Split-Half Critic Updates Improve Short-Horizon SAC AUC on Three MuJoCo Tasks

Rongxuan Deng
fdeng@andrew.cmu.edu

Carnegie Mellon University

Abstract

Soft actor-critic (SAC) implementations commonly update both critics on the same replay minibatch. We study a minimal split-half update that samples one batch of size 512, trains the two critics on disjoint halves, and leaves the actor, clipped target, network size, replay buffer, and environment interaction budget unchanged. Although evaluated on reward-based MuJoCo tasks, the question is a lower-level data-reuse issue that also appears in beyond-reward pipelines once preferences, demonstrations, language objectives, or unsupervised objectives produce replay data for an off-policy learner. In a fixed reported evaluation grid on HalfCheetah-v5, Hopper-v5, and Walker2d-v5 with held-out seeds 6–15, 150k training steps, and deterministic evaluation every 10k steps, split-half critic updates improve normalized train-AUC IQM to 1.2201 from 1.0023/1.0009 and evaluation-AUC IQM to 1.3041 from 0.9853/0.9748 against batch-size-512 SAC and a delayed-actor-update control. Paired evaluation-AUC wins are 22/30 and 25/30. The evidence supports a scoped short-horizon implementation result; it does not establish broad MuJoCo/DeepMind Control generality, directly measure overestimation-bias reduction, or isolate replay-overlap reduction from per-critic batch-size reduction.

1 Introduction

Small implementation choices can dominate empirical conclusions in online deep reinforcement learning. This is especially relevant for continuous-control SAC, where the method is widely used because it is simple, strong, and sample efficient, but where many proposed stabilizers are expensive to validate fairly.

Although our experiments use standard reward-based MuJoCo tasks, this is not a reward-free task-specification paper. The connection to this workshop is the lower-level data-reuse problem that many beyond-reward agents inherit: after reward-free exploration, preferences, demonstrations, language-conditioned objectives, or unsupervised objectives produce trajectories, labels, or surrogate rewards, an off-policy learner still has to reuse that data through value-function updates. We therefore study split-half critic updates as a training-reliability ingredient for such pipelines, not as a new way to specify tasks or rewards.

SAC uses two critics and a clipped double-Q target (Haarnoja et al., 2018; Fujimoto et al., 2018). In common single-file implementations, both critics are trained on the same replay minibatch at each update. This is a reasonable default, but it also makes the two critic losses see identical sampled transitions. The question we study is deliberately small: can split-half critic updates improve short-horizon learning AUC without changing the actor, target construction, network size, or number of environment steps?

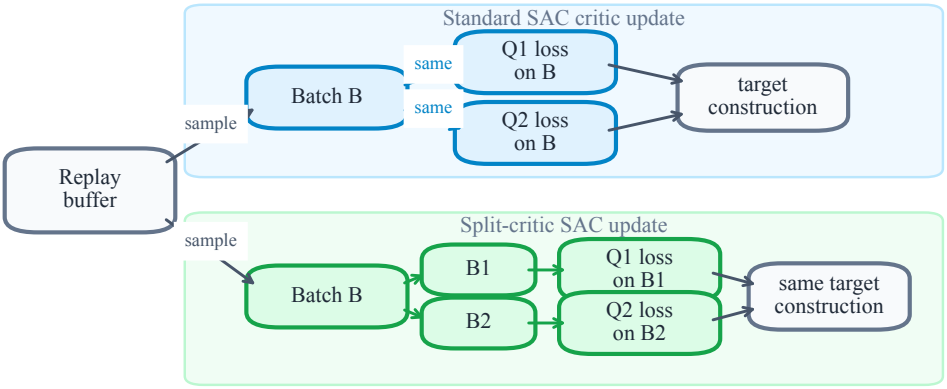

Figure 1: Split-critic SAC changes critic data exposure while preserving the standard SAC actor, replay buffer, target construction, and environment interaction budget. Standard SAC trains both critics on the same replay minibatch; split-critic SAC partitions the same sampled batch into disjoint critic-specific halves.

We introduce *split-critic SAC*, a minimal modification to the critic update. Given a replay minibatch of size 512, we partition it into two disjoint halves, update the first critic on one half and the second critic on the other, and keep the usual clipped target and actor update unchanged. The method does not add critics, use larger networks, or collect extra data. It does, however, change the per-critic critic-loss batch size from 512 to 256; we treat this as part of the scoped intervention rather than a separately isolated mechanism.

Figure 1 shows the intervention. The important property is that the sampled replay budget remains fixed: only critic data exposure changes.

Our contribution is empirical and scoped:

1. We implement a batch-disjoint twin-critic update in CleanRL SAC (Huang et al., 2022) while preserving the standard SAC interaction budget, target construction, actor update, and model size.
2. We run a 120-run reported evaluation grid on HalfCheetah-v5, Hopper-v5, and Walker2d-v5 using held-out seeds 6–15, 150k training steps, deterministic evaluation every 10k steps, and 10 evaluation episodes per checkpoint.
3. We find that split-critic SAC improves normalized train and evaluation AUC over both batch-size-512 vanilla SAC and a fixed-delay SAC control under this protocol. The evaluation-AUC paired wins are 22/30 and 25/30, respectively.
4. We disclose secondary low-return collapse and throughput measurements, while explicitly not claiming all-MuJoCo, DeepMind Control, longer-horizon, or state-of-the-art generality.

The paper is organized as follows. Section 2 positions the result relative to SAC, clipped double-Q learning, and critic-diversity methods. Section 3 describes the split-critic update. Section 4 gives the reported evaluation grid and main results. Section 5 states the boundaries of the evidence.

2 Related Work

Soft actor-critic and clipped double-Q learning. SAC is an off-policy actor-critic method that combines entropy-regularized policy optimization with learned Q-functions (Haarnoja et al., 2018). Its twin-critic target follows the clipped double-Q idea used in TD3 (Fujimoto et al., 2018), which in turn is motivated by overestimation concerns related to double Q-learning (van Hasselt, 2010). Our

method does not change the clipped target or the actor objective. It instead changes which replay samples each of the two critics receives during the critic-loss update.

Critic diversity and value-estimate reliability. The motivation for using two critics is not merely architectural redundancy: clipped double-Q methods are meant to make value estimates less vulnerable to overoptimistic errors. Many later approaches pursue the same reliability goal by increasing critic diversity, adding critic ensembles, or changing the way target values are formed. REDQ combines a high update-to-data ratio with a Q-function ensemble and random subset target minimization (Chen et al., 2021); DroQ studies dropout Q-functions as a cheaper ensemble-style alternative (Hiraoka et al., 2022). Those directions can improve sample efficiency, but they change the compute and implementation surface of the SAC baseline. The present paper asks a narrower question: whether the two existing SAC critics benefit from disjoint replay exposure when the total sampled batch, environment steps, and network count are held fixed.

Beyond-reward agents still need stable off-policy reuse. Reward-free exploration, imitation, preference learning, and language-conditioned control differ in how they obtain task signal, but many practical pipelines still train an off-policy actor-critic once data and scalar or surrogate training targets are available. In that setting, critic reliability can remain an important concern even when the original task signal is not a hand-written reward. Our experiments use MuJoCo rewards because they provide a controlled benchmark for the low-level update rule; the intended relevance is to the SAC-style learner inside broader beyond-reward systems, not to the task-specification layer itself.

Evaluation practice. Empirical RL results are sensitive to seeds, hyperparameters, codebase details, and evaluation choices (Henderson et al., 2018; Engstrom et al., 2020). Agarwal et al. (2021) argue for robust aggregate statistics such as IQM and probability of improvement instead of relying only on mean scores over a small number of seeds. We follow this practice with held-out seeds, paired comparisons over matched environment/seed tuples, normalized AUC metrics, and bootstrap confidence intervals. We use CleanRL (Huang et al., 2022) and Gymnasium MuJoCo tasks backed by MuJoCo physics (Todorov et al., 2012) to keep the implementation change inspectable.

3 Split-Critic SAC

We start from standard SAC with policy $\pi_\theta(a \mid s)$, two critics Q_{ϕ_1} and Q_{ϕ_2}, target critics $Q_{\bar{\phi}_1}$ and $Q_{\bar{\phi}_2}$, and replay buffer $\mathcal{D}$. For a transition (s, a, r, s', d), the usual clipped target is

$$y = r + \gamma(1-d)\left[\min_{i\in\{1,2\}} Q_{\bar{\phi}_i}(s', a') - \alpha \log \pi_\theta(a' \mid s')\right], \quad a' \sim \pi_\theta(\cdot \mid s').$$

Standard SAC then minimizes both critic losses over the same sampled minibatch $\mathcal{B}$:

$$\mathcal{L}_i^{\text{std}}(\phi_i) = \frac{1}{|\mathcal{B}|} \sum_{(s,a,r,s',d)\in\mathcal{B}} \left(Q_{\phi_i}(s,a) - y\right)^2, \qquad i \in \{1,2\}.$$

Split-critic SAC samples the same total batch size but partitions the batch into two disjoint subsets, $\mathcal{B}_1$ and $\mathcal{B}_2$, with $\mathcal{B}_1 \cap \mathcal{B}_2 = \emptyset$ and $|\mathcal{B}_1| + |\mathcal{B}_2| = |\mathcal{B}|$. The critic losses are

$$\mathcal{L}_1^{\text{split}}(\phi_1) = \frac{1}{|\mathcal{B}_1|} \sum_{(s,a,r,s',d)\in\mathcal{B}_1} \left(Q_{\phi_1}(s,a) - y\right)^2,$$

$$\mathcal{L}_2^{\text{split}}(\phi_2) = \frac{1}{|\mathcal{B}_2|} \sum_{(s,a,r,s',d)\in\mathcal{B}_2} \left(Q_{\phi_2}(s,a) - y\right)^2.$$

The split losses use the same clipped SAC target defined above. Actor updates, entropy-temperature updates, replay sampling frequency, target-network updates, and environment interaction are unchanged.

What is controlled. All compared variants sample the same total replay batch size, use the same actor and critic architectures, collect the same number of environment steps, and compute the same clipped SAC target. The split variants change two inseparable properties of the critic loss: the two critics no longer train on identical transitions, and each critic receives 256 rather than 512 transitions per critic update. The evidence therefore supports split-half critic updates as a scoped implementation change; it does not isolate replay-overlap reduction from per-critic batch-size reduction.

Algorithm 1 One split-critic SAC update

1: Sample replay minibatch $\mathcal{B}$ of size 512 from $\mathcal{D}$.
2: Randomly partition $\mathcal{B}$ into disjoint halves $\mathcal{B}_1$ and $\mathcal{B}_2$.
3: Compute clipped SAC targets y using both target critics.
4: Update Q_{ϕ_1} using transitions in $\mathcal{B}_1$.
5: Update Q_{ϕ_2} using transitions in $\mathcal{B}_2$.
6: Update the actor and entropy temperature exactly as in SAC.

Variants. The main method, Split, uses batch size 512 and the standard actor-update period. The primary baseline, Vanilla, uses the same batch size with shared critic batches. The schedule control, Fixed delay, keeps shared critic batches but updates the actor every fourth critic update. The secondary ablation, Split+delay4, combines split critic batches with that same delayed actor schedule.

Diagnostics. We log two mask diagnostics to verify which transitions each critic uses. The Q1 replay fraction is the fraction of the sampled minibatch included in the Q1 loss, and the Q2 replay fraction is defined analogously. Mask overlap is the fraction of sampled transitions used by both critic losses in the same update. Thus Vanilla and Fixed delay have Q1/Q2 fractions 1.0/1.0 and overlap 1.0, while Split and Split+delay4 have fractions 0.5/0.5 and overlap 0.0. These diagnostics confirm that the intended data partition is active. We also log the mean absolute disagreement between the two critics on sampled replay actions, but use it only as a descriptive value-disagreement diagnostic. The reported evidence does not establish that disagreement magnitude, or overestimation-bias reduction, is the mechanism behind the gains.

4 Experiments

Protocol. We used development runs on seeds 1–5 to choose a candidate split configuration and the controls to report. Before inspecting the held-out seeds, we fixed the final evaluation grid: environments, four variants, 150k horizon, evaluation schedule, normalization, and primary AUC/IQM metrics. No reported environment or variant was added or dropped after inspecting held-out seed outcomes. This held-out-seed protocol reduces seed-level selection leakage, but it does not remove all residual selection risk across candidate variants or the three-task benchmark choice; we therefore keep the claim scoped to this evaluation grid. The reported grid contains all fixed variants: three Gymnasium MuJoCo environments (HalfCheetah-v5, Hopper-v5, Walker2d-v5) $\times$ ten held-out seeds (6–15) $\times$ four variants, for 120 completed runs. Deterministic evaluation runs every 10k steps, with 10 evaluation episodes per checkpoint. All compared variants use CleanRL SAC, batch size 512, and the same environment-step budget.

Why these four variants. The four reported variants isolate the main alternatives a reader might attribute the effect to:

Variant	Role in the comparison
Vanilla	Batch-size-512 SAC with shared critic batches; the direct baseline.
Fixed delay	Shared critic batches with actor period 4; controls for delayed actor updates.
Split	Main method: disjoint half-minibatches with the standard actor period.
Split+delay4	Secondary ablation testing whether splitting and delayed actor updates combine.

The appendix control table gives the corresponding per-critic batch sizes and actor-update periods.

Metrics. The primary metrics are normalized train AUC IQM and normalized evaluation AUC IQM. AUC is computed from the training-return and periodic-evaluation curves over the 150k-step horizon, so it measures short-horizon learning progress rather than only the last checkpoint. Normalization uses the environment-specific vanilla batch-size-512 scale before aggregation. Bootstrap intervals are computed over run-level records after normalization. Paired effects compare matched environment/seed tuples, so each paired comparison has $n = 30$. Because the three environments are a fixed benchmark suite rather than a random sample of tasks, these intervals quantify variability within this protocol and should not be interpreted as uncertainty over broader MuJoCo or DeepMind Control task distributions. AUC and paired wins are performance metrics; they do not directly measure overestimation bias or value calibration. Section B gives the exact formulas and bootstrap conventions.

Main AUC result. Figure 2 shows the main evidence. Split-critic SAC improves normalized train-AUC IQM from 1.0023 for vanilla and 1.0009 for fixed delay to 1.2201. Normalized evaluation-AUC IQM improves from 0.9853 and 0.9748 to 1.3041. Paired normalized evaluation-AUC differences are positive against both controls: +0.3668 versus vanilla, with 95% bootstrap CI [0.2134, 0.5335], and +0.3626 versus fixed delay, with CI [0.2187, 0.5350].

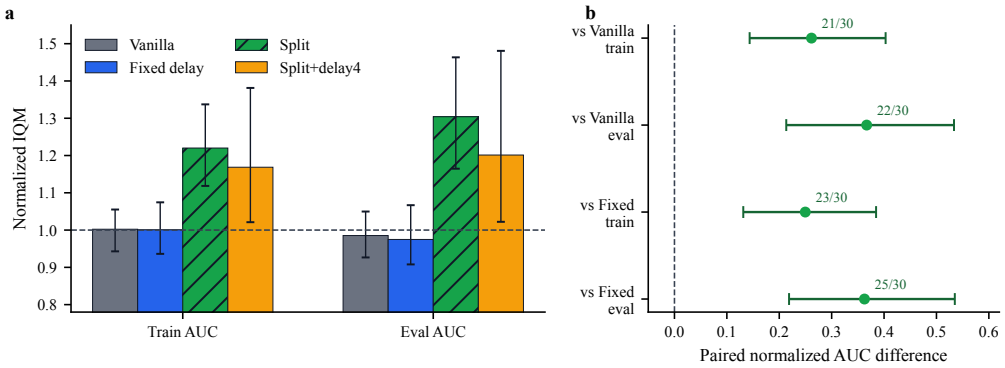

Figure 2: Main reported evaluation-grid AUC evidence on HalfCheetah-v5, Hopper-v5, and Walker2d-v5 with held-out seeds 6–15 and 150k training steps. Left: normalized train/evaluation AUC IQM with 95% bootstrap confidence intervals over 30 environment-seed runs per variant. Right: paired normalized AUC differences for split-critic SAC against vanilla SAC and the fixed-delay control, which keeps shared critic batches but delays actor updates to every fourth critic update. Labels show paired wins over 30 matched environment-seed comparisons.

Table 1: Aggregate reported evaluation-grid results on HalfCheetah-v5, Hopper-v5, and Walker2d-v5 with held-out seeds 6–15 and 150k training steps. AUC metrics are normalized IQM over 30 environment-seed runs per variant; uncertainty is shown in Figure 2. Fixed delay keeps shared critic batches but delays actor updates to every fourth critic update. Split+delay4 combines split critic batches with that same delayed actor schedule. Collapse is normalized final training return below 0.5 under the vanilla-baseline environment scale or raw final return below 0; SPS overhead is measured relative to vanilla. Collapse and overhead are secondary evidence.

Variant	Train AUC	Eval AUC	Wins vs van.	Wins vs delay	Collapse (%)	SPS overhead vs vanilla (%)
Vanilla	1.0023	0.9853	–	–	13.3	0.00
Fixed delay	1.0009	0.9748	15/30	–	6.7	0.00
Split	**1.2201**	**1.3041**	22/30	25/30	3.3	0.60
Split+delay4	1.1686	1.2013	20/30	22/30	6.7	0.39

Paired wins and secondary outcomes. Table 1 gives the compact aggregate table. Split-critic SAC wins 22/30 paired evaluation-AUC comparisons against vanilla and 25/30 against fixed delay. The same table reports secondary quantities. Under a low-return collapse criterion fixed before summarizing the reported grid, normalized final return below 0.5 under the vanilla-baseline environment scale or raw final return below 0, the observed collapse rate is 3.3% for split-critic SAC, compared with 13.3% for vanilla and 6.7% for fixed delay. Measured SPS overhead on the local implementation is 0.60%. These secondary measurements are not hardware-independent efficiency or broad robustness claims.

Final-return disclosure. The final-return table in Table 5 reports normalized final train and final evaluation returns for transparency. Split-critic SAC also has higher point-estimate final-return IQM under this protocol, but final returns are noisier than AUC and are not the headline supported claim. We use AUC as the primary metric because it reflects the full 150k-step learning trajectory.

Per-environment effects and ablation. Figure 3 breaks down normalized evaluation-AUC effects by environment, with paired per-environment intervals in Table 7. The main split-critic variant has positive mean effects against both controls on all three tested environments, but the HalfCheetah-v5 intervals include zero. The secondary Split+delay4 ablation is positive in aggregate, but its Hopper-v5 effect is negative relative to both controls. We therefore use Split as the main method and treat Split+delay4 only as a supporting ablation.

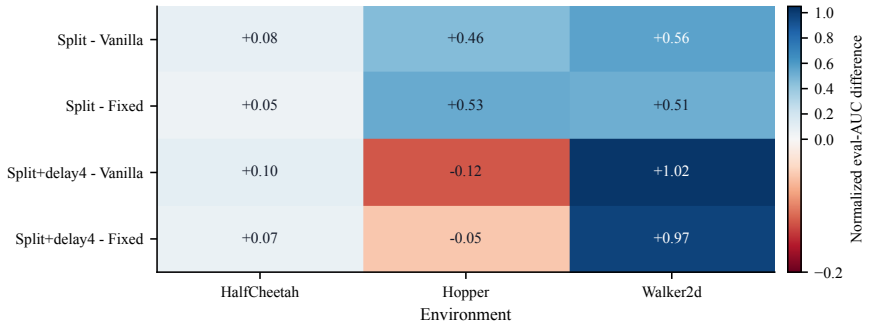

Figure 3: Per-environment normalized evaluation-AUC mean differences under the reported evaluation grid. The main split-critic variant has positive mean effects against both controls on all three tested environments; paired intervals in Table 7 show that HalfCheetah-v5 remains inconclusive. Split+delay4 denotes split critic batches with actor updates delayed to every fourth critic update; it is positive in aggregate but has less clean per-environment behavior and is treated as a secondary ablation.

Integrity checks. All 120 raw run records report 150k total timesteps, 16 evaluation checkpoints, and non-empty training returns. These records match the stated environment, seed, step-count, evaluation-frequency, and device fields. We therefore keep the claim boundary limited to this protocol and do not claim broader all-MuJoCo, DeepMind Control, longer-horizon, state-of-the-art, or mechanism-proof results.

5 Discussion and Limitations

The result suggests that split-half critic updates are a meaningful implementation axis for SAC under this protocol. The intervention is small enough that the supported interpretation is also narrow: under the reported evaluation grid, disjoint critic half-minibatches improve AUC over two batch-size-512 controls. This is different from claiming that critic diversity generally explains SAC stability. The mask diagnostics confirm that the split is active, but the critic value-disagreement diagnostic is not used as causal evidence.

The main limitation is scope. We evaluate only HalfCheetah-v5, Hopper-v5, and Walker2d-v5, only for 150k steps, and only in this CleanRL/Gymnasium implementation at this budget. The held-out seeds reduce seed-level overfitting but do not make the candidate selection independent of the development grid or establish generality over environments. The experiments do not cover Ant, Humanoid, DeepMind Control, longer horizons, or stronger ensemble-style off-policy baselines. They also do not isolate whether the improvement comes from lower replay overlap, the smaller per-critic critic-loss batch, or their combination. Final returns are disclosed in Table 5, but the strongest supported evidence remains train and evaluation AUC.

The experiments also do not directly measure the overestimation-bias mechanism that motivates many twin-critic methods. AUC can show that the learner improves faster under this protocol, and mask overlap can show that the critics receive disjoint data, but neither quantity proves that value estimates are less biased or better calibrated. A stronger mechanism study would compare critic estimates against rollout-based return proxies or other value-calibration diagnostics while controlling replay overlap and per-critic batch size independently.

The overhead measurement has the same scope. The 0.60% value is measured SPS overhead in this local code path and should not be read as a hardware-independent cost claim. Likewise, the observed final-training-return collapse-rate difference is tied to the fixed low-return criterion and the small number of observed events. We include these measurements because they are useful for interpreting the AUC result, but they are secondary evidence.

Future work should test the same update on broader MuJoCo and DeepMind Control suites, including higher-dimensional tasks such as Humanoid, longer budgets, and stronger off-policy baselines. It should also isolate mechanism more directly, for example by controlling critic correlation, target disagreement, value-estimate bias, and replay overlap independently.

6 Conclusion

We studied a minimal SAC modification: train the two critics on disjoint halves of each replay minibatch while preserving the standard actor, clipped target, batch size, and environment interaction budget. In a scoped 120-run reported evaluation grid on three Gymnasium MuJoCo tasks, this split-critic update improved normalized train and evaluation AUC over vanilla SAC and a fixed-delay SAC control. The evidence supports a narrow but concrete claim about this protocol and suggests a simple data-reuse knob for off-policy learners used inside broader beyond-reward systems. Broader environment coverage, longer training horizons, stronger baselines, and direct value-bias diagnostics remain necessary before making general claims about SAC stability.

A Protocol Details

Table 2: Reported evaluation grid on HalfCheetah-v5, Hopper-v5, and Walker2d-v5 with held-out seeds 6–15. All variants use CleanRL SAC, batch size 512, 150k training steps, and deterministic evaluation every 10k steps with 10 episodes.

Variant	Critic data	Actor update period	Role
Vanilla	Shared batch	2	baseline
Fixed delay	Shared batch	4	schedule control
Split	Disjoint half-batches	2	main method
Split+delay4	Disjoint half-batches	4	secondary ablation

Table 3: Controlled and changed quantities in the reported evaluation-grid variants. All rows use the same clipped target construction, replay buffer settings, network count, and environment-step budget. Split rows change the per-critic effective critic-loss batch size as part of the combined intervention.

Variant	Sampled batch	Per-critic batch	Critic loss terms	Actor period	Other controls
Vanilla	512	512 / 512	1024	2	same
Fixed delay	512	512 / 512	1024	4	same
Split	512	256 / 256	512	2	same
Split+delay4	512	256 / 256	512	4	same

Table 4: Implementation and hyperparameter details for the reported evaluation grid. Values are read from the raw JSON run arguments unless otherwise noted.

Item	Value
Code base	CleanRL SAC implementation
Implementation entry point	Continuous-action SAC training script
Environments	HalfCheetah-v5, Hopper-v5, Walker2d-v5
Seeds	6–15
Total timesteps	150,000
Replay buffer	1,000,000
Learning starts	5,000
Batch size sampled from replay	512
Discount / target smoothing	$\gamma = 0.99$, $\tau = 0.005$
Actor / critic learning rates	3×10^{-4} / 1×10^{-3}
Target network frequency	1
Entropy coefficient	autotuned from initial $\alpha = 0.2$
Evaluation	deterministic, every 10,000 steps, 10 episodes
Software	Torch 2.12.0; NumPy 2.4.6; Gymnasium 1.2.3

All reported evaluation-grid runs use the same environment-step budget and the same held-out seed list, 6–15. The compared environments are HalfCheetah-v5, Hopper-v5, and Walker2d-v5. Each run evaluates deterministically every 10k steps with 10 evaluation episodes.

The run configuration uses total timesteps 150k, evaluation frequency 10k, and 10 evaluation episodes. Each raw JSON record stores the resolved arguments, train returns, evaluation returns, and timing fields used for the tables in this paper. The main text is standalone: appendix formulas and tables provide audit detail but are not needed to interpret the primary AUC claim.

B Metric Definitions

For a run with curve values u_j at timesteps t_j over horizon $T = 150000$, we compute area under the learning curve with trapezoidal integration,

$$\mathrm{AUC}(u) = \frac{1}{T} \sum_{j=1}^{J-1} \frac{u_j + u_{j+1}}{2} (t_{j+1} - t_j).$$

If the first or last recorded point is not at 0 or T, the nearest observed endpoint value is extended to the boundary. Train AUC uses episodic training returns; evaluation AUC uses deterministic evaluation mean returns.

For environment e and metric m, normalization uses the vanilla batch-size-512 environment scale

$$s_{e,m} = \max\left(\left| \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} z_{\mathrm{vanilla},e,s,m} \right|, 1 \right), \qquad \tilde{z}_{v,e,s,m} = z_{v,e,s,m} / s_{e,m},$$

where $\mathcal{S} = \{6, \ldots, 15\}$ and z is the raw run-level metric. Aggregate normalized IQM sorts all 30 normalized records for a variant and averages the middle 50%:

$$\mathrm{IQM}(\tilde{z}_{1:n}) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \tilde{z}_{(i)}, \quad \mathcal{I} = \{\lfloor n/4 \rfloor + 1, \ldots, \lceil 3n/4 \rceil\}.$$

Paired effects average matched environment/seed differences $\tilde{z}_{v,e,s,m} - \tilde{z}_{b,e,s,m}$. Aggregate intervals use 2,000 percentile-bootstrap resamples over run-level records or paired tuples with RNG seed 1 in the analysis script. Per-environment intervals in Table 7 use 2,000 percentile-bootstrap resamples over the ten paired seeds, with deterministic seeds 1000–1011 in the figure/table script.

C Additional Learning Curves

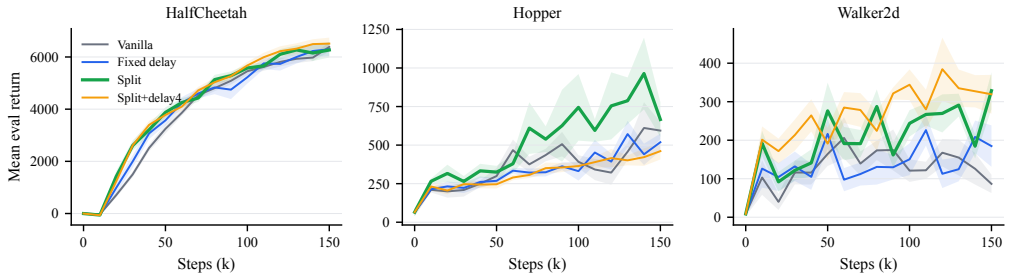

Figure 4: Mean deterministic evaluation returns over training. Shaded regions show standard error over seeds 6–15.

D Secondary Stability and Runtime Measurements

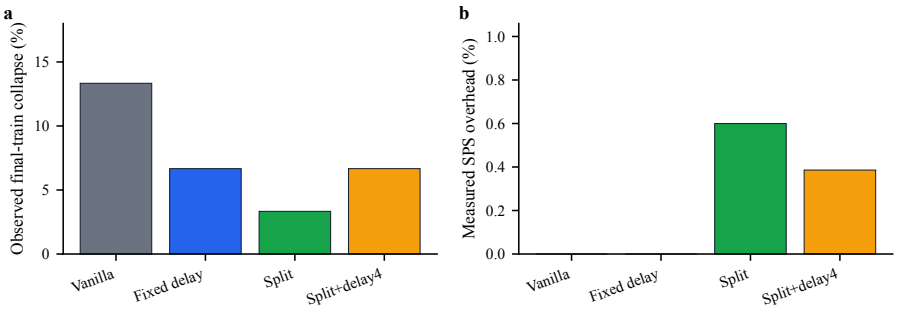

Figure 5: Secondary evidence. Left: observed final-training-return collapse rate, defined as normalized final return below 0.5 under the vanilla-baseline environment scale or raw final return below 0. Right: measured SPS overhead relative to vanilla SAC under the reported evaluation grid.

Table 5: Final-return disclosure for the reported evaluation grid. Values are normalized IQM with 95% bootstrap intervals over 30 environment-seed runs. Final returns are reported for transparency but are not the headline supported claim. Eval wins are shown versus vanilla; versus fixed-delay.

Variant	Final train	Final eval	Final-eval wins	Collapse count
Vanilla	0.945 [0.817, 1.111]	0.951 [0.825, 1.074]	–	4/30
Fixed delay	1.155 [0.990, 1.425]	1.003 [0.865, 1.270]	15/30; –	2/30
Split	**1.538** [1.193, 2.210]	**1.433** [1.062, 2.122]	19/30; 21/30	1/30
Split+delay4	1.401 [0.997, 2.420]	1.239 [0.907, 1.967]	16/30; 20/30	2/30

E Per-Environment Effect Table

Table 6: Per-environment normalized evaluation-AUC mean differences for the reported evaluation grid on held-out seeds 6–15. Split is the main method. Split+delay4 combines split critic batches with actor updates delayed to every fourth critic update and is treated as a secondary ablation.

Environment	Split-Vanilla	Split-Fixed	Split+delay4-Vanilla	Split+delay4-Fixed
HalfCheetah	+0.083	+0.051	+0.100	+0.068
Hopper	+0.456	+0.527	-0.124	-0.053
Walker2d	+0.561	+0.510	+1.022	+0.970

Table 7: Per-environment paired normalized evaluation-AUC uncertainty for the reported evaluation grid. Intervals use 2,000 percentile-bootstrap resamples over the ten paired held-out seeds in each environment; deterministic bootstrap seeds 1000–1011 are used by the figure/table script.

Environment	Variant	Baseline	Mean diff	95% CI	Wins
HalfCheetah	Split	Vanilla	+0.083	[-0.014, +0.185]	6/10
HalfCheetah	Split	Fixed delay	+0.051	[-0.044, +0.143]	6/10
HalfCheetah	Split+delay4	Vanilla	+0.100	[+0.026, +0.170]	7/10
HalfCheetah	Split+delay4	Fixed delay	+0.068	[-0.015, +0.145]	7/10
Hopper	Split	Vanilla	+0.456	[+0.162, +0.855]	8/10
Hopper	Split	Fixed delay	+0.527	[+0.238, +0.929]	10/10
Hopper	Split+delay4	Vanilla	-0.124	[-0.268, +0.009]	3/10
Hopper	Split+delay4	Fixed delay	-0.053	[-0.220, +0.105]	5/10
Walker2d	Split	Vanilla	+0.561	[+0.299, +0.836]	8/10
Walker2d	Split	Fixed delay	+0.510	[+0.307, +0.720]	9/10
Walker2d	Split+delay4	Vanilla	+1.022	[+0.740, +1.312]	10/10
Walker2d	Split+delay4	Fixed delay	+0.970	[+0.765, +1.148]	10/10

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/f514cec81cb148559cf475e7426eed5e-Paper.pdf.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1etN1rtPB.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018. DOI: 10.1609/aaai.v32i1.11694. URL https://ojs.aaai.org/index.php/AAAI/article/view/11694.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xCVJMsPv3RT.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G. M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL https://www.jmlr.org/papers/v23/21-1342.html.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. DOI: 10.1109/IROS.2012.6386109. URL https://doi.org/10.1109/IROS.2012.6386109.

Hado van Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, volume 23, 2010. URL https://proceedings.neurips.cc/paper/2010/hash/091d584fced301b442654dd8c23b3fc9-Abstract.html.