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

Vaibhav Bajaj *vaibhavbajaj@tamu.edu*
Department of Computer Science & Engineering
Texas A&M University

Bo Liu *bliu@cs.utexas.edu*
Department of Computer Science
The University of Texas at Austin

Peter Stone *pstone@cs.utexas.edu*
Department of Computer Science
The University of Texas at Austin

Guni Sharon *guni@tamu.edu*
Department of Computer Science & Engineering
Texas A&M University

Reviewed on OpenReview: *https://openreview.net/forum?id=S98skK3FfD*

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 a negative inner product between IL and RL gradients might result in a counter-productive policy update step. Addressing both challenges, we introduce the Conflict-Averse IL-RL (CAIR) framework. Given IL and RL baseline algorithms, CAIR combines them through two components: (1) Loss Manipulation: adaptive weighting of the IL and RL losses targeting maximal RL weight subject to a capped gradient misalignment between the policy update direction and the IL gradient. (2) Gradient Manipulation: Conflict-Averse Gradient Descent (CAGrad) adjusts the resulting policy update step to balance the normalized predicted first-order decreases of both objectives (IL and RL). For an idealized stagewise optimization process, we establish conditional monotonic non-deterioration in expected return between successive stages. We evaluate CAIR in (1) sparse-reward robotic manipulation (Fetch and Adroit), where existing RL algorithms (without demonstrations) struggle to converge, and (2) dense-reward continuous control locomotion (Gymnasium MuJoCo), where existing RL algorithms (without demonstrations) can converge effectively yet demonstrations can be used to improve sample efficiency. We evaluate CAIR over various IL (BC, DAgger) and RL (DDPG, SAC, PPO) combinations. The results show that the BC+SAC variant and DAgger+DDPG variant attain the highest average normalized reward among the baseline algorithms in the sparse-reward and dense-reward domains respectively. Evaluations across IL and RL combinations demonstrate CAIR's applicability to diverse IL–RL pairings, with pairing- and domain-dependent performance and no consistently dominant combination.

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; Sharon, 2021; Mirzaei et al., 2018), 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., 2022), 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 direction (negative cosine similarity), leading to unstable or suboptimal updates; differences in their magnitudes can further skew the combined gradient. 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., 2022). 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 aims to mitigate negative transfer arising from gradient conflict during the IL-to-RL transition. Rather than following a predefined schedule, its adaptive loss annealing iteratively adjusts the weights of individually normalized IL and RL gradients toward the largest RL weight for which their combination remains non-negatively aligned with the IL gradient. When these gradients conflict, the corresponding IL target is the smallest feasible IL weight, which retains more imitation as the conflict increases. Normalization bases the criterion on directional compatibility rather than relative gradient magnitudes.

Conflict-Averse Gradient Descent (CAGrad) (Liu et al., 2021) complements loss annealing by balancing progress on the two objectives at the selected weights. It adjusts the same normalized gradient mixture within a bounded region to maximize the smaller predicted first-order decrease, measured relative to each objective's gradient norm. Annealing determines how imitation is withdrawn, whereas CAGrad balances the policy update without changing those weights. Once annealing completes, CAIR bypasses CAGrad and optimizes the RL objective alone. During the transition, the two components aim to limit both interference from IL that impedes RL fine-tuning and interference from RL that prematurely reverses useful imitation-induced behavior. In the representative analyses presented in Section 3.3, combining adaptive annealing with CAGrad reduces negative alignment and improves the measured worst-case gradient alignment.

For an idealized stagewise optimization process, we establish conditions under which expected return does not decrease between successive stages. Following trust-region analysis (Schulman et al., 2015), these conditions require KL-proximity between successive policies and surrogate improvement sufficient to offset the KL penalty at each stage. We also identify local conditions under which both requirements hold jointly for

sufficiently small annealing steps. 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). The results show that CAIR can be effectively instantiated with diverse IL–RL pairings, while also demonstrating that the strongest-performing pairing varies across benchmark tasks i.e., no combination universally dominates the others. On the other hand, the on-policy BC+PPO combination is generally weaker than the off-policy pairings in the evaluated tasks. 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 Gymnasium MuJoCo locomotion tasks. The results demonstrate that CAIR achieves the strongest overall performance on three of the six dense-reward locomotion tasks, while RLPD performs better on the remaining three. Overall, the results show that two variants of CAIR (DAgger+DDPG and BC+SAC) provide competitive performance relative to hybrid IL–RL baselines across both benchmark suites.

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 a *trajectory*, 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. We evaluate both sparse-reward manipulation and dense-reward locomotion problems.

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, IL performance is strongly influenced by the quality and coverage of the 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., 2022). Because the weighted formulation does not require a particular functional form for L_{IL} or L_{RL}, beyond both yielding gradients with respect to the shared policy parameters, it provides a general mechanism for transitioning between the two objectives. Formally, the conventional combined objective is:

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

Gradient Conflict and Negative Transfer. Let $g_{\mathrm{IL}} = \nabla_\theta L_{\mathrm{IL}}$ and $g_{\mathrm{RL}} = \nabla_\theta L_{\mathrm{RL}}$ denote the unweighted loss gradients. The conventional combined-loss gradient is

$$g_\lambda = \nabla_\theta L_\lambda = (1-\lambda) g_{\mathrm{RL}} + \lambda g_{\mathrm{IL}}.$$

We measure *alignment* using the inner product or its normalized form (cosine similarity), where $\langle g_i, g_j\rangle > 0$ indicates aligned directions and $\langle g_i, g_j\rangle < 0$ indicates conflicting directions. CAIR instead combines individually normalized loss gradients, as defined in Section 3.1.

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 a proposed direct descent step $-\alpha d$ has $\langle g_{\mathrm{IL}}, d\rangle < 0$. 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 target by adapting the relative gradient weights and the proposed update direction. 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; 2015). Alternatively, off-policy training methods define implicit curricula through shifted sample distributions (Hester et al., 2018; Nair et al., 2018; Zhu et al., 2022), though they typically lack formal convergence guarantees and rely on heuristically tuned schedules. A related line of work transfers *control* rather than loss weight: JIRL (Dey et al., 2021) imitates a baseline controller and gradually hands control to an off-policy RL agent, targeting low regret with respect to that baseline rather than the alignment of the two objectives' gradients.

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. The offline and hybrid RL baselines considered here do not explicitly manipulate gradient conflict between imitation and reinforcement 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., 2022). 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.

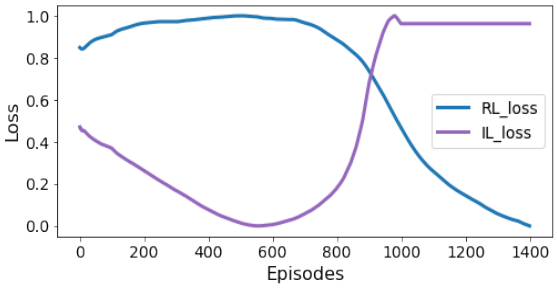
Gradient Manipulation in Multi-Task Learning. Conflicting gradients have been previously addressed in the context of Multi-Task Learning (MTL) through gradient manipulation methods (Désidéri, 2012; Yu et al., 2020; Sener & Koltun, 2018; Liu et al., 2021; 2023), which resolve gradient conflicts through joint objective optimization. Notably, CAGrad (Liu et al., 2021) guarantees asymptotic first-order stationarity of the average loss under its stated regularity and update conditions, 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).

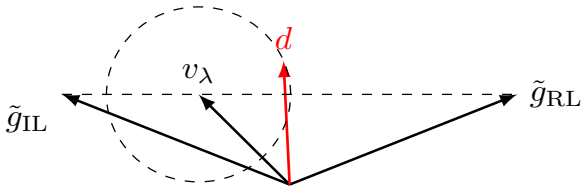
3 Conflict-Averse IL to RL

In this section, we formally present our framework, *Conflict-Averse IL-RL* (CAIR). CAIR adapts the relative weights of individually normalized IL and RL policy gradients to reduce harmful interference during the transition from imitation to reinforcement learning.

We achieve this through two complementary mechanisms: (1) **Adaptive Loss Annealing**, which iteratively adjusts the IL weight λ using an RL-alignment objective with a penalty for IL misalignment; and (2) **Conflict-Averse Gradient Descent (CAGrad)**, which adjusts the resulting normalized gradient mixture within a bounded region to maximize its minimum inner product with the two normalized constituent gradients. Annealing controls the transition, while CAGrad balances the update at the applied weights.



(a) Loss Manipulation



$$\max_{d \in \mathbb{R}^m} \min_{i \in \{\text{RL},\text{IL}\}} \langle \tilde{g}_i, d \rangle$$

$$\text{s.t. } \|d - v_\lambda\| \leq c\|v_\lambda\|$$

(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 that shifting the loss weights can move the policy away from imitation-induced behavior. (x-axis: Episodes; y-axis: normalized loss). **(b)** The optimization landscape for the CAGrad method. v_λ combines individually normalized gradients (drawn at a common display scale), while d maximizes their minimum inner product within the dashed deviation ball centered at v_λ.

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), fixed annealing illustrates tension between the two training objectives on the *FetchPush-v1* task (see Appendix A.1 for experimental details). We observe that the RL loss decreases while the IL loss spikes as the IL weight is reduced. This suggests that shifting the loss weights can move the policy away from imitation-induced behavior.

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., 2022), we optimize an alignment objective. For nonzero loss gradients, let

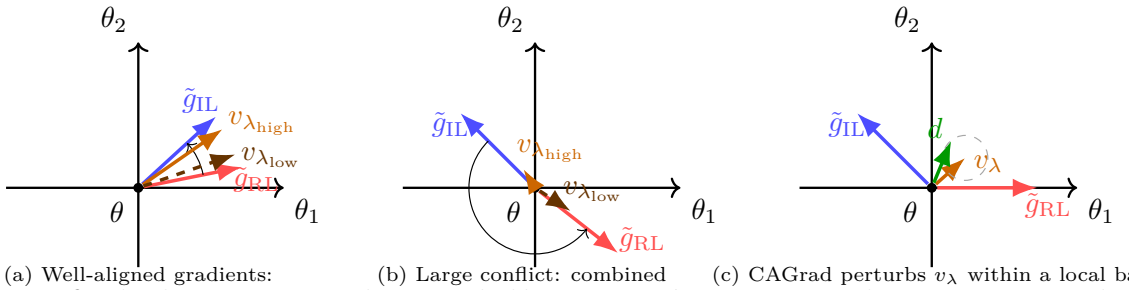
$$\tilde{g}_i = \frac{g_i}{\|g_i\|}, \qquad i \in \{\text{IL}, \text{RL}\}, \qquad v_\lambda = (1-\lambda)\tilde{g}_{\text{RL}} + \lambda \tilde{g}_{\text{IL}}.$$

The **alignment vector** v_λ is also the gradient mixture used for the policy update before CAGrad. Thus λ weights normalized loss gradients, rather than the raw losses in Equation 1. At a given policy, this construction is invariant to positive rescaling of either loss. The instantaneous target criterion is

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

For $v_\lambda \neq 0$, the constraint is equivalent to $\langle \tilde{g}_{\text{IL}}, v_\lambda \rangle \geq 0$.

Writing $q = \langle \tilde{g}_{\text{IL}}, \tilde{g}_{\text{RL}} \rangle$, for $-1 < q < 1$ the criterion has the unique solution $\lambda^\star = \max\{0, -q/(1-q)\}$: zero IL weight when the normalized gradients do not conflict, and otherwise the smallest IL weight satisfying the constraint, which grows with conflict severity and is less than $1/2$. Because $\lambda^\star$ depends only on q, whether



(a) Well-aligned gradients: pure RL satisfies the alignment criterion

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

(c) CAGrad perturbs v_λ within a local ball to maximize the minimum inner product

Figure 2: Geometric interpretation of loss annealing and gradient manipulation. Each panel shows a 2-D policy-parameter space (θ_1, θ_2) with normalized IL gradient $\tilde{g}_{\text{IL}}$ (blue), normalized RL gradient $\tilde{g}_{\text{RL}}$ (red), and mixture v_λ (unit constituents drawn at a common display scale). $v_{\lambda_{high}}$ (orange) uses a larger λ than $v_{\lambda_{low}}$ (dashed-brown). **(a)** When gradients are aligned, annealing λ (orange to dashed brown) maintains a consistent direction. **(b)** With strongly conflicting IL and RL gradients, a small decrease in λ can substantially rotate the combined gradient toward the RL gradient. **(c)** CAGrad calculates a corrected direction d (green) within a gradient-space deviation constraint around v_λ to maximize the minimum inner product with the normalized IL and RL gradients.

it reaches the completion threshold ϵ_λ is determined by directional compatibility rather than by gradient magnitudes.

CAIR does not set λ to the instantaneous solution $\lambda^\star$ directly. After the imitation-only warm start, it maintains an auxiliary weight λ_{opt}, which is updated by Adam at each policy-gradient update on a penalized form of Equation 2. The objective rewards exponentially smoothed cosine similarity between the normalized mixture and $\tilde{g}_{\text{RL}}$ and penalizes negative smoothed similarity with $\tilde{g}_{\text{IL}}$ through a multiplier that increases under negative smoothed IL alignment and decreases under positive alignment. This smoothing reduces the sensitivity of the weight adaptation to variation in individual minibatch gradient estimates.

The applied weight λ, which determines the policy-update direction, is refreshed from λ_{opt} only at fixed intervals and is held constant between refreshes. This is a stability-oriented design choice that separates the continual adaptation of λ_{opt} from changes to the objective used for policy optimization, rather than allowing the objective weighting to change after every gradient update. Periodic application does not constrain the magnitude of a change in λ when it is refreshed. Once $\lambda \leq \epsilon_\lambda$, it is fixed at zero and subsequent updates optimize only the RL objective. Smoothed, stochastic weight adaptation and periodic application need not satisfy the instantaneous constraint at every update. Algorithms 1 and 2 (Appendix A.4) give the exact update rules and boundary cases.

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 $\tilde{g}_{\text{IL}}$ and $\tilde{g}_{\text{RL}}$ is small, and the combined gradients $v_{\lambda_{\text{high}}}$ (solid orange) and $v_{\lambda_{\text{low}}}$ (dashed brown) remain close in direction. In this case, decreasing λ changes the combined gradient direction only slightly, and the alignment criterion permits a transition to pure RL.

Panel (b) illustrates how, with strongly conflicting IL and RL gradients, a small decrease in λ can substantially rotate the combined gradient toward the RL gradient. This motivates holding the applied weight between refreshes while using the alignment penalty to guide its adaptation under conflict, with the aim of keeping v_λ non-negatively aligned with the IL gradient.

3.2 Gradient Manipulation

Adaptive loss annealing controls the IL-to-RL transition through the weight λ. At the applied weight, CAIR uses Conflict-Averse Gradient Descent (CAGrad) (Liu et al., 2021) to balance progress on the two objectives independently of their gradient scales. For a direct step $-\alpha d$, the predicted first-order decrease of L_i per unit

step size is $\langle g_i, d\rangle$; dividing it by $\|g_i\|$ gives $\langle \tilde{g}_i, d\rangle$. These normalized decreases can differ even though both constituent gradients have unit norm: at the mixture, $\langle \tilde{g}_{\mathrm{IL}}, v_\lambda\rangle - \langle \tilde{g}_{\mathrm{RL}}, v_\lambda\rangle = (2\lambda - 1)(1 - q)$. For distinct directions, the RL decrease is smaller when $\lambda > 1/2$, and the IL decrease is smaller when $\lambda < 1/2$. CAGrad balances the update at these weights without changing λ, by solving

$$\max_{d\in\mathbb{R}^m} \min_{i\in\{\mathrm{IL},\mathrm{RL}\}} \langle \tilde{g}_i, d\rangle \quad \text{s.t.} \quad \|d - v_\lambda\| \le c\|v_\lambda\|, \qquad 0 \le c < 1. \tag{3}$$

The parameter c controls the allowable deviation from v_λ, as illustrated in Figure 2(c).

Since v_λ is feasible, an exact solution $d^\star$ satisfies $\min_i \langle \tilde{g}_i, d^\star\rangle \ge \min_i \langle \tilde{g}_i, v_\lambda\rangle$. Every feasible direction also satisfies $\langle v_\lambda, d\rangle \ge (1 - c)\|v_\lambda\|^2$. Thus exact CAGrad improves or preserves the smaller normalized predicted decrease while retaining positive alignment with the mixture when $v_\lambda \neq 0$. In the mixed phase $0 < \lambda < 1$, it preserves any component already non-negatively aligned with v_λ (Appendix A.6.2). In particular, whenever the instantaneous annealing constraint holds at the applied weight, $\langle \tilde{g}_{\mathrm{IL}}, d^\star\rangle \ge 0$. This links annealing's alignment criterion directly to the exact CAGrad direction without a condition on the constituent gradient norms.

The deviation constraint limits which conflicts CAGrad can correct. For $0 < \lambda < 1$ and $v_\lambda \neq 0$, an exact solution is non-negatively aligned with both components if and only if $\min_i \cos(\tilde{g}_i, v_\lambda) \ge -c$ (Appendix A.6.2). Every feasible direction is within angle $\arcsin c$ of the mixture. Annealing determines the center of this ball, while CAGrad balances the update within it. Increasing c widens the correction margin but weakens the lower bound $(1 - c)\|v_\lambda\|^2$ on mixture alignment.

The optimality, preservation, and recoverability statements concern the exact direction $d^\star$ before any optimizer transformation. The implementation uses a finite numerical solve on minibatch gradients and passes its output to the policy optimizer. Asymptotic first-order stationarity applies only to the fixed-scale scalarization and direct-update idealization specified in Section 3.4 and Appendix A.6.1, rather than to normalization recomputed at every minibatch or to Adam-transformed steps. Once annealing completes, CAIR bypasses CAGrad and optimizes the RL objective alone. Algorithm 1 summarizes the complete framework.

3.3 Empirical Gradient Analysis

A central challenge in IL-to-RL transfer lies in how the *normalized gradient mixture* v_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 (Dey et al., 2024); (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), an unadjusted mixture with $\langle g_{\mathrm{IL}}, v_\lambda\rangle < 0$ opposes the imitation gradient and may unlearn useful imitation-induced behavior. We examine whether annealing and CAGrad reduce this opposition in the direction supplied to the policy optimizer.

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. CAIR completes the transition by fixing $\lambda = 0$ once it reaches or falls below ϵ_λ. Subsequent updates use only the RL gradient, bypassing both λ-optimization and CAGrad (Algorithm 1).

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 normalized mixture v_λ, and the direction d supplied to the policy optimizer.

Results are presented for a single run on the *FetchPickAndPlace-v1* task. The *FetchPickAndPlace-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* (BC) and *DDPG* as the IL and RL algorithms, respectively. Representative results for the *BC+SAC* pairing on *FetchPush-v1* are additionally provided in Figure 16 in the appendix. Similar qualitative trends were observed for other IL–RL combinations listed in Section 3.5. To preserve the clarity of the temporal gradient evolution, these figures show a single representative seed, since averaging across seeds can smooth the transient gradient conflict events that the analysis is intended to illustrate. For clarity, each row in Figure 3 contains two plots for the same setting: the left column shows cosine similarity with the imitation gradient, $\cos(g_{\mathrm{IL}}, \cdot)$, and the right column shows cosine similarity with the RL gradient, $\cos(g_{\mathrm{RL}}, \cdot)$. This layout enables 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 v_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 v_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 illustrates that gradient manipulation cannot always compensate for conflicts induced by a fixed annealing schedule, consistent with CAGrad's bounded correction at the selected loss weights (Section 3.2).

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), with $c = 0.2$ for the CAGrad variant. Adaptive annealing alone improves alignment relative to fixed schedules but can still produce negatively aligned updates because its weight is adapted from smoothed, stochastic estimates and applied periodically. In contrast, CAIR—by jointly adapting λ and refining the update direction with CAGrad—maintains positive alignment with both the IL and RL gradients over the plotted mixed phase of this representative run. These results highlight the complementary roles of the two components: adaptive loss annealing guides changes in the loss weights using gradient alignment, while CAGrad balances the predicted first-order decreases normalized by each loss gradient norm at the update level.

3.4 Conditional Monotonic Non-Deterioration

We describe an idealized stagewise view of CAIR's annealing and give conditions for non-decreasing expected return between stages (Theorem 1). Because these conditions trade off against each other, we then identify local conditions under which they hold jointly for sufficiently small annealing steps (Proposition 1). Proofs are given in Appendix A.6.

Setting. To relate normalized gradient weighting to a fixed objective, fix positive scale factors $\kappa_{\mathrm{IL}}, \kappa_{\mathrm{RL}}$ and write $\kappa = (\kappa_{\mathrm{IL}}, \kappa_{\mathrm{RL}})$. Consider

$$L^{\kappa}_{\lambda}(\theta) = \frac{\lambda}{\kappa_{\mathrm{IL}}} L_{\mathrm{IL}}(\theta) + \frac{1-\lambda}{\kappa_{\mathrm{RL}}} L_{\mathrm{RL}}(\theta). \tag{4}$$

At a policy where $\kappa_i = \|g_i\|$ for $i \in \{\mathrm{IL}, \mathrm{RL}\}$, its gradient equals v_λ when these scales are treated as constants. For the stagewise analysis, the scales remain fixed along the family, and we use the finite schedule $1 = \lambda_0 > \lambda_1 > \cdots > \lambda_N = 0$. CAGrad is centered at $\nabla L^{\kappa}_{\lambda_k}$ and uses the component gradients g_i/κ_i. Under Assumption 1 and idealized exact-gradient direct updates, the deviation constraint yields asymptotic first-order stationarity of each fixed scalarization (Appendix A.6.1).

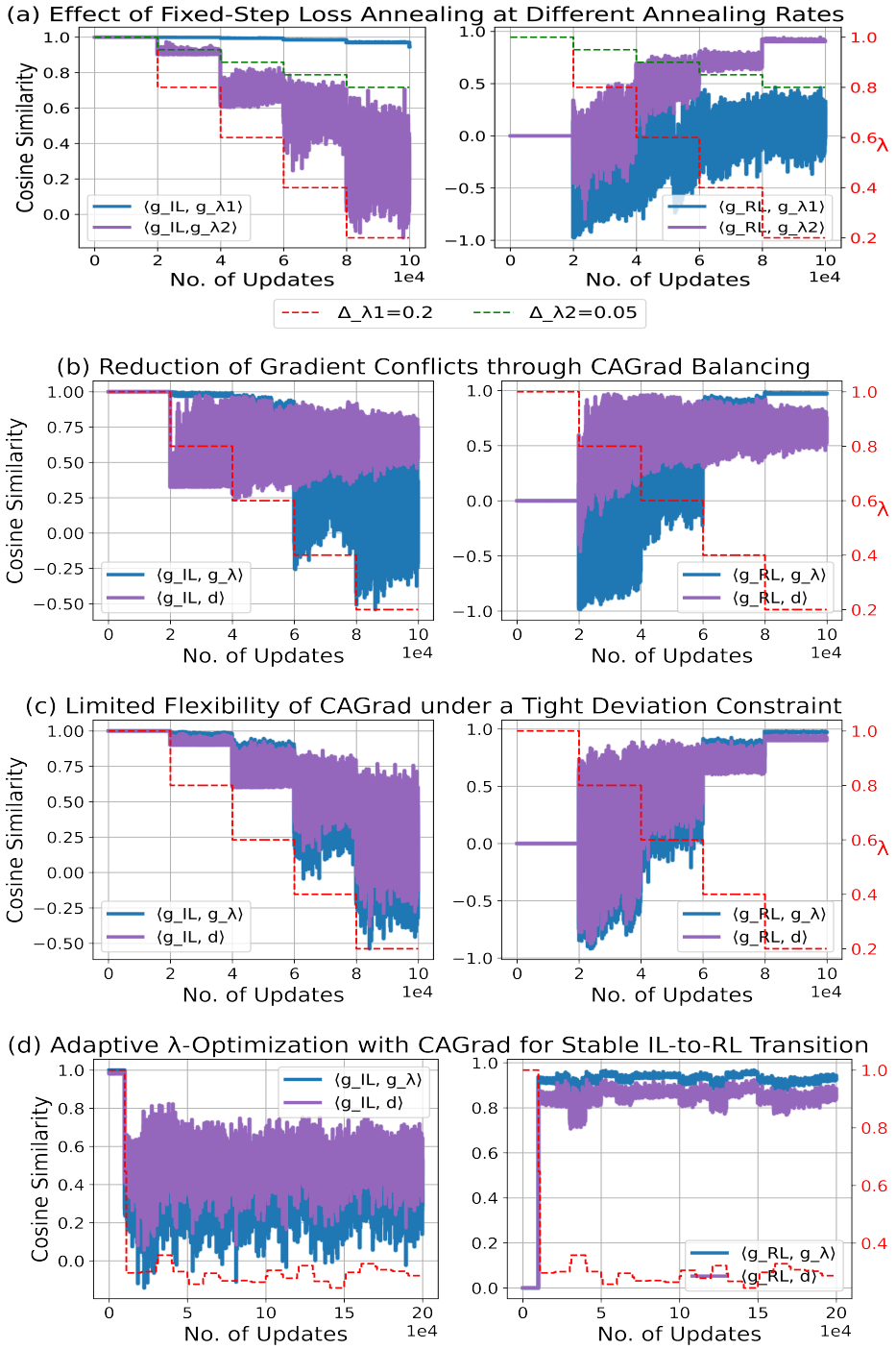


Figure 3: Cosine similarity analysis on *FetchPickAndPlace-v1* during the IL-to-RL transition. Each row shows cosine similarity between the direction supplied to the policy optimizer and g_{IL} (left) or g_{RL} (right). **(a)** Naïve fixed annealing leads to negative alignment (conflict). **(b)** CAGrad 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 over the plotted mixed phase of this representative run, even with a tight constraint. (x-axis: gradient updates; y-axis: cosine similarity averaged over 10 updates).

We additionally assume that each stage converges to a finite stationary parameter vector θ_k^*. This point is Pareto-stationary for the IL–RL bi-objective; we denote its policy π_k^*. Upon convergence, the weight is updated to λ_{k+1} and optimization of $L^{\kappa}_{\lambda_{k+1}}$ is initialized at θ_k^*, thereby warm-starting the next stage. Holding the scales fixed is an idealization: the implementation recomputes the gradient norms at each minibatch and does not optimize one such fixed scalarization throughout a stage.

Assume bounded rewards, $|R(s,a)| \le R_{\max} < \infty$ for all (s,a). For $\gamma \in (0,1)$, let ρ_π denote the unnormalized discounted state-visitation measure under the initial-state distribution used to define J:

$$\rho_\pi(ds) := \sum_{t=0}^{\infty} \gamma^t \Pr_\pi(s_t \in ds).$$

Assumption 1 supports the stagewise construction; Assumptions 2–3 and the dominance condition below ensure non-decreasing expected return between consecutive policies.

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 $L^{\kappa}_{\lambda_k}(\theta)$ is bounded below for every λ_k, with fixed $\kappa_{\mathrm{IL}}, \kappa_{\mathrm{RL}} > 0$.

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

$$D_k^{\max} := \sup_{s \in \mathcal{S}} D_{\mathrm{KL}}\big(\pi_k^*(\cdot \mid s) \,\big\|\, \pi_{k+1}^*(\cdot \mid s)\big) \;\le\; \delta_k,$$

for some stage-specific $\delta_k > 0$, using natural logarithms.

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 lower bounded:*

$$S_k := \int_{\mathcal{S}} \rho_{\pi_k^*}(ds)\, \mathbb{E}_{a \sim \pi_{k+1}^*(\cdot|s)}\Big[A_{\pi_k^*}(s,a)\Big] \;\ge\; \eta_k,$$

for some $\eta_k > 0$, which may vary across stages.

Let

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

where C_k is the per-stage KL penalty constant obtained from the TRPO performance bound (Theorem 1 in Schulman et al. (2015)) using Pinsker's inequality (Appendix A.6.3). The bounded-reward assumption ensures $\epsilon_k \le 2R_{\max}/(1-\gamma) < \infty$.

Theorem 1 (Conditional Monotonic Non-Deterioration)**.** *In a discounted MDP with bounded rewards and $\gamma \in (0,1)$, let $\pi_0^*, \ldots, \pi_N^*$ be a sequence of policies. Suppose Assumptions 2–3 hold and $\eta_k \ge C_k\delta_k$ for every $k = 0, \ldots, N-1$. Then the expected return is non-decreasing along the sequence:*

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

The proof is given in Appendix A.6.3. The theorem applies to the stationary policies π_k^* above whenever they satisfy its conditions. Because η_k and δ_k are stage-specific, the essential requirement is $S_k \ge C_k D_k^{\max}$: each stage's surrogate improvement must dominate its own KL penalty.

Discussion. Assumption 2 (Stagewise KL Proximity) is motivated by CAIR's reuse of the current policy parameters when the annealing weight is updated (see Algorithm 1). Because the objectives differ only in their scalarization weight, smaller changes in λ may help maintain policy proximity under continuity conditions made precise in Proposition 1. Assumption 3 (Minimum Surrogate Improvement) requires positive surrogate improvement at each stage; increasing the RL weight motivates it, and the proposition identifies a local positive-progress condition under which it holds.

Joint satisfiability of Assumptions 2 and 3. The proximity and improvement bounds cannot be chosen independently: both depend on the policy change between stages. We give sufficient local conditions under which small decreases in λ satisfy both bounds and the dominance condition. Fix $\lambda_k > 0$ and consider a parameter path $\theta^*(\lambda)$ with $\theta^*(\lambda_k) = \theta_k^*$, e.g., a path of stationary points of L_λ^κ followed by warm-started optimization with fixed κ. For sufficiently small $h \geq 0$, let $\pi_k(h) = \pi_{\theta^*(\lambda_k - h)}$, and let $S_k(h)$ and $D_k^{\max}(h)$ denote the surrogate improvement and maximum KL divergence defined above, with π_{k+1}^* replaced by $\pi_k(h)$.

Condition C1 (Differentiable Path and Positive Progress)**.** *The parameter path is differentiable from the left at λ_k, and $S_k(h)$ is right-differentiable at $h = 0$, with $a_k := S_k'(0^+) > 0$.*

Condition C2 (Uniform KL Curvature)**.** *For every state s, the function $K_s(\theta) := D_{\mathrm{KL}}(\pi_k^*(\cdot \mid s) \| \pi_\theta(\cdot \mid s))$ is twice continuously differentiable on an open ball U centered at θ_k^*, with*

$$\sup_{s \in \mathcal{S},\, \theta \in U} \|\nabla_\theta^2 K_s(\theta)\|_{\mathrm{op}} \leq M_k < \infty.$$

Proposition 1 (Local Joint Satisfiability)**.** *Under Conditions C1–C2, if $\lambda_{k+1} = \lambda_k - h$ and $\pi_{k+1}^* = \pi_k(h)$, then for all sufficiently small $h > 0$, the bounds in Assumptions 2–3 and the dominance condition hold at stage k, with $\eta_k = \frac{1}{2} a_k h$ and $\delta_k = b_k h^2$, for some finite $b_k > 0$ independent of h.*

The proof is given in Appendix A.6.4. Along this path, $S_k(h) = a_k h + o(h)$ while $D_k^{\max}(h) = O(h^2)$, so the KL penalty shrinks faster than the positive surrogate improvement. The surrogate and return change differ by $O(h^2)$ (Appendix A.6.4); hence $a_k > 0$ is equivalent to positive first-order return improvement along the path.

Scope. Theorem 1 is conditional on Assumptions 2–3 and the stagewise dominance condition, which neither CAIR nor the SAC (Haarnoja et al., 2018) and PPO (Schulman et al., 2017) solvers enforce by themselves. Distinct deterministic DDPG policies (Lillicrap et al., 2015) have infinite KL divergence at any state where their actions differ, and therefore cannot satisfy Assumption 2. Likewise, Proposition 1 does not establish that its local conditions hold at every CAIR transition, including the finite step that fixes λ at zero once it reaches or falls below ϵ_λ (Section 4.2). The experiments in Section 4 therefore evaluate CAIR beyond the theorem's formal scope. Their changing normalization scales, smoothed auxiliary-weight updates, finite CAGrad solves, and policy optimizers also differ from the fixed-scale, converged-stage, direct-update idealization.

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 can be instantiated with multiple IL–RL pairings. Its empirical performance remains dependent on the selected algorithms and task.

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. **Applicability:** Is CAIR applicable to different combinations of IL and RL algorithms? and **Generality:** Does CAIR provide consistent performance benefits across 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.

- **Gymnasium MuJoCo Locomotion Tasks** (Towers et al., 2024): We evaluate continuous locomotion control using the v5 environments for six tasks: *HalfCheetah*, *Hopper*, *Walker2d*, *Ant*, *Swimmer*, and *Humanoid*.

The benchmark environments have their own native episode horizons: 50 environment steps for the Fetch tasks, 200 for AdroitHammer, 100 for AdroitPen, and 1,000 for the Gymnasium MuJoCo locomotion tasks. For consistency in our experimental reporting and scheduling, however, we define one *episode* as 1,000 environment steps for Fetch and locomotion and 4,000 environment steps for Adroit. Unless explicitly stated otherwise, all episode counts in the experimental sections, figures, tables, and update schedules use this convention.

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 Gymnasium MuJoCo 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. The main manipulation and locomotion evaluations use five and ten random seeds, respectively; where the number of available runs differs in an ablation, the exact sample size is reported in the corresponding table. We report the mean reward across runs with a 1-σ shaded error region. For a horizon

H, AUC@H denotes the mean reward over the corresponding learning curve, AUC@$H = \frac{1}{H}\sum_{t=1}^{H}\bar{R}_t$, where $\bar{R}_t$ is the mean reward across seeds at evaluation point t. AUC is computed from the unsmoothed reward curve; smoothing is used only for visualization.

We additionally report aggregate normalized reward, NormReward $= \frac{R-R_{\min}}{R_{\text{demo}}-R_{\min}}$, using the task-specific reference scores reported in Table 11. For the manipulation tasks, $R_{\min}$ is a task-specific lower reward reference determined from the task reward scale. For the locomotion tasks, $R_{\min}$ is the lowest score observed among the methods in the baseline comparison and is reused for the combinations table; it is therefore dependent on the baseline method set. Values near 1 indicate performance similar to the demonstrator. We use normalized reward as a complementary cross-task summary and retain raw return as the primary task-level measure.

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 applicable to different combinations of IL and RL algorithms and does it provide consistent performance improvement?*

To evaluate the applicability of CAIR across different IL–RL pairings, we instantiate the framework using multiple combinations of Imitation Learning (IL) and Reinforcement Learning (RL) algorithms. We pair two common IL approaches: (1) **Behavioral Cloning (BC)** (Bain & Sammut, 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 $a_D = \pi_D(s)$:

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

After the imitation-only warm start of $W = 40$ episodes, CAIR begins adapting the IL weight. The auxiliary weight λ_{opt} is updated by Adam at every policy-gradient update, while the applied weight λ is refreshed from λ_{opt} every T_λ episodes, with $T_\lambda = 10$ for manipulation and $T_\lambda = 5$ for locomotion, and is held fixed between refreshes (Line 24 of Algorithm 1). The auxiliary-weight Adam learning rate is $\eta_\lambda = 10^{-4}$, the cosine smoothing coefficient is $\beta = 0.9$, and the persistent multiplier uses $\mu_0 = 1$ and $\eta_\mu = 1$. The default CAGrad deviation parameter is $c = 0.7$; explicitly specified ablation settings override this value. These settings apply across IL–RL pairings and domains unless stated otherwise. When the applied weight satisfies $\lambda \le \epsilon_\lambda = 0.05$, it is fixed at zero permanently and subsequent policy updates use the RL gradient alone (Line 8 of Algorithm 1).

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 (8.07). BC+PPO is the weakest pairing,

[1] `https://github.com/ParanoidAndroid96/CAIR`

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 fixed task-specific lower reward reference reported in Table 11; 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 methods not marked with †, AUC determines the highest-ranked method, which is bolded. The final rows report average NormReward across tasks.

Task	Pairing	Mean	± SD	AUC	NormReward
FetchPickAndPlace (10,000k steps)	BC+DDPG	-13.18	± 4.26	-18.03	2.45
	BC+SAC	-19.52	± 4.88	-29.78	2.03
	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
FetchPush (10,000k steps)	BC+DDPG	†-13.57	± 2.2	-18.12	1.82
	BC+SAC	-10.39	± 1.8	-20.64	1.98
	BC+PPO	-26.95	± 6.02	-28.74	1.15
	DAgger+DDPG	-12.05	± 2.18	-17.18	1.90
	DAgger+SAC	**-9.80**	**± 1.23**	**-15.29**	**2.01**
	DAgger+PPO	†-19.10	± 2.98	-22.02	1.55
FetchSlide (10,000k steps)	BC+DDPG	-38.74	± 3.38	-38.78	2.25
	BC+SAC	**-35.55**	**± 4.83**	**-37.15**	**2.89**
	BC+PPO	†-43.45	± 2.43	-44.03	1.31
	DAgger+DDPG	-37.84	± 3.69	-40.38	2.43
	DAgger+SAC	-36.77	± 2.72	-41.64	2.65
	DAgger+PPO	-38.24	± 2.35	-40.97	2.35
AdroitHammer (3,000k steps)	BC+DDPG	1,354	± 287	718	29.87
	BC+SAC	**1,426**	**± 356**	**1,024**	**31.43**
	BC+PPO	†6	± 0	-1	0.57
	DAgger+DDPG	609	± 666	277	13.67
	DAgger+SAC	650	± 751	338	14.57
	DAgger+PPO	809	± 412	471	18.02
AdroitPen (8,000k steps)	BC+DDPG	†2,593	± 411	1,500	0.74
	BC+SAC	**7,041**	**± 744**	**6,419**	**2.01**
	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
Average Normalized Reward Across Tasks	BC+DDPG				7.43
	BC+SAC				**8.07**
	BC+PPO				1.11
	DAgger+DDPG				4.27
	DAgger+SAC				4.53
	DAgger+PPO				5.08

failing to match demonstrator performance on *AdroitHandHammerSparse-v1* and *AdroitHandPenSparse-v1* (Normalized Reward < 1.0), while BC+DDPG, DAgger+DDPG, and DAgger+SAC each fall short of the demonstrator on *AdroitHandPenSparse-v1*. Overall, SAC-based variants achieve the strongest performance across the manipulation tasks.

Locomotion Tasks. Table 2 summarizes similar measurements on the Gymnasium MuJoCo locomotion tasks. The corresponding learning curves are provided in Figure 6 in Appendix A.4. In five of the six tasks, at least one IL–RL pairing matches or exceeds the reported demonstrator return (Normalized Reward ≥ 1.00); on the remaining task, *HalfCheetah-v5*, the highest mean normalized reward is 0.96. 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

Table 2: Learning performance of CAIR instantiated with different combinations of IL algorithms (BC, DAgger) and RL algorithms (SAC, DDPG, PPO) on Gymnasium MuJoCo 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 task-specific lower reference reported in Table 11, taken from the baseline comparison and reused for the combinations table; 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 methods not marked with †, 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	**3,668**	**± 902**	**1,886**	**1.04**
	BC+PPO	†1,270	± 442	1,233	0.67
	DAgger+DDPG	†1,253	± 681	1,500	0.66
	DAgger+SAC	†2,275	± 1,066	1,881	0.82
	DAgger+PPO	†1,976	± 863	1,920	0.78
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				**0.97**
	DAgger+PPO				0.69

highest-ranked performance on *Walker2d-v5*. BC+SAC achieves the highest-ranked performance on *Ant-v5*. Consistent with these task-level results, DAgger+SAC achieves the highest average normalized reward across

tasks (0.97), 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, no single IL–RL pairing consistently achieves the strongest performance across all benchmark tasks. 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. We study sensitivity to three hyperparameters: (i) the CAGrad deviation parameter c; (ii) the interval controlling when the applied weight is refreshed from the auxiliary weight; and (iii) the learning rate η_λ used for updating λ_{opt}.

- **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 normalized mixture v_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:** After the warm start, the applied weight is refreshed from the auxiliary weight every tenth episode in manipulation or fifth episode in locomotion. It is held fixed between refreshes while the auxiliary weight continues to adapt at each policy-gradient update. This interval therefore controls a trade-off between responsiveness of the applied weight and stability of the policy objective. Its impact is detailed in Figures 12 and 13 (Appendix A.3).

- **λ Learning Rate (η_λ):** The auxiliary-weight learning rate controls how quickly the weight responds to smoothed gradient alignment. Figure 11 compares $\eta_\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}\}$ on the Fetch tasks; in these experiments, both extremes reduce sample efficiency relative to 10^{-4}. This task-specific sensitivity supports controlling the transition rate, consistent with prior findings that excessively rapid transitions from IL to RL can cause performance collapse (Uchendu et al., 2022).

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, the normalization and cosine smoothing used by CAIR help 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:

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 fixed task-specific lower reward reference reported in Table 11; 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 methods not marked with †, AUC determines the highest-ranked method, which is bolded. The final rows report average NormReward across tasks.

Task	Algorithm	Mean	± SD	AUC	NormReward
FetchPickAndPlace (15,000k steps)	**BC+DDPG**	**-14.07**	**± 5.50**	**-16.52**	**2.40**
	BC+SAC	-14.88	± 3.88	-26.04	2.34
	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
FetchPush (10,000k steps)	**BC+DDPG**	**-13.57**	**± 2.2**	**-18.12**	**1.82**
	BC+SAC	-10.39	± 1.8	-20.64	1.98
	RP	†-46.90	± 2.0	-46.85	0.16
	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
FetchSlide (18,500k steps)	BC+DDPG	†-38.84	± 2.86	-38.75	2.23
	BC+SAC	**-33.02**	**± 3.34**	**-36.29**	**3.40**
	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	1.44
AdroitHammer (3,000k steps)	BC+DDPG	1,354	± 287	718	29.87
	BC+SAC	1,426	± 356	1,024	31.43
	RP	†-19	± 1	-19	0.02
	HER+Demos	†14	± 37	13	0.74
	SAIL	†10	± 30	12	0.65
	JumpStart	†-15	± 7	-12	0.11
	RLPD	**1,358**	**± 596**	**1,212**	**29.96**
Pen (8,000k steps)	BC+DDPG	†2,593	± 411	1,500	0.74
	BC+SAC	7,041	± 744	6,419	2.01
	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**
Average Normalized Reward Across Tasks	BC+DDPG				7.41
	BC+SAC				**8.23**
	RP				1.00
	HER+Demos				1.07
	SAIL				0.34
	JumpStart				1.18
	RLPD				7.11

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).

Table 4: Learning performance of CAIR (BC+SAC and DAgger+DDPG) compared to hybrid IL–RL baselines on Gymnasium MuJoCo 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 task-specific lower reference reported in Table 11, taken from the baseline comparison and reused for the combinations table; 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 methods not marked with †, 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	0.00
	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	†1,253	± 681	1,500	0.66
	BC+SAC	**3,668**	**± 902**	**1,886**	**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	0.20
	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

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., 2022) 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.

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

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 achieves the highest average normalized reward across the locomotion benchmark tasks, BC+SAC is included to enable a more direct comparison with prior hybrid IL–RL baselines, which primarily rely on offline demonstrations rather than interactive demonstrators.

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 *FetchPickAndPlace-v1*, *FetchPush-v1*, and *FetchSlide-v1*. On *AdroitHandHammerSparse-v1*, BC+SAC and RLPD have statistically comparable final return, with BC+SAC having the higher mean final return and RLPD the higher 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 (7.11) 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 Gymnasium MuJoCo 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*, RLPD achieves higher AUC and significantly higher final return than both evaluated CAIR variants. RLPD also achieves the strongest overall performance on *Walker2d-v5* and *Humanoid-v5*. As a complementary aggregate summary, 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 loss annealing. Specifically, we compare variants with and without CAGrad under matched annealing configurations. Within each comparison, the underlying IL–RL learner, annealing type, and, for fixed schedules, the step size Δ are held constant. The analysis includes both the proposed adaptive schedule and fixed schedules with $\Delta \in \{0.01, 0.05, 0.1, 0.15\}$. This isolates the effect of conflict-aware gradient manipulation for a given annealing configuration while also evaluating whether its effect is consistent across annealing schedules. At each recorded update, we define the worst-case cosine alignment as $c_{\text{worst}} = \min\{\cos(d, g_{\text{IL}}), \cos(d, g_{\text{RL}})\}$, where d is the direction supplied to the policy optimizer. This quantity measures alignment with the less-aligned of the two objectives. For each configuration, the reported mean worst-case cosine is averaged over its recorded gradient updates. For a matched comparison, we report

$$\Delta c_{\text{worst}} = c_{\text{worst}}^{+\text{CAGrad}} - c_{\text{worst}}^{\text{No CAGrad}}$$

so positive values indicate improved alignment with the less-aligned objective gradient.

The learning-curve ablations include DAgger+DDPG and DAgger+SAC, while Tables 13 and 15 report controlled matched comparisons for DAgger+DDPG. Final reward is compared using a two-sided Welch's t-test. When one configuration is significantly worse, its matched alternative is treated as favored. When the test does not distinguish the two configurations, they are treated as statistically comparable, and AUC is used only to determine which configuration is favored for the summary counts.

Manipulation Tasks. The manipulation results are presented in Figure 9 and Table 13. Across the 15 matched adaptive and fixed-schedule comparisons, incorporating CAGrad is favored under this evaluation rule in 11 configurations, statistically comparable but not AUC-favored in two, and significantly worse in two. Thus, CAGrad is favored or statistically comparable in 13 of 15 comparisons.

The mean change in worst-case cosine is positive for each Fetch task when aggregated across its five annealing configurations, with an overall paired increase of 0.072 ± 0.070. This indicates that CAGrad generally shifts the joint update toward better alignment with the less-aligned objective gradient. However, the two configurations in which CAGrad obtains lower reward despite a positive cosine improvement show that improved worst-case alignment does not necessarily translate into improved policy return.

Locomotion Tasks. Figure 10 and Table 15 present the corresponding analysis on the Gymnasium MuJoCo locomotion tasks. Across 30 matched adaptive and fixed-schedule comparisons, CAGrad is favored under this evaluation rule in 15 configurations, statistically comparable but not AUC-favored in five, and significantly worse in ten. It is therefore favored or statistically comparable in 20 of 30 comparisons.

CAGrad improves mean worst-case cosine in all 30 locomotion comparisons, with an average paired increase of 0.202 ± 0.133. This provides consistent evidence that CAGrad alters the update in the intended conflict-averse direction. Nevertheless, its less uniform effect on final reward and AUC shows that improved gradient alignment alone is not sufficient to guarantee better learning performance for every task and annealing schedule.

Overall Assessment. Overall, the ablation results provide qualified support for Research Question 3. CAGrad consistently improves the measured worst-case gradient alignment and is frequently favored or statistically comparable in terms of reward. Its performance benefit is nevertheless task- and schedule-dependent. We therefore interpret CAGrad as an effective mechanism for mitigating IL–RL gradient conflict, rather than as a guarantee of improved sample efficiency or asymptotic performance in every configuration.

4.6 Ablation Study: Impact of Adaptive Annealing

To study the effect of the proposed adaptive annealing strategy, we compare adaptive and fixed-step annealing under matched gradient-update settings within each benchmark suite. CAGrad is enabled for all manipulation variants and disabled for all locomotion variants (Tables 12 and 14). The fixed schedules use $\lambda \leftarrow \lambda - \Delta$ for $\Delta \in \{0.01, 0.05, 0.1, 0.15\}$. These comparisons isolate the effect of the annealing rule within each suite; the complementary CAGrad ablation tables compare adaptive annealing with and without CAGrad. Final reward

is compared using a two-sided Welch's t-test. Schedules significantly worse than the highest-mean schedule are treated as underperforming; among the remaining statistically comparable schedules, AUC determines which schedule is the most favored.

Manipulation Tasks. Figure 14 and Table 12 show that adaptive annealing is favored under this evaluation rule in one of the three Fetch tasks and is statistically comparable, but not AUC-favored, in the remaining two. It is not significantly worse in any manipulation task. The favored schedule varies across tasks: adaptive annealing, $\Delta = 0.01$, and $\Delta = 0.05$ are each favored once, while neither $\Delta = 0.1$ nor $\Delta = 0.15$ is favored. More aggressive schedules also degrade performance in several cases, particularly at $\Delta = 0.15$.

These results indicate that no single fixed annealing rate is consistently favored across the Fetch tasks. Adaptive annealing therefore avoids the need to choose Δ separately for each task while remaining favored or statistically comparable to the strongest tested fixed schedule in all three tasks.

Locomotion Tasks. Figure 15 and Table 14 present the corresponding analysis on the Gymnasium MuJoCo locomotion tasks. Adaptive annealing is favored under this evaluation rule in two of six tasks, is statistically comparable but not AUC-favored in two others, and is significantly worse than the highest-mean fixed schedule in the remaining two. Among the four tasks in which a fixed schedule is favored, $\Delta = 0.01$ and $\Delta = 0.05$ are each favored twice, whereas neither $\Delta = 0.1$ nor $\Delta = 0.15$ is favored.

The learning curves further show that larger fixed rates can remove imitation guidance too quickly, producing a pronounced decline as λ approaches zero. Adaptive annealing does not eliminate such failures in every task, but it remains favored or statistically comparable in four of six domains without requiring a task-specific choice of Δ.

Overall Assessment. Across the nine manipulation and locomotion tasks, adaptive annealing is favored in three, statistically comparable but not AUC-favored in four, and significantly worse in two. Among the six tasks in which a fixed schedule is favored, $\Delta = 0.01$ is favored three times and $\Delta = 0.05$ three times, while the larger rates are never favored. Thus, the results do not support the claim that adaptive annealing uniformly outperforms an oracle-tuned fixed schedule. Instead, they show that it provides a competitive, data-dependent alternative that avoids the need to choose a fixed annealing rate separately for each task.

5 Conclusions

We introduced *Conflict-Averse IL-RL* (CAIR), a framework for combining Imitation Learning (IL) and Reinforcement Learning (RL) while addressing gradient conflict as one potential contributor to negative transfer during the IL-to-RL transition. CAIR adaptively adjusts the relative weights of normalized IL and RL gradients using smoothed cosine alignment and incorporates Conflict-Averse Gradient Descent (CAGrad) to balance their normalized predicted first-order decreases. Together, these mechanisms are designed to mitigate harmful gradient interference as optimization shifts from imitation-driven initialization to reward-driven learning.

CAIR admits a conditional monotonic non-deterioration result for the idealized stagewise policy sequence satisfying the stated KL-proximity and minimum surrogate-improvement assumptions together with the stagewise dominance condition. We also identify local conditions under which these requirements hold jointly for sufficiently small annealing steps. CAIR does not itself enforce these conditions, and the formal result therefore does not extend directly to the SAC, PPO, and DDPG implementations. Empirically, we evaluate CAIR with PPO, SAC, and DDPG across five sparse-reward manipulation tasks and six dense-reward locomotion tasks. Relative to the evaluated hybrid IL–RL baselines, at least one reported CAIR variant achieves the highest-ranked result in three of five manipulation tasks and three of six locomotion tasks. CAIR is competitive with the strongest baseline on the remaining manipulation tasks, while RLPD performs better on *Hopper-v5*, *Walker2d-v5*, and *Humanoid-v5*. These results show that CAIR is competitive across both benchmark suites, although its relative performance remains task- and pairing-dependent.

Our ablation studies clarify the roles of the two CAIR components. CAGrad improves the measured worst-case gradient alignment, but its effect on final return and AUC depends on the task and annealing schedule, showing that improved alignment alone does not guarantee better policy performance. Adaptive annealing

provides a competitive, data-dependent alternative to fixed schedules by avoiding task-specific selection of a fixed annealing rate while remaining favored or statistically comparable to the strongest tested fixed schedule in the majority of the evaluated tasks.

A practical limitation of CAIR is its sensitivity to the compatibility of the underlying IL and RL algorithms. Performance varies substantially across IL–RL pairings, and no single pairing consistently achieves the strongest result across all tasks. Pairing selection therefore remains domain-dependent. Future work will investigate methods for characterizing and improving compatibility across IL–RL pairings and extend CAIR to multimodal demonstration data (Dey & Sharon, 2024; Zhang & Sharon, 2025).

Acknowledgements

The research was conducted in the PiStar AI and Optimization Lab at Texas A&M University. PiStar is supported in part by the NSF (IIS-2238979).

A portion of this work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (FAIN-2019844, NRT-2125858, OIC-2535195), ONR (N00014-24-1-2550), ARO (W911NF-17-2-0181, W911NF-23-2-0004, W911NF-25-1-0065), Lockheed Martin, Lyda Hill, and Good Systems. Peter Stone serves as the Chief Scientist of Sony CTC and receives financial compensation for that role. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

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.

Sheelabhadra Dey and Guni Sharon. Comparing deterministic and soft policy gradients for optimizing gaussian mixture actors. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=qS9pPu80Dt.

Sheelabhadra Dey, Sumedh Pendurkar, Guni Sharon, and Josiah P. Hanna. A joint imitation-reinforcement learning framework for reduced baseline regret. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3485–3491. IEEE Press, 2021. doi: 10.1109/IROS51168.2021.9636294.

Sheelabhadra Dey, James Ault, and Guni Sharon. Continual optimistic initialization for value-based reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 453–462. International Foundation for Autonomous Agents and Multiagent Systems, 2024. doi: 10.5555/3635637.3662895.

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.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

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.

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.

Hamid Mirzaei, Guni Sharon, Stephen Boyles, Tony Givargis, and Peter Stone. Enhanced delta-tolling: Traffic optimization via policy gradient reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 47–52, 2018. doi: 10.1109/ITSC.2018.8569737.

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 https://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 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 2015. 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.

Guni Sharon. Alleviating road traffic congestion with artificial intelligence. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 4965–4969. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/704. Early Career.

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.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

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.

Wei Zhang and Guni Sharon. AEAP: A reinforcement learning actor ensemble algorithm with adaptive pruning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=I5ymMVdmaR.

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. 2022. 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 (Gymnasium 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 environment 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.
- **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. 0 if the object is within 5cm of the target, -1 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.

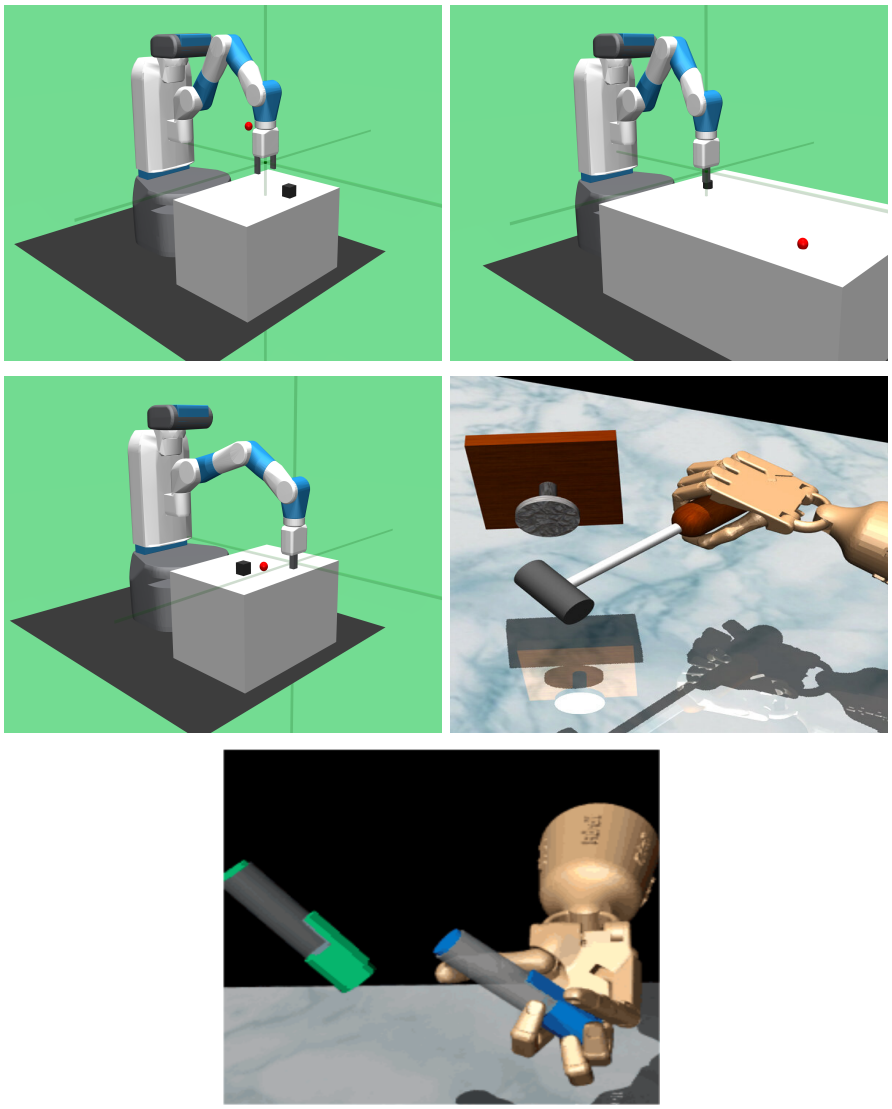


Figure 4: Visualizations of the five MuJoCo sparse-reward tasks used in our experiments. Top-left: FetchPickAndPlace-v1 task; Top-right: FetchSlide-v1 task; Middle-left: FetchPush-v1 task; Middle-right: AdroitHandHammerSparse-v1; Bottom: AdroitHandPenSparse-v1

- **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-v1* (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 Gymnasium MuJoCo Locomotion Tasks

We additionally evaluate CAIR on six continuous locomotion tasks from the Gymnasium MuJoCo benchmark: *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 counts follow the convention defined in Section 4.1.

Locomotion Tasks: For the Gymnasium MuJoCo 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.

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

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

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

Task	**Network**	γ	λ_{GAE}	**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 8: Hyperparameters for CAIR (BC/DAgger + SAC) on locomotion tasks

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

A.3 Sensitivity Analysis

We provide additional empirical analysis of the CAIR framework components.

- **CAGrad Ablation (Fig. 9):** Evaluates how CAGrad affects gradient alignment and return under matched annealing configurations.
- **Sensitivity to η_λ (Fig. 11):** Compares the effect of the auxiliary-weight learning rate on sample efficiency in the Fetch tasks.

Table 9: Hyperparameters for CAIR (BC/DAgger + DDPG) 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 10: Hyperparameters for CAIR (BC/DAgger + PPO) on locomotion tasks

Task	**Network**	γ	λ_{GAE}	**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

- **Update Interval (Fig. 12):** Highlights the trade-off between responsiveness and stability in λ updates.

A.4 Conflict-Averse IL to RL Algorithm

CAIR maintains an auxiliary weight optimized at each policy-gradient update and an applied weight used to combine individually normalized IL and RL gradients. Algorithm 1 separates these updates from the learner's data collection, critic updates, and minibatch schedule. These learner-specific operations follow SAC, DDPG, or PPO; the IL objective is supplied by BC or DAgger. Tables 5, 6, 7, 8, 9, and 10 report the underlying RL hyperparameters. Remaining RL hyperparameters follow Stable-Baselines3 (Raffin et al., 2021). The PPO tables' λ_{GAE} denotes generalized advantage estimation, independently of CAIR's IL weight.

After the warm start, whenever the episode counter $e \geq W$ is a multiple of T_λ, the applied weight is copied from the auxiliary weight after the learner training call. Between these refreshes it is held fixed while the auxiliary weight continues to adapt. Pure IL and post-completion RL updates use their unnormalized loss gradients; mixed-phase updates use the normalized mixture, with CAGrad where enabled.

CAGrad is solved numerically through its simplex dual, minimizing $\langle g_w, v_\lambda \rangle + c\|v_\lambda\|\|g_w\|$, where $g_w = \sum_i w_i \tilde{g}_i$ and w is parameterized by a softmax. The solver takes 20 SGD steps with learning rate 2 and momentum 0.5, starting from equal logits and retaining the iterate with the smallest evaluated dual objective. It reconstructs $d = v_\lambda + c\|v_\lambda\| g_w / \|g_w\|$ when the denominator is nonzero. This finite solve is not assumed to attain the exact maximizer in the alignment results. The resulting vector is passed as the gradient input to the policy optimizer; pure IL and post-completion RL updates use their unnormalized loss gradients.

Algorithm 2 holds the loss gradients fixed during each auxiliary-weight update. The operator sg stops differentiation through the previous cosine histories and loss gradients. The auxiliary weight is clipped to $[0, 1]$, while the cosine histories, multiplier, and Adam state persist across updates. The multiplier update increases the penalty under negative smoothed IL alignment and decreases it under positive alignment; together with the hinge-penalized objective, it pursues rather than exactly solves Equation 2.

The cosine denominator is $\max(\|\tilde{g}_i\|, 10^{-8}) \max(\|v_\lambda\|, 10^{-8})$, assigning zero similarity to $v_\lambda = 0$. When $q = 1$, every weight is optimal. For exactly opposite directions ($q = -1$), the extended criterion selects $\lambda^\star = 1/2$. For $-1 < q < 0$ and $0 < \epsilon_\lambda < 1$, $\lambda^\star \leq \epsilon_\lambda$ exactly when $-q \leq \epsilon_\lambda / (1 - \epsilon_\lambda)$. This characterizes the instantaneous target; smoothed stochastic adaptation need not attain it at the applied weight.

Algorithm 1 Conflict-Averse IL to RL (CAIR)

Input: Policy parameters θ_0, demonstrations D, policy optimizer $\texttt{Opt}$, c, W, T_λ, ϵ_λ, η_λ, η_μ, β, μ_0
Initialize $\theta \leftarrow \theta_0$, $\lambda \leftarrow 1$, $\lambda_{\text{opt}} \leftarrow 1$, $\mu \leftarrow \mu_0$, $s_{\text{IL}} \leftarrow 1$, $s_{\text{RL}} \leftarrow 0$, persistent auxiliary-weight Adam state A, episode counter $e \leftarrow 0$
for each learner training call **do**
  Collect data and update the episode counter e according to the underlying learner's training schedule
  for each policy-gradient update **do**
    Compute $g_{\text{RL}} \leftarrow \nabla_\theta L_{\text{RL}}$
    if $\lambda \leq \epsilon_\lambda$ **then**
      $\lambda \leftarrow 0$; $d \leftarrow g_{\text{RL}}$
    else
      Compute $g_{\text{IL}} \leftarrow \nabla_\theta L_{\text{IL}}$
      if $e < W$ or $\lambda = 1$ **then**
        $d \leftarrow g_{\text{IL}}$
      else
        $\tilde{g}_i \leftarrow g_i / \|g_i\|$, $i \in \{\text{IL}, \text{RL}\}$
        $v_\lambda \leftarrow \lambda \tilde{g}_{\text{IL}} + (1-\lambda)\tilde{g}_{\text{RL}}$
        $d \leftarrow \textbf{CAGrad}(v_\lambda, [\tilde{g}_{\text{IL}}, \tilde{g}_{\text{RL}}], c)$
      end if
      if $e \geq W$ **then**
        Update $(\lambda_{\text{opt}}, \mu, s_{\text{IL}}, s_{\text{RL}}, A)$ using Algorithm 2
      end if
    end if
    $\theta \leftarrow \texttt{Opt}(\theta, d)$
  end for
  if $\lambda > 0$, $e \geq W$, and $e \bmod T_\lambda = 0$ **then**
    $\lambda \leftarrow \lambda_{\text{opt}}$
  end if
end for

Algorithm 2 One auxiliary-weight update: OptimizeLambda

Input: Nonzero unweighted gradients $g_{\text{IL}}, g_{\text{RL}}$; persistent state $(\lambda_{\text{opt}}, \mu, s_{\text{IL}}, s_{\text{RL}}, A)$; η_λ, η_μ, β
$\tilde{g}_i \leftarrow \text{sg}(g_i / \|g_i\|)$, $i \in \{\text{IL}, \text{RL}\}$
$v \leftarrow \lambda_{\text{opt}} \tilde{g}_{\text{IL}} + (1-\lambda_{\text{opt}})\tilde{g}_{\text{RL}}$
$s_i \leftarrow \beta\, \text{sg}(s_i) + (1-\beta)\cos(\tilde{g}_i, v)$, $i \in \{\text{IL}, \text{RL}\}$
$\mathcal{J}_\lambda \leftarrow -s_{\text{RL}} + \mu \max(0, -s_{\text{IL}})$
$(\lambda_{\text{opt}}, A) \leftarrow \texttt{AdamStep}(\lambda_{\text{opt}}, \nabla_{\lambda_{\text{opt}}} \mathcal{J}_\lambda, A, \eta_\lambda)$
$\mu \leftarrow \max\{0, \mu - \eta_\mu\, \text{sg}(s_{\text{IL}})\}$
$\lambda_{\text{opt}} \leftarrow \text{clip}(\lambda_{\text{opt}}, 0, 1)$
return $(\lambda_{\text{opt}}, \mu, s_{\text{IL}}, s_{\text{RL}}, A)$

Table 11: Task-specific reference scores used to compute normalized reward, NormReward $= \frac{R - R_{\min}}{R_{\text{demo}} - R_{\min}}$. Here, $R_{\min}$ denotes the task-specific normalization reference and R_{demo} denotes the corresponding demonstrator return.

Suite	Task	$R_{\min}$	R_{demo}
Manipulation	*FetchPickAndPlace-v1*	-50	-35
	FetchPush-v1	-50	-30
	FetchSlide-v1	-50	-45
	AdroitHandHammerSparse-v1	-20	26
	AdroitHandPenSparse-v1	0	3,500
Locomotion	*HalfCheetah-v5*	-557	10,000
	Hopper-v5	976	3,350
	Walker2d-v5	-187	4,100
	Swimmer-v5	-0.2	330
	Ant-v5	-3,001	3,400
	Humanoid-v5	4,177	5,250

Table 12: Adaptive-annealing ablation on Fetch manipulation tasks with DAgger+DDPG and CAGrad enabled in all variants. Values report final reward (mean ± standard deviation) and area under the learning curve over the first 8,000 episodes (AUC@8k). Methods marked with † are significantly worse than the highest-mean schedule for the same task according to a two-sided Welch's t-test ($p < 0.05$; $n = 5$ per configuration). Among the remaining schedules, the method with the highest AUC@8k is bolded.

Task	Annealing	Δ	Mean	± SD	AUC@8k
FetchPickAndPlace (8k episodes)	**Adaptive**	–	**-20.11**	± **12.20**	**-30.52**
	Fixed	0.01	-16.96	± 13.13	-31.81
	Fixed	0.05	-28.87	± 27.18	-39.07
	Fixed	0.10	-36.06	± 17.45	-45.99
	Fixed	0.15	-35.70	± 17.15	-46.33
FetchPush (8k episodes)	Adaptive	–	-10.70	± 2.24	-20.22
	Fixed	**0.01**	**-9.49**	± **1.07**	**-16.17**
	Fixed	0.05	-9.72	± 1.22	-18.89
	Fixed	0.10	-9.91	± 1.25	-21.81
	Fixed	0.15	†-46.42	± 2.19	-46.61
FetchSlide (8k episodes)	Adaptive	–	-36.54	± 5.34	-37.73
	Fixed	0.01	-38.81	± 5.38	-43.79
	Fixed	**0.05**	**-32.94**	± **1.85**	**-35.63**
	Fixed	0.10	†-49.35	± 0.00	-48.18
	Fixed	0.15	†-44.87	± 3.99	-48.49

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 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.

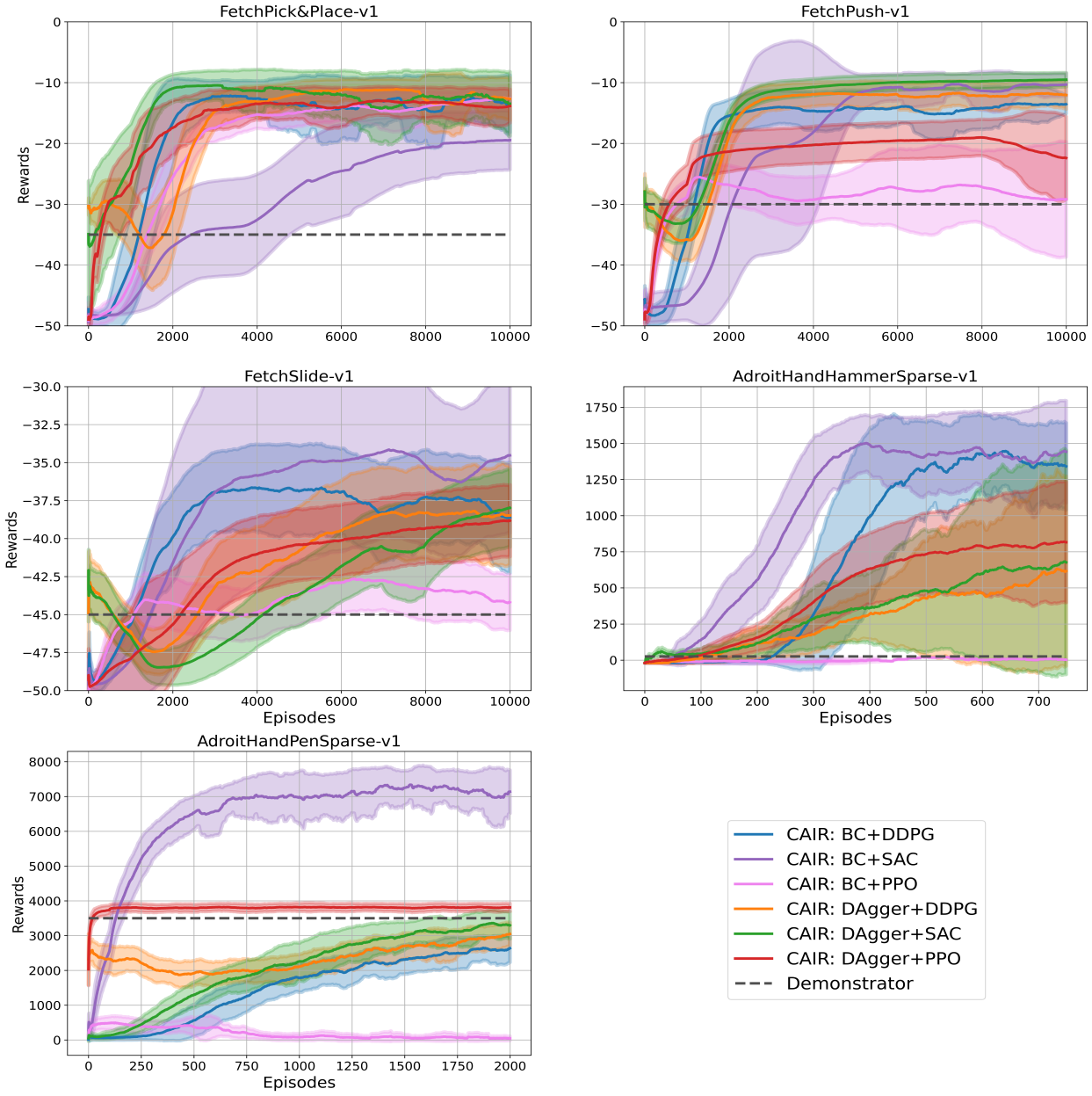


Figure 5: Learning curves for CAIR instantiated with different combinations of IL algorithms (BC, DAgger) and RL algorithms (SAC, DDPG, PPO). Positive learning trends are observed for all evaluated IL–RL pairings except BC+PPO on *AdroitHandHammerSparse-v1* and *AdroitHandPenSparse-v1*.

The HER buffer is initialized with demonstration data. All remaining hyperparameters follow the default HER implementation.

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 (Fujimoto et al., 2018) as the policy optimization algorithm. All hyperparameters, network architectures, and buffer configurations are taken directly from the authors' released implementation (Zhu et al., 2022).

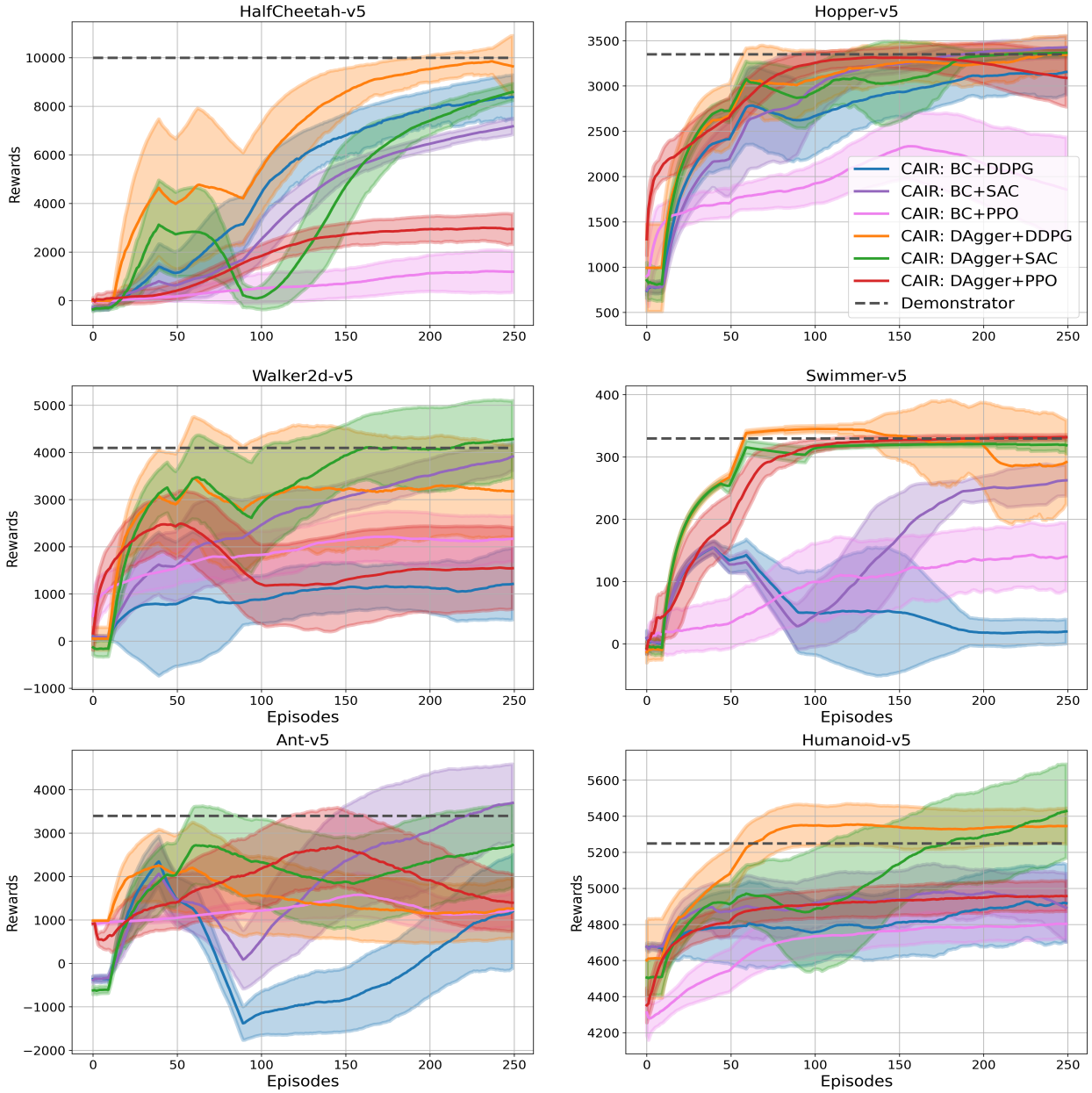


Figure 6: Learning curves for CAIR instantiated with different combinations of IL algorithms (BC, DAgger) and RL algorithms (SAC, DDPG, PPO) on Gymnasium MuJoCo locomotion tasks. Differences in learning performance are observed across IL–RL pairings, highlighting the impact of the underlying imitation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

A.5.4 Jump-Start Reinforcement Learning (JSRL)

JSRL initializes training using a guide policy that steers the agent toward advantageous states at the start of each environment 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 (native environment episode length 50) and 150 for AdroitHandHammerSparse (native environment episode length 200).

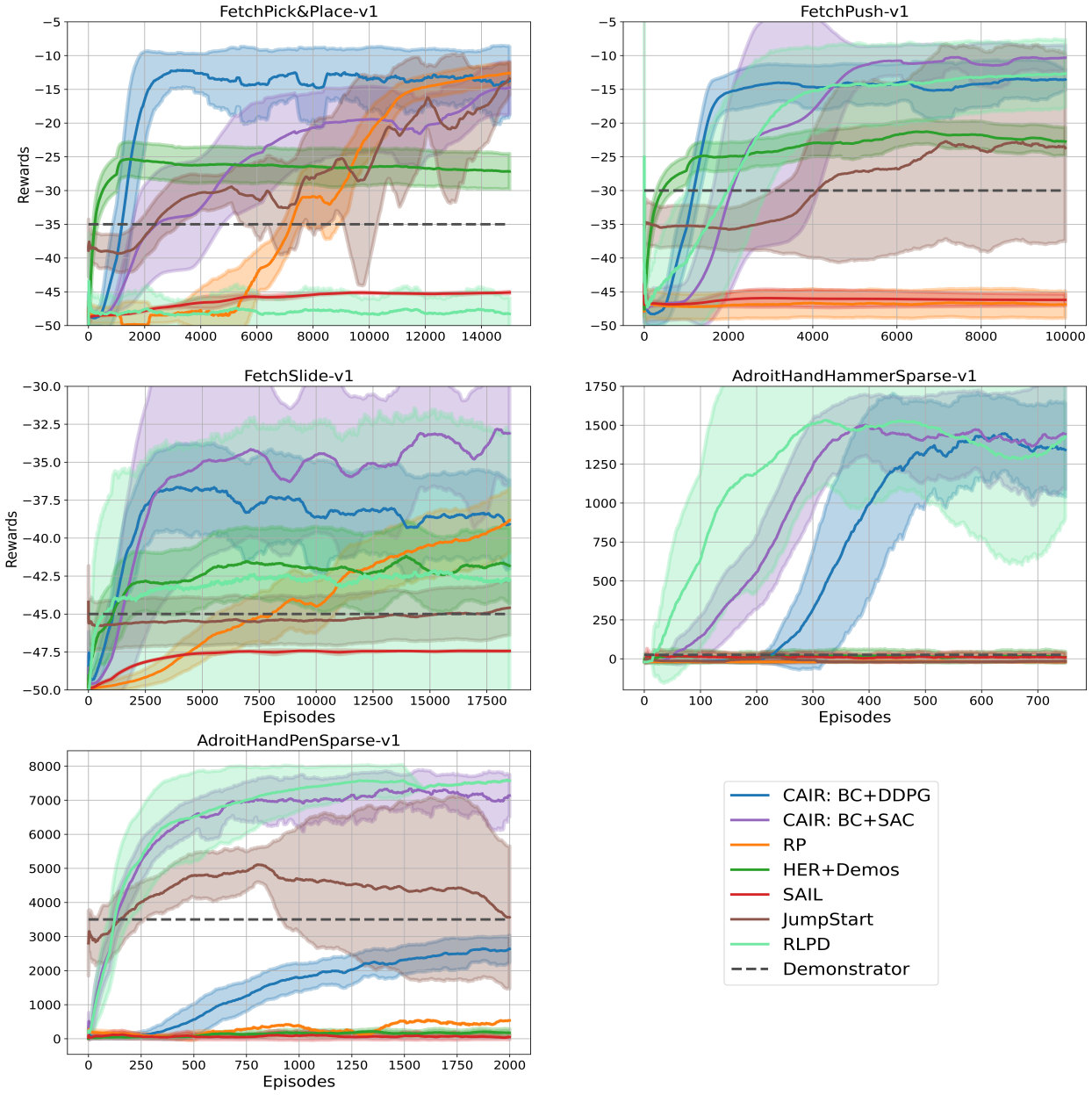


Figure 7: Learning trends for CAIR (BC+SAC and BC+DDPG) compared to hybrid-RL baselines on sparse-reward robotic manipulation tasks.

A.6 Theoretical Details

A.6.1 CAGrad Update Conditions

For a fixed stage, write $F = L^{\kappa}_{\lambda_k}$ with constant $\kappa_{\mathrm{IL}}, \kappa_{\mathrm{RL}} > 0$. Under Assumption 1, F is H_k-smooth, where $H_k = H[\lambda_k/\kappa_{\mathrm{IL}} + (1-\lambda_k)/\kappa_{\mathrm{RL}}]$. The idealized updates are $\theta_{t+1} = \theta_t - \alpha d_t$, with exact gradients $G_t = \nabla F(\theta_t)$, $\|d_t - G_t\| \le c\|G_t\|$, $0 \le c < 1$, and a fixed $0 < \alpha \le 1/H_k$. This is the deviation-ball descent argument underlying CAGrad (Liu et al., 2021), applied here to the weighted center G_t.

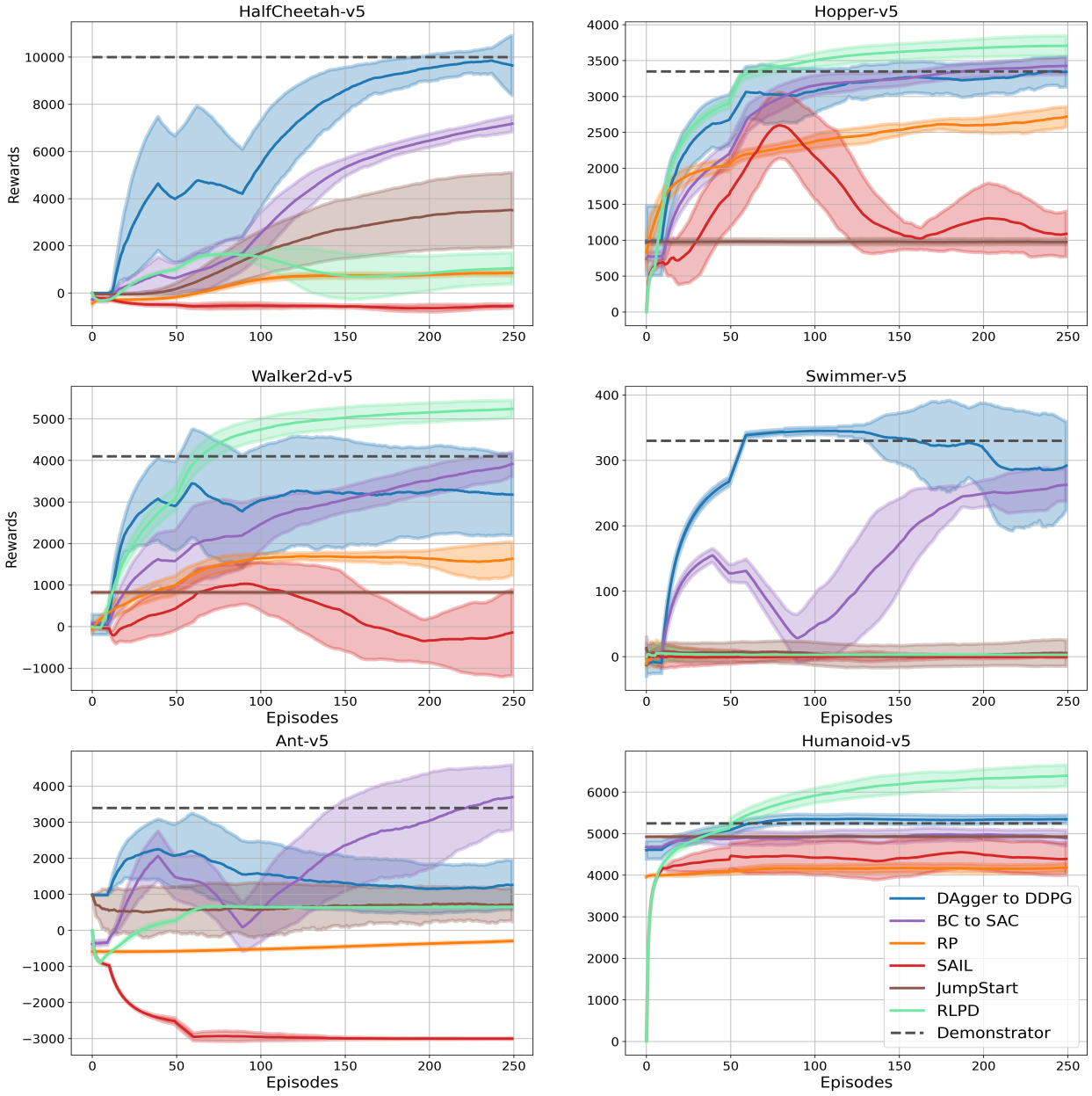


Figure 8: Learning trends for CAIR (BC+SAC and DAgger+DDPG) compared to hybrid-RL baselines on Gymnasium MuJoCo locomotion tasks.

Writing $e_t = d_t - G_t$, smoothness gives

$$\begin{aligned} F(\theta_t - \alpha d_t) &\le F(\theta_t) - \alpha\langle G_t, d_t\rangle + \frac{\alpha}{2}\|d_t\|^2 \\ &= F(\theta_t) - \frac{\alpha}{2}\big(\|G_t\|^2 - \|e_t\|^2\big) \\ &\le F(\theta_t) - \frac{\alpha}{2}(1-c^2)\|G_t\|^2. \end{aligned}$$

If $F_{\inf}$ is a lower bound, summing over T updates yields

$$\min_{0\le t<T} \|\nabla F(\theta_t)\|^2 \le \frac{2[F(\theta_0) - F_{\inf}]}{\alpha(1-c^2)T} \longrightarrow 0.$$

Table 13: CAGrad performance ablation on Fetch manipulation tasks using matched no-CAGrad and CAGrad configurations. For each row, DAgger+DDPG, the annealing type, and the fixed rate Δ are held constant. Values report final reward (mean ± standard deviation) and area under the learning curve over the first 8,000 episodes (AUC@8k). Within each matched pair, the preferred configuration is bolded. A gray result marked with † is significantly worse than its matched alternative according to a two-sided Welch's t-test ($p < 0.05$; $n = 5$ per configuration). When neither configuration fails the test, both are treated as comparable and the configuration with the higher AUC@8k is bolded. The final column reports the paired improvement in mean worst-case cosine, $\Delta c_{\text{worst}} = c_{\text{worst}}^{+\text{CAGrad}} - c_{\text{worst}}^{\text{No CAGrad}}$, as mean ± sample standard deviation across the adaptive schedule and four fixed rates for each task.

			No CAGrad			+CAGrad			
Task	Annealing	Δ	Mean	± SD	AUC@8k	Mean	± SD	AUC@8k	$\Delta c_{\text{worst}} \uparrow$
FetchPickAndPlace (8k episodes)	Adaptive	–	**-14.52**	**± 2.01**	**-20.70**	-20.11	± 12.20	-30.52	+0.025 ± 0.113
	Fixed	0.01	-25.26	± 5.50	-35.40	**-16.96**	**± 13.13**	**-31.81**	
	Fixed	0.05	-34.20	± 7.87	-42.36	**-28.87**	**± 27.18**	**-39.07**	
	Fixed	0.10	-42.34	± 6.38	-47.54	**-36.06**	**± 17.45**	**-45.99**	
	Fixed	0.15	-40.41	± 7.69	-48.15	**-35.70**	**± 17.15**	**-46.33**	
FetchPush (8k episodes)	Adaptive	–	†-17.66	± 2.79	-21.68	**-10.70**	**± 2.24**	**-20.22**	+0.091 ± 0.017
	Fixed	0.01	†-12.50	± 1.50	-20.27	**-9.49**	**± 1.07**	**-16.17**	
	Fixed	0.05	†-13.52	± 2.78	-21.50	**-9.72**	**± 1.22**	**-18.89**	
	Fixed	0.10	†-15.50	± 2.50	-25.29	**-9.91**	**± 1.25**	**-21.81**	
	Fixed	0.15	**-12.50**	**± 2.37**	**-24.88**	†-46.42	± 2.19	-46.61	
FetchSlide (8k episodes)	Adaptive	–	†-43.17	± 2.85	-45.21	**-36.54**	**± 5.34**	**-37.73**	+0.100 ± 0.016
	Fixed	0.01	-43.50	± 3.89	-46.23	-38.81	± 5.38	-43.79	
	Fixed	0.05	†-39.42	± 3.46	-42.63	**-32.94**	**± 1.85**	**-35.63**	
	Fixed	0.10	**-42.50**	**± 2.50**	**-45.53**	†-49.35	± 0.00	-48.18	
	Fixed	0.15	**-41.50**	**± 2.80**	**-44.50**	-44.87	± 3.99	-48.49	
								Across all 15 matched configurations	+0.072 ± 0.070

Visual key: **bold black** = preferred within the matched pair; regular black = statistically comparable but not selected by AUC; † gray = significantly worse under the Welch's t-test. For +CAGrad across all 15 comparisons: **best in 11 (73.3%)**, comparable in 2 (13.3%), and significantly worse in 2 (13.3%).

This establishes asymptotic first-order stationarity for the fixed-scale direct-update idealization, not convergence to a finite parameter vector; the latter is separately assumed in Section 3.4. In the implementation, normalization scales are recomputed from stochastic minibatches, CAGrad is solved finitely, and its output is passed to the policy optimizer. Adam can transform the vector when computing the parameter step. These practical updates do not satisfy the fixed-objective exact-gradient premises above.

A.6.2 Recoverable and Preserved Component Alignment

Fix two nonzero normalized component gradients $\tilde{g}_{\text{IL}}$ and $\tilde{g}_{\text{RL}}$, a weight $0 < \lambda < 1$, and write $V = v_\lambda = \lambda\tilde{g}_{\text{IL}} + (1-\lambda)\tilde{g}_{\text{RL}}$. Let $\mathcal{B} = \{d : \|d - V\| \le c\|V\|\}$, with $0 \le c < 1$, and let $d^\star$ be any exact solution of Equation 3. Continuity of the objective and compactness of $\mathcal{B}$ ensure existence. For every feasible d,

$$\lambda\langle\tilde{g}_{\text{IL}}, d\rangle + (1-\lambda)\langle\tilde{g}_{\text{RL}}, d\rangle = \langle V, d\rangle \ge (1-c)\|V\|^2. \tag{5}$$

Feasibility of V also gives $\min_i\langle\tilde{g}_i, d^\star\rangle \ge \min_i\langle\tilde{g}_i, V\rangle$.

Recoverability. An exact solution is non-negatively aligned with both normalized components if and only if

$$\langle\tilde{g}_i, V\rangle + c\|V\| \ge 0, \qquad i \in \{\text{IL}, \text{RL}\}. \tag{6}$$

For $V \neq 0$, this is equivalent to $\min_i \cos(\tilde{g}_i, V) \ge -c$.

Proof. If $V = 0$, the feasible set is $\{0\}$ and the claim is immediate. Otherwise, $\max_{d\in\mathcal{B}}\langle\tilde{g}_i, d\rangle = \langle\tilde{g}_i, V\rangle + c\|V\|$, proving necessity. For sufficiency, if both inner products are non-negative at V, feasibility of V proves the

Table 14: Adaptive-annealing ablation on Gymnasium MuJoCo locomotion tasks with DAgger+DDPG and CAGrad disabled. Values report final reward (mean ± standard deviation) and area under the learning curve over the first 250 episodes (AUC@250). Methods marked with † are significantly worse than the highest-mean schedule for the same task according to a two-sided Welch's t-test ($p < 0.05$; $n = 10$ per configuration). Among the remaining schedules, the method with the highest AUC@250 is bolded.

Task	Annealing	Δ	Mean	± SD	AUC@250
HalfCheetah (250k steps)	Adaptive	–	10,980	± 644	4,728
	Fixed	**0.01**	**10,590**	**± 78**	**9,262**
	Fixed	0.05	10,548	± 106	9,211
	Fixed	0.10	†4,235	± 432	7,398
	Fixed	0.15	†5,324	± 990	6,094
Hopper (250k steps)	**Adaptive**	–	**3,594**	**± 150**	**2,921**
	Fixed	0.01	†3,425	± 7	3,171
	Fixed	0.05	†3,429	± 9	3,148
	Fixed	0.10	†2,952	± 168	3,018
	Fixed	0.15	†2,332	± 1,191	2,763
Walker2d (250k steps)	Adaptive	–	†3,563	± 432	1,976
	Fixed	0.01	4,899	± 144	4,310
	Fixed	**0.05**	**4,834**	**± 99**	**4,329**
	Fixed	0.10	†2,756	± 229	3,716
	Fixed	0.15	†2,347	± 664	3,047
Swimmer (250k steps)	**Adaptive**	–	**345.3**	**± 2.5**	**275.1**
	Fixed	0.01	†335.7	± 1.5	300.1
	Fixed	0.05	†334.2	± 1.2	298.0
	Fixed	0.10	†69.3	± 14.2	229.7
	Fixed	0.15	†19.1	± 21.8	158.9
Ant (250k steps)	Adaptive	–	†1,394	± 801	-444
	Fixed	**0.01**	**3,489**	**± 333**	**3,053**
	Fixed	0.05	3,086	± 906	2,968
	Fixed	0.10	†-464	± 443	1,919
	Fixed	0.15	†-1,062	± 511	969
Humanoid (250k steps)	Adaptive	–	5,128	± 119	4,822
	Fixed	0.01	†5,042	± 11	5,017
	Fixed	**0.05**	**5,057**	**± 11**	**5,022**
	Fixed	0.10	†4,795	± 148	4,953
	Fixed	0.15	†4,759	± 218	4,882

claim. Otherwise, exactly one component has a negative inner product at V, because their positively weighted sum is $\|V\|^2 > 0$. Equation 6 supplies a feasible point with a non-negative inner product for that component. On the segment from V to this point, it crosses zero at $\bar{d}$. The other component is strictly positive there by Equation 5 and its positive weight. Thus a feasible direction has a non-negative minimum, and every exact maximizer does as well. □

Preservation. If $\langle \tilde{g}_i, V \rangle \geq 0$ for either component, then $\langle \tilde{g}_i, d^\star \rangle \geq 0$.

Proof. The case $V = 0$ is immediate. If the baseline inner product is zero, the other is positive by Equation 5, so the baseline minimum is zero and optimality implies both final inner products are non-negative. If the baseline inner product is positive but its final value is negative, it crosses zero on the feasible segment from V to $d^\star$. The other component is strictly positive at that crossing by Equation 5. Immediately before the crossing, both are positive, contradicting the negative optimal minimum at $d^\star$. □

Angular interpretation and scope. For $V \neq 0$, Equation 5 puts the angle between any feasible d and V below $\pi/2$. The distance from V to the ray through d is $\|V\|$ times the sine of that angle and is at most $\|d - V\| \leq c\|V\|$. Hence the angle is at most $\arcsin c$: a component more than $\pi/2 + \arcsin c$ away from V cannot be made non-negatively aligned within the ball. Increasing c widens this margin while weakening the lower bound on mixture alignment. The preservation and recoverability results require an exact maximizer

Table 15: CAGrad performance ablation on Gymnasium MuJoCo locomotion tasks using matched no-CAGrad and CAGrad configurations. For each row, DAgger+DDPG, the annealing type, and the fixed rate Δ are held constant. Values report final reward (mean ± standard deviation) and area under the learning curve over the first 250 episodes (AUC@250). Within each matched pair, the preferred configuration is bolded. A gray result marked with † is significantly worse than its matched alternative according to a two-sided Welch's t-test ($p < 0.05$; $n = 10$ per configuration). When neither configuration fails the test, both are treated as comparable and the configuration with the higher AUC@250 is bolded. The final column reports the paired improvement in mean worst-case cosine, $\Delta c_{\text{worst}} = c_{\text{worst}}^{\text{+CAGrad}} - c_{\text{worst}}^{\text{No CAGrad}}$, as mean ± sample standard deviation across the adaptive schedule and four fixed rates for each task. Positive values indicate improved alignment with the less-aligned objective gradient.

			No CAGrad			+CAGrad			
Task	Annealing	Δ	Mean	± SD	AUC@250	Mean	± SD	AUC@250	$\Delta c_{\text{worst}} \uparrow$
HalfCheetah (250k steps)	Adaptive	–	10,980	± 644	4,728	**10,994**	**± 216**	**7,082**	+0.088 ± 0.099
	Fixed	0.01	**10,590**	**± 78**	**9,262**	†9,578	± 521	1,220	
	Fixed	0.05	†10,548	± 106	9,211	**10,930**	**± 383**	**3,579**	
	Fixed	0.10	†4,235	± 432	7,398	**10,787**	**± 650**	**3,961**	
	Fixed	0.15	†5,324	± 990	6,094	**10,115**	**± 2,127**	**4,213**	
Hopper (250k steps)	Adaptive	–	**3,594**	**± 150**	**2,921**	†3,424	± 156	2,974	+0.325 ± 0.060
	Fixed	0.01	**3,425**	**± 7**	**3,171**	3,430	± 181	2,979	
	Fixed	0.05	**3,429**	**± 9**	**3,148**	3,225	± 573	3,089	
	Fixed	0.10	†2,952	± 168	3,018	**3,520**	**± 172**	**2,818**	
	Fixed	0.15	†2,332	± 1,191	2,763	**3,721**	**± 185**	**2,871**	
Walker2d (250k steps)	Adaptive	–	3,563	± 432	1,976	**3,946**	**± 869**	**3,011**	+0.156 ± 0.068
	Fixed	0.01	**4,899**	**± 144**	**4,310**	†2,317	± 302	4,308	
	Fixed	0.05	**4,834**	**± 99**	**4,329**	†2,531	± 854	1,675	
	Fixed	0.10	†2,756	± 229	3,716	**3,541**	**± 630**	**1,632**	
	Fixed	0.15	†2,347	± 664	3,047	**3,818**	**± 506**	**1,690**	
Swimmer (250k steps)	Adaptive	–	345.3	± 2.5	275.1	**334.8**	**± 19.1**	**290.6**	+0.314 ± 0.091
	Fixed	0.01	**335.7**	**± 1.5**	**300.1**	†13.5	± 18.2	301.1	
	Fixed	0.05	**334.2**	**± 1.2**	**298.0**	†32.7	± 26.4	135.4	
	Fixed	0.10	**69.3**	**± 14.2**	**229.7**	†42.7	± 12.4	98.4	
	Fixed	0.15	**19.1**	**± 21.8**	**158.9**	124.9	± 196.8	103.8	
Ant (250k steps)	Adaptive	–	1,394	± 801	-444	**1,014**	**± 350**	**1,425**	+0.291 ± 0.041
	Fixed	0.01	**3,489**	**± 333**	**3,053**	†-1,111	± 381	1,580	
	Fixed	0.05	**3,086**	**± 906**	**2,968**	†736	± 1,134	-1,143	
	Fixed	0.10	†-464	± 443	1,919	**1,315**	**± 346**	**-983**	
	Fixed	0.15	†-1,062	± 511	969	**1,275**	**± 999**	**-927**	
Humanoid (250k steps)	Adaptive	–	†5,128	± 119	4,822	**5,526**	**± 175**	**5,270**	+0.035 ± 0.041
	Fixed	0.01	**5,042**	**± 11**	**5,017**	†4,711	± 165	5,005	
	Fixed	0.05	**5,057**	**± 11**	**5,022**	4,981	± 259	4,913	
	Fixed	0.10	**4,795**	**± 148**	**4,953**	4,905	± 514	4,802	
	Fixed	0.15	†4,759	± 218	4,882	**5,098**	**± 370**	**4,762**	
									Across all 30 matched configurations +0.202 ± 0.133

Visual key: **bold black** = preferred within the matched pair; regular black = statistically comparable but not selected by AUC; † gray = significantly worse under the Welch's t-test. For +CAGrad across all 30 comparisons: **best in 15 (50.0%)**, comparable in 5 (16.7%), and significantly worse in 10 (33.3%).

and concern its direction before optimizer transformation; they do not establish the same properties for a finite numerical solution or an Adam parameter step.

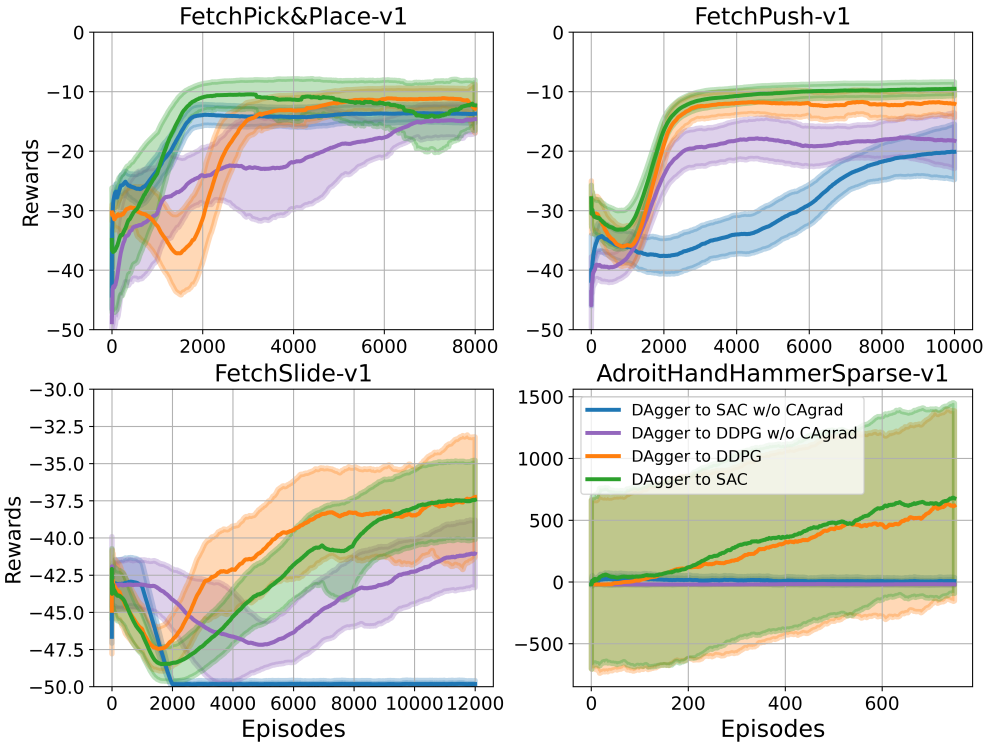


Figure 9: Learning curves with and without CAGrad on the manipulation tasks. The effect on sample efficiency and final return depends on the task and IL–RL pairing; controlled matched comparisons are reported in Table 13.

A.6.3 Proof of Theorem 1

Proof. Let D_{TV} denote total variation distance, $D_{\mathrm{TV}}(p,q) = \frac{1}{2}\|p-q\|_1$. The TRPO performance bound (Theorem 1 in Schulman et al. (2015)) gives

$$J(\pi^*_{k+1}) \geq J(\pi^*_k) + S_k - \frac{4\gamma\epsilon_k}{(1-\gamma)^2} \sup_s D_{\mathrm{TV}}\big(\pi^*_k(\cdot \mid s), \pi^*_{k+1}(\cdot \mid s)\big)^2 .$$

Pinsker's inequality with natural logarithms gives

$$\sup_s D_{\mathrm{TV}}\big(\pi^*_k(\cdot \mid s), \pi^*_{k+1}(\cdot \mid s)\big)^2 \leq \tfrac{1}{2} D_k^{\max}.$$

Therefore,

$$J(\pi^*_{k+1}) \geq J(\pi^*_k) + S_k - C_k D_k^{\max}.$$

By Assumptions 2–3 and the stagewise dominance condition,

$$S_k \geq \eta_k \geq C_k \delta_k \geq C_k D_k^{\max}.$$

Thus $J(\pi^*_{k+1}) \geq J(\pi^*_k)$ for every $k = 0, \ldots, N-1$. □

A.6.4 Proof of Proposition 1

Proof. Condition C1 gives $\|\theta^*(\lambda_k - h) - \theta^*_k\| = O(h)$, and $\theta^*(\lambda_k - h) \in U$ for sufficiently small h by continuity. For every state s, non-negativity of KL divergence and $K_s(\theta^*_k) = 0$ imply $\nabla_\theta K_s(\theta^*_k) = 0$. Taylor's theorem and Condition C2 therefore give

$$D_k^{\max}(h) \leq \tfrac{1}{2} M_k \|\theta^*(\lambda_k - h) - \theta^*_k\|^2 \leq b_k h^2$$

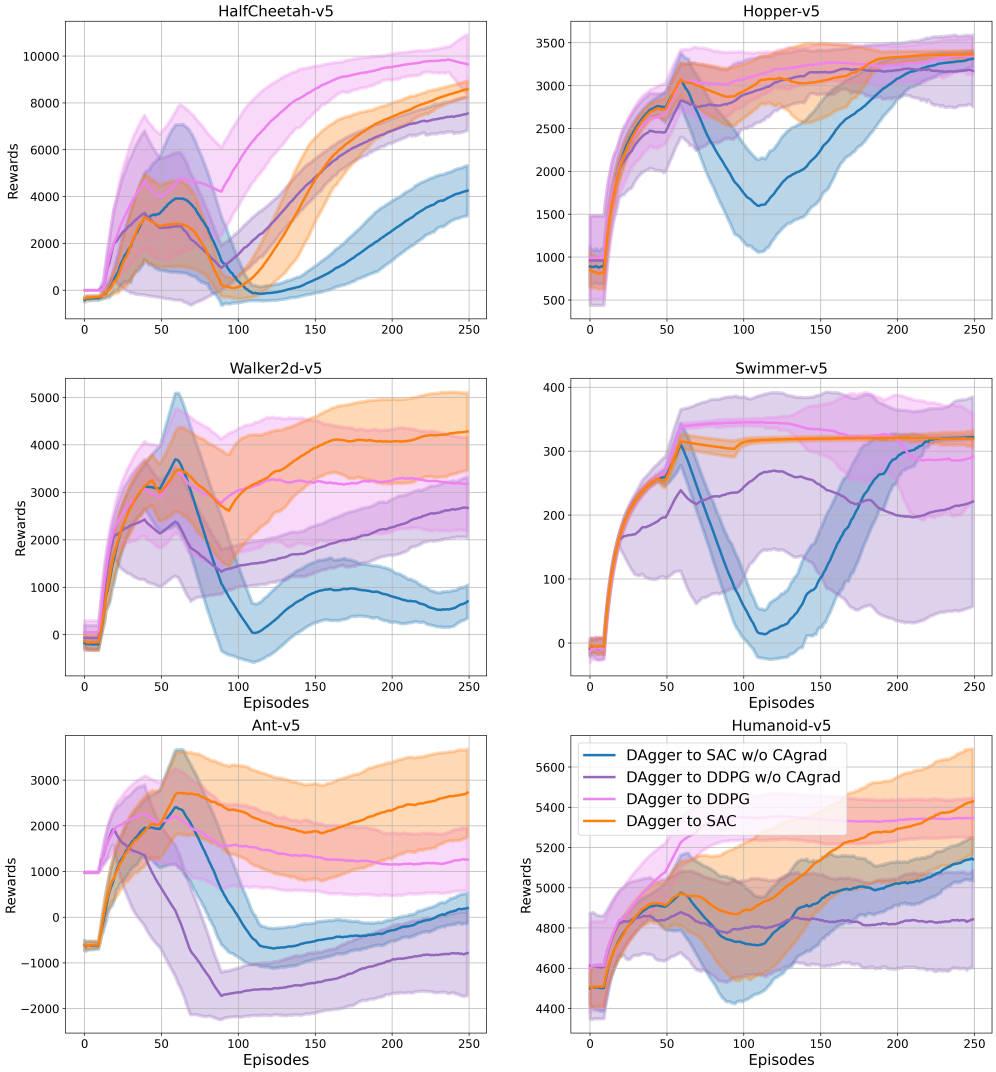


Figure 10: Learning curves with and without CAGrad on the Gymnasium MuJoCo locomotion tasks. Its effect on mid-training return drops and final performance depends on the task and IL–RL pairing; controlled matched comparisons are reported in Table 15.

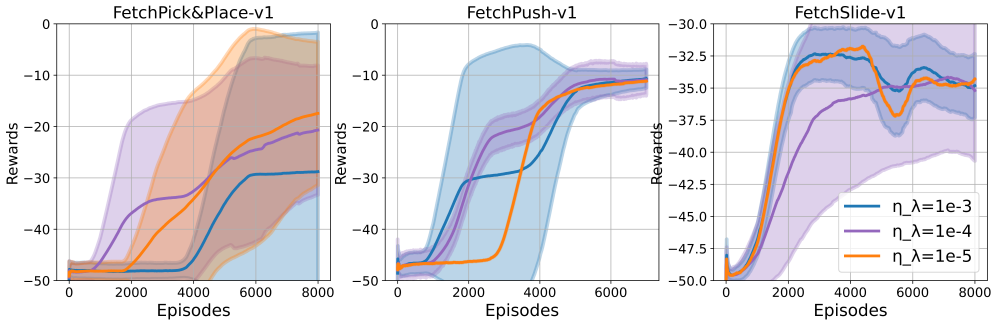


Figure 11: Sensitivity analysis: Effect of λ-optimizer learning rate (η_λ) on Fetch tasks.

for some finite $b_k > 0$ and all sufficiently small h. Since $S_k(0) = 0$, Condition C1 also gives $S_k(h) = a_k h + o(h) \geq \frac{1}{2} a_k h$ for sufficiently small h. Reducing h further if necessary yields

$$S_k(h) \geq \eta_k \geq C_k \delta_k \geq C_k D_k^{\max}(h),$$

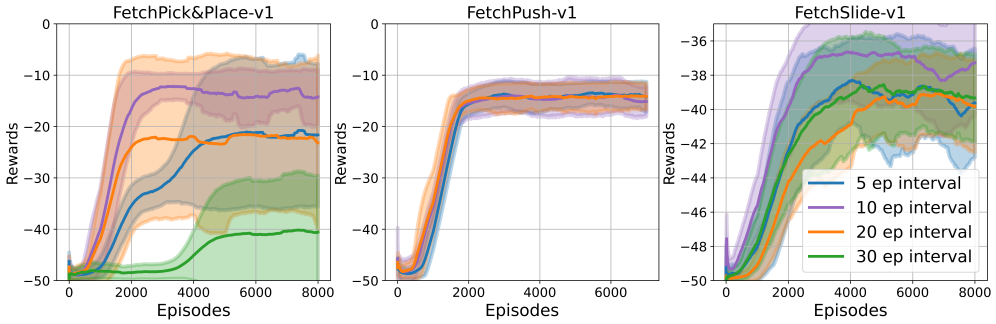


Figure 12: Sensitivity analysis: Effect of λ update interval on Fetch tasks.

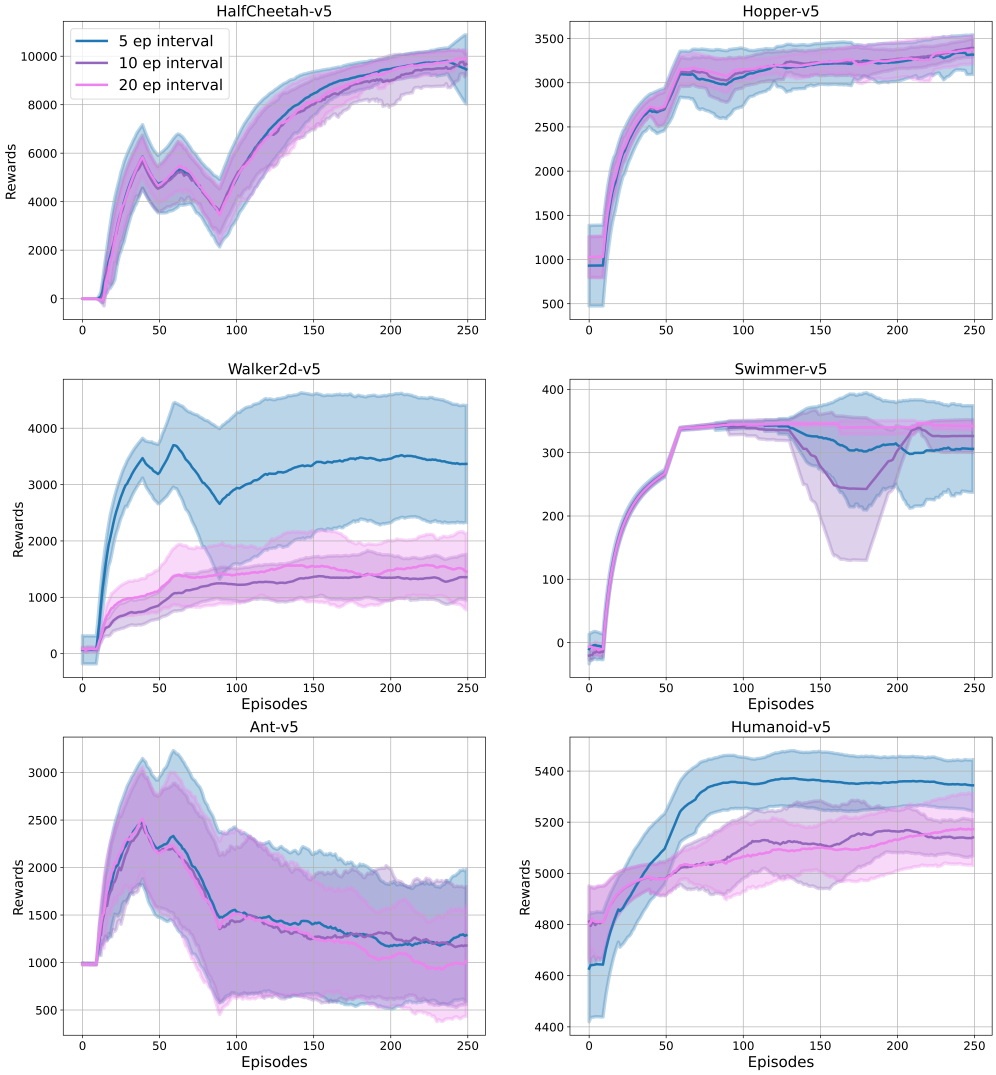


Figure 13: Sensitivity analysis: Effect of λ update interval on Gymnasium MuJoCo locomotion tasks.

which establishes both bounds and their dominance condition. □

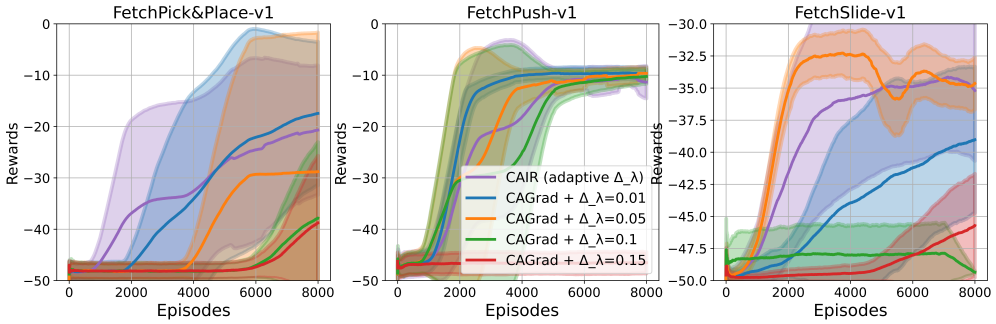


Figure 14: Ablation study comparing adaptive annealing in CAIR with fixed schedules $\lambda \leftarrow \lambda - \Delta$ across the Fetch tasks, with CAGrad enabled for all variants. Adaptive annealing is favored or statistically comparable to the strongest tested fixed schedule in all three tasks while avoiding task-specific selection of Δ; larger fixed rates also produce substantial degradation in several settings.

Applying the absolute-value bound from the proof of Theorem 1 (Appendix A) in Schulman et al. (2015), together with Pinsker's inequality, gives

$$|J(\pi_k(h)) - J(\pi_k^*) - S_k(h)| \leq C_k D_k^{\max}(h) = O(h^2).$$

Thus the surrogate and the return change agree to first order in h.

A.6.5 Example of Uniform KL Curvature

Condition C2 holds, for example, for diagonal Gaussian policies whose mean and log-standard-deviation outputs are twice continuously differentiable, with first and second parameter derivatives uniformly bounded over states on U, and standard deviations uniformly bounded below by a positive constant on U. The example also covers SAC's tanh-squashed Gaussian policies (Haarnoja et al., 2018) under these regularity conditions, since the common, parameter-independent tanh bijection preserves KL divergence.

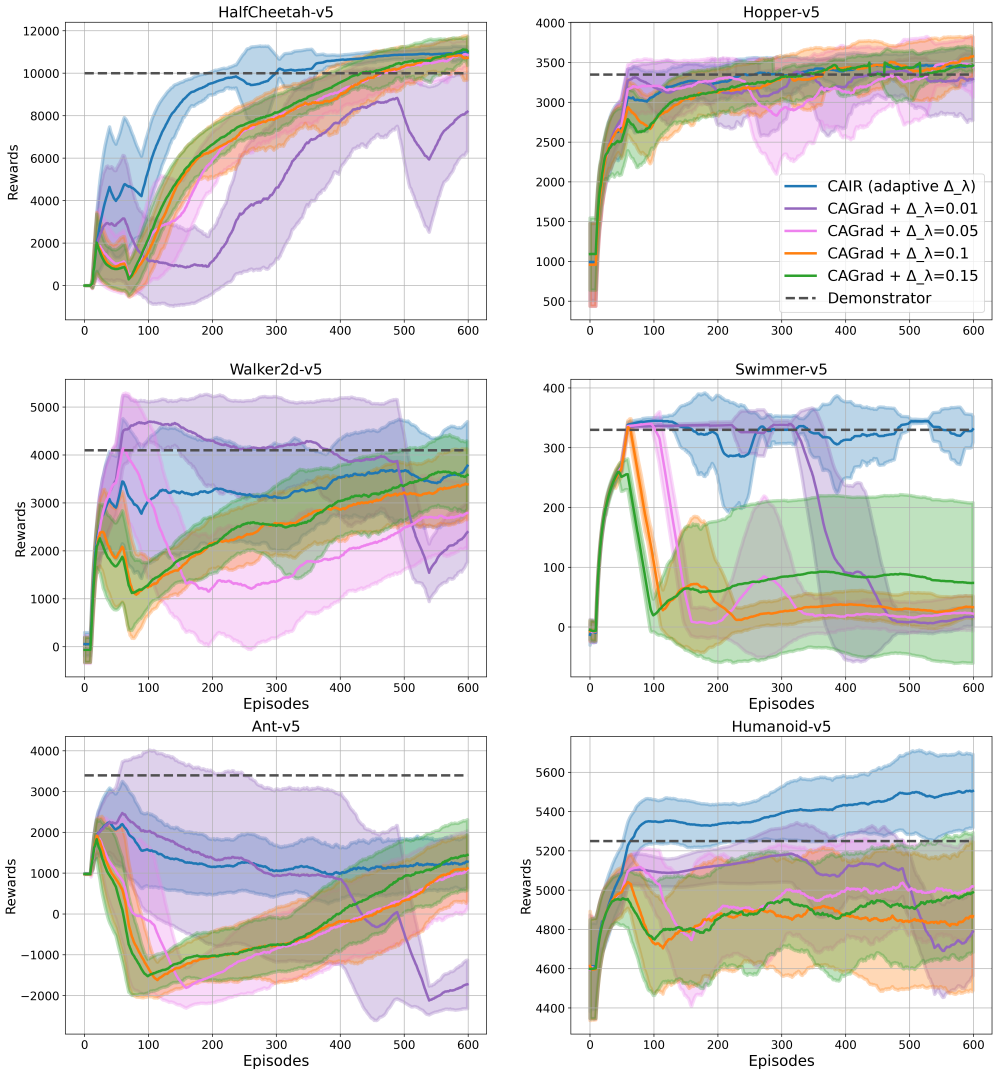


Figure 15: Ablation study comparing adaptive and fixed-schedule loss annealing ($\lambda \leftarrow \lambda - \Delta$) on Gymnasium MuJoCo locomotion tasks using DAgger+DDPG with CAGrad disabled. Under the reported evaluation rule, adaptive annealing is favored in two tasks, statistically comparable but not AUC-favored in two, and significantly worse in two (Table 14).

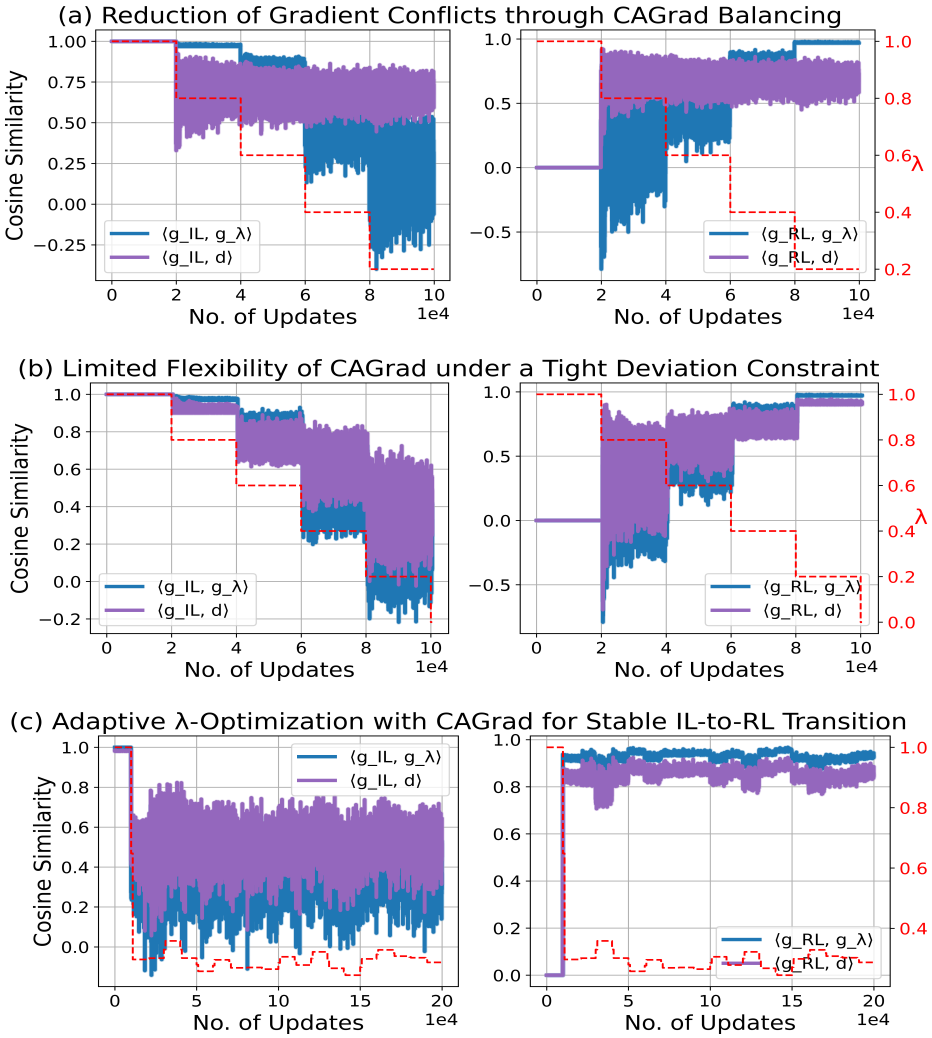


Figure 16: Cosine similarity analysis on *FetchPush-v1* during the IL-to-RL transition. Each row shows cosine similarity between the direction supplied to the policy optimizer and g_{IL} (left) or g_{RL} (right). **(a)** CAGrad resolves gradient conflict during fixed schedule annealing, but requires tuning. **(b)** CAGrad with a tight constraint fails to resolve gradient conflict under fixed annealing. **(c)** CAIR (Adaptive Annealing + CAGrad) maintains positive alignment over the plotted mixed phase of this representative run, even with a tight constraint. (x-axis: gradient updates; y-axis: cosine similarity averaged over 10 updates).