

JumpStart Your Policy Learning with Lessons from 160,000 Training Runs

Nabil Omi^{1*} Eric Bae¹ Chung Yik Edward Yeung¹ Siddhartha Sen² Ali Farhadi^{1,2}
¹University of Washington ²Microsoft

Abstract

Reliable progress in offline policy learning depends on careful reporting, well-tuned baselines, and evaluation across diverse conditions. Prior work has shown that results can be sensitive to reporting choices, hyperparameter tuning, and dataset properties, but these sources of variability have not been systematically investigated together at the scale needed to understand how they shape conclusions. To address this gap, we present a large-scale empirical study of offline reinforcement and imitation learning, training over 160,000 policies across 114 datasets. At this scale, no algorithm dominates: aggregate performance among the strongest methods is often close, but the leaders differ substantially across environments. We find that proper hyperparameter tuning frequently reshuffles perceived algorithm rankings and that benchmark composition can produce conflicting conclusions. We also study hyperparameter sensitivity and transfer across environments, identifying a simple strategy for deriving strong default configurations. We use our findings to develop a dataset-conditioned recommender that provides task-specific algorithm recommendations for practitioners. Finally, we release JumpStart: a resource suite containing every trained policy, per-model scores and hyperparameters, strong baselines across all environments, training and evaluation code, and an extensible website for retrieving, analyzing, and contributing results. Together, these resources aim to make offline policy-learning research more reliable and enable future work beyond the scope of this study.

[Code](#) • [Models and Data](#) • [Website](#)

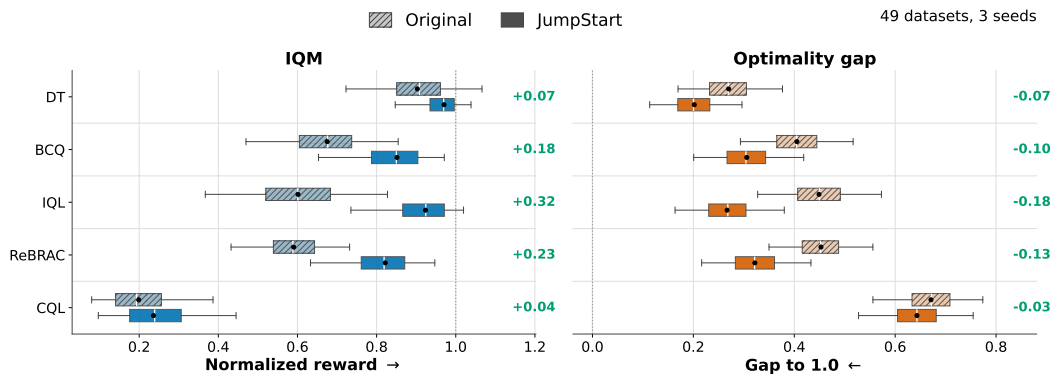


Figure 1: Our tuned models substantially outperform published baselines on Minari and change the relative ordering of algorithms. Original paper implementations use the authors’ recommended or closest available hyperparameters. Further details are provided in Appendix B.

*Corresponding author: nabilomi@cs.uw.edu

1 Introduction

Learning policies from fixed datasets offers an attractive alternative when online interaction is expensive, slow, or unsafe. Offline Reinforcement Learning (ORL) uses reward-labeled data to optimize a policy without further interaction, while Imitation Learning (IL) learns to reproduce behavior from demonstrations. We use the term *offline policy learning* to encompass both settings, since they share the central empirical challenge studied here: reliably training and comparing policies using pre-collected data.

This comparison is harder than standard benchmark results often suggest. Performance can change substantially with the choice of dataset, environment, hyperparameters, evaluation protocol, and aggregation rule. When these factors vary across studies, it becomes difficult to determine whether an apparent improvement comes from the algorithm itself or from the comparison surrounding it. Small evaluation suites and limited tuning budgets make these sensitivities particularly difficult to see.

Prior work has exposed individual parts of this problem. Offline RL performance can depend strongly on reward normalization, initialization, policy parameterization, network architecture, learning-rate schedules, and other implementation choices [1]. Hidden layer size [2], dataset coverage and trajectory quality [3], reward sparsity, horizon, and stochasticity [4], and the use of human rather than synthetic demonstrations [5] can all change which methods appear strongest. Aggregation and uncertainty reporting introduce further ambiguity [6]. Cross-paper comparisons compound these problems: studies use different environment subsets, evaluation budgets, and benchmark versions, and numerical results are sometimes transferred between settings where they are not directly comparable, such as different environment versions [7].

We present a large-scale empirical study designed to make these sensitivities visible. Using nearly one million GPU-hours, we train more than 160,000 policies across 114 datasets spanning continuous control, robotics, navigation, and Atari. Figure 1 shows that our tuned models not only substantially outperform reference runs constructed from the original implementations and published hyperparameters, but also change the relative ordering of the algorithms. Across the 49 shared datasets, IQL overtakes BCQ under both IQM and optimality gap after tuning. Hyperparameter selection can therefore change the conclusion of a comparison, not simply the absolute scores it reports. The reference implementations are modified only as needed to load the standardized datasets we use, we otherwise follow the authors’ tuning guidance or use the configuration from the most similar reported dataset. Full matching and tuning details are provided in Appendix B. We use the resulting corpus to study how tuning, benchmark composition, aggregation, and algorithm design affect empirical conclusions.

Our primary contributions are:

- We conduct an extensively tuned comparison of offline policy-learning algorithms across 114 datasets, producing more than 160,000 trained policies and a controlled view of performance across continuous- and discrete-control domains.
- We characterize how hyperparameter choice, tuning budget, and random seed affect absolute performance and algorithm rankings, identify the most influential hyperparameters and practical tuning budgets, and derive strong default configurations that transfer across datasets.
- We show that benchmark composition can produce conflicting algorithm rankings even when the selected subsets are comparable in size to established evaluation suites, and we propose reporting practices that make this sensitivity visible.
- We systematically analyze how dataset characteristics relate to algorithm performance and use these relationships to recommend promising algorithms for new tasks.
- We release the JumpStart corpus, including trained policies, hyperparameters, training code, and a website for retrieving, comparing, analyzing, and extending results.

2 Background and Related Work

In this section, we briefly review the background needed to place our study in context. We first outline the main algorithm families we study, then summarize the datasets and benchmarks used in

offline policy learning, and finally discuss prior empirical studies on what makes algorithms work or fail, along with related issues in evaluation, reporting, and tuning.

Algorithms. We study a mix of popular imitation learning and offline RL methods. Behavior Cloning (BC) fits actions directly through supervised learning. Action Chunking with Transformers (ACT; 8) predicts short action sequences to reduce compounding error in manipulation settings. VQ-BeT [9] uses vector-quantized latent actions to capture multimodal behavior. Diffusion Policy [10] models action sequences through conditional denoising, allowing it to represent multimodal action distributions and perform receding-horizon control. Decision Transformer (DT; 11) models trajectories autoregressively and conditions its predictions on a desired return-to-go. Implicit Q-Learning (IQL; 7) improves a policy through implicit value estimation without querying out-of-distribution actions. Conservative Q-Learning (CQL; 12) regularizes value estimates conservatively to stabilize offline Q-learning. ReBRAC [13] builds on TD3+BC with a set of specific design choices, including layer normalization, a deeper critic, a higher discount factor, and decoupled actor and critic penalties. Batch-Constrained deep Q-learning (BCQ; 14) constrains candidate actions to remain close to the support of the dataset.

Datasets. The data used for offline policy learning can vary widely in quality, coverage, and domain, and these properties can strongly influence which algorithms perform well. The most widely used benchmark is D4RL [15], which introduced a standardized suite for evaluating offline, or data-driven, reinforcement learning. The original suite includes Maze2D and AntMaze [15], Gym-MuJoCo [16, 17], Adroit [18], FrankaKitchen [19], Flow [20], and CARLA [21]. In practice, most subsequent evaluations focus on the first five groups and omit Flow and CARLA.

Prior to D4RL, studies generally generated their own non-standardized evaluation datasets [14, 22, 23]. These datasets were commonly produced by training an expert through online reinforcement learning and then retaining either its training replay buffer or trajectories generated by selected checkpoints. Later, Mandlekar et al. [5] introduced RoboMimic, a collection of human teleoperation trajectories for the RoboSuite framework [24]. They found that conclusions obtained from machine-generated datasets did not necessarily transfer to human demonstrations, a distinction that we examine more broadly in Section 4.2.

More recent benchmarks have aimed to address limitations of D4RL. Lu et al. [25] introduce V-D4RL, which provides pixel-based analogues of D4RL tasks for evaluating offline RL from visual observations. Liu et al. [26] release LIBERO, a procedurally generated suite of 130 language-conditioned manipulation tasks built around RoboSuite and designed to study lifelong knowledge transfer in imitation learning. Rafailov et al. [27] propose D5RL as a successor to D4RL, motivated by saturation on the original tasks, with more realistic manipulation and locomotion environments and heterogeneous data sources spanning scripted, teleoperated, and play-style data. Finally, Park et al. [28] introduce OGBench, an offline goal-conditioned RL benchmark spanning eight environment types and 85 datasets designed to probe capabilities such as stitching, long-horizon reasoning, and robustness to stochasticity.

In parallel, the Farama Foundation released Minari [29], a standardized dataset format and Python API that hosts reimplementations of D4RL and other reference offline RL datasets. Minari has partially supplanted the original D4RL codebase, which is no longer actively maintained. For reproducibility and ease of maintenance, as discussed further in Section 3.1, we use a 114-dataset subset of Minari. These datasets were collected from several sources, including humans, RL agents, and scripted planners. This diversity allows us to study not only transfer between human- and machine-generated data, but transfer across expert sources more generally in Section 4.2.

Reporting practices and reproducibility. Inconsistent reporting has long been an issue in deep RL. Agarwal et al. [6] recommend robust aggregate metrics accompanied by uncertainty estimates. We adopt two such metrics throughout: the interquartile mean (IQM), which discards the top and bottom 25% of normalized scores before averaging, and the optimality gap, which measures the average shortfall from a target performance level; we use expert performance as the target.

Canonical offline policy-learning papers differ in ways that make direct comparison difficult. For example, BCQ [14], BEAR [22], and BRAC [23] primarily present reward-over-training curves rather than consolidated benchmark tables, while Decision Transformer [11] reports results on a comparatively restricted subset of D4RL. In Section 4, we show that benchmark subsets comparable in size to established evaluation suites can nevertheless produce sharply different algorithm rankings.

Results are also sometimes copied across papers despite differences in the underlying environment versions, such as v0 versus v2, as noted by Kostrikov et al. [7].

Tarasov et al. [13] address part of this problem by providing single-file implementations of several algorithms and evaluating them on a unified set of D4RL dataset versions. Their primary goal, however, is implementation standardization rather than extensive per-dataset tuning. Our results show why this distinction matters: tuning can substantially change both absolute performance and the relative ordering of algorithms. For example, BC frequently appears near the bottom of their leaderboards, whereas after tuning we find it substantially more competitive.

Empirical studies. Andrychowicz et al. [30] train tens of thousands of on-policy RL agents while varying more than 50 implementation choices, finding that low-level details such as policy initialization, observation normalization, and action-distribution parameterization can matter as much as the high-level algorithm. Orsini et al. [31] reach a similar conclusion for adversarial imitation learning and additionally show that algorithm rankings shift substantially between synthetic and human demonstrations.

Paine et al. [32] conduct an early study of hyperparameter selection in offline RL. Across three algorithms and ten continuous-control tasks, they train 2,560 policies and find that performance varies substantially with hyperparameter choice. They also show that Fitted Q Evaluation can often rank the resulting policies without additional environment interaction. This addresses a different problem from ours: their policies have already been trained, and the goal is to select a strong candidate when the budget for evaluating those policies is limited.

This setup treats training many candidate policies as feasible while evaluation is the primary bottleneck. Although this remains relevant when deployment is expensive or risky, the balance is less clear for modern policy learning, where models may contain billions of parameters and be trained on large datasets [33]. Our goal is instead to compare algorithms after adequate tuning while reducing the cost of reaching that point. Across 162,864 policies from ten algorithms and 114 datasets, we identify which hyperparameters warrant attention and how their importance changes across datasets (Section 5.1). We then use the same sweeps to derive stronger default parameters, improving the performance of an initial run and reducing the amount of target-specific tuning required (Section 5.2).

Kang et al. [1] ablate 20 low-level implementation choices across CQL, CRR, and IQL on D4RL and RL Unplugged. They show that many reported gains arise from these choices rather than the central algorithm and that their effects depend strongly on the data distribution. Some of their recommendations do not transfer uniformly across our broader suite; for example, we find that the importance of learning rate varies substantially across datasets. Cetin et al. [2] similarly argue that scale can matter more than novelty in offline RL, showing that appropriately scaled simple methods can match or outperform more elaborate algorithms on D4RL. To control for model scale, we either include larger hidden sizes in our search spaces or use networks comparable in scale to the larger configurations in their study while varying critic count and layer width.

Dataset properties provide another explanation for inconsistent results. Schweighofer et al. [3] find that exploration coverage and trajectory quality strongly predict which methods perform well, while Bhargava et al. [4] show that the relative ordering of offline RL and imitation-style methods depends on properties such as reward sparsity, horizon, and demonstration source. We extend this line of work by automatically parsing environment and dataset descriptions into structured characteristics and using them in a practitioner-facing recommender that identifies promising algorithms for a given task. Prior studies therefore establish that implementations, hyperparameters, model scale, and dataset properties can all affect performance; our study brings these factors together and makes the resulting relationships directly actionable.

Our study brings these lines of analysis together across ten algorithms, 114 datasets, continuous- and discrete-control domains, and more than 160,000 trained policies. Because we extensively tune each algorithm on each dataset, we can analyze dataset effects without relying on potentially weak default configurations. In Section 6, we study these effects systematically and use environment and dataset characteristics to recommend promising algorithms for a given task. We additionally release the complete corpus of trained policies, hyperparameters, and training code so that future studies can reproduce, inspect, and extend our analysis. Table 1 summarizes the components combined in our study relative to the most closely related empirical work.

Table 1: Comparison with selected large-scale empirical studies in offline policy learning. Prior-work counts describe the principal offline benchmarks; JumpStart counts include the full hyperparameter-tuning corpus.

Study	Datasets or tasks	Methods / variants	Policies counted	Continuous	Discrete	Trained models	Policy metadata	Training code	Extensible leaderboard
Agarwal et al. [34] (Atari)	60 games	6	1,800 [†]	–	✓	–	–	✓	–
Paine et al. [32]	10 tasks	3	2,560 [†]	✓	–	–	–	–	–
Mandlekar et al. [5]	8 tasks	6	NR	✓	–	△	△	✓	–
Schweighofer et al. [3]	150 (6 envs.)	9	6,750	–	✓	–	–	✓	–
Seno and Imai [35]	17 datasets/tasks	12	1,400 [†]	✓	✓	–	✓	✓	–
Tarasov et al. [13]	30 datasets	10	1,200 [†]	✓	–	–	✓	✓	–
Kang et al. [1]	26 tasks	9	702 [†]	✓	–	–	–	✓	–
Bhargava et al. [4]	3 principal suites	3	NR	✓	✓	–	–	✓	–
JumpStart (ours)	114	10	162,864	✓	✓	✓	✓	✓	✓

Notes. NR denotes an unreported total; △ denotes a partial release. Method counts include separately evaluated variants. [†]Counts are reconstructed from reported experimental settings and exclude auxiliary ablations and offline-to-online fine-tuning. Policy metadata comprises policy-level scores paired with hyperparameters or experiment configurations. An extensible leaderboard accepts externally contributed results.

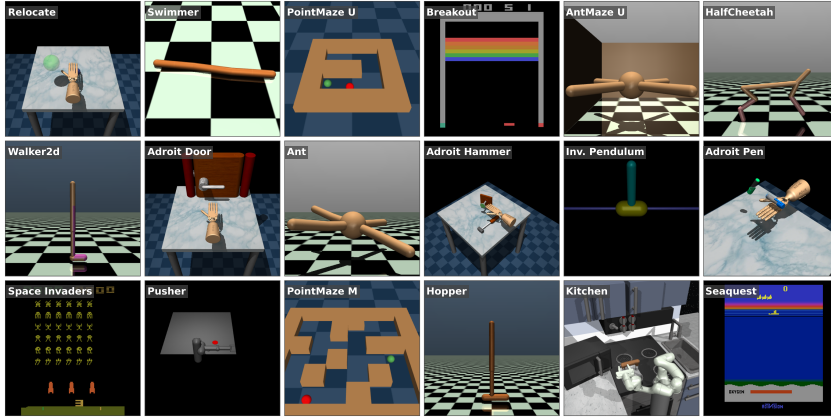


Figure 2: An 18-environment subset of the diverse tasks used in the study. A full list of environments and datasets can be found in Appendix A.

3 Training Methods

3.1 Datasets

Minari follows the same recipe as D4RL, generating offline datasets from trained policies [29], and additionally includes 7 human-collected datasets. We use Minari rather than the original D4RL release because it is the actively maintained successor, with newer environment versions, clearer documentation, and openly available dataset-generation code. Because the underlying datasets are not bit-identical to D4RL, some absolute numbers (e.g., CQL on AntMaze) may differ from prior reports. A subset of the tasks used in our study is visualized in Figure 2. For Figure 1, we adapted the source code of prior work [11, 13, 14, 7, 12] to support Minari and adapted hyper-parameters using the procedure from Tarasov et al. [13] (additionally following tuning suggestions from their papers, if available), and re-evaluated their models against ours.

We made two additions. The Atari datasets in Minari were too small for the discrete-control comparisons we wanted, so we collected new ones with 100k states per game using their dataset generation code, following the Atari100k setting [36]. We also omit a number of environments whose default datasets are trivially solvable across all methods we tested. The remaining 114 datasets (57 continuous control, 57 discrete) cover a wide range of distributional shapes, including some near-zero-reward degenerate datasets that we filter from our analyses, though they are still included in artifact releases for future studies. After filters, we land at 52 discrete and 57 continuous datasets. We provide the full set of datasets and their descriptions in Appendix A.

Table 2: Number of trained policies by algorithm and action space.

	BC	BCP	BCQ	CQL	DT	IQL	ReBRAC	ACT	VQ-BeT	Diffusion	Total
Continuous	11,400	11,400	11,400	11,400	11,400	11,302	11,401	10,830	8,833	10,590	109,956
Discrete	11,400	11,400	8,687	10,021	11,400	–	–	–	–	–	52,908
Total	22,800	22,800	20,087	21,421	22,800	11,302	11,401	10,830	8,833	10,590	162,864

D4RL provides fixed expert and random reference returns for each environment, whereas Minari currently provides these values only for datasets imported from D4RL. In our work, we define "expert" as the policy that generated the dataset used for offline policy learning. To apply a consistent normalization procedure across all environments, we estimate the expert reference directly from each dataset and estimate the random reference by running a random policy for 100 trajectories in each unique environment.

For a trajectory τ , let

$$R(\tau) = \sum_{t=0}^{T_\tau} r_t$$

denote its undiscounted return. For dataset $\mathcal{D}$ from environment e , we define

$$R_{\text{expert}}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} R(\tau), \quad R_{\text{random}}(e) = \frac{1}{100} \sum_{i=1}^{100} R(\tau_i^{\text{rand}}).$$

Given an evaluation return R_{eval} , we report

$$R_{\text{norm}} = \frac{R_{\text{eval}} - R_{\text{random}}(e)}{R_{\text{expert}}(\mathcal{D}) - R_{\text{random}}(e)}.$$

3.2 Model training

Our pipeline is built on d3rlpy [35] for verified algorithm implementations and Ray Tune [37] for distributed hyperparameter tuning. We completed full hyperparameter tuning runs with BC, BCP (BC with percentile-based data filtering, also called BC%, a common baseline [13, 11]), BCQ, CQL, and DT across all 114 continuous and discrete datasets; VQ-BeT, ACT, IQL, and ReBRAC run on 57 continuous control datasets. We start with the LeRobot² implementations for ACT and VQ-BeT and modify them to work with our codebase. For Diffusion Policy, we used a modified implementation from the original paper’s code release³. We validated the modified implementations on a subset of MuJoCo tasks in Minari by converting datasets to a format supported by the original code.

Training budgets. Value-based methods (CQL, BCQ, IQL, ReBRAC) use 500,000 gradient steps on continuous environments and 250,000 on discrete. BC, BCP, and DT use 50 epochs. Diffusion and ACT uses 100 epochs, and VQ-BeT uses 300 total split among the stages. These budgets were chosen via an exponentially increasing sweep on a small set of testing datasets (3 MuJoCo HalfCheetah datasets and D4RL AntMaze) under default and four random hyperparameter configurations, picking the smallest budget at which rewards plateau.

Hyperparameter tuning. We use random search [38] and the Tree-structured Parzen Estimator (TPE), which models better- and worse-performing configurations to bias subsequent trials toward promising regions [39]. We run TPE through Ray Tune’s OptunaSearch wrapper [40]. The search spaces are reported in Tables 4 and 5, and model counts are reported in Table 2. To test whether these ranges constrain performance, we compare how often the top 10% of trials and the full sweep contain at least one hyperparameter within 5% of a search-range boundary. If there is a significant difference in prevalence within the boundaries, the sweep may have been limited in width. As shown in Table 3, the median enrichment among top trials is only +0.1 percentage points, with no consistent increase across algorithms. Strong configurations therefore do not systematically accumulate near the edges of our search spaces, suggesting that the selected ranges are sufficiently broad.

²<https://github.com/huggingface/lerobot/tree/main/src/lerobot/policies/vqbet>,
<https://github.com/huggingface/lerobot/tree/main/src/lerobot/policies/act>

https://github.com/real-stanford/diffusion_policy

³https://github.com/real-stanford/diffusion_policy

Table 3: Search-space boundary occupancy. We report the percentage of trials containing at least one hyperparameter within 5% of either search-range boundary. Enrichment is the difference between the top 10% of trials and all trials.

Algorithm	Top 10%	All trials	Enrichment
ACT	32.8%	35.3%	−2.5 pp
BC	33.5%	33.9%	−0.4 pp
BCP	41.6%	40.3%	+1.3 pp
BCQ	85.4%	85.2%	+0.2 pp
CQL	83.3%	78.5%	+4.8 pp
DT	70.1%	69.9%	+0.1 pp
IQL	71.2%	71.1%	+0.1 pp
ReBRAC	74.6%	71.6%	+3.1 pp
VQ-BeT	62.9%	64.6%	−1.7 pp

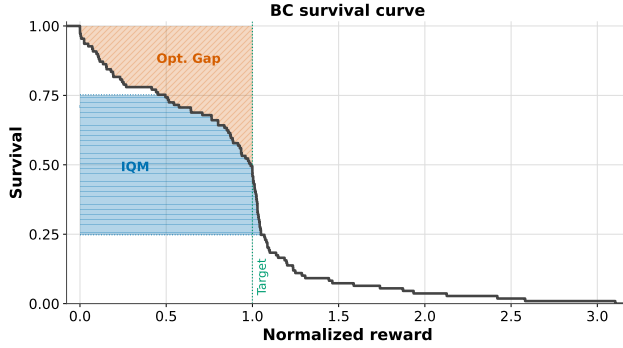


Figure 3: A visual example of optimality gap and the IQM overlaid on a survival chart for behavior cloning on our suite.

Evaluation. For all evaluations, we use the final checkpoint evaluated across 100 episodes. Minari provides a method to recover the environment used to collect each dataset, but sometimes omits wrappers (e.g., resizing or frame stacking), as in Atari. In these cases, we recover the missing wrappers from Minari’s data-generation scripts.⁴

4 Evaluation Sensitivity

Aggregates can hide more than they show. The apparent best algorithm depends materially on the choice of **metric**, **data**, and **tuning budget**. We treat each as a distinct failure mode and recommend evaluation practices that make the resulting ambiguity legible.

The choice of evaluation metric has been studied extensively in reinforcement learning. Following Agarwal et al. [6], we avoid reporting only means or medians over many runs and datasets, and instead use the interquartile mean (IQM) and optimality gap as robust aggregate metrics. For normalized scores $x_1, \dots, x_n$, let $q_{0.25}$ and $q_{0.75}$ denote the empirical first and third quartiles. The IQM is

$$\text{IQM}(x) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} x_i, \quad \mathcal{I} = \{i : q_{0.25} \leq x_i \leq q_{0.75}\}.$$

For an optimality threshold γ , the optimality gap is

$$\text{OG}_\gamma(x) = \frac{1}{n} \sum_{i=1}^n \max(\gamma - x_i, 0).$$

In our setting, normalized expert performance corresponds to $\gamma = 1$. Since metric aggregation is already well explored, we adopt these methodologies and focus our analysis on the remaining sources of uncertainty. We provide a visual explanation of IQM and optimality gap in Figure 3.

⁴<https://github.com/Farama-Foundation/minari-dataset-generation-scripts/tree/main>

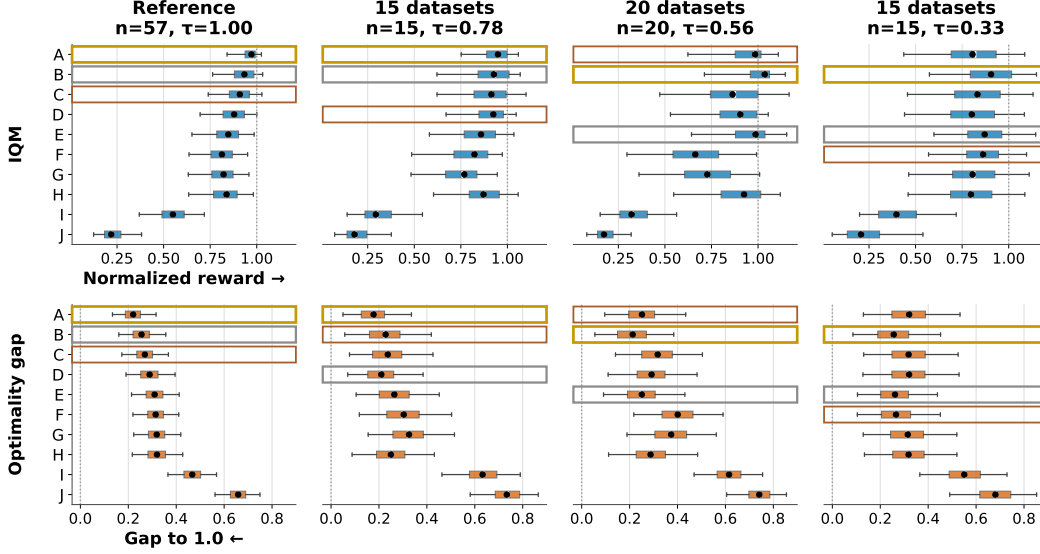


Figure 4: 15-dataset subset leaderboard constructions with different Kendall’s τ to the full-environment leaderboard. Even with 15-20 different datasets, it is possible to have rankings in different orders, with the top 3 (gold, silver, bronze outlined) varying substantially. Letters A-J indicate different algorithms.

We compare algorithm orderings, or leaderboards, using Kendall’s τ . Let $r_i^{(a)}$ and $r_i^{(b)}$ denote the rank of algorithm i under two leaderboards a and b , respectively, with rank 1 corresponding to the best-performing method. For each pair of algorithms (i, j) , we compute

$$c_{ij} = \text{sign} \left[\left(r_i^{(a)} - r_j^{(a)} \right) \left(r_i^{(b)} - r_j^{(b)} \right) \right],$$

so that $c_{ij} = 1$ if the two leaderboards order the pair consistently, $c_{ij} = -1$ if they reverse the pair, and $c_{ij} = 0$ if the pair is tied in either ordering. To visualize the disagreement values, we show a reference leaderboard and a few alternatives with varying Kendall’s τ values in Figure 4. Letters A-J indicate different algorithms, and as ordering deviates from the reference leaderboard, the Kendall’s τ drops.

We begin by analyzing how the tuning and budget can materially change the conclusions of a study in Section 4.1. We additionally compare the variance across different hyperparameter sets to the seed variance, and suggest an allocation of resources that removes more uncertainty than simply running more seeds.

In Section 4.2, we analyze how the separation, choice and quantity of datasets impacts reliability, and generalize on prior work finding human-generated datasets behaving differently from machine-generated ones, finding that this can be applied to *any* different expert source: planner-based solvers, reinforcement learning, cloned datasets, and human.

Finally, in Section 4.3, we construct leaderboards and summary tables showing final algorithm performance after hyperparameter tuning.

Takeaway. We use boxes like this to close each analysis with its main takeaway.

4.1 Tuning budget and seed variance

Running 200 hyperparameter trials for every algorithm–dataset pair is beyond the budget of many evaluations. We therefore use the complete sweep as a reference and ask how quickly smaller searches recover both its best observed performance and its algorithm ordering.

In continuous-control datasets, these two quantities converge at different rates. Figure 5 shows that most of the improvement in normalized score occurs within the first 25–50 trials, with smaller

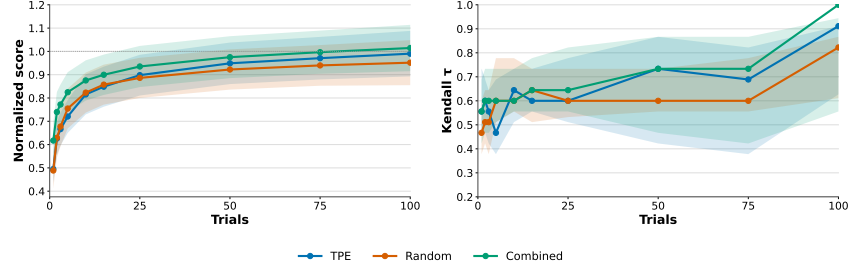


Figure 5: Tuning progress in continuous-control. Most of the improvement in best observed normalized score occurs within 25–50 trials, whereas agreement with the full-budget algorithm ranking improves more gradually. Confidence intervals are across datasets, where peak performance can vary substantially.

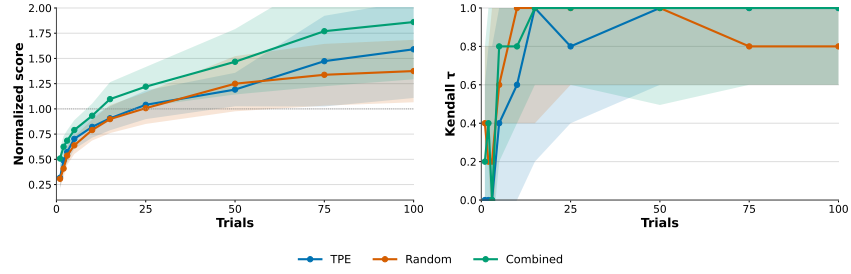


Figure 6: Tuning progress in discrete-control. Agreement with the full-budget ranking increases rapidly during the first 10–25 trials, while the best observed normalized score continues to improve with additional tuning. Confidence intervals are across datasets, where peak performance can vary substantially.

gains thereafter. Agreement with the full-budget ranking improves more slowly. A search may therefore find strong configurations before it provides a reliable comparison between algorithms: continuous-control leaderboards remain sensitive to tuning budget even after performance begins to plateau.

Ranking agreement increases more quickly in discrete control. Figure 6 shows a sharp improvement during the first 10–25 trials, after which Kendall's τ remains generally high despite some fluctuations. The best observed score continues to increase throughout the search.

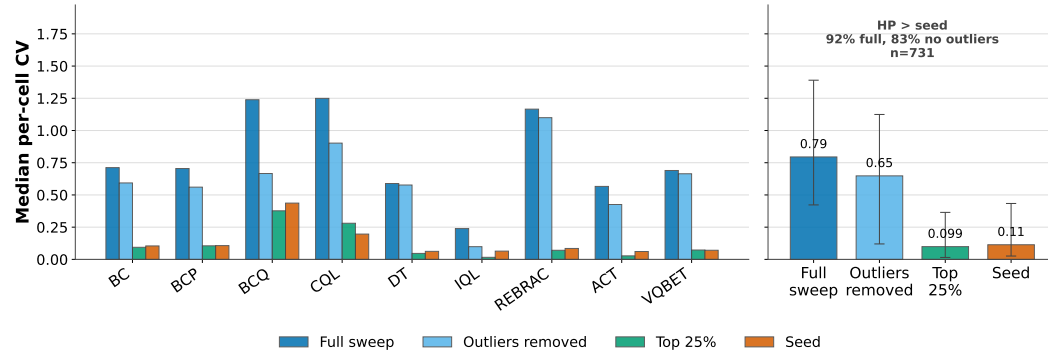


Figure 7: Variability across hyperparameter configurations and random seeds. The left panel reports the median per-cell coefficient of variation for each algorithm, and the right panel summarizes 731 algorithm–dataset cells. Hyperparameter variability exceeds seed-wise variability in 92% of cells, or 83% after removing outlier configurations. Within the top 25% of configurations, however, variability is comparable to that observed across seeds.

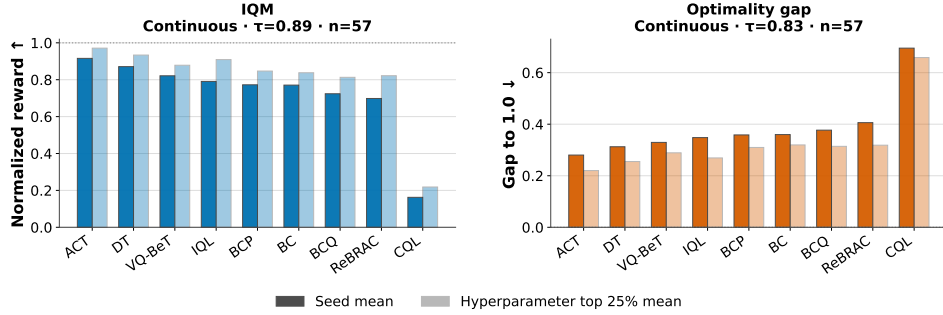


Figure 8: Comparison between continuous-control leaderboards constructed from the top-quartile mean of the hyperparameter sweep and from independently seeded runs. IQM and optimality-gap estimates are nearly identical, and the resulting algorithm rankings have Kendall correlations of $\tau = 0.94$ for IQM and $\tau = 0.83$ for optimality gap.

Tuning and seed replication answer different questions. Additional configurations explore sensitivity to hyperparameter choice and increase the chance of finding a strong policy, whereas repeated seeds estimate the stochastic variability of a fixed configuration. A finite evaluation budget must therefore balance search-space exploration against replication of the selected configurations.

Across the complete sweep, hyperparameter choice is the larger source of variation. As shown in Figure 7, variation across configurations exceeds seed-wise variation in 92% of algorithm–dataset cells and in 83% after removing outlier configurations. Results based on only a few configurations can therefore depend more strongly on which hyperparameters were tried than on which random seeds were used. Seed replication is necessary for estimating uncertainty between runs, but additional seeds cannot compensate for an inadequately explored search space and leaves a much larger source of uncertainty unexplored. For our experiments we pick the 95th percentile hyperparameter for each dataset–algorithm pair tested, and train for 5 seeds.

The upper-performing region is considerably more concentrated. Among the top 25% of configurations, variation across hyperparameters is comparable to variation across seeds. We therefore average the normalized scores of the top 25% of configurations within each algorithm–dataset pair, producing one robust near-optimal score per pair, and compute IQM and optimality gap across these dataset-level scores.

Figure 8 validates this approximation against independently seeded runs. The two procedures produce nearly identical IQM and optimality-gap estimates, while their continuous-control rankings agree strongly: Kendall’s τ is 0.94 for IQM and 0.83 for optimality gap.

Takeaway. Hyperparameter searches may find many strong configurations long before providing a stable comparison between algorithms.

Takeaway. Hyperparameter choice dominates variation across the full search, while the top of the search is much more concentrated. A budget of 25–50 trials provides a useful practical compromise.

4.2 Datasets and Environments

The datasets and environments included in an evaluation shape both the stability and interpretation of its leaderboard. We examine how rankings change with benchmark size and composition, and whether they transfer across different expert sources.

4.2.1 Benchmark Size and Composition

Benchmark composition can materially change the conclusion of an evaluation. Small benchmarks are naturally vulnerable to unrepresentative sampling, but size alone may not solve the problem: we

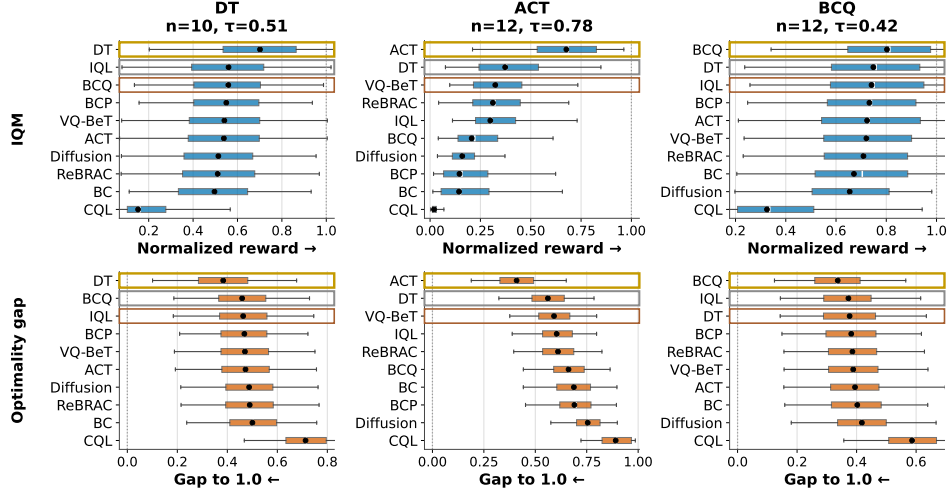


Figure 9: Selected benchmark subsets that favor different algorithms. We sample 2,000 subsets at each tested size ($n \in \{10, 12, 15, 20\}$). Among these candidates, subsets containing 12, 10, and 12 environment–dataset pairs place ACT, DT, and BCQ first by a margin, respectively, under both IQM and optimality gap.

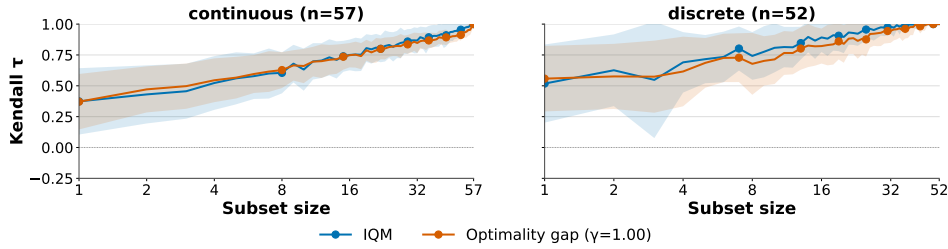


Figure 10: Agreement between subset and full-suite leaderboards as a function of subset size. Lines show the mean Kendall’s τ over randomly sampled subsets, and shading shows one standard deviation. Agreement increases and variability across subsets decreases as more environment–dataset pairs are included.

ask whether subsets comparable in size to established evaluation suites can still produce substantially different algorithm rankings.

We draw 2,000 random subsets independently at each of four sizes, $n \in \{10, 12, 15, 20\}$, producing 8,000 candidate benchmarks. We then search these candidates for subsets that favor different algorithms. Figure 9 shows that ACT, DT, and BCQ can each be placed first under both IQM and optimality gap. The selected subsets contain 12, 10, and 12 environment–dataset pairs, respectively, and are therefore not unusually small. An algorithm other than the winner on the full 57-dataset continuous-control benchmark ranks first under both IQM and optimality gap in 24.70%, 23.80%, 20.80%, and 15.45% of subsets of sizes $n = 10, 12, 15, 20$, respectively. For comparison, the continuous-control evaluation in Decision Transformer contains four environments under three dataset regimes each, yielding 12 environment–dataset pairs [11].

These examples show that favorable leaderboards can be found through a moderate random search over benchmark composition. A selectively composed benchmark can therefore match the size of a complete suite used in prior work while supporting a different conclusion about which algorithm performs best.

Random subsets nevertheless become more reliable as they grow. Figure 10 shows that their mean agreement with the full-suite leaderboard increases with subset size, while variability across subsets decreases. We show in Figure 11 that the probability of producing a conflicting winner to the full-suite leaderboard decreases as more benchmarks are added.

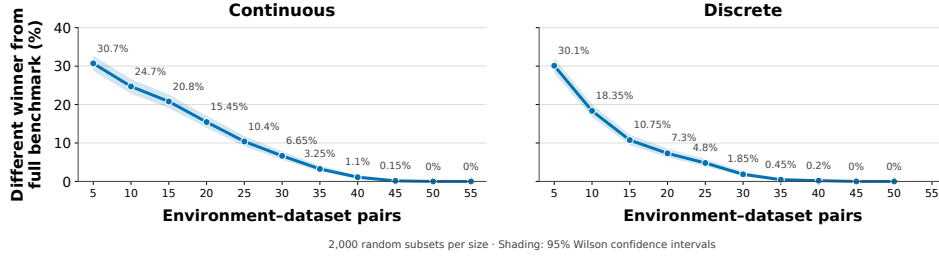


Figure 11: Probability of displaying a winner conflicting with the true top algorithm as the benchmark size increases.

Within our suite, evaluations therefore become less sensitive to the particular pairs sampled as more pairs are included. Strongly discordant leaderboards become less likely to arise through random selection. Carefully selected subsets remain a concern, however: Figure 9 shows that benchmarks comparable in size to prior evaluation suites can still be composed to favor different algorithms.

Takeaway. Larger benchmark suites produce more stable algorithm rankings. In our continuous-control experiments, subsets of 10–15 environment–dataset pairs, comparable to prior evaluations, yield a different winner from the full benchmark under both metrics roughly 20–25% of the time. Increasing the subset size to 25 pairs reduces this rate to about 10%.

4.2.2 Transfer Across Expert Sources

Prior work has found that results on agent-generated datasets do not necessarily transfer to human demonstrations [5]. We examine whether this is part of a broader pattern by comparing algorithm rankings across several sources of expert data.

We divide the 57 continuous-control datasets according to how their trajectories were generated:

- **Cloned (4 datasets).** Datasets generated wholly or partly by policies trained through behavior cloning.
- **Human (7 datasets).** Datasets collected through direct human control or teleoperation.
- **RL + Planner (6 datasets).** Datasets for which both a planner and a policy trained through reinforcement learning contribute to trajectory generation.
- **RL (32 datasets).** Datasets generated by expert policies trained through reinforcement learning.
- **Scripted (8 datasets).** Datasets generated by a predefined planner or scripted controller rather than a learned policy.

Figure 12 shows that algorithm rankings transfer unevenly across expert sources. Pairwise Kendall’s τ ranges from -0.16 to 0.60 under optimality gap and from -0.07 to 0.47 under IQM. Most pairs exhibit little or negative agreement. In particular, poor transfer is not confined to comparisons between human and machine-generated data; it also occurs between different methods of generating trajectories computationally.

Averaged over the ten source pairs, Kendall’s τ is 0.164 under optimality gap and 0.173 under IQM. Randomly constructed groups with the same sizes reach approximately 0.26 under both metrics. Rankings transfer unusually poorly, even compared to sampling random datasets to predict the ranking.

Group size explains part of the variation between sources. The raw averages make RL appear more transferable, but this category contains 32 datasets, compared with only 4–8 for each remaining source. After controlling for the log number of datasets, the apparent advantage of RL largely disappears, and every source has lower mean agreement than the random baseline. No single expert source consequently produces a ranking that is broadly representative of the others.

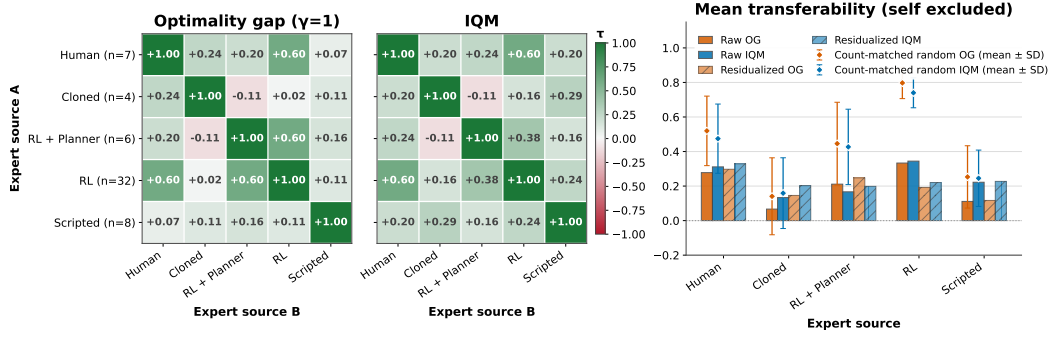


Figure 12: Agreement between algorithm rankings across expert sources. The left and center panels report pairwise Kendall’s τ under optimality gap and IQM. The right panel reports mean off-diagonal agreement before and after controlling for the log number of datasets in each source. The diamond with whiskers represents the count-matched random baseline, with whiskers representing 1 standard deviation.

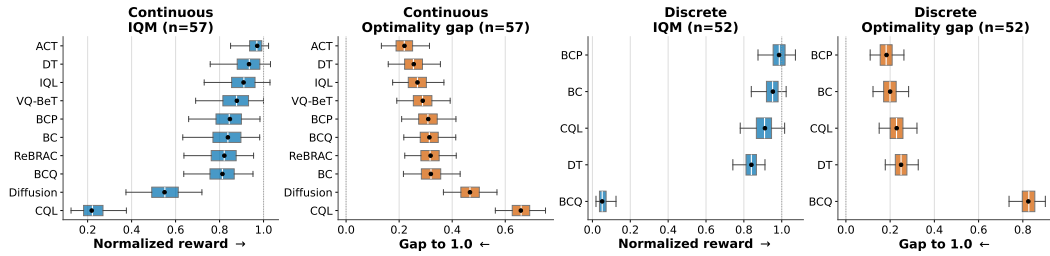


Figure 13: Overall algorithm performance on continuous- and discrete-control datasets. We report interquartile mean (IQM; higher is better) and optimality gap with threshold $\gamma = 1$ (lower is better). Markers denote point estimates, while boxes and whiskers summarize bootstrap uncertainty. ACT leads the continuous-control aggregate, whereas BCP leads the discrete-control aggregate under both metrics.

These results extend the distinction previously observed between human and agent-generated demonstrations. Changes in algorithm ordering seem to occur not only between human and machine-generated data, but also between cloned, scripted, planned, and RL-generated datasets.

Takeaway. Algorithm rankings transfer poorly across expert sources and agree less than rankings obtained from size-matched random groups. This failure extends beyond human-versus-machine comparisons to different sources of machine-generated data.

4.3 Leaderboard and evaluations

We next ask which algorithms perform well across the suite and where their relative strengths change. Figure 13 presents the overall continuous- and discrete-control leaderboards, while Figures 14 and 15 separate continuous-control results by environment collection and expert source. Full per-dataset scores are reported in Tables 6 and 7.

ACT is the strongest algorithm in the overall continuous-control aggregate, ranking first under both IQM and optimality gap. Its advantage is broad rather than universal: ACT also leads on AntMaze, Kitchen, human demonstrations, and RL + planner data, but different methods take the lead elsewhere. BCQ performs best on PointMaze and scripted datasets, DT leads on Adroit and cloned data, and IQL obtains the highest IQM on MuJoCo and RL-generated data. The aggregate leaderboard therefore provides a useful summary, but it obscures substantial changes in algorithm ordering across evaluation regimes.

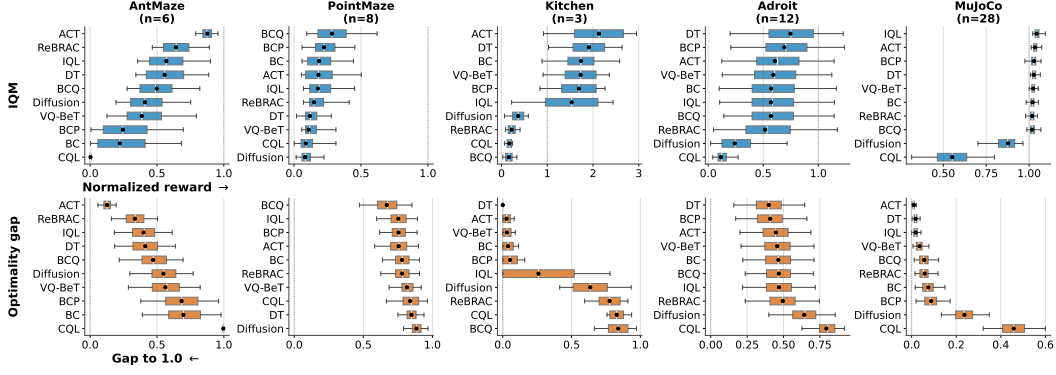


Figure 14: Continuous-control performance separated by environment collection. The leading algorithm changes across collections: ACT leads on AntMaze and Kitchen, BCQ on PointMaze, DT on Adroit, and IQL or ACT on MuJoCo depending on the aggregation metric.

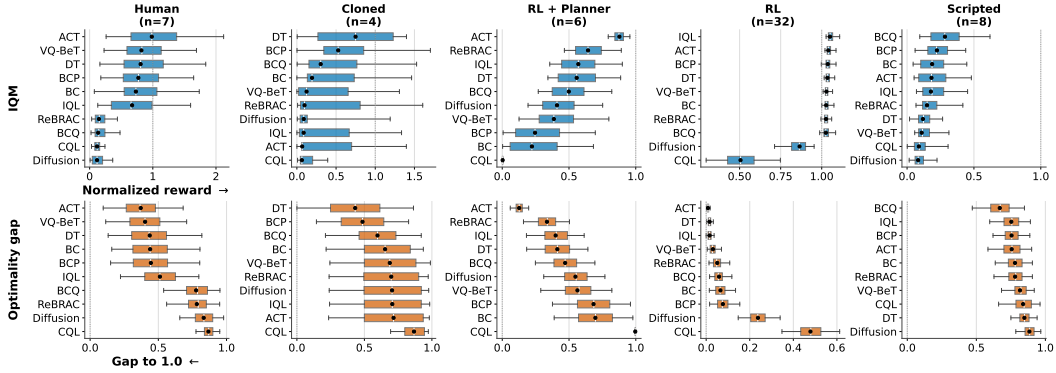


Figure 15: Continuous-control performance separated by the source of the trajectories in each dataset. The leading algorithm changes from ACT on human and RL + planner data, to DT on cloned data, IQL on RL data, and BCQ on scripted data.

The human-data results illustrate this distinction. ACT remains first and VQ-BeT rises to second under both metrics, consistent with VQ-BeT’s design for multimodal behavior generation from demonstration data [9]. DT follows, while BCP places fourth under IQM.

The simple cloning baselines are especially competitive. BCP uses a small MLP and differs from ordinary behavior cloning primarily through data filtering, yet it remains close to the leading group in several continuous-control settings. Its discrete-control performance is even stronger: BCP ranks first on Atari under both IQM and optimality gap, with unfiltered BC second. Prior studies such as CORL [13] often have behavior cloning at the bottom of their leaderboards, but in our continuous control suite they appear to be closer to the middle post-tuning. These are by far the cheapest and simplest methods we study, the performance here is thus quite impressive.

Diffusion and CQL produce the weakest continuous-control aggregates, although for different reasons. We evaluate the U-Net variant of Diffusion Policy as a standalone policy trained separately on each dataset [10]. It achieves competitive returns on many AntMaze, Adroit, and MuJoCo datasets but fails severely on others. These results should not be interpreted as evidence that diffusion-based action generation is flawed. In foundation VLAs such as π_0 , the related flow-matching objective is used to model multimodal action chunks within a much larger pretrained system [33]. Its action expert is also substantially smaller than the VLM backbone, and cached representations avoid repeatedly evaluating the full model during generation. This differs considerably from training Diffusion Policy from scratch as the complete policy. Action chunking itself is unlikely to explain our results because other chunked policies perform strongly under the same evaluation pipeline. Finally, extending training from 100 to 300 epochs produced little additional reward at substantially greater cost.

CQL exhibits a sharp domain reversal. It ranks last in the overall continuous-control comparison but third on Atari, ahead of DT and far ahead of BCQ. Because the original CQL results were obtained on a different set of datasets [12], we checked whether this discrepancy could be explained by our implementation. We adapted the original repository to load our Minari datasets and evaluated it on our suite of environments and datasets, represented in Figure 1, excluding PointMaze because the original work reports neither results nor tuning guidance for that collection. Following the procedure used by CORL, we applied the authors’ tuning recommendations when available and otherwise used the hyperparameters reported for the most similar dataset. When these configurations failed, we additionally evaluated a small set of existing published configurations. The original implementation produced performance comparable to our tuned models across the overlapping settings, making a simple implementation failure an unlikely explanation for CQL’s continuous-control performance. Prior reproduction and benchmarking efforts similarly find that CQL is highly sensitive to hyperparameters, low-level implementation choices, and the distribution of the offline data [13, 1]. Further implementation and tuning details are provided in Appendix B.

Finally, the value-function-based methods do not behave as a single coherent group. IQL is strongest on RL-generated and MuJoCo data, ReBRAC is competitive on RL + planner data, and BCQ leads on scripted and PointMaze datasets. Their relative performance changes sharply elsewhere: BCQ performs poorly on Atari despite leading PointMaze, while IQL and ReBRAC are much less competitive on human demonstrations.

Takeaway. ACT is the strongest overall continuous-control method, while the simple BCP baseline leads Atari. No algorithm dominates every data regime: environment collection and expert source substantially change which method performs best, particularly beyond the top position.

5 Hyperparameter Sensitivity and Transfer

The previous section showed that tuning budget can materially change algorithm rankings. We now ask which hyperparameters drive those changes, whether the same hyperparameters matter across environments, and how a broad sweep can be distilled into a strong default configuration. We examine sensitivity first and then turn to transfer.

5.1 Hyperparameter sensitivities

Figures 16 and 17 visualize how sensitivity is distributed across algorithms and environments. For each algorithm–environment pair, we measure the marginal association between hyperparameter h and trial performance using Spearman’s rank correlation ρ_h . Because the direction of the relationship is secondary to whether a hyperparameter affects performance, we take the absolute correlation and normalize it within the environment:

$$s_h = \frac{|\rho_h|}{\sum_j |\rho_j|}.$$

The resulting value measures each hyperparameter’s share of the total marginal rank association within an environment. We report both the median $|\rho_h|$ and the median normalized share in Tables 8 and 9.

We find that most algorithms are primarily sensitive to one to three hyperparameters, but their identities often change across datasets. Although learning rate is the most important hyperparameter for ACT and VQ-BeT in aggregate, action horizon dominates ACT on kitchen/complete, humanoid/simple, humanoid/medium, pointmaze/medium-dense, and door/cloned. For VQ-BeT, it dominates on both inverteddoublependulum datasets, invertedpendulum/expert, and antmaze/medium-diverse. Even rarely dominant parameters can matter locally: VQ-BeT’s commitment weight leads on hammer/human, while its secondary-code weight leads on pen/human. Full per-environment heatmaps are provided in Figure 25 of Appendix C.

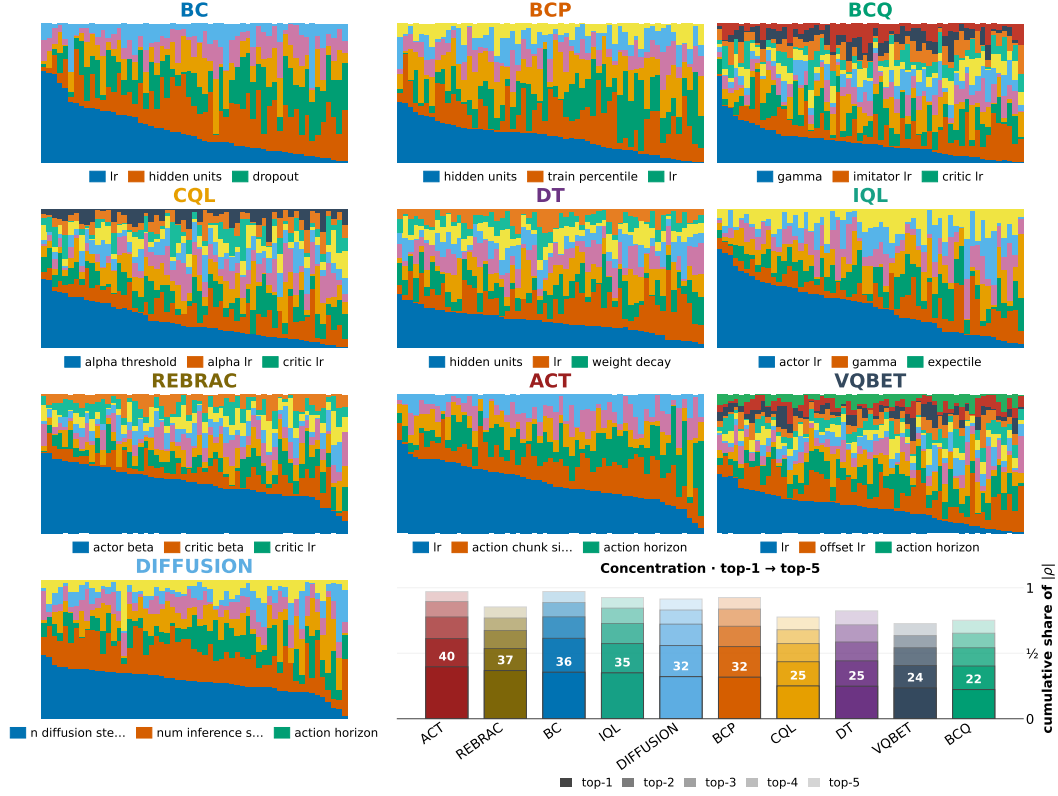


Figure 16: Hyperparameter sensitivity across the 57 continuous-control environments. For each algorithm–environment pair, we compute the absolute Spearman correlation $|\rho_h|$ between each hyperparameter and trial score and normalize these magnitudes as $s_h = |\rho_h| / \sum_j |\rho_j|$. Each vertical bar represents one environment and shows the resulting sensitivity shares. The bottom panel reports the median cumulative share of the one through five most influential hyperparameters within each environment; the displayed value is the median top-one share. Environment ordering is chosen separately within each algorithm for readability.

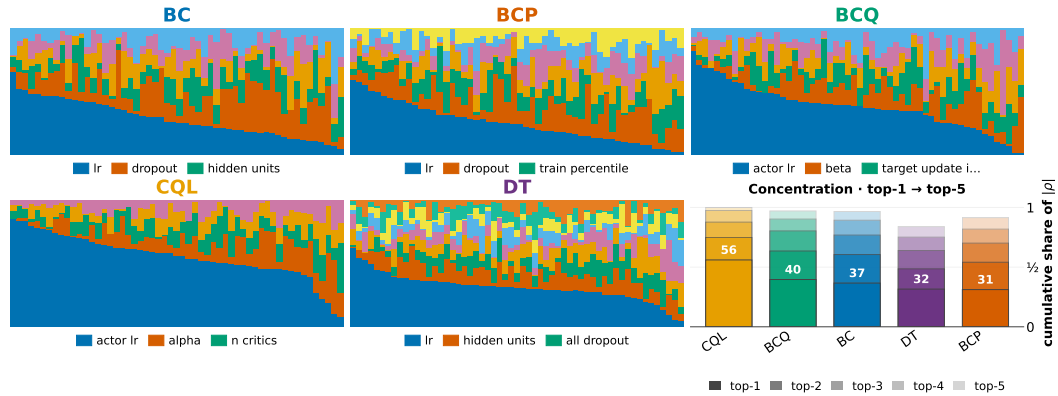


Figure 17: Hyperparameter sensitivity across the 54 Atari environments, computed as in Figure 16. The sensitivity profiles are more consistent than in continuous control: learning rate is the most influential hyperparameter for every algorithm, with particularly strong concentration for CQL and BCQ.

Sensitivity remains concentrated even after accounting for differences in search-space size. For VQ-BeT, the three most important hyperparameters within an environment account for a median of 74.5% of total sensitivity, compared with 25% under a uniform allocation across its 12 tuned parameters. The corresponding values are 70.9% versus 27.3% for BCQ and 85.8% versus 33.3% for ReBRAC. By comparison, ACT’s top-three share is higher in absolute terms at 88.7%, but its six-parameter search space gives a uniform reference of 50%.

The dominant hyperparameter is not always the one conventionally emphasized. Learning rate ranks first for every discrete algorithm, but the continuous-control results are more varied: γ leads for BCQ, hidden-unit count for DT, actor regularization β for ReBRAC, the conservative-loss threshold for CQL, and the number of diffusion steps for Diffusion Policy. Consequently, tuning only learning rate and a few standard optimization parameters can miss the principal source of variation for several algorithms.

We additionally investigate whether the sensitivities are driven by suboptimal (too wide) ranges, resulting in divergent runs. We trimmed 10% and 25% of the runs at the boundaries, and find that the original top hyperparameter remains completely unchanged. The same leading hyperparameter remained ranked first for all algorithms under both trims.

Takeaway. Sensitivity is usually concentrated, but its identity is not always transferable. In continuous control, the top three hyperparameters capture 54%–78% of total absolute rank association, yet the dominant parameters often change across environments.

5.2 Better Default Hyperparameters

Given the cost of running 25–50 tuning trials, we attempt to reduce the burden on practitioners by deriving strong default configurations for existing algorithms. Algorithm libraries commonly provide a single default configuration, but these defaults can leave substantial performance on the table. We therefore ask whether sweeps already completed by an algorithm’s developers can be distilled into data-driven defaults that transfer to previously unseen environments without target-specific tuning.

Deriving improved defaults. Our heuristic is motivated by the hypothesis that, although the best hyperparameter value may vary across environments, good values occupy broad, overlapping regions of the search space. Values near the center of high-performing configurations may therefore transfer more reliably than either a library default or the best configuration from a single environment.

For each algorithm and training environment, we retain the top 10% of trials by score. We pool these trials across environments and select each numerical hyperparameter by its median, computing the median in log-space for log-scaled parameters. For categorical parameters, we use the mode. Restricting the calculation to high-performing trials avoids clearly poor regions, while the median reduces the influence of noisy trials and environment-specific extremes. This procedure produces one fixed configuration per algorithm that can directly replace its existing library default.

Do the improved defaults work? We evaluate BC, BCQ, CQL, DT, IQL, and ReBRAC against their existing `d3rlpy` defaults [35]. The evaluation uses five folds containing five held-out environments each. For every fold, its test environments are excluded from the construction of the transferred configuration.

As shown in Figure 18, our defaults increase IQM from 0.47 to 0.70 and reduce optimality gap from 0.55 to 0.41. Fully tuned JumpStart results reach an IQM of 0.93 and an optimality gap of 0.27. Median transfer therefore recovers roughly half of the improvement obtained through full target-specific tuning without evaluating any configuration on the target environment. Recovering the remaining improvement without target-specific search is difficult because both the identities of the dominant hyperparameters and their best values vary across environments, preventing a single coordinate-wise default from reproducing each target’s specialized joint configuration.

These aggregate results establish that the complete configurations transfer, but do not explain why. We next test whether their individual hyperparameters fall within regions associated with strong performance on unseen environments.

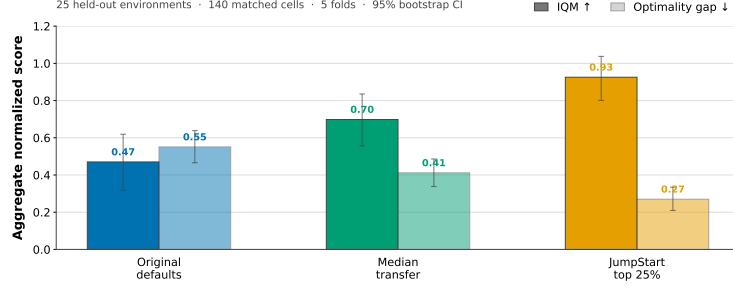


Figure 18: Performance of existing d3r1py defaults, median-transfer defaults, and fully tuned JumpStart top-quartile results. The evaluation contains five folds of five held-out environments, yielding 25 environments and 140 matched algorithm–environment cells. Each fold’s test environments are excluded when constructing its transferred defaults. Error bars are 95% bootstrap confidence intervals.

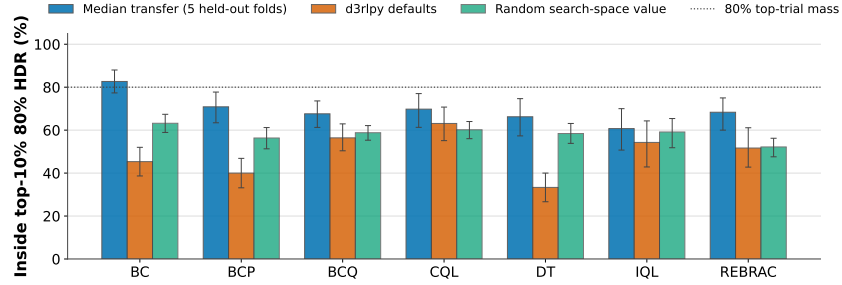


Figure 19: Marginal support for candidate defaults across five folds. Within each fold, its five final test environments are excluded, and each remaining environment is held out in turn. A value is supported when it lies inside the 80% highest-density region (HDR) of hyperparameter values among the target environment’s top 25% of trials. We compare our median-transfer defaults with the existing d3r1py defaults and the exact coverage of a value drawn from the configured search distribution. The dotted line denotes the 80% probability mass contained by the HDR by construction, not an expected transfer rate. Error bars are 95% environment-bootstrap confidence intervals.

Why do the defaults transfer? Within each fold, we exclude its five final test environments and hold out each remaining training environment in turn. For every algorithm, target environment, and hyperparameter, we recompute the transferred default using only the other training environments. We then estimate the distribution $\hat{p}_{e,h}$ of good values on the target from its top 25% of trials, using a boundary-corrected kernel density estimate for numerical parameters and a lightly smoothed empirical probability mass function for categorical parameters.

For numerical parameters, we evaluate $\hat{p}_{e,h}$ on a dense grid spanning the configured search bounds and normalize it using numerical integration. We then lower a density threshold until the grid intervals above that threshold contain 80% of the estimated probability mass. Boundary correction is applied when estimating the density, before this numerical integration. For categorical parameters, we analogously select the highest-probability categories containing at least 80% of the smoothed probability mass. We record whether each candidate default lies inside the resulting region. Random search-space coverage is computed directly from the probability assigned to the same region by the original tuning distribution.

Figure 19 shows that median-transfer coverage ranges from approximately 61% to 83% across algorithms, compared with 33%–63% for the existing defaults and 52%–63% for a random search-space value. The median-transfer default has the highest coverage for every evaluated algorithm. This supports the intuition behind the heuristic: high-performing configurations commonly occupy regions shared across environments rather than isolated, environment-specific optima.

Algorithm 1 Evaluation of each candidate recommender

```
1: for all recommender models  $m$  do
2:   for all environment families  $g$  do
3:     Hold out every dataset from family  $g$ 
4:     Tune  $m$  on the remaining datasets using three-fold grouped cross-validation
5:     Refit  $m$  on all remaining datasets
6:     Predict scores for the ten algorithms on each held-out dataset
7:     Recommend the three algorithms with the highest predicted scores
8:   end for
9:   Evaluate all out-of-family recommendations
10: end for
```

This density analysis is marginal and does not capture interactions between hyperparameters. Nevertheless, its agreement with the held-out evaluation suggests that the full configurations transfer because their individual values repeatedly fall within broad regions associated with strong performance. The transferred configurations should therefore be interpreted as strong starting points rather than replacements for full tuning.

Takeaway. Taking the coordinate-wise median of high-performing configurations from prior environments produces strong transferable defaults. Across 25 held-out environments, these defaults raise IQM from 0.47 to 0.70 and reduce optimality gap from 0.55 to 0.41, recovering roughly half of the benefit of full tuning without target-specific search.

6 Recommendation System for Practitioners

The preceding analyses show that the best algorithm depends on the dataset. We therefore train a dataset-conditioned recommender that uses observable properties of a continuous-action dataset and its environment to rank ten offline policy-learning algorithms and return the three most promising.

Prediction task. For each algorithm–dataset pair, the target is the normalized mean performance of the trials in the top 25%,

$$y_{a,d} = \frac{\bar{R}_{a,d}^{\text{top-25\%}} - R_d^{\text{random}}}{R_d^{\text{dataset}} - R_d^{\text{random}}},$$

where R_d^{dataset} is the mean trajectory return in the dataset. Each dataset is one training example, with target vector $\mathbf{y}_d = (y_{1,d}, \dots, y_{10,d})$. The recommender predicts this vector and returns the three algorithms with the highest predicted scores.

Features include dataset size, episode statistics, observation and action dimensionality, task domain, expert source, reward type, and trajectory-return statistics. We exclude collection names, dataset identities, exact data-generation algorithms, and benchmark results. Missing categorical values are represented explicitly as unknown, while numerical values are imputed within each training fold.

Candidate regressors. We evaluate Ridge, Elastic Net, partial least squares, k -nearest neighbors, RBF-SVR, random forest, histogram gradient boosting, and ExtraTrees. These models span linear, low-rank, local, kernel, and tree-based approaches suited to our small tabular dataset; we omit neural predictors because only 57 dataset-level examples are available.

Evaluation protocol. We evaluate on 57 datasets from 30 environment families using leave-one-environment-family-out validation, as summarized in Algorithm 1. All variants of the target environment are removed together. Model selection, preprocessing, and hyperparameter tuning occur entirely within the outer training split, so the held-out family is used only for final evaluation.

We compare against a fold-local global-mean baseline, which recommends the three algorithms with the highest mean performance on the outer training datasets regardless of target characteristics. This represents the common strategy of selecting whichever algorithms are strongest overall.

We report top-three overlap and best-of-three regret. Top-three overlap is the fraction of the true top three recovered by the recommendation. Best-of-three regret measures the score lost after

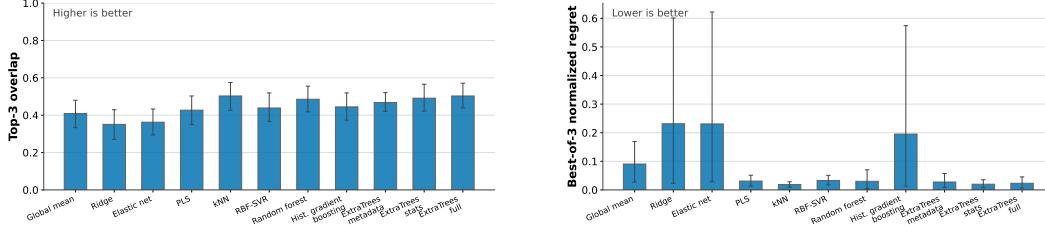


Figure 20: Algorithm-recommendation performance under leave-one-environment-family-out validation. The global-mean baseline recommends the fold-local leaderboard to every target, while the learned models condition on observable dataset features and are tuned entirely within each outer training split. Higher top-three overlap and lower best-of-three regret are better. Error bars show 95% environment-clustered bootstrap confidence intervals.

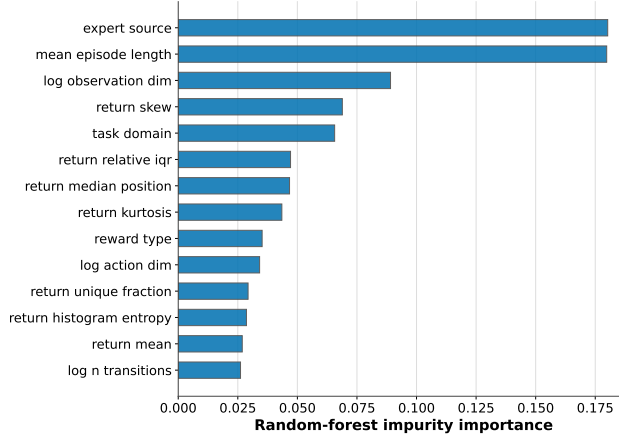


Figure 21: Impurity-based feature importance for the random-forest recommender. Expert source and mean episode length provide the strongest predictive signal, followed by observation dimensionality, task domain, and properties of the return distribution.

trying all three recommendations,

$$\text{regret}_d = \max_a y_{a,d} - \max_{a \in \hat{\mathcal{A}}_d^{(3)}} y_{a,d},$$

where $\hat{\mathcal{A}}_d^{(3)}$ is the recommended set. Regret is low whenever at least one recommended algorithm performs close to the true winner, even if the complete ordering is imperfect.

Main recommendation results. Figure 20 shows that both random forest and ExtraTrees improve substantially over the global leaderboard. Random forest increases top-three overlap from 0.409 to 0.485, includes the true winner among its recommendations on 63.2% rather than 47.4% of datasets, and reduces best-of-three regret from 0.091 to 0.030. It also improves NDCG@3 from 0.769 to 0.847 and Kendall’s τ from 0.324 to 0.401.

ExtraTrees obtains slightly stronger point estimates on the main evaluation, with 0.503 overlap and 0.023 regret, but the two tree models are statistically indistinguishable. Its overlap advantage over random forest is 0.018, with a paired 95% confidence interval of $[-0.035, 0.077]$, and its regret advantage is similarly uncertain. Relative to the global baseline (ranked by mean performance across all training datasets), random forest improves overlap by 0.076 and reduces regret by 0.061; the paired confidence interval for the regret reduction is $[0.011, 0.130]$.

What information supports the recommendations? Figure 21 shows that expert source and mean episode length are the most informative features, followed by observation dimensionality, return skew, and task domain. Several additional return-distribution statistics provide secondary signal, whereas dataset size contributes relatively little. Because expert source is the strongest

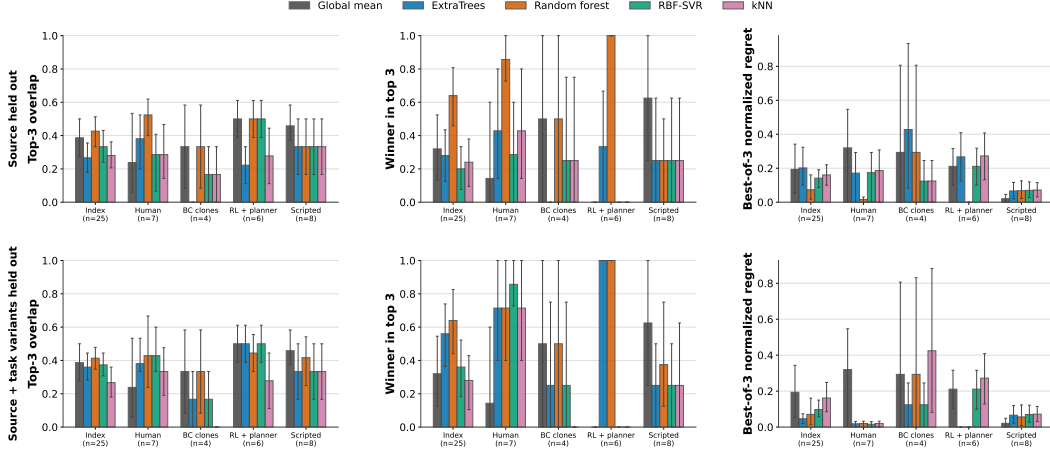


Figure 22: Transfer to minority expert-source categories. The top row removes each target source from training; the bottom row additionally removes all training datasets from environment families represented in the corresponding test set. “Index” aggregates the 25 datasets across Human, BC-clone, RL-plus-planner, and Scripted sources. Error bars show 95% environment-clustered bootstrap confidence intervals.

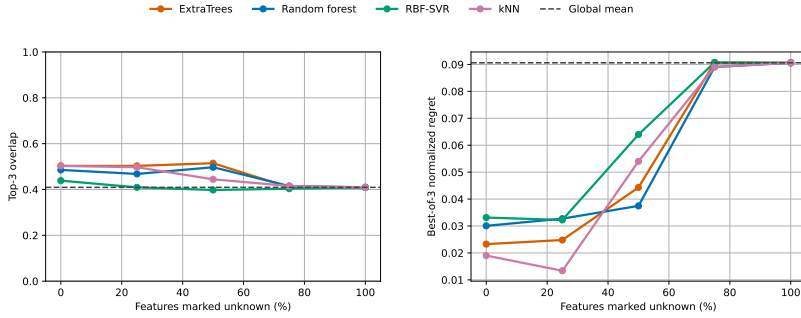


Figure 23: Recommendation quality as test features are randomly marked unknown. The tree models remain effective with half of the features missing. When fewer than 35% are available, the recommender explicitly falls back to the fold-local global leaderboard, causing all methods to converge to the baseline at high missingness.

feature, the source-held-out evaluation below is an important test of whether the recommender can generalize when this information has not been observed during training.

Transfer to unseen expert sources. We stress-test the recommender on four minority source categories: Human demonstrations, BC clones, RL-plus-planner data, and Scripted data. Each test dataset is first predicted using a model trained without any datasets from its source category. As shown in Figure 22, random forest obtains 0.427 aggregate top-three overlap, includes the winner on 64.0% of datasets, and achieves regret of 0.073. ExtraTrees obtains 0.267 overlap, 28.0% winner coverage, and 0.202 regret, while the global baseline obtains 0.387, 32.0%, and 0.194, respectively.

In the stricter test, we also remove every training dataset from environment families represented in the test set. Random forest retains 0.413 overlap and 64.0% winner coverage, compared with 0.360 and 56.0% for ExtraTrees.

We do not perform a corresponding source-held-out test for RL-generated data. These datasets account for 32 of the 57 datasets, contain all locomotion coverage, and supply much of the observed variation in trajectory and action dimensionality. Removing them would therefore confound expert-source shift with task-domain and feature-support shifts. Generalization among RL-generated datasets remains covered by the main environment-family holdout.

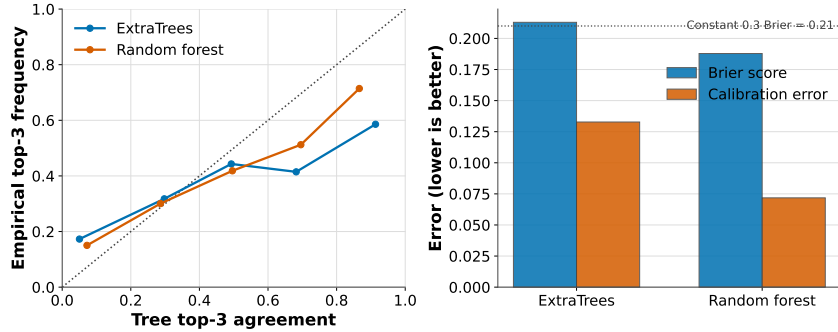


Figure 24: Reliability of ensemble agreement as a recommendation-confidence measure. Left: empirical top-three frequency against the fraction of trees placing an algorithm in the top three. Right: Brier score and calibration error. Random forest is better calibrated.

Incomplete dataset information. Figure 23 shows that the tree recommenders remain stable under moderate feature masking. With 50% of features unknown, random forest obtains 0.497 top-three overlap and 0.037 regret, compared with 0.515 and 0.044 for ExtraTrees. At higher missingness, both models converge to the explicit global-leaderboard fallback rather than making poorly supported predictions.

Interpreting confidence. Tree agreement provides a measure of how stable a recommendation is across the ensemble. Random forest produces better-calibrated agreement scores than ExtraTrees, with a Brier score of 0.188 rather than 0.213 and expected calibration error of 0.072 rather than 0.133.

We select random forest as the final recommender because it appears to be the most robust, while providing strong predictive performance.

7 Takeaways and Discussion

Reliable evaluation and comparison have been longstanding challenges across machine learning. A benchmark should do more than produce a single ordering of methods: it should reveal where each method succeeds or fails, and how strongly its conclusions depend on tuning, randomness, and benchmark composition. We distill our findings into practical guidance for developing, evaluating, and reporting offline policy-learning methods.

Pilot experiments. Early in a project, researchers commonly compare a proposed method with a small set of baselines to decide whether further investment is justified. Such pilots are only informative when the baselines are competitive; otherwise, an apparent improvement may reflect weak configurations rather than a better method. Figure 1 shows that tuning can substantially improve published baselines and change their relative ordering. JumpStart provides a consistent set of strong, downloadable baselines and their configurations, allowing new methods to be compared against competitive reference points without first reproducing a large tuning sweep. We hope this makes pilot experiments more informative and allows unpromising directions to be identified earlier.

Hyperparameters. Hyperparameters should be treated as part of a method’s evaluation rather than as incidental implementation details. Studies should report their search spaces, search procedure, tuning budget, and model-selection rule, along with the hyperparameters to which performance is most sensitive. Repeating a fixed configuration across seeds answers a different question from exploring additional configurations and cannot compensate for an inadequately searched space. In our experiments, 25–50 trials provides a useful practical compromise, although continuous-control rankings remain sensitive to further tuning. When target-specific tuning is infeasible, the transferred defaults developed in Section 5.2 provide a stronger starting point than existing library defaults, but they should not be interpreted as a replacement for tuning when accurate comparisons are required.

Evaluation suites and methodology. Broader evaluation generally produces more stable conclusions, although size alone does not guarantee a representative benchmark. For context, the main Decision Transformer comparisons contain nine D4RL locomotion datasets and four Atari games [11]; d3rlpy evaluates 17 datasets [35]; and the unified evaluations of Kang et al. [1] and Tarasov et al. [13] cover 26 and 30 tasks or datasets, respectively. Our results show that subsets of only 10–12 environment–dataset pairs can produce conflicting winners, and that even similarly sized suites can support different conclusions depending on their composition. Researchers should therefore report the complete benchmark composition, use as broad a suite as practical, and disaggregate results across important dataset properties. In particular, rankings transfer poorly across expert sources, so evaluations containing human, scripted, planned, or RL-generated data should report these regimes separately rather than relying only on a pooled aggregate.

Evaluation protocols must also be consistent across methods. We recommend robust aggregate metrics such as IQM and optimality gap, accompanied by uncertainty estimates and per-dataset results [6]. Baseline scores should be obtained using the same environment versions, wrappers, normalization procedure, checkpoint-selection rule, and evaluation budget as the proposed method. Numbers should not be copied from prior papers unless these details are known to match, particularly when environment versions differ. By releasing trained policies, their scores and hyperparameters, and the corresponding training and evaluation code, JumpStart allows baselines to be reevaluated as protocols change rather than leaving their published scores permanently tied to one implementation.

Choosing an algorithm. Our results do not support a single universally best algorithm. Performance depends on observable properties of the dataset and environment, and rankings can change across task domains and expert sources. The practitioner recommender in Section 6 uses these properties to produce a shortlist of three promising algorithms rather than asserting a universal winner. This shortlist is intended to guide an initial pilot: when its recommendations are confident, it can reduce the number of methods that must be tested; when confidence is low or task information is limited, the global leaderboard remains a safer fallback.

Living benchmarks and leaderboards. Static leaderboards become outdated as algorithms, datasets, and evaluation practices change. We therefore release an open-source website and extensible result database that support external submissions and self-hosting. New algorithms and datasets can be added while retaining the policies, scores, hyperparameters, and evaluation provenance underlying each entry. Because we release every trained policy, existing results can also be reevaluated as environments, metrics, and evaluation practices evolve. This allows the benchmark to develop with the field while keeping its comparisons reproducible and auditable.

8 Conclusion

We presented JumpStart, an extensively tuned comparison of ten offline policy-learning algorithms across 114 datasets and more than 160,000 trained policies. Our results show that algorithm rankings depend substantially on hyperparameter tuning, benchmark composition, and dataset characteristics, limiting the conclusions that can be drawn from small or weakly tuned evaluations. We translate these findings into practical tuning guidance, stronger default configurations, and dataset-conditioned algorithm recommendations. By releasing every trained policy with its metadata and code through an extensible leaderboard, JumpStart can incorporate new results and reevaluate existing models as evaluation practices change, providing an evolving foundation for future offline policy-learning research.

Table 4: Hyperparameter search spaces for continuous-control algorithms. Fixed parameters are omitted.

Notation: U is uniform, LogU is log-uniform, Int is inclusive integer, and braces denote uniformly sampled choices.

BC		BCP		BCQ	
Parameter	Range	Parameter	Range	Parameter	Range
Learning rate	$\text{LogU}[10^{-5}, 10^{-2}]$	Learning rate	$\text{LogU}[10^{-5}, 10^{-2}]$	Actor LR	$\text{LogU}[10^{-5}, 10^{-2}]$
Weight decay	$\text{LogU}[10^{-4}, 10^{-1}]$	Weight decay	$\text{LogU}[10^{-4}, 10^{-1}]$	Critic LR	$\text{LogU}[10^{-5}, 10^{-2}]$
Gradient clip	$\text{U}[0.1, 2]$	Gradient clip	$\text{U}[0.1, 2]$	Imitator LR	$\text{LogU}[10^{-5}, 10^{-2}]$
Dropout	$\text{U}[0, 0.3]$	Dropout	$\text{U}[0, 0.3]$	Discount γ	$\text{LogU}[0.9, 0.999]$
Hidden units	$\{128, 256, 512\} \times 2$	Hidden units	$\{128, 256, 512\} \times 2$	Target τ	$\text{U}[0.001, 0.02]$
Policy type	{det., stoch.}	Train percentile	$\text{U}[0, 90]$	Critics	$\text{Int}[1, 4]$
		Policy type	{det., stoch.}	Actor-update interval	$\text{Int}[1, 4]$
				λ	$\text{U}[0.5, 0.95]$
				Action samples	$\text{Int}[20, 200]$
				RL start step	$\text{Int}[0, 10000]$
				β	$\text{U}[0.1, 0.9]$
CQL		DT		IQL	
Parameter	Range	Parameter	Range	Parameter	Range
Actor LR	$\text{LogU}[10^{-5}, 10^{-2}]$	Learning rate	$\text{LogU}[10^{-5}, 10^{-2}]$	Actor LR	$\text{LogU}[10^{-5}, 10^{-2}]$
Critic LR	$\text{LogU}[10^{-5}, 10^{-2}]$	Weight decay	$\text{LogU}[10^{-4}, 10^{-1}]$	Critic LR	$\text{LogU}[10^{-5}, 10^{-2}]$
Temperature LR	$\text{LogU}[10^{-5}, 10^{-2}]$	Gradient clip	$\text{U}[0.1, 2]$	Discount γ	$\text{LogU}[0.9, 0.999]$
Alpha LR	$\text{LogU}[10^{-5}, 10^{-2}]$	Context size	$\{10, 20, 30\}$	Target τ	$\text{U}[0.001, 0.02]$
Discount γ	$\text{LogU}[0.9, 0.999]$	Hidden units	$\{128, 256\} \times 3$	Critics	$\text{Int}[1, 4]$
Target τ	$\text{U}[0.001, 0.02]$	Dropout	$\text{U}[0, 0.3]$	Exptile	$\text{U}[0.5, 0.9]$
Conservative weight	$\text{U}[0.1, 10]$	Attention heads	$\{1, 2, 4\}$	Weight temperature	$\text{U}[1, 10]$
Alpha threshold	$\text{U}[-1, 20]$	Layers	$\text{Int}[2, 5]$		
Action samples	$\text{Int}[5, 20]$	Discount γ	$\text{LogU}[0.9, 0.999]$		
Critics	$\text{Int}[1, 4]$	Reward scale	$\{0, 1\}$		
ReBRAC		ACT		VQ-BeT	
Parameter	Range	Parameter	Range	Parameter	Range
Actor LR	$\text{LogU}[10^{-5}, 10^{-2}]$	Learning rate	$\text{LogU}[10^{-5}, 10^{-2}]$	Learning rate	$\text{LogU}[10^{-5}, 10^{-2}]$
Critic LR	$\text{LogU}[10^{-5}, 10^{-2}]$	KL weight	$\text{LogU}[1, 10]$	Offset LR	$\text{LogU}[10^{-5}, 10^{-2}]$
Actor β	$\text{LogU}[10^{-5}, 1]$	Action chunk size	$\{2, 8\}$	Action chunk size	$\{2, 8\}$
Critic β	$\text{LogU}[10^{-5}, 1]$	Action horizon	$\{1, 4\}$	Action horizon	$\{1, 4\}$
Discount γ	$\text{LogU}[0.9, 0.999]$	Weight decay	$\text{LogU}[10^{-6}, 10^{-2}]$	VQ-VAE step ratio	$\text{U}[0.1, 0.4]$
Target τ	$\text{U}[0.001, 0.02]$	Gradient clip	$\text{U}[0.1, 2]$	Commitment weight	$\text{LogU}[0.01, 1]$
Critics	$\text{Int}[1, 4]$			Focal γ	$\text{U}[0.5, 5]$
Smoothing σ	$\text{U}[0, 0.5]$			Offset-loss weight	$\text{U}[0.1, 5]$
Smoothing clip	$\text{U}[0, 1]$			Primary-code weight	$\text{U}[1, 10]$
				Secondary-code weight	$\text{U}[0.1, 3]$
				Weight decay	$\text{LogU}[10^{-6}, 10^{-2}]$
				Gradient clip	$\text{U}[0.1, 2]$
Diffusion Policy					
Parameter	Range				
Learning rate	$\text{LogU}[10^{-5}, 10^{-3}]$				
Diffusion steps	$\{50, 100\}$				
Inference steps	$\text{Int}[20, 100]$				
Action chunk size	$\{8, 16\}$				
Action horizon	$\{1, 4\}$				
Weight decay	$\text{LogU}[10^{-6}, 10^{-2}]$				
Gradient clip	$\text{U}[0.1, 2]$				

Table 5: Hyperparameter search spaces for discrete-control algorithms. Fixed parameters are omitted.

Notation: U is uniform, LogU is log-uniform, Int is inclusive integer, and braces denote uniformly sampled choices.

BC		BCP		BCQ	
Parameter	Range	Parameter	Range	Parameter	Range
Learning rate	$\text{LogU}[10^{-5}, 10^{-2}]$	Learning rate	$\text{LogU}[10^{-5}, 10^{-2}]$	Learning rate	$\text{LogU}[10^{-5}, 10^{-2}]$
Weight decay	$\text{LogU}[10^{-4}, 10^{-1}]$	Weight decay	$\text{LogU}[10^{-4}, 10^{-1}]$	Discount γ	$\text{LogU}[0.9, 0.999]$
Gradient clip	$\text{U}[0.1, 2]$	Gradient clip	$\text{U}[0.1, 2]$	Critics	$\text{Int}[1, 4]$
Dropout	$\text{U}[0, 0.3]$	Dropout	$\text{U}[0, 0.3]$	Target interval	$\text{Int}[2000, 20000]$
Hidden units	$\{128, 256, 512\} \times 2$	Hidden units	$\{128, 256, 512\} \times 2$	Action flexibility	$\text{U}[0.01, 0.1]$
Regularization	$\text{U}[0, 1]$	Train percentile	$\text{U}[0, 90]$	β	$\text{U}[0.1, 0.9]$
		Regularization	$\text{U}[0, 1]$		

CQL		DT	
Parameter	Range	Parameter	Range
Learning rate	$\text{LogU}[10^{-5}, 10^{-2}]$	Learning rate	$\text{LogU}[10^{-5}, 10^{-2}]$
Discount γ	$\text{LogU}[0.9, 0.999]$	Weight decay	$\text{LogU}[10^{-4}, 10^{-1}]$
Critics	$\text{Int}[1, 4]$	Gradient clip	$\text{U}[0.1, 2]$
Target interval	$\text{Int}[1000, 10000]$	Context size	$\{10, 20, 30\}$
α	$\text{U}[0.1, 2]$	Hidden units	$\{128, 256\} \times 3$
		Dropout	$\text{U}[0, 0.3]$
		Attention heads	$\{1, 2, 4\}$
		Layers	$\text{Int}[2, 5]$
		Discount γ	$\text{LogU}[0.9, 0.999]$
		Reward scale	$\{0, 1\}$

Table 6: Normalized scores on continuous-control datasets, grouped by benchmark collection. Entries report the point estimate $\pm$ half-width of its 95% confidence interval. Dataset rows summarize the top 25% of evaluated hyperparameter configurations; shaded rows report collection-level interquartile mean (IQM; $\uparrow$) and optimality gap ($\downarrow$). The best point estimate in each row is bold.

Dataset	BC	BCP	BCQ	CQL	DT	IQL	ReBRAC	ACT	VQ-BeT	Diff.
AntMaze										
6 datasets										
large-diverse	0.004±0.005	0.000±0.000	0.288±0.017	0.000±0.001	0.454±0.024	0.322±0.024	0.464±0.026	0.804±0.019	0.113±0.026	0.187±0.017
large-play	0.007±0.007	0.007±0.004	0.295±0.013	0.000±0.000	0.297±0.013	0.397±0.010	0.421±0.019	0.745±0.020	0.100±0.009	0.130±0.022
medium-diverse	0.000±0.000	0.020±0.013	0.245±0.015	0.000±0.000	0.449±0.034	0.367±0.015	0.610±0.033	0.849±0.023	0.192±0.030	0.327±0.017
medium-play	0.115±0.035	0.194±0.035	0.479±0.018	0.002±0.002	0.325±0.014	0.545±0.011	0.556±0.016	0.883±0.009	0.446±0.017	0.372±0.021
umaze-diverse	0.764±0.031	0.762±0.023	0.935±0.010	0.012±0.007	1.003±0.005	0.970±0.005	0.940±0.009	0.975±0.005	0.795±0.025	0.941±0.009
umaze	0.927±0.012	0.916±0.019	0.933±0.025	0.010±0.010	1.045±0.004	1.041±0.004	1.008±0.008	0.983±0.009	0.976±0.009	0.756±0.032
IQM ↑	0.222±0.340	0.246±0.345	0.499±0.272	0.003±0.004	0.558±0.272	0.570±0.272	0.643±0.213	0.878±0.083	0.387±0.334	0.410±0.279
Opt. gap ↓	0.697±0.294	0.684±0.291	0.471±0.238	0.996±0.004	0.412±0.227	0.400±0.216	0.335±0.172	0.127±0.068	0.563±0.268	0.548±0.237
Adroit										
12 datasets										
door-cloned	0.253±0.029	0.711±0.056	0.154±0.056	0.044±0.004	1.229±0.065	0.032±0.003	0.045±0.004	0.060±0.021	0.022±0.002	0.042±0.000
door-expert	1.036±0.000	1.044±0.001	1.049±0.002	0.198±0.093	1.045±0.000	1.040±0.000	1.036±0.002	1.037±0.001	1.034±0.001	0.837±0.022
door-human	0.075±0.018	0.094±0.018	0.013±0.005	0.050±0.021	0.041±0.010	0.199±0.010	0.018±0.000	0.521±0.067	0.290±0.035	0.011±0.001
hammer-cloned	0.134±0.011	0.344±0.043	0.454±0.099	0.084±0.025	0.268±0.018	0.141±0.018	0.150±0.051	0.075±0.012	0.225±0.028	0.133±0.016
hammer-expert	1.306±0.001	1.313±0.001	1.353±0.002	0.155±0.074	1.311±0.001	1.314±0.001	1.315±0.003	1.311±0.001	1.303±0.002	0.718±0.039
hammer-human	0.071±0.015	0.048±0.011	0.180±0.045	0.023±0.009	0.073±0.010	0.140±0.010	0.047±0.008	0.088±0.016	0.137±0.011	0.003±0.000
pen-cloned	1.463±0.022	1.703±0.023	1.527±0.022	0.393±0.048	1.398±0.021	1.335±0.018	1.606±0.041	1.397±0.016	1.308±0.015	1.190±0.024
pen-expert	2.420±0.014	2.544±0.016	2.592±0.020	2.180±0.079	2.429±0.013	2.536±0.015	2.534±0.025	2.398±0.014	2.396±0.020	1.941±0.030
pen-human	0.888±0.016	0.900±0.017	0.884±0.010	0.336±0.030	0.828±0.015	0.857±0.015	0.785±0.017	0.873±0.019	0.842±0.013	0.066±0.004
relocate-cloned	0.000±0.000	0.002±0.001	0.004±0.001	0.009±0.000	0.004±0.000	0.001±0.000	0.009±0.000	0.000±0.000	0.000±0.000	0.005±0.000
relocate-expert	1.049±0.002	1.055±0.002	0.697±0.024	0.166±0.045	1.038±0.002	1.042±0.002	1.040±0.007	1.039±0.001	1.023±0.002	0.475±0.046
relocate-human	0.000±0.000	-0.001±0.000	0.000±0.000	0.002±0.000	-0.001±0.000	0.001±0.000	0.002±0.000	0.004±0.001	0.000±0.000	-0.001±0.000
IQM ↑	0.573±0.532	0.691±0.516	0.570±0.501	0.116±0.115	0.747±0.514	0.570±0.521	0.517±0.563	0.605±0.510	0.592±0.497	0.241±0.347
Opt. gap ↓	0.465±0.244	0.409±0.245	0.468±0.234	0.795±0.146	0.399±0.244	0.469±0.250	0.495±0.255	0.448±0.243	0.457±0.250	0.643±0.228
Kitchen										
3 datasets										
complete	0.885±0.054	0.841±0.062	0.028±0.011	0.064±0.052	1.028±0.048	0.221±0.021	0.175±0.075	0.917±0.060	0.910±0.046	0.067±0.009
mixed	1.740±0.095	1.974±0.068	0.120±0.053	0.211±0.119	2.065±0.056	1.942±0.056	0.091±0.076	2.521±0.054	1.921±0.127	0.440±0.083
partial	2.581±0.079	2.262±0.104	0.334±0.083	0.242±0.154	2.634±0.106	2.436±0.063	0.405±0.172	2.949±0.081	2.357±0.096	0.586±0.069
IQM ↑	1.735±0.848	1.692±0.710	0.161±0.153	0.172±0.089	1.909±0.803	1.533±1.107	0.224±0.157	2.129±1.016	1.729±0.724	0.364±0.260
Opt. gap ↓	0.038±0.057	0.053±0.080	0.839±0.153	0.828±0.089	0.000±0.000	0.260±0.389	0.776±0.157	0.028±0.041	0.030±0.045	0.636±0.260
PointMaze										
8 datasets										
large-dense	0.104±0.011	0.202±0.019	0.156±0.026	0.125±0.021	0.156±0.030	0.079±0.018	0.116±0.023	0.111±0.007	0.082±0.009	0.004±0.011
large	0.024±0.003	0.022±0.002	0.020±0.002	0.003±0.002	0.023±0.003	0.025±0.002	0.022±0.004	0.024±0.002	0.055±0.001	0.131±0.030
medium-dense	0.109±0.017	0.215±0.025	0.252±0.032	0.235±0.023	0.140±0.028	0.112±0.018	0.126±0.026	0.083±0.018	0.056±0.011	0.051±0.011
medium	0.024±0.004	0.021±0.003	0.054±0.004	-0.003±0.005	0.014±0.003	0.065±0.005	0.033±0.005	0.077±0.015	0.058±0.005	0.053±0.006
open-dense	0.452±0.013	0.403±0.020	0.576±0.027	0.180±0.042	0.154±0.043	0.356±0.027	0.243±0.042	0.546±0.056	0.174±0.044	-0.045±0.024
open	0.095±0.012	0.073±0.012	0.151±0.013	-0.009±0.014	-0.006±0.008	0.166±0.011	0.112±0.012	0.043±0.013	0.123±0.003	0.095±0.006
umaze-diverse	0.441±0.026	0.504±0.013	0.770±0.049	0.711±0.051	0.375±0.055	0.548±0.024	0.587±0.049	0.459±0.006	0.359±0.027	0.257±0.015
umaze	0.510±0.033	0.517±0.036	0.664±0.003	0.045±0.041	0.358±0.033	0.622±0.006	0.514±0.039	0.609±0.003	0.561±0.024	0.377±0.015
IQM ↑	0.188±0.192	0.223±0.197	0.283±0.262	0.088±0.156	0.118±0.130	0.178±0.191	0.149±0.171	0.183±0.224	0.109±0.128	0.083±0.104
Opt. gap ↓	0.780±0.134	0.755±0.136	0.670±0.190	0.839±0.148	0.848±0.094	0.753±0.148	0.781±0.140	0.756±0.159	0.816±0.116	0.885±0.088
MuJoCo										
28 datasets										
ant-expert	1.005±0.002	1.003±0.002	0.997±0.002	0.319±0.079	1.005±0.001	1.012±0.001	0.992±0.003	1.008±0.002	0.974±0.015	0.929±0.019
ant-medium	1.042±0.004	1.029±0.005	1.035±0.004	0.413±0.068	1.060±0.003	1.069±0.002	0.989±0.013	1.057±0.002	0.935±0.042	0.497±0.066
ant-simple	1.100±0.004	1.094±0.006	1.153±0.008	0.464±0.087	1.109±0.003	1.100±0.003	1.079±0.012	1.070±0.005	1.034±0.014	0.739±0.038
halfcheetah-expert	0.960±0.016	0.862±0.038	0.910±0.034	0.040±0.010	1.005±0.007	1.051±0.001	0.893±0.051	1.062±0.002	1.073±0.002	0.347±0.020
halfcheetah-medium	1.187±0.012	1.144±0.025	0.954±0.021	0.074±0.034	1.098±0.011	1.259±0.002	0.951±0.043	1.161±0.018	1.203±0.030	0.197±0.015
halfcheetah-simple	1.040±0.001	1.040±0.001	1.049±0.003	0.348±0.107	1.033±0.001	1.045±0.000	1.048±0.005	1.044±0.001	1.035±0.001	0.968±0.006
hopper-expert	0.935±0.016	0.816±0.025	0.644±0.034	0.525±0.034	0.831±0.016	1.070±0.001	0.776±0.036	0.950±0.011	1.045±0.003	0.810±0.025
hopper-medium	1.249±0.007	1.255±0.005	1.226±0.017	1.238±0.010	1.199±0.015	1.267±0.001	1.046±0.063	1.269±0.001	1.243±0.007	1.068±0.024
hopper-simple	1.935±0.002	1.940±0.002	1.819±0.057	1.215±0.188	1.929±0.001	1.936±0.001	1.921±0.008	1.927±0.002	1.860±0.047	1.154±0.063
humanoid-expert	1.149±0.012	1.120±0.016	1.004±0.056	0.030±0.008	1.229±0.002	1.242±0.001	1.185±0.011	1.089±0.014	1.056±0.022	0.237±0.029
humanoid-medium	0.135±0.017	0.110±0.020	0.117±0.007	0.059±0.009	0.138±0.005	1.155±0.003	1.084±0.014	1.149±0.002	1.019±0.056	0.414±0.064
humanoid-simple	0.574±0.024	0.497±0.032	0.803±0.020	0.126±0.034	0.895±0.007	0.930±0.008	0.774±0.024	1.018±0.005	0.764±0.067	0.602±0.038
humanoidstandup-expert	0.191±0.008	0.207±0.014	0.273±0.020	0.107±0.007	0.825±0.014	0.785±0.023	0.268±0.014	0.956±0.008	0.499±0.046	0.678±0.035
humanoidstandup-medium	0.495±0.042	0.316±0.031	0.887±0.022	0.192±0.020	0.977±0.004	1.008±0.002	0.792±0.034	0.958±0.005	0.955±0.028	0.948±0.036
humanoidstandup-simple	0.872±0.011	0.875±0.015	0.956±0.008	0.823±0.028	0.994±0.003	1.007±0.003	0.952±0.011	0.988±0.004	0.970±0.005	0.916±0.009
inverteddoublependulum-expert	1.000±0.000	1.000±0.000	1.000±0.000	0.591±0.129	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.001	0.079±0.009
inverteddoublependulum-medium	0.875±0.021	1.241±0.000	1.235±0.002	0.912±0.054	0.977±0.016	0.778±0.023	1.086±0.039	0.870±0.026	1.014±0.020	0.305±0.042
inverteddoublependulum-expert	1.000±0.000	1.000±0.000	1.000±0.000	0.036±0.035	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.990±0.000	0.962±0.008
invertedpendulum-expert	1.587±0.008	1.689±0.018	1.130±0.000	1.130±0.000	1.457±0.042	1.130±0.000	1.130±0.000	1.504±0.018	1.181±0.027	0.922±0.037
invertedpendulum-medium	1.015±0.001	1.009±0.002	1.003±0.001	0.847±0.006	1.019±0.001	1.024±0.000	1.009±0.003	1.023±0.000	1.022±0.001	0.968±0.006
pusher-medium	1.023±0.001	1.023±0.001	1.023±0.001	0.969±0.006	1.025±0.001	1.024±0.001	1.017±0.001	1.024±0.001	1.020±0.001	0.987±0.003
reacher-expert	1.029±0.001	1.028±0.001	1.029±0.001	1.029±0.001	1.026±0.001	1.032±0.001	1.019±0.001	1.030±0.000	1.028±0.001	1.006±0.001
reacher-medium	1.029±0.001	1.028±0.001	1.040±0.001	1.036±0.001	1.036±0.001	1.034±0.001	1.034±0.001	1.030±0.001	1.027±0.001	1.000±0.001
swimmer-expert	1.025±0.000	1.025±0.000	1.020±0.001	0.835±0.067	1.015±0.001	1.024±0.000	0.823±0.000	1.022±0.000	1.022±0.000	1.000±0.000
swimmer-medium	0.999±0.003	1.094±0.005	1.187±0.017	1.198±0.019	1.059±0.003	1.131±0.006	1.121±0.016	1.170±0.012	1.140±0.008	0.171±0.001
walker2d-expert	1.007±0.002	0.973±0.011	0.986±0.004	0.369±0.071	1.001±0.002	1.015±0.000	0.983±0.010	1.002±0.001	1.007±0.003	0.868±0.023
walker2d-medium	1.070±0.002	1.075±0.002	1.018±0.006	0.180±0.062	1.064±0.002	1.082±0.001	1.060±0.003	1.077±0.002	1.071±0.003	0.999±0.006
walker2d-simple	1.031±0.001	1.033±0.001	1.035±0.001	0.908±0.036	1.035±0.002	1.029±0.001	1.032±0.002	1.019±0.001	1.019±0.001	0.990±0.005
IQM ↑	1.020±0.036	1.028±0.047	1.017±0.041	0.552±0.241	1.027±0.029	1.043±0.037	1.018±0.034	1.035±0.030	1.024±0.026	0.878±0.131
Opt. gap ↓	0.075±0.067	0.088±0.076	0.057±0.055	0.458±0.139	0.018±0.018	0.018±0.021	0.058±0.052	0.010±0.010	0.035±0.036	0.237±0.101

Table 7: Normalized scores on Atari environments. Entries report the point estimate $\pm$ half-width of its 95% confidence interval. Environment rows summarize the top 25% of evaluated hyperparameter configurations; shaded rows report the collection-level interquartile mean (IQM; $\uparrow$) and optimality gap ($\downarrow$). The best point estimate in each row is bold.

Environment	BC	BCP	BCQ	CQL	DT
Atari	<i>52 environments</i>				
Alien	1.201 $\pm$ 0.120	1.062 $\pm$ 0.132	0.261 $\pm$ 0.048	1.210 $\pm$ 0.173	0.828 $\pm$ 0.010
Amidar	0.938 $\pm$ 0.009	1.230 $\pm$ 0.036	0.010 $\pm$ 0.004	1.156 $\pm$ 0.054	0.962 $\pm$ 0.016
Assault	0.244 $\pm$ 0.006	0.229 $\pm$ 0.008	0.006 $\pm$ 0.001	0.165 $\pm$ 0.010	0.249 $\pm$ 0.005
Asterix	1.087 $\pm$ 0.107	2.580 $\pm$ 0.128	1.133 $\pm$ 0.080	1.335 $\pm$ 0.115	0.760 $\pm$ 0.049
Asteroids	0.763 $\pm$ 0.079	0.838 $\pm$ 0.065	0.271 $\pm$ 0.068	0.867 $\pm$ 0.075	1.075 $\pm$ 0.009
Atlantis	1.096 $\pm$ 0.001	1.101 $\pm$ 0.003	0.011 $\pm$ 0.002	1.093 $\pm$ 0.002	1.089 $\pm$ 0.001
Bank Heist	0.838 $\pm$ 0.053	1.309 $\pm$ 0.058	-0.009 $\pm$ 0.008	1.185 $\pm$ 0.044	1.077 $\pm$ 0.014
Battle Zone	1.087 $\pm$ 0.054	1.146 $\pm$ 0.084	0.123 $\pm$ 0.019	0.965 $\pm$ 0.065	0.798 $\pm$ 0.005
Beam Rider	0.152 $\pm$ 0.021	0.112 $\pm$ 0.021	0.026 $\pm$ 0.005	0.086 $\pm$ 0.006	0.157 $\pm$ 0.009
Berzerk	0.268 $\pm$ 0.025	0.243 $\pm$ 0.022	0.039 $\pm$ 0.015	0.476 $\pm$ 0.072	0.425 $\pm$ 0.007
Bowling	1.444 $\pm$ 0.025	1.631 $\pm$ 0.036	-0.113 $\pm$ 0.173	1.502 $\pm$ 0.052	1.059 $\pm$ 0.008
Boxing	1.049 $\pm$ 0.003	1.049 $\pm$ 0.003	0.018 $\pm$ 0.007	1.031 $\pm$ 0.010	0.960 $\pm$ 0.002
Breakout	0.180 $\pm$ 0.064	0.328 $\pm$ 0.107	0.002 $\pm$ 0.001	0.182 $\pm$ 0.099	0.315 $\pm$ 0.029
Centipede	1.013 $\pm$ 0.071	1.193 $\pm$ 0.237	0.455 $\pm$ 0.131	1.131 $\pm$ 0.109	0.713 $\pm$ 0.025
Chopper Command	1.046 $\pm$ 0.086	1.022 $\pm$ 0.087	0.003 $\pm$ 0.005	0.881 $\pm$ 0.123	0.668 $\pm$ 0.017
Crazy Climber	1.236 $\pm$ 0.036	1.234 $\pm$ 0.028	-0.064 $\pm$ 0.004	1.149 $\pm$ 0.029	1.097 $\pm$ 0.006
Defender	0.917 $\pm$ 0.025	0.856 $\pm$ 0.023	0.046 $\pm$ 0.014	0.813 $\pm$ 0.031	0.720 $\pm$ 0.008
Demon Attack	0.048 $\pm$ 0.006	0.083 $\pm$ 0.041	0.001 $\pm$ 0.000	0.315 $\pm$ 0.074	0.038 $\pm$ 0.002
Double Dunk	1.004 $\pm$ 0.005	1.004 $\pm$ 0.005	1.000 $\pm$ 0.000	1.017 $\pm$ 0.011	0.849 $\pm$ 0.013
Fishing Derby	1.019 $\pm$ 0.000	1.019 $\pm$ 0.000	0.043 $\pm$ 0.007	1.019 $\pm$ 0.000	0.902 $\pm$ 0.006
Freeway	1.013 $\pm$ 0.000	1.013 $\pm$ 0.000	0.677 $\pm$ 0.021	1.013 $\pm$ 0.000	0.992 $\pm$ 0.003
Frostbite	2.127 $\pm$ 0.495	2.629 $\pm$ 0.555	0.633 $\pm$ 0.023	1.034 $\pm$ 0.024	1.037 $\pm$ 0.010
Gopher	0.975 $\pm$ 0.097	1.064 $\pm$ 0.090	0.064 $\pm$ 0.054	1.221 $\pm$ 0.102	0.938 $\pm$ 0.009
Gravitar	1.287 $\pm$ 0.207	1.265 $\pm$ 0.186	0.372 $\pm$ 0.133	1.369 $\pm$ 0.217	0.940 $\pm$ 0.013
H.E.R.O.	1.137 $\pm$ 0.003	1.141 $\pm$ 0.001	0.021 $\pm$ 0.013	1.134 $\pm$ 0.002	1.030 $\pm$ 0.005
Ice Hockey	0.801 $\pm$ 0.021	0.674 $\pm$ 0.017	0.128 $\pm$ 0.061	0.748 $\pm$ 0.029	0.661 $\pm$ 0.008
James Bond	0.643 $\pm$ 0.133	0.472 $\pm$ 0.100	0.015 $\pm$ 0.004	0.529 $\pm$ 0.113	0.659 $\pm$ 0.019
Kangaroo	0.801 $\pm$ 0.048	0.814 $\pm$ 0.053	0.026 $\pm$ 0.008	0.456 $\pm$ 0.096	0.805 $\pm$ 0.015
Krull	1.197 $\pm$ 0.012	1.205 $\pm$ 0.012	0.239 $\pm$ 0.050	1.099 $\pm$ 0.017	1.025 $\pm$ 0.002
Kung-Fu Master	0.990 $\pm$ 0.039	1.097 $\pm$ 0.040	0.019 $\pm$ 0.006	1.127 $\pm$ 0.039	0.888 $\pm$ 0.005
Ms. Pac-Man	0.935 $\pm$ 0.068	0.943 $\pm$ 0.077	0.132 $\pm$ 0.028	1.159 $\pm$ 0.083	0.781 $\pm$ 0.012
Name This Game	1.235 $\pm$ 0.032	1.285 $\pm$ 0.046	0.096 $\pm$ 0.026	1.146 $\pm$ 0.047	0.898 $\pm$ 0.006
Phoenix	0.193 $\pm$ 0.015	0.157 $\pm$ 0.012	0.008 $\pm$ 0.004	0.170 $\pm$ 0.011	0.160 $\pm$ 0.006
Pitfall!	1.023 $\pm$ 0.000	1.023 $\pm$ 0.000	1.023 $\pm$ 0.000	1.023 $\pm$ 0.000	0.992 $\pm$ 0.004
Pong	1.873 $\pm$ 0.001	1.872 $\pm$ 0.000	-0.010 $\pm$ 0.009	0.305 $\pm$ 0.180	1.549 $\pm$ 0.036
Private Eye	1.000 $\pm$ 0.000	1.000 $\pm$ 0.000	5.669 $\pm$ 3.896	1.000 $\pm$ 0.000	2.505 $\pm$ 1.654
Q*bert	1.074 $\pm$ 0.018	1.074 $\pm$ 0.020	0.005 $\pm$ 0.002	0.984 $\pm$ 0.027	0.892 $\pm$ 0.005
River Raid	0.852 $\pm$ 0.018	0.835 $\pm$ 0.026	-0.003 $\pm$ 0.010	0.869 $\pm$ 0.017	0.822 $\pm$ 0.006
Road Runner	1.029 $\pm$ 0.025	0.936 $\pm$ 0.025	0.002 $\pm$ 0.003	0.846 $\pm$ 0.038	0.924 $\pm$ 0.005
Robot Tank	1.081 $\pm$ 0.028	0.983 $\pm$ 0.028	0.130 $\pm$ 0.036	0.729 $\pm$ 0.024	0.811 $\pm$ 0.013
Seaquest	1.028 $\pm$ 0.000	1.028 $\pm$ 0.000	0.085 $\pm$ 0.010	1.024 $\pm$ 0.002	1.014 $\pm$ 0.001
Space Invaders	1.042 $\pm$ 0.107	1.362 $\pm$ 0.095	0.330 $\pm$ 0.053	1.115 $\pm$ 0.080	0.894 $\pm$ 0.022
Star Gunner	0.233 $\pm$ 0.019	0.706 $\pm$ 0.131	0.005 $\pm$ 0.002	0.490 $\pm$ 0.081	0.327 $\pm$ 0.029
Surround	0.861 $\pm$ 0.020	0.881 $\pm$ 0.021	0.000 $\pm$ 0.001	0.564 $\pm$ 0.024	0.513 $\pm$ 0.022
Tennis	0.644 $\pm$ 0.003	0.642 $\pm$ 0.000	0.642 $\pm$ 0.000	0.643 $\pm$ 0.001	0.200 $\pm$ 0.061
Time Pilot	0.416 $\pm$ 0.034	0.418 $\pm$ 0.040	0.034 $\pm$ 0.007	0.231 $\pm$ 0.010	0.359 $\pm$ 0.011
Tutankham	0.515 $\pm$ 0.096	1.051 $\pm$ 0.107	-0.087 $\pm$ 0.050	0.626 $\pm$ 0.107	0.991 $\pm$ 0.028
Up 'n Down	0.981 $\pm$ 0.018	0.939 $\pm$ 0.023	0.006 $\pm$ 0.003	0.714 $\pm$ 0.034	0.717 $\pm$ 0.014
Video Pinball	0.154 $\pm$ 0.097	0.136 $\pm$ 0.081	-0.039 $\pm$ 0.002	-0.012 $\pm$ 0.016	0.189 $\pm$ 0.017
Wizard of Wor	3.104 $\pm$ 1.238	2.202 $\pm$ 1.061	0.433 $\pm$ 1.123	3.723 $\pm$ 1.124	1.297 $\pm$ 0.089
Yars' Revenge	0.709 $\pm$ 0.036	0.713 $\pm$ 0.045	0.009 $\pm$ 0.003	0.698 $\pm$ 0.039	0.719 $\pm$ 0.009
Zaxxon	0.545 $\pm$ 0.044	0.509 $\pm$ 0.048	0.002 $\pm$ 0.003	0.715 $\pm$ 0.045	0.612 $\pm$ 0.006
<i>IQM $\uparrow$</i>	0.953 $\pm$ 0.092	0.986 $\pm$ 0.099	0.052 $\pm$ 0.053	0.911 $\pm$ 0.117	0.840 $\pm$ 0.085
<i>Opt. gap $\downarrow$</i>	0.200 $\pm$ 0.080	0.184 $\pm$ 0.076	0.825 $\pm$ 0.082	0.230 $\pm$ 0.085	0.249 $\pm$ 0.074

Table 8: Hyperparameter sensitivity in continuous control. Within each environment, ρ is the Spearman correlation between the hyperparameter and trial score, and Share is its L1-normalized magnitude, $|\rho_h|/\sum_j |\rho_j|$. Entries report the median magnitude and share across 57 environments; rows are ordered by median $|\rho|$. Because each statistic is aggregated independently, the displayed shares need not sum to 100%.

BC			BCP		
Hyperparameter	$ \rho $	Share (%)	Hyperparameter	$ \rho $	Share (%)
Hidden units	0.206	22.5	Train percentile	0.245	17.1
Learning rate	0.190	19.2	Hidden units	0.233	19.4
Dropout	0.171	19.8	Learning rate	0.202	13.8
Weight decay	0.127	12.1	Dropout	0.194	17.9
Policy type	0.094	10.7	Policy type	0.138	9.2
Grad-clip norm	0.092	8.7	Weight decay	0.104	10.2
			Grad-clip norm	0.099	7.2
BCQ			CQL		
Hyperparameter	$ \rho $	Share (%)	Hyperparameter	$ \rho $	Share (%)
γ	0.173	13.7	α threshold	0.204	15.3
Critic LR	0.151	11.2	No. critics	0.119	9.6
Imitator LR	0.131	10.1	α LR	0.117	10.5
λ	0.111	8.6	Critic LR	0.115	10.5
Actor update interval	0.096	7.3	γ	0.111	9.6
No. critics	0.090	7.6	Actor LR	0.104	8.5
Action samples	0.087	6.3	Action samples	0.073	6.8
β	0.086	7.3	Conserv. weight	0.068	6.2
Actor LR	0.080	6.4	τ	0.055	5.7
τ	0.079	6.4	Temp. LR	0.050	4.9
RL start step	0.072	5.0			
DT			IQL		
Hyperparameter	$ \rho $	Share (%)	Hyperparameter	$ \rho $	Share (%)
Hidden units	0.269	19.0	Actor LR	0.256	32.5
Learning rate	0.223	16.7	Expectile	0.085	10.4
Dropout (all)	0.146	11.1	No. critics	0.082	10.2
Weight decay	0.146	10.5	τ	0.082	10.6
Layers	0.141	10.5	Critic LR	0.081	9.6
γ	0.128	8.0	Weight temp.	0.081	11.3
Context size	0.112	8.0	γ	0.072	9.3
Heads	0.097	6.5			
Grad-clip norm	0.076	5.1			
ReBRAC			ACT		
Hyperparameter	$ \rho $	Share (%)	Hyperparameter	$ \rho $	Share (%)
Actor β	0.703	36.8	Learning rate	0.525	36.1
γ	0.142	8.2	Chunk size	0.238	17.0
Critic β	0.141	8.7	Action horizon	0.174	15.0
No. critics	0.134	8.0	Grad-clip norm	0.122	8.6
Critic LR	0.134	8.1	KL weight	0.113	9.2
τ	0.115	6.8	Weight decay	0.108	8.9
Target smooth. σ	0.114	7.0			
Actor LR	0.091	5.8			
Target smooth. clip	0.091	5.8			
VQ-BeT			Diffusion		
Hyperparameter	$ \rho $	Share (%)	Hyperparameter	$ \rho $	Share (%)
Learning rate	0.380	18.6	Diffusion steps	0.489	27.6
Offset LR	0.265	14.5	Inference steps	0.376	21.3
Action horizon	0.256	12.0	Learning rate	0.178	10.5
Chunk size	0.188	8.4	Action horizon	0.155	10.8
Focal γ	0.123	5.7	Chunk size	0.129	9.9
VQ-VAE step ratio	0.117	6.0	Grad-clip norm	0.096	6.6
Commitment weight	0.107	5.6	Weight decay	0.093	6.9
Weight decay	0.104	4.9			
Offset loss weight	0.101	4.6			
Grad-clip norm	0.091	4.6			
Secondary code wt.	0.087	4.4			
Primary code wt.	0.081	5.1			

Table 9: Hyperparameter sensitivity in discrete control (Atari). Within each environment, ρ is the Spearman correlation between the hyperparameter and trial score, and Share is its L1-normalized magnitude, $|\rho_h|/\sum_j |\rho_j|$. Entries report the median magnitude and share across 54 environments; rows are ordered by median $|\rho|$. Because each statistic is aggregated independently, the displayed shares need not sum to 100%.

BC			BCP		
Hyperparameter	$ \rho $	Share (%)	Hyperparameter	$ \rho $	Share (%)
Learning rate	0.233	25.7	Learning rate	0.233	24.6
Dropout	0.161	21.1	Dropout	0.156	17.3
Hidden units	0.097	12.0	Train percentile	0.127	13.3
Reg. factor	0.092	11.0	Reg. factor	0.101	11.9
Weight decay	0.084	10.4	Grad-clip norm	0.087	9.9
Grad-clip norm	0.076	9.7	Hidden units	0.079	10.2
			Weight decay	0.077	8.9
BCQ			CQL		
Hyperparameter	$ \rho $	Share (%)	Hyperparameter	$ \rho $	Share (%)
Actor LR	0.234	38.7	Actor LR	0.655	56.0
Target update interval	0.079	11.4	α	0.143	12.2
β	0.079	12.4	No. critics	0.109	10.6
γ	0.078	9.4	Target update interval	0.102	10.2
Action flexibility	0.073	9.4	γ	0.096	7.3
No. critics	0.066	9.6			
DT					
Hyperparameter	$ \rho $	Share (%)			
Learning rate	0.465	31.2			
Hidden units	0.191	12.5			
Dropout (all)	0.143	9.6			
Context size	0.113	8.3			
Heads	0.101	7.3			
Layers	0.092	7.8			
Grad-clip norm	0.092	7.5			
γ	0.088	7.0			
Weight decay	0.087	6.0			

References

- [1] Bingyi Kang, Xiao Ma, Yirui Wang, Yang Yue, and Shuicheng Yan. Improving and benchmarking offline reinforcement learning algorithms. *arXiv preprint arXiv:2306.00972*, 2023.
- [2] Edoardo Cetin, Andrea Tirinzoni, Matteo Pirota, Alessandro Lazaric, Yann Ollivier, and Ahmed Touati. Simple ingredients for offline reinforcement learning. *arXiv preprint arXiv:2403.13097*, 2024.
- [3] Kajetan Schweighofer, Andreas Radler, Marius-Constantin Dinu, Markus Hofmarcher, Vihang Patil, Angela Bitto-Nemling, Hamid Eghbal-zadeh, and Sepp Hochreiter. A dataset perspective on offline reinforcement learning. *arXiv preprint arXiv:2111.04714*, 2021.
- [4] Prajwal Bhargava, Rohan Chitnis, Alborz Geramifard, Shagun Sodhani, and Amy Zhang. When should we prefer decision transformers for offline reinforcement learning? *arXiv preprint arXiv:2305.14550*, 2023.
- [5] 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, 2021. URL <https://arxiv.org/abs/2108.03298>.
- [6] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Belle-mare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2108.13264.
- [7] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *CoRR*, abs/2110.06169, 2021.
- [8] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *CoRR*, abs/2304.13705, 2023.
- [9] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Vq-bet: Behavior generation with latent actions. In *International Conference on Machine Learning (ICML)*, 2024.
- [10] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *CoRR*, abs/2303.04137, 2023.
- [11] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [12] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [13] Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *CoRR*, abs/2305.09836, 2023.
- [14] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration, 2019. URL <https://arxiv.org/abs/1812.02900>.
- [15] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021. URL <https://arxiv.org/abs/2004.07219>.
- [16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [17] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012.

- [18] 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. URL <https://arxiv.org/abs/1709.10087>.
- [19] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. arXiv:1910.11956.
- [20] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M. Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. In *arXiv preprint arXiv:1710.05465*, 2017.
- [21] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*, 2017.
- [22] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction, 2019. URL <https://arxiv.org/abs/1906.00949>.
- [23] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2019. URL <https://arxiv.org/abs/1911.11361>.
- [24] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.
- [25] 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*, 2023. URL <https://arxiv.org/abs/2206.04779>.
- [26] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023. arXiv:2306.03310.
- [27] Rafael Rafailov, Kyle Beltran Hatch, Anikait Singh, Laura Smith, Aviral Kumar, Ilya Kostrikov, Philippe Hansen-Estruch, Victor Kolev, Philip J. Ball, Jiajun Wu, Chelsea Finn, and Sergey Levine. D5RL: Diverse datasets for data-driven deep reinforcement learning. *Reinforcement Learning Journal*, 5:2178–2197, 2024. arXiv:2408.08441.
- [28] Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025. URL <https://arxiv.org/abs/2410.20092>.
- [29] Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. Minari, September 2024. URL <https://doi.org/10.5281/zenodo.13767625>.
- [30] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study, 2020. URL <https://arxiv.org/abs/2006.05990>.
- [31] Manu Orsini, Anton Raichuk, Léonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. What matters for adversarial imitation learning?, 2021. URL <https://arxiv.org/abs/2106.00672>.
- [32] Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning, 2020. URL <https://arxiv.org/abs/2007.09055>.

- [33] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0 : A vision-language-action flow model for general robot control, 2026. URL <https://arxiv.org/abs/2410.24164>.
- [34] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 104–114. PMLR, 2020. URL <https://proceedings.mlr.press/v119/agarwal20c.html>.
- [35] Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *Journal of Machine Learning Research*, 23(315):1–20, 2022. URL <https://www.jmlr.org/papers/v23/22-0017.html>.
- [36] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłoś, Błażej Osipiński, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1903.00374.
- [37] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training, 2018. URL <https://arxiv.org/abs/1807.05118>.
- [38] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.
- [39] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24 (NeurIPS)*, pages 2546–2554, 2011.
- [40] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.
- [41] 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.
- [42] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL <https://arxiv.org/abs/1801.01290>.
- [43] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL <http://jmlr.org/papers/v22/20-1364.html>.
- [44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.
- [45] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5556–5566. PMLR, 13–18 Jul 2020. URL <https://proceedings.mlr.press/v119/kuznetsov20a.html>.
- [46] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL <http://dx.doi.org/10.1613/jair.3912>.

- [47] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, and Jeff Braga. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms, 2021. URL <https://arxiv.org/abs/2111.08819>.
- [48] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018. URL <https://arxiv.org/abs/1802.01561>.

A Datasets

We describe the collection process for each dataset used in our analysis. We did not create or contribute these datasets, but include these details as they may be relevant to readers. Dataset identifiers below follow the Minari registry. Our benchmark contains 114 environment–dataset pairs: 29 from the Minari release of D4RL (commit 6c51941431a72f66721e5b4ed92c7ce6f116c816), 28 from the Minari MuJoCo collection (commit 8e62dc7f7fcb4a19f8f869c65402d4bb60049117), and 57 from our larger reproduction of Minari’s Atari collection, the 100,000-transition `omi-n/atari-medium` collection.

AntMaze. The six AntMaze datasets require a MuJoCo Ant to navigate a maze under sparse reward. They were collected with a hierarchical controller: a Q-iteration planner selected waypoints and a goal-conditioned SAC policy supplied the low-level actions [41, 42]. `umaze-v1` uses a fixed start and goal, whereas `umaze-diverse-v1` varies both over a set of valid maze cells. The medium and large layouts each have a `play` dataset, built from hand-selected start–goal pairs, and a `diverse` dataset, which samples both starts and goals more broadly. These give the four datasets `medium-play-v1`, `medium-diverse-v1`, `large-play-v1`, and `large-diverse-v1`.

Adroit Door. Door uses a 24-degree-of-freedom Adroit hand to unlatch and open a door. `human-v2` contains 25 teleoperated human demonstrations. `expert-v2` contains rollouts of a policy trained by demo-augmented policy gradient (DAPG) [18]. `cloned-v2` is deliberately imperfect: a behavioral-cloning policy was trained on the union of the human and expert data, and its rollouts were mixed approximately equally with the source demonstrations [41].

Adroit Hammer. Hammer requires the Adroit hand to pick up a hammer and drive a nail. Its `human-v2`, `expert-v2`, and `cloned-v2` datasets were collected in the same three regimes as Door: 25 human teleoperation demonstrations, DAPG-policy rollouts, and an approximately equal mixture of source demonstrations and rollouts from a behavioral clone, respectively [18, 41].

Adroit Pen. Pen requires reorienting a pen to a target pose. `human-v2` contains 25 human teleoperation demonstrations, `expert-v2` contains DAPG-policy rollouts, and `cloned-v2` combines the source demonstrations with rollouts of a behavioral clone trained on the human and expert data [18, 41].

Adroit Relocate. Relocate requires the Adroit hand to move an object to a target position. As in the other Adroit environments, `human-v2` contains 25 teleoperated demonstrations, `expert-v2` contains DAPG-policy trajectories, and `cloned-v2` mixes the source demonstrations approximately equally with trajectories from a behavioral-cloning policy [18, 41].

Kitchen. The three Franka Kitchen datasets contain human demonstrations collected through virtual-reality teleoperation [19, 41]. Each trajectory may include interactions with several appliances or fixtures. In `complete-v2`, all four target subtasks are completed in order. `partial-v2` contains mixed-task demonstrations with sub-trajectories that complete the target subtasks in sequence, while `mixed-v2` contains broader subtask combinations in which the four target subtasks are never all completed in sequence. Unlike the locomotion datasets below, these variants therefore differ in task coverage rather than in the training checkpoint of a single behavior policy.

PointMaze—Open. PointMaze controls a point mass rather than a legged robot. A hand-coded proportional-derivative controller followed waypoints produced by a Q-iteration planner toward sampled goals [41, 29]. `open-v2` records the sparse-reward version of the open layout, and `open-dense-v2` uses dense distance-based reward; the behavior policy is otherwise the same.

PointMaze—U-Maze. `umaze-v2` and `umaze-dense-v2` were collected in the U-shaped maze with the same scripted controller and Q-iteration planner. The former uses sparse goal reward and the latter dense shaped reward.

PointMaze—Medium. `medium-v2` and `medium-dense-v2` increase the size and path complexity of the maze. Both were generated by the scripted planner–controller pair; they differ only in whether the recorded reward is sparse or dense.

PointMaze—Large. `large-v2` and `large-dense-v2` use the largest PointMaze layout in the benchmark. As above, a Q-iteration planner supplied waypoints to a proportional-derivative controller, and the two datasets use sparse and dense reward, respectively.

MuJoCo collection. The MuJoCo datasets were generated by rolling out checkpoints of policies trained in the corresponding Gymnasium environments [17, 29]. The behavior policies were implemented with Stable-Baselines3 and use SAC, PPO, or TQC [43, 42, 44, 45]. The labels `simple`, `medium`, and `expert` denote increasing training budgets, not mixtures of trajectories from policies of different quality. Except where noted below, the variants for an environment are checkpoints from one training run.

Ant. Ant is a quadrupedal locomotion task. `simple-v0` and `medium-v0` were collected from SAC checkpoints after 1 and 10 million training steps. `expert-v0` was generated separately from an externally trained SAC checkpoint selected as the best pretrained model.

HalfCheetah. HalfCheetah asks a planar articulated agent to run forward. Its three datasets use truncated quantile critics (TQC), a distributional variant of SAC: `simple-v0`, `medium-v0`, and `expert-v0` correspond to checkpoints after 1, 6, and 25 million training steps.

Hopper. Hopper is one-legged forward locomotion. The `simple-v0`, `medium-v0`, and `expert-v0` datasets were generated from SAC checkpoints after 1, 4, and 25 million training steps, respectively.

Walker2d. Walker2d is planar bipedal locomotion. Its `simple-v0`, `medium-v0`, and `expert-v0` datasets use SAC checkpoints after 1.5, 6, and 25 million training steps.

Swimmer. Swimmer propels a three-link body through fluid by actuating its joints. It has no `simple` variant. `medium-v0` and `expert-v0` were collected from PPO checkpoints after 300 thousand and 100 million training steps. PPO was trained with an undiscounted objective because SAC-style discounting was ineffective for this environment.

Reacher. Reacher asks a two-link arm to move its end effector to a target. It has two SAC-generated datasets: `medium-v0` uses a checkpoint after 200 thousand training steps and `expert-v0` a checkpoint after 10 million steps.

Pusher. Pusher requires a robotic arm to push an object toward a target. `medium-v0` and `expert-v0` were generated from SAC checkpoints after 1 and 10 million training steps; no `simple` variant is provided.

Humanoid. Humanoid is high-dimensional bipedal locomotion. Its trajectories were collected from TQC policies: `simple-v0`, `medium-v0`, and `expert-v0` use checkpoints after 2, 5, and 20 million training steps.

HumanoidStandup. HumanoidStandup requires a fallen humanoid to rise to an upright posture. The behavior policy was SAC trained with a reduced observation representation. `simple-v0`, `medium-v0`, and `expert-v0` were collected after 2, 10, and 50 million training steps.

InvertedPendulum. InvertedPendulum requires balancing a pole on a moving cart. `medium-v0` and `expert-v0` contain rollouts of SAC checkpoints after 10 thousand and 1 million training steps, respectively.

InvertedDoublePendulum. InvertedDoublePendulum extends the balancing task to two hinged links. Its `medium-v0` and `expert-v0` datasets were collected from SAC checkpoints after 50 thousand and 10 million training steps.

Atari. The discrete-control portion of the benchmark contains one `expert-v0` dataset for each of 57 Atari games in the Arcade Learning Environment [46]: Alien, Amidar, Assault, Asterix, Asteroids, Atlantis, Bank Heist, Battle Zone, Beam Rider, Berzerk, Bowling, Boxing, Breakout, Centipede, Chopper Command, Crazy Climber, Defender, Demon Attack, Double Dunk, Enduro, Fishing Derby, Freeway, Frostbite, Gopher, Gravitar, H.E.R.O., Ice Hockey, James Bond, Kangaroo, Krull, Kung-Fu Master, Montezuma’s Revenge, Ms. Pac-Man, Name This Game, Phoenix, Pitfall!, Pong, Private Eye, Q*bert, River Raid, Road Runner, Robot Tank, Seaquest, Skiing, Solaris, Space Invaders, Star Gunner, Surround, Tennis, Time Pilot, Tutankham, Up’n Down, Venture, Video Pinball, Wizard of Wor, Yars’ Revenge, and Zaxxon. For each game, the same pretrained CleanBA PPO policy with an IMPALA-style convolutional encoder that Minari used was rolled out in the corresponding ALE environment using the original Minari data generation scripts until a target of 100,000 transitions was reached [47, 44, 48]. Actions were sampled stochastically from the policy logits rather than chosen greedily, and sticky actions were disabled. The policy received resized grayscale stacks of four frames, but collection occurred before those wrappers, so the datasets retain raw image observations. The

atari-medium label describes dataset size; all 57 datasets use the same expert-policy collection procedure and do not represent medium-quality behavior.

B Original Paper Baselines

In Figure 1, we compare our trained models with the original authors’ implementations, each trained across three random seeds. We use the official repositories for BCQ, CQL, DT, IQL, and ReBRAC.⁵

To preserve the original training implementations, we restrict our modifications to data-ingestion adapters and environment construction. For CQL, we load the Minari HDF5 data and insert its transitions into `rlkit`’s `EnvReplayBuffer`. For DT, we convert each Minari dataset to the expected pickle format. For BCQ, we reconstruct episode buffers from Minari’s stored episodes and load the observations, actions, rewards, terminal indicators, and next observations into BCQ’s `ReplayBuffer`. We apply the same principle to the remaining methods, converting each dataset into the format expected by the original framework while leaving the algorithm and training procedure unchanged. This avoids introducing unintended differences in preprocessing, numerical precision, or trajectory construction.

To ensure that all baselines are evaluated consistently, we adapt each repository’s evaluation procedure to match our framework: we evaluate only the final checkpoint over 100 episodes. We reconstruct the exact environment and version associated with each Minari dataset and add any omitted constructor arguments or wrappers according to Minari’s data-generation scripts. Because most original implementations use the Gym API [16], these changes are limited primarily to environment construction and evaluation, leaving the underlying policies and training procedures unchanged.

C Hyperparameter Sensitivity Breakdown

We provide detailed heatmaps of the hyperparameter sensitivities from Section 5 in Figure 25.

⁵Repositories: BCQ, CQL, DT, IQL, and ReBRAC.

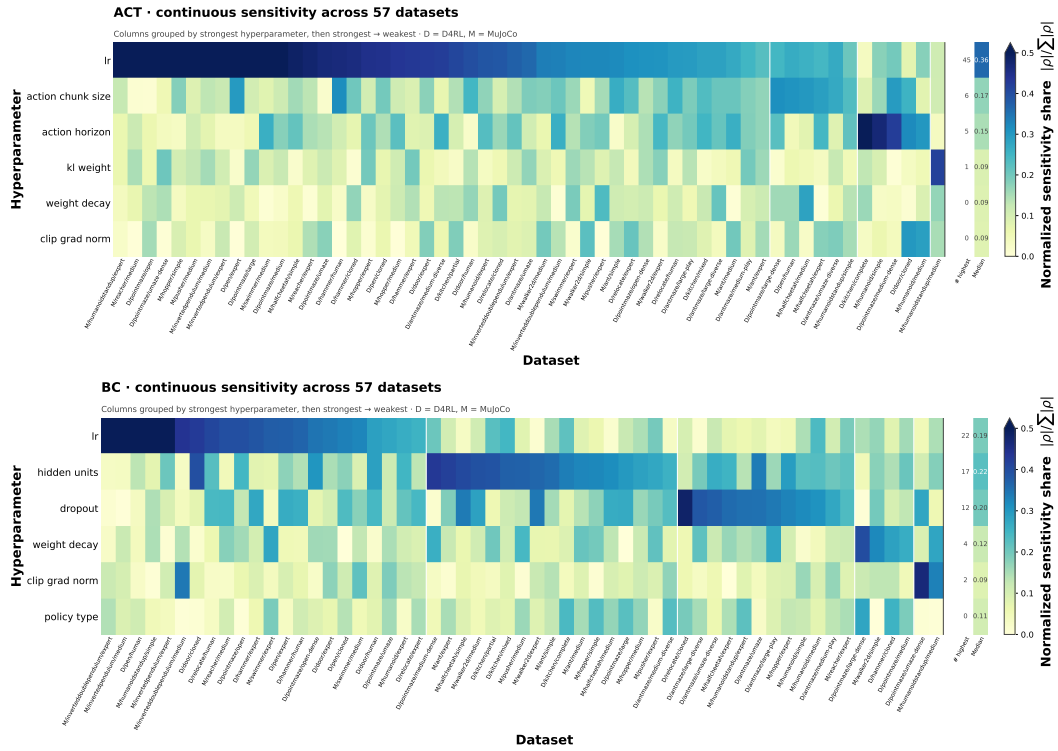


Figure 25: Per-environment hyperparameter-sensitivity breakdowns for the continuous-control algorithms. Each heatmap reports relative absolute Spearman associations between hyperparameters and trial performance. Zoom in for environment and hyperparameter labels.

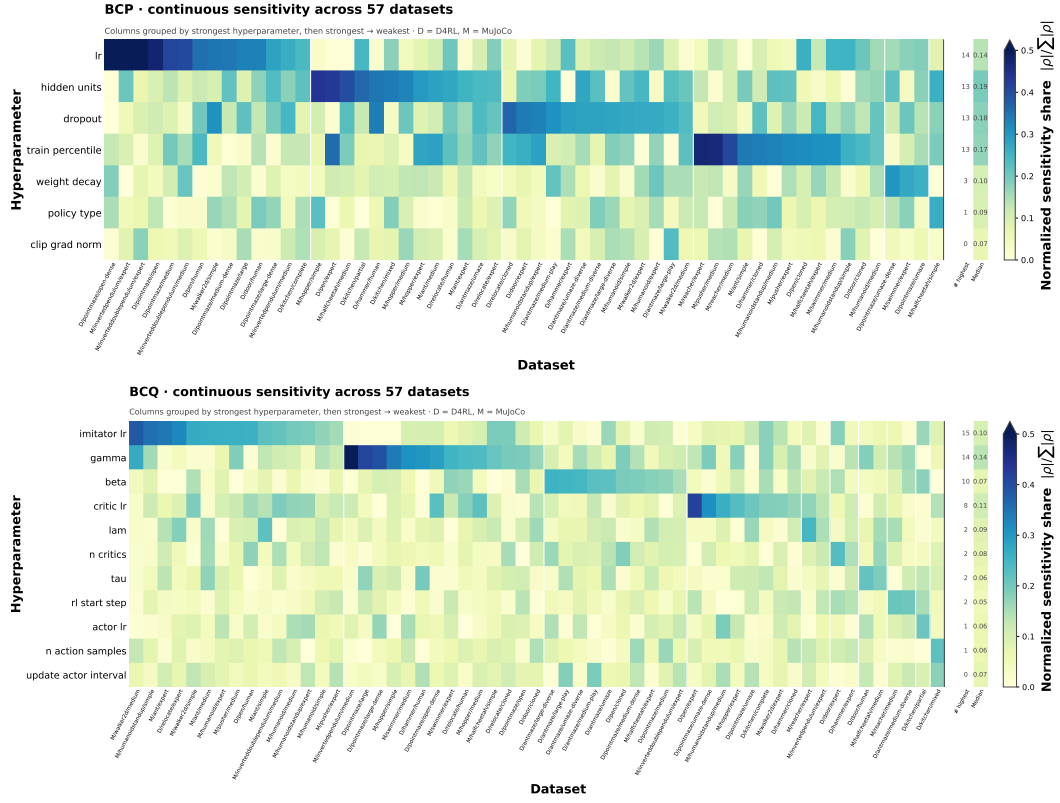


Figure 25: Hyperparameter-sensitivity breakdowns (continued).

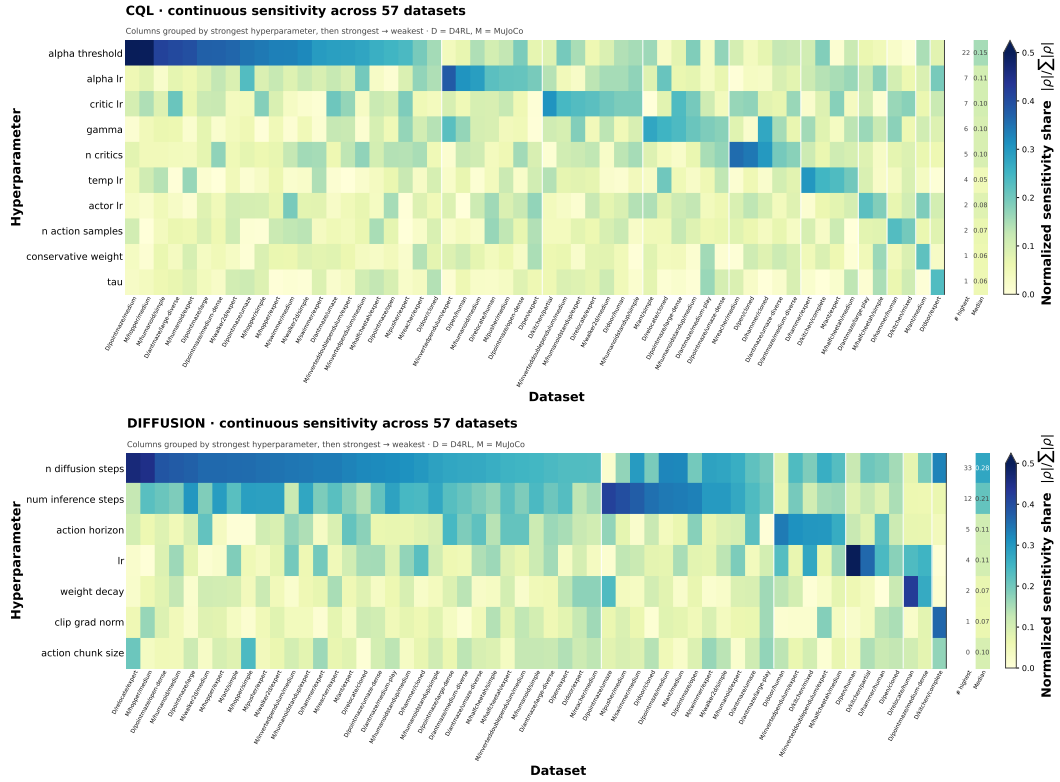


Figure 25: Hyperparameter-sensitivity breakdowns (continued).

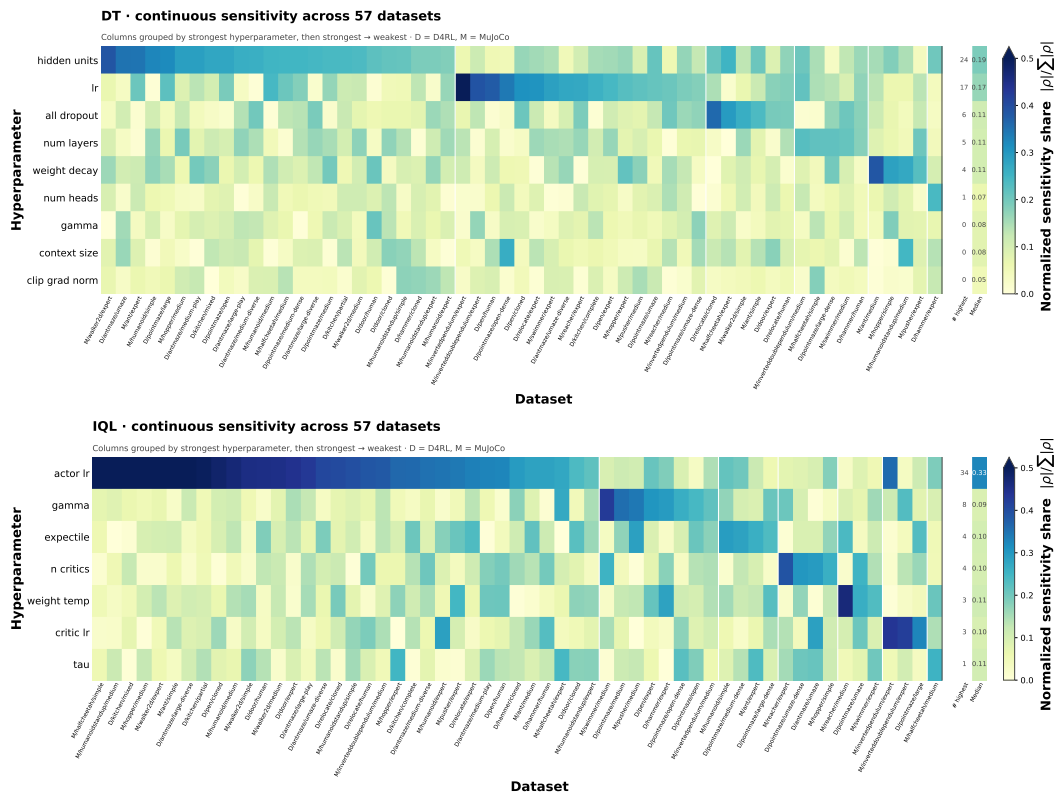


Figure 25: Hyperparameter-sensitivity breakdowns (continued).

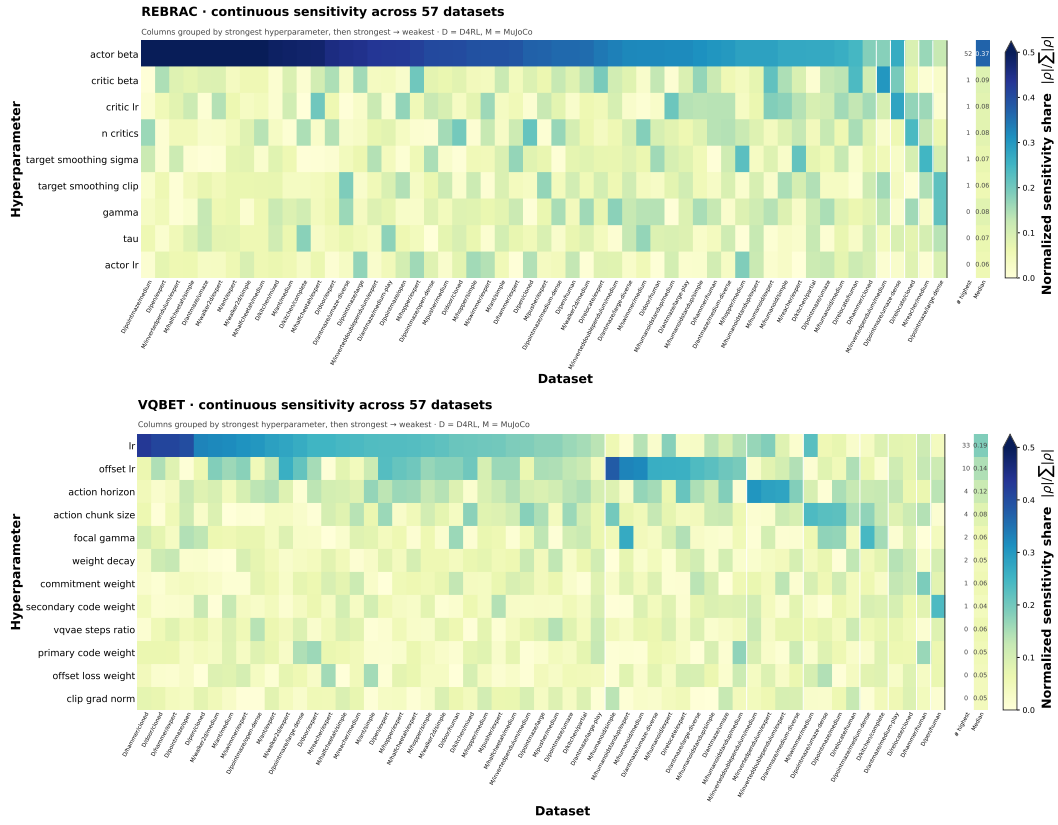


Figure 25: Hyperparameter-sensitivity breakdowns (continued).