ROLE-DEPENDENT USE OF MIXED-QUALITY DATA IN OFFLINE RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning learns from fixed datasets whose transitions may vary substantially in behavioral quality. Data generated by poor behavior, however, may still provide useful transitions for critic learning or useful states for policy improvement even when their recorded actions are undesirable targets for imitation. We study this distinction by separating three roles of offline data: critic learning, Q-guided policy improvement at recorded states, and behavioral supervision from recorded actions. Controlled routing experiments with expert–random mixtures show that the effect of the same additional data can differ substantially depending on the learning operation in which it is used, and can even change sign across environments and data regimes. Motivated by these findings, we introduce Trust-Mix, which uses the full dataset for critic learning and Q-guided policy improvement while selectively weighting recorded actions for behavioral supervision, without requiring source-policy labels. Experiments across three continuous-control environments and four expert-data budgets show that the benefit of additional random data varies across environments, data regimes, and offline RL algorithms.

1 INTRODUCTION

Offline reinforcement learning (RL) learns a policy from previously recorded experience without collecting new transitions during training (Levine et al., 2020). When data come from different types of behavior, however, the usefulness of a transition need not depend only on the quality of its recorded action. An undesirable action may still yield reward and next-state information useful for value learning, and its state may provide a useful point for learning a different action. Thus, a transition can remain useful even when reproducing its recorded action is undesirable.

This distinction is important when a small amount of high-quality experience is combined with a much larger amount of lower-quality data. Existing approaches to heterogeneous offline data include trajectory weighting, distribution reweighting, dataset selection, and behavior-regularized policy learning (Hong et al., 2023a;b; Yang et al., 2025; Wu et al., 2019; Fujimoto & Gu, 2021), while other methods selectively weight recorded actions during policy learning (Peng et al., 2019; Kostrikov et al., 2022). Neither separating value learning from policy learning nor selectively weighting behavioral supervision is new. We instead ask: *how does the effect of additional data depend on what the learner uses that data for?*

We distinguish three roles for an offline transition: the complete transition (s, a, r, s') can support critic learning; its state s can provide a point for Q-guided policy improvement; and the recorded pair (s, a) can directly supervise the policy through behavioral imitation. These roles need not assign the same utility to the same data. In particular, data generated by poor behavior may provide useful transition or state information while providing undesirable actions for direct supervision.

We investigate this distinction using datasets containing a small expert-policy component and a much larger random-policy component. Source-aware diagnostic controls progressively make the same random component available to critic learning, Q-guided policy improvement, and the behavioral branch. The effects differ substantially across roles. In HalfCheetah, extending random-data access to Q-guided states improves return at both the 1k and 100k expert-data budgets, while the subsequent extension to BC/proximal supervision and initialization reduces it. Hopper and Walker2d exhibit

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

different patterns across roles and expert-data regimes. These results show that the effect of the same additional data can change with the learning operation through which it enters the learner.

Motivated by this finding, we introduce TRUST-MIX, which implements role-dependent data use without source-policy labels. TRUST-MIX keeps the full dataset available for critic learning and Q-guided policy improvement while assigning state–action-specific weights to recorded actions used for behavioral supervision. A recorded action can therefore receive little or no imitation weight while its transition and state remain available to other learning operations.

We evaluate TRUST-MIX on HalfCheetah, Hopper, and Walker2d using four expert-data budgets and a fixed one-million-transition random component. Paired expert-only and mixed-data experiments, together with comparisons to offline RL baselines, show substantial variation in the effect of adding the same random data across environments, data regimes, and algorithms.

Our contributions are: (1) a formulation of mixed-quality offline data in terms of three learning roles: critic learning, Q-guided policy-improvement states, and behavioral supervision; (2) controlled routing experiments showing that the same additional data can have different, and sometimes opposite, effects across these roles; (3) TRUST-MIX, a source-label-free method that implements role-dependent data use through selective behavioral weighting; and (4) an empirical characterization of how the effect of adding the same random-data source varies across environments, expert-data budgets, and offline RL algorithms.

2 RELATED WORK

Heterogeneous offline data. Prior work has addressed variation in offline-data quality through weighting, selection, and changes to data sampling. Trajectory quality and state–action coverage are distinct dataset properties (Schweighofer et al., 2021). Trajectory weighting assigns sampling weights using returns or trajectory-level advantages (Hong et al., 2023a), while density-ratio weighting can shift sampling toward higher-return behavior (Hong et al., 2023b). ReDOR selects weighted subsets whose training gradients approximate those of the full dataset (Yang et al., 2025). Related-task data can also reduce offline RL performance, as studied in the MOOD benchmark (Cetin et al., 2024). Our experiments instead hold the additional same-task data fixed and examine how its effect changes with the learning operation in which it is used.

Selective behavioral regularization. Several offline RL methods already separate the use of data for value learning from its influence on the learned policy. AWR and IQL use advantage-weighted policy regression (Peng et al., 2019; Kostrikov et al., 2022), and CRR learns from transitions while filtering or weighting recorded actions during policy regression (Wang et al., 2020). Behavior-regularized methods such as BRAC, TD3+BC, and ReBRAC combine value-guided policy improvement with constraints toward dataset behavior (Wu et al., 2019; Fujimoto & Gu, 2021; Tarasov et al., 2023). More recent approaches select or adapt behavioral regularization using value- or advantage-related signals, including ReDS, A2PR, A2PO, VACO, and SSAR (Singh et al., 2023; Liu et al., 2024; Qing et al., 2024; Jiang et al., 2025; Luo et al., 2025). In particular, SSAR's TD3+BC extension combines full-dataset Q-guided policy improvement with selective behavioral regularization. Thus, selective imitation and broad Q-state access are not themselves new. Our focus is the empirical question of whether the effect of the *same additional data* changes across critic learning, Q-guided policy-improvement states, and behavioral supervision, together with Trust-Mix as one implementation of this role separation.

3 PROBLEM SETTING AND DATA ROLES

We consider offline reinforcement learning in a discounted Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \mu_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s' \mid s, a)$ is the transition probability, $r(s, a)$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and μ_0 is the initial-state distribution. The learner receives a fixed dataset

$$D = \{(s_i, a_i, r_i, s'_i, d_i)\}_{i=1}^{N},$$

where s_i and a_i are the recorded state and action, r_i is the observed reward, s'_i is the next state, and d_i indicates a true environment termination. Recorded episode boundaries and truncations are retained in the dataset. No additional environment interactions are used during learning.

Mixed-quality data. We study datasets constructed from two behavior policies. Let $D_E^{(b)}$ contain b transitions generated by an expert policy and let D_R contain transitions generated by a random policy in the same environment. We compare learning from the expert-only dataset $D = D_E^{(b)}$ with learning from the mixed dataset $D = D_E^{(b)} \cup D_R$. The expert transitions are identical between the corresponding expert-only and mixed conditions. Source labels are used only to construct the datasets and for the diagnostic experiments described in Section 5; they are not provided to Trust-Mix during training. We refer to D_R as low-quality data because it is generated by random behavior; this does not imply that every transition in D_R is uninformative.

Three learning roles. A recorded transition can affect an actor–critic learner through different operations. We distinguish three roles:

1. **Critic learning.** The complete transition (s_i, a_i, r_i, s'_i) can be used to train a critic $Q(s, a)$. The recorded action contributes information about its consequences without requiring the actor to reproduce it.
2. **Q-guided policy improvement.** The recorded state s_i can be used as a state at which the actor proposes an action and receives a value-guided update from the critic. The actor can therefore learn a different action at s_i without imitating the recorded action a_i.
3. **Behavioral supervision.** The recorded state–action pair (s_i, a_i) can directly supervise the actor, encouraging the policy to reproduce a_i at s_i.

These roles separate learning from the consequences and states of recorded behavior from directly imitating its actions. A transition generated by poor behavior may therefore still provide useful information for critic learning or a useful state for policy improvement, even when its recorded action is undesirable as a target for behavioral supervision. Our central empirical question is whether the effect of additional low-quality data depends on how it is made available to these learning operations.

Section 4 introduces Trust-Mix, which implements this distinction without access to source labels. Section 5 introduces source-aware diagnostic controls that successively extend the random component to different learning operations.

4 TRUST-MIX

Trust-Mix uses the full offline dataset for critic learning and Q-guided policy improvement while selectively weighting recorded actions used for behavioral supervision. It does not require behavior-policy source labels.

4.1 CRITIC LEARNING

Trust-Mix uses one actor π_θ and two critic pairs. The *trust critics*, Q_1^T and Q_2^T, estimate values used to weight behavioral supervision, while the *policy critics*, Q_1^P and Q_2^P, provide values for Q-guided actor updates. All four critics are trained on transitions sampled from the full dataset D. The trust critics use multi-step targets; the policy critics use one-step targets with a sampled-action pessimistic penalty. Critic objectives, target construction, and update schedules are given in Appendix A.

4.2 ACTOR LEARNING

Selective behavioral supervision. After initialization, Trust-Mix samples B_t uniformly from D. For each (s_i, a_i), it compares the recorded action with the target actor's action. The estimated advantage and trust-critic disagreement are

$$A_i(t) = \widetilde{Q}^T_{\min}(s_i, a_i) - \widetilde{Q}^T_{\min}\left(s_i, a_{\bar{\theta}_t}(s_i)\right), \qquad U_i(t) = \left|\widetilde{Q}^T_1(s_i, a_i) - \widetilde{Q}^T_2(s_i, a_i)\right|. \tag{1}$$

Here, $A_i(t) > 0$ indicates that the recorded action has higher estimated value than the target actor's action, while $U_i(t)$ provides a disagreement signal for weighting. Trust-Mix defines

$$f_i(t) = \left(1 - \exp\left[-\frac{A_i(t)}{\beta_A s_A(t)}\right]\right) \exp\left[-\beta_U \frac{U_i(t)}{u_{75}(t)}\right], \tag{2}$$

and

$$w_i(t) = \begin{cases} 0, & A_i(t) \le 0, \\ \mathrm{clip}\left(f_i(t), w_{\min}, 1\right), & A_i(t) > 0. \end{cases} \tag{3}$$

Thus, actions with non-positive estimated advantage receive no behavioral weight; for positive advantages, weighting increases with estimated advantage and decreases with critic disagreement. Definitions of the robust batch scales $s_A(t)$ and $u_{75}(t)$ and the weighting constants are given in Appendix A.

Weights are recomputed at each actor step and detached from the actor gradient, $\hat{w}_i = \mathrm{sg}[w_i(t)]$. Let

$$Z_t = \max\left\{\sum_{i \in B_t} \hat{w}_i, \epsilon_Z\right\}. \tag{4}$$

The weighted behavioral-cloning objective is

$$\mathcal{L}_{\mathrm{BC}}(\theta; t) = -\frac{1}{Z_t} \sum_{i \in B_t} \hat{w}_i \log \pi_\theta(a_i \mid s_i). \tag{5}$$

An action with $w_i(t) = 0$ makes no contribution to this objective, while its transition and state remain available to the other learning operations.

Q-guided policy improvement. Trust-Mix independently samples a uniform state batch C_t from D and minimizes

$$\mathcal{L}_Q(\theta; t) = -\frac{1}{|C_t|} \sum_{s_i \in C_t} Q^P_{\min}\left(s_i, a_\theta(s_i)\right). \tag{6}$$

Policy-critic parameters are fixed during this update. States in C_t are not filtered or weighted by $w_i(t)$, so they remain available for Q-guided policy improvement regardless of the behavioral weight of their recorded actions.

Proximal stabilization. Using B_t, $\hat{w}_i$, and Z_t, Trust-Mix also penalizes changes from a recent actor copy π_{prox}:

$$\mathcal{L}_{\mathrm{prox}}(\theta; t) = \frac{1}{Z_t} \sum_{i \in B_t} \hat{w}_i \left\| a_\theta(s_i) - a_{\mathrm{prox}}(s_i) \right\|_2^2. \tag{7}$$

The proximal actor is distinct from the target actor in Eq. 1; their update schedules are given in Appendix A.

4.3 Combined Actor Update

Define

$$g_{\mathrm{BC}} = \nabla_\theta[\alpha_{\mathrm{BC}}(t)\mathcal{L}_{\mathrm{BC}}], \qquad g_Q = \nabla_\theta \mathcal{L}_Q, \qquad g_{\mathrm{prox}} = \nabla_\theta[\lambda_{\mathrm{prox}}\mathcal{L}_{\mathrm{prox}}]. \tag{8}$$

Trust-Mix caps the Q-guided gradient relative to the behavioral gradient:

$$c_Q = \min\left(1, \frac{\rho_Q \|g_{\mathrm{BC}}\|_2}{\|g_Q\|_2}\right), \qquad g_\theta = g_{\mathrm{BC}} + c_Q g_Q + g_{\mathrm{prox}}. \tag{9}$$

The scale c_Q is held fixed during the actor update and can reduce, but not amplify, the Q-guided component. Initialization, zero-norm handling, gradient clipping, coefficients, optimizer details, and update schedules are provided in Appendix A.

5 Experimental Design

We study two questions: (1) How does adding a large random-policy dataset to a smaller expert dataset affect offline RL performance? (2) How does this effect change as the additional data are made available to different learning operations?

5.1 Environments and Evaluation

We evaluate on HalfCheetah, Hopper, and Walker2d continuous-control environments (Towers et al., 2024). For each environment, we construct expert subsets with $b \in \{1\text{k}, 5\text{k}, 10\text{k}, 100\text{k}\}$ transitions and a random-policy dataset D_R containing one million transitions. For each budget, we compare the expert-only dataset $D_E^{(b)}$ with the mixed dataset $D_E^{(b)} \cup D_R$, using identical expert transitions in the paired conditions.

We retain episode boundaries and distinguish true terminations from time-limit truncations. Observations and rewards are not normalized. Unless otherwise stated, methods are trained for one million gradient steps with ten learner seeds and evaluated every 50,000 steps using 20 episodes. Primary comparisons use the policy after one million updates.

For matched expert-only and mixed runs, we report paired mixed-minus-expert differences in evaluation return and 95% confidence intervals obtained by bootstrapping matched learner seeds. Positive differences indicate higher return after adding D_R.

5.2 Source-Aware Data-Routing Controls

To examine how the effect of D_R depends on its use, we construct four source-aware diagnostic conditions (Figure 1). C1 (*Expert only*) uses D_E throughout. C2 (*Mixed critics*) also uses D_R for critic learning. C3 (*Mixed critics + Q states*) further makes states from D_R available for Q-guided policy improvement. C4 (*Mixed throughout*) additionally uses D_R for BC/proximal supervision and initialization. Each condition is trained independently.

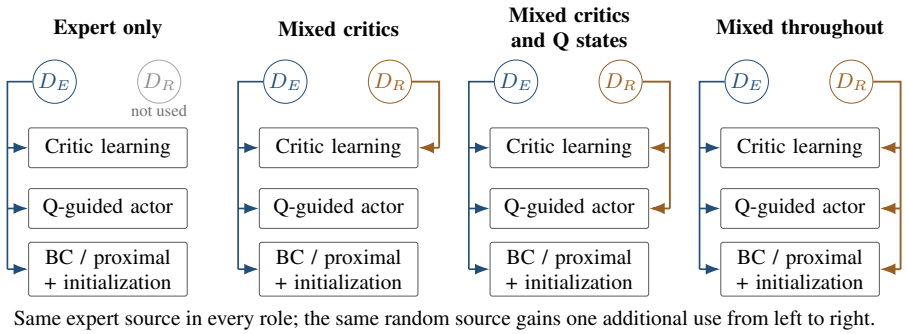

Figure 1: Source-aware routing controls. Expert data (D_E) are used throughout; from C1 to C4, the same random component (D_R) is progressively added to critic learning, Q-guided policy improvement, and BC/proximal supervision and initialization.

We report matched-seed differences for the successive contrasts C2−C1, C3−C2, and C4−C3 at the 1k and 100k expert budgets. These effects are conditional on the preceding routing choices and are not an additive decomposition of performance. The C4−C3 contrast jointly changes BC/proximal supervision and initialization and therefore does not isolate behavioral cloning. These controls use source identities and unit behavioral weights only for diagnosis; Trust-Mix itself does not use source labels.

5.3 Baselines and Ablations

We compare Trust-Mix with seven offline RL baselines on the same expert-only and mixed datasets. We additionally evaluate a CQL-based data-weighting adaptation as a diagnostic comparison; it is not a verified reproduction of the published implementation. Finally, we ablate behavioral weighting, Q-guided policy improvement, Q-gradient capping, and proximal regularization. Trust-Mix details are provided in Appendix A. The standard baselines use OfflineRL-Kit (Sun & Zhang, 2023), except ReBRAC, which uses a CORL-based adaptation. Their source settings and evaluation alignment are in Appendix C.4; CQL+DW-AW is detailed in Appendix C.5.

6 RESULTS

All endpoint comparisons use the prespecified policy after one million updates, without checkpoint selection based on evaluation return.

6.1 EFFECT OF ADDING RANDOM DATA

Figure 2 shows the paired effect of adding D_R. In HalfCheetah, the effect is positive at all four expert-data budgets, with all 95% intervals above zero. In Hopper, the intervals are above zero at 1k and 10k and include zero at 5k and 100k. None of the Walker2d intervals excludes zero. Thus, the effect of adding the same random-data component varies across environments and expert-data budgets. Absolute returns are reported in Appendix C.1.

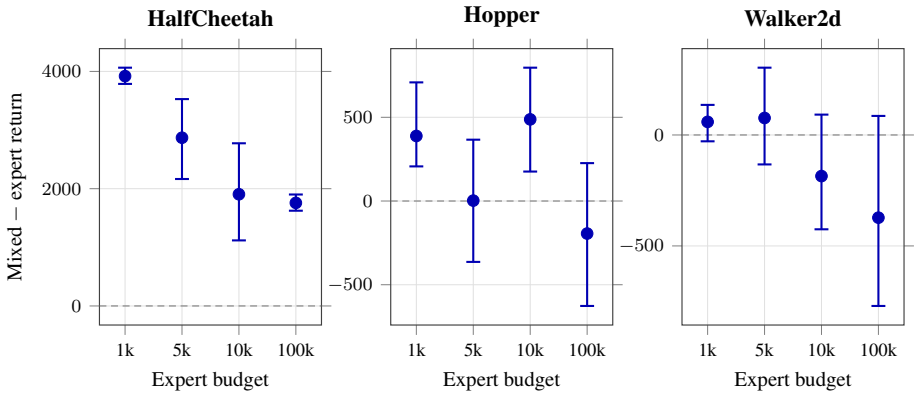

Figure 2: Paired effect of adding D_R. Points show mean mixed-minus-expert differences in evaluation return at one million updates; error bars are 95% bootstrap confidence intervals over matched learner seeds. The dashed line marks zero effect.

6.2 EFFECT OF ROUTING THE RANDOM DATA

Figure 3 reports the successive routing contrasts defined in Section 5.2. HalfCheetah shows the clearest role-dependent pattern: moving from C2 to C3 substantially increases return at both expert-data budgets, while moving from C3 to C4 substantially decreases it. At 100k, the respective changes are $+5668.2$ and -4119.6; at 1k, they are $+1252.4$ and -1000.7, with all four intervals excluding zero. Hopper-1k also shows a positive C2→C3 effect ($+221.0$, 95% CI $[126.1, 318.2]$).

Walker2d-100k exhibits a different pattern: C1→C2 and C2→C3 decrease return by 3134.0 and 269.1, respectively, whereas C3→C4 increases it by 346.6; all three intervals exclude zero. Together, these controls show that the effect of the same additional data can change in magnitude and direction as its use changes, and that these effects vary across environments and data regimes.

6.3 COMPARISON WITH OFFLINE RL BASELINES

Table 1 provides performance context on the mixed datasets. Trust-Mix is competitive in several settings but is not uniformly the highest-performing method; the largest mean return is obtained by different methods across environments and budgets. Complete expert-only and mixed-data results are reported in Appendix C.4.

The CQL+DW-AW adaptation is included as an additional diagnostic comparison. Trust-Mix has a higher mixed-data mean in all twelve settings, but differences in backbone and preprocessing prevent attributing this comparison specifically to data weighting. We therefore do not claim superiority over CQL+DW-AW or over selective-regularization methods not directly evaluated here.

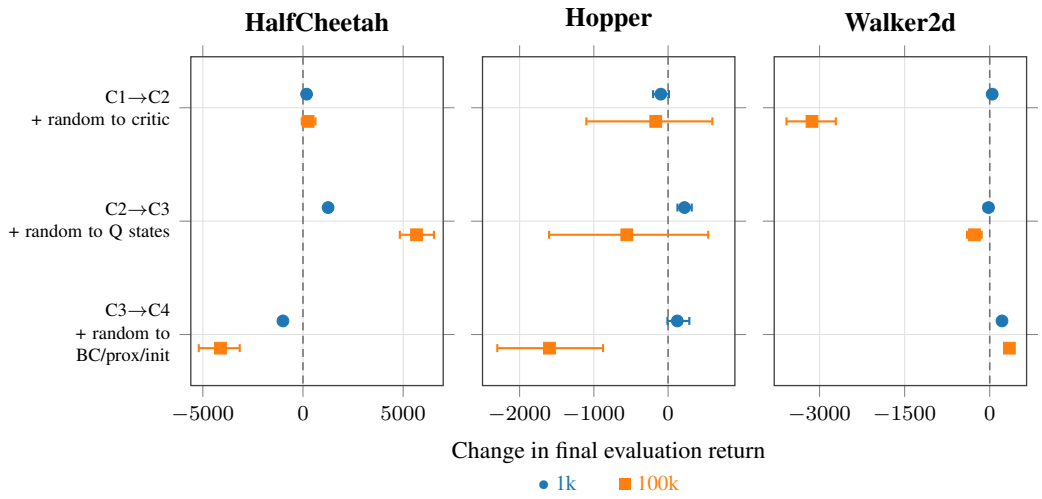

Figure 3: Matched-seed effects of successive source-aware routing changes at one million updates. Rows compare C2−C1 (+ random to critic), C3−C2 (+ random to Q states), and C4−C3 (+ random to BC/prox/init). Points are mean differences over ten matched learner seeds; horizontal bars are 95% paired-bootstrap intervals. Blue circles show 1k and orange squares show 100k expert budgets. Positive values indicate higher return in the next condition. Row labels apply to all panels; horizontal scales differ. The final contrast jointly changes BC/proximal supervision and initialization, not BC alone.

Table 1: Mixed-data return at one million updates (ten-seed means). Row maxima are bold. HC denotes HalfCheetah and Walker denotes Walker2d.

Env.	Budget	BC	CQL	IQL	TD3+BC	ReBRAC	MCQ	EDAC	CQL+ DW-AW*	Trust-Mix
HC	1k	5.6	2422.9	48.1	593.4	-0.1	2199.8	3055.8	-401.9	**3411.3**
HC	5k	6.5	2379.7	32.2	234.4	-12.8	1813.7	**3237.5**	-311.8	2419.0
HC	10k	-4.8	1630.7	62.4	469.5	-7.5	1791.5	**3144.9**	-259.0	1495.3
HC	100k	1.0	2475.3	160.1	2577.2	55.7	2036.8	**2970.8**	-379.5	1531.5
Hopper	1k	136.8	442.7	301.5	244.4	**450.2**	428.0	225.2	2.4	415.1
Hopper	5k	109.8	481.4	373.3	713.1	163.4	**726.7**	362.6	2.0	518.5
Hopper	10k	140.6	331.9	308.2	759.5	135.5	**791.7**	360.1	15.3	640.7
Hopper	100k	118.9	762.0	257.3	**801.9**	222.4	647.8	564.1	5.2	564.3
Walker	1k	60.0	128.8	**257.6**	169.8	7.9	156.3	252.8	97.4	137.6
Walker	5k	58.1	100.8	275.2	249.9	51.8	245.5	223.0	110.7	**372.0**
Walker	10k	58.2	236.4	279.7	141.3	25.5	191.6	63.7	67.3	**373.3**
Walker	100k	54.3	255.0	**584.4**	192.7	137.8	139.7	235.8	78.3	346.9

*PyTorch adaptation; not a verified reproduction or matched-backbone CQL ablation. See Appendix C.5.

7 Discussion and Limitations

Role-dependent data utility. The source-aware controls show that the effect of the same random-data component can change in magnitude and direction depending on how it enters learning. In HalfCheetah, extending random-data access to Q-guided policy-improvement states improves performance, while the subsequent extension to BC/proximal supervision and initialization reduces it. Walker2d-100k shows a different pattern. These results suggest that the utility of an offline data source cannot be characterized independently of the learning operation in which it is used.

This distinction also qualifies the meaning of "low-quality" data. Here, the random component is low quality with respect to its generating behavior, but its transitions, states, and recorded actions need not have equal utility for learning. Poor behavioral targets may still occur in transitions or at states that are useful for other learning operations. At the same time, the Walker2d results show

that broader access to random data is not always beneficial for critic learning or Q-guided policy improvement.

Implications and future directions. Trust-Mix implements one form of role-dependent data use: all transitions and states remain available to the critics and Q-guided actor update, while the influence of recorded actions on behavioral supervision is learned. Its performance and ablations show that this particular assignment is not uniformly preferable across environments or data regimes. For example, removing Q-guided policy improvement or replacing selective behavioral weights with unit weights has different effects across settings, and removing the Q-gradient cap substantially improves HalfCheetah-100k.

These observations suggest a broader direction in which the learner estimates not only how strongly to weight a recorded action, but also which learning operations should use a transition or state. Such decisions could depend on the current policy, critic estimates, uncertainty, dataset composition, or training stage. The present experiments do not determine which signals are sufficient for making these decisions, but the variation across environments and expert-data budgets suggests that fixed routing rules may leave useful information unexploited or expose some learning operations to harmful data. Future work could therefore study adaptive, role-specific data selection or weighting and whether the appropriate use of a sample changes during training.

Limitations. The routing controls are diagnostic interventions using source identities, not deployable source-label-free mechanisms. Their successive contrasts are conditional on the preceding routing choices and do not form an additive decomposition of performance. In addition, C3→C4 jointly changes BC/proximal supervision and initialization, so further controls are needed to separate these effects.

Our experiments are limited to three continuous-control environments and expert–random mixtures. Real offline datasets may contain intermediate-quality behavior, multiple behavior policies, nonstationarity, and more complex state distributions. Evaluating role-dependent data use under these forms of heterogeneity is therefore an important next step. Trust-Mix also relies on learned value estimates and critic disagreement for behavioral weighting; critic disagreement is a weighting signal here, not a guarantee of value accuracy.

Some expert-only results require additional investigation. In HalfCheetah, expert-only policies remain weak despite the high returns of the expert trajectories. A post-hoc analysis of saved evaluations (Appendix C.6) shows that this behavior is present from the first recorded evaluation at 50k updates, but does not identify its cause. We therefore report absolute returns alongside paired effects and do not use improvements over these policies as evidence of general superiority.

Finally, the baseline study provides performance context but does not include all closely related selective-regularization methods. The CQL+DW-AW comparison is an adaptation, not a verified matched reproduction. Broader comparisons, together with experiments that independently manipulate each data role, would help determine how broadly the observed role-dependent effects generalize.

8 Conclusion

We studied how the effect of additional offline data depends on the learning operation through which it enters the learner. Controlled routing experiments show that the same random-data source can have substantially different, and sometimes opposite, effects across critic learning, Q-guided policy improvement, and behavioral supervision, with patterns varying across environments and data regimes. Trust-Mix provides one source-label-free implementation of role-dependent data use. More broadly, our results motivate treating data utility in offline RL as role dependent and studying how these roles can be assigned adaptively during learning.

AI Use Statement

Generative AI tools were used to assist with research ideation and methodological refinement, experiment-design feedback, implementation and debugging, numerical analysis and interpreta-

tion, literature search and retrieval, drafting and editing, and preparation of figures, software, and manuscript artifacts. AI-assisted outputs were treated as suggestions rather than authoritative scientific evidence.

All reported experimental results were obtained from recorded experiment outputs, and numerical summaries were checked against the underlying experiment records and source artifacts. The dataset-distribution illustration is schematic and is not presented as a measured or fitted density. The authors reviewed the AI-assisted material and retain full responsibility for the final scientific claims, code, citations, measurements, figures, and supplementary materials.

Ethics Statement

The study uses stored transitions and simulated control environments. It does not involve human participants or claim safe deployment in a physical system. Learned values, weights, and gradient constraints are not safety certificates for real-world control.

Reproducibility Statement

Appendix A specifies the Trust-Mix architecture, objectives, initialization, hyperparameters, numerical safeguards, and update order. Appendix B documents the available dataset provenance, expert-subset construction checks, treatment of episode boundaries, evaluation protocol, and matched-seed bootstrap procedure. Appendix C.4 identifies the baseline implementation sources, recovered hyperparameters, preprocessing, and evaluation alignment with Trust-Mix, distinguishing verified settings from unresolved historical provenance. Appendix C.5 describes the independently written CQL+DW-AW PyTorch adaptation and its differences from the original paper and public implementation.

Appendix C provides endpoint returns, source-aware routing contrasts, component ablations, and saved learning histories. The post-hoc HalfCheetah analysis and its limitations are documented in Appendix C.6. The code supplement provides the main Trust-Mix implementation, fixed configurations, training and evaluation commands, input checks, and numerical/protocol tests. Accompanying audit records document source/configuration checks and numerical comparisons against the available experimental records.

These checks support verification of the reported summaries; they do not constitute independent reruns of all training experiments or fresh evaluations of every saved policy. Original processed training buffers and pretrained checkpoints are not included. Reproducing the training experiments additionally requires the corresponding input buffers and compatible software and runtime configurations. Unverified dataset-generation details, unresolved subset checks, and incomplete historical implementation provenance are explicitly identified in the appendix rather than inferred from published defaults.

References

Edoardo Cetin, Andrea Tirinzoni, Matteo Pirotta, Alessandro Lazaric, Yann Ollivier, and Ahmed Touati. Simple ingredients for offline reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6020–6047. PMLR, 2024. URL https://proceedings.mlr.press/v235/cetin24a.html.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/a8166da05c5a094f7dc03724b41886e5-Abstract.html.

Zhang-Wei Hong, Pulkit Agrawal, Rémi Tachet des Combes, and Romain Laroche. Harnessing mixed offline reinforcement learning datasets via trajectory weighting. In *International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=OhUAblg27z.

Zhang-Wei Hong, Aviral Kumar, Sathwik Karnik, Abhishek Bhandwaldar, Akash Srivastava, Joni K. Pajarinen, Romain Laroche, Abhishek Gupta, and Pulkit Agrawal. Beyond uniform sampling: Offline reinforcement learning with imbalanced datasets. In *Advances in Neural Information Processing Systems*, volume 36, 2023b. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/0ff3502bb29570b219967278db150a50-Abstract-Conference.html.

Xingyu Jiang, Ning Gao, Xiuhui Zhang, Hongkun Dou, and Yue Deng. Value-aligned behavior cloning for offline reinforcement learning via bi-level optimization. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/17ab5964d90f38b174a068646c958789-Abstract-Conference.html.

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

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. URL https://arxiv.org/abs/2005.01643.

Tenglong Liu, Yang Li, Yixing Lan, Hao Gao, Wei Pan, and Xin Xu. Adaptive advantage-guided policy regularization for offline reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 31406–31424. PMLR, 2024. URL https://proceedings.mlr.press/v235/liu24ai.html.

Qin-Wen Luo, Ming-Kun Xie, Ye-Wen Wang, and Sheng-Jun Huang. Learning to trust Bellman updates: Selective state-adaptive regularization for offline RL. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41450–41467. PMLR, 2025. URL https://proceedings.mlr.press/v267/luo25p.html.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. URL https://arxiv.org/abs/1910.00177.

Yunpeng Qing, Shunyu Liu, Jingyuan Cong, Kaixuan Chen, Yihe Zhou, and Mingli Song. A2PO: Towards effective offline reinforcement learning from an advantage-aware perspective. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/333a7697dbb67f09249337f81c27d749-Abstract-Conference.html.

Kajetan Schweighofer, Markus Hofmarcher, Marius-Constantin Dinu, Philipp Renz, Angela Bitto-Nemling, Vihang Patil, and Sepp Hochreiter. Understanding the effects of dataset characteristics on offline reinforcement learning. *arXiv preprint arXiv:2111.04714v1*, 2021. URL https://arxiv.org/abs/2111.04714v1. Version 1, submitted 8 November 2021.

Anikait Singh, Aviral Kumar, Quan Vuong, Yevgen Chebotar, and Sergey Levine. ReDS: Offline RL with heteroskedastic datasets via support constraints. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4b5d47949866d06ab5c03022b4a5a551-Abstract-Conference.html.

Yihao Sun and Zhilong Zhang. Offlinerl-kit: An elegant pytorch offline reinforcement learning library. https://github.com/yihaosun1124/OfflineRL-Kit, 2023.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/26cce1e512793f2072fd27c391e04652-Abstract-Conference.html.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024. URL https://arxiv.org/abs/2407.17032.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S. Merel, Jost Tobias Springenberg, Scott E. Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://papers.neurips.cc/paper_files/paper/2020/hash/588cb956d6bbe67078f29f8de420a13d-Abstract.html.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. URL https://arxiv.org/abs/1911.11361.

Yiqin Yang, Quanwei Wang, Chenghao Li, Hao Hu, Chengjie Wu, Yuhua Jiang, Dianyu Zhong, Ziyou Zhang, Qianchuan Zhao, Chongjie Zhang, and Bo Xu. Fewer may be better: Enhancing offline reinforcement learning with reduced dataset. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1530249fcd5e2996b6248c5d9fc0ab64-Abstract-Conference.html.

A TRUST-MIX IMPLEMENTATION DETAILS

This appendix specifies the implementation of Section 4. Trust-Mix uses one configuration across environments, expert budgets, and expert-only/mixed input conditions.

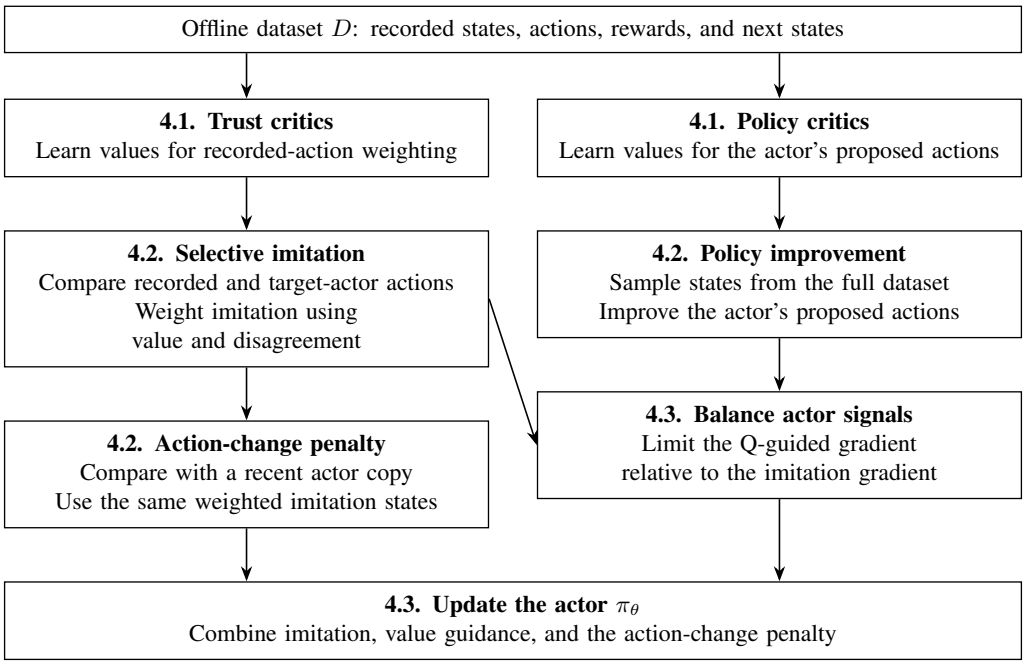

Figure 4: **Trust-Mix architecture.** Both critic pairs use the full input. The Q-guided actor branch uses an independent uniform state batch; BC and the proximal loss share learned behavioral weights. Numbers refer to the method subsections in Section 4.

A.1 NETWORKS AND POLICY ACTIONS

Each of the four critics is an independent network on concatenated state and action coordinates, with two 256-unit ReLU hidden layers and one scalar output. Corresponding trust and policy-critic heads start with identical parameters, use matched transition-sampling streams, and have separate optimizers.

The actor has two 256-unit ReLU hidden layers and separate mean and log-standard-deviation heads. With coordinatewise action center $b_a = (a_{\max} + a_{\min})/2$ and half range $h_a = (a_{\max} - a_{\min})/2$, it samples

$$u \sim \mathcal{N}(\mu_\theta(s), \mathrm{diag}(\sigma_\theta^2(s))), \qquad a = b_a + h_a \odot \tanh(u),$$

where $\log \sigma_\theta(s)$ is clipped to $[-5, 2]$. Evaluation, value guidance, and proximal regularization use the deterministic action

$$a_\theta(s) = b_a + h_a \odot \tanh(\mu_\theta(s)). \tag{10}$$

Target and proximal actors use the same action map with their own parameters. BC uses the transformed Gaussian density, including the inverse-$\tanh$ transformation and Jacobian. Recorded actions are normalized to the action range and clipped to $[-1 + 10^{-6}, 1 - 10^{-6}]$ before inversion; the Jacobian factor inside its logarithm is lower bounded by 10^{-6}.

A.2 TARGET AND REFERENCE UPDATES

All targets and references initially copy their trainable counterparts. After each actor step, the target actor is updated by

Table 2: Networks and reference copies. A bar denotes a target network; a tilde denotes an averaged trust critic used for behavioral weighting.

Object	Role	Update
π_θ	Trainable policy; output at one million iterations	Actor optimizer
Q_1^T, Q_2^T	Recorded-prefix value estimates	Trust-critic optimizer
Q_1^P, Q_2^P	Values used for direct actor improvement	Policy-critic optimizer
$\pi_{\bar{\theta}}$	Action comparison and critic bootstraps	Parameter averaging
$\bar{Q}_1^T, \bar{Q}_2^T$	Trust-critic bootstrap targets	Hard copy
$\bar{Q}_1^P, \bar{Q}_2^P$	Policy-critic bootstrap targets	Hard copy
$\widetilde{Q}_1^T, \widetilde{Q}_2^T$	Recorded-action weighting	Parameter averaging
π_{prox}	Anchor for the action-change penalty	Hard copy

$$\bar{\theta} \leftarrow (1-\tau)\bar{\theta} + \tau\theta, \qquad \tau = 0.005. \tag{11}$$

Critic bootstraps and behavioral scores use its pre-update parameters. Target critics $\bar{Q}_k^T$ and $\bar{Q}_k^P$ are hard-copied from their online counterparts every 250 iterations. After the initialization synchronization, each averaged trust critic is updated every 250 iterations by

$$\widetilde{\phi}^T \leftarrow 0.995\,\widetilde{\phi}^T + 0.005\,\phi^T. \tag{12}$$

The proximal actor is hard-copied every 5,000 post-initialization actor steps. Appendix A.5 specifies initialization synchronization; Appendix A.8 gives the order of updates. For every critic pair, the subscript $\min$ denotes the minimum of its two heads.

A.3 Multi-step trust targets

Trust critics request horizons $\mathcal{H} = \{1, 3, 5\}$. Reward prefixes follow consecutive recorded transitions and stop at termination, truncation, a recorded episode change, or the dataset end. Only a true termination removes the bootstrap.

For anchor i and requested horizon h, let $m_{i,h} \le h$ be the number of rewards available before the requested horizon or segment boundary. The target is

$$\begin{aligned} y_i^{T,(h)} = &\sum_{j=0}^{m_{i,h}-1} \gamma^j r_{i+j} \\ &+ \gamma^{m_{i,h}}(1 - d_{i+m_{i,h}-1})\bar{Q}_{\min}^T\big(s'_{i+m_{i,h}-1}, a_{\bar{\theta}}(s'_{i+m_{i,h}-1})\big). \end{aligned} \tag{13}$$

If a required transition is absent from D before a boundary, omit that requested horizon. Horizon one is valid for every sampled anchor. For the valid requested horizons $\mathcal{H}_i$, the mean target and trust-critic objective are

$$\begin{aligned} y_i^T &= \frac{1}{|\mathcal{H}_i|} \sum_{h \in \mathcal{H}_i} y_i^{T,(h)}, \\ L_{Q^T} &= \mathbb{E}_{i \sim D} \sum_{k=1}^{2} \big(Q_k^T(s_i, a_i) - y_i^T\big)^2. \end{aligned} \tag{14}$$

Each valid requested horizon has equal weight, including when two horizons shorten to the same boundary. Each critic incurs one squared error against the mean target. Targets are detached, and transition sampling uses the full input D rather than a behaviorally weighted subset.

A.4 POLICY-CRITIC TARGETS AND SAMPLED PESSIMISM

Policy critics use the one-step target and objective

$$\begin{aligned} y_i^P &= r_i + \gamma(1-d_i)\bar{Q}^P_{\min}\big(s'_i, a_{\bar{\theta}}(s'_i)\big), \\ L_{Q^P} &= \mathbb{E}_{i\sim D}\sum_{k=1}^{2}\big(Q_k^P(s_i,a_i) - y_i^P\big)^2 + \lambda_P(t)\sum_{k=1}^{2} L_{\text{pess},k}. \end{aligned} \tag{15}$$

For candidate-action multiset $\mathcal{C}_i$, the sampled penalty is

$$L_{\text{pess},k} = \mathbb{E}_i\left[\tau_P \log \sum_{a\in\mathcal{C}_i} \exp\left(\frac{Q_k^P(s_i,a)}{\tau_P}\right) - Q_k^P(s_i,a_i)\right]. \tag{16}$$

Targets are detached and d_i is the true-termination indicator. On iterations with positive pessimism coefficient, use the first 256 states of the uniform policy-critic batch. At each state, $\mathcal{C}_i$ contains 10 bounded-uniform actions and 10 stochastic target-actor actions, detached from that actor. Set $\tau_P = 1$ and

$$\lambda_P(t) = \text{clip}\left(\frac{t - 10{,}000}{40{,}000}, 0, 1\right). \tag{17}$$

Thus pessimism is zero through 10,000 iterations and reaches full strength at 50,000. The log-sum contains 20 unweighted exponentials, without sampling-density correction or candidate-count normalization. The TD term is averaged over the full batch and the pessimistic term over its 256-state subset. Both critic pairs use batch size 1024, learning rate 3×10^{-4}, and gradient clipping to norm 10.

A.5 BEHAVIORAL-CLONING INITIALIZATION

Iterations 1–5,000. Sample uniformly from $I_r = \{i : r_i \geq \text{median}_{j\in D}\, r_j\}$ and minimize unweighted negative log likelihood with actor learning rate 3×10^{-4}. The implementation's robust reward normalization and zero threshold produce this same set for finite rewards. Both critic pairs train on D throughout initialization; the target actor is updated after each actor step using Eq. (11).

Iterations 5,001–10,000. Save the actor a_{θ_5} at iteration 5,000. Using the online trust critics, compute

$$A_i^{\text{init}} = Q^T_{\min}(s_i,a_i) - Q^T_{\min}(s_i, a_{\theta_5}(s_i)), \qquad U_i^{\text{init}} = |Q_1^T(s_i,a_i) - Q_2^T(s_i,a_i)|,$$

and select

$$I = \{i : A_i^{\text{init}} > 0.25 s_A^{\text{init}}, \quad U_i^{\text{init}} \leq q_{0.75}(U^{\text{init}})\}.$$

The dataset-wide advantage scale uses the median absolute deviation, population-standard-deviation fallback, and 10^{-6} safeguard of Appendix A.6. Initialization medians average the two middle observations for even sample counts. Selected actions receive the positive-branch weights of Eq. (3), evaluated using these dataset-wide scores.

Sample $i \in I$ with probability proportional to $\sqrt{w_i^{\text{init}}}$ and multiply its negative log likelihood by $\sqrt{w_i^{\text{init}}}$, dividing by the sum of sampled multipliers. The set and weights remain fixed until iteration 10,000; an empty set raises an error.

Synchronization. At iteration 10,000, retain the actor parameters but reset its Adam optimizer to learning rate 10^{-5}. Copy the actor into the target and proximal actors, and the online trust critics into the averaged trust critics. Subsequent behavioral candidates are sampled uniformly from D and weighted using Eq. (3).

A.6 ACTOR BATCHES AND SCORE NORMALIZATION

After initialization, B_t contains 1024 indices sampled uniformly with replacement from D. Independently, C_t contains 1024 uniformly sampled states from D, without filtering or weighting by behavioral trust. Scores use the averaged trust critics available at the actor step and the target actor before that step.

All advantages in B_t, including non-positive values, enter the scale calculation:

$$\begin{aligned} m_A &= \mathrm{median}_{i\in B_t} A_i(t), \\ s_{\mathrm{MAD}} &= 1.4826\,\mathrm{median}_{i\in B_t} |A_i(t) - m_A|, \\ s_{\mathrm{SD}} &= \left[|B_t|^{-1} \sum_{i\in B_t} \left(A_i(t) - \overline{A}_{B_t}\right)^2 \right]^{1/2}. \end{aligned} \tag{18}$$

Here $\overline{A}_{B_t}$ is the batch mean. Set $s_A = s_{\mathrm{MAD}}$ when finite and greater than 10^{-6}; otherwise use s_{SD} if it satisfies those conditions, and 10^{-6} otherwise. Actor-batch medians use the lower middle observation for even sample counts. The disagreement scale is

$$u_{75}(t) = \max\left\{ q_{0.75}\left(\{U_i(t) : i \in B_t\}\right), 10^{-6} \right\}. \tag{19}$$

The quantile is linearly interpolated. The median enters the scale calculation only; the admission threshold remains $A_i(t) > 0$, without median-centering. Weighting constants are $\beta_A = \beta_U = 0.50$, $w_{\min} = 0.05$, and $\epsilon_Z = 10^{-6}$. Non-positive advantages therefore receive zero weight, while positive advantages receive weights in $[0.05, 1]$. Scores, scales, and weights are recomputed and detached at every actor step.

BC and the proximal loss share B_t, its detached weights, and the normalizing denominator. The proximal actor is fixed between copies. For the Q objective, policy-critic parameters are fixed while gradients through the actor's proposed action are retained.

A.7 ACTOR-GRADIENT SCALING AND NUMERICAL CASES

The three gradients in Section 4.3 are computed separately. Use $\lambda_{\mathrm{prox}} = 0.10$, $\rho_Q = 0.50$, and

$$\alpha_{\mathrm{BC}}(t) = 1 - 0.5\frac{t - 10{,}000}{1{,}000{,}000 - 10{,}000}, \qquad 10{,}000 < t \le 10^6. \tag{20}$$

For $\epsilon_g = 10^{-8}$, the numerical form of the Q/BC cap is

$$c_Q = \begin{cases} 1, & \|g_Q\|_2 \le \epsilon_g, \\ 0, & \|g_{\mathrm{BC}}\|_2 \le \epsilon_g \text{ and } \|g_Q\|_2 > \epsilon_g, \\ \min\left(1, \dfrac{0.50\|g_{\mathrm{BC}}\|_2}{\|g_Q\|_2}\right), & \text{otherwise.} \end{cases} \tag{21}$$

No higher-order differentiation is taken through the gradient norms or c_Q. For finite gradients,

$$\|c_Q g_Q\|_2 \le \max\{\epsilon_g, 0.50\|g_{\mathrm{BC}}\|_2\}.$$

This bound applies before combined clipping and Adam preconditioning. The three components are combined as in Eq. (9), clipped to norm 10, and supplied to Adam.

If all behavioral weights are zero and outputs are finite, BC and proximal gradients vanish and the cap suppresses any Q gradient above ϵ_g. Adam is still called, so stored moments may move parameters even when the supplied gradient is zero. The cap does not repair non-finite values or gradients.

A.8 Update order and output

Algorithm 1 gives the update order. Both critic pairs train from the first iteration. Reference updates due at an iteration precede behavioral scoring; the target-actor update follows the actor step. The output is the trainable actor at one million total iterations, including initialization.

Algorithm 1: Trust-Mix

Input: offline dataset D and Table 3.

1. Initialize the actor, four critics, and target/reference copies. Perform the two-stage initialization and synchronization in Appendix A.5.
2. For $t = 10{,}001, \ldots, 1{,}000{,}000$:
   (a) Update trust critics using Eq. (14). On multiples of 250, hard-copy their targets and update their averaged copies using Eq. (12).
   (b) Update policy critics using Eq. (15). On multiples of 250, hard-copy their targets after this step.
   (c) Draw B_t, compute the scores and detached weights in Eqs. (1) and (3), and independently draw C_t from the full input.
   (d) Differentiate the BC, Q, and proximal losses separately. Apply Eq. (21), combine the gradients, clip to norm 10, and update the actor with Adam.
   (e) Update the target actor using Eq. (11). Copy the proximal actor every 5,000 post-initialization actor steps.
   (f) Evaluate at scheduled reporting steps without using returns for learning or checkpoint selection.

Output: the actor at one million iterations, using Eq. (10) for evaluation.

A.9 Hyperparameters

Table 3 specifies the shared configuration.

A.10 Interpretation of the weighting and update rules

The action advantage compares learned values, not exact on-policy returns. Critic disagreement is not calibrated uncertainty, and the sampled pessimistic penalty is not a guaranteed lower bound. Batch-based score scales and normalization define a stochastic weighted objective, without an unbiasedness claim for a normalized full-dataset objective. The positive weight floor retains imitation influence even at high disagreement. Gradient capping limits magnitude rather than directional accuracy; proximal regularization is a soft penalty, not a hard movement constraint.

B Data and Reporting Details

B.1 Stored inputs and pairing

For Trust-Mix, at each environment and expert budget, the expert-only input is the expert slice of its matched mixed buffer; the mixed input adds one million random-source transitions. Source labels define these input conditions and the diagnostic controls, but do not enter Trust-Mix learning. Ten learner seeds train on each fixed input.

Trust-Mix loads float32 observations, actions, rewards, and next observations, checks finiteness and action bounds, and retains true termination and recorded-boundary information. It does not standardize observations or normalize TD rewards. Its initialization reward transformation only determines the median-threshold BC set. Baseline-specific preprocessing is in Appendix C.4.

Dataset sources. Table 4 lists the Minari expert-source identifiers recorded in the buffers. Collection/export metadata specify `random_seed=0`, Minari 0.5.3, Gymnasium 1.3.0, MuJoCo 3.2.3, and NumPy 1.26.4.

Table 3: Trust-Mix hyperparameters for the 240-run evaluation.

Setting	Value
Hidden layers / units per layer / activation	2 / 256 / ReLU
Discount γ	0.99
Critic, behavioral, and Q-state batch sizes	1024 each
Critic learning rate	3×10^{-4}
Actor learning rate: initialization / later	3×10^{-4} / 10^{-5}
Actor log-standard-deviation bounds	$[-5, 2]$
Optimizer / gradient clipping	Adam / norm 10
Reward-based BC stage / total BC initialization	5,000 / 10,000 iterations
Initialization admission: advantage / disagreement	$> 0.25 s_A$ / $\leq q_{0.75}$
Initialization sampling and loss-weight powers	0.50 / 0.50
Requested trust-target horizons	$1, 3, 5$
Target-actor averaging rate / interval	0.005 / every actor step
Target-critic copy interval	250 iterations
Trust-critic averaging rate / interval	0.005 / 250 iterations
Proximal coefficient / copy interval	0.10 / 5,000 actor steps
BC coefficient after initialization	Linear $1 \rightarrow 0.5$
Q/BC norm-ratio cap / norm tolerance	0.50 / 10^{-8}
Post-initialization candidate sampling	Uniform, with replacement
Weight computation interval	Every actor step
Non-positive score weight / positive weight range	0 / $[0.05, 1]$
Advantage temperature / disagreement coefficient	0.50 / 0.50
Disagreement scale quantile / scale safeguard	0.75 / 10^{-6}
BC and proximal denominator safeguard	10^{-6}
Pessimism start / full strength	10,000 / 50,000 iterations
Maximum pessimism coefficient / temperature	1 / 1
Pessimism state batch	256
Uniform / target-policy candidate actions per state	10 / 10
Training horizon / reporting interval	1,000,000 / 50,000 iterations
Evaluation episodes / reset seeds	20 / 100000–100019
Resumable checkpoint interval	250,000 iterations

Table 4: Minari expert-source identifiers and evaluation environments.

Environment	Recorded expert source	Evaluation environment
HalfCheetah	`mujoco/halfcheetah/expert-v0`	`HalfCheetah-v5`
Hopper	`mujoco/hopper/expert-v0`	`Hopper-v5`
Walker2d	`mujoco/walker2d/expert-v0`	`Walker2d-v5`

The mixed buffers retain source and termination/truncation flags. Segments are reconstructed from these flags and observation continuity; stored segments can include budget-cut fragments rather than complete episodes.

Nesting across expert budgets. The saved audit compares the expert components with ordered prefixes of the 100k component using dtype-sensitive composite fingerprints (Table 5). All checks pass for HalfCheetah and Walker2d, supporting

$$D_E^{(1k)} \subseteq D_E^{(5k)} \subseteq D_E^{(10k)} \subseteq D_E^{(100k)}$$

in those buffers. Hopper passes at 1k but fails at 5k and 10k; full cross-budget nesting is therefore not assumed for Hopper. The composite-fingerprint checks do not identify which fields differ.

Table 5: Ordered-prefix checks against the 100k expert component. A failed composite-fingerprint check does not identify the differing field.

Environment	1k prefix	5k prefix	10k prefix
HalfCheetah	Pass	Pass	Pass
Hopper	Pass	Fail	Fail
Walker2d	Pass	Pass	Pass

The random component has the same ordered core-array fingerprint across all four budgets within each environment and is reused, not independently recollected. Cross-budget nesting is distinct from within-budget expert-only/mixed pairing.

Standalone expert inputs. The BC/CQL/IQL/TD3+BC data generator constructs a seed-0 shuffled stream of Minari episodes and takes nested prefixes of exactly 1k, 5k, 10k, and 100k transitions. The final transition at each budget is marked as a timeout, retaining its next observation and true-terminal flag. These within-stream prefixes are distinct from the mixed-buffer checks in Table 5. Standalone files store observations, next observations, and rewards as float64; the training loader converts them to float32.

B.2 Stored dataset behavior

Tables 6–8 describe the stored segments, including budget-cut fragments. Segment returns are source-behavior measurements, not evaluations of the learned policies.

Table 6: Recorded segment counts. Budgets count transitions, not complete episodes. The random component is reused across all four budgets within each environment.

	Expert-transition budget				Random
Environment	1k	5k	10k	100k	1M
HalfCheetah	1	5	10	100	1,000
Hopper	1	6	12	111	44,866
Walker2d	1	5	10	100	47,475

All 1,000 HalfCheetah random segments end in stored truncations. Hopper has 44,865 terminal-ended random segments and one truncation; Walker2d has 47,474 and one. These flags do not establish the cause of termination. Shared starts in Table 8 mean identical observation vectors, not necessarily identical full simulator states.

In 50,000-transition samples of each random component, action-coordinate means range from -0.0040 to 0.0029, sample SDs from 0.5746 to 0.5784, and absolute within-segment lag-one correlations are at most approximately 0.016. These statistics do not verify the action generator or complete state–action coverage.

Table 7: Stored segment lengths and returns: mean ± sample SD across reconstructed segments at expert-100k and random-1M. Budget-cut segments remain included.

Environment	Component	Segment length	Segment return
HalfCheetah	Expert 100k	1000.00 ± 0.00	16284.92 ± 341.43
HalfCheetah	Random 1M	1000.00 ± 0.00	-285.21 ± 80.22
Hopper	Expert 100k	900.90 ± 184.00	3776.64 ± 760.28
Hopper	Random 1M	22.29 ± 12.11	17.56 ± 17.55
Walker2d	Expert 100k	1000.00 ± 0.00	6849.72 ± 59.05
Walker2d	Random 1M	21.06 ± 8.62	1.62 ± 5.98

Table 8: Distinct stored segment starts and exact expert–random overlap at expert-100k.

Environment	Expert starts	Random starts	Shared expert starts
HalfCheetah	100	1,000	87/100
Hopper	111	44,866	111/111
Walker2d	100	47,475	100/100

B.3 Training input

All transitions available to the selected Trust-Mix input condition are available for learning, with no held-out gate reserve: $D_{\text{learn}} = D_{\text{available}}$. Multi-step prefixes remain within the constructed input and recorded segment boundaries.

B.4 Endpoint statistics and paired uncertainty

Evaluations use deterministic actions in the corresponding Gymnasium MuJoCo-v5 environment every 50,000 training iterations. Each policy score is the mean raw, undiscounted return of 20 episodes with reset seeds 100000–100019. Episodes end on termination or time-limit truncation. The reported policy is always the actor at exactly one million iterations; intermediate evaluations do not select a checkpoint.

For n learner-seed scores, report the arithmetic mean and sample SD:

$$\overline{R} = n^{-1} \sum_{k=1}^{n} R_k, \qquad s_R = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} (R_k - \overline{R})^2}.$$

This SD concerns per-policy episode means, not pooled episodes. Main, routing, and CQL+DW-AW conditions use learner seeds 0–9; component ablations use seeds 0–4. For matched environment e, budget b, and seed k, define

$$\Delta_{e,b,k} = R^M_{e,b,k} - R^E_{e,b,k}. \tag{22}$$

Resample complete learner-seed pairs with replacement, calculate their mean difference, and take the 2.5th and 97.5th percentiles over 20,000 resamples with resampling seed 1729. The same paired procedure is used for successive routing contrasts and CQL+DW-AW composition effects. Component effects use the matching five parent seeds. Differences and intervals are computed before rounding.

Every prespecified seed is retained in reporting. Low completed returns, numerical failures, and unfinished runs are distinguished; a retry remains the same learner seed rather than an additional observation.

B.5 Learning-path measurements

Learning curves describe the twenty scheduled evaluations from 50k to 1M, not convergence or checkpoint selection. Pointwise uncertainty and the descriptive path audit are specified in Appendix C.6.

B.6 Randomness and checkpoint recovery

Python, NumPy, and PyTorch generators are seeded. Deterministic PyTorch execution is requested and cuDNN benchmarking is disabled. Actor initialization/sampling use learner-seed offset 1000; critic initialization and matched critic-sampling streams use offset 4000. Paired input conditions use the same offsets.

Checkpoints retain networks and reference copies, optimizers, counters, and random-generator state. Compatible checkpoints resume at the saved iteration plus one. Different GPU models can still produce different numerical trajectories despite deterministic settings.

C Additional Experimental Results

C.1 Complete paired Trust-Mix endpoints

Table 9 summarizes all 240 Trust-Mix runs behind Figure 2, with ten matched pairs per setting. Rounding can make the displayed difference differ by 0.1 from the subtraction of displayed means.

Table 9: Trust-Mix final returns at one million updates: mean ± sample SD over ten seeds, with paired effects and 95% intervals.

Environment	Budget	Expert only	Expert+random	Δ	95% CI
HalfCheetah	1k	-508.8 ± 109.9	3411.3 ± 217.5	$+3920.1$	$[+3787.1, +4063.9]$
HalfCheetah	5k	-450.3 ± 68.2	2419.0 ± 1149.9	$+2869.3$	$[+2165.3, +3528.1]$
HalfCheetah	10k	-409.8 ± 97.8	1495.3 ± 1382.0	$+1905.1$	$[+1118.2, +2774.5]$
HalfCheetah	100k	-226.4 ± 111.1	1531.5 ± 265.9	$+1757.9$	$[+1624.4, +1901.0]$
Hopper	1k	26.6 ± 22.6	415.1 ± 476.3	$+388.5$	$[+206.8, +708.6]$
Hopper	5k	516.1 ± 459.3	518.5 ± 228.0	$+2.4$	$[-363.7, +366.3]$
Hopper	10k	153.0 ± 291.3	640.7 ± 377.8	$+487.6$	$[+176.0, +796.4]$
Hopper	100k	758.7 ± 895.7	564.3 ± 353.8	-194.3	$[-626.8, +226.3]$
Walker2d	1k	78.6 ± 106.8	137.6 ± 53.4	$+59.0$	$[-28.6, +135.6]$
Walker2d	5k	295.4 ± 165.4	372.0 ± 338.1	$+76.6$	$[-132.7, +303.3]$
Walker2d	10k	558.6 ± 399.0	373.3 ± 349.3	-185.3	$[-425.1, +91.7]$
Walker2d	100k	719.6 ± 471.2	346.9 ± 363.6	-372.7	$[-770.6, +85.8]$

C.2 Source-aware data-role controls

Table 10 defines C1–C4 in the order used in Section 5.2. All four controls are separately trained and use unit BC/proximal weights, including during the 10,000-update initialization. Unlike Trust-Mix, they use source labels for diagnostic routing. The study has 240 runs: three environments, the 1k/100k expert budgets, four conditions, and ten seeds.

Tables 11 and 12 report endpoints and matched-seed contrasts. C2−C1 adds random transitions to critic learning; C3−C2 adds random states to the Q-guided actor branch; C4−C3 adds random-source access to BC/proximal supervision and initialization. These effects are conditional on preceding routing choices, not order-independent component attributions. The final contrast does not isolate behavioral cloning.

The role-dependent signs differ across environments. Both HalfCheetah Q-state contrasts are positive and both subsequent behavioral-branch contrasts negative, with intervals excluding zero. Walker2d's behavioral-branch contrasts are positive at both budgets; at 100k, however, this gain

Table 10: Source-aware routing conditions. The final column jointly specifies BC data, proximal states, and initialization data. All behavioral weights are one.

Configuration	Critic training	Actor Q-update states	BC/proximal and initialization
C1 (Expert only)	D_E	D_E	D_E
C2 (Mixed critics)	$D_E \cup D_R$	D_E	D_E
C3 (Mixed critics and Q states)	$D_E \cup D_R$	$D_E \cup D_R$	D_E
C4 (Mixed throughout)	$D_E \cup D_R$	$D_E \cup D_R$	$D_E \cup D_R$

Table 11: Routing-control returns at one million updates: mean and sample SD over ten seeds per condition.

Environment	Budget	C1 Expert only	C2 Mixed critics	C3 Mixed critics + Q states	C4 Mixed throughout
HalfCheetah	1k	-431.2 ± 74.7	-251.7 ± 130.2	1000.7 ± 102.6	-0.1 ± 0.7
HalfCheetah	100k	144.9 ± 334.8	401.3 ± 535.4	6069.5 ± 1602.1	1949.9 ± 416.2
Hopper	1k	157.2 ± 139.3	59.5 ± 84.0	280.5 ± 141.4	404.5 ± 195.2
Hopper	100k	2819.6 ± 433.6	2653.3 ± 1392.0	2096.9 ± 1127.3	498.6 ± 294.2
Walker2d	1k	38.7 ± 32.8	82.1 ± 88.3	60.3 ± 96.5	276.6 ± 33.3
Walker2d	100k	3426.2 ± 613.1	292.2 ± 199.0	23.0 ± 61.8	369.7 ± 50.1

only partly offsets the preceding critic and Q-state losses. These controls do not establish a universally preferred routing.

Table 12: Successive routing effects at one million updates: mean paired change and 95% percentile-bootstrap interval over ten matched seeds. Positive values favor the next routing condition. C4−C3 jointly changes BC/proximal supervision and initialization.

Environment	Budget	C2−C1 Critic training	C3−C2 Q-update states	C4−C3 BC/proximal/init
HalfCheetah	1k	$+179.5$ $[+96.1, +258.7]$	$+1252.4$ $[+1159.7, +1342.8]$	-1000.7 $[-1056.0, -935.4]$
HalfCheetah	100k	$+256.4$ $[-61.4, +620.8]$	$+5668.2$ $[+4835.8, +6537.3]$	-4119.6 $[-5203.7, -3158.5]$
Hopper	1k	-97.6 $[-202.2, +11.6]$	$+221.0$ $[+126.1, +318.2]$	$+124.0$ $[-9.0, +286.2]$
Hopper	100k	-166.3 $[-1101.1, +596.4]$	-556.4 $[-1602.8, +541.2]$	-1598.3 $[-2300.7, -875.9]$
Walker2d	1k	$+43.4$ $[-7.5, +107.7]$	-21.7 $[-103.6, +53.9]$	$+216.3$ $[+145.9, +277.1]$
Walker2d	100k	-3134.0 $[-3582.9, -2713.5]$	-269.1 $[-399.4, -141.5]$	$+346.6$ $[+288.0, +404.2]$

C.3 Final-parent component ablations

The component study changes one learning rule after the shared 10,000-update initialization (Table 13). It contains 150 exact-1M runs: five ablations, three environments, the 1k/100k budgets, and seeds 0–4. Each is paired with Trust-Mix using the same environment, budget, and seed.

Table 13: Component changes applied after the shared initialization.

Ablation	Change relative to Trust-Mix
Unit behavioral weights	Set all candidate BC and proximal weights to one.
Advantage-only weights	Remove disagreement attenuation ($\beta_U = 0$), retaining the positive-advantage test and weight floor.
No Q-guided actor update	Remove g_Q from the actor update; retain critic learning.
No Q/BC gradient cap	Set $c_Q = 1$; retain clipping of the combined gradient.
No proximal penalty	Set $\lambda_{\text{prox}} = 0$.

Table 14 uses the same five parent seeds for each effect, not the main benchmark's ten-seed mean. Negative values favor the full parent.

Table 14: Component effects on mixed inputs at one million updates: $\Delta = R_{\text{ablation}} - R_{\text{Trust-Mix}}$ and 95% paired percentile-bootstrap intervals over five matched seeds. Negative values favor the full parent.

Cell	Unit weights	Adv.-only	No Q actor	No Q/BC cap	No prox.
HalfCheetah-1k	-696.5 $[-848.8, -544.3]$	-460.0 $[-668.3, -251.8]$	-903.1 $[-1160.8, -695.0]$	$+119.7$ $[-92.6, +258.5]$	-558.4 $[-796.7, -320.1]$
HalfCheetah-100k	$+35.3$ $[-751.6, +822.2]$	-98.4 $[-316.3, +119.4]$	-722.5 $[-1097.6, -227.2]$	$+2003.3$ $[+1524.5, +2477.8]$	$+14.3$ $[-232.4, +268.6]$
Hopper-1k	-268.2 $[-896.2, +103.7]$	-206.0 $[-929.8, +335.1]$	-255.4 $[-859.6, +83.9]$	-202.2 $[-813.8, +162.2]$	-304.6 $[-899.1, +39.4]$
Hopper-100k	-46.2 $[-403.4, +431.8]$	-275.3 $[-513.4, -18.5]$	-383.9 $[-502.3, -271.6]$	$+208.4$ $[-43.7, +500.1]$	-317.9 $[-537.5, +34.1]$
Walker2d-1k	$+320.0$ $[+62.4, +631.1]$	$+157.4$ $[+1.1, +397.0]$	$+223.0$ $[+154.6, +282.9]$	$+43.5$ $[-99.2, +172.7]$	$+140.3$ $[+18.9, +271.0]$
Walker2d-100k	-39.0 $[-405.9, +238.2]$	-138.4 $[-518.7, +89.1]$	-99.8 $[-468.9, +141.3]$	-52.4 $[-597.7, +461.9]$	-147.0 $[-531.8, +101.1]$

Effects are regime dependent. Removing Q-guided actor updates lowers returns in HalfCheetah-1k, HalfCheetah-100k, and Hopper-100k, but raises Walker2d-1k; these intervals exclude zero. Unit

weights, advantage-only weights, and no proximal penalty also hurt HalfCheetah-1k while helping Walker2d-1k. Removing the Q/BC cap improves HalfCheetah-100k by $+2003.3$; its other intervals include zero. The study therefore does not support uniform necessity of every retained component.

C.4 Absolute baseline endpoint returns

Tables 18 and 19 report expert-only and mixed means and sample SDs over ten seeds. Table 17 gives differences between each method's condition means. Standard-baseline differences use the supplied summary precision; paired intervals are not inferred from marginal SDs.

Implementation sources. BC, CQL, IQL, TD3+BC, MCQ, and EDAC use OfflineRL-Kit (Sun & Zhang, 2023), available at `https://github.com/yihaosun1124/OfflineRL-Kit`. ReBRAC uses a JAX/Flax adaptation of CORL's `algorithms/offline/rebrac.py` (Tarasov et al., 2023), available at `https://github.com/tinkoff-ai/CORL`. CQL+DW-AW uses the separate adapter in Appendix C.5.

Common evaluation protocol. All baseline evaluations, for both expert-only and mixed-data runs, follow the same protocol as Trust-Mix. Policies are evaluated deterministically in the corresponding Gymnasium MuJoCo-v5 environment using 20 episodes with fixed reset seeds 100000–100019, every 50,000 training iterations. Each episode ends at true termination or time-limit truncation, and its score is the raw, undiscounted sum of environment rewards. The reported result for each learner seed is the mean return of the final policy at exactly one million training iterations; no best-checkpoint selection, return-based early stopping, or averaging over multiple checkpoints is used. Baseline comparisons use learner seeds 0–9.

The shared evaluation rule and training-iteration horizon do not imply identical learning objectives, preprocessing, or compute. TD3+BC and ReBRAC delay actor updates, while Trust-Mix includes initialization within its horizon. Method-specific observation transformations are also applied at evaluation; environment rewards remain raw.

Hyperparameters and preprocessing. Tables 15 and 16 give the implementation settings. BC, CQL, IQL, and TD3+BC use minibatches of 256 transitions. The critic-based configurations use $\gamma = 0.99$ and target averaging $\tau = 0.005$. CQL, MCQ, and EDAC use automatic entropy adjustment with target $-\dim(\mathcal{A})$.

Table 15: OfflineRL-Kit BC, CQL, IQL, and TD3+BC configuration. All four use batch size 256. Learning rates use Adam.

Method	Architecture and important hyperparameters	Preprocessing and reporting distinction
BC	Deterministic actor, widths 256,256; Adam actor rate 3×10^{-4}.	Float32 observations/actions/rewards; no observation or reward normalization.
CQL	Actor and twin critics, widths 256,256,256; actor/entropy rate 10^{-4}, critic rate 3×10^{-4}; CQL coefficient 5, temperature 1, 10 repeated actions. Automatic entropy tuning; no Lagrange or max-Q backup; deterministic backup enabled.	Raw observations and rewards. This differs from the separate CQL+DW-AW backbone in Appendix C.5.
IQL	Actor, twin Q critics and value network, widths 256,256; Adam rates 3×10^{-4}; expectile 0.7, temperature 3.0; orthogonal linear initialization; cosine actor-rate schedule, advanced once per 1,000 learning iterations.	No observation standardization. IQL applies trajectory-return-range scaling to training rewards. Evaluation rewards remain raw.
TD3+BC	Actor and twin critics, widths 256,256; Adam rates 3×10^{-4}; coefficient 2.5; target-policy noise 0.2, noise clip 0.5; actor update interval 2.	Observations standardized by dataset mean and population SD plus 10^{-3}; raw rewards. The same observation statistics are used for replay samples and policy inputs at evaluation.

Table 16: MCQ, EDAC, and ReBRAC configurations for expert-only and mixed inputs. All use raw rewards and no observation standardization. Learning rates use Adam; EDAC's condition-dependent diversity coefficients are shown explicitly.

Method / implementation	Networks, optimizers, and batch	Method-specific configuration
MCQ / OfflineRL-Kit	Actor and twin critics: 400,400; actor dropout 0.1; actor/critic/entropy rates 3×10^{-4}; batch 256. Behavior VAE: hidden width 750, latent size $2\dim(\mathcal{A})$, rate 10^{-3}.	$\lambda = 0.9$; 10 sampled actions; automatic entropy adjustment with target $-\dim(\mathcal{A})$. Same numerical configuration for expert-only and mixed inputs. Dropout is disabled for evaluation.
EDAC / OfflineRL-Kit	Actor/critics: 256,256,256; actor/entropy rate 10^{-4}, critic rate 3×10^{-4}; batch 256; 10/50/10 critics for HalfCheetah/Hopper/Walker2d.	Expert-only diversity coefficients: 5/1/5; mixed coefficients: 1/1/1 in the same environment order. Automatic entropy adjustment; max-Q and deterministic backup both disabled.
ReBRAC / CORL adaptation	Deterministic actor and twin critics: three hidden layers of width 256; actor/critic rates 10^{-3}; batch 1024; critic layer normalization on, actor layer normalization off.	Actor and critic BC coefficients both 1.0; target-policy noise 0.2, noise clip 0.5; actor update interval 2; Q-magnitude normalization on. State and reward normalization off. Same configuration for both inputs.

Training differences relevant to comparisons. EDAC uses diversity coefficient 5 for expert-only HalfCheetah and Walker2d, versus 1 for their mixed inputs; Hopper uses 1 in both. Those EDAC composition comparisons therefore combine a data change with a change in the diversity coefficient. Trust-Mix uses the same configuration in both input conditions.

ReBRAC requires a recorded next action for its critic BC term. For $i < N - 1$, its valid-row rule is

$$\neg\text{timeout}_i \ \wedge \ \left(d_i \ \vee \ \|s_{i+1} - s'_i\|_2 \leq 10^{-5}\right).$$

It drops timeout rows, nonterminal discontinuities, and the final buffer row; true-terminal rows may be retained with the next-action target masked. Thus it uses the same source buffers but not necessarily the same retained transitions as Trust-Mix. The four-method OfflineRL-Kit loader passes the stored terminal indicators to the replay buffer.

Table 17: Mean mixed-minus-expert change at one million updates, using each method's own ten-seed expert-only reference. Row maxima are bold; these are differences of condition means, not significance tests. HC denotes HalfCheetah.

Env.	Budget	BC	CQL	IQL	TD3+BC	ReBRAC	MCQ	EDAC	Trust-Mix
HC	1k	+412.2	+2447.0	+493.4	+1192.9	+221.8	+2220.6	+3095.5	**+3920.1**
HC	5k	+557.3	+2543.8	+621.8	+726.7	+475.8	+1921.3	**+3365.0**	+2869.3
HC	10k	+559.4	+1882.3	+618.6	+982.4	+525.5	+1669.3	**+3404.0**	+1905.1
HC	100k	+150.6	+2487.2	−3356.6	+2162.7	+359.0	+2068.8	**+3051.3**	+1757.9
Hopper	1k	−119.7	+286.7	−60.2	−16.5	−151.8	**+401.1**	+76.5	+388.5
Hopper	5k	−209.2	−38.3	+26.3	−185.9	−881.9	**+553.4**	−20.7	+2.4
Hopper	10k	−1274.9	−763.0	−822.0	−1385.9	−814.8	**+734.1**	−209.2	+487.6
Hopper	100k	−3775.3	−1991.2	−3631.5	−2769.6	−3219.1	**+539.2**	+247.4	−194.3
Walker2d	1k	−2.3	−88.9	**+201.3**	+70.6	−283.9	−115.3	+187.4	+59.0
Walker2d	5k	+31.1	−491.5	**+247.1**	+155.6	−248.5	−50.5	+178.5	+76.6
Walker2d	10k	−30.8	−737.5	**+201.2**	−233.7	−112.3	−136.6	+29.0	−185.3
Walker2d	100k	−6795.0	−3103.6	−6239.6	−6244.2	−5110.9	−155.1	**+194.8**	−372.7

Table 18: Expert-only returns at one million updates. Each entry gives the ten-seed mean above its sample SD; row maxima are bold. HC denotes HalfCheetah.

Env.	Budget	BC	CQL	IQL	TD3+BC	ReBRAC	MCQ	EDAC	Trust-Mix
HC	1k	−406.6 ±87.5	−24.1 ±100.3	−445.3 ±77.2	−599.5 ±103.7	−221.9 ±101.7	**−20.8** ±37.3	−39.7 ±64.1	−508.8 ±109.9
HC	5k	−550.8 ±55.5	−164.1 ±132.4	−589.6 ±56.4	−492.3 ±63.5	−488.6 ±107.4	**−107.6** ±94.5	−127.5 ±75.4	−450.3 ±68.2
HC	10k	−564.2 ±44.3	−251.6 ±95.8	−556.2 ±59.9	−512.9 ±32.5	−533.0 ±76.0	**122.2** ±91.7	−259.1 ±140.2	−409.8 ±97.8
HC	100k	−149.6 ±190.1	−11.9 ±78.1	**3516.7** ±1458.1	414.5 ±476.6	−303.3 ±77.3	−32.0 ±47.8	−80.5 ±208.4	−226.4 ±111.1
Hopper	1k	256.5 ±155.9	156.0 ±196.3	361.7 ±146.1	260.9 ±176.2	**602.0** ±107.5	26.9 ±30.1	148.7 ±154.8	26.6 ±22.6
Hopper	5k	319.0 ±201.7	519.7 ±365.5	347.0 ±179.1	899.0 ±379.5	**1045.3** ±117.7	173.3 ±233.3	383.3 ±214.8	516.1 ±459.3
Hopper	10k	1415.5 ±341.1	1094.9 ±210.9	1130.2 ±263.4	**2145.4** ±298.0	950.3 ±203.8	57.6 ±50.7	569.3 ±236.9	153.0 ±291.3
Hopper	100k	**3894.2** ±140.5	2753.2 ±451.6	3888.8 ±152.4	3571.5 ±742.5	3441.5 ±498.9	108.6 ±89.1	316.7 ±219.5	758.7 ±895.7
Walker2d	1k	62.3 ±54.4	217.7 ±70.5	56.3 ±33.3	99.2 ±80.3	**291.8** ±73.5	271.6 ±78.3	65.4 ±52.9	78.6 ±106.8
Walker2d	5k	27.0 ±14.1	**592.3** ±97.3	28.1 ±12.2	94.3 ±67.7	300.3 ±65.0	296.0 ±26.9	44.5 ±18.1	295.4 ±165.4
Walker2d	10k	89.0 ±51.5	**973.9** ±232.1	78.5 ±42.8	375.0 ±312.9	137.8 ±35.3	328.2 ±174.4	34.7 ±36.5	558.6 ±399.0
Walker2d	100k	**6849.3** ±136.3	3358.6 ±847.9	6824.0 ±141.4	6436.9 ±552.6	5248.7 ±582.6	294.8 ±242.2	41.0 ±73.8	719.6 ±471.2

Table 19: Mixed-data returns at one million updates. Each entry gives the ten-seed mean above its sample SD; row maxima are bold. HC denotes HalfCheetah.

Env.	Budget	BC	CQL	IQL	TD3+BC	ReBRAC	MCQ	EDAC	Trust-Mix
HC	1k	5.6 ±11.3	2422.9 ±704.7	48.1 ±39.2	593.4 ±1020.4	−0.1 ±13.0	2199.8 ±374.2	3055.8 ±583.0	**3411.3** ±217.5
HC	5k	6.5 ±27.4	2379.7 ±547.4	32.2 ±28.9	234.4 ±386.2	−12.8 ±26.2	1813.7 ±348.0	**3237.5** ±367.4	2419.0 ±1149.9
HC	10k	−4.8 ±5.4	1630.7 ±609.8	62.4 ±79.9	469.5 ±810.4	−7.5 ±20.7	1791.5 ±134.2	**3144.9** ±344.0	1495.3 ±1382.0
HC	100k	1.0 ±15.4	2475.3 ±622.3	160.1 ±53.0	2577.2 ±670.0	55.7 ±39.5	2036.8 ±337.1	**2970.8** ±134.6	1531.5 ±265.9
Hopper	1k	136.8 ±58.7	442.7 ±176.5	301.5 ±104.1	244.4 ±31.0	**450.2** ±379.1	428.0 ±285.5	225.2 ±62.6	415.1 ±476.3
Hopper	5k	109.8 ±47.2	481.4 ±424.3	373.3 ±203.1	713.1 ±380.0	163.4 ±47.3	**726.7** ±653.0	362.6 ±222.9	518.5 ±228.0
Hopper	10k	140.6 ±51.7	331.9 ±233.9	308.2 ±106.9	759.5 ±380.4	135.5 ±63.2	**791.7** ±503.8	360.1 ±186.1	640.7 ±377.8
Hopper	100k	118.9 ±49.6	762.0 ±406.8	257.3 ±73.1	**801.9** ±295.4	222.4 ±166.9	647.8 ±528.6	564.1 ±467.1	564.3 ±353.8
Walker2d	1k	60.0 ±11.0	128.8 ±108.1	**257.6** ±9.1	169.8 ±80.0	7.9 ±3.4	156.3 ±46.1	252.8 ±308.5	137.6 ±53.4
Walker2d	5k	58.1 ±20.1	100.8 ±112.1	275.2 ±22.6	249.9 ±94.7	51.8 ±70.6	245.5 ±84.4	223.0 ±203.0	**372.0** ±338.1
Walker2d	10k	58.2 ±28.1	236.4 ±243.5	279.7 ±16.6	141.3 ±116.4	25.5 ±30.1	191.6 ±61.5	63.7 ±91.5	**373.3** ±349.3
Walker2d	100k	54.3 ±19.1	255.0 ±272.5	**584.4** ±353.5	192.7 ±96.6	137.8 ±394.7	139.7 ±51.1	235.8 ±244.7	346.9 ±363.6

Table 20 summarizes comparisons of mixed-data cell means: Trust-Mix is higher in 61/84 comparisons (72.6%), leads 3/12 settings, and ranks in the top two in 5/12. These are descriptive counts, not seed-wise win probabilities or significance tests.

Table 20: Settings in which Trust-Mix exceeds the baseline's mixed-data mean, out of twelve.

Baseline	BC	CQL	IQL	TD3+BC	ReBRAC	MCQ	EDAC
Trust-Mix higher	12/12	8/12	10/12	7/12	11/12	5/12	8/12

C.5 CQL+DW-AW ADAPTATION

Implementation. The comparator is an independently written PyTorch adaptation of CQL+DW-AW (Hong et al., 2023b), using the authors' repository (`https://github.com/Improbable-AI/dw-offline-rl`) and JaxCQL (`https://github.com/young-geng/JaxCQL`) as references. The experiments use this PyTorch adaptation rather than the authors' JAX implementation. Its configuration is given in Table 21.

Advantage-weighted sampling. Trajectories/fragments are reconstructed from termination, timeout, available episode IDs, and next-observation continuity. The final fragment is closed; continuity tolerances are 10^{-5} (absolute and relative), and mean segment length below 2 is rejected. Trajectory returns are min–max normalized within the constructed input. An ordinary-least-squares model with an intercept predicts these normalized returns from standardized initial observations (Hong et al., 2023a). If A_j^{traj} is the return minus this prediction, transition sampling uses

$$p_i \propto \exp\Big(A_{j(i)}^{\mathrm{traj}}/0.1\Big),$$

where $j(i)$ contains transition i. This sampling distribution remains fixed throughout training. Standardization is confined to the regression covariates, not actor/critic inputs.

State and state–action weighting. The log-ratio networks $\phi(s)$ and $\psi(s,a)$ give

$$v(s) = \exp\{\phi(s)\}, \qquad w(s,a) = \exp\{\phi(s) + \psi(s,a)\}.$$

Detached state ratios weight actor and entropy losses; detached state–action ratios weight Bellman and conservative critic losses. Each type is separately normalized to batch mean one in the RL losses. The density objective combines weighted return, KL, and flow terms. Return/KL terms use sum-one batch weights; flow uses raw exponential ratios and the next-state marginal $\exp\{\phi(s')\}$, discount one, and no terminal mask. The Bellman target instead uses discount 0.99 and true-terminal masking. Density networks receive one update per RL iteration, without actor/critic gradients through the weights.

Backbone and comparison scope. Conservative candidates come from uniform actions, current-state policy actions, and next-state policy actions, with every candidate evaluated by Q at the current state. The adapter uses LeCun-style hidden initialization, small output layers, and a learned actor log-standard-deviation multiplier/offset. Table 22 summarizes differences from the paper and public launcher; the state/state–action weighting distinction follows the original method.

The fragment reconstruction, ratio normalization, and AW preprocessing used by the adapter are defined above. Compared with our standard CQL implementation, the adapter uses two rather than three 256-unit hidden layers, actor/entropy rates 3×10^{-4} rather than 10^{-4}, and CQL coefficient 10 rather than 5. Its difference from standard CQL therefore does not isolate the effect of adding DW-AW.

Results and configuration selection. The 240-run study includes the initial thirty mixed diagnostic runs (HalfCheetah-1k, Hopper-10k, Walker2d-100k; ten seeds each). Their respective means were -401.9, 15.3, and 78.3. The expanded study retained the same numerical settings despite these weak returns, without per-cell retuning or counting inherited runs as additional seeds.

Table 21: CQL+DW-AW PyTorch-adapter configuration.

Setting	Recorded value
Actor, twin critics, and density-weight hidden layers	256,256 (ReLU)
Batch size / RL updates	256 / 1,000,000
Actor / critic / entropy learning rates	3×10^{-4} each
Density learning rate / updates per RL step	10^{-4} / 1
Bellman discount / target averaging	0.99 / 0.005
CQL coefficient / temperature	10 / 1
Actions per proposal distribution	10
CQL importance correction / Lagrange variant	On / off
Entropy backup / BC warmup	Off / 0 updates
Entropy target	$-\dim(\mathcal{A})$
AW temperature / baseline	0.1 / initial-state OLS
Density KL / flow coefficients	0.2 / 0.1
Actor/critic observation normalization	None
CQL and density-loss rewards	Raw stored rewards
AW trajectory-return preprocessing	Dataset min–max
Recorded-action clipping threshold	0.999
Gradient clipping / mixed precision	None / off
Evaluation interval / episodes	50,000 / 20
Evaluation reset seeds	100000–100019

Table 22: Original CQL+DW-AW and the present adapter. Published settings follow Hong et al. (2023b); launcher settings refer to `experiments/d4rl/run_cql.py`, not every configuration used in the original paper.

Aspect	Original paper / public launcher	Present experiment
Implementation	JaxCQL-based JAX implementation; authors' public repository.	Independent PyTorch implementation described in this appendix.
Datasets	D4RL-v2 locomotion mixtures, with standard, diverse and small-data settings.	Minari-derived MuJoCo-v5 buffers; 1M random transitions plus 1k/5k/10k/100k expert transitions.
Weight use	State weights in actor/entropy losses; state–action weights in critic losses.	Same role distinction; detached RL weights normalized separately to mean one per sampled batch.
Density settings	Two 256-unit ReLU layers; Adam 10^{-4}; batch 256; one density step per RL step; selected CQL $(\lambda_K, \lambda_F) = (0.2, 0.1)$.	These settings retained. Explicit sum-one normalization in return/KL terms; raw ratios in flow.
AW temperature	Paper: 0.1 for standard-sized CQL mixtures and 5.0 for its small datasets. Public launcher includes `AW-0.2`.	Fixed 0.1 for every input/budget, including expert-only; no small-dataset temperature switch.
Reward handling	Public launcher selects `reward_norm=max-min`.	Raw rewards in CQL and density losses; only trajectory returns used for AW are min–max normalized before initial-state OLS.
Evaluation	Paper: three learner seeds; 20 episodes every 1,000 updates; last ten rounds averaged; normalized cross-dataset IQM.	Ten learner seeds; 20 deterministic episodes every 50k; raw return of the exact-1M policy; reset seeds 100000–100019, matching Trust-Mix.

Table 23 reports the complete exact-1M endpoints and ten-seed paired composition effects. HalfCheetah-5k has a positive interval and Walker2d-1k a negative interval; the other ten include zero. These results characterize the adapter, not the performance of the authors' official implementation.

Table 23: CQL+DW-AW PyTorch-adapter returns at one million updates: mean ± sample SD and mixed-minus-expert effects with 95% paired intervals. Each row uses ten matched pairs.

Environment	Budget	Expert only	Expert+random	Paired effect 95% interval
HalfCheetah	1k	-348.8 ± 155.0	-401.9 ± 187.5	-53.2 $[-213.7, +105.7]$
HalfCheetah	5k	-512.4 ± 130.0	-311.8 ± 172.8	$+200.6$ $[+85.1, +321.8]$
HalfCheetah	10k	-162.2 ± 319.4	-259.0 ± 184.7	-96.9 $[-299.2, +93.5]$
HalfCheetah	100k	-340.3 ± 109.3	-379.5 ± 189.4	-39.3 $[-172.8, +80.9]$
Hopper	1k	3.2 ± 2.2	2.4 ± 2.2	-0.8 $[-3.0, +1.6]$
Hopper	5k	1.4 ± 3.7	2.0 ± 3.5	$+0.6$ $[-3.0, +4.3]$
Hopper	10k	14.6 ± 43.4	15.3 ± 18.9	$+0.6$ $[-29.4, +23.1]$
Hopper	100k	1.0 ± 0.7	5.2 ± 13.4	$+4.2$ $[-0.7, +12.9]$
Walker2d	1k	175.1 ± 74.1	97.4 ± 12.1	-77.7 $[-119.5, -37.8]$
Walker2d	5k	114.8 ± 24.5	110.7 ± 79.2	-4.1 $[-49.8, +55.0]$
Walker2d	10k	73.9 ± 70.2	67.3 ± 43.7	-6.6 $[-44.9, +31.4]$
Walker2d	100k	96.2 ± 113.6	78.3 ± 46.5	-18.0 $[-105.2, +58.8]$

C.6 Learning histories and HalfCheetah path audit

The main study contains 4,800 recorded evaluation means: 240 histories with twenty scheduled checkpoints each, from 50k through 1M. Every history has the complete grid and agrees with the endpoint summary. Table 24 reports all input conditions.

For learner seed k, define the descriptive peak-minus-final loss

$$L_k = \max_{t \in \mathcal{T}} R_k(t) - R_k(10^6), \qquad \mathcal{T} = \{50{,}000, 100{,}000, \ldots, 10^6\}.$$

The table reports $\bar{L}_{\text{peak}} = 10^{-1} \sum_k L_k$. This generally differs from the peak of the mean curve minus its endpoint because seeds peak at different iterations. Finite-episode noise affects the observed peaks, which are never used for policy selection.

HalfCheetah. All 800 expert-only checkpoint evaluation means are negative (4 budgets $\times 10$ seeds $\times 20$ checkpoints). All forty first evaluations at 50k are negative, and every history remains negative at each saved checkpoint through 1M. At the 100k expert budget, the mean improves from -607.8 to -226.4 but remains negative. The weakness is therefore not confined to a single final checkpoint.

Walker2d. At the expert-only 100k budget, the mean rises from 22.2 at 50k to 719.6 at 1M. The mean curve peaks at 898.5 at 850k, while the mean per-seed peak-minus-final loss is 410.9. Even the largest recorded mean is below the separate expert-only BC endpoint mean of 6849.3.

Table 24: Saved learning histories, with ten seeds and twenty checkpoints per row. $\bar{R}_{50k}$ and $\bar{R}_{1M}$ are across-seed means; $\bar{L}_{\text{peak}}$ averages each seed's peak-minus-final loss. $N_{<0}$ counts negative checkpoint evaluation means out of 200, not individual episodes. E: expert only; M: expert+random.

Environment	Budget	Input	$\bar{R}_{50k}$	$\bar{R}_{1M}$	$\bar{L}_{\text{peak}}$	$N_{<0}$
HalfCheetah	1k	E	-329.8	-508.8	292.7	200/200
HalfCheetah	1k	M	1418.4	3411.3	61.1	0/200
HalfCheetah	5k	E	-437.7	-450.3	217.1	200/200
HalfCheetah	5k	M	102.4	2419.0	558.7	3/200
HalfCheetah	10k	E	-488.8	-409.8	87.1	200/200
HalfCheetah	10k	M	190.2	1495.3	668.8	19/200
HalfCheetah	100k	E	-607.8	-226.4	52.1	200/200
HalfCheetah	100k	M	27.1	1531.5	204.9	7/200
Hopper	1k	E	12.7	26.6	75.1	14/200
Hopper	1k	M	435.4	415.1	481.6	0/200
Hopper	5k	E	82.1	516.1	199.1	0/200
Hopper	5k	M	374.9	518.5	594.7	0/200
Hopper	10k	E	39.9	153.0	159.2	0/200
Hopper	10k	M	385.2	640.7	550.0	0/200
Hopper	100k	E	18.5	758.7	214.0	0/200
Hopper	100k	M	280.3	564.3	280.2	0/200
Walker2d	1k	E	83.2	78.6	123.4	2/200
Walker2d	1k	M	247.8	137.6	448.8	0/200
Walker2d	5k	E	20.6	295.4	256.2	10/200
Walker2d	5k	M	236.8	372.0	322.7	0/200
Walker2d	10k	E	41.2	558.6	194.5	9/200
Walker2d	10k	M	220.4	373.3	340.0	0/200
Walker2d	100k	E	22.2	719.6	410.9	0/200
Walker2d	100k	M	124.9	346.9	160.2	0/200

Interpretation. This post-hoc analysis establishes when weak returns are visible, not their cause. With no evaluations before 50k, it cannot distinguish failure during the 10k initialization from deterioration before the first recorded evaluation. Return histories alone also do not diagnose behavioral prediction errors, data transformations, or the effects of gradient capping and weight concentration.

C.6.1 Complete Trust-Mix learning curves

Figures 5–7 show all four budgets for each environment. Lines average the same ten seeds; bands are pointwise 95% intervals from resampling whole seed histories, not simultaneous confidence bands.

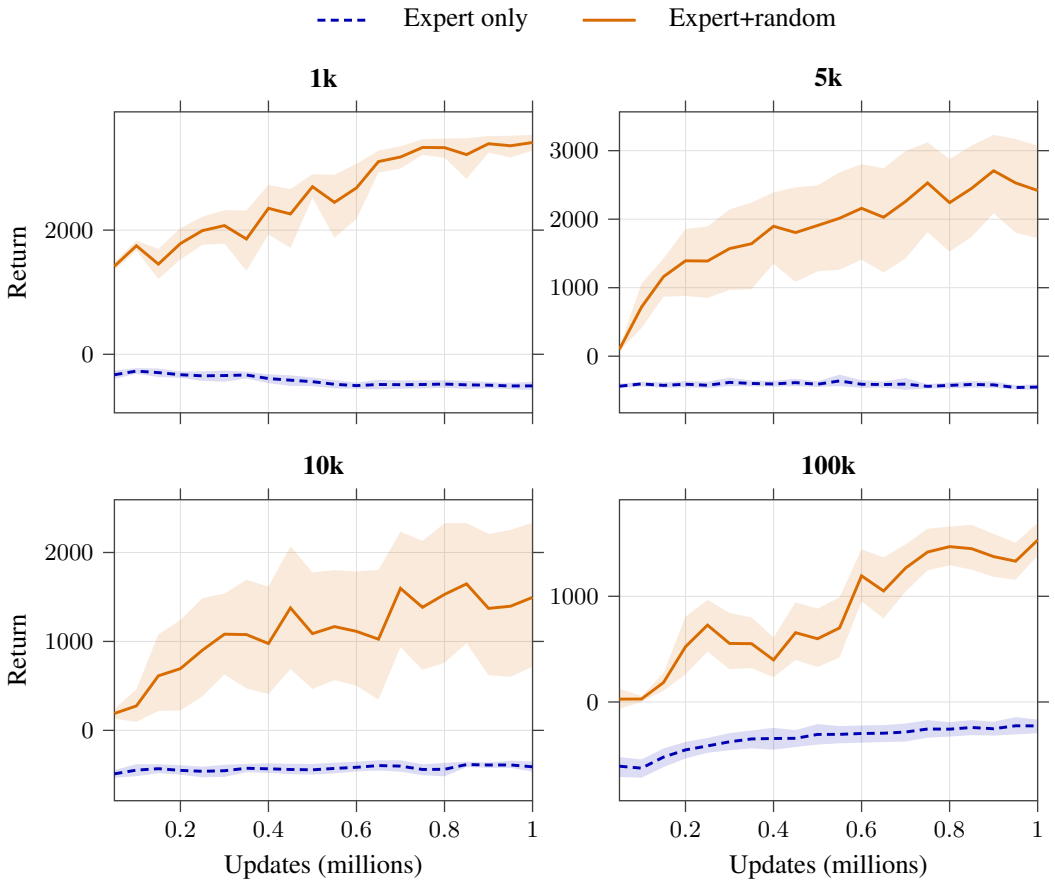

Figure 5: HalfCheetah learning curves: mean and pointwise 95% seed-bootstrap intervals, ten seeds per condition.

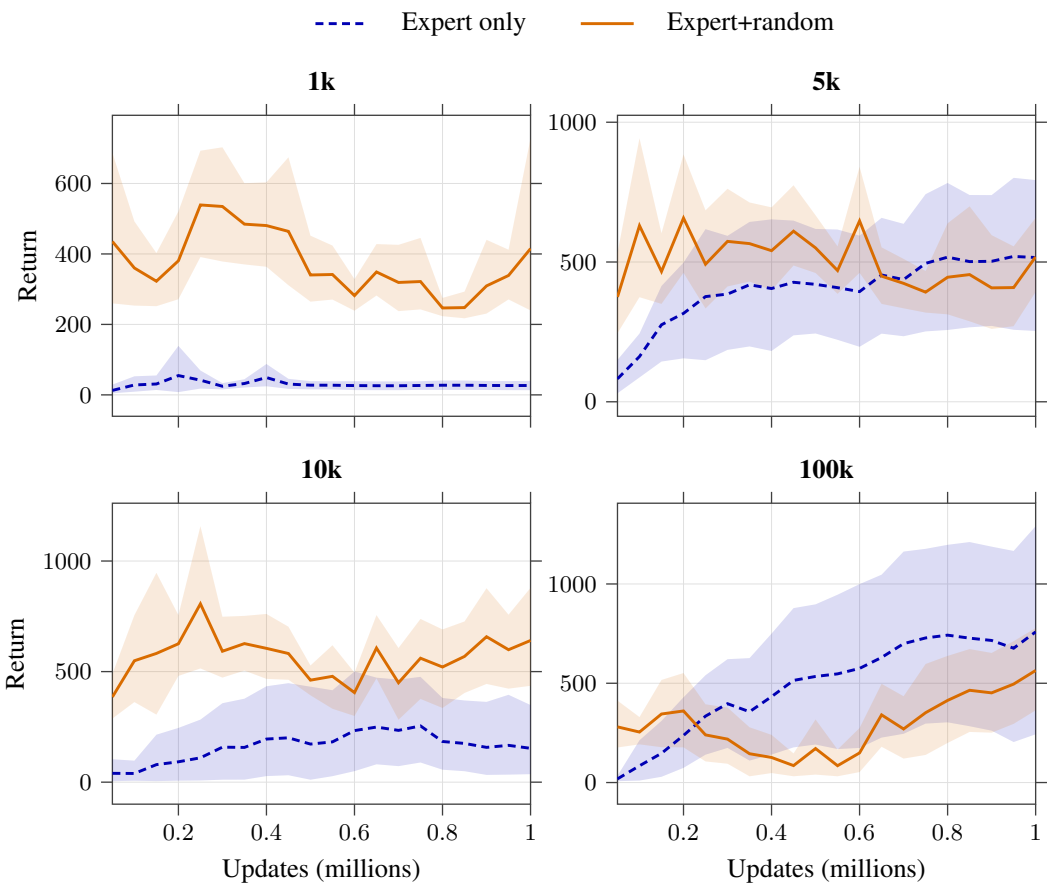

Figure 6: Hopper learning curves: mean and pointwise 95% seed-bootstrap intervals, ten seeds per condition.

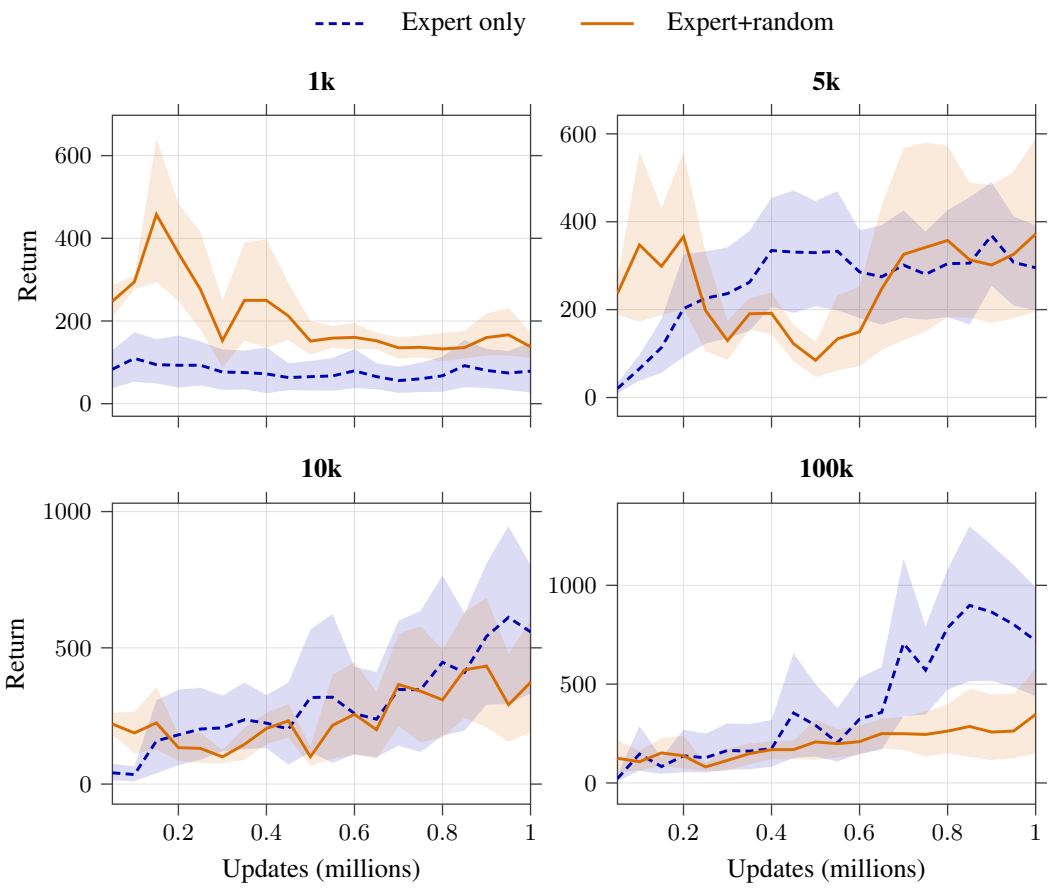

Figure 7: Walker2d learning curves: mean and pointwise 95% seed-bootstrap intervals, ten seeds per condition.