \ldots, 32\}$, $\delta_x = x_2 - x_1$, $\delta_y = y_2 - y_1$, and $x_1, \ldots, x_{32}$ and $y_1, \ldots, y_{32}$ are defined as `numpy.linspace(-2, 38, 32)` and `numpy.linspace(-2, 26, 32)`, respectively. Denoting the 27-D proprioceptive state as s_{proprio}, $\phi(s)$ is defined as follows: $\phi([x, y; s_{\mathrm{proprio}}]) = [\tilde{x}_1, \ldots, \tilde{x}_{32}, \tilde{y}_1, \ldots, \tilde{y}_{32}; s_{\mathrm{proprio}}]$, where ';' denotes concatenation. Intuitively, ϕ is similar to the discretization of the x-y dimensions with 32 bins, but with a continuous, invertible $\tanh$ transformation instead of binary discretization.

D.7 Implementation details

Our implementation is based on `jaxrl_minimal` [19] and the official implementation of HIQL [41] (for offline goal-conditioned RL). We use an internal cluster consisting of A5000 GPUs to run our experiments. Each experiment in our work takes no more than 18 hours.

D.7.1 Data-scaling analysis

Default hyperparameters. We mostly follow the original hyperparameters for IQL [24], goal-conditioned IQL [41], and CRL [11]. Tables 1 and 2 list the common and environment-specific hyperparameters, respectively. For SARSA, we use the same implementation as IQL, but with the standard ℓ^2 loss instead of an expectile loss. For pixel-based environments (*i.e.*, `gc-roboverse`), we use the same architecture and image augmentation as Park et al. [41]. In goal-conditioned environments and the `antmaze` tasks, we subtract 1 from rewards, following previous works [24, 41].

Policy extraction methods. We use Gaussian distributions (without $\tanh$ squashing) to model action distributions. We use a fixed standard deviation of 1 for AWR and DDPG+BC and a learnable standard deviation for SfBC. For DDPG+BC, we clip actions to be within the range of $[-1, 1]$ in the deterministic policy gradient term in Equation (2). We empirically find that this is better than $\tanh$ squashing [14] across the board, and is important to achieving strong performance in some environments. We list the policy extraction hyperparameters we consider in our experiments in curly brackets in Table 2.

Table 1: **Common hyperparameters for data-scaling matrices.**

Hyperparameter	Value
Learning rate	0.0003
Optimizer	Adam [23]
Target smoothing coefficient	0.005
Discount factor γ	0.99

Table 2: **Environment-specific hyperparameters for data-scaling matrices.**

Environment	`gc-antmaze-large`	`antmaze-large`	`d4rl-hopper`	`d4rl-walker`
# gradient steps	10^6	10^6	10^6	10^6
Minibatch size	1024	256	256	256
MLP dimensions	$(512, 512, 512)$	$(256, 256)$	$(256, 256)$	$(256, 256)$
IQL expectile	0.9	0.9	0.7	0.7
LayerNorm [3]	True	False	True	True
AWR α (IQL)	$\{0, 1, 3, 10\}$	$\{0, 3, 10, 30\}$	$\{0, 1, 3, 10\}$	$\{0, 1, 3, 10\}$
AWR α (SARSA/CRL)	$\{0, 10, 30, 100\}$	$\{0, 3, 10, 30\}$	$\{0, 1, 3, 10\}$	$\{0, 1, 3, 10\}$
DDPG+BC α (IQL)	$\{0.1, 0.3, 1, 3\}$	$\{0.1, 0.3, 1, 3\}$	$\{1, 3, 10, 30\}$	$\{1, 3, 10, 30\}$
DDPG+BC α (SARSA/CRL)	$\{0.1, 0.3, 1, 3\}$	$\{0.1, 0.3, 1, 3\}$	$\{1, 3, 10, 30\}$	$\{1, 3, 10, 30\}$
SfBC N (IQL)	$\{1, 16, 64\}$	$\{1, 16, 64\}$	$\{1, 16, 64\}$	$\{1, 16, 64\}$
SfBC N (SARSA/CRL)	$\{1, 16, 64\}$	$\{1, 16, 64\}$	$\{1, 16, 64\}$	$\{1, 16, 64\}$

Environment	`exorl-walker`	`exorl-cheetah`	`kitchen`	`gc-roboverse`
# gradient steps	10^6	10^6	10^6	5×10^5
Minibatch size	1024	1024	1024	256
MLP dimensions	$(512, 512, 512)$	$(512, 512, 512)$	$(512, 512, 512)$	$(512, 512, 512)$
IQL expectile	0.9	0.9	0.7	0.7
LayerNorm [3]	True	True	False	True
AWR α (IQL)	$\{0, 1, 10, 100\}$	$\{0, 1, 10, 100\}$	$\{0, 1, 3, 10\}$	$\{0, 0.1, 1, 10\}$
AWR α (SARSA/CRL)	$\{0, 1, 10, 100\}$	$\{0, 1, 10, 100\}$	$\{0, 1, 3, 10\}$	$\{0, 1, 10, 100\}$
DDPG+BC α (IQL)	$\{0, 0.01, 0.1, 1\}$	$\{0, 0.01, 0.1, 1\}$	$\{10, 30, 100, 300\}$	$\{3, 10, 30, 100\}$
DDPG+BC α (SARSA/CRL)	$\{0, 0.01, 0.1, 1\}$	$\{0, 0.01, 0.1, 1\}$	$\{10, 30, 100, 300\}$	$\{3, 10, 30, 100\}$
SfBC N (IQL)	$\{1, 16, 64\}$	$\{1, 16, 64\}$	$\{1, 16, 64\}$	$\{1, 16, 64\}$
SfBC N (SARSA/CRL)	$\{1, 16, 64\}$	$\{1, 16, 64\}$	$\{1, 16, 64\}$	$\{1, 16, 64\}$

D.7.2 Policy generalization analysis

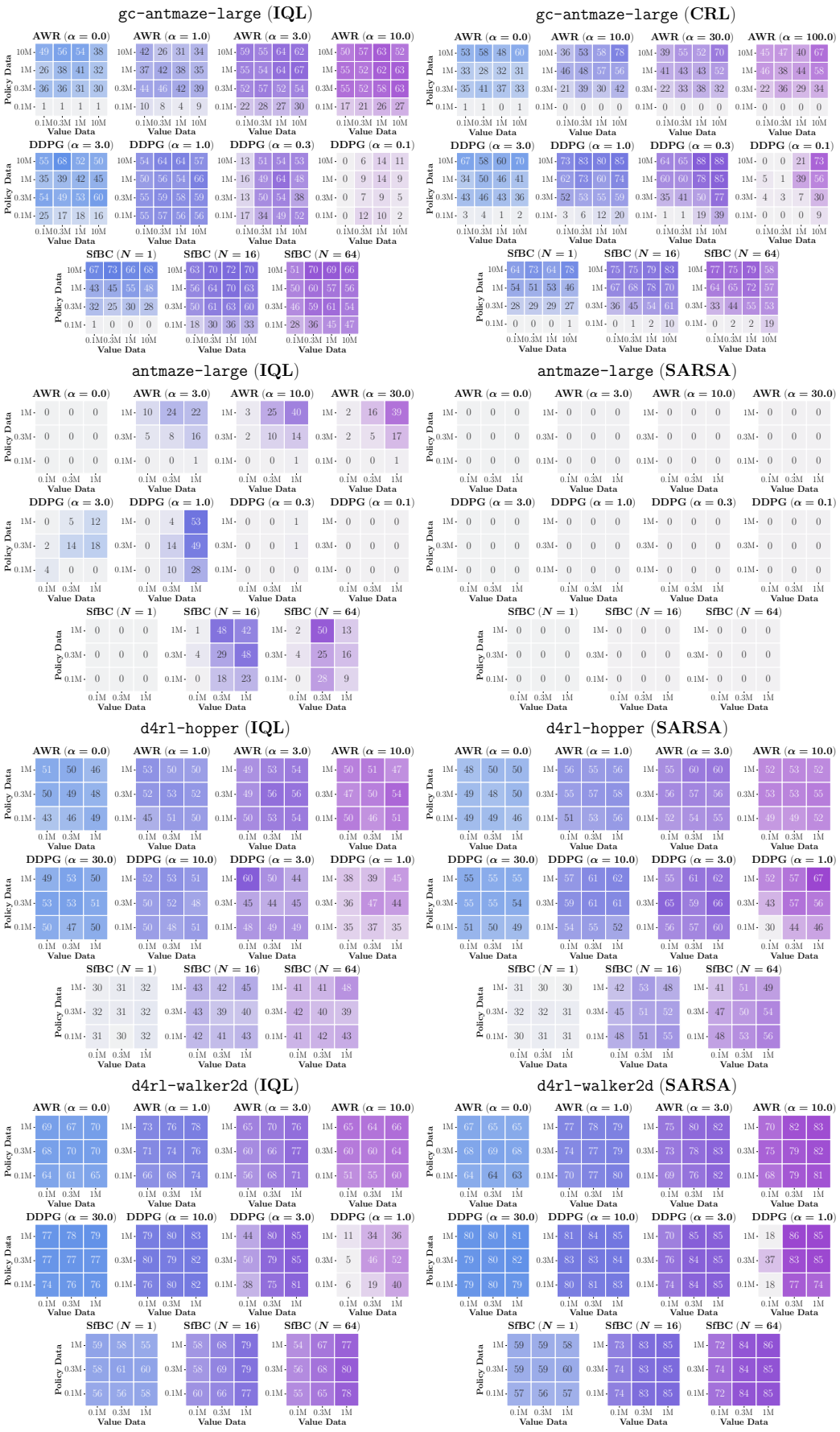
Hyperparameters. Table 3 lists the hyperparameters that we use in our offline-to-online RL and test-time policy improvement experiments. In these experiments, we use Gaussian distributions with learnable standard deviations for action distributions.

E Additional results

We provide the full data-scaling matrices with different policy extraction hyperparameters (α for AWR and DDPG+BC, and N for SfBC) in Figure 10.

Table 3: **Hyperparameters for policy generalization analysis.**

Hyperparameter	Value
Learning rate	0.0003
Optimizer	Adam [23]
# offline gradient steps	10^6 (`antmaze`), 5×10^5 (`kitchen`, `adroit`)
# total gradient steps	2×10^6
# gradient steps per environment step	1
Minibatch size	1024 (`kitchen`), 256 (`antmaze`, `adroit`)
MLP dimensions	$(512, 512, 512)$ (`kitchen`), $(256, 256)$ (`antmaze`, `adroit`)
Target smoothing coefficient	0.005
Discount factor γ	0.99
LayerNorm [3]	True (`kitchen`), False (`antmaze`, `adroit`)
IQL expectile	0.9 (`antmaze`), 0.7 (`kitchen`, `adroit`)
Policy extraction method	AWR
AWR α	10 (`antmaze`), 0.5 (`kitchen`), 3 (`adroit`)
SfBC N	16
OPEX β	0.3 (`antmaze`), 0.0003 (`kitchen`)
TTT β	0.3 (`antmaze`), 5 (`kitchen`)


gc-antmaze-large (IQL)
gc-antmaze-large (CRL)
antmaze-large (IQL)
antmaze-large (SARSA)
d4rl-hopper (IQL)
d4rl-hopper (SARSA)
d4rl-walker2d (IQL)
d4rl-walker2d (SARSA)
AWR (α = 0.0)
AWR (α = 1.0)
AWR (α = 3.0)
AWR (α = 10.0)
AWR (α = 30.0)
AWR (α = 100.0)
DDPG (α = 3.0)
DDPG (α = 1.0)
DDPG (α = 0.3)
DDPG (α = 0.1)
DDPG (α = 30.0)
DDPG (α = 10.0)
SfBC (N = 1)
SfBC (N = 16)
SfBC (N = 64)
Policy Data
Value Data

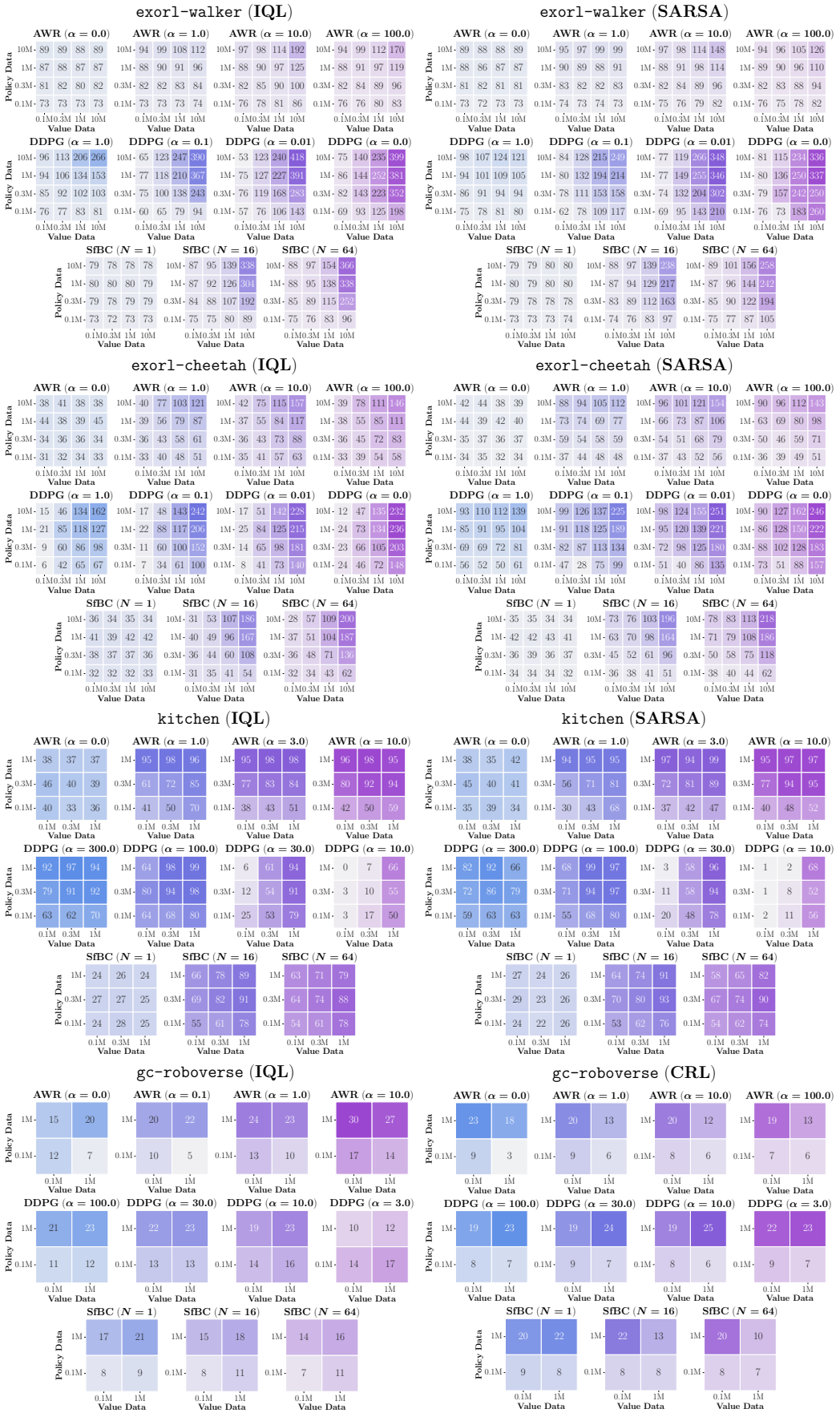


Figure 10: **Full data-scaling matrices of AWR, DDPG+BC, and SfBC with different hyperparameters.**