Conflict-Averse IL-RL: Resolving Gradient Conflicts for Stable Imitation-to-Reinforcement Learning Transfer

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement Learning (RL) and Imitation Learning (IL) offer complementary capabilities: RL can learn high-performing policies but is data-intensive, whereas IL enables rapid learning from demonstrations but is limited by the demonstrator's quality. Combining them offers the potential for improved sample efficiency in learning high-performing policies, yet naïve integrations often suffer from two fundamental issues: (1) *negative transfer*, where optimizing the IL loss hinders effective RL fine-tuning, and (2) *gradient conflict*, where differences in the scale or direction of IL and RL gradients lead to unstable updates. We introduce *Conflict-Averse IL-RL* (CAIR), a general framework that addresses both challenges by combining two key components: (1) *Loss Manipulation*: an adaptive annealing mechanism utilizing a convex combination of IL and RL losses. This mechanism dynamically increases the weight of the RL loss when its gradient aligns with the IL gradient and decreases it otherwise, mitigating instabilities during the transition from IL to RL. (2) *Gradient Manipulation*: to further reduce conflict, we incorporate CAGrad to compute a joint gradient that balances IL and RL objectives while avoiding detrimental interference. We show that under standard trust-region assumptions, CAIR guarantees monotonic improvement in the expected return when the loss weights are annealed monotonically. In order to demonstrate its effectiveness, we evaluate CAIR on two benchmark suites: (1) sparse-reward robotic manipulation benchmark (Fetch and Adroit), which require demonstration data for meaningful learning, and (2) dense-reward continuous control locomotion benchmark (D4RL), where demonstrations serve to accelerate standard RL. Compared against relevant hybrid IL–RL baselines, CAIR demonstrates competitive performance across both sparse- and dense-reward tasks. While CAIR does not achieve state-of-the-art results on every individual task, it yields the highest aggregate normalized performance—outperforming baselines by a large margin in sparse-reward settings and remaining competitive in dense-reward environments. Results are presented across multiple combinations of IL (BC, DAgger) and RL (DDPG, SAC, PPO) algorithms, suggesting that CAIR provides a general mechanism for combining imitation and reinforcement learning across diverse continuous-control settings.

1 Introduction

Recent advancements in automation demonstrate a shift from traditional rule-based controllers to adaptive, AI-driven controllers in applications such as data center cooling (Heimerson et al., 2022), traffic management (Ault & Sharon, 2021; Ault et al., 2020), chatbots (Li et al., 2016), and self-driving cars (Kiran et al., 2021). These systems are often modeled as Markov Decision Processes (MDPs), enabling the use of Reinforcement Learning (RL) to optimize control policies. RL aims to learn the optimal policy but often requires substantial training data, which can be prohibitively expensive in many applications. By contrast, Imitation Learning (IL) offers a sample-efficient alternative where existing sub-optimal controllers or human demonstrations are leveraged to quickly train a competent controller. However, the competency of a controller trained via IL is often limited by the quality of the demonstrations used to train it.

In recent years, researchers have proposed combined IL and RL frameworks aiming to achieve the best of both worlds: sample efficiency and optimal asymptotic performance. Such approaches differ in how they

incorporate demonstrator feedback—e.g., using demonstration-based rewards (Kang et al., 2018; Bajaj et al., 2023), replay experience relabeling (Nair et al., 2018; Zhu et al., 2022a), or goal-based feedback (Nair et al., 2018)—and in the type of demonstrations utilized, such as static datasets or interactive feedback (Ziebart et al., 2008; Warnell et al., 2018; Christiano et al., 2017). Despite recent progress in integrating IL and RL, two significant challenges remain: (1) *Negative Transfer*—where naïve sequencing of IL followed by RL may force the RL policy to relearn or even discard imitation guidance, degrading the RL agent's ability to converge to the optimal policy compared to learning from scratch (Taylor & Stone, 2009; Zhang et al., 2023); and (2) *Gradient Conflict*—when IL and RL objectives are optimized simultaneously, their gradients may conflict in scale or direction (negative cosine similarity), leading to unstable or suboptimal updates. Gradient conflicts are particularly severe when IL relies on suboptimal demonstrations, as updates may push the policy away from the optimal RL solution. Such misalignment can result in learning instabilities and might contribute to negative transfer (Taylor & Stone, 2009).

Traditional annealing-based approaches mitigate negative transfer by considering a convex loss combination over the IL and RL objectives. They typically decay the weight of the IL component over fixed schedules (Rengarajan et al., 2022; Goecks et al., 2019; Zhu et al., 2022a). However, such heuristics require careful tuning and might still produce adverse policy updates, especially when the IL loss gradients conflict with, or are overwhelmed by, the RL loss gradients. To address this limitation, we introduce *Conflict-Averse IL-RL (CAIR)*. CAIR introduces a principled *adaptive loss annealing* framework that optimizes an *alignment objective* rather than relying on predefined annealing schedules. Specifically, CAIR defines the weights of the convex loss combination via a constrained optimization formulation: it sets the weights such that the gradient of the combined loss (denoted the *combined gradient*) maximizes alignment with the RL gradient, subject to the constraint that the cosine similarity between the combined gradient and the IL gradient remains positive.

While this principled adaptive loss annealing reduces the occurrence of conflicting gradients, it is not guaranteed to eliminate this issue altogether (e.g., when the optimal combination still yields a negative alignment). To further mitigate gradient conflicts in such cases, we incorporate Conflict-Averse Gradient Descent (CAGrad) (Liu et al., 2021), which explicitly resolves gradient disagreements between IL and RL objectives. By combining the proposed adaptive loss annealing with conflict-aware gradient manipulation, CAIR can successfully eliminate conflicting gradients, as demonstrated empirically in Section 3.3. This ensures that progress toward the RL objective occurs only when it remains compatible with IL guidance, mitigating negative transfer and enabling a stable transition from imitation-driven training to reward-driven optimization.

We present this framework in detail and show that it is theoretically grounded, offering monotonic improvement guarantees under trust-region-constrained optimization (Schulman et al., 2017; 2015b). Empirical evaluations on sparse-reward MuJoCo environments (Todorov et al., 2012) demonstrate the effectiveness of CAIR across diverse RL algorithms, including off-policy methods such as SAC (Haarnoja et al., 2018) and DDPG (Lillicrap et al., 2015), as well as the on-policy method PPO (Schulman et al., 2017). Relative to the evaluated hybrid IL–RL baselines, CAIR consistently achieves competitive performance on the sparse-reward manipulation tasks, attaining the strongest overall results in a majority of the evaluated tasks while learning policies that match or exceed demonstrator performance across all five tasks. To evaluate whether these benefits extend beyond sparse-reward settings, we further study six D4RL (Fu et al., 2020) locomotion tasks. The results demonstrate that CAIR remains competitive in dense-reward locomotion settings, achieving the strongest overall performance on multiple tasks while remaining comparable to established hybrid IL–RL baselines on others. Overall, the results show that CAIR provides competitive performance relative to hybrid IL–RL baselines while remaining applicable across multiple combinations of IL and RL algorithms.

2 Preliminaries

Problem Formulation. We consider control problems modeled as a Markov Decision Process (MDP) defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and γ is the discount factor. At each timestep t, the agent executes an action a_t in state s_t, leading to a new state s_{t+1} and a reward $r_t = R(s_t, a_t)$. The tuple

(s_t, a_t, r_t, s_{t+1}) is denoted as a *transition*. A finite sequence of successive transitions starting from an initial state s_0 and leading to a terminal state s_T forms an *episode*, denoted by $\tau = (s_0, a_0, r_0, s_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$.

Reinforcement Learning (RL). Given an MDP, the Reinforcement Learning (RL) objective is to learn a policy π that maximizes the expected return. A policy is defined either as a mapping from states to actions $\pi : \mathcal{S} \to \mathcal{A}$ (deterministic policy) or as a mapping from states to a probability simplex over the action space $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ (stochastic policy). The expected return is defined as $J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$. The optimal policy is defined by $\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi)$. We consider the policy to be a function approximator (e.g., a neural network) with tunable parameters θ, denoted by π_θ. In this work, we focus on *sparse-reward* control problems, where the reward function provides informative feedback only in a small subset of states, making it challenging for standard RL methods to learn effective policies.

Imitation Learning (IL). Imitation Learning (IL) aims to learn control policies directly from a demonstrator, bypassing the need for a reward function. IL assumes an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma)$ *without* a reward function, together with either a fixed set of offline demonstrations or, in interactive IL (Ross et al., 2011), access to a demonstrator policy.

Offline Demonstration. A dataset $\mathcal{D} = \{\tau_i\}$ of trajectories generated by a demonstrator is available offline.

Demonstrator Query. An oracle π^d is available for querying demonstrator actions given a state.

Note that querying a demonstrator does not necessitate that the demonstrator function is known, only that it can be queried. In this sense, a human can be considered an interactive demonstrator. Moreover, the underlying demonstrator policy may be stochastic; the assumption simply implies that an action sample is provided rather than a full action distribution. Finally, the demonstrator—in either the offline or online case—is not assumed to act optimally. Indeed, in this work, we focus on sub-optimal demonstrations that necessitate RL fine-tuning beyond the IL objective.

The objective in IL is to learn a policy that matches the demonstrator's behavior. IL is commonly formulated as minimizing a divergence between trajectories generated by the learner and those in the demonstration dataset:

$$L_{\mathrm{IL}}(\theta) = \mathbb{E}_{\tau_\pi\sim\pi,\, \tau_i\sim\mathcal{D}}\big[\mathrm{Div}(\tau_\pi, \tau_i)\big],$$

where Div measures trajectory- or state-action-level mismatch. A widely used special case is *Behavioral Cloning* (BC) (Bain & Sammut, 1995; Torabi et al., 2018), which reduces IL to supervised learning. Here the divergence is defined at the one-step action level as:

$$\mathrm{Div}_{\mathrm{BC}}(\tau_\pi, \tau_i) = -\sum_{(s,a^d)\in\tau_i} \log \pi_\theta(a^d \mid s),$$

which corresponds to the negative log-likelihood of demonstrated actions under the learner policy. IL methods are typically more *sample efficient* than RL because they rely solely on demonstrations without requiring environment exploration. However, their performance is ultimately bounded by the quality and coverage of the available demonstrations.

Combined IL–RL Loss. Access to both the reward function R and demonstrations $\mathcal{D}$ (or an interactive demonstrator π^d) enables the integration of imitation and reinforcement learning objectives. IL can guide early learning in sparse-reward settings, while RL drives the policy toward maximizing expected return.

A weighted combination of IL and RL losses provides a common and flexible way to unify these objectives. Such formulations have been used with on-policy trust-region RL algorithms (Rengarajan et al., 2022; Kang et al., 2018), and off-policy actor-critic algorithms (Hester et al., 2018; Nair et al., 2020; Zhu et al., 2022a). As the weighted formulation makes no assumptions about the internal form of L_{IL} or L_{RL}, it offers a general mechanism for transitioning between the two. Formally, we define the combined objective as:

$$L_\lambda(\theta) = (1-\lambda)\, L_{\mathrm{RL}}(\theta) + \lambda\, L_{\mathrm{IL}}(\theta), \qquad \lambda \in [0, 1]. \tag{1}$$

Gradient Conflict and Negative Transfer. To analyze how the combined objective in equation 1 behaves during optimization, we examine the relationships between the gradients of the IL, RL, and combined losses, denoted by $g_{\mathrm{IL}} = \lambda \nabla_\theta L_{\mathrm{IL}}(\theta)$, $g_{\mathrm{RL}} = (1-\lambda)\nabla_\theta L_{\mathrm{RL}}(\theta)$, and:

$$g_\lambda = \nabla_\theta L_\lambda(\theta) = g_{\mathrm{RL}} + g_{\mathrm{IL}}.$$

We measure *alignment* using the inner product or its normalized form (cosine similarity), where $\langle g_i, g_j \rangle > 0$ indicates aligned updates, and $\langle g_i, g_j \rangle < 0$ indicates conflicting updates.

Gradient Conflict arises when the IL and RL gradients disagree, i.e., $\langle g_{\mathrm{IL}}, g_{\mathrm{RL}} \rangle < 0$, indicating that the two objectives recommend incompatible parameter updates. In such cases, their convex combination g_λ can also become negatively aligned with one of the objectives, leading to unstable or ineffective updates.

Negative Transfer is classically defined as a degradation in learning caused by using additional data or objectives, relative to learning without them (Taylor & Stone, 2009). In IL–RL training, this corresponds to a drop in performance when transitioning from IL-dominated to RL-dominated optimization as λ decreases. However, such performance drops are typically difficult to predict *a priori* and are often only visible after many updates (Taylor & Stone, 2009; Taylor, 2009; Wang et al., 2021). Consequently, the literature lacks a general-purpose mechanism for detecting and avoiding negative transfer at the level of individual optimization updates, particularly when multiple objectives are jointly optimized.

In this work, we focus on gradient conflicts as a plausible cause for negative transfer. Specifically, in the IL–RL setting, we flag an update as *susceptible to negative transfer* when the combined gradient moves against the IL gradient, i.e., $\langle g_{\mathrm{IL}}, g_\lambda \rangle < 0$. When this occurs, the resulting update direction may move the policy away from demonstrator-guided behavior. Since the demonstrations are precisely what provide a strong initialization and help overcome sparse-reward exploration difficulties, such anti-aligned updates may cause drops in performance.

This characterization is specific to the IL–RL combined-objective setting and is not intended as a universal definition of negative transfer. Rather, it targets one important mechanism by which negative transfer can arise—destructive interference between IL and RL update directions—and, crucially, provides a *pre-update* condition that we can enforce by manipulating L_λ and its gradients. This is in line with findings from multi-task learning, where gradient cosine similarity is widely used to quantify task interference (Liu et al., 2021; Jiang et al., 2023), even though it does not explain all forms of negative transfer (Jiang et al., 2023).

2.1 Related Work

Combining IL and RL. Prior work has explored the integration of IL and RL objectives through linear combinations. For instance, *LOGO* (Rengarajan et al., 2022), *POfD* (Kang et al., 2018), and *Reward Phasing* (Bajaj et al., 2023) leverage demonstration data with trust-region-based guarantees but are often limited to on-policy gradient algorithms that use constrained optimization (Schulman et al., 2017; 2015b). Alternatively, off-policy training methods define implicit curricula through shifted sample distributions (Hester et al., 2018; Nair et al., 2018; Zhu et al., 2022a), though they typically lack formal convergence guarantees and rely on heuristically tuned schedules. In contrast, we propose CAIR, a principled framework that supports both on-policy and off-policy RL algorithms combined with either offline or interactive IL methods. When a trust-region solver is employed, CAIR additionally provides monotonic improvement guarantees along the IL-to-RL transition.

Offline and Hybrid RL. Offline RL learns reward-driven policies from fixed datasets without additional environment interaction (Levine et al., 2020). Although some offline RL methods incorporate auxiliary imitation losses (Hester et al., 2018; Nair et al., 2018; 2020), these are typically used as regularizers rather than as part of an explicit IL–RL optimization framework. A common focus of offline RL is mitigating distributional shift and out-of-distribution action selection, through conservative value-learning approaches such as CQL (Kumar et al., 2020) and IQL (Kostrikov et al., 2021). Hybrid RL methods combine offline datasets with online interaction to improve the sample efficiency of the RL agent (Song et al., 2022; Ball et al., 2023). Similar to offline RL, these methods tend to focus on stabilizing value learning under distribution shift. Neither offline nor hybrid RL explicitly accounts for gradient conflicts between imitation (demonstration

driven) and reinforcement (reward driven) objectives—a challenge our method, CAIR, is designed to address directly.

Negative Transfer. Negative transfer broadly refers to situations in which incorporating additional data or objectives degrades learning performance relative to training without them (Taylor & Stone, 2009). In IL–RL settings, it commonly arises during the transition from imitation-dominated to reward-dominated optimization, where imitation-driven policy updates reduce the sample efficiency or asymptotic performance of the RL agent (Goecks et al., 2019; Rengarajan et al., 2022). Some prior approaches mitigate this effect using fixed or hand-designed schedules for annealing the IL update weight (Rengarajan et al., 2022; Goecks et al., 2019; Zhu et al., 2022a). Such approaches are commonly agnostic to IL–RL gradient interactions and can still produce adverse update directions. Recent work addresses negative transfer by detecting when auxiliary signals harm the main objective: Du et al. (2018) adapt auxiliary-loss weights using gradient cosine similarity to reduce interference with a fixed primary task, though their method does not model a staged IL–to–RL transition and may downweight imitation losses prematurely in sparse-reward settings. Wang et al. (2019) mitigate negative transfer in transfer learning by filtering irrelevant source data. ForkMerge (Jiang et al., 2023) proposes a validation-driven branching strategy to manage task interference.

Gradient Manipulation in Multi-Task Learning. Conflicting gradients have been previously addressed in the context of Multi-Task Learning (MTL) through gradient manipulation methods (Désidéri, 2012; Yu et al., 2020; Sener & Koltun, 2018; Liu et al., 2021; 2023), which resolve gradient conflicts through joint objective optimization. Notably, CAGrad (Liu et al., 2021) provably converges to the optimum of the average loss, unlike MGDA (Sener & Koltun, 2018) and PCGrad (Yu et al., 2020), which may converge to arbitrary Pareto stationary points. Our proposed approach, CAIR, builds on this perspective by framing IL–RL integration as a dynamic MTL problem, justifying the use of CAGrad to resolve gradient conflicts between IL and RL objectives. This formalization enables mitigation of gradient conflicts through per-update adaptive objective weighting and gradient manipulation.

While we target gradient conflict as a primary contributing factor to negative transfer in IL–RL training, we acknowledge that other factors may also contribute (Jiang et al., 2023).

3 Conflict-Averse IL to RL

In this section, we formally present our framework, *Conflict-Averse IL-RL* (CAIR). CAIR addresses negative transfer when optimizing a convex combination of IL and RL by actively resolving gradient conflicts. We achieve this through two complementary mechanisms: (1) **Adaptive Loss Annealing**, which dynamically reweights the IL and RL losses to maximize RL contributions only when they align with the IL objective; and (2) **Conflict-Averse Gradient Descent (CAGrad)**, which projects the update direction to balance both gradients even when their magnitudes differ. Together, these mechanisms enable the transition from IL to RL to avoid regressions caused by gradient misalignment.

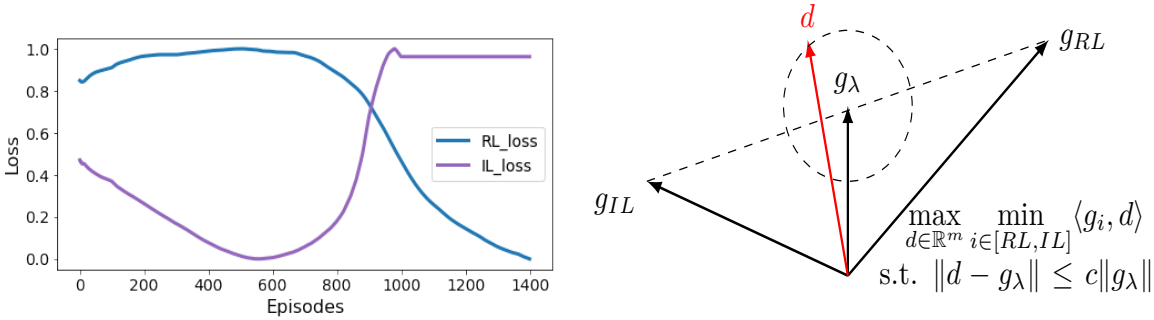
Furthermore, under standard assumptions, CAIR provides monotonic improvement guarantees in expected return when used with trust-region-based optimization.

3.1 Adaptive Loss Annealing

Algorithms that combine IL and RL typically prioritize the IL objective in early training to quickly imitate the demonstrator, before shifting focus to RL to optimize expected returns $J(\pi)$ (Kober et al., 2010; Rajeswaran et al., 2017). However, this sequential transition is prone to *negative transfer*, where the policy experiences a drop in performance during the shift from IL-dominated to RL-dominated training.

As shown in Figure 1(a), optimizing a convex combination of IL and RL losses reveals a fundamental conflict on the *FetchPush-v1* task (see Appendix A.1 for experimental details). We observe that reducing the IL loss often increases the RL loss and vice versa. This mismatch indicates that naïve IL-to-RL transitions can drive updates away from imitation-induced behavior, leading to temporary performance degradation.

In contrast to prior work that anneals the IL weight λ over fixed heuristics (Rengarajan et al., 2022; Goecks et al., 2019; Zhu et al., 2022a), we propose a principled mechanism that *optimizes an alignment objective* to



(a) Loss Manipulation

(b) Gradient Manipulation

Figure 1: **(a)** Evolution of Soft Actor-Critic (RL) and Behavioral Cloning (IL) losses on the *FetchPush-v1* task using standard linear annealing ($\lambda : 1 \to 0$) without gradient manipulation. As the IL weight decreases, the RL loss decreases while the IL loss spikes, suggesting the objectives are conflicting. (x-axis: Episodes $\times 10^3$; y-axis: Normalized loss). **(b)** The optimization landscape for the CAGrad method. g_λ represents the naive combined gradient, while d is the optimized update direction found by projecting g_λ to maximize its worst-case inner-product between g_{IL} and g_{RL}, while satisfying the constraints (depicted by the acceptance region within the dashed circle).

select λ dynamically. Specifically, we choose λ by solving:

$$\max_{\lambda\in[0,1]} \cos(\tilde{g}_\lambda, \tilde{g}_{\text{RL}}) \quad \text{s.t.} \quad \cos(\tilde{g}_\lambda, \tilde{g}_{\text{IL}}) \geq 0. \tag{2}$$

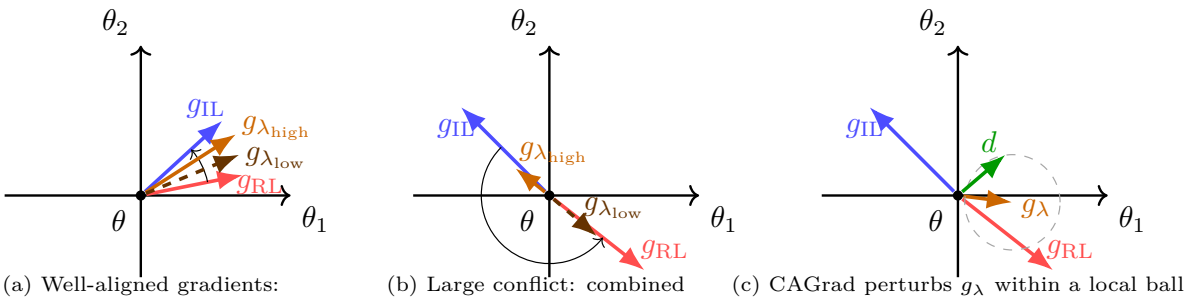
Here, $\tilde{g}_{\text{IL}}$ and $\tilde{g}_{\text{RL}}$ denote the L_2-normalized IL and RL gradients, and $\tilde{g}_\lambda$ is the normalized convex combination used for alignment checks. We employ L_2-normalization to ensure that λ is optimized based solely on directional alignment, preventing bias from differing gradient magnitudes (e.g., if the RL loss scale is significantly larger than the IL loss scale).

The constraint in Equation 2 ensures that updates remain positively aligned with the IL objective, preventing regressions where the combined update would oppose the demonstration. Simultaneously, the maximization objective favors progress toward the RL goal. This enables a stable transition, allowing λ to decrease only when the IL and RL gradients are sufficiently aligned. However, if the gradients do not sufficiently align, the constraint prevents further annealing, maintaining a nonzero λ and preserving the influence of the imitation objective.

By framing loss annealing as an optimization problem rather than a fixed schedule, CAIR dynamically balances IL and RL contributions based on the local geometry of the loss landscape. This is particularly important in settings where RL gradients may be noisy or weak, as it avoids prematurely discounting imitation signals that are still essential for effective learning. The λ optimization procedure is summarized in Algorithm 2 (Appendix A.4).

The effect of λ-annealing can be understood geometrically through the gradient-alignment illustration in Figure 2. Panel (a) shows the case where IL and RL gradients are well aligned: the angle between g_{IL} and g_{RL} is small, and the combined gradients $g_{\lambda_{\text{high}}}$ (solid orange) and $g_{\lambda_{\text{low}}}$ (dashed brown) remain nearly identical. In this case, decreasing λ produces consistent update directions, and the transition from IL to RL can proceed aggressively without destabilizing learning.

Conversely, Panel (b) represents the situation where gradients strongly conflict. Here, the direction of g_λ becomes highly sensitive to the choice of λ: a small decay can rotate the update direction dramatically toward the RL gradient, causing abrupt policy shifts. This motivates our strategy of updating λ only intermittently and annealing it conservatively whenever the gradients are misaligned.



(a) Well-aligned gradients: annealing can be more aggressive

(b) Large conflict: combined direction is highly sensitive to λ

(c) CAGrad perturbs g_λ within a local ball to improve alignment w.r.t. both objectives

Figure 2: Geometric interpretation of loss annealing and gradient manipulation. Each panel shows a 2-D policy-parameter space (θ_1, θ_2) with IL gradient g_{IL} (blue), RL gradient g_{RL} (red), and combined gradient g_λ. $g_{\lambda_{high}}$ (orange) uses a larger λ than $g_{\lambda_{low}}$ (dashed-brown). **(a)** When gradients are aligned, annealing λ (orange to dashed brown) maintains a consistent direction. **(b)** When gradients conflict, small changes in λ cause drastic shifts in update direction, risking negative transfer. **(c)** CAGrad calculates a corrected direction d (green) within a trust region around g_λ to maximize joint agreement.

3.2 Gradient Manipulation

While our adaptive loss annealing guarantees that the combined gradient g_λ remains positively aligned with the IL gradient whenever a feasible solution exists, it does not strictly eliminate conflicts between the underlying IL and RL components. When g_{IL} and g_{RL} oppose each other, the resulting g_λ may effectively cancel out or be dominated by a single objective, preventing meaningful policy updates. This issue is often overlooked in previous approaches that jointly optimize IL and RL objectives via simple linear scalarization (Nair et al., 2018; Rajeswaran et al., 2017; Rengarajan et al., 2022; Zhu et al., 2022a).

To address this, we adapt *Conflict-Averse Gradient Descent* (CAGrad) (Liu et al., 2021). Originally developed to resolve gradient conflicts in Multi-Task Learning (MTL), we repurpose CAGrad for the hybrid IL-to-RL setting to treat the IL and RL objectives as competing components within the combined loss $L_\lambda(\theta)$ (defined in Equation 1). CAGrad seeks an update direction d that maximizes the worst-case improvement across both objectives, subject to a constraint that keeps d close to the original combined gradient g_λ. Formally, the optimization is defined as:

$$\max_{d \in \mathbb{R}^m} \min_{i \in \{RL, IL\}} \langle g_i, d \rangle \quad \text{s.t.} \quad \|d - g_\lambda\| \leq c\|g_\lambda\|. \tag{3}$$

Here, $c \in [0, 1)$ is a hyper-parameter controlling the size of the trust region (or allowable deviation). The vector d represents the optimal update direction within a local ball centered at g_λ that maximizes the projection onto both g_{RL} and g_{IL}.

CAGrad explicitly optimizes the minimum improvement across objectives at each step. It provably converges to a stationary point of the average loss while ensuring each update satisfies the locality constraint and corresponds to a Pareto-stationary direction (Liu et al., 2021). The complete CAIR framework, integrating both adaptive loss annealing and CAGrad, is summarized in Algorithm 1 (Appendix A.4).

The geometric intuition is illustrated in Figure 2(c). When g_{IL} and g_{RL} point in nearly opposite directions, the naïve combination g_λ (orange arrow) may itself be poorly aligned with both objectives. CAGrad searches within a bounded region (dashed circle) around g_λ to find a direction d (green arrow) that maximizes the minimum agreement with the underlying gradients. The resulting update is balanced, avoiding the scenario where one objective dominates or cancels the other.

3.3 Empirical Gradient Analysis

A central challenge in IL-to-RL transfer lies in how the *combined gradient* g_λ evolves as the weighting parameter λ decreases. To understand this, it is useful to conceptually decompose the IL-to-RL transition into three stages: (1) an initial stage where the IL signal provides meaningful guidance but the RL signal is

not yet informative, e.g., due to sparse rewards and insufficient exploration; (2) an intermediate stage where both signals are meaningful and, assuming the demonstrations are task-relevant, are mostly aligned; and (3) a final stage where the demonstrations become counterproductive due to their suboptimality, and the RL signal dominates. During stages (1) and (2), we are particularly interested in avoiding situations where g_λ becomes *negatively aligned* with the imitation gradient g_{IL}, i.e., $\langle g_{\text{IL}}, g_\lambda \rangle < 0$, as this indicates that the combined update moves the policy *against* the direction favored by the demonstrator, potentially unlearning useful imitation-induced behavior.

During stage (3), however, enforcing alignment with g_{IL} is neither necessary nor desirable, as the RL objective should be free to surpass the demonstrator. Recall from Section 2 that $g_{\text{IL}} = \lambda \nabla_\theta L_{\text{IL}}(\theta)$, so as $\lambda \to 0$, g_{IL} approaches the zero vector. Since the cosine similarity constraint $\langle g_{\text{IL}}, g_\lambda \rangle \geq 0$ is trivially satisfied for any g_λ when $g_{\text{IL}} = \mathbf{0}$, the alignment constraint becomes inactive precisely when it should — as the transition to pure RL optimization is complete. This means CAIR naturally transitions to standard RL optimization in stage (3) without requiring any explicit mechanism to disable the alignment constraint.

Following this intuition, we empirically investigate the impact of CAIR and its two components—adaptive loss annealing and gradient manipulation—on gradient alignment, focusing on g_{IL}, g_{RL}, the combined gradient g_λ, and the final update direction d.

Results are presented for a single run on the *FetchPick&Place-v1* task. The *FetchPickPlace-v1* task requires a robotic arm to grasp an object and move it to a target location. While demonstrations follow a particular trajectory distribution, multiple trajectories can achieve the same goal state. RL optimization may therefore favor alternative solutions, which can lead to disagreements between the imitation and RL gradients during stage (3) of the IL–to–RL transition.

The presented results demonstrate gradient alignment for *Behavior Cloning* and *DDPG* as the IL and RL algorithms respectively. Similar general trends (with respect to gradients alignment) were observed for other IL–RL algorithms combinations (these combinations are listed in Section 3.5). For clarity, each row in Figure 3 contains two plots for the same setting: the left column shows cosine similarity with the imitation gradient, $\cos(g_{\text{IL}}, \cdot)$, and the right column shows cosine similarity with the RL gradient, $\cos(g_{\text{RL}}, \cdot)$. This layout allows direct comparison of alignment with both the IL and RL objectives during annealing.

Figure 3(a) illustrates the conflict effect under *fixed-schedule loss annealing*. Let Δ_λ denote the annealing step used to update $\lambda \leftarrow \lambda - \Delta_\lambda$. The blue curve corresponds to a slower schedule ($\Delta_\lambda = 0.05$), while the purple curve corresponds to a faster schedule ($\Delta_\lambda = 0.2$). In the fast-annealing case, the cosine similarity between the IL gradient and the combined gradient drops below zero, indicating that the update direction actively opposes the imitation objective. While the slower schedule (blue) exhibits less abrupt fluctuations, it remains a heuristic approach that does not explicitly account for the alignment between g_{IL}, g_{RL}, and g_λ.

We next examine whether gradient manipulation alone (using CAGrad) is sufficient to improve alignment between the IL and combined gradients. Figure 3(b) compares fixed-step annealing ($\Delta_\lambda = 0.2$) with and without CAGrad using a moderate constraint parameter ($c = 0.75$). In this setting, CAGrad (purple) successfully adjusts the update direction so that it remains positively aligned with the IL gradient, mitigating gradient conflict despite aggressive annealing. However, CAGrad's effectiveness depends on the allowable deviation from g_λ. Figure 3(c) shows the same comparison with a tighter constraint ($c = 0.2$). Here, the adjusted direction cannot deviate sufficiently from the combined gradient to resolve the conflict, and negative alignment with g_{IL} re-emerges. This demonstrates that CAGrad is sensitive to hyperparameter tuning and is not robust to large conflicts induced by rapid or poorly timed changes in λ.

Finally, Figure 3(d) illustrates the full CAIR approach by comparing adaptive loss annealing without CAGrad (blue) to adaptive loss annealing combined with CAGrad (purple), both with $c = 0.2$. Adaptive annealing alone improves alignment relative to fixed schedules but can still produce negatively aligned updates due to discrete update intervals or temporarily infeasible alignment constraints. In contrast, CAIR—by jointly adapting λ and refining the update direction with CAGrad—maintains positive alignment with both the IL and RL gradients throughout training. These results highlight the complementary roles of the two components: adaptive loss annealing shapes the objective to avoid inducing severe conflicts, while CAGrad

resolves residual gradient disagreements at the update level. Together, they enable CAIR to consistently prevent gradient-induced negative transfer during the IL-to-RL transition.

3.4 Monotonic Improvement Guarantee

Setting. We consider a finite sequence of scalarized objectives parameterized by

$$1 = \lambda_0 > \lambda_1 > \cdots > \lambda_N = 0,$$

where each λ_k defines our λ-combined loss L_{λ_k} (Equation 1). The sequence begins at $\lambda_0 = 1$ (pure IL) and terminates at $\lambda_N = 0$ (pure RL), representing the full IL-to-RL transition.

For each λ_k, CAGrad is applied to optimize $L_{\lambda_k}(\theta)$. Under Assumption 1 below, CAGrad guarantees convergence to a first-order stationary point of $L_{\lambda_k}(\theta)$, satisfying $\min_{t \le T} \mathbb{E}\|\nabla L_{\lambda_k}(\theta_t)\|^2 \to 0$ as $T \to \infty$, where t indexes CAGrad gradient update steps (Theorem 3.2 in (Liu et al., 2021)). The resulting stationary policy corresponds to a Pareto-stationary solution of the IL–RL bi-objective. We denote the resulting policy π_k^*. Upon convergence to π_k^*, the annealing weight is updated to λ_{k+1} and the optimization of $L_{\lambda_{k+1}}(\theta_{k+1})$ is initialized at the current policy π_k^*, ensuring continuity across successive stages.

The following assumptions ensure monotonic non-decreasing expected return between consecutive stationary policies, i.e., between π_k^* and π_{k+1}^* along the annealing schedule.

Assumption 1 (CAGrad Regularity)**.** *The objectives $L_{\mathrm{IL}}(\theta)$ and $L_{\mathrm{RL}}(\theta)$ are differentiable and H-smooth, i.e.,*

$$\|\nabla L_i(\theta) - \nabla L_i(\theta')\| \le H\|\theta - \theta'\|, \qquad \forall\, \theta, \theta', \quad i \in \{\mathrm{IL}, \mathrm{RL}\},$$

and $L_{\lambda_k}(\theta)$ is bounded below for every λ_k.

Assumption 2 (Stagewise KL Proximity)**.** *For each consecutive pair π_k^* and π_{k+1}^*, the expected state-conditional KL divergence is bounded:*

$$\mathbb{E}_{s\sim\rho_{\pi_k^*}}\big[D_{\mathrm{KL}}\big(\pi_k^*(\cdot \mid s) \,\big\|\, \pi_{k+1}^*(\cdot \mid s)\big)\big] \;\le\; \delta,$$

for some $\delta > 0$.

Assumption 3 (Minimum Surrogate Improvement)**.** *For each consecutive pair π_k^* and π_{k+1}^*, the advantage-based surrogate improvement under π_{k+1}^* is uniformly lower bounded:*

$$\mathbb{E}_{s\sim\rho_{\pi_k^*}}\Big[\mathbb{E}_{a\sim\pi_{k+1}^*(\cdot|s)} A_{\pi_k^*}(s,a)\Big] \;\ge\; \eta,$$

for some $\eta > 0$ that holds uniformly across all $k = 0, \ldots, N-1$.

Lemma 1 (Stagewise Surrogate–KL Dominance)**.** *Let*

$$C_k \;:=\; \frac{2\gamma}{(1-\gamma)^2}\,\epsilon_k, \qquad \epsilon_k \;=\; \sup_{s,a} |A_{\pi_k^*}(s,a)|,$$

denote the per-stage constant appearing in the TRPO performance lower bound (Theorem 1 in Schulman et al. (2015a)), and let $C^* = \max_k C_k$. Under Assumptions 2–3, if $\eta \ge C^*\delta$, then for every $k = 0, \ldots, N-1$,

$$\mathbb{E}_{s\sim\rho_{\pi_k^*}}\Big[\mathbb{E}_{a\sim\pi_{k+1}^*(\cdot|s)} A_{\pi_k^*}(s,a)\Big] \;\ge\; C_k\, \mathbb{E}_{s\sim\rho_{\pi_k^*}}\big[D_{\mathrm{KL}}\big(\pi_k^*(\cdot \mid s) \,\big\|\, \pi_{k+1}^*(\cdot \mid s)\big)\big]\,.$$

Proof. By Assumption 3, the left-hand side is at least η for every k. By Assumption 2, the right-hand side satisfies

$$C_k\, \mathbb{E}_{s\sim\rho_{\pi_k^*}}\big[D_{\mathrm{KL}}(\pi_k^* \| \pi_{k+1}^*)\big] \;\le\; C_k\,\delta \;\le\; C^*\delta \;\le\; \eta,$$

which yields the claim. □

Theorem 1 (Monotonic Non-Deterioration)**.** *Under Assumptions 1–3 and the condition $\eta \ge C^*\delta$, the expected return is non-decreasing along the λ-schedule:*

$$J(\pi_{k+1}^*) \;\ge\; J(\pi_k^*), \qquad \forall\, k = 0, \ldots, N-1.$$

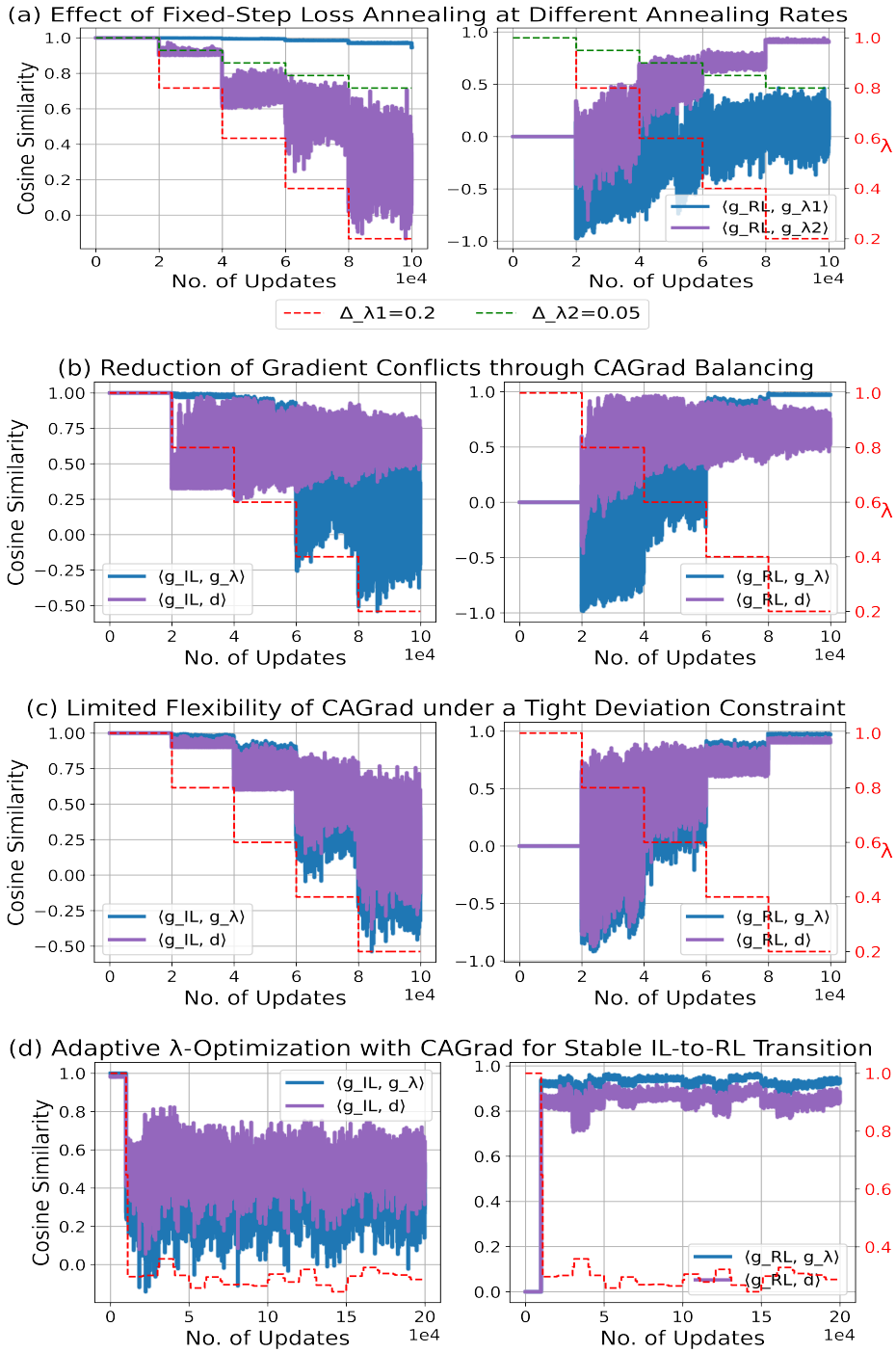


Figure 3: Cosine similarity analysis on *FetchPick&Place-v1* during the IL-to-RL transition. Each of the six plots reports the alignment between the combined update direction and the task-specific (IL, RL) gradients: cosine similarity with g_{IL} (left column) and with g_{RL} (right column). **(a)** Naïve fixed annealing leads to negative alignment (conflict). **(b)** CAGrad with a loose constraint resolves conflict but requires tuning. **(c)** CAGrad with a tight constraint fails to resolve conflict under fixed annealing. **(d)** CAIR (Adaptive Annealing + CAGrad) maintains positive alignment robustly, even with tight constraints. (x-axis: gradient updates; y-axis: cosine similarity averaged over 10 updates).

Proof. By Assumption 1 and Theorem 3.2 of (Liu et al., 2021), CAGrad converges to a stationary point of $L_{\lambda_k}(\theta)$, yielding a well-defined π_k^*. The TRPO performance lower bound (Theorem 1 in Schulman et al. (2015a)) gives

$$J(\pi_{k+1}^*) \;\geq\; J(\pi_k^*) + \mathbb{E}_{s\sim\rho_{\pi_k^*},\, a\sim\pi_{k+1}^*(\cdot|s)}\Big[A_{\pi_k^*}(s,a)\Big] - C_k\, \mathbb{E}_{s\sim\rho_{\pi_k^*}}\big[D_{\mathrm{KL}}\big(\pi_k^*(\cdot \mid s) \,\|\, \pi_{k+1}^*(\cdot \mid s)\big)\big]\,.$$

By Lemma 1, the surrogate term dominates the KL penalty, so the right-hand side is at least $J(\pi_k^*)$, proving $J(\pi_{k+1}^*) \geq J(\pi_k^*)$. □

Discussion. Assumption 2 (KL Proximity) is naturally satisfied in CAIR, where optimization of $L_{\lambda_{k+1}}$ is initialized at π_k^* (see Algorithm 1, Line 13). Since the two objectives L_{λ_k} and $L_{\lambda_{k+1}}$ differ only in their scalarization weight, the resulting stationary point π_{k+1}^* is expected to remain close to π_k^*, with the bound δ becoming tighter as the step size $\lambda_k - \lambda_{k+1}$ decreases. Assumption 3 (Minimum Surrogate Improvement) requires that the advantage of π_{k+1}^* under the state distribution of π_k^* is uniformly positive across all stages. This is reasonable since π_{k+1}^* optimizes an objective with a larger RL weight than π_k^*, steering the policy locally toward higher expected return. The linking condition $\eta \geq C^*\delta$ ties Assumptions 2 and 3 together: it is satisfiable by choosing a sufficiently small δ, i.e., a gradual enough λ-schedule, since C^* is a fixed constant determined by the advantage function bounds. Finally, Theorem 1 ties these ingredients together: Assumption 1 guarantees that CAGrad converges to a well-defined stationary policy π_k^* at each stage—without which Assumptions 2 and 3 would be vacuous—while the latter two assumptions, grounded in the TRPO performance bound, ensure that the resulting sequence of policies is non-deteriorating in expected return.

Theorem 1 does not claim that CAIR itself enforces these bounds; rather, it shows that if the sequence of policies produced by the CAIR annealing procedure satisfies the stated stagewise conditions, then the resulting policy sequence exhibits monotonic improvement. Although this theorem strictly applies under trust-region assumptions, we show empirically in Section 4 that CAIR yields stable, monotonic learning curves even when paired with off-policy algorithms such as SAC (Haarnoja et al., 2018) and DDPG (Lillicrap et al., 2015) that do not formally enforce these conditions.

3.5 Generality of the Framework

Different pairings of IL and RL algorithms may behave differently in practice depending on the learning assumptions of the underlying methods. For example, off-policy RL algorithms such as SAC or DDPG may work well with offline IL methods such as Behavioral Cloning (Bain & Sammut, 1995; Torabi et al., 2018), whereas on-policy RL algorithms like PPO may align more naturally with interactive IL algorithms that collect data under the current policy, such as DAgger (Ross et al., 2011).

These considerations motivate examining multiple IL and RL combinations in our experiments. However, it is important to note that our primary goal is not to identify the single "best" pairing, but rather to demonstrate that CAIR provides a principled, algorithm-agnostic mechanism for managing the transition between IL and RL objectives, regardless of the specific solvers employed.

4 Experimental Study

Our experiments are designed to evaluate the performance of CAIR when paired with various combinations of IL and RL solvers. Specifically, we aim to answer the following research questions:

1. **Robustness:** Is CAIR robust to different combinations of IL and RL algorithms?
2. **Performance:** Can CAIR achieve improved sample efficiency and asymptotic performance compared to baseline hybrid-RL approaches?
3. **Ablation-Impact of CAgrad:** What is the relative impact of incorporating Conflict-Averse Gradient Descent (CAGrad) alongside adaptive loss annealing?
4. **Ablation-Impact of Adaptive Annealing:** What is the relative impact of incorporating adaptive loss annealing instead of fixed-schedule loss annealing?

4.1 Settings

Benchmark Tasks. We conduct experiments across two broad classes of MuJoCo benchmark tasks (Todorov et al., 2012), categorized by their reward structure: sparse reward and dense reward.

Sparse-Reward tasks. This class presents significant exploration challenges due to sparse reward feedback and includes two sets of tasks:

- **Fetch Tasks** (Plappert et al., 2018): This set of tasks evaluates robotic arm manipulation. We use three tasks: *FetchPickAndPlace-v1* (P&P), *FetchSlide-v1* (FS), and *FetchPush-v1* (FP). These tasks require the agent to manipulate an object toward a desired goal location.
- **Adroit Tasks** (Rajeswaran et al., 2017): This set of tasks evaluates dexterous manipulation involving high-dimensional control of a simulated anthropomorphic hand. We use two tasks: *AdroitHandHammerSparse-v1* and *AdroitHandPenSparse-v1*, which feature highly complex dynamics.

Dense-Reward Tasks. Unlike the sparse manipulation tasks, this class uses dense reward functions to evaluate continuous control under high-dimensional dynamics. While not required in order to perform meaningful learning, demonstrations can facilitate better sample efficiency in such tasks.

- **D4RL Locomotion Tasks**: We evaluate continuous locomotion control using six tasks from the locomotion benchmark: *HalfCheetah*, *Hopper*, *Walker2d*, *Ant*, *Swimmer*, and *Humanoid*.

Episode lengths are capped at 1,000 steps for tasks in the Fetch and D4RL benchmark suites, and 4,000 steps for tasks in the Adroit benchmark suite.

Demonstrations. Each task is associated with an offline dataset $\mathcal{D}$ of demonstration trajectories and an interactive demonstrator policy π^d available for querying during training.

Fetch tasks. We use the suboptimal rule-based demonstrator introduced by Bajaj et al. (2023). This heuristic controller selects actions based on the relative Cartesian distances between the robot gripper, the manipulated object, and the target position. Although competent, the resulting π^d does not produce optimal trajectories. The offline dataset $\mathcal{D}$ is constructed by executing π^d in the environment and recording 50 resulting trajectories.

Adroit tasks. For each manipulation task, the demonstrations and the corresponding interactive demonstrator policy π^d are obtained from the benchmark introduced by Rajeswaran et al. (2017). We use 50 demonstration trajectories per task.

Locomotion tasks. For the D4RL locomotion tasks, we use the `simple` Minari datasets (Younis et al., 2024), which provide offline demonstration trajectories generated by trained RL policies. Following the dataset generation procedure described in the Minari documentation, we reproduce the corresponding demonstrator policies and use them as the interactive demonstrator π^d during training.

Evaluation Protocol. All manipulation experiments are repeated with 5 random seeds and all locomotion experiments with 10 random seeds. We report the mean reward across runs with a 1-σ shaded error region.

Implementation. We implement CAIR by extending the `stable-baselines3` library (Raffin et al., 2021). To support reproducibility, our codebase and execution instructions are publicly available.[1] Detailed hyperparameters and hardware specifications are provided in Appendix A.2.

4.2 Generality of CAIR

Our first set of experiments addresses Research Question 1: *Is CAIR robust to different combinations of IL and RL algorithms?*

To demonstrate generality, we evaluate CAIR using multiple combinations of Imitation and Reinforcement Learning algorithms. We pair two common IL approaches: (1) **Behavioral Cloning (BC)** (Bain & Sammut,

[1] `https://osf.io/hukge/?view_only=8f303806d77e4faf83371c36f8c5b4b2`

Table 1: Learning performance of CAIR instantiated with different combinations of IL algorithms (BC, DAgger) and RL algorithms (SAC, DDPG, PPO) on sparse-reward manipulation tasks. Values report final reward (mean ± standard deviation), area under the curve (AUC), and normalized reward (NormReward), computed as $(R - R_{\text{min}})/(R_{\text{demo}} - R_{\text{min}})$, where R_{min} is the minimum score observed across all evaluated methods; values above 1 indicate performance exceeding the demonstrator. Training budgets are shown beneath each task. Methods marked with † are significantly worse than the highest-mean method in the same task (Welch's t-test, $p < 0.05$). Among the remaining methods, AUC determines the highest-ranked method, which is bolded. The final rows report average NormReward across tasks.

Task	Pairing	Mean	± SD	AUC	NormReward
	BC+DDPG	-13.18	± 4.26	-18.03	2.45
FetchPickPlace	BC+SAC	-19.52	± 4.88	-29.78	2.03
(10,000k steps)	BC+PPO	-12.61	± 1.90	-20.20	2.49
	DAgger+DDPG	-12.48	± 3.33	-17.26	2.50
	DAgger+SAC	**-12.88**	**± 4.75**	**-14.13**	**2.47**
	DAgger+PPO	-14.03	± 3.00	-16.44	2.40
	BC+DDPG	-14.35	± 3.49	-19.23	1.82
FetchPush	BC+SAC	-10.92	± 2.55	-22.41	1.98
(10,000k steps)	BC+PPO	-26.95	± 6.02	-28.74	1.04
	DAgger+DDPG	-12.05	± 2.18	-17.18	1.90
	DAgger+SAC	**-9.80**	**± 1.23**	**-15.29**	**2.02**
	DAgger+PPO	†-19.10	± 2.98	-22.02	1.40
	BC+DDPG	-38.74	± 3.38	-38.78	2.32
FetchSlide	**BC+SAC**	**-35.55**	**± 4.83**	**-37.15**	**3.05**
(10,000k steps)	BC+PPO	†-43.45	± 2.43	-44.03	1.17
	DAgger+DDPG	-37.84	± 3.69	-40.38	2.30
	DAgger+SAC	-36.77	± 2.72	-41.64	2.37
	DAgger+PPO	-38.24	± 2.35	-40.97	2.22
	BC+DDPG	1,354	± 287	718	29.87
AdroitHammer	**BC+SAC**	**1,426**	**± 356**	**1,024**	**31.43**
(3,000k steps)	BC+PPO	†6	± 0	-1	0.58
	DAgger+DDPG	609	± 666	277	13.66
	DAgger+SAC	650	± 751	338	14.57
	DAgger+PPO	809	± 412	471	18.02
	BC+DDPG	†2,593	± 411	1,500	0.74
AdroitPen	**BC+SAC**	**7,041**	**± 744**	**6,419**	**2.01**
(8,000k steps)	BC+PPO	†48	± 68	193	0.01
	DAgger+DDPG	†3,009	± 353	2,349	0.86
	DAgger+SAC	†3,306	± 389	2,063	0.94
	DAgger+PPO	†3,808	± 188	3,794	1.09
	BC+DDPG				7.44
Average Normalized Reward	**BC+SAC**				**8.10**
Across Tasks	BC+PPO				1.06
	DAgger+DDPG				4.25
	DAgger+SAC				4.48
	DAgger+PPO				5.03

1995; Torabi et al., 2018), which learns from offline demonstrations; and (2) **DAgger** (Ross et al., 2011), which learns from an interactive demonstrator; with three distinct RL algorithms covering different learning paradigms: (1) **SAC** (Haarnoja et al., 2018) (stochastic off-policy); (2) **DDPG** (Lillicrap et al., 2015) (deterministic off-policy); and (3) **PPO** (Schulman et al., 2017) (stochastic on-policy).

Implementation Details. For both BC and DAgger, we use the Mean Squared Error (MSE) loss to minimize the divergence between the learner's action $a \sim \pi_\theta(\cdot|s)$ and the demonstrator's action $a_D = \pi_D(s)$:

$$L_{IL} = \frac{1}{N} \sum_{i=1}^{N} \|\pi_D(s_i) - a_i\|^2.$$

Table 2: Learning performance of CAIR instantiated with different combinations of IL algorithms (BC, DAgger) and RL algorithms (SAC, DDPG, PPO) on D4RL locomotion tasks. Values report final reward (mean ± standard deviation), area under the curve (AUC), and normalized reward (NormReward), computed as $(R - R_{\text{min}})/(R_{\text{demo}} - R_{\text{min}})$, where R_{min} is the minimum score observed across all evaluated methods; values above 1 indicate performance exceeding the demonstrator. Methods marked with † are significantly worse than the highest-mean method in the same task (Welch's t-test, $p < 0.05$). Among the remaining methods, AUC determines the highest-ranked method, which is bolded. The final rows report average NormReward across tasks.

Task	Pairing	Mean	± SD	AUC	NormReward
HalfCheetah (250k steps)	BC+DDPG	†8,462	± 661	4,909	0.85
	BC+SAC	†7,113	± 332	3,677	0.73
	BC+PPO	†1,197	± 863	626	0.17
	DAgger+DDPG	**9,584**	**± 1,119**	**6,649**	**0.96**
	DAgger+SAC	†8,504	± 348	3,929	0.86
	DAgger+PPO	†2,963	± 617	1,890	0.33
Hopper (250k steps)	BC+DDPG	†3,140	± 256	2,682	0.91
	BC+SAC	3,424	± 119	2,864	1.03
	BC+PPO	†1,883	± 562	1,951	0.38
	DAgger+DDPG	**3,328**	**± 210**	**2,944**	**0.99**
	DAgger+SAC	3,363	± 49	2,876	**1.01**
	DAgger+PPO	†3,102	± 306	3,021	0.90
Walker2d (250k steps)	BC+DDPG	†1,200	± 743	1,038	0.32
	BC+SAC	3,814	± 325	2,534	0.93
	BC+PPO	†2,161	± 489	1,870	0.55
	DAgger+DDPG	3,369	± 1,044	3,072	0.83
	DAgger+SAC	**4,171**	**± 971**	**3,278**	1.02
	DAgger+PPO	†1,550	± 875	1,613	0.41
Swimmer (250k steps)	BC+DDPG	†15.6	± 17.2	58.3	0.05
	BC+SAC	†256.8	± 22.2	153.6	0.78
	BC+PPO	†138.6	± 54.6	92.6	0.42
	DAgger+DDPG	**306.1**	**± 68.5**	**289.3**	**0.93**
	DAgger+SAC	318.0	± 12.6	284.8	0.96
	DAgger+PPO	331.5	± 4.4	275.1	1.00
Ant (250k steps)	BC+DDPG	†1,085	± 1,402	97	0.64
	BC+SAC	3,013	± 1,252	1,465	1.04
	BC+PPO	†1,270	± 442	1,233	0.66
	DAgger+DDPG	†1,202	± 668	1,634	0.67
	DAgger+SAC	2,275	± 1,066	1,881	0.87
	DAgger+PPO	**1,976**	**± 863**	**1,920**	**0.69**
Humanoid (250k steps)	BC+DDPG	†4,917	± 219	4,810	0.69
	BC+SAC	5,024	± 171	4,934	0.79
	BC+PPO	†4,803	± 81	4,685	0.58
	DAgger+DDPG	**5,346**	**± 97**	**5,248**	**1.09**
	DAgger+SAC	5,416	± 266	5,026	1.15
	DAgger+PPO	†4,957	± 84	4,881	0.73
Average Normalized Reward Across Tasks	BC+DDPG				0.58
	BC+SAC				0.88
	BC+PPO				0.46
	DAgger+DDPG				0.91
	DAgger+SAC				**0.98**
	DAgger+PPO				0.68

The annealing weight λ is updated every $T_\lambda = 10$ episodes in the D4RL robotic manipulation tasks and every $T_\lambda = 5$ episodes in the locomotion tasks. [2] (see Line 22 of Algorithm 1 in Appendix A.4). Annealing is

[2]While we found $T_\lambda = 10$ and $T_\lambda = 5$ to work well in the reported tasks, we acknowledge that this value might require tuning in other tasks.

considered complete when λ falls below a small threshold $\epsilon_\lambda < 0.02$; rather than checking for exact convergence to zero, we use a small positive threshold to account for approximation noise (see Line 11 of Algorithm 2 in Appendix A.4). Once this condition is met, λ is fixed at zero and remains unchanged for the rest of training, ensuring that demonstrator guidance is fully withdrawn and the agent continues learning solely from environmental feedback.

Results.

Manipulation Tasks. Table 1 summarizes the final performance and area under the learning curve (AUC) achieved by different combinations of IL algorithms (BC, DAgger) and RL algorithms (SAC, DDPG, PPO) on the sparse-reward manipulation tasks. The corresponding learning curves are provided in Figure 5 in Appendix A.4. Across the evaluated pairings, all IL–RL combinations exceed demonstrator performance in 3/5 tasks. Among the evaluated pairings, BC+SAC achieves the strongest overall performance on *FetchSlide-v1*, *AdroitHandHammerSparse-v1*, and *AdroitHandPenSparse-v1*, while DAgger+SAC achieves the strongest overall performance on *FetchPickAndPlace-v1* and *FetchPush-v1*. Consistent with these task-level results, BC+SAC achieves the highest average normalized reward across tasks (8.16). BC+PPO is the weakest pairing, failing to match demonstrator performance on *AdroitHandHammerSparse-v1* and *AdroitHandPenSparse-v1* (Normalized Reward < 1.0), while BC+DDPG and DAgger+DDPG each fall short of the demonstrator in 1/5 tasks. Overall, SAC-based variants achieve the strongest performance across the manipulation tasks.

Locomotion Tasks. Table 2 summarizes similar measurements on the D4RL locomotion tasks. The corresponding learning curves are provided in Figure 6 in Appendix A.4. Across all six tasks, at least one IL–RL pairing matches or exceeds demonstrator performance (Normalized Reward $>\simeq 1.0$). However, no single IL–RL combination consistently achieves this across all six tasks. DAgger+DDPG achieves the highest-ranked performance on *HalfCheetah-v5*, *Hopper-v5*, *Swimmer-v5*, and *Humanoid-v5*, while DAgger+SAC achieves the highest-ranked performance on *Walker2d-v5*. DAgger+PPO achieves the highest-ranked performance on *Ant-v5*. Consistent with these task-level results, DAgger+SAC achieves the highest average normalized reward across tasks (0.98), followed by DAgger+DDPG (0.91) and BC+SAC (0.88). Overall, DAgger-based variants achieve the strongest performance across the locomotion tasks.

Taken together, the manipulation and locomotion results provide a qualified positive answer to Research Question 1. CAIR can be instantiated with multiple IL–RL pairings across both sparse-reward manipulation and dense-reward locomotion tasks. However, performance differences between pairings remain significant. Across both benchmark suites, SAC-based variants achieve the strongest performance in the largest number of tasks. More specifically, BC-based variants achieve the strongest performance in the majority of the manipulation tasks, while DAgger-based variants achieve the strongest performance in the majority of the locomotion tasks.

Sensitivity Analysis. The previous experiments evaluate CAIR across different IL–RL pairings. We now examine the sensitivity of the proposed adaptive annealing and conflict-aware optimization mechanisms to hyperparameter selection. CAIR exposes three primary hyperparameters: (i) the CAGrad deviation parameter c; (ii) the update interval for the adaptive λ-optimization; and (iii) the learning rate η_λ used for updating λ.

- **CAGrad Constraint (c):** Across tasks, we observe that moderate values (e.g., $c \in [0.45, 0.95]$) produce consistent behavior with only minor variations in final performance. However, when c is set too small, the adjusted gradient is forced to remain close to the original combined gradient g_λ, preventing CAGrad from effectively resolving IL–RL conflicts. As observed in our ablation studies (Figure 9 in Appendix A.3), this can lead to severe performance degradation due to unmitigated gradient interference.
- **Update Interval:** To provide a stable learning signal, we update λ every 10 episodes for the manipulation tasks and every 5 episodes for the locomotion tasks. This allows the policy to train under a fixed weight before recomputing the alignment balance. Updating too frequently makes the weighting overly sensitive to short-term gradient noise, whereas updating too infrequently produces a sluggish transition toward RL. The impact of this interval is detailed in Figures 12 and 13 (Appendix A.3).

- **Annealing Rate (η_λ):** The learning rate of the λ-optimizer exhibits a similar trade-off. Very small values stall the annealing process, while larger values produce abrupt shifts driven by noisy gradient estimates. Consequently, we default to $\eta_\lambda = 10^{-4}$. Figure 11 compares learning curves for $\eta_\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}\}$, showing that both extremes reduce sample efficiency. This aligns with prior findings (Uchendu et al., 2022) that excessively rapid transitions from IL to RL can cause performance collapse, further motivating the need for controlled, alignment-aware annealing.

While several IL–RL pairings achieve demonstrator-level performance in the locomotion tasks, reaching demonstrator performance is only one aspect of successfully combining imitation learning and reinforcement learning. A policy trained through imitation can reproduce demonstrator behavior, but hybrid IL–RL methods must additionally maintain performance while gradually shifting optimization from the imitation objective to the RL objective. As shown in the adaptive annealing ablation studies (Section 4.6), this transition is non-trivial: policies can attain demonstrator-level performance and subsequently experience a decline in return as the influence of the imitation objective is reduced.

Note on On-Policy Stability. For on-policy methods like PPO, the transition is notably more brittle. We find that reducing the policy learning rate and entropy coefficient as $\lambda \to 0$ leads to more stable post-annealing performance. Furthermore, normalizing IL and RL gradients and maintaining a weighted moving average of their cosine similarity helps counteract high-variance gradient estimates, supporting a smoother transition into RL-dominated training.

4.3 Benchmark Comparison

Our next set of experiments addresses Research Question 2: *Can CAIR achieve improved sample efficiency and asymptotic performance compared to baseline hybrid-RL approaches?*

To answer this, we compare CAIR against the following common baseline algorithms:

Reward Phasing (RP). The *Reward Phasing* algorithm (Bajaj et al., 2023) is a recent IL-to-RL transfer method that explicitly phases out the imitation signal which is designed as an IRL Arora & Doshi (2021) approximated reward function. It has demonstrated state-of-the-art performance in the standard Fetch tasks (Plappert et al., 2018).

Hindsight Experience Replay (HER) with Demonstrations. Hindsight Experience Replay (Andrychowicz et al., 2017) addresses sparse-reward learning by relabeling failed trajectories as successes with respect to the achieved goal. We compare against the extension proposed by Nair et al. (Nair et al., 2018), which explicitly incorporates demonstrations by seeding the replay buffer with demonstration data and maintaining a fixed ratio of demonstration transitions during mini-batch sampling. This guides exploration toward relevant regions of the state space. Since *HER* relies on goal relabeling, we evaluate this baseline only on the goal-conditioned manipulation tasks and exclude it from the locomotion experiments.

Self-Adaptive Imitation Learning (SAIL). SAIL (Zhu et al., 2022b) is an off-policy algorithm that dynamically expands the teacher's demonstration buffer with high-quality, self-generated trajectories. The approach effectively optimizes a convex combination of Behavioral Cloning and Q-learning objectives, where the weight is adapted based on the estimated quality of the generated data.

JumpStart Reinforcement Learning (JSRL). JSRL (Uchendu et al., 2022) is a curriculum-based meta-algorithm that initializes the RL policy using a pre-trained guide (the demonstrator). It progressively increases the difficulty of the task by initializing the agent at states further from the goal (using the demonstrator to reach those states), thereby creating a curriculum of starting distributions that gradually transitions from the demonstrator's capabilities to the RL agent's control.

Reinforcement Learning with Prior Data (RLPD). RLPD (Ball et al., 2023) is a hybrid method designed for sample efficiency. It combines offline demonstrations with online interactions using a high Update-to-Data (UTD) ratio, layer normalization, and an ensemble of critics. Unlike other methods that rely on explicit imitation losses, RLPD biases the learning through symmetric sampling of demonstration data and aggressive gradient updates.

Table 3: Learning performance of CAIR (BC+DDPG and BC+SAC) compared to hybrid IL–RL baselines on sparse-reward manipulation tasks. Values report final reward (mean ± standard deviation), AUC, and normalized reward (NormReward), computed as $(R - R_{\text{min}})/(R_{\text{demo}} - R_{\text{min}})$, where R_{min} is the minimum score observed across all evaluated methods; values above 1 indicate performance exceeding the demonstrator. Training budgets are shown beneath each task. Methods marked with † are significantly worse than the highest-mean method in the same task (Welch's t-test, $p < 0.05$). Among the remaining methods, AUC determines the highest-ranked method, which is bolded. The final rows report average NormReward across tasks.

Task	Algorithm	Mean	± SD	AUC	NormReward
	BC+DDPG	**-14.07**	**± 5.50**	**-16.52**	**2.40**
FetchPickPlace	BC+SAC	-14.88	± 3.88	-26.04	2.34
(15,000k steps)	RP	-12.75	± 1.63	-32.91	2.48
	HER	†-27.14	± 2.63	-26.80	1.52
	SAIL	†-45.14	± 0.13	-46.09	0.32
	JumpStart	-14.40	± 3.18	-27.68	2.37
	RLPD	†-48.16	± 2.51	-48.00	0.12
	BC+DDPG	**-13.57**	**± 2.2**	**-18.12**	**1.82**
FetchPush	BC+SAC	-10.39	± 1.8	-20.64	1.98
(10,000k steps)	RP	†-46.90	± 2.0	-46.85	0.15
	HER	†-22.62	± 2.1	-23.79	1.37
	SAIL	†-46.21	± 0.9	-46.20	0.19
	JumpStart	-23.53	± 14.2	-28.99	1.32
	RLPD	-12.78	± 5.1	-20.20	1.86
	BC+DDPG	†-38.84	± 2.86	-38.75	2.23
FetchSlide	**BC+SAC**	**-33.02**	**± 3.34**	**-36.29**	**3.40**
(18,500k steps)	RP	†-39.10	± 2.12	-44.34	2.18
	HER	†-41.70	± 2.38	-42.43	1.66
	SAIL	†-47.44	± 0.03	-47.72	0.51
	JumpStart	†-44.65	± 1.80	-45.32	1.07
	RLPD	†-42.80	± 9.79	-43.04	0.91
	BC+DDPG	1,354	± 287	718	29.87
AdroitHammer	BC+SAC	1,426	± 356	1,024	31.43
(3,000k steps)	RP	†-19	± 1	-19	0.00
	HER+Demos	†14	± 37	13	0.75
	SAIL	†10	± 30	12	0.65
	JumpStart	†-15	± 7	-12	0.05
	RLPD	**1,358**	**± 596**	**1,212**	**29.96**
	BC+DDPG	†2,593	± 411	1,500	0.74
Pen	BC+SAC	7,041	± 744	6,419	2.01
(8,000k steps)	RP	†527	± 0	285	0.15
	HER+Demos	†177	± 108	133	0.05
	SAIL	†37	± 56	69	0.01
	JumpStart	†3,636	± 2,104	4,372	1.04
	RLPD	**7,565**	**± 474**	**6,654**	**2.16**
	DAgger+DDPG				7.41
Average Normalized Reward	**BC+SAC**				**8.23**
Across Tasks	RP				0.99
	HER+Demos				1.07
	SAIL				0.34
	JumpStart				1.17
	RLPD				7.00

Further details on these methods and their specific hyperparameters are provided in Appendix A.5 and Appendix A.2, respectively.

Table 4: Learning performance of CAIR (BC+SAC and DAgger+DDPG) compared to hybrid IL–RL baselines on D4RL locomotion tasks. Values report final reward (mean ± standard deviation), AUC, and normalized reward (NormReward), computed as $(R - R_{\text{min}})/(R_{\text{demo}} - R_{\text{min}})$, where R_{min} is the minimum score observed across all evaluated methods; values above 1 indicate performance exceeding the demonstrator. Methods marked with † are significantly worse than the highest-mean method in the same task (Welch's t-test, $p < 0.05$). Among the remaining methods, AUC determines the highest-ranked method, which is bolded. The final rows report average NormReward across tasks.

Task	Algorithm	Mean	± SD	AUC	NormReward
HalfCheetah (250k steps)	**DAgger+DDPG**	**9,584**	**± 1,119**	**6,649**	**0.96**
	BC+SAC	†7,113	± 332	3,677	0.73
	RP	†848	± 112	439	0.13
	SAIL	†-557	± 91	-537	0.00
	JumpStart	†3,502	± 1,575	1,914	0.38
	RLPD	†1,037	± 662	935	0.15
Hopper (250k steps)	DAgger+DDPG	†3,328	± 210	2,944	0.99
	BC+SAC	†3,424	± 119	2,864	1.03
	RP	†2,704	± 146	2,344	0.73
	SAIL	†1,083	± 306	1,425	0.05
	JumpStart	†976	± 38	978	0.00
	RLPD	**3,705**	**± 141**	**3,288**	**1.15**
Walker2d (250k steps)	DAgger+DDPG	†3,369	± 1,044	3,072	0.83
	BC+SAC	†3,814	± 325	2,534	0.93
	RP	†1,609	± 420	1,395	0.42
	SAIL	†-187	± 1,002	266	0.00
	JumpStart	†827	± 14	826	0.24
	RLPD	**5,232**	**± 213**	**4,241**	**1.26**
Swimmer (250k steps)	**DAgger+DDPG**	**306.1**	**± 68.5**	**289.3**	**0.93**
	BC+SAC	256.8	± 22.2	153.6	0.78
	RP	†0.1	± 0.2	1.2	0.01
	SAIL	†-0.2	± 1.1	-0.2	0.00
	JumpStart	†5.8	± 20.2	4.2	0.02
	RLPD	†2.9	± 1.1	3.1	0.01
Ant (250k steps)	DAgger+DDPG	†1,275	± 672	1,550	0.67
	BC+SAC	**3,666**	**± 902**	**1,853**	**1.04**
	RP	†-301	± 21	-471	0.42
	SAIL	†-3,001	± 16	-2,743	0.00
	JumpStart	†710	± 454	643	0.58
	RLPD	†654	± 15	471	0.57
Humanoid (250k steps)	DAgger+DDPG	†5,346	± 97	5,248	1.09
	BC+SAC	†5,024	± 171	4,934	0.79
	RP	†4,177	± 100	4,134	0.00
	SAIL	†4,392	± 392	4,368	0.30
	JumpStart	†4,927	± 10	4,927	0.70
	RLPD	**6,386**	**± 258**	**5,765**	**2.06**
Average Normalized Reward Across Tasks	**DAgger+DDPG**				**0.91**
	BC+SAC				0.88
	RP				0.28
	SAIL				0.04
	JumpStart				0.32
	RLPD				0.87

4.4 Sample Efficiency and Asymptotic Performance Results

To evaluate the effectiveness of CAIR, we compare representative CAIR variants against the baselines. For the manipulation tasks, we report BC+DDPG and BC+SAC, which achieve the strongest overall performance among the evaluated manipulation pairings. For the locomotion tasks, we report BC+SAC and

DAgger+DDPG. Although DAgger+SAC outperforms BC+SAC on overall performance in several locomotion benchmarks, BC+SAC is included to enable a more direct comparison with prior hybrid IL–RL baselines, which primarily rely on offline demonstrations rather than on online interactive demonstrator.

Manipulation Tasks. Table 3 summarizes the final performance and area under the learning curve (AUC) achieved by CAIR and the evaluated baselines on the manipulation tasks. The corresponding learning curves are provided in Figure 7 in Appendix A.4. Relative to the evaluated baselines, CAIR achieves the highest-ranked performance in three of the five manipulation tasks, namely *FetchPickPlace-v1*, *FetchPush-v1*, and *FetchSlide-v1*. On *AdroitHandHammerSparse-v1*, BC+SAC achieves performance comparable to RLPD, while RLPD attains slightly higher final performance and AUC. On *AdroitHandPenSparse-v1*, BC+SAC achieves performance comparable to RLPD, with both methods substantially outperforming the remaining baselines. Consistent with these task-level results, BC+SAC achieves the highest average normalized reward across tasks (8.23), exceeding both RLPD (7.00) and the remaining baselines. Across all five manipulation tasks, CAIR consistently learns policies that match or exceed demonstrator performance.

Locomotion Tasks. Table 4 summarizes the final performance and area under the learning curve (AUC) achieved by CAIR and the evaluated baselines on the D4RL locomotion tasks. The corresponding learning curves are provided in Figure 8 in Appendix A.4. Relative to the evaluated baselines, CAIR achieves the highest-ranked performance in three of the six locomotion tasks (*HalfCheetah-v5*, *Swimmer-v5*, and *Ant-v5*). On *Hopper-v5*, BC+SAC achieves performance comparable to RLPD, although RLPD achieves the stronger overall result. RLPD also achieves the strongest overall performance on *Walker2d-v5* and *Humanoid-v5*. Despite not achieving the highest-ranked result in every task, DAgger+DDPG attains the highest average normalized reward across tasks (0.91), marginally exceeding both BC+SAC (0.88) and RLPD (0.87).

4.5 Ablation Study: Impact of CAGrad

To address Research Question 3, we investigate the contribution of Conflict-Averse Gradient Descent (CAGrad) beyond adaptive loss annealing.

For both the manipulation and locomotion tasks, we compare DAgger+DDPG and DAgger+SAC against variants where the CAGrad component is disabled, relying only on adaptive loss annealing.

Manipulation Tasks. The manipulation results, presented in Figure 9 (Appendix A.3), show that incorporating CAGrad improves sample efficiency and asymptotic performance relative to adaptive annealing alone, yielding an average performance improvement of approximately 20% across the Fetch tasks. The benefits are particularly pronounced in the high-dimensional *AdroitHandHammerSparse-v1* and *AdroitHandPenSparse-v1* tasks, where removing CAGrad substantially reduces performance in certain IL–RL combinations.

Locomotion Tasks. Figure 10 presents the corresponding analysis on the D4RL locomotion tasks. Relative to adaptive annealing alone, incorporating CAGrad improves sample efficiency, allowing higher returns to be attained earlier during training. CAGrad also improves asymptotic performance across all evaluated locomotion experiments, with a median improvement of approximately 25%.

Overall Assessment. Overall, the ablation results indicate that incorporating CAGrad improves sample efficiency and asymptotic performance relative to adaptive annealing alone across both manipulation and locomotion tasks.

4.6 Ablation Study: Impact of Adaptive Annealing

To study the effect of the proposed adaptive annealing strategy, we compare two variants of CAIR: one with our proposed adaptive annealing and one with fixed-step schedules in which the mixture weight is updated with a constant step size, as $\lambda \leftarrow \lambda - \Delta$, for constant step sizes $\Delta \in 0.01, 0.05, 0.1, 0.15$ across both the manipulation and locomotion tasks.

Manipulation Tasks. Figure 14 shows that the performance of fixed annealing schedules is sensitive to the choice of Δ, with the best-performing value varying across tasks. In *FetchPickAndPlace-v1* and *FetchSlide-v1*, $\Delta = 0.05$ yields the highest asymptotic performance, whereas *FetchPush-v1* favors $\Delta = 0.01$. Larger step

sizes degrade performance: in *FetchPickAndPlace-v1* and *FetchSlide-v1*, asymptotic performance deteriorates substantially at $\Delta \geq 0.1$, while in *FetchPush-v1*, learning remains stable at $\Delta = 0.1$.

These results indicate that fixed annealing requires task-specific tuning of Δ. In contrast, CAIR's adaptive annealing achieves asymptotic performance comparable to well-tuned fixed schedules without requiring manual schedule selection.

Locomotion Tasks. Figure 15 presents the corresponding analysis on the D4RL locomotion tasks. Smaller values of Δ maintain demonstrator-level performance for longer, whereas larger values transition more rapidly toward RL-only optimization and often exhibit a substantial reduction in performance.

Across all fixed schedules, a noticeable decline in performance is observed near the completion of annealing ($\lambda \to 0$), after which the policy must recover through additional RL training. Adaptive annealing produces a smoother transition from imitation learning to reinforcement learning, reducing the magnitude of this decline and maintaining performance as the weighting shifts from the IL objective toward the RL objective.

Overall Assessment. Overall, these results indicate that performance is sensitive to the choice of annealing schedule. Fixed schedules require task-specific tuning of Δ and can exhibit substantial performance degradation as the influence of the imitation objective is reduced. Notably, the locomotion experiments show that reaching demonstrator-level performance does not guarantee a successful transition from imitation learning to reinforcement learning, as several fixed schedules attain demonstrator-level performance before experiencing a decline in performance. In contrast, adaptive annealing adjusts the IL–RL weighting based on gradient alignment, reducing these performance declines without requiring manual schedule selection.

5 Conclusions

We introduced *Conflict-Averse IL-RL* (CAIR), a general framework that integrates Imitation Learning (IL) and Reinforcement Learning (RL) by addressing the fundamental challenge of negative transfer stemming from conflicts between the IL and RL gradients. CAIR dynamically adapts the weighting of the IL and RL objectives using gradient alignment, increasing the influence of the RL objective only when its gradient remains compatible with the IL gradient. Furthermore, by incorporating Conflict-Averse Gradient Descent (CAGrad), CAIR actively projects update directions to maintain alignment with the IL and RL gradients. Together, these mechanisms help mitigate conflicts between the IL and RL gradients, enabling effective transitions from imitation-driven initialization to reward-driven optimization.

The method is theoretically grounded, offering convergence guarantees under trust-region constraints, while demonstrating applicability across diverse RL paradigms, including PPO, SAC, and DDPG. Across five sparse-reward and six dense-reward robotic manipulation tasks, the best CAIR variant achieves higher normalized performance compared to baseline methods. These results demonstrate that CAIR remains effective across both sparse-reward manipulation and dense-reward locomotion settings.

Our ablation studies further highlight the contribution of both components of CAIR. Incorporating CAGrad improves sample efficiency and asymptotic performance relative to variants trained without gradient manipulation, while adaptive annealing provides a principled mechanism for transitioning from imitation learning to reinforcement learning without requiring manual schedule selection.

A practical limitation of CAIR is its sensitivity to the compatibility of the underlying IL and RL algorithms. As observed in our results, performance can vary substantially across different IL–RL pairings, and no single pairing consistently achieves the strongest performance across all tasks. Future work will explore methods for characterizing and improving compatibility across IL–RL pairings, as well as extending CAIR to multi-modal demonstration data.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural*

Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

James Ault and Guni Sharon. Reinforcement learning benchmarks for traffic signal control. In *NeurIPS Datasets and Benchmarks*, 2021.

James Ault, Josiah P. Hanna, and Guni Sharon. Learning an interpretable traffic signal control policy. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, pp. 88–96, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, 1995. URL https://api.semanticscholar.org/CorpusID:10738655.

Vaibhav Bajaj, Guni Sharon, and Peter Stone. Task phasing: Automated curriculum learning from demonstrations. *Proceedings of the International Conference on Automated Planning and Scheduling*, 33(1): 542–550, Jul. 2023. doi: 10.1609/icaps.v33i1.27235. URL https://ojs.aaai.org/index.php/ICAPS/article/view/27235.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5):313–318, 2012. ISSN 1631-073X. doi: https://doi.org/10.1016/j.crma.2012.03.014. URL https://www.sciencedirect.com/science/article/pii/S1631073X12000738.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Vinicius G Goecks, Gregory M Gremillion, Vernon J Lawhern, John Valasek, and Nicholas R Waytowich. Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments. *arXiv preprint arXiv:1910.04281*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Albin Heimerson, Johannes Sjölund, Rickard Brännvall, Jonas Gustafsson, and Johan Eker. Adaptive control of data center cooling using deep reinforcement learning. In *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pp. 1–6, 2022. doi: 10.1109/ACSOSC56246.2022.00018.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

Junguang Jiang, Baixu Chen, Junwei Pan, Ximei Wang, Liu Dapeng, Jie Jiang, and Mingsheng Long. Forkmerge: Mitigating negative transfer in auxiliary-task learning. In *Neural Information Processing Systems*, 2023. URL https://api.semanticscholar.org/CorpusID:258865647.

Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2469–2478. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/kang18a.html.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

Jens Kober, Betty Mohler, and Jan Peters. Imitation and reinforcement learning for motor primitives with perceptual coupling. In *From motor learning to interaction learning in robots*, pp. 209–225. Springer, 2010.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *ArXiv*, abs/2110.06169, 2021. URL https://api.semanticscholar.org/CorpusID:238634325.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020. URL https://api.semanticscholar.org/CorpusID:218486979.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *arXiv preprint arXiv:2306.03792*, 2023.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299. IEEE, 2018.

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

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Joshua Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *ArXiv*, abs/1802.09464, 2018. URL https://api.semanticscholar.org/CorpusID:3619030.

Antonin Raffin. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Desik Rengarajan, Gargi Vaidya, Akshay Sarvesh, Dileep Kalathil, and Srinivas Shakkottai. Reinforcement learning with sparse rewards using guidance from offline demonstration. *arXiv preprint arXiv:2202.04628*, 2022.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, Lille, France, 07–09 Jul 2015b. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://dblp.uni-trier.de/db/journals/corr/corr1707.html#SchulmanWDRK17.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Yuda Song, Yi Zhou, Ayush Sekhari, J. Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid rl: Using both offline and online data can make rl efficient. *ArXiv*, abs/2210.06718, 2022. URL https://api.semanticscholar.org/CorpusID:252873449.

Matthew E Taylor. Assisting transfer-enabled machine learning algorithms: Leveraging human knowledge for curriculum design. In *AAAI Spring Symposium: Agents that Learn from Human Teachers*, pp. 141–143, 2009.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

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.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. pp. 4950–4957, 2018.

Ike Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matt Bennice, Chuyuan Kelly Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning. In *NeurIPS 2021 Robot Learning Workshop, RSS 2022 Scaling Robot Learning Workshop*, 2022.

Liyuan Wang, Mingtian Zhang, Zhongfan Jia, Qian Li, Chenglong Bao, Kaisheng Ma, Jun Zhu, and Yi Zhong. Afec: Active forgetting of negative transfer in continual learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 22379–22391. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/bc6dc48b743dc5d013b1abaebd2faed2-Paper.pdf.

Zirui Wang, Zihang Dai, Barnabas Poczos, and Jaime Carbonell. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. Minari, September 2024. URL `https://doi.org/10.5281/zenodo.13767625`.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

Wen Zhang, Lingfei Deng, Lei Zhang, and Dongrui Wu. A survey on negative transfer. *IEEE/CAA Journal of Automatica Sinica*, 10(2):305–329, 2023. doi: 10.1109/JAS.2022.106004.

Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Self-adaptive imitation learning: Learning tasks with delayed rewards from sub-optimal demonstrations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):9269–9277, Jun. 2022a. doi: 10.1609/aaai.v36i8.20914. URL `https://ojs.aaai.org/index.php/AAAI/article/view/20914`.

Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Self-adaptive imitation learning: Learning tasks with delayed rewards from sub-optimal demonstrations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):9269–9277, Jun. 2022b. doi: 10.1609/aaai.v36i8.20914. URL `https://ojs.aaai.org/index.php/AAAI/article/view/20914`.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, volume 8, pp. 1433–1438, 2008.

A Appendix

A.1 Task Descriptions

We evaluate CAIR on two continuous control benchmark suites categorized by sparse reward manipulation tasks (Fetch and Adroit Tasks) and dense-reward locomotion tasks (D4RL MujoCo tasks).

A.1.1 Fetch Robotics Tasks

We use three tasks based on the 7-DoF Fetch robotics arm (Plappert et al., 2018): *FetchPickAndPlace-v1* (P&P), *FetchSlide-v1* (FS), and *FetchPush-v1* (FP). Visualizations are provided in Figure 4. These are multi-goal tasks where the target position is randomized every episode. A rule-based suboptimal demonstrator (Bajaj et al., 2023) is used for each task.

Common Properties:

- **State Space $\mathcal{S}$:** Observations include the Cartesian position, linear velocity, and gripper state of the robot, along with the object's position, rotation (Euler angles), velocities, and relative position to the gripper. The desired target coordinates and achieved goals are also observed (required for HER).
- **Action Space $\mathcal{A}$:** 4-dimensional continuous space. 3 dimensions control gripper movement in Cartesian coordinates ($\in [-1, 1]$), and the last dimension controls the gripper state ($\in [-1, 1]$). In *FetchSlide* and *FetchPush*, the gripper is locked.

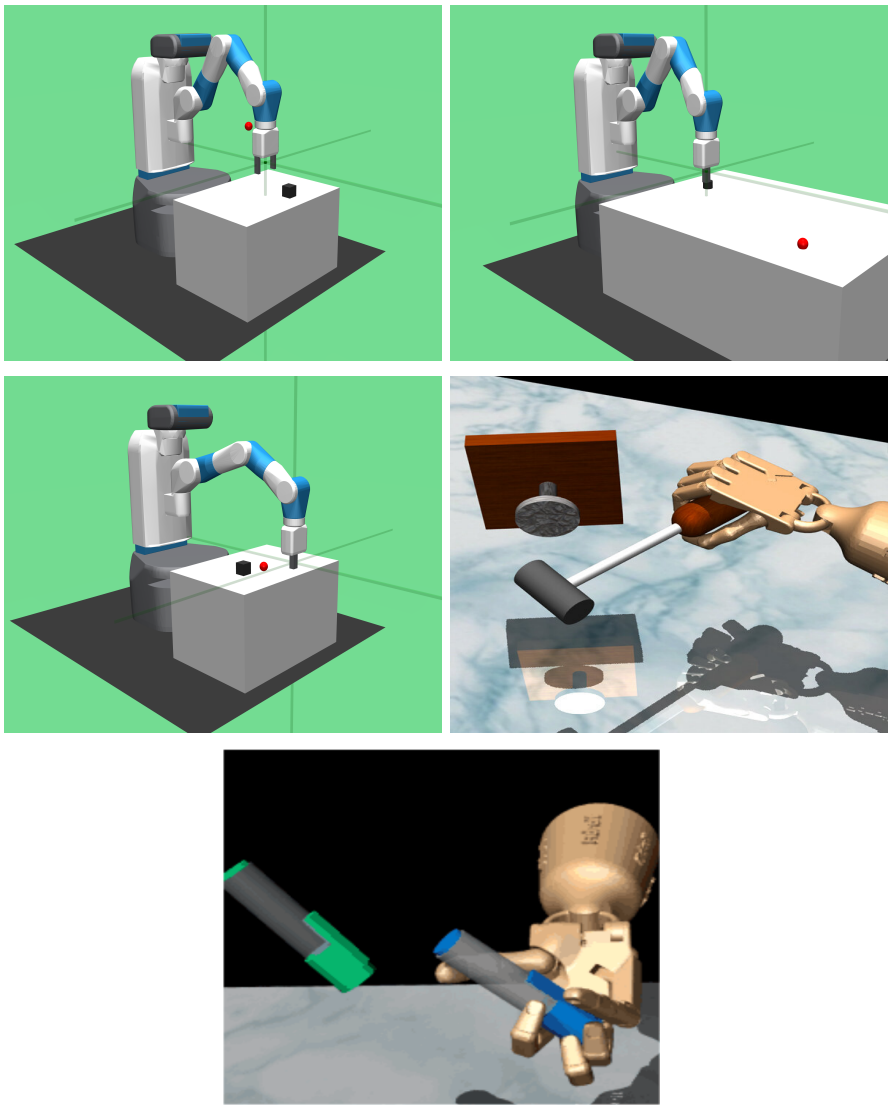


Figure 4: Visualizations of the four MuJoCo sparse-reward tasks used in our experiments. Top-left: FetchPickAndPlace-v1 task; Top-right: FetchSlide-v1 task; Middle-left: FetchPush-v1 task; Middle-right: AdroitHandHammerSparse-v1; Bottom: AdroitHandPenSparse-v1

- **Transition Function $\mathcal{P}$:** The robot updates its position based on the action. Each step corresponds to 20 simulator steps ($\Delta t = 0.002s$).

- **Reward Function R_T:** Sparse binary reward. +1 if the object is within 5cm of the target, 0 otherwise.

Task Specifics:

- **FetchPickAndPlace-v1:** The robot must grasp a box and move it to a target in the air or on the table. This is the most complex Fetch task due to the grasping requirement.

- **FetchSlide-v1:** The robot must push a puck across a long table to a target outside its immediate reach. This requires learning the object's sliding dynamics to apply precise force.

- **FetchPush-v1:** The robot pushes a box to a target location within its reach. This is simpler than Slide but still requires precise manipulation.

A.1.2 Adroit Manipulation Tasks

We evaluate CAIR on two dexterous manipulation tasks from the D4RL benchmark (Fu et al., 2020): *AdroitHandHammerSparse-v1* (Hammer) and *AdroitPenSparse-v0* (Pen).Visualizations are provided in Figure 4. Both tasks use the 30-DoF Shadow Hand to perform fine-grained manipulation tasks requiring coordinated finger control. Expert demonstrations for these tasks are obtained from the D4RL datasets.

Common Properties:

- **State Space $\mathcal{S}$:** Observations include the positions, orientations, and velocities of manipulated objects, along with the Shadow Hand joint angles and joint velocities.
- **Action Space $\mathcal{A}$:** 24-dimensional continuous action space representing joint torques of the Shadow Hand ($\in [-1, 1]$).
- **Transition Function $\mathcal{P}$:** Actions apply torques to the hand joints, producing object manipulation dynamics governed by MuJoCo physics.

Task Specifics:

- **AdroitHandHammerSparse-v1:** The hand must grasp a hammer and drive a nail into a board. The state includes hammer pose, nail position, and nail depth. The reward is sparse: $+10$ when the nail is fully driven and -0.1 otherwise to encourage efficiency.
- **AdroitHandPenSparse-v1:** The hand must reorient a pen to match a target orientation. Observations include the pen pose, angular velocities, and target orientation. The reward is sparse: $+50$ when the pen orientation is close.

A.1.3 D4RL Locomotion Tasks

We additionally evaluate CAIR on six continuous locomotion tasks from the D4RL benchmark (Fu et al., 2020): *HalfCheetah*, *Hopper*, *Walker2d*, *Ant*, *Swimmer*, and *Humanoid*. These environments involve high-dimensional continuous control tasks where the objective is to maximize forward locomotion reward under complex MuJoCo dynamics. Unlike the sparse-reward manipulation tasks, these tasks use dense reward functions and evaluate sustained locomotion behavior.

Common Properties:

- **State Space $\mathcal{S}$:** Observations consist of joint positions, joint velocities, body orientations, and task-specific proprioceptive features describing the current locomotion state.
- **Action Space $\mathcal{A}$:** Continuous action spaces corresponding to actuator torques applied to the robot joints.
- **Transition Function $\mathcal{P}$:** Actions apply torques to the articulated body, with future states determined by MuJoCo physics simulation.
- **Reward Function R_T:** Dense reward functions encouraging forward velocity while penalizing unstable motion and excessive control effort.

Task Specifics:

- **HalfCheetah:** A planar two-legged agent that learns fast forward locomotion through coordinated joint movement.

- **Hopper:** A single-legged hopping robot that must balance while maintaining forward movement.
- **Walker2d:** A bipedal robot that must learn stable walking and running behavior while avoiding falls.
- **Ant:** A four-legged quadrupedal robot with a high-dimensional action space requiring coordinated locomotion control.
- **Swimmer:** A low-dimensional swimming agent that learns propulsion through oscillatory body motion.
- **Humanoid:** A high-dimensional humanoid robot requiring coordinated full-body locomotion and balance.

A.2 Hyperparameters

Experiments were conducted on an AMD Ryzen Threadripper PRO 5975WX (32-Core, 4.3GHz). Tables 5–7 and Tables 8–10 detail the hyperparameters used across the manipulation and locomotion tasks, respectively. Unless otherwise specified, all network architectures use tanh activations.

Manipulation Tasks: For the Fetch and Adroit environments, we use 50 demonstration trajectories per task. Episode lengths are capped at 1,000 steps for the Fetch tasks and 4,000 steps for the Adroit tasks.

Locomotion Tasks: For the D4RL locomotion environments, we use the corresponding `simple` Minari datasets (Younis et al., 2024). Hyperparameters are initialized using the default RL Baselines3 Zoo benchmark configurations (Raffin, 2020), with only minor modifications where required for stable IL–RL annealing.

Task	**Network**	γ	**LR**	**Batch**	τ	**Ent Coef**
P&P, Slide	(512, 512)	0.98	$3 \cdot 10^{-4}$	128	0.02	Auto
Push	(512, 512)	0.98	$3 \cdot 10^{-5}$	256	0.02	Auto
AdroitHammer/Pen	(512, 512)	0.98	$3 \cdot 10^{-4}$	256	0.02	Auto

Table 5: Hyperparameters for CAIR (BC/DAgger + SAC)

Task	**Network**	γ	**LR**	**Batch**	τ	**Noise** σ
P&P, Slide	(512, 512)	0.98	$3 \cdot 10^{-4}$	128	0.02	0.15
Push	(512, 512)	0.98	$3 \cdot 10^{-5}$	256	0.02	0.15
AdroitHammer/Pen	(512, 512)	0.98	$3 \cdot 10^{-4}$	128	0.02	0.10

Table 6: Hyperparameters for CAIR (BC/DAgger + DDPG)

Task	**Network**	γ	**Init** λ	**LR**	**Rollout Steps**	**Epochs**	**Ent Coef**
P&P	(512, 512)	0.98	0.95	$3 \cdot 10^{-4}$	1024	30	0.2
Slide, Push	(512, 512)	0.98	0.95	$3 \cdot 10^{-4}$	1024	20	0.02
AdroitHammer/Pen	(512, 512)	0.98	0.95	$3 \cdot 10^{-4}$	1024	20	0.02

Table 7: Hyperparameters for CAIR (BC/DAgger + PPO)

A.3 Sensitivity Analysis

We provide additional empirical analysis of the CAIR framework components.

- **CAGrad Ablation (Fig. 9):** Demonstrates the consistent performance gains from adding gradient conflict resolution.

Task	Network	γ	LR	Batch	τ	Ent Coef
HalfCheetah	(256, 256)	0.99	$3 \cdot 10^{-4}$	256	0.005	Auto
Hopper	(256, 256)	0.99	$3 \cdot 10^{-4}$	256	0.005	Auto
Walker2d	(256, 256)	0.99	$3 \cdot 10^{-4}$	256	0.005	Auto
Ant	(256, 256)	0.99	$3 \cdot 10^{-4}$	256	0.005	Auto
Swimmer	(256, 256)	0.999	$3 \cdot 10^{-4}$	256	0.005	Auto
Humanoid	(256, 256)	0.99	$3 \cdot 10^{-4}$	256	0.005	Auto

Table 8: Hyperparameters for CAIR (BC/DAgger + SAC) on locomotion tasks

Task	Network	γ	LR	Batch	τ	Noise σ
HalfCheetah	(256, 256)	0.99	$1 \cdot 10^{-3}$	256	0.005	0.10
Hopper	(256, 256)	0.99	$1 \cdot 10^{-3}$	256	0.005	0.10
Walker2d	(256, 256)	0.99	$1 \cdot 10^{-3}$	256	0.005	0.10
Ant	(256, 256)	0.99	$1 \cdot 10^{-3}$	256	0.005	0.10
Swimmer	(256, 256)	0.9999	$1 \cdot 10^{-3}$	256	0.005	0.10
Humanoid	(256, 256)	0.99	$1 \cdot 10^{-3}$	256	0.005	0.10

Table 9: Hyperparameters for CAIR (BC/DAgger + DDPG) on locomotion tasks

Task	Network	γ	Init λ	LR	Rollout Steps	Epochs	Ent Coef	Clip Range
HalfCheetah	(256, 256)	0.98	0.92	$2.06 \cdot 10^{-5}$	512	20	0.000401	0.1
Hopper	(256, 256)	0.999	0.99	$9.808 \cdot 10^{-5}$	512	5	0.002295	0.2
Walker2d	(256, 256)	0.99	0.95	$5.05 \cdot 10^{-5}$	512	20	0.000585	0.1
Ant	(256, 256)	0.99	0.95	$3 \cdot 10^{-4}$	2048	10	0.0	0.2
Swimmer	(256, 256)	0.9999	0.98	$3 \cdot 10^{-4}$	1024	10	0.0	0.2
Humanoid	(256, 256)	0.95	0.9	$3.569 \cdot 10^{-5}$	512	5	0.00238	0.3

Table 10: Hyperparameters for CAIR (BC/DAgger + PPO) on locomotion tasks

- **Sensitivity to η_λ (Fig. 11):** Shows that extreme learning rates for the λ-optimizer destabilize learning.
- **Update Interval (Fig. 12):** Highlights the trade-off between responsiveness and stability in λ updates.

A.4 Conflict-Averse IL to RL Algorithm

CAIR optimizes a convex combination of imitation and reinforcement learning losses, $L_\lambda = (1-\lambda)L_{\text{RL}} + \lambda L_{\text{IL}}$, while mitigating gradient conflicts via CAGrad and adaptive optimization of the mixture coefficient λ.

We evaluate CAIR with SAC, DDPG, and PPO as the RL algorithms, combined with Behavior Cloning (BC) and DAgger as the IL algorithms. The corresponding hyperparameters are reported in Tables 5, 6, and 7. All remaining hyperparameters follow the default settings of Stable-Baselines3 (Hill et al., 2018). The constraint parameter c for CAGrad and the learning rate for the λ-optimizer are fixed across tasks unless stated otherwise. The adaptive λ-optimization procedure is summarized in Algorithm 2 (Appendix A.4).

A.5 Baseline Implementation Details

A.5.1 Reward Phasing (RP)

Reward Phasing uses demonstrations to learn an auxiliary reward function via Adversarial Inverse Reinforcement Learning (AIRL) (Fu et al., 2017), which is then annealed during RL training. We follow the formulation and hyperparameters from Bajaj et al. (Bajaj et al., 2023). Annealing is performed every 200

Algorithm 1 Conflict-Averse IL to RL (CAIR)

Input: Policy Parameters θ_0, CAGrad constraint $c \in [0, 1)$, Demonstrations D, Optimizer `Opt`, Task update frequency T_λ.
Initialize $\lambda \leftarrow 1.0$, $\theta \leftarrow \theta_0$, $k \leftarrow 1$, $\mathcal{B} \leftarrow \emptyset$
for each episode **do**
  for each step t **do**
    $a_t \sim \pi(a_t \mid s_t; \theta_k)$
    Execute a_t, observe s_{t+1}, r_t
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
  end for
  for each gradient step **do**
    if Annealing Complete ($\lambda = 0$) **then**
      Compute gradients g_{RL}
      $\theta_{k+1} \leftarrow \texttt{Opt}(\theta_k, g_{RL})$
    else
      $L_\lambda \leftarrow (1 - \lambda) L_{\text{RL}}(\theta, \mathcal{B}) + \lambda L_{\text{IL}}(\theta, D)$
      Compute gradients $g_{IL}, g_{RL}, g_\lambda$
      $d = \textbf{CAGrad}(g_\lambda, [g_{IL}, g_{RL}], c)$
      $\theta_{k+1} \leftarrow \texttt{Opt}(\theta_k, d)$
    end if
    $k \leftarrow k + 1$
  end for
  if $\lambda > 0$ **then**
    Every T_λ episodes do $\lambda \leftarrow \textsc{OptimizeLambda}(g_{\text{IL}}, g_{\text{RL}}, \lambda)$
  end if
end for

Algorithm 2 $\textsc{OptimizeLambda}(g_{\text{IL}}, g_{\text{RL}}, \lambda_{\text{cur}})$

Input: Gradients $g_{\text{IL}}, g_{\text{RL}}$, current λ_{cur}, learning rate η_λ, threshold $\epsilon_\lambda > 0$
Normalize: $\tilde{g}_{\text{IL}} \leftarrow g_{\text{IL}}/\|g_{\text{IL}}\|$, $\tilde{g}_{\text{RL}} \leftarrow g_{\text{RL}}/\|g_{\text{RL}}\|$
Initialize primal $\lambda \leftarrow \lambda_{\text{cur}}$, dual $\mu \leftarrow \mu_0$
for $k = 1$ to K **do**
  $\tilde{g}_\lambda \leftarrow (1 - \lambda)\tilde{g}_{\text{RL}} + \lambda \tilde{g}_{\text{IL}}$
  $c_{\text{RL}} \leftarrow \langle \tilde{g}_{\text{RL}}, \tilde{g}_\lambda \rangle$, $c_{\text{IL}} \leftarrow \langle \tilde{g}_{\text{IL}}, \tilde{g}_\lambda \rangle$ {Cosine similarity between gradients}
  $J(\lambda, \mu) \leftarrow c_{\text{RL}} - \mu \max(0, -c_{\text{IL}})$ {RL alignment objective with IL misalignment penalty}
  $\lambda \leftarrow \lambda + \eta_\lambda \nabla_\lambda J$ {Gradient Ascent}
  $\lambda \leftarrow \mathit{clamp}(\lambda, 0, 1)$ {Project to $[0, 1]$}
  $\mu \leftarrow \max(0, \mu + \eta_\mu \max(0, -c_{\text{IL}}))$ {Dual Update}
  if $\lambda < \epsilon_\lambda$ **then**
    $\lambda \leftarrow 0$ {Annealing Completion Condition}
  end if
end for
return λ

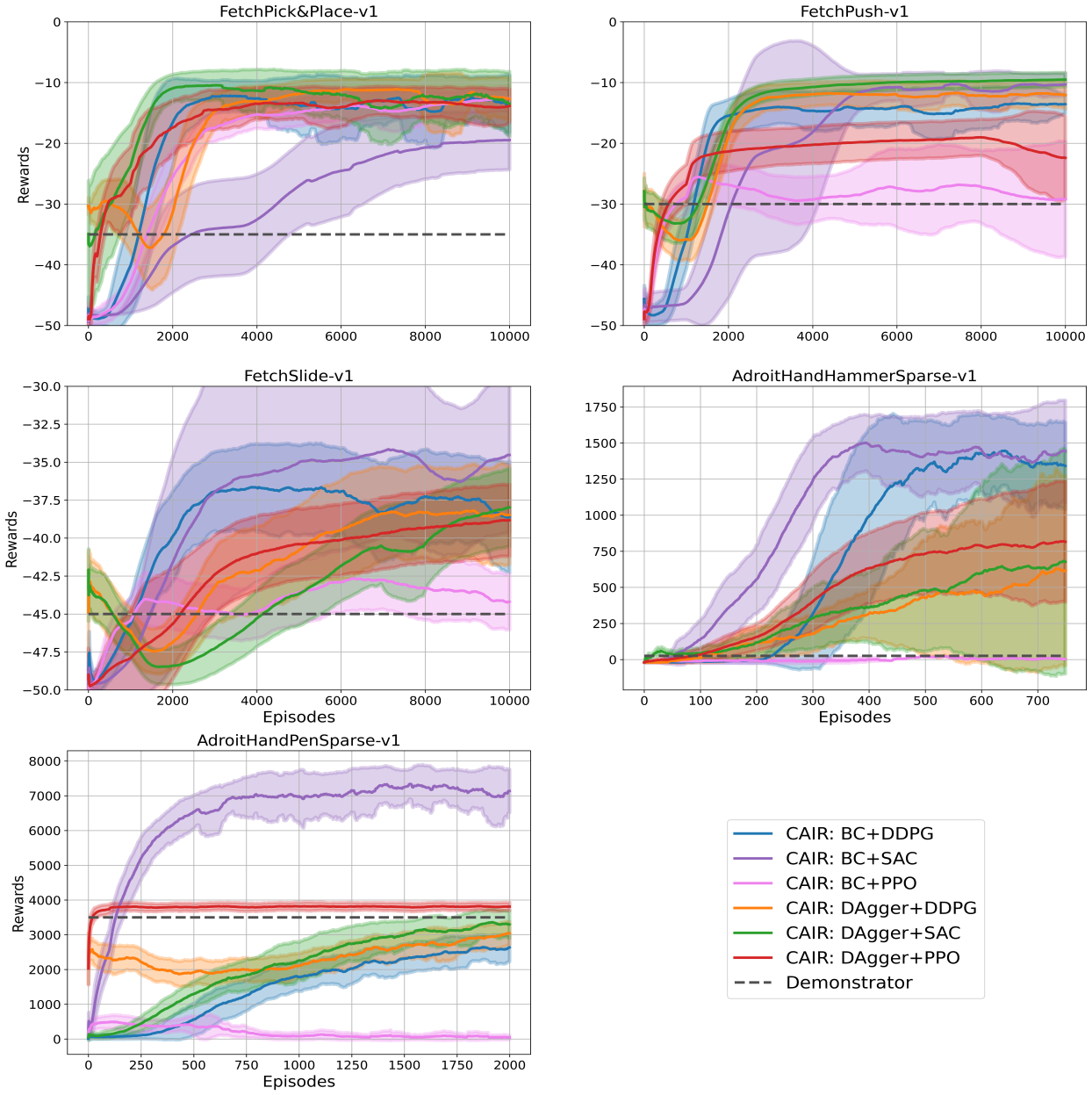


Figure 5: Learning curves for CAIR instantiated with different combinations of IL algorithms (BC, DAgger) and RL algorithms (SAC, DDPG, PPO). Positive learning trends are observed for all evaluated IL–RL pairings except BC+PPO on *AdroitHandHammerSparse-v1* and *AdroitHandPenSparse-v1*.

training episodes for Fetch tasks and every 20 episodes for AdroitHandHammer. To prevent performance collapse, the entropy coefficient is annealed from 0.005 to 0.001, the learning rate from 3×10^{-4} to 7×10^{-5}, and the AIRL reward weight α to 0.001 after 75% of the phasing schedule.

A.5.2 Hindsight Experience Replay (HER) with Demonstrations

HER with demonstrations augments the replay buffer with demonstration trajectories to reduce exploration difficulty in sparse-reward environments. We use DDPG (Lillicrap et al., 2015) as the underlying RL algorithm. The HER buffer is initialized with demonstration data. All remaining hyperparameters follow the default HER implementation.

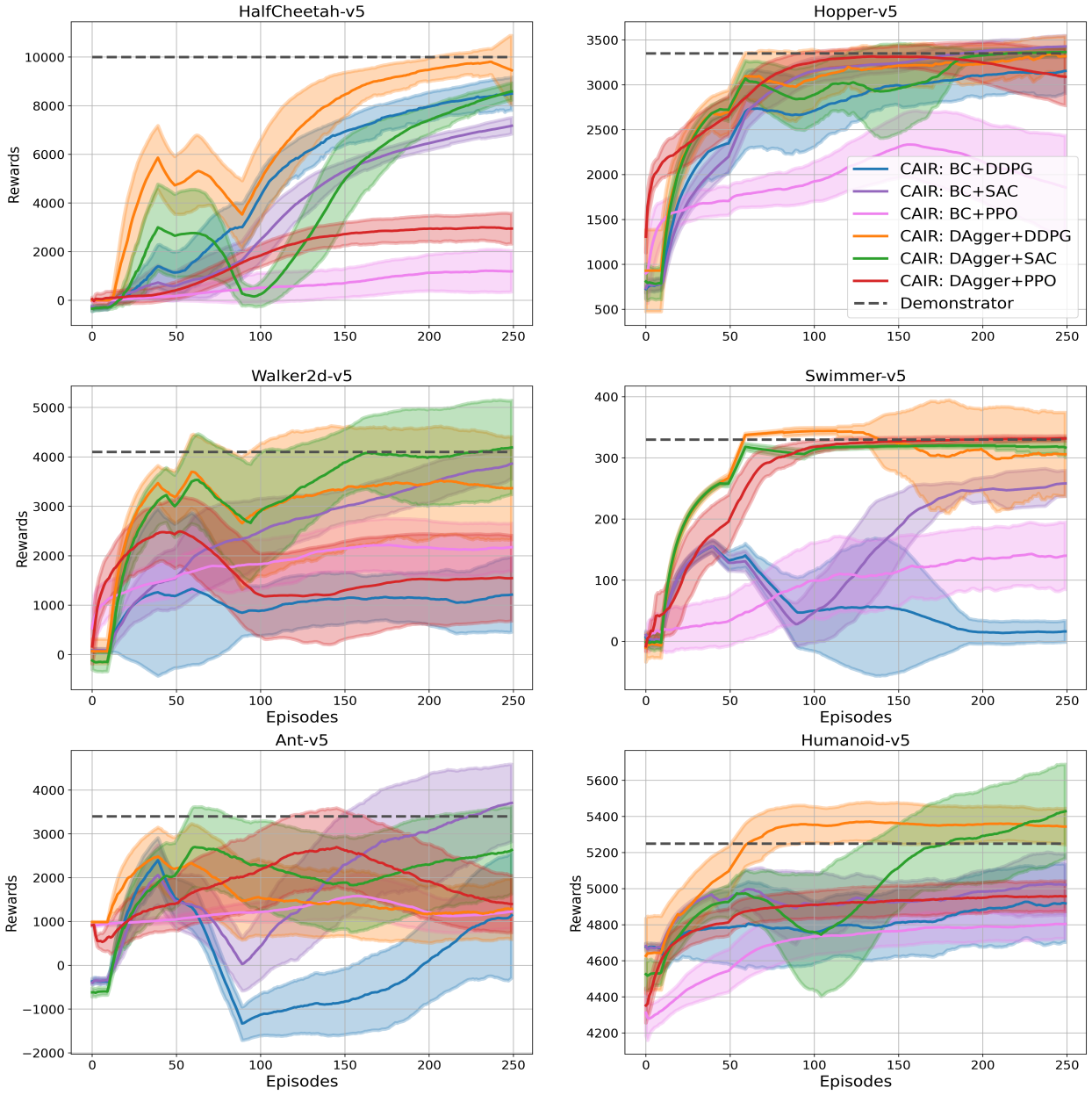


Figure 6: Learning curves for CAIR instantiated with different combinations of IL algorithms (BC, DAgger) and RL algorithms (SAC, DDPG, PPO) on D4RL locomotion tasks. Differences in learning performance are observed across IL–RL pairings, highlighting the impact of the underlying imitation and reinforcement learning algorithms.

A.5.3 Self-Adaptive Imitation Learning (SAIL)

SAIL dynamically augments the demonstration buffer with high-quality self-generated trajectories during training. We use TD3 (Schulman et al., 2015a) as the policy optimization algorithm. All hyperparameters, network architectures, and buffer configurations are taken directly from the authors' released implementation (Zhu et al., 2022b).

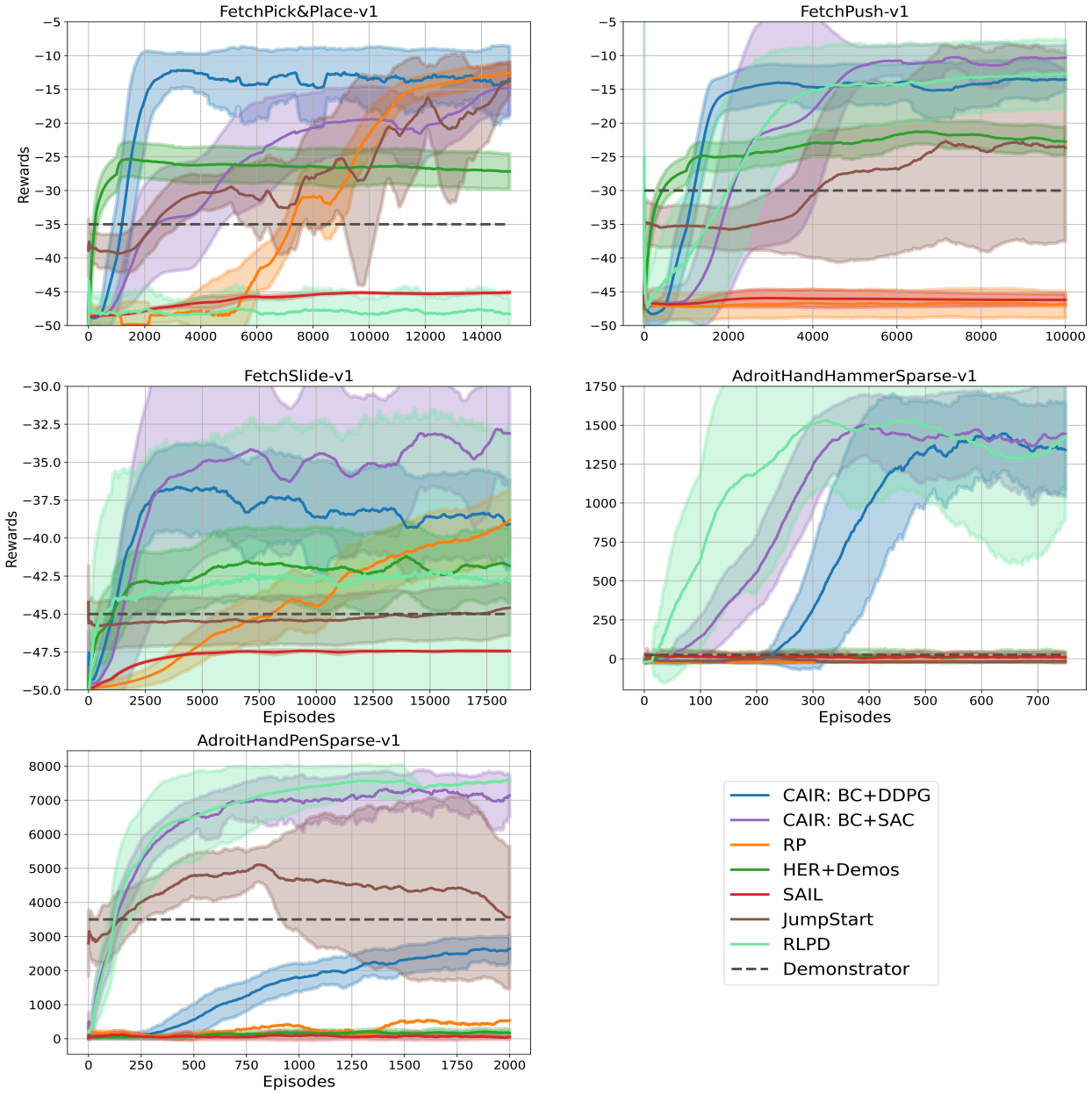


Figure 7: Learning trends for CAIR (BC+SAC and BC+DDPG) compared to hybrid-RL baselines on sparse-reward robotic manipulation tasks.

A.5.4 Jump-Start Reinforcement Learning (JSRL)

JSRL initializes training using a guide policy that steers the agent toward advantageous states at the start of each episode. The guide policy is gradually phased out by reducing the maximum guidance horizon. We use the formulation and hyperparameters from (Uchendu et al., 2022). The maximum guidance horizon is set to 40 for Fetch tasks (episode length 50) and 150 for AdroitHandHammerSparse (episode length 200).

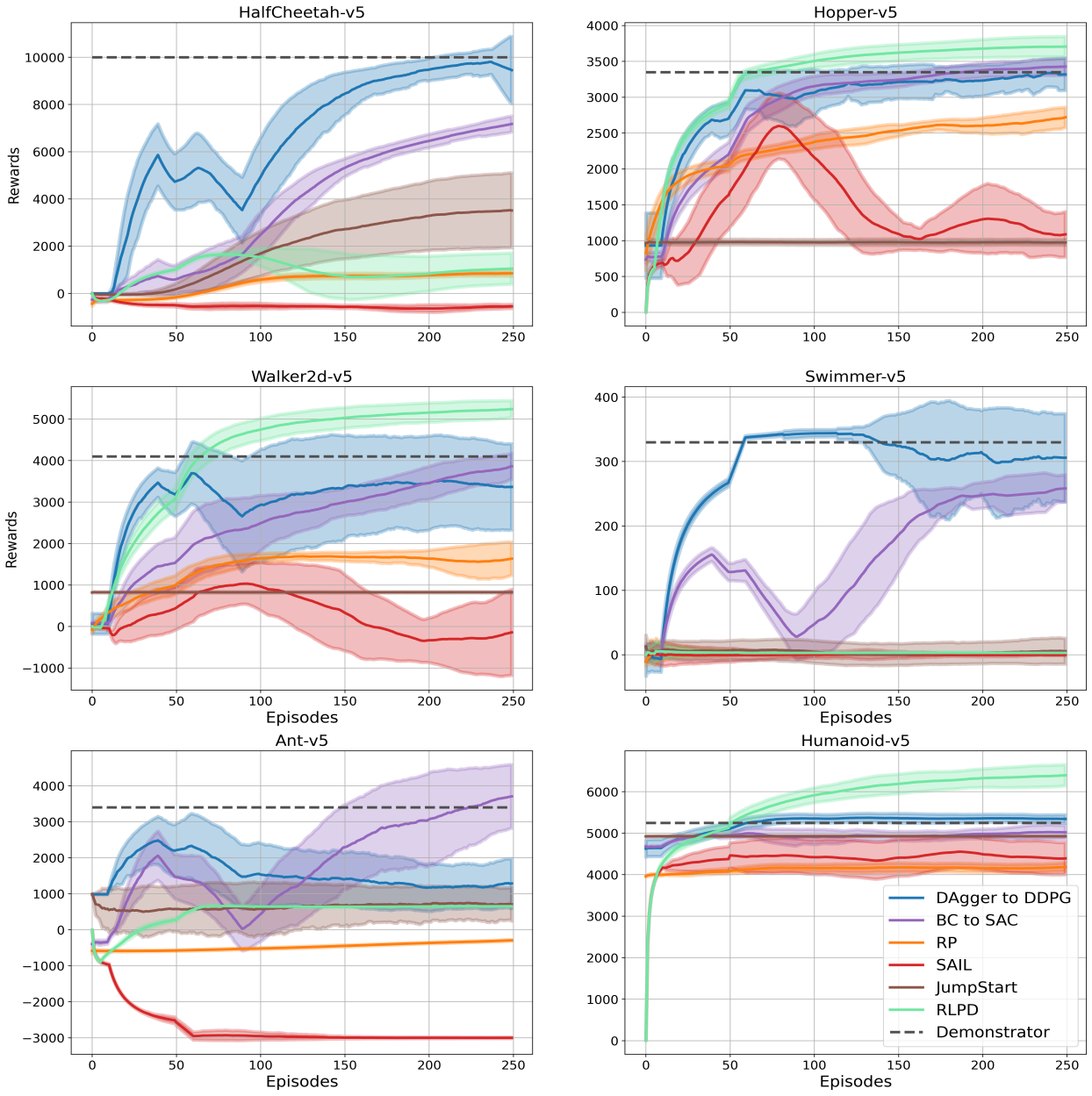


Figure 8: Learning trends for CAIR (BC+SAC and DAgger+DDPG) compared to hybrid-RL baselines on D4RL MuJoCo locomotion tasks.

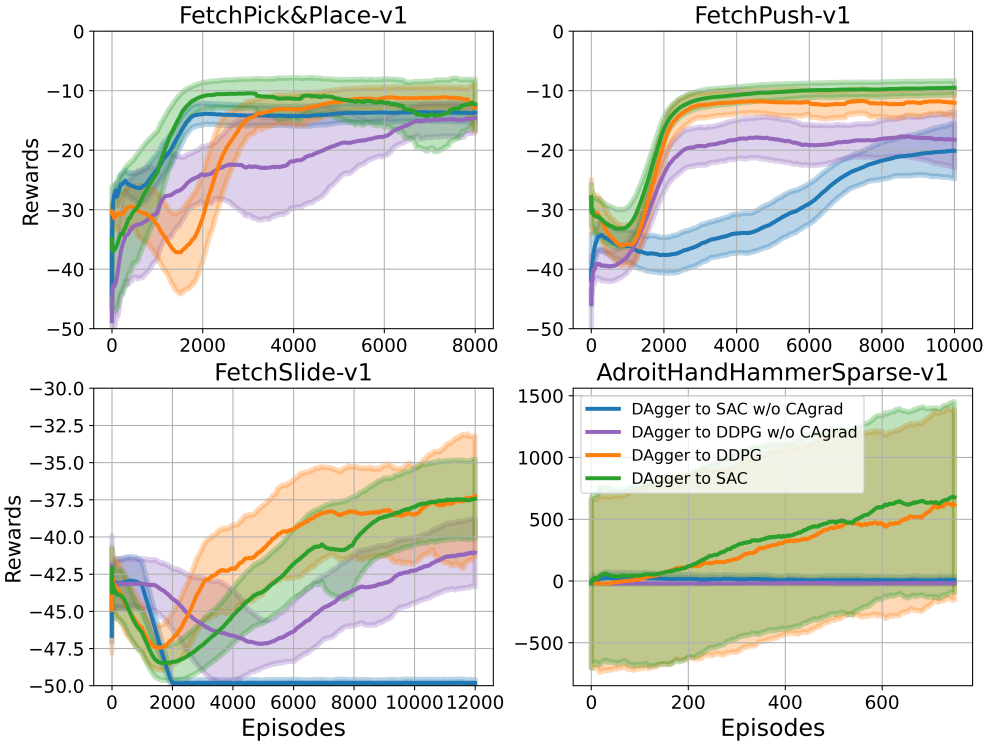


Figure 9: Ablation study evaluating the effect of CAGrad on learning performance across tasks. Improvements are observed in sample efficiency and asymptotic performance. (Fetch: 1 ep = 1k steps, Adroit: 1 ep = 4k steps).

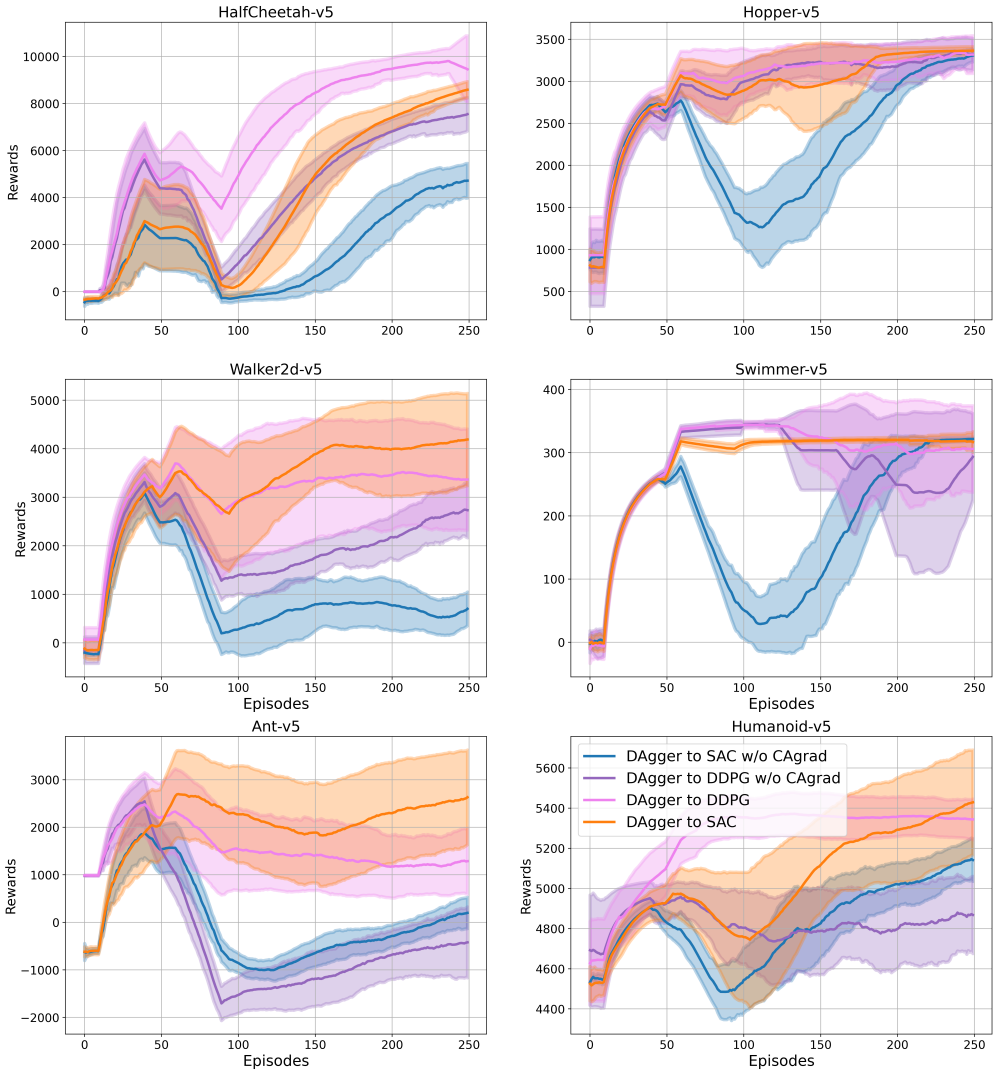


Figure 10: Ablation study evaluating the effect of CAGrad on learning performance across D4RL locomotion tasks. Incorporating CAGrad generally improves sample efficiency, enabling higher-return policies to be reached earlier during training. (Locomotion: 1 ep = 1k steps).

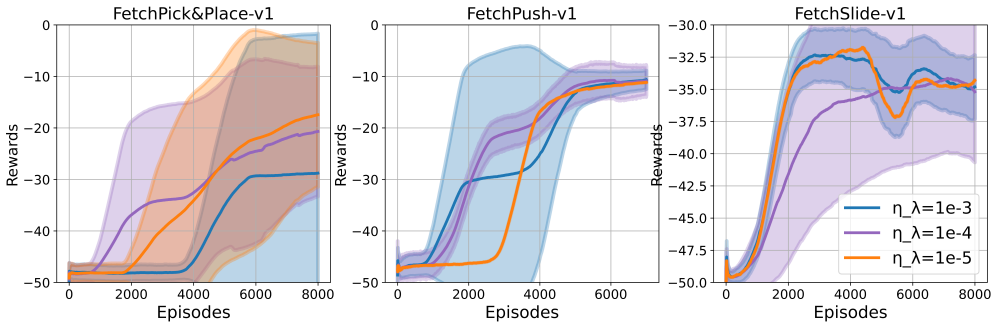


Figure 11: Sensitivity analysis: Effect of λ-optimizer learning rate (η_λ) on Fetch tasks.

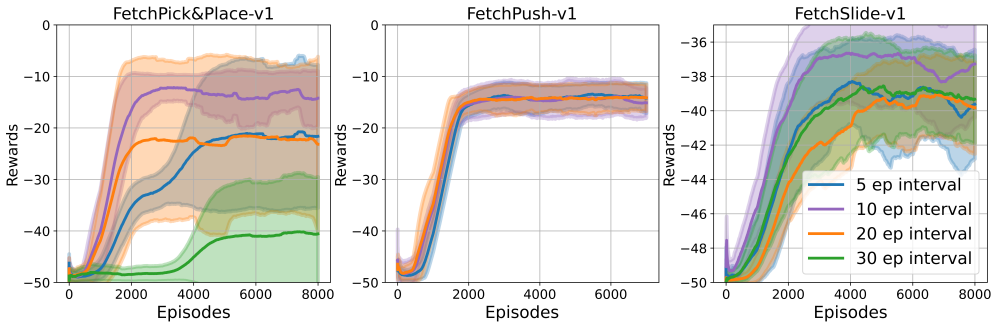


Figure 12: Sensitivity analysis: Effect of λ update interval on Fetch tasks.

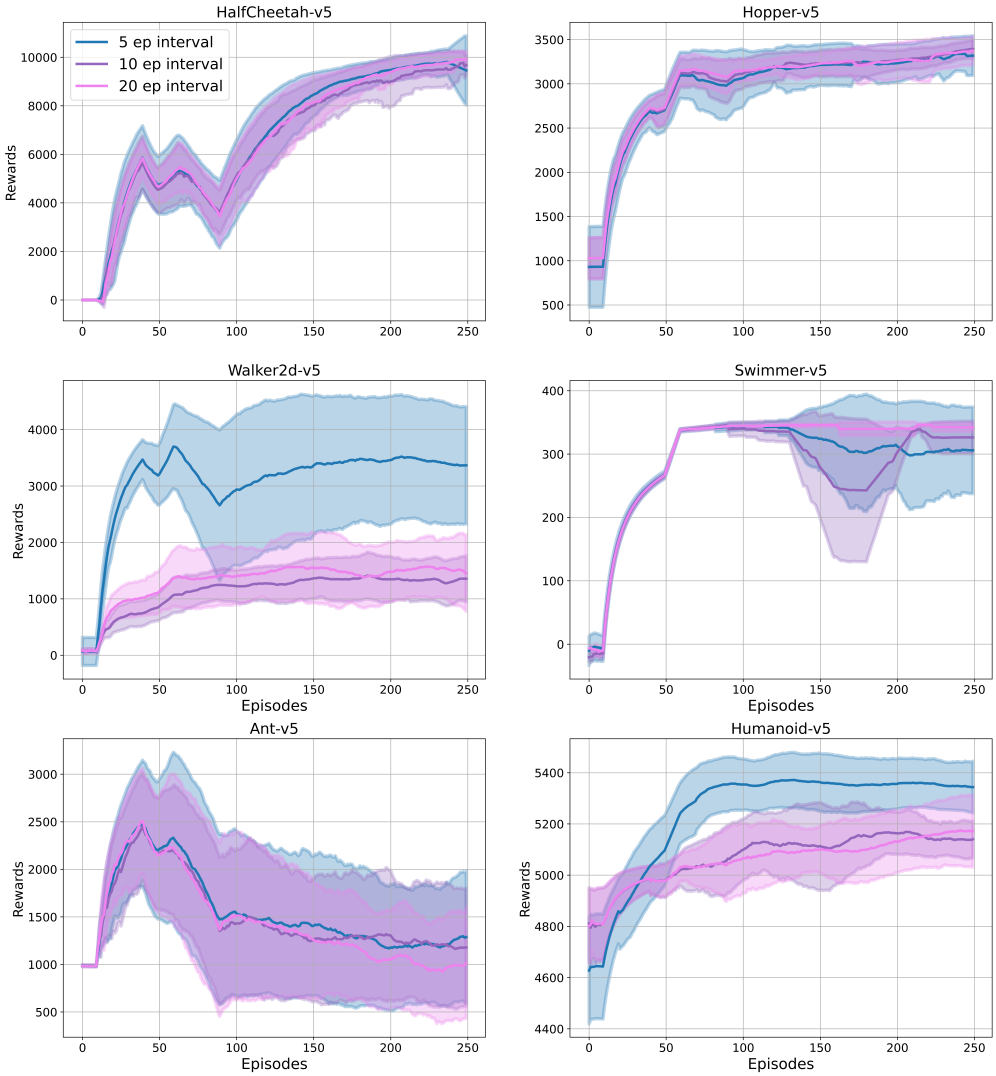


Figure 13: Sensitivity analysis: Effect of λ update interval on D4RL MuJoCo locomotion tasks.

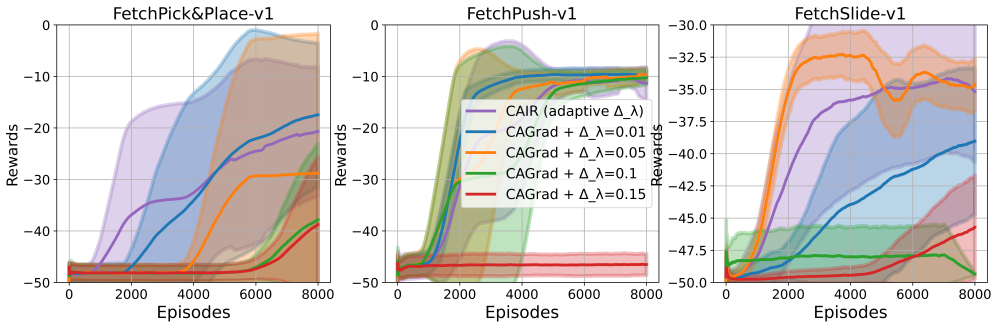


Figure 14: Ablation study comparing adaptive annealing (CAIR) with fixed schedules $\lambda \leftarrow \lambda - \Delta$ across Fetch tasks. Fixed schedules are sensitive to the choice of Δ, whereas adaptive annealing achieves consistent learning without schedule tuning.

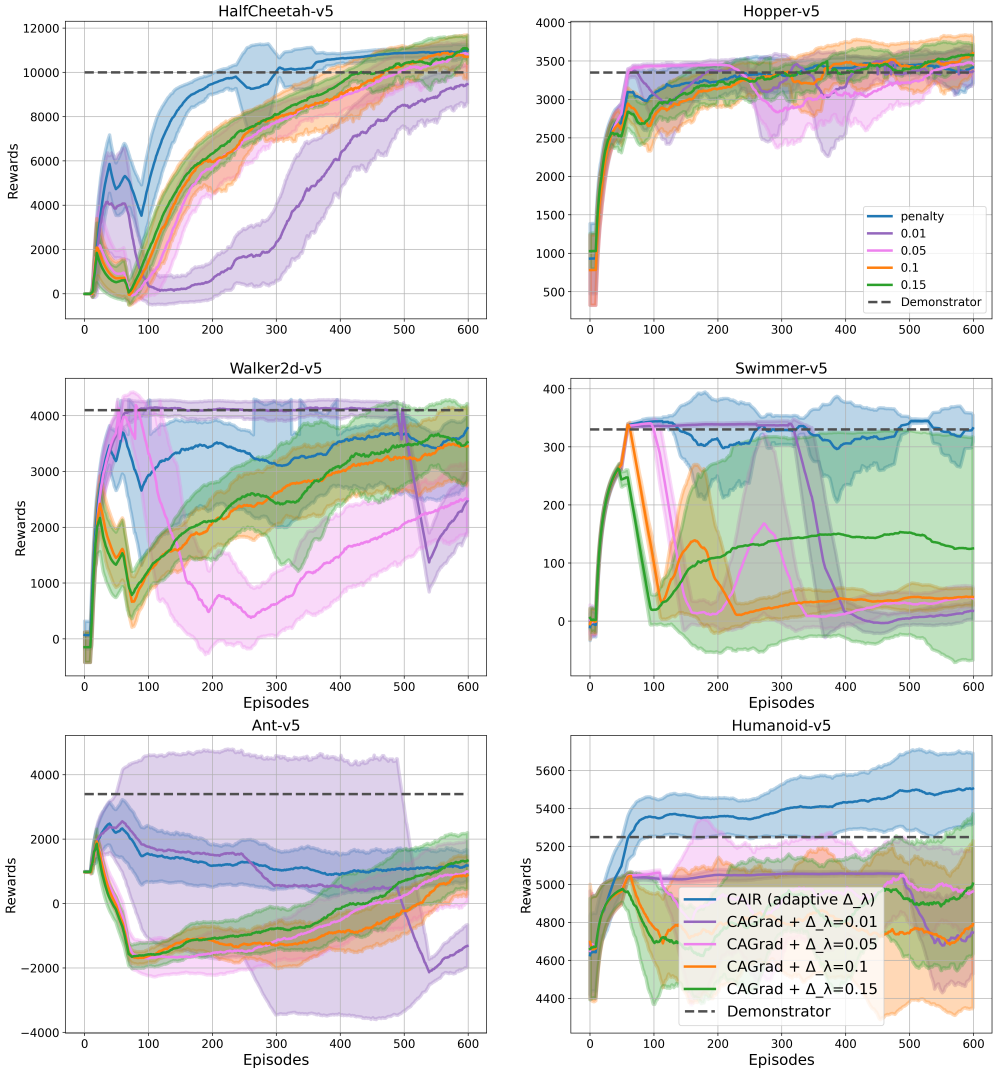


Figure 15: Ablation study comparing adaptive annealing (CAIR) with fixed schedule annealing $\lambda \leftarrow \lambda - \Delta$ across D4RL locomotion tasks. Adaptive annealing achieves improved sample efficiency and smoother reward progression, whereas fixed schedules often exhibit performance drops near the end of annealing before recovering.