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

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement Learning (RL) and Imitation Learning (IL) offer complementary capabilities: RL can learn high-performing policies but is data-intensive, whereas IL enables rapid learning from demonstrations but is limited by the demonstrator's quality. Combining them offers the potential for improved sample efficiency in learning high-performing policies, yet naïve integrations often suffer from two fundamental issues: (1) *negative transfer*, where optimizing the IL loss hinders effective RL fine-tuning, and (2) *gradient conflict*, where differences in the scale or direction of IL and RL gradients lead to unstable updates. We introduce *Conflict-Averse IL–RL* (CAIR), a general framework that addresses both challenges by combining two key components: (1) *Loss Manipulation*: an adaptive annealing mechanism utilizing a convex combination of IL and RL losses. This mechanism dynamically increases the weight of the RL loss when its gradient aligns with the IL gradient and decreases it otherwise, mitigating instabilities during the transition from IL to RL. (2) *Gradient Manipulation*: to further reduce conflict, we incorporate CAGrad to compute a joint gradient that balances IL and RL objectives while avoiding detrimental interference. Under standard trust-region assumptions, CAIR guarantees monotonic improvement in the expected return when the loss weights are annealed monotonically. Our empirical study evaluates CAIR on four sparse-reward MuJoCo domains, where pure RL algorithms typically struggle. Compared against relevant hybrid RL baselines, CAIR improves sample efficiency in three out of four domains and asymptotic performance in two, while performing comparably on the remainder. These trends are consistent across multiple combinations of IL (BC, DAgger) and RL (DDPG, SAC, PPO) methods, demonstrating the robustness of the novel framework.

1 Introduction

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

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

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

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

This article presents this framework in detail and shows that it is theoretically grounded, offering monotonic improvement guarantees under trust-region-constrained optimization (Schulman et al., 2017; 2015b). Empirical evaluations on sparse-reward MuJoCo environments (Todorov et al., 2012) demonstrate its effectiveness 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 novel framework outperforms baselines that use demonstration data in sample efficiency on three out of four tasks and in asymptotic performance on two out of four tasks, while performing comparably on the remaining domains. These results suggest that CAIR is a robust and effective approach for leveraging demonstration data to enhance RL training.

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, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and γ is the discount factor. At each timestep t, the agent executes an action a_t in state s_t, leading to a new state s_{t+1} and a reward $r_t = R(s_t, a_t)$. The tuple (s_t, a_t, r_t, s_{t+1}) is denoted as a *transition.* A finite sequence of successive transitions starting from an initial state s_0 and leading to a terminal state s_T forms an *episode*, denoted by $\tau = (s_0, a_0, r_0, s_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$.

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

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

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

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

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

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

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

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

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

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

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

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

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

We denote by T_λ the task associated with optimizing $L_\lambda(\pi_\theta)$ at a fixed value of λ, with the pure RL task corresponding to $\lambda = 0$. The optimal policy for task T_λ is $\pi^*_\lambda = \arg\min_{\pi_\theta} L_\lambda(\pi_\theta)$.

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

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

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

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

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

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

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

2.1 Related Work

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

Addressing 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, negative transfer commonly arises during the transition from imitation-dominated to reward-dominated optimization, where RL-driven updates can interfere with useful imitation-induced behavior (Goecks et al., 2019; Rengarajan et al., 2022). Many prior approaches mitigate this effect using fixed or hand-designed schedules for annealing the IL weight (Rengarajan et al., 2022; Goecks et al., 2019; Zhu et al., 2022a). However, such heuristics are agnostic to gradient interactions and can still produce adverse update directions. Recent work addresses negative transfer by detecting when auxiliary or transferred signals harm the main objective. Du et al. (Du et al., 2018) adapt auxiliary-loss weights using gradient cosine similarity to reduce interference with a fixed primary task. While conceptually related, their method does not explicitly model a staged IL–to–RL transition and, in sparse-reward settings, may downweight imitation losses prematurely when reward gradients are weak or noisy. Wang et al. (Wang et al., 2019) study negative transfer in transfer learning and mitigate it through filtering irrelevant source data. In contrast, our setting does not involve domain shift: we assume demonstrations are task-relevant and focus on preventing optimization-level interference between imitation and reinforcement objectives during the IL–to–RL transition, without discarding demonstrations. ForkMerge (Jiang et al., 2023) further argues that negative transfer is not always explained by gradient conflict and proposes a validation-driven branching strategy. We do not claim gradient conflict captures all forms of negative transfer; rather, CAIR targets a specific and actionable mechanism in IL–RL training—destructive interference between imitation and reinforcement updates—and mitigates it through per-update objective weighting and gradient manipulation without requiring validation signals or multiple training branches.

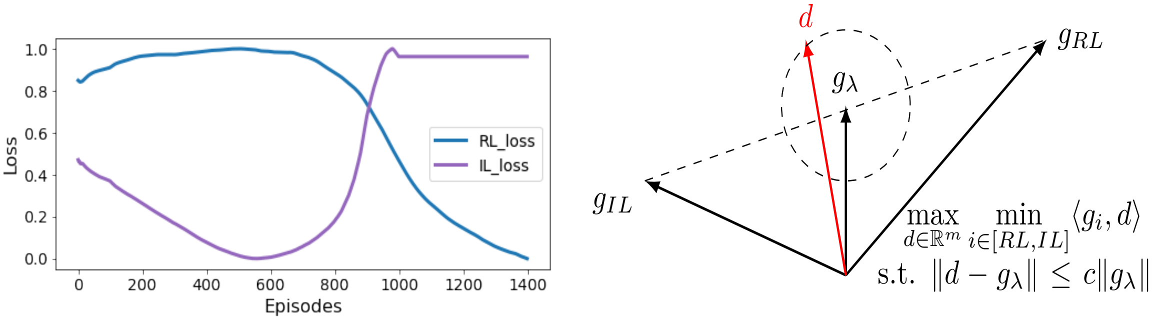
Offline RL. Offline RL learns reward-driven policies from fixed datasets without additional environment interaction (Levine et al., 2020). Although some offline RL methods incorporate auxiliary imitation losses (Hester et al., 2018; Nair et al., 2018; 2020), these objectives are typically used as regularizers rather than as part of an explicit IL–RL optimization framework. The primary focus of offline RL is mitigating distributional shift and out-of-distribution action selection, often through conservative value-learning approaches such as CQL (Kumar et al., 2020) and IQL (Kostrikov et al., 2021). Hybrid RL methods similarly combine offline datasets with online interaction to improve sample efficiency (Song et al., 2022; Ball et al., 2023), but their focus is on stabilizing value learning under distribution shift rather than addressing IL–RL optimization dynamics. Neither offline RL nor hybrid RL explicitly accounts for gradient conflicts between imitation and reinforcement signals - a challenge our method is designed to address directly.

Gradient Manipulation. Gradient manipulation is often used in Multi-Task Learning (MTL) (Désidéri, 2012; Yu et al., 2020; Sener & Koltun, 2018; Liu et al., 2021; 2023) to resolve conflicting gradients arising from different task objectives. These techniques commonly define a joint objective that optimizes the average loss across tasks using gradient manipulation algorithms (Désidéri, 2012; Yu et al., 2020; Chen et al., 2018). Our method repurposes these techniques (specifically CAGrad (Liu et al., 2021)) to minimize conflicts between IL and RL gradients while guaranteeing convergence to a stationary point of the combined loss. Unlike prior approaches such as MGDA (Sener & Koltun, 2018) and PCGrad (Yu et al., 2020), which converge to arbitrary points on the Pareto front, CAGrad provably converges to the optimum of the average loss (see Theorem 3.2 in (Liu et al., 2021)).

3 Conflict-Averse IL to RL

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

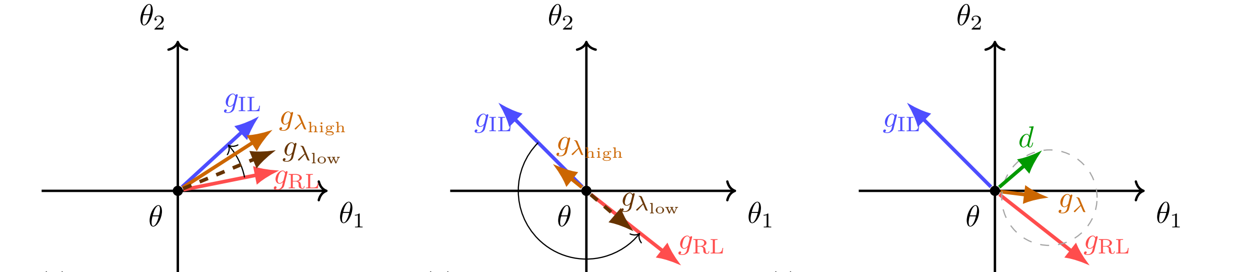
Furthermore, under standard assumptions, CAIR provides monotonic improvement guarantees in expected return when used with trust-region-based optimization.



(a) Loss Manipulation

(b) Gradient Manipulation

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



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

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

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

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

3.1 Adaptive Loss Annealing

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

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

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

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

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

The constraint in Equation 2 ensures that updates remain positively aligned with the IL objective, preventing regressions where the combined update would oppose the demonstration. Simultaneously, the maximization objective favors progress toward the RL goal. This enables a stable transition, allowing λ to decrease only when the IL and RL gradients are sufficiently aligned.

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

The effect of λ-annealing can be understood geometrically through the gradient-alignment illustration in Figure 2. Panel (a) shows the case where IL and RL gradients are well aligned: the angle between g_{IL} and g_{RL} is small, and the combined gradients $g_{\lambda_{\mathrm{high}}}$ (solid orange) and $g_{\lambda_{\mathrm{low}}}$ (dashed brown) remain nearly identical.

In this case, decreasing λ produces consistent update directions, and the transition from IL to RL can proceed aggressively without destabilizing learning.

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

3.2 Gradient Manipulation

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

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

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

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

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

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

3.3 Empirical Gradient Analysis

A central challenge in IL-to-RL transfer lies in how the *combined gradient* g_λ evolves as the weighting parameter λ decreases. We are particularly interested in avoiding situations where g_λ becomes *negatively aligned* with the imitation gradient g_{IL}, i.e., $\langle g_{\mathrm{IL}}, g_\lambda\rangle < 0$. A negative cosine similarity indicates that the combined update moves the policy *against* the direction favored by the demonstrator. Ensuring that g_λ remains positively aligned with g_{IL} avoids RL-driven updates that unlearn useful imitation behavior during the transition. Following this intuition, we turn to empirically investigate the impact CAIR and its two components, namely adaptive loss annealing and gradient manipulation, on gradients alignments, focusing on g_{IL}, g_{RL}, the combined gradient g_λ, and the final update direction d. The results are presented for a single run on the *FetchPick&Place-v1* task, with *Behavior Cloning* and *DDPG* as the IL and RL algorithms respectively. Nonetheless, similar general trends (with respect to gradients alignment) were observed across 5 other seeds and other IL-RL algorithms combinations (as listed in Section 4.1).

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

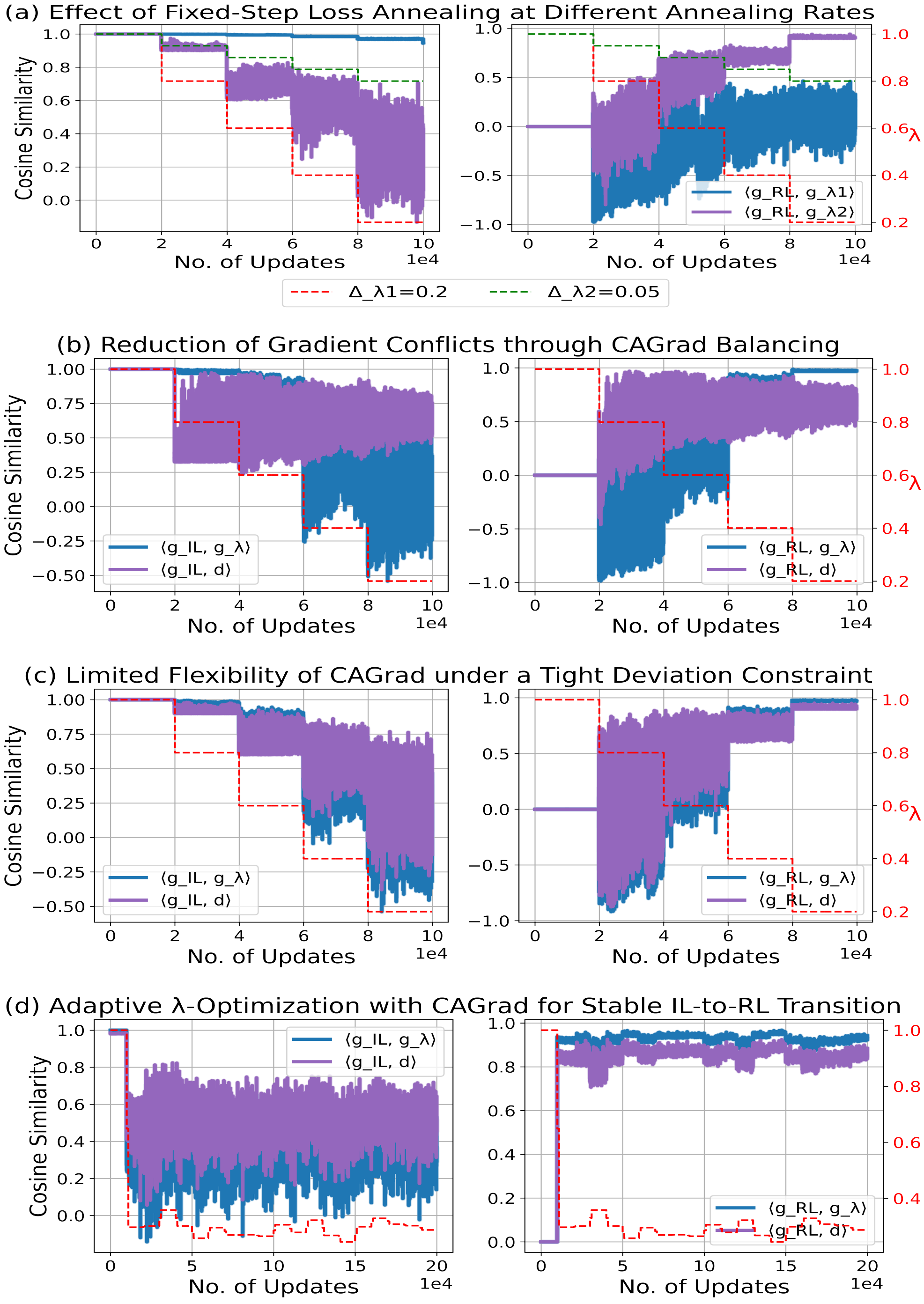


Figure 3: Cosine similarity analysis on *FetchPick&Place-v1* during the IL-to-RL transition. The plots track the alignment between the combined update direction and the individual task gradients: g_{IL} on the left column and g_{RL} on the right column. **(a)** Naïve fixed annealing leads to negative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

between the IL gradient and the combined gradient drops below zero, indicating that the update direction actively opposes the imitation objective. While the slower schedule (blue) exhibits less abrupt fluctuations, it remains a heuristic approach that does not explicitly account for the alignment between g_{IL}, g_{RL}, and g_λ.

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

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

3.4 Monotonic Improvement Guarantee

The following Lemma demonstrates that, under standard trust-region assumptions (Schulman et al., 2015b) and monotonic annealing, CAIR induces a sequence of tasks whose optimal policies enjoy monotonic improvement in expected return. This theoretical grounding ensures that the algorithm prevents temporary performance drops (regressions) during the critical IL–to–RL transition.

Lemma 1. ***(Monotonic Improvement).*** *The Conflict-Averse IL-RL framework guarantees monotonic non-decreasing expected return $J(\pi)$ between any two successive tasks, T_λ and $T_{\hat{\lambda}}$, subject to the following assumptions:*

1. *A trust-region policy optimization method is used as the underlying solver.*
2. *The policy converges to the optimum π^*_λ for task T_λ before transitioning.*
3. *The optimal policy trajectory π^*_λ is continuous with respect to λ.*
4. *The annealing schedule is monotonically non-increasing ($\hat{\lambda} < \lambda$).*
5. *The step size $\lambda - \hat{\lambda}$ is sufficiently small.*

Proof. The proof is established through the following three claims:

1. Proximity of optima: Given Assumption 3 (continuity), for a sufficiently small step $\lambda - \hat{\lambda}$ (Assumption 5), the Kullback-Leibler divergence between successive optimal policies is bounded by a small constant ϵ:

$$D_{KL}(\pi^*_\lambda \| \pi^*_{\hat{\lambda}}) \leq \epsilon.$$

2. Convergence to the new optimum: Starting from $\pi = \pi^*_\lambda$, we solve the new task $T_{\hat{\lambda}}$. Since the initial policy is within the trust region of the new optimum ($D_{KL} \leq \epsilon$), Theorem 3.2 of Liu et al. (Liu et al., 2021) guarantees that optimizing the convex combination $L_{\hat{\lambda}}(\theta)$ using CAGrad converges to the stationary point of the average loss, i.e., $\pi \to \pi^*_{\hat{\lambda}}$.

3. Improvement in Expected Return: Finally, we show that this transition improves the RL objective. Since $\pi^*_{\hat{\lambda}}$ minimizes $L_{\hat{\lambda}}$, we have:

$$L_{\hat{\lambda}}(\pi^*_{\hat{\lambda}}) \leq L_{\hat{\lambda}}(\pi^*_{\lambda}).$$

Expanding the loss $L_\lambda = (1-\lambda)L_{RL} + \lambda L_{IL}$, and given that $\hat{\lambda} < \lambda$, standard sensitivity results in scalarized vector optimization (Boyd & Vandenberghe, 2004, Sec. 4.7.4) imply that shifting the weight towards the RL objective yields a solution with lower RL loss. Consequently, the RL loss decreases:

$$L_{\mathrm{RL}}(\pi^*_{\hat{\lambda}}) \leq L_{\mathrm{RL}}(\pi^*_{\lambda}).$$

Since maximizing expected return is equivalent to minimizing L_{RL} (neglecting constants), this implies $J(\pi^*_{\hat{\lambda}}) \geq J(\pi^*_{\lambda})$. □

Although this Lemma strictly applies under trust-region assumptions, we show empirically in Section 4 that CAIR results in stable, monotonic learning curves even when paired with algorithms lacking these formal guarantees, such as SAC (Haarnoja et al., 2018) and DDPG (Lillicrap et al., 2015), and even when λ is annealed adaptively rather than strictly monotonically.

3.5 Generality of the Framework

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

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

4 Experimental Study

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

1. **Robustness:** Is CAIR robust to different combinations of IL and RL algorithms?

2. **Performance:** Can CAIR achieve improved sample efficiency and asymptotic performance compared to baseline hybrid-RL approaches?

3. **Ablation:** What the relative impact of incorporating Conflict-Averse Gradient Descent (CAGrad) alongside adaptive loss annealing?

4.1 Settings

For each domain, we define a suboptimal rule-based demonstrator following (Bajaj et al., 2023). Both interactive demonstrators and offline datasets for our approach and baselines are generated using these policies. We implement our algorithm by modifying the `stable-baselines3` library (Raffin et al., 2021).[1] To support full reproducibility, our codebase and execution instructions are publicly available.[2] Detailed hyperparameters and hardware specifications are provided in Appendix A.2.

[1] `https://github.com/DLR-RM/stable-baselines3`

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

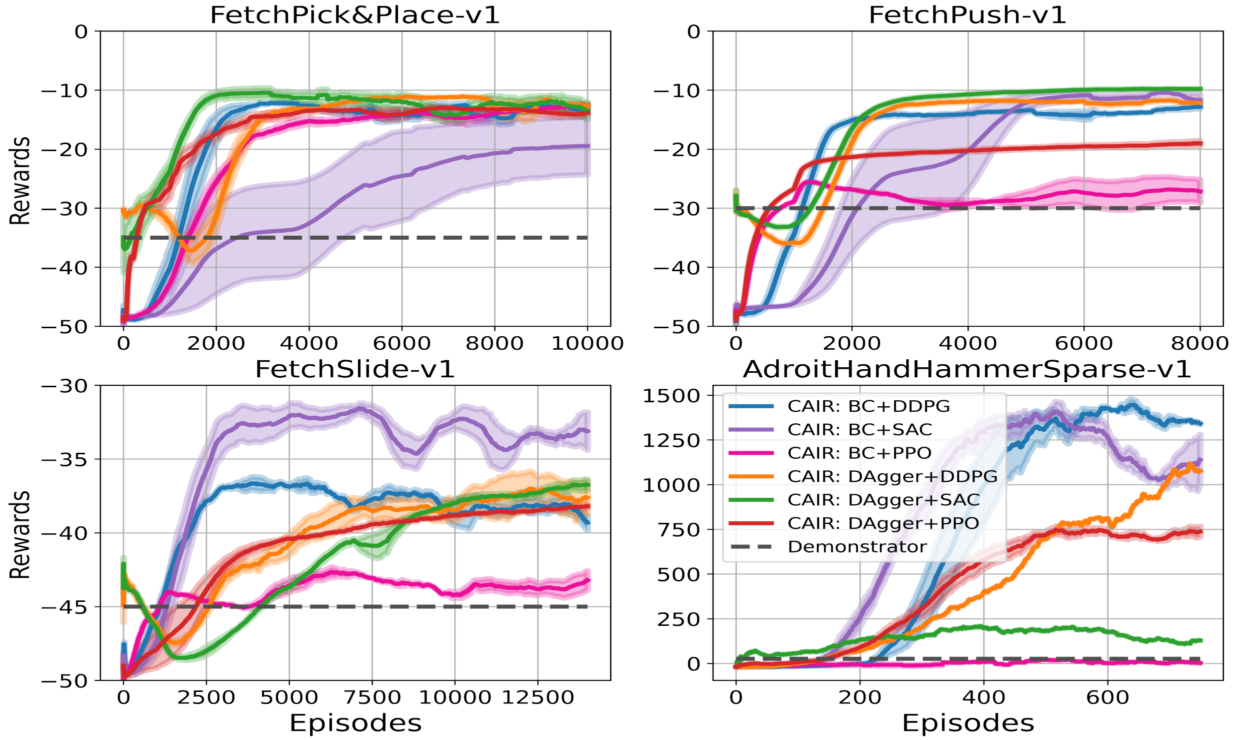


Figure 4: Robustness analysis demonstrating the broad applicability of CAIR across different combinations of IL algorithms (BC, DAgger) and RL algorithms (SAC, DDPG, PPO). CAIR enables stable transitions in all IL–RL combinations except BC+PPO on *AdroitHandHammerSparse-v1*.

Domains. We conduct experiments on four challenging MuJoCo domains (Todorov et al., 2012): *FetchPickAndPlace-v1* (P&P), *FetchSlide-v1* (FS), *FetchPush-v1* (FP) (Plappert et al., 2018), and *AdroitHandHammerSparse-v1* (Rajeswaran et al., 2017). These tasks represent distinct manipulation challenges, ranging from precision placement and trajectory planning to high-dimensional control. Due to sparse rewards and complex dynamics, simple trial-and-error RL struggles on these tasks, making them ideal testbeds for IL-to-RL transfer. (Note: Episode lengths are 1,000 steps for Fetch domains and 4,000 steps for Adroit). All experiments are repeated with 5 different random seeds; we report the mean performance with a 1-σ shaded error region.

4.2 Generality of CAIR

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

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

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

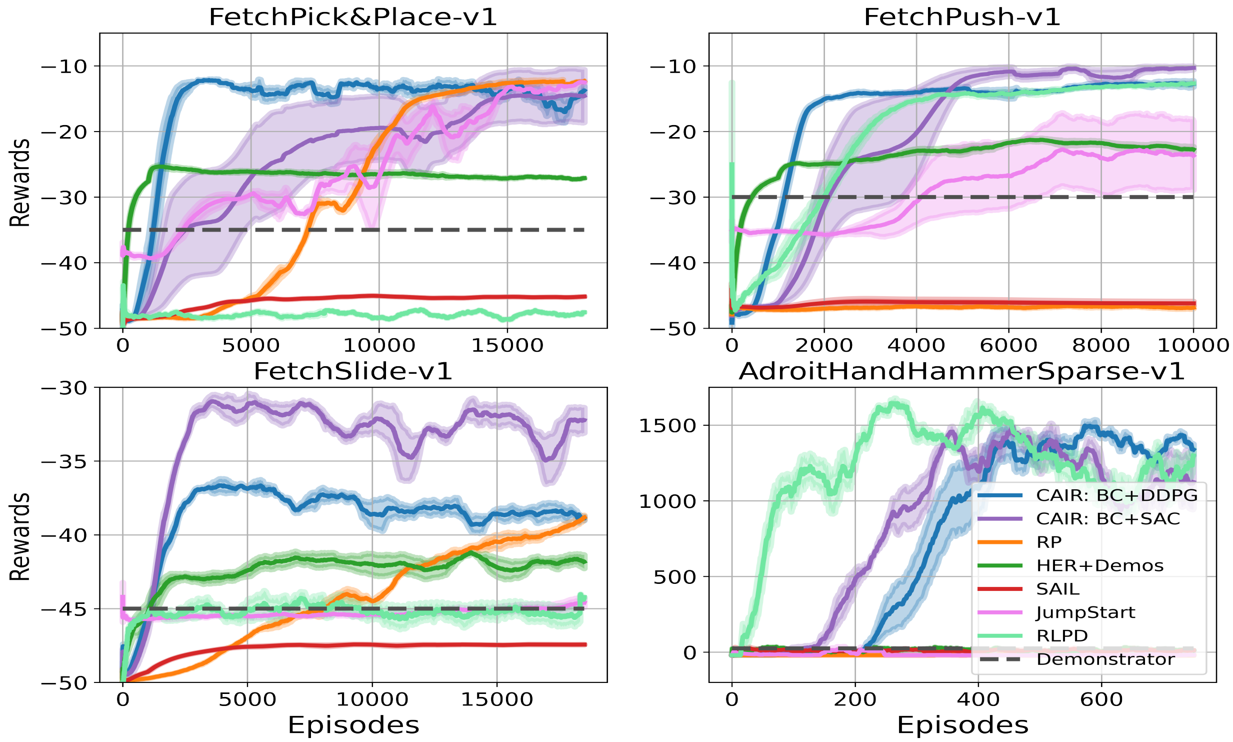


Figure 5: Learning trends for CAIR (BC to SAC and BC to DDPG) compared to hybrid-RL baselines.

The annealing weight λ is updated every 10 policy updates (see Line 16 of Algorithm 1 in Appendix A.4).

Results. Figure 4 demonstrates that CAIR successfully trains agents to outperform the suboptimal demonstrator (black dashed line) in five out of six tested IL–RL combinations across all domains. This confirms that the framework is robust to changes in both the imitation and reinforcement components.

We observe interesting nuances regarding sampling assumptions:

- **Off-Policy RL (SAC, DDPG):** Both BC and DAgger pairings demonstrate meaningful learning. DAgger+RL tends to yield better sample efficiency or final performance in tasks like *FetchPick&Place* and *FetchPush*, whereas BC+RL performs better in *FetchSlide* and *AdroitHandHammer*. These differences appear to be domain-dependent rather than limitations of the CAIR framework itself.

- **On-Policy RL (PPO):** The choice of imitation method matters substantially here. **DAgger+PPO** consistently outperforms the demonstrator, whereas **BC+PPO** performs similarly to the demonstrator without significant improvement in three out of four tasks. This result aligns with the fundamental sampling mismatch: BC relies on a static, off-policy dataset, whereas PPO requires on-policy rollouts collected under the current policy to estimate advantages effectively.

Overall, these results provide a positive answer to Research Question 1, confirming that CAIR is broadly applicable while highlighting that IL–RL combinations with compatible sampling assumptions (e.g., interactive IL with on-policy RL) naturally benefit most from the framework.

Sensitivity Analysis. Complementing the generality analysis, we examine whether the robust learning observed across IL–RL pairings is sensitive to hyperparameter selection. CAIR exposes three primary hyperparameters: (i) the CAGrad deviation parameter c; (ii) the update interval for the adaptive λ-optimization; and (iii) the learning rate η_λ used for updating λ.

- **CAGrad Constraint (c):** Across tasks, we observe that moderate values (e.g., $c \in [0.45, 0.95]$) produce consistent behavior with only minor variations in final performance. However, when c is set too small, the adjusted gradient is forced to remain close to the original combined gradient g_λ, preventing CAGrad from effectively resolving IL–RL conflicts. As observed in our ablation studies (Figure 7 in Appendix A.3), this can lead to severe performance degradation due to unmitigated gradient interference.

- **Update Interval:** To provide a stable learning signal, we update λ every 10 task episodes. This allows the policy to train under a fixed weight before recomputing the alignment balance. Updating too frequently makes the weighting overly sensitive to short-term gradient noise, whereas updating too infrequently produces a sluggish transition toward RL. The impact of this interval is detailed in Figure 9 (Appendix A.3).

- **Annealing Rate (η_λ):** The learning rate of the λ-optimizer exhibits a similar trade-off. Very small values stall the annealing process, while larger values produce abrupt shifts driven by noisy gradient estimates. Consequently, we default to $\eta_\lambda = 10^{-4}$. Figure 8 compares learning curves for $\eta_\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}\}$, showing that both extremes reduce sample efficiency. This aligns with prior findings (Uchendu et al., 2022) that excessively rapid transitions from IL to RL can cause performance collapse, further motivating the need for controlled, alignment-aware annealing.

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

4.3 Benchmark Comparison

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

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

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

Hindsight Experience Replay (HER) with Demonstrations. Hindsight Experience Replay (**?**) 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.

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

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

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

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

4.4 Results

Sample Efficiency and Asymptotic Performance. To evaluate the effectiveness of CAIR, we compare our best-performing off-policy IL–RL combinations (BC+DDPG and BC+SAC) against the baselines. As shown in Figure 5, CAIR achieves comparable or superior sample efficiency and final performance in **three out of four** domains.

Notably, while RLPD demonstrates strong sample efficiency on *AdroitHandHammerSparse-v1*, CAIR matches its asymptotic performance. However, RLPD fails to consistently outperform the demonstrator on the *FetchPickAndPlace-v1* and *FetchSlide-v1* domains, indicating a lack of robustness across task types. In contrast, CAIR (in both SAC and DDPG variants) is the only algorithm among the evaluated baselines that consistently learns to outperform the demonstrator across **all four domains**.

This consistency suggests that by actively resolving gradient conflicts, CAIR prevents the destructive interference that often destabilizes other hybrid methods. Overall, these results provide a positive answer to Research Question 2, demonstrating that CAIR yields strong, robust performance where other baselines struggle.

4.5 Ablation Study: Impact of CAGrad

To address Research Question 3, we investigate the relative impact of incorporating Conflict-Averse Gradient Descent (CAGrad) alongside adaptive loss annealing.

We compare our best-performing off-policy configurations (BC+DDPG and BC+SAC) against variants where the CAGrad component is disabled (relying solely on adaptive annealing). The results, presented in Figure 7 (Appendix A.3), indicate that incorporating CAGrad generally enhances sample efficiency and yields an average asymptotic performance improvement of approximately 20% across the Fetch domains.

The benefits are particularly pronounced in the high-dimensional *AdroitHandHammerSparse-v1* domain, where meaningful learning is difficult to achieve without CAGrad. Crucially, we observed no instances where adding CAGrad was detrimental to performance, suggesting it is a safe and effective addition to the IL–to–RL pipeline.

5 Conclusions

We introduced *Conflict-Averse IL-RL* (CAIR), a general framework that integrates Imitation Learning (IL) and Reinforcement Learning (RL) by addressing the fundamental challenge of negative transfer stemming from conflicting gradients. CAIR dynamically adapts the weighting of IL and RL objectives, shifting focus toward the RL direction only when it remains compatible with the imitation signal. Furthermore, by incorporating Conflict-Averse Gradient Descent (CAGrad), CAIR actively projects update directions to maintain alignment with both objectives. Together, these mechanisms mitigate destructive interference, enabling smooth and stable transitions from imitation-driven initialization to reward-driven optimization.

The method is theoretically grounded, offering convergence guarantees under trust-region constraints, while proving empirically effective across diverse RL paradigms (including PPO, SAC, and DDPG). Experiments on four sparse-reward continuous control domains demonstrate that CAIR improves sample efficiency in three out of four tasks and asymptotic performance in two, consistently outperforming the demonstrator, as opposed to competitive hybrid-RL baselines.

A practical limitation of CAIR is its sensitivity to the compatibility of the underlying algorithms. As observed in our results, the framework is less effective when the IL and RL components rely on conflicting sampling assumptions, such as combining static, off-policy demonstrations (BC) with strictly on-policy RL updates (PPO). Future work may explore unifying these sampling distributions or extending CAIR to multi-modal demonstration data.

References

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.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A Appendix

A.1 Domain Descriptions

We evaluate CAIR on four continuous control domains simulated in MuJoCo. Visualizations are provided in Figure 6.

A.1.1 Fetch Robotics Domains

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). These are multi-goal domains where the target position is randomized every episode. A rule-based suboptimal demonstrator (Nair et al., 2018) is used for all tasks.

Common Properties:

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

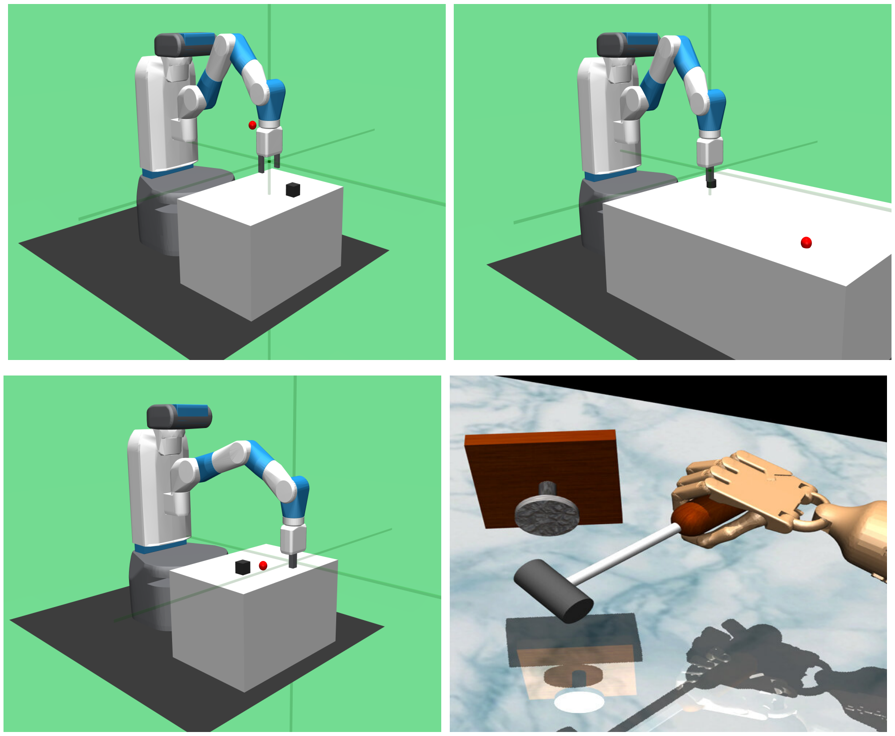


Figure 6: Visualizations of the four MuJoCo sparse-reward domains used in our experiments. Top-left: FetchPickAndPlace-v1 domain; Top-right: FetchSlide-v1 domain; Bottom-left: FetchPush-v1 domain; Bottom-right: AdroitHandHammerSparse-v1

- **Transition Function $\mathcal{P}$:** The robot updates its position based on the action. Each step corresponds to 20 simulator steps ($\Delta t = 0.002s$).
- **Reward Function R_T:** Sparse binary reward. +1 if the object is within 5cm of the target, 0 otherwise.

Task Specifics:

- **FetchPickAndPlace-v1:** The robot must grasp a box and move it to a target in the air or on the table. This is the most complex Fetch task due to the grasping requirement.
- **FetchSlide-v1:** The robot must push a puck across a long table to a target outside its immediate reach. This requires learning the object's sliding dynamics to apply precise force.
- **FetchPush-v1:** The robot pushes a box to a target location within its reach. This is simpler than Slide but still requires precise manipulation.

A.1.2 AdroitHandHammerSparse-v1

This domain from the D4RL benchmark (Fu et al., 2020) features a 30-DoF Shadow Hand that must grasp a hammer and drive a nail into a board.

- **State Space $\mathcal{S}$:** Includes hammer position/orientation/velocity, hand joint angles/velocities, nail position, and nail depth.
- **Action Space $\mathcal{A}$:** 24-dimensional continuous space representing joint torques ($\in [-1, 1]$).
- **Reward Function R_T:** Sparse. +10.0 when the nail is fully driven; -0.1 otherwise to encourage efficiency.

A.2 Hyperparameters

Experiments were conducted on an AMD Ryzen Threadripper PRO 5975WX (32-Core, 4.3GHz). Tables 1–3 detail the hyperparameters. Network architectures use tanh activations. **Demonstrations:** 50 episodes of data were provided for each task. Episode lengths: 1000 steps (Fetch), 4000 steps (Adroit).

Domain	**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	(512, 512)	0.98	$3 \cdot 10^{-4}$	256	0.02	Auto

Table 1: Hyperparameters for CAIR (BC/DAgger + SAC)

Domain	**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	(512, 512)	0.98	$3 \cdot 10^{-4}$	128	0.02	0.10

Table 2: Hyperparameters for CAIR (BC/DAgger + DDPG)

Domain	**Network**	γ	**Init** λ	**LR**	**Batch**	**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	(512, 512)	0.98	0.95	$3 \cdot 10^{-4}$	1024	20	0.02

Table 3: Hyperparameters for CAIR (BC/DAgger + PPO)

A.3 Sensitivity Analysis

We provide additional empirical analysis of the CAIR framework components.

- **CAGrad Ablation (Fig. 7):** Demonstrates the consistent performance gains from adding gradient conflict resolution.
- **Sensitivity to η_λ (Fig. 8):** Shows that extreme learning rates for the λ-optimizer destabilize learning.
- **Update Interval (Fig. 9):** Highlights the trade-off between responsiveness and stability in λ updates.

A.4 Conflict-Averse IL to RL Algorithm

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

We evaluate CAIR with SAC, DDPG, and PPO as the RL algorithms, combined with Behavior Cloning (BC) and DAgger as the IL algorithms. The corresponding hyperparameters are reported in Tables 1, 2, and 3. All remaining hyperparameters follow the default settings of Stable-Baselines3 (Hill et al., 2018). The constraint parameter c for CAGrad and the learning rate for the λ-optimizer are fixed across domains unless stated otherwise. The adaptive λ-optimization procedure is summarized in Algorithm 2 (Appendix A.4).

Algorithm 1 Conflict-Averse IL to RL (CAIR)

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

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

```
**Input:** Gradients $g_{\text{IL}}, g_{\text{RL}}$, current $\lambda_{\text{cur}}$, learning rate $\eta_\lambda$
Normalize: $\tilde{g}_{\text{IL}} \leftarrow g_{\text{IL}}/\|g_{\text{IL}}\|$, $\tilde{g}_{\text{RL}} \leftarrow g_{\text{RL}}/\|g_{\text{RL}}\|$
Initialize primal $\lambda \leftarrow \lambda_{\text{cur}}$, dual $\mu \leftarrow \mu_0$
**for** $k = 1$ to $K$ **do**
    $\tilde{g}_\lambda \leftarrow (1 - \lambda)\tilde{g}_{\text{RL}} + \lambda \tilde{g}_{\text{IL}}$
    $c_{\text{RL}} \leftarrow \langle \tilde{g}_{\text{RL}}, \tilde{g}_\lambda \rangle$, $c_{\text{IL}} \leftarrow \langle \tilde{g}_{\text{IL}}, \tilde{g}_\lambda \rangle$
    $J(\lambda, \mu) \leftarrow c_{\text{RL}} - \mu \max(0, -c_{\text{IL}})$
    $\lambda \leftarrow \lambda + \eta_\lambda \nabla_\lambda J$ {Gradient Ascent}
    $\lambda \leftarrow \min(\lambda_{\text{cur}}, \max(0, \lambda))$ {Project to $[0, \lambda_{cur}]$}
    $\mu \leftarrow \max(0, \mu + \eta_\mu \max(0, -c_{\text{IL}}))$ {Dual Update}
**end for**
**return** $\lambda$
```

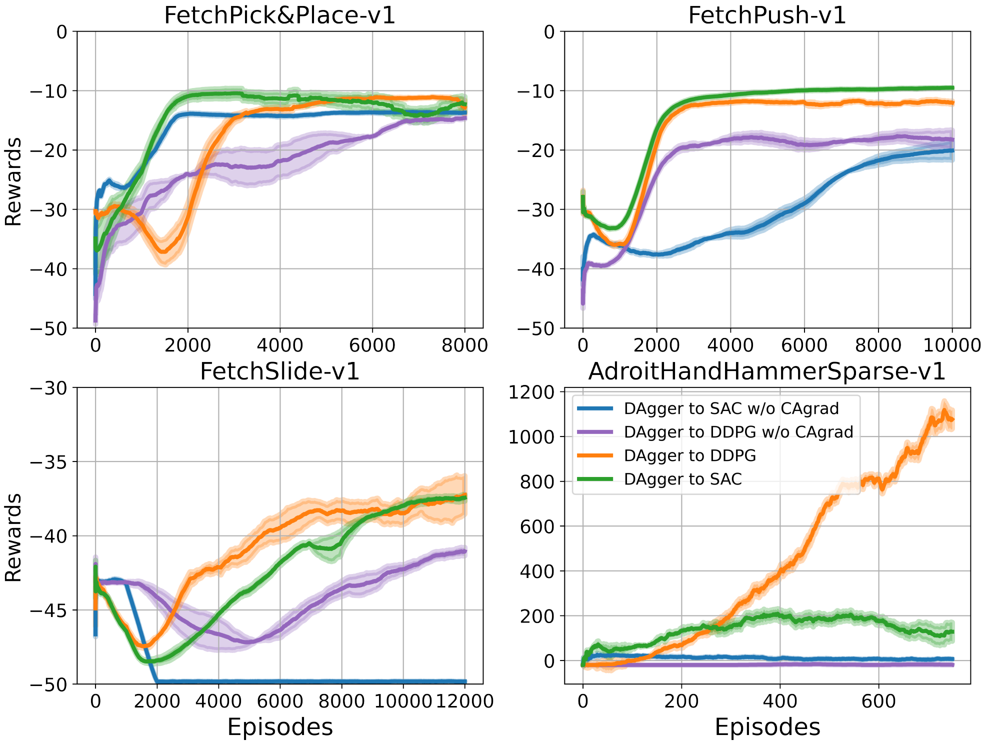


Figure 7: Ablation study: Impact of CAGrad on sample efficiency and asymptotic performance across domains. (Fetch: 1 ep = 1k steps, Adroit: 1 ep = 4k steps).

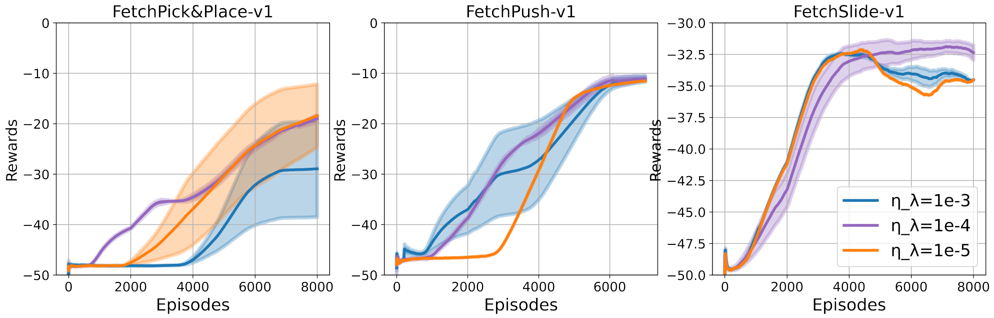


Figure 8: Sensitivity analysis: Effect of λ-optimizer learning rate (η_λ) on Fetch domains.

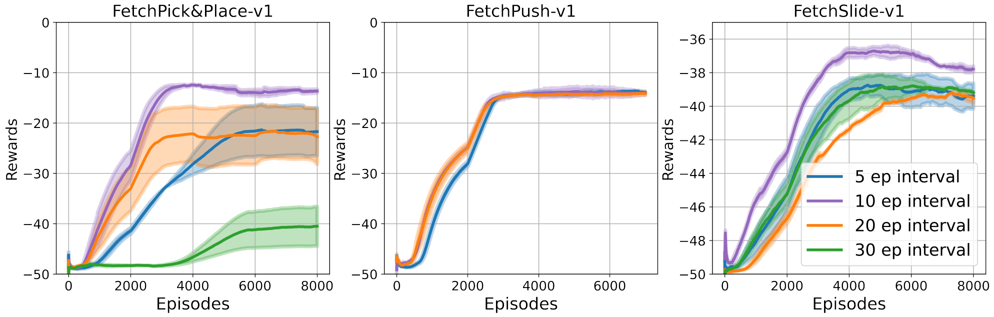


Figure 9: Sensitivity analysis: Effect of λ update interval on Fetch domains.

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

A.5.2 Hindsight Experience Replay (HER) with Demonstrations

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

A.5.3 Self-Adaptive Imitation Learning (SAIL)

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

A.5.4 Jump-Start Reinforcement Learning (JSRL)

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