Extended Abstract: Algorithm Sequencing and Curriculum Learning in Deep Reinforcement Learning

Deep reinforcement learning (RL) algorithms are typically evaluated as fixed, end-to-end training procedures. However, the demands of training are not uniform across time. Early learning may benefit from useful state coverage or non-random initialization, middle learning may require sample-efficient online improvement, and late learning may favor stable policy refinement. This project studies whether RL algorithms should instead be treated as phase-specific components whose sequencing, switch timing, and transfer mechanisms affect performance under a fixed online interaction budget. We investigate algorithm sequencing in continuous-control MuJoCo environments, focusing primarily on Hopper-v4 and Walker2d-v4, with additional short-horizon checks on HalfCheetah-v4 and Ant-v4. Our study compares standalone Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO) baselines against fixed and adaptive handoff schedules, including SAC $\rightarrow$ PPO, PPO $\rightarrow$ SAC, behavior cloning (BC) warm starts, Advantage-Weighted Actor-Critic (AWAC) warm starts, interleaved BC anchoring, and curriculum-style easy-environment SAC pretraining.

We frame a training run as a schedule over a fixed online budget. At phase boundaries, information may be transferred through policy behavior, value estimates, replay data, or offline/curriculum-derived policy priors. Policy transfer is performed through behavioral distillation rather than direct weight loading. We additionally test value-initialization strategies, including random initialization, self-warmup, and source-aligned value transfer. For adaptive sequencing, we use a simple no-improvement trigger that switches after a minimum first phase and multiple stagnant evaluation checkpoints.

Our results show that sequencing is not a universal replacement for a strong single algorithm. In our experiments, SAC remains the strongest standalone baseline on both Hopper-v4 and Walker2d-v4. On Hopper-v4, SAC achieves a final return of 2895.6 and normalized AUC of 2260.239, while on Walker2d-v4 it achieves a final return of 3310.4 and normalized AUC of 1642.230. This weakens the original hypothesis that PPO can reliably serve as a late-stage stabilization phase after SAC. The reverse direction reveals a more favorable phase allocation. In PPO $\rightarrow$ SAC, early switches substantially outperform late switches, suggesting that PPO can provide a short warm-start phase only if SAC receives sufficient remaining budget to improve. For example, in self-warmup PPO $\rightarrow$ SAC experiments, the 25% switch achieves normalized AUCs of 1411.49 on Hopper-v4 and 1280.69 on Walker2d-v4, compared with 392.75 and 349.03 for 75% switches. A simple adaptive no-improvement trigger switches near 150k steps and performs strongly, especially on Walker2d-v4, where it reaches an AUC of 1498.34 and final return of 3777.64. These results suggest that adaptive switching can reduce sensitivity, though broader validation is needed.

Offline and curriculum-assisted experiments further support the importance of target-learner compatibility. BC $\rightarrow$ SAC, AWAC $\rightarrow$ SAC, and Easy SAC $\rightarrow$ SAC produce strong results, whereas BC $\rightarrow$ PPO and AWAC $\rightarrow$ PPO remain weak. The full three-phase BC $\rightarrow$ SAC $\rightarrow$ PPO scheduler also underperforms BC $\rightarrow$ SAC, reinforcing that handoffs into PPO are fragile under the tested protocol. Value initialization is similarly mixed as source-aligned and self-warmup value transfer do not provide universal gains, and confidence intervals include zero in the main SAC $\rightarrow$ PPO ablation.

Overall, this work supports a phase-allocation view of deep RL training. Algorithm sequencing is useful because phase boundaries expose compatibility effects between the information produced by one learner and the update dynamics of the next. From our tests, schedules that end in SAC are consistently more effective than schedules that switch into PPO. Thus, the central contribution is a mechanistic framework for reasoning about when RL algorithms should switch, supported by empirical evidence on timing, transfer, adaptive switching, and offline/curriculum initialization.

Algorithm Sequencing and Curriculum Learning in Deep Reinforcement Learning

Stanford CS224R
Repo: https://github.com/abhinav-chinta/CS224R-final-project

Ryan D'Cunha **Ethan Hersch** **Abhinav Chinta**
Department of Computer Science, Stanford University
{rdcunha, ehersch, achinta}@stanford.edu

Abstract

Deep reinforcement learning algorithms are typically evaluated as fixed end-to-end training procedures, but the demands of learning change over time. Early training may require exploration or non-random initialization, middle training may require sample-efficient improvement, and late training may require policy preservation. We study algorithm sequencing with respect to phase-allocation. Given a fixed online interaction budget, when should reinforcement learning algorithms switch, and what information should transfer across phases? We evaluate PPO, SAC, fixed and adaptive PPO/SAC handoffs, value-initialization strategies, behavior cloning, AWAC and IQL warm starts, interleaved imitation anchoring, and curriculum-style Easy SAC pretraining in MuJoCo continuous-control environments. Our results show that sequencing is not a universal replacement for single algorithms. SAC remains the strongest online baseline, and SAC $\rightarrow$ PPO handoffs fail to preserve SAC-level performance despite policy distillation and value warmup. In contrast, PPO $\rightarrow$ SAC works better when switching occurs early or adaptively, preserving enough budget for SAC to improve. Offline, imitation, and curriculum warm starts are most effective when transferred into SAC, while handoffs into PPO remain fragile. These findings support a compatibility view of sequencing: phase schedules help only when the target learner can preserve and improve the source phase.

1 Introduction

Deep reinforcement learning (RL) has produced strong empirical results across continuous-control, robotics, and simulated decision-making tasks, yet training remains sensitive to algorithm choice, initialization, and interaction budget. Standard empirical practice treats an RL algorithm as a fixed end-to-end training procedure. A policy is initialized, trained with a single update rule, and evaluated by its final return or sample efficiency. This framing obscures the fact that RL training is not a homogeneous process. The demands placed on the learner early in training may differ substantially from those later in training. Early learning often requires broad exploration, non-random behavior, and useful state coverage. Intermediate learning may require sample-efficient improvement from collected experience. Late learning may require stable updates that preserve already useful behavior.

This observation motivates a phase-based view of RL training. Rather than treating algorithms as monolithic choices, one can ask whether different algorithms are better suited to different stages of learning. On-policy methods such as Proximal Policy Optimization (PPO) [1] are often valued for stable policy updates, while off-policy methods such as Soft Actor-Critic (SAC) [2] can reuse replay data and often achieve strong sample efficiency in continuous-control domains. Offline, imitation-based, and curriculum-based methods can also provide useful initial policies or state distributions before online fine-tuning. These differences suggest that the relevant design question may not be which algorithm is universally best, but how learning should be allocated across algorithms, initialization mechanisms, and transfer interfaces.

We study this question through algorithm sequencing, involving training schedules that divide a fixed interaction budget across learning phases. The target learner may inherit policy behavior, value

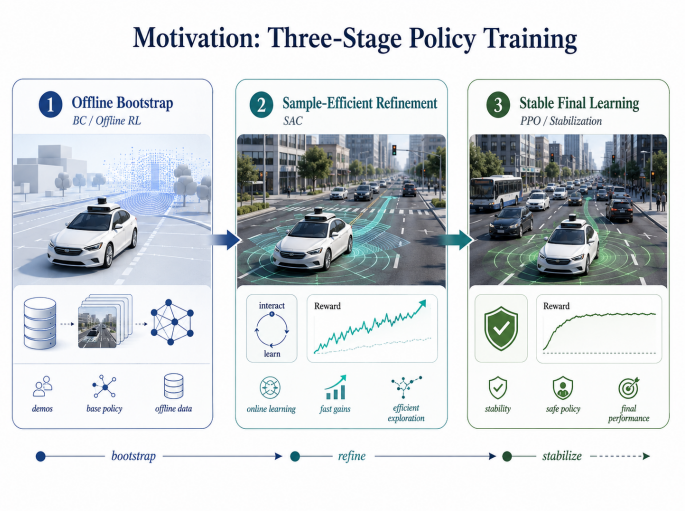

Figure 1: **Motivating phase-based view of RL training.** We study whether RL training can be decomposed into warm-start, online-improvement, and refinement phases.

estimates, replay data, or a pretrained policy prior. Whether this inheritance is useful depends on the source of the transferred information and update dynamics of the target algorithm. A policy that is beneficial for one learner may be poorly preserved by another. A value estimate calibrated under one objective may be mismatched to a different target update. Similarly, a warm-start policy may only be useful if the subsequent learner can preserve and improve it rather than overwrite or destabilize it.

This paper investigates algorithm sequencing in MuJoCo continuous-control environments [3], focusing on interactions among PPO, SAC, and several warm-start mechanisms. We evaluate sequencing as a mechanistic question rather than a simple leaderboard problem: when does a first phase provide useful initialization for a second phase, and when does switching disrupt a productive learner? We consider online handoffs such as PPO $\rightarrow$ SAC and SAC $\rightarrow$ PPO, value-initialization variants at phase boundaries, adaptive switching based on stalled improvement, and offline-assisted schedules using behavior cloning (BC) and Advantage-Weighted Actor-Critic (AWAC) [4] from D4RL expert data [5]. We also evaluate curriculum-style warm starts in which SAC is pretrained in an easier environment before being transferred to the target task. Together, these settings test whether sequencing effects arise from algorithm direction, switch timing, value transfer, offline policy priors, curriculum initialization, or the interaction between a warm-start phase and the online target learner.

Our study supports a cautious view of algorithm sequencing. Strong single algorithms remain difficult to outperform under matched online budgets. However, handoff experiments reveal that switch direction and timing substantially affect learning, adaptive switching can reduce sensitivity to fixed switch fractions, value transfer is environment-dependent, and warm-start policies interact strongly with the target online algorithm. In particular, warm-starting SAC from imitation, offline RL, or curriculum-based pretraining can be effective, whereas switching from a strong SAC trajectory into PPO often fails to preserve performance. These results suggest that sequencing should not be understood as a universal performance trick, but as a framework for reasoning about phase allocation, transfer compatibility, and adaptive training schedules in deep RL.

The contributions of this paper are:

1. We frame deep RL training as a phase-allocation problem in which algorithms are treated as stage-specific components rather than fixed end-to-end choices.
2. We formulate algorithm handoffs as transfer problems involving policy behavior, value estimates, replay data, and pretrained policy priors.
3. We analyze switch direction and timing under matched online budgets, including fixed PPO/SAC handoffs and an adaptive no-improvement switching rule.
4. We evaluate offline-, imitation-, and curriculum-assisted warm starts and show that their effectiveness depends strongly on the target online learner.

2 Related Work

Online deep reinforcement learning. Deep RL algorithms are commonly compared as fixed end-to-end procedures. PPO [1] is an on-policy policy-gradient method that uses a clipped surrogate objective to improve update stability, building on trust-region ideas from TRPO [6]. SAC [2] instead uses an off-policy maximum-entropy actor-critic objective, allowing replay-based learning while encouraging exploration through entropy regularization. These methods represent different points in the stability–sample-efficiency tradeoff. PPO is often valued for conservative, stable updates, whereas SAC often achieves strong sample efficiency in continuous-control domains through off-policy replay. This contrast motivates our study of whether such algorithms should be treated as phase-specific components rather than mutually exclusive training choices.

Warm starts from offline, imitation, and curriculum learning. A separate line of work studies how prior data or easier training settings can initialize or accelerate RL. Behavior cloning treats imitation learning as supervised prediction of expert actions and has long been used to obtain non-random policies from demonstrations [7]. D4RL provides standardized offline RL datasets, including expert and mixed-quality data, for evaluating data-driven RL methods [5]. AWAC [4] and IQL [8] address offline-to-online or offline RL by learning from previously collected data while mitigating failures from distribution shift. Curriculum RL similarly studies how agents can benefit from structured sequences of tasks, environments, or difficulty levels [9]. These works motivate our BC, AWAC, and curriculum-style Easy SAC warm starts.

Transfer, sequencing, and adaptive phase allocation. Prior work on transfer RL and policy reuse examines how policies or value functions learned in one task can accelerate learning in a related target task [10, 11]. Our work focuses on handoffs between learning algorithms within a single training schedule. A phase transition may transfer policy behavior, value estimates, replay data, or a pretrained policy prior from one algorithmic phase to another. This makes target-learner compatibility an important factor. A policy useful for SAC may not be preserved by PPO, and a value estimate learned under one objective may be poorly calibrated for another. We also study adaptive phase allocation, where a no-improvement trigger switches algorithms after evaluation performance stalls rather than relying only on fixed switch fractions. By comparing SAC $\rightarrow$ PPO, PPO $\rightarrow$ SAC, BC/AWAC/IQL-assisted warm starts, interleaved BC anchoring, and curriculum-style Easy SAC $\rightarrow$ SAC pretraining, we position algorithm sequencing as a framework for studying compatibility between initialization, transfer substrate, target algorithm, and remaining interaction budget.

3 Approach

We study algorithm sequencing under a fixed online interaction budget B, with a training run:

$$(A_1, T_1), (A_2, T_2), \ldots, (A_k, T_k), \qquad \sum_{i=1}^{k} T_i = B,$$

where each phase uses an RL, imitation, offline, or curriculum-based learner for T_i environment steps. At phase boundaries, the target learner may inherit policy behavior, value estimates, replay data, or a pretrained policy prior from the source phase. This formulation investigates if different algorithms are suited to different stages of training.

3.1 Phase-Based Scheduling

Our schedules are organized around testing three possible phase roles (Fig. 1 and Fig. 2). A warm-start phase aims to produce non-random behavior or useful state coverage through PPO warmup, BC, AWAC, or SAC pretraining in an easier environment. An online improvement phase uses a sample-efficient learner, SAC, to improve performance under limited interaction. A refinement phase tests whether a more conservative learner, PPO, can preserve or stabilize a policy late in training. We treat PPO refinement as a hypothesis rather than an assumed property of the scheduler.

We separate schedules into two budget categories. Online-only schedules, including PPO, SAC, PPO $\rightarrow$ SAC, SAC $\rightarrow$ PPO, and adaptive PPO $\rightarrow$ SAC, use only target-environment interaction and are compared under matched online budgets. Offline or pretraining-assisted schedules, including BC $\rightarrow$ SAC, BC $\rightarrow$ PPO, AWAC $\rightarrow$ SAC, AWAC $\rightarrow$ PPO, BC $\rightarrow$ SAC $\rightarrow$ PPO, interleaved BC anchoring, and Easy SAC $\rightarrow$ SAC, use additional demonstrations, offline updates, or modified-environment pretraining. We therefore report these separately from strictly online matched-budget comparisons.

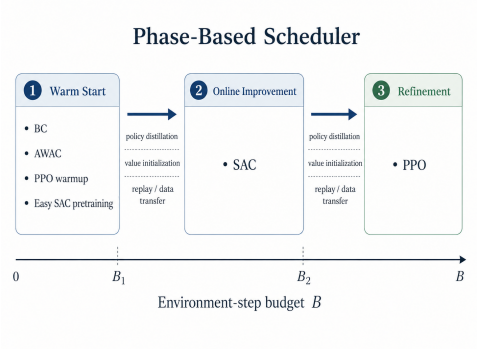

Figure 2: **Phase-based training scheduler.** We model RL training as a sequence of phases under a fixed online budget. Warm-start phases can use PPO, BC, AWAC, or Easy SAC pretraining. Online improvement is primarily performed by SAC and PPO is tested as a possible final refinement phase. Handoffs transfer policy behavior, value estimates, replay data, or pretrained initialization.

3.2 Handoff Mechanisms

A schedule is defined by the order of algorithms and by what is transferred across the phase boundary. In our implementation, PPO and SAC use different actor architectures, so we do not directly transfer policies. We use behavioral distillation. Given states $\mathcal{S}$ sampled from source rollouts, replay data, or offline demonstrations, the target actor is initialized by matching the source actor's mean action:

$$\mathcal{L}_{\text{distill}}(\theta) = \mathbb{E}_{s\sim\mathcal{S}}\left[\|\mu_\theta(s) - \mu_{\text{src}}(s)\|_2^2\right].$$

This gives the target learner a behaviorally similar initialization while preserving its architecture.

We also ablate value initialization at online handoffs. Holding policy distillation fixed, the target value function or critic is initialized using one of three variants: random, self-warmup, source-aligned. In the random condition, the target value function is initialized normally. In self-warmup, the target value function is warmed up using its own native objective before online training resumes. In source-aligned transfer, the target value function is trained to match value targets derived from the source phase, such as SAC critic estimates for SAC $\rightarrow$ PPO or PPO rollout returns for PPO $\rightarrow$ SAC. This tests whether handoff quality is explained by policy transfer, value transfer, or the remaining budget.

Switch timing is the final handoff variable. For fixed schedules, the switch occurs after a fraction f of the budget,

$$t_s = \lfloor fB \rfloor, \qquad f \in \{0.25, 0.50, 0.75\}.$$

This sweep tests whether the first algorithm is useful only as an early warm start or should consume most of the budget. We also evaluate an adaptive PPO $\rightarrow$ SAC schedule in which PPO runs for a minimum of $0.25B$ steps and then switches to SAC after evaluation return fails to improve for three checkpoints. This simple plateau rule tests whether progress-based switching can reduce sensitivity to manually chosen switch fractions.

3.3 Offline, Imitation, and Curriculum Warm Starts

Offline-assisted schedules test whether a prior policy is useful only when the target online learner can preserve and improve it (Fig. 3). BC provides a policy-only imitation prior from expert demonstrations. AWAC provides a value-aware offline RL prior using advantage-weighted updates. We transfer both into PPO and SAC, yielding BC $\rightarrow$ PPO, BC $\rightarrow$ SAC, AWAC $\rightarrow$ PPO, and AWAC $\rightarrow$ SAC. We also test BC $\rightarrow$ SAC $\rightarrow$ PPO, which directly instantiates the proposed warm-start, online-improvement, and refinement decomposition.

To test whether repeated imitation regularization helps preserve useful behavior, we evaluate interleaved BC anchoring during SAC, where short BC-style updates are periodically applied every K environment steps. Finally, we evaluate a curriculum-style Easy SAC $\rightarrow$ SAC schedule, where SAC is first trained in a simplified environment and then transferred to the original task. This differs from BC and AWAC because the warm start comes from interaction with a modified task rather than demonstrations. Because these methods use additional data or pretraining, they are interpreted as transfer-compatibility experiments rather than pure online-budget comparisons.

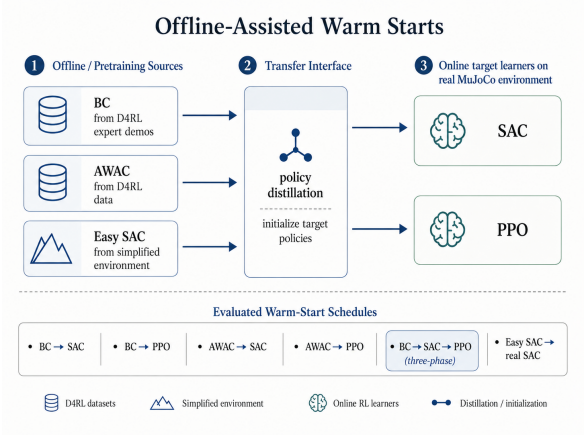

Figure 3: **Offline and curriculum-assisted schedules.** BC and AWAC are trained from D4RL demonstrations, while Easy SAC is pretrained in a simplified environment. These source policies are transferred into PPO or SAC target learners, enabling comparisons of how the same warm start interacts with different online algorithms.

3.4 Experimental Protocol and Metrics

We evaluate schedules primarily on Hopper-v4 and Walker2d-v4, with HalfCheetah-v4 and Ant-v4 used only in short-horizon checks. Online training uses a budget of 500k environment steps. Core online experiments use five seeds when available, while adaptive schedules, curriculum transfer, interleaved-BC sweeps, and long-horizon checks use fewer seeds due to compute constraints. We report these lower-seed experiments as supporting evidence rather than definitive comparisons. Selected Hopper-v4 runs are extended to 1M steps to test whether SAC continues improving rather than saturating before the handoff.

We report final return, normalized area under the learning curve (AUC), seed variance, worst-seed return, and collapse count. Final return measures end-of-budget policy quality, while AUC captures sample efficiency across the full learning curve. Worst-seed return and collapse count measure robustness, which is important because a schedule that occasionally performs well but frequently fails is not a reliable improvement. For handoff experiments, we additionally track switch step, switch reason, phase identity, and handoff transients when available. These metrics allow us to distinguish final performance from learning speed, stability, and transfer compatibility.

4 Experiments

SAC → PPO handoff and value initialization. We first test whether a strong off-policy SAC phase can be converted into a PPO refinement phase under a matched 500k-step online budget. In each handoff run, SAC trains for the first 50% of the budget, the policy is transferred to PPO through behavioral distillation, and PPO trains for the remaining steps. To isolate whether critic transfer improves the handoff, we hold policy distillation fixed and vary PPO value initialization: random initialization, self-warmup, and source-aligned value regression. We compare these schedules against standalone SAC and PPO baselines on Hopper-v4 and Walker2d-v4.

Continued SAC is the strongest online method in both environments. On Hopper-v4, SAC reaches a final return of 2895.6 and normalized AUC of 2260.2, while the best SAC → PPO variant reaches only 439.2 final return and 1219.3 normalized AUC. On Walker2d-v4, SAC reaches a final return of 3310.4 and normalized AUC of 1642.2, while the best SAC → PPO variant reaches 646.6 final return and 718.3 normalized AUC. SAC → PPO improves over PPO alone in AUC, but it does not preserve the trajectory of continued SAC. Full learning curves and return summaries are provided in the Appendix Figures 8, 9, and 10.

The value-initialization ablation shows no reliable critic-transfer benefit. Relative to random PPO value initialization, self-warmup and source-aligned transfer have negative AUC deltas on Hopper-v4 but positive deltas on Walker2d-v4, with all paired confidence intervals crossing zero (Fig. 4). These results suggest that the main limitation is not simply poor PPO value initialization. Rather, under

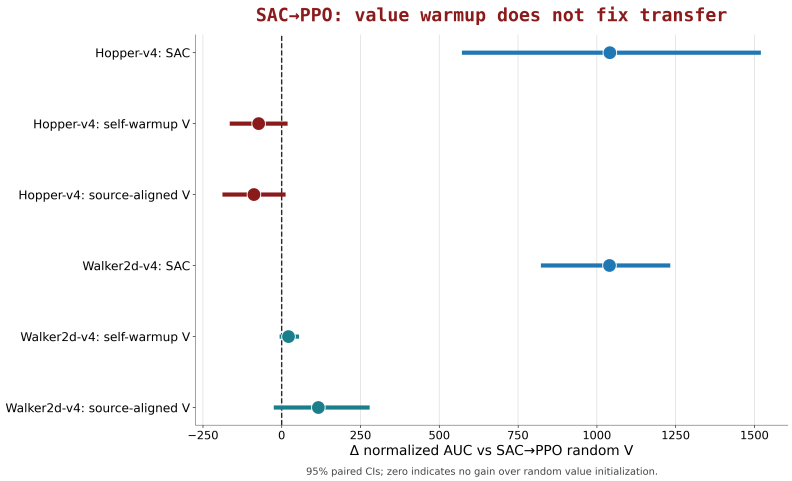

Figure 4: **SAC → PPO value warmup does not fix transfer.** Points show normalized AUC differences relative to SAC → PPO with random PPO value initialization, with 95% paired confidence intervals. Self-warmup and source-aligned value initialization do not consistently improve over random value initialization. Both have confidence intervals crossing zero. This suggests value transfer alone does not explain failure of SAC → PPO handoffs.

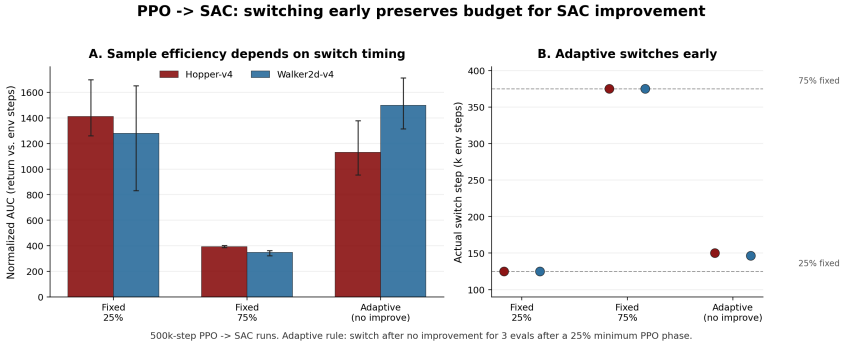

Figure 5: **PPO → SAC benefits from early or adaptive switching.** (A) Normalized AUC is higher when PPO switches to SAC after 25% of the budget than after 75%, indicating that SAC needs sufficient remaining interaction budget to exploit the warm start. The adaptive no-improvement rule switches early and achieves competitive AUC, especially on Walker2d-v4. (B) The adaptive rule switches near the early fixed-switch regime rather than waiting until the late 75% switch point.

this protocol, switching from SAC to PPO replaces a still-productive off-policy learner with a target learner that does not reliably preserve or improve the transferred policy.

PPO → SAC switch timing and adaptive handoff. We next test the reverse direction. Can PPO can serve as a short warm-start phase before transferring into SAC? Unlike SAC → PPO, this schedule assigns SAC the role of the main online improvement learner. We evaluate fixed PPO → SAC switches at different fractions of the budget and compare them with an adaptive no-improvement rule that switches after PPO fails to improve for three evaluation checkpoints following a 25% minimum PPO phase.

As shown in Fig. 5, PPO → SAC is strongly timing-sensitive. Early switching substantially outperforms late switching: the 25% fixed switch reaches normalized AUCs of 1411.5 on Hopper-v4 and 1280.7 on Walker2d-v4, compared with 392.8 and 349.0 for the 75% switch. This suggests that PPO is more useful as a brief warm-start phase than as the dominant learner. If PPO consumes most of the budget, SAC can not fully exploit its sample-efficient off-policy updates.

The adaptive no-improvement rule also switches near the early-switch regime rather than the late 75% switch point (Fig. 5B). It switches near 150k steps in both environments and achieves competitive performance. On Walker2d-v4, adaptive PPO → SAC reaches the strongest AUC among these timing

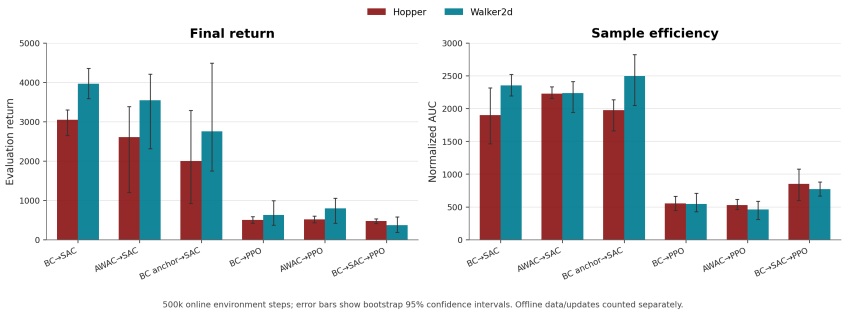

Figure 6: **Offline and imitation warm starts are most effective when transferred into SAC.** Across Hopper-v4 and Walker2d-v4, BC → SAC and AWAC → SAC achieve substantially higher final return and normalized AUC than the corresponding handoffs into PPO. Interleaved BC anchoring improves sample efficiency in some settings but does not consistently improve final return. The three-phase BC → SAC → PPO schedule performs poorly, suggesting that the final PPO phase fails to preserve the policy produced by the SAC phase.

variants. These results support the interpretation that PPO can provide a stable initial policy or state distribution, but SAC must receive enough remaining budget to perform the main improvement phase.

Offline and imitation warm starts. We next evaluate whether offline or imitation-derived policies provide useful initializations for online RL. Because these schedules use additional demonstrations or offline updates, we report them separately from the matched online-only PPO/SAC comparisons. We compare BC and AWAC warm starts transferred into either SAC or PPO, as well as interleaved BC anchoring and the three-phase BC → SAC → PPO schedule.

As shown in Fig. 6, the same warm start behaves very differently depending on the target learner. BC → SAC achieves strong returns on both Hopper-v4 and Walker2d-v4, while BC → PPO remains near PPO-level performance. The same pattern holds for AWAC as AWAC → SAC is competitive, but AWAC → PPO is weak. The three-phase BC → SAC → PPO schedule underperforms BC → SAC, reinforcing the failure mode observed in SAC → PPO. Additional offline-transfer summaries, including IQL warm starts, are provided in Appendix Figures 14, 15, and 16.

Curriculum-style SAC pretraining. We also test whether pretraining SAC in an easier environment can provide a useful curriculum initialization for the real task. In this setting, Easy SAC is trained in a forgiving version of the environment and then distilled into SAC on the original benchmark. This experiment is not directly comparable to pure online SAC because it uses additional interaction in a modified environment.

Easy SAC → SAC performs competitively with or above the SAC baseline on Hopper-v4, reaching a final return of 3333.1 and normalized AUC of 2133.9. This supports the broader warm-start pattern of useful initial behavior helping when the target learner is SAC. However, the gain should be interpreted as curriculum-assisted transfer rather than a matched-budget improvement. Additional curriculum learning curves and stability diagnostics are in Appendix Figures 11, 12, and 13.

Synthesis across schedules. Fig. 7 summarizes the main empirical pattern from our experiments. The strongest schedules either continue SAC or hand into SAC. In contrast, SAC → PPO and BC → SAC → PPO are consistently below SAC parity despite receiving useful behavior from earlier phases. Offline- and curriculum-assisted methods such as BC → SAC, AWAC → SAC, and Easy SAC → SAC can exceed SAC on some aggregate metrics, but these runs use additional demonstrations, offline updates, or modified-environment pretraining and are not strict matched-budget improvements over online SAC. Thus, sequencing is not beneficial simply because multiple algorithms are combined. It helps when the first phase produces information that the target learner can preserve and improve, and when enough budget remains for that target learner to act on the initialization.

5 Analysis

The experiments support a phase-allocation interpretation of algorithm sequencing. The key finding is not that one algorithm should always replace another, but that a handoff is only useful when the

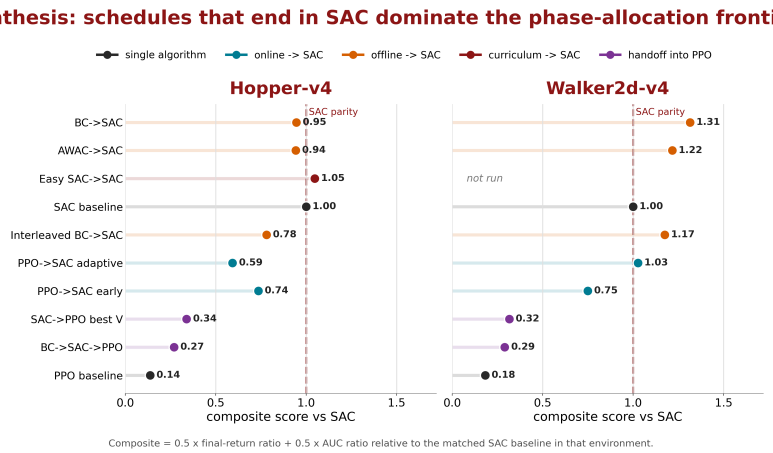

Figure 7: **Schedules that end in SAC define the strongest observed phase-allocation frontier.** Composite score is the average of final-return ratio and AUC ratio relative to the matched SAC baseline in each environment. The strongest schedules either continue SAC or hand into SAC, while schedules that hand into PPO remain consistently weak. Offline- and curriculum-assisted schedules use additional data or pretraining and should not be interpreted as strict matched-budget improvements over online SAC.

target learner can preserve and improve the information produced by the source phase. Across the tested settings, this compatibility is highly asymmetric as schedules that end in SAC are consistently more effective than schedules that switch into PPO.

Switching away from a productive learner is costly. The SAC $\rightarrow$ PPO experiments show that sequencing can fail even when the first phase produces useful behavior. Continued SAC dominates both final return and AUC, while SAC $\rightarrow$ PPO improves over PPO but remains far below the SAC trajectory. This indicates that the handoff is not failing because SAC produced a poor initialization, but because the PPO phase does not reliably preserve or improve the transferred policy. The value-initialization ablation strengthens this interpretation. If poor critic initialization were the main bottleneck, self-warmup or source-aligned value transfer should consistently improve the handoff. Instead, both variants have environment-dependent effects and confidence intervals that cross zero (Fig. 4). Thus, value transfer alone does not fix the instability of switching into PPO.

SAC is more effective as the target learner than the source learner. The reverse PPO $\rightarrow$ SAC experiments show that direction matters. PPO appears useful as a short warm-start phase, but only when SAC receives enough remaining budget to perform the main improvement. Early PPO $\rightarrow$ SAC switching substantially outperforms late switching, and the adaptive no-improvement rule tends to switch near the early fixed-switch regime (Fig. 5). This suggests that SAC's benefit comes not only from inheriting a better initial policy, but from having sufficient interaction budget to exploit that initialization through off-policy replay and sample-efficient updates. In this sense, PPO is better interpreted as a possible initializer than as a final refinement stage under the tested protocol.

Warm starts help only when the online learner can exploit them. The offline and imitation experiments provide the clearest evidence for target-learner compatibility. BC and AWAC warm starts are useful when transferred into SAC, but the same kinds of priors transfer poorly into PPO (Fig. 6). This pattern rules out the simple explanation that any strong initial policy is sufficient. Instead, the target update rule determines whether the prior is retained, overwritten, or improved. Interleaved BC anchoring further shows that preserving demonstrated behavior can improve sample efficiency in some settings but may constrain final performance. The IQL appendix results show the same failure mode as IQL $\rightarrow$ SAC performs strongly, while IQL $\rightarrow$ SAC $\rightarrow$ PPO loses much of the performance gained during the SAC phase (Appendix Fig. 14, Fig. 16).

Curriculum transfer supports the same mechanism. Easy SAC $\rightarrow$ SAC provides a separate form of warm start as the source policy comes from a simplified environment instead of demonstrations.

Its strong performance on Hopper-v4 suggests that curriculum pretraining can create useful behavior for the target task, but the result should be interpreted separately from matched online comparisons because it uses additional pretraining interaction. Mechanistically, however, it reinforces the same point as BC, AWAC, and IQL. Warm starts are most useful when the final online learner is able to preserve and improve them. Additional curriculum learning curves support this interpretation in Appendix Fig. 11 and Fig. 12.

Implications for algorithm sequencing. Taken together, these results argue against a naive view of combining algorithms. The strongest schedules are not those with the most phases, but those with compatible phase boundaries. In our experiments, schedules ending in SAC dominate the phase-allocation frontier, while handoffs into PPO consistently fall below SAC parity (Fig. 7). This does not imply that PPO can not be useful in sequenced training, but it does show that PPO did not function as a reliable late-stage stabilizer in our implementation. Better PPO retention mechanisms, stronger distillation objectives, KL constraints after handoff, or adaptive rules for deciding whether to switch at all may be necessary before PPO can serve as a dependable refinement phase.

Overall, the project reframes sequencing as a question of transfer compatibility rather than algorithm aggregation. A successful phase schedule must answer three questions: what information does the source phase produce, can the target learner preserve that information, and does enough budget remain for the target learner to improve it? Under the tested conditions, the answer is most often favorable when the final learner is SAC.

6 Conclusion

This project studied algorithm sequencing as a phase-allocation problem in deep RL. Instead of asking whether PPO, SAC, or offline methods are universally best, we evaluated when one training phase produces information that a later learner can preserve and improve. Across online handoffs, adaptive switching, offline warm starts, and curriculum pretraining, the central theme is that sequencing helps only when the phase boundary is compatible with the target learner.

Our strongest online baseline is SAC, and switching away from SAC into PPO is usually harmful under the tested protocol. SAC $\rightarrow$ PPO improves over PPO alone, but it fails to preserve SAC-level performance, and value initialization does not reliably fix this failure. In contrast, PPO $\rightarrow$ SAC is more promising when the switch occurs early or adaptively, suggesting that PPO can provide a short warm start while SAC performs the main sample-efficient improvement phase. Offline and imitation-based experiments reinforce this finding as BC, AWAC, and IQL warm starts are much more effective when transferred into SAC than PPO. Curriculum-style Easy SAC $\rightarrow$ SAC further supports the idea that useful initial behavior can help, provided the final learner can exploit it.

The broader takeaway is that algorithm sequencing is not a simple recipe for combining the strengths of multiple RL algorithms. More phases do not automatically improve performance. In our experiments, schedules that end in SAC dominate the phase-allocation frontier, while schedules that end in PPO often degrade strong intermediate policies. This suggests that future work should focus less on fixed algorithm orderings and more on transfer compatibility (when to switch, what to transfer, and whether the target learner can retain useful behavior after the handoff).

Future extensions could improve adaptive switching rules using diagnostics such as entropy, critic error, KL drift, or return plateaus. We can also develop stronger policy-retention mechanisms for PPO after handoff and test whether these phase-allocation patterns hold across broader environments and larger seed counts. Overall, this work suggests that sequencing is valuable not as a universal alternative to strong single algorithms, but it can be a framework for understanding how initialization, transfer, and remaining budget shape deep RL training.

Team contributions

Ryan implemented the SAC $\rightarrow$ PPO experiments as well as testing the different value initialization policies. Ethan ran PPO$\rightarrow$SAC experiments with adaptive scheduling and Easy SAC$\rightarrow$SAC experiments. Abhinav implemented curriculum style training, warm starts involving AWAC and BC, and W&B and Modal integration. All contributed to the paper and poster.

References

[1] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[2] 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*, pages 1861–1870. PMLR, 2018.

[3] 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*, pages 5026–5033. IEEE, 2012.

[4] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[5] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[6] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897. PMLR, 2015.

[7] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, volume 1. Morgan Kaufmann, 1989.

[8] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[9] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[10] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 720–727. ACM, 2006.

[11] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

A Additional Experimental Results

A.1 SAC → PPO Handoff Diagnostics

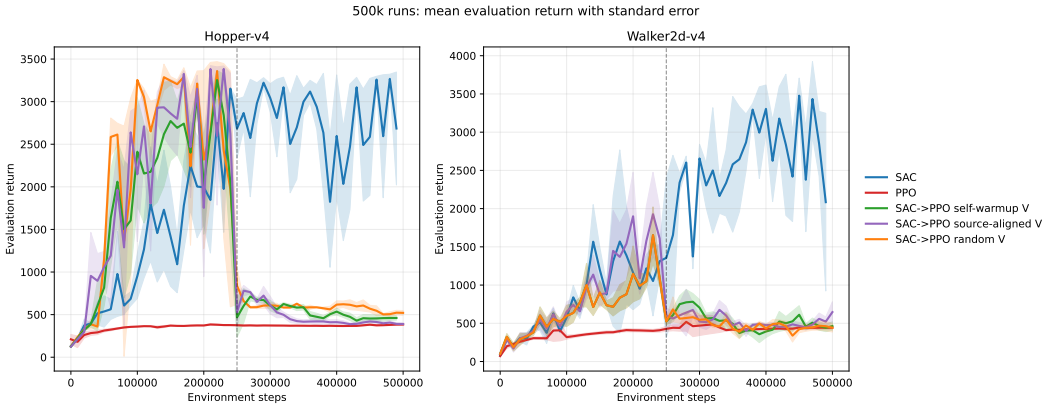

Figure 8: **Learning curves for 500k-step SAC → PPO handoffs.** Mean evaluation return is shown for standalone SAC, standalone PPO, and SAC → PPO handoffs on Hopper-v4 and Walker2d-v4. Handoff schedules train SAC for the first half of the online budget, transfer policy behavior to PPO through behavioral distillation, and then train PPO for the remaining steps. Continued SAC remains the strongest trajectory in both environments, while SAC → PPO improves over PPO but fails to preserve SAC-level performance after the switch.

A.2 Curriculum and Warm-Started SAC Diagnostics

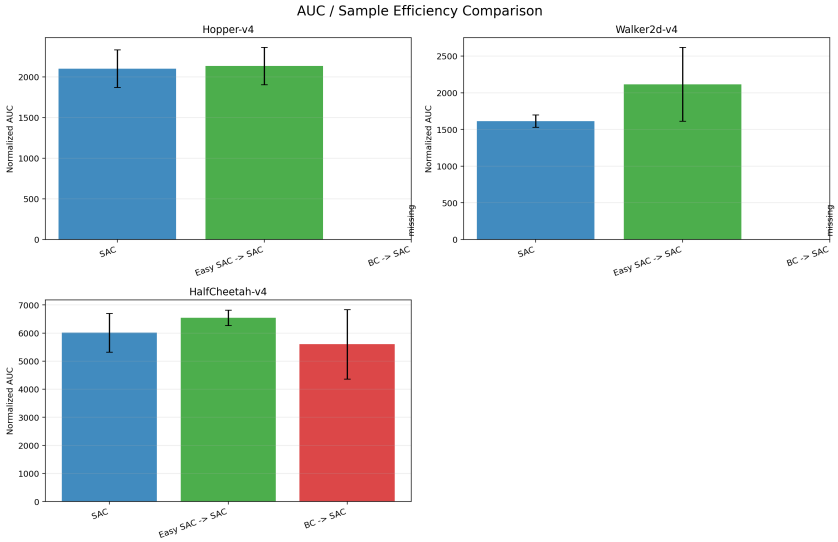

Figure 9: **AUC comparison for SAC, Easy SAC → SAC, and BC → SAC.** Normalized AUC summarizes sample efficiency over the full training run. Easy SAC → SAC is competitive with or above the SAC baseline in the shown environments, while BC → SAC is shown where available. These curriculum and offline-assisted runs use additional data or pretraining and are therefore interpreted separately from strictly online matched-budget comparisons.

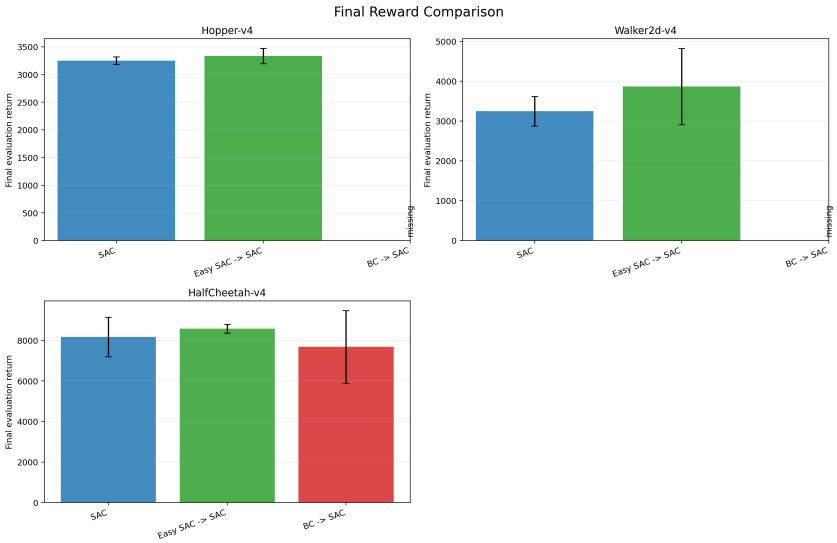

Figure 10: **Final return comparison for SAC, Easy SAC → SAC, and BC → SAC.** Final evaluation return is reported at the end of the online training budget. Easy SAC → SAC reaches comparable or higher final return than SAC in the displayed environments, supporting the claim that curriculum-style pretraining can provide a useful warm start when the target learner remains SAC.

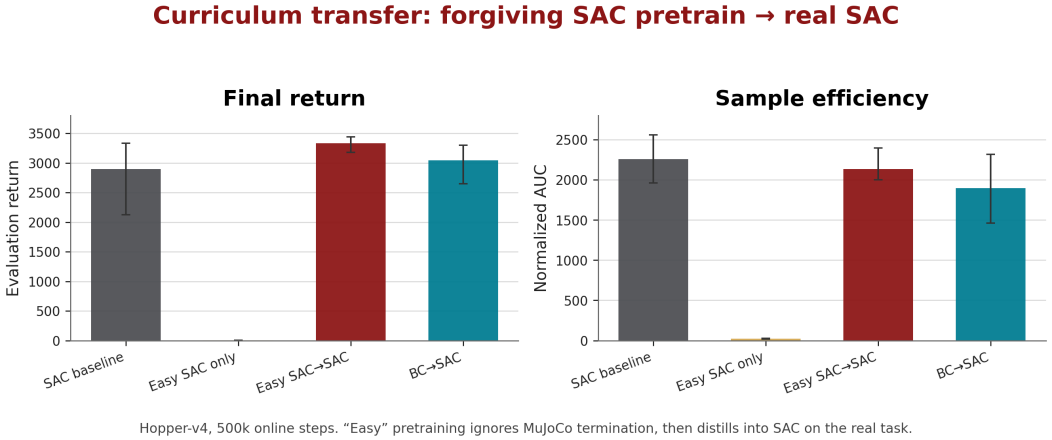

Figure 11: **Curriculum-style Easy SAC pretraining transfers effectively into real SAC.** On Hopper-v4, SAC is first pretrained in a forgiving environment variant and then distilled into SAC on the real task. Easy SAC → SAC achieves strong final return and sample efficiency relative to the SAC baseline and BC → SAC. Because Easy SAC pretraining uses additional interaction in a modified environment, this result is interpreted as curriculum-assisted transfer rather than a pure matched-budget online comparison.

A.3 Learning-Curve and Convergence Diagnostics

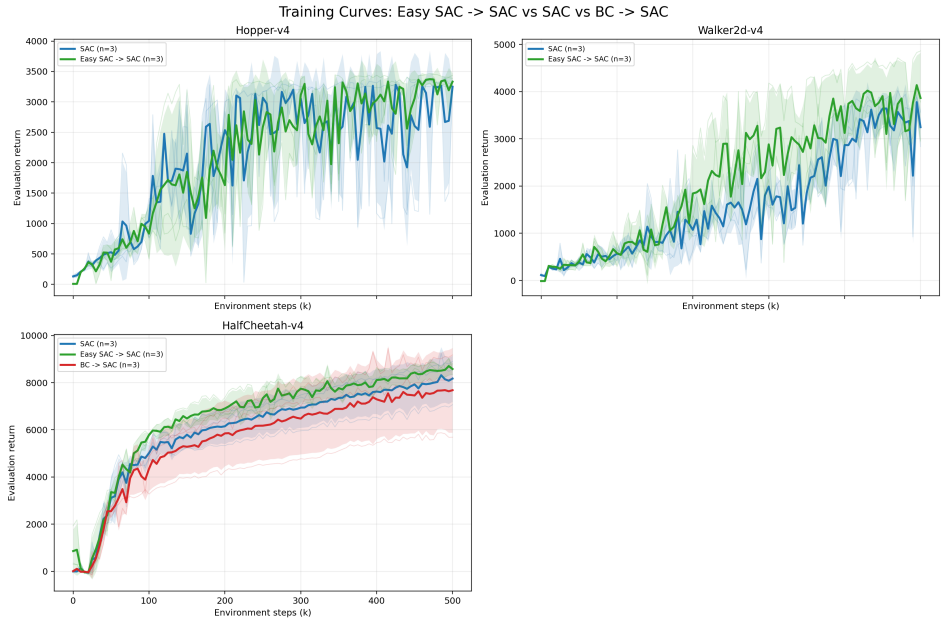

Figure 12: **Training curves for SAC, Easy SAC → SAC, and BC → SAC.** Learning curves show evaluation return over online environment steps for SAC baselines and warm-started SAC variants. Easy SAC → SAC often follows a competitive or improved trajectory relative to SAC, while BC → SAC is more variable across environments. These curves support the interpretation that warm starts are useful only when the target learner can preserve and improve the transferred behavior.

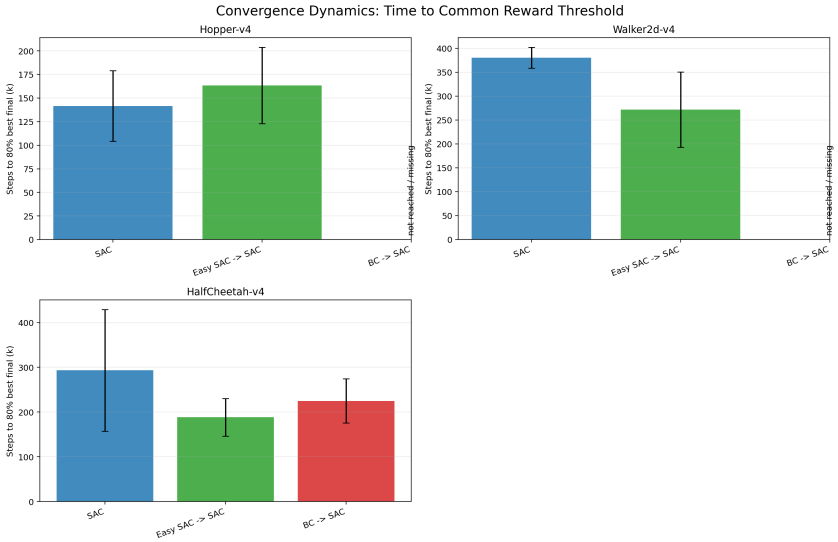

Figure 13: **Convergence dynamics measured by time to a common reward threshold.** Bars report the number of environment steps required to reach 80% of the final reward threshold. Lower values indicate faster convergence. Easy SAC → SAC reaches the threshold earlier than SAC in some environments, suggesting that curriculum-style pretraining can improve early learning dynamics, though this benefit is not uniform across tasks.

A.4 IQL Warm-Start Experiments

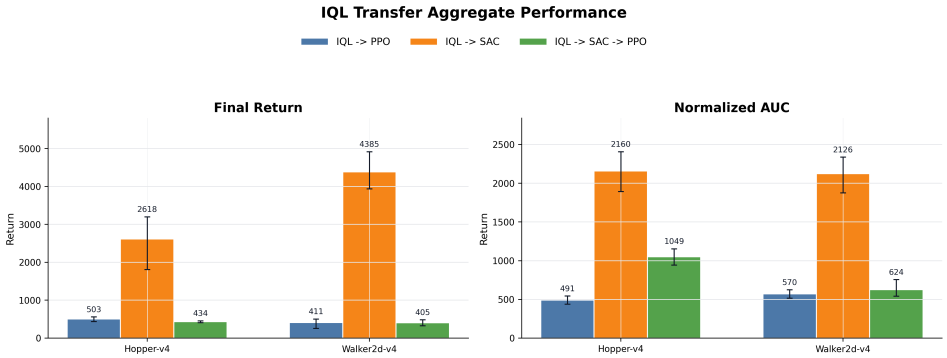

Figure 14: **Aggregate performance of IQL warm-start transfer schedules.** IQL → SAC achieves substantially higher final return and normalized AUC than IQL → PPO or IQL → SAC → PPO on both Hopper-v4 and Walker2d-v4. This mirrors the BC and AWAC results: offline warm starts are most effective when the online target learner is SAC, while adding a final PPO phase degrades performance.

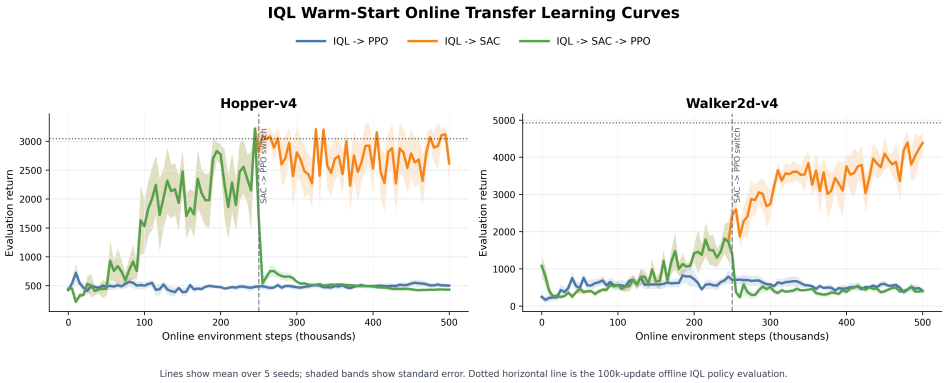

Figure 15: **Learning curves for IQL warm-start online transfer.** IQL → SAC improves sharply after the switch to SAC and maintains strong performance through the end of training. In contrast, IQL → PPO remains near low-return PPO behavior, and IQL → SAC → PPO loses much of the performance gained during the SAC phase. The vertical dashed line marks the SAC → PPO switch in the three-phase schedule.

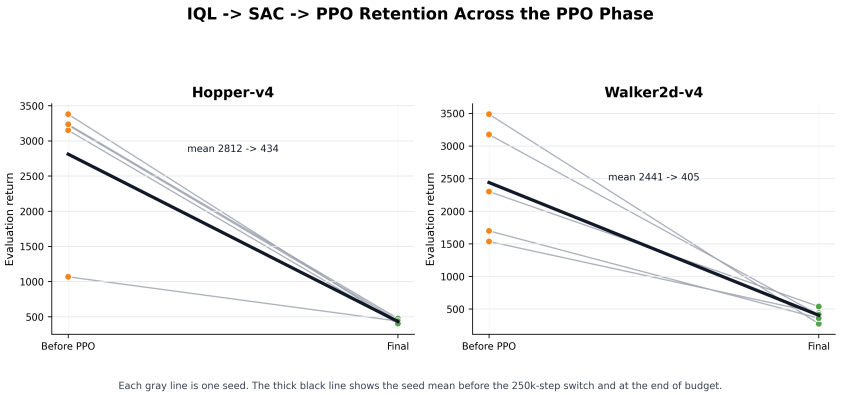

Figure 16: **Policy retention failure during the PPO phase of IQL → SAC → PPO.** Each gray line represents one seed, and the thick black line shows the seed mean before the PPO switch and at the end of training. In both Hopper-v4 and Walker2d-v4, evaluation return drops sharply after switching from SAC to PPO, indicating that the final PPO phase fails to retain the policy improvement produced by the SAC phase.