Imitation Bootstrapped Multi-Agent Reinforcement Learning

Tony H. Zhang*
University of Utah
SLC, UT, USA
tony.h.zhang@utah.edu

Connor Mattson*
University of Utah
SLC, UT, USA
c.mattson@utah.edu

Daniel S. Brown
University of Utah
SLC, UT, USA
daniel.s.brown@utah.edu

Abstract

Demonstrations are an effective way to accelerate single-agent reinforcement learning (RL), but extending this idea to multi-agent reinforcement learning (MARL) is non-trivial as it is difficult and often impossible for a single person to teleoperate multiple agents simultaneously. Round-Robin Behavior Cloning (R2BC) addresses this by letting a single demonstrator control one agent at a time while the others execute their current policies. However, the resulting imitation policies are limited by the quality of the demonstrations. We propose Imitation Bootstrapped Multi-Agent Reinforcement Learning (IBMARL), a framework that extends Imitation Bootstrapped Reinforcement Learning (IBRL) to cooperative teams and uses R2BC to train the imitation policies. At each time step, every agent receives one action proposal from its imitation policy and one from its RL policy, and an ensemble of centralized critics selects among the per-agent combinations of the two. A gated imitation penalty, a soft Q-filtered behavior cloning loss, regularizes each RL policy toward its imitation policy only where the critics value the imitation action more, so that after training each agent executes its RL policy alone, without the centralized critics. On four cooperative multi-agent RL benchmarks, IBMARL surpasses its imitation policies 2.6–4.2× faster than demonstration-based baselines on two tasks, is the only method to surpass it on the other two, and exceeds the demonstrator itself on two of the four. With real human demonstrations IBMARL still surpasses imitation learning alone, and does so more than twice as fast as the best baseline. Code, rollout videos and full results are available at https://zhanghutony.github.io/ibmarl/.

Keywords

Multi-Agent Reinforcement Learning, Learning from Demonstration, Imitation Learning, Human-in-the-loop Learning

1 Introduction

Cooperative multi-agent tasks are difficult for Reinforcement Learning (RL) to solve: the joint-action space grows exponentially with the number of agents, credit assignment across agents is ambiguous, and rewards are often inherently sparse [2]. Even with a known model, finding an optimal joint policy for a finite-horizon Decentralized Partially Observable Markov Decision Process (Dec-POMDP) is NEXP-complete [4], so a learner that must also discover the reward through random joint exploration faces a search that grows with every agent added. In single-agent RL, demonstrations have been shown to help with this exploration problem, as examples of good behavior can often lead agents toward a reward signal even in sparse-reward tasks. Methods such as DQfD [12], DDPGfD [32], and DDPG + BC [20] seed the replay buffer or regularize the learner toward expert data, and Imitation Bootstrapped Reinforcement Learning (IBRL) [13] goes further by keeping a behavior-cloned policy in the loop, using it to propose actions during exploration and to improve the bootstrap target (Sec. 3.1).

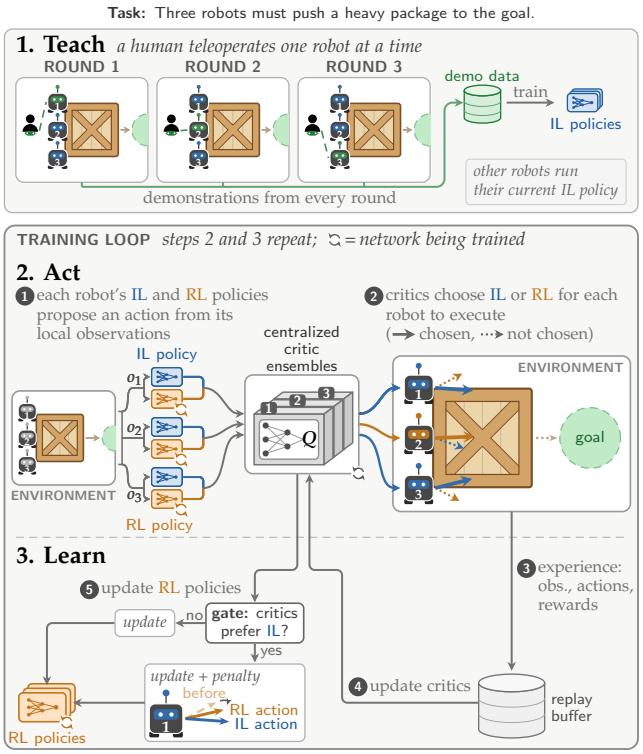


Figure 1: IBMARL overview. (1) Teach: a human teleoperates one robot at a time while the others run their current IL policies (R2BC), producing a frozen IL policy for each robot. **(2) Act:** each robot's IL and RL policies propose an action, and the centralized critic ensembles choose, per robot, which one the team executes. **(3) Learn:** the experience trains the critics, and each RL policy is pulled toward its IL action only where the critics prefer it. At deployment, each robot runs its RL policy alone.

We address two obstacles that block the extension of such frameworks to MARL: **(i)** *Demonstrations.* Nearly all multi-agent imitation learning (MAIL) assumes joint demonstrations, in which an expert must simultaneously command all N agents—data a lone human operator cannot realistically provide for collaborating teams [18, 30]. To address this, we leverage Round-Robin Behavior Cloning (R2BC) [18] (Sec. 3.2), which has the demonstrator cycle through controlling each agent of the team, while the other agents

*Authors contributed equally to this work.

execute their current learned policies, yielding decentralized cooperative policies from realistic single-human demonstrations. However, R2BC inherits the limitations of behavior cloning: covariate shift, a performance ceiling at the demonstrator, and a prohibitive demonstration burden for the last increments of task performance. **(ii)** *Decentralized execution.* IBRL's deployed policy relies on the critic at every step to choose between the IL action and the RL action. For a team, a critic that judges joint actions must have access to every agent's observation and action, so keeping it during deployment would require centralized execution or communication. To avoid this, the RL actors must be deployable on their own, but nothing in IBRL ensures that they are, and the RL policies lag behind the policy that was actually executed during training (Sec. 6.3).

We address both obstacles with **Imitation Bootstrapped Multi-Agent Reinforcement Learning (IBMARL)**, which keeps the R2BC policy in the loop rather than only in the data (Fig. 1). Assuming access to a task reward, we ask: *Can MARL be bootstrapped from policies trained from realistic single-agent demonstrations, and can the result be deployed without the critics?* During training (Fig. 1, Act), every agent receives one action from its imitation policy and one from its RL actor, and centralized critic ensembles choose, per agent, which to execute. The same choice is made inside the TD target, so the critics learn the value of the better of the two policies rather than of the RL actors alone. RL thus improves on the R2BC policy where it is wrong while still following it where it is right, so the demonstrator's ceiling of (i) is not inherited. For (ii), the critics' preference also sets, state by state, how strongly each policy is pulled toward its imitation action (Fig. 1, Learn), so the actors converge to the policy the critics execute and can be deployed alone, without the critics or the imitation policy.

The contributions of this work are: **(i)** We formulate **IBMARL**, a generalization of IBRL to cooperative Dec-POMDPs in which the imitation-learned (IL) policy is a set of N decentralized R2BC policies and a centralized critic ensemble arbitrates among per-agent combinations of IL and RL proposals. **(ii)** We introduce a **gated imitation penalty** that pulls each RL policy toward the IL proposal in proportion to the critics' preference for it, so that the actors alone match the centralized policy and execution is decentralized. **(iii)** We evaluate IBMARL on four cooperative tasks in VMAS [5] using synthetic demonstrators. Here, IBMARL surpasses the IL policies 2.6–4.2× faster than demonstration-based baselines on Navigation and 2.8× faster on the physically coupled Buzzwire task, is the only method to surpass the IL policies on Balance and Transport, and exceeds the performance of the demonstrator on Buzzwire and Balance. **(iv)** We evaluate IBMARL with a real human operator using R2BC's round-robin interface on Buzzwire, where IBMARL surpasses the human-trained IL policy more than twice as fast as the best baseline, suggesting that the gains carry over from synthetic to real demonstrators.

2 Related Work

Single-Agent RL with Demonstrations: A significant body of prior work combines expert demonstrations with single-agent RL. In the simplest instantiation, offline demonstrations pretrain the starting policy by behavior cloning [3, 24, 25] or, for off-policy methods, populate the replay buffer with the demonstrated transitions [12, 19]. Both remove much of the early random exploration and give the critic high-value transitions from the start, which stabilizes early training and raises early returns over RL alone [12, 14, 16, 24]. A second family treats the demonstrations as offline data and adds a surrogate loss that keeps the policy or critic close to the demonstrated actions: DQfD's large-margin term on the Double DQN loss [12, 31], which surpasses both imitation and RL alone in nearly all of 42 Atari games within a million steps, and similar extensions in DDPGfD [32], POfD [14], NAC [11], Hybrid RL [28] and TD3+BC [10]. When the reward is sparse, the demonstrations provide the signal: Nair et al. [20] add a behavior-cloning loss to the DDPG policy and, for suboptimal demonstrations, gate it with a *Q-filter* so that it applies only where the critic rates the demonstrated action above the actor's. IBRL [13] keeps the imitation policy itself in the loop, letting the critic choose between its proposal and the actor's at every step (Sec. 3.1). We propose IBMARL, the multi-agent generalization of IBRL; its gated imitation penalty is a soft, state-wise Q-filter applied to the IL policy's action on the states the learner visits rather than to stored demonstrations.

Multi-Agent RL with Demonstrations: Compared to the single-agent literature, MARL extensions that utilize demonstrations are few. Alagha et al. [1] let a synthetic expert take control of one agent during interaction so that part of the replay buffer is expert-provided, which improves early team performance; PRIMAL [27] trains decentralized policies by combining RL with imitation of a centralized planner; others reformulate particular MARL algorithms around demonstrations, namely MADDPG from demonstrations [23], skill learning from demonstrations [33] and hybrid training [36]; a separate line imitates opponents in competitive games [6]. The "-fD" regularizers above could in principle be added to QMIX [26], VDN [29], FACMAC [22] or LIIR [9], but to our knowledge MADDPG from demonstrations [23] is the only work that has done so. By far the most relevant prior work is PegMARL [35], which also avoids joint demonstrations: single-agent demonstrations, given with *no other agents present*, train a behavioral discriminator and a transition discriminator that shape the sparse team reward online, leaving all teaching of cooperation to the reward. IBMARL differs in three ways. Its demonstrations are given through R2BC in the presence of teammates, so the IL policy already encodes cooperation; the IL policy acts through action proposals, bootstrap targets and a critic-gated pull on the policy rather than through reward shaping or a scheduled cloning penalty [10, 20]; and the output is a set of decentralized actors that need neither the centralized critics nor the IL policy at run time.

3 Background

IBMARL combines IBRL, which utilizes an imitation policy with RL, with R2BC, which makes a cooperative imitation policy obtainable for multi-agent systems from one human demonstrator.

3.1 Imitation Bootstrapped RL

Imitation Bootstrapped Reinforcement Learning (IBRL) [13] is a single-agent off-policy actor-critic method that starts from a behavior-cloned *IL policy* μ_ψ, pretrained on a small set of demonstrations $\mathcal{D}$, and keeps it in the loop in two places. When acting, both policies

propose an action for the current observation o and the target critic $\bar{Q}_\phi$ picks the better one,

$$a = \underset{a' \in \{\mu_\psi(o),\, \pi_\theta(o)\}}{\arg\max} \bar{Q}_\phi(o, a'), \tag{1}$$

so early exploration follows the IL policy wherever the critic has found nothing better. The same choice is made at the next observation inside the TD target,

$$y = r + \gamma \max_{a' \in \{\mu_\psi(o'),\, \bar{\pi}_\theta(o')\}} \bar{Q}_\phi(o', a'), \tag{2}$$

so the critic estimates the value of the better of the two policies and value propagates from states the IL policy can reach before the policy can. The replay buffer is pre-loaded with $\mathcal{D}$, and the policy π_θ is trained with the ordinary deterministic policy gradient on Q_ϕ. With 10–50 demonstrations and a poor IL policy, IBRL reaches 85–100% success on real-robot manipulation tasks. Two features matter for what follows. First, the IL policy only ever supplies an action for the critic to pick or reject; it never enters the actor's loss, so the policy itself is not trained toward it. Second, the propose-and-select step is also used at test time, so that the critic and IL policies are required for deployment.

3.2 Round-Robin Behavior Cloning

Round-Robin Behavior Cloning (R2BC) [18] is an online multi-agent imitation learning method that learns a cooperative policy for N agents from one demonstrator who controls a single agent at a time. A session proceeds in rounds. In each round, the demonstrator controls one agent i through a teleoperation interface for one or more episodes while every other agent $j \neq i$ executes its current learned policy μ_{ψ_j}. Every step of the episode is recorded as a joint transition, and the controlled agent's pairs (o_i, a_i) form its cloning set $\mathcal{D}_i$; we also record the rewards observed at collection so that the transitions can later be replayed by RL. Control then passes to the next agent in round-robin order, and after each round every policy μ_{ψ_j} is updated by behavior cloning, supervised regression from o_j to a_j, on $\mathcal{D}_j$. Because the teammates run their current policies while a demonstration is given, the demonstrator reacts to the team as it actually behaves, which is what lets single-agent demonstrations produce cooperative behavior, and the data is collected on the state distribution the current team visits, which limits covariate shift. In the original work, 30–50 demonstrations per agent suffice for high task performance in simulation and on hardware. R2BC inherits the limits of behavior cloning: its performance is bounded by the demonstrator's, deployment noise can still take the team to states no demonstration covered, and the last increments of performance cost the most demonstrations. IBMARL addresses these with online RL under a known team reward, using IBRL's mechanism to decide, state by state, when the IL policy is still worth following.

4 Problem Formulation

Following prior work, we model a cooperative team of N agents as a partially observable stochastic game defined by the tuple $\langle \mathcal{S}, \{\mathcal{A}_i\}, \{\Omega_i\}, P, O, \{r_i\}, \gamma, T \rangle$. At step t, agent i receives a private observation $o_{i,t} \in \Omega_i$ of the state $s_t \in \mathcal{S}$ and selects a continuous action $a_{i,t} \in \mathcal{A}_i$; the joint action $\mathbf{a}_t = (a_{1,t}, \ldots, a_{N,t})$ moves the team to $s_{t+1} \sim P(\cdot \mid s_t, \mathbf{a}_t)$, and each agent receives a reward $r_{i,t}$, discounted by γ. Generally, these rewards may differ across agents. We write $\mathbf{o}_t = (o_{1,t}, \ldots, o_{N,t})$ for the joint observation. In our work, we study the cooperative case [21]: a decentralized policy $\pi = (\pi_1, \ldots, \pi_N)$, $\pi_i : \Omega_i \to \mathcal{A}_i$, is scored by its expected team return

$$J(\pi) = \mathbb{E}\Big[\tfrac{1}{N} \textstyle\sum_{t=0}^{T-1} \sum_{i=1}^{N} r_{i,t} \;\Big|\; a_{i,t} = \pi_i(o_{i,t}) \Big], \tag{3}$$

over episodes of at most T steps.

We utilize the outputs of R2BC (Sec. 3.2): The first is the *IL policy* $\mu_\psi = (\mu_{\psi_1}, \ldots, \mu_{\psi_N})$, $\mu_{\psi_i} : \Omega_i \to \mathcal{A}_i$, frozen once the session ends. The second is the *demonstration set* $\mathcal{D}$ of the session's recorded episodes: in each, the demonstrator, a person or a synthetic policy, controls one agent while the others act with the IL policies of that round, and every step is stored as a joint transition $(\mathbf{o}, \mathbf{a}, \mathbf{r}, \mathbf{o}')$ with $\mathbf{r} = (r_1, \ldots, r_N)$, so each joint action mixes one demonstrator action with actions of the then-current IL policies. The goal is to learn decentralized RL policies $\pi_\theta = (\pi_{\theta_1}, \ldots, \pi_{\theta_N})$ that surpass the IL policy, $J(\pi_\theta) > J(\mu_\psi)$, in as few environment steps as possible, and that maximize Eq. 3 within a limited environment interaction budget. Two assumptions define our application setting: **(i)** *A known reward.* The team reward is given, so the demonstrations are not needed to specify the task, only to make it learnable. The clearest motivating cases for IBMARL are environments with sparse rewards or limited reward signals, as we study in our experiments. **(ii)** *Centralized training, decentralized execution.* During learning, critics may condition on $(\mathbf{o}, \mathbf{a})$, but at deployment, each agent executes $\pi_{\theta_i}(o_i)$ alone, with no communication, critics or IL policy. This rules out IBRL's test-time arbitration (Sec. 3.1).

IBMARL in detail. Left:

5 Imitation Bootstrapped MARL

IBMARL extends IBRL (Sec. 3.1) to cooperative teams under centralized training with decentralized execution [17] (Fig. 2). As in IBRL, the IL policy proposes actions during exploration and for the bootstrap target and pre-populates the replay buffer; unlike IBRL, the choice is made per agent by a critic-scored *arbiter* over 2^N joint action candidates, and the same critics gate an imitation term that pulls the RL actors toward the IL actions, so that the actors can be deployed alone. Algorithm 1 summarizes the procedure.

5.1 Proposals and the Arbiter

Given a team observation $\mathbf{o} = (o_1, \ldots, o_N)$, each agent forms two candidate actions,

$$a_i^{\mathrm{RL}} = \pi_{\theta_i}(o_i) + \varepsilon_i, \qquad a_i^{\mathrm{IL}} = \mu_{\psi_i}(o_i) + \varepsilon_i', \tag{4}$$

with i.i.d. exploration noise $\varepsilon, \varepsilon' \sim \mathcal{N}(0, \sigma^2 I)$ on both proposals. The arbiter's candidate set is the product of the per-agent choices,

$$\mathcal{A}(\mathbf{o}) = \{a_1^{\mathrm{IL}}, a_1^{\mathrm{RL}}\} \times \cdots \times \{a_N^{\mathrm{IL}}, a_N^{\mathrm{RL}}\}, \tag{5}$$

so it may follow the IL policy for one agent and the RL policy for another. Each agent maintains an ensemble of E centralized critics $\{Q_{\Phi_i}^k(\mathbf{o}, \mathbf{a})\}_{k=1}^E$ with target copies $\bar{Q}_{\Phi_i}^k$. Following REDQ [7], the arbiter samples a subset $K \subseteq \{1, \ldots, E\}$ of the target critics, and scores each candidate $\mathbf{a} \in \mathcal{A}(\mathbf{o})$ by

$$\bar{Q}(\mathbf{o}, \mathbf{a}) \;=\; \sum_{i=1}^{N} \min_{k \in K} \bar{Q}_{\Phi_i}^k(\mathbf{o}, \mathbf{a})\,. \tag{6}$$

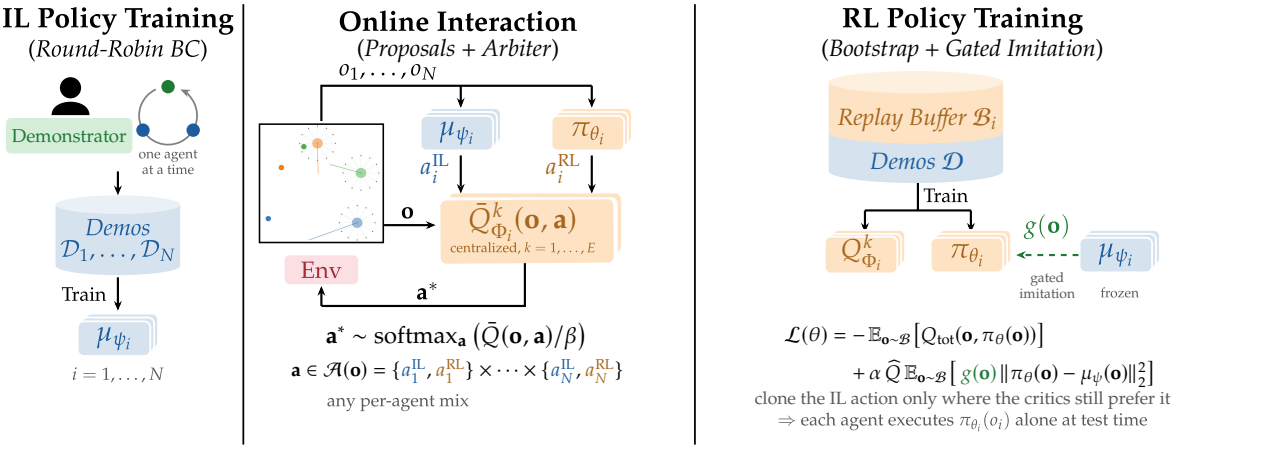


Figure 2: IBMARL Pipeline. Left: R2BC trains the decentralized IL policy μ_ψ from single-agent demonstrations (Sec. 3.2). **Middle:** the agents' IL and RL proposals form the 2^N joint candidates $\mathcal{A}(\mathbf{o})$, which the target critics score and the arbiter samples (Eqs. (5)–(6)). **Right:** the critics bootstrap through the same arbiter (Eq. (7)), and the actor loss adds the gated imitation term (Eqs. (8)–(9))

Algorithm 1 IBMARL

Require: frozen R2BC IL policies μ_{ψ_i} and demonstrations $\mathcal{D}$ (joint transitions), $i = 1, \ldots, N$
Require: actors π_{θ_i}; for each agent i, an ensemble of E centralized critics $\{Q^k_{\Phi_i}\}_{k=1}^E$; target copies $\bar{\pi}_{\theta_i}$, $\bar{Q}^k_{\Phi_i}$
Require: subset size $|K|$, temperature β, warm-up size W, update interval F, gradient steps G, minibatch size M, Polyak rate τ
Initialize the replay buffer $\mathcal{B} \leftarrow \mathcal{D}$
for each environment step **do**
  Form noised proposals $a_i^{\mathrm{RL}}, a_i^{\mathrm{IL}}$ for all i (4)
  Sample $K \subset \{1, \ldots, E\}$; score every $\mathbf{a} \in \mathcal{A}(\mathbf{o})$ with $\bar{Q}$ (6)
  $\mathbf{a}^* \sim \mathrm{softmax}\big(\bar{Q}(\mathbf{o}, \cdot)/\beta\big)$; execute $\mathbf{a}^*$; store the transition in $\mathcal{B}$
  if F steps have elapsed **and** $|\mathcal{B}| \geq W$ **then**
    for G gradient steps **do**
      Sample M transitions; arbitrate noise-free target proposals at $\mathbf{o}'$ to get $\mathbf{a}'^*$
      Update every critic $Q^k_{\Phi_i}$ toward y_i (7)
      Compute the gate $g(\mathbf{o})$; update actors (9), (8)
      Polyak-update target networks with rate τ
    end for
  end if
end for
return decentralized actors $\{\pi_{\theta_i}\}_{i=1}^N$

The team acts with a joint action sampled from a Boltzmann distribution over the candidates, $\mathbf{a}^* \sim \mathrm{softmax}_{\mathbf{a}\in\mathcal{A}(\mathbf{o})}\left(\bar{Q}(\mathbf{o}, \mathbf{a})/\beta\right)$ with temperature β (evaluation takes the argmax).

Per-agent mixing and credit assignment. Restricting the arbiter to the two pure joint actions, all-IL or all-RL, would reproduce IBRL's binary choice. Allowing mixed candidates does two things. It lets the team keep a reliable IL action for an agent whose RL policy is still poor while another agent, whose policy has improved, acts on its own, so the centralized policy improves one agent at a time rather than all at once. It also gives the critics a form of counterfactual information: the candidates in $\mathcal{A}(\mathbf{o})$ differ from one another in a single agent's action, so the scores of (Eq. 6) compare joint actions that isolate each agent's contribution at the current state. The single-agent demonstrations therefore supply a per-agent learning signal that a joint demonstration would not, which is the signal that credit assignment in cooperative MARL lacks [2]. The sum in (Eq. 6) resembles the additive decomposition of VDN [29], but each term conditions on the full joint observation and action, so no factorization is assumed; what makes the joint maximization tractable with continuous actions is that it ranges over the 2^N candidates rather than over the joint action space itself.

5.2 Learning

Critics are trained with a TD(0) target that bootstraps through the arbiter: at the next observation $\mathbf{o}'$, the arbiter is applied to noise-free proposals from the target actors $\bar{\pi}_\theta$ and the IL policy, yielding a bootstrap action $\mathbf{a}'^*$, and every ensemble member regresses to

$$y_i = r_i + \gamma\,(1 - \mathbb{1}[\text{terminal}])\, \min_{k\in K} \bar{Q}^k_{\Phi_i}(\mathbf{o}', \mathbf{a}'^*), \tag{7}$$

so members differ only by initialization. This is the multi-agent form of (Eq. 2): the target evaluates the best arbitrated candidate at $\mathbf{o}'$ rather than the actors' own joint action.

gated imitation penalty. The actors are trained with the deterministic policy gradient through the centralized critics, as in MADDPG [15, 17], but on the team objective $Q_{\mathrm{tot}}(\mathbf{o}, \pi_\theta(\mathbf{o})) = \sum_i Q^1_{\Phi_i}(\mathbf{o}, \pi_\theta(\mathbf{o}))$, in which every actor's current action enters the joint action and one ensemble member per agent suffices. This gradient only climbs the critics locally around the actors' own actions. When the arbiter executes IL actions, their successes raise the critics' value of those actions, but nothing in the gradient moves the actors toward them, so the RL actors lag behind the centralized policy that is actually executed (Sec. 6.3). We therefore add an imitation penalty to the policy loss:

$$\mathcal{L}(\theta) = -\mathbb{E}\left[Q_{\mathrm{tot}}(\mathbf{o}, \pi_\theta(\mathbf{o}))\right] + \alpha\hat{Q}\,\mathbb{E}\left[g(\mathbf{o})\|\pi_\theta(\mathbf{o}) - \mu_\psi(\mathbf{o})\|_2^2\right], \tag{8}$$

where the gate $g(\mathbf{o})$ is the critics' preference for the IL policy at $\mathbf{o}$, the Boltzmann (sigmoid) probability of the pure IL joint action over the actor's noise-free joint action under the scoring of (Eq. 6),

$$g(\mathbf{o}) = \sigma\big(\left[\bar{Q}(\mathbf{o}, \mu_\psi(\mathbf{o})) - \bar{Q}(\mathbf{o}, \pi_\theta(\mathbf{o}))\right]/\beta_g\big), \tag{9}$$

with gate temperature β_g. $\hat{Q}$ is the detached minibatch mean of $|Q_{\text{tot}}|$, so that α is dimensionless and transfers across tasks with different return scales, as in TD3+BC [10]. The penalty clones the IL action only where the critics still prefer it and switches itself off, state by state, as the policy overtakes the IL policy. It needs no annealing schedule and acts on the states the learner visits rather than on the demonstration set. It is in this sense a soft version of the Q-filter of Nair et al. [20], with the IL policy standing in for the stored demonstrations, and it recovers the hard filter as $\beta_g \to 0$.

Two properties follow from (Eq. 9). First, the term acts only where the policy differs from the IL policy and when the critics prefer the IL action. These are the exact states where the arbiter would override the actor, as elsewhere either the squared distance or the gate vanishes. Second, the gate uses the same target critics and scoring as the arbiter, so the policy is pulled toward the action the team would execute at that state, never toward one the arbiter would reject. The actors are therefore trained to reproduce the actions the arbiter would execute, so after training each agent can run $\pi_{\theta_i}(o_i)$ alone.

5.3 Data and Implementation

A single replay buffer, shared by all agents, is pre-loaded with $\mathcal{D}$ and sampled uniformly: online transitions gradually dilute the demonstrations. Each stored transition carries the per-agent rewards $\mathbf{r}$, and agent i's critics regress to the target of (Eq. 7) built from its own r_i. Updates begin after the buffer holds W transitions. Optionally, as in IBRL, the IL policy's noised proposal is executed unconditionally during warm-up so that the first arbitrated decisions follow observed successes. Relative to IBRL, the only new hyperparameters are the weight α, made scale-free by the normalization of (Eq. 8), and the gate temperature β_g.

6 Experiments with Synthetic Demonstrations

6.1 Setup

Tasks. We evaluate on the *Navigation*, *Buzzwire*, *Balance* and *Transport* tasks of the Vectorized Multi-Agent Simulator (VMAS) [5] (Fig. 3, top). In *Navigation*, N=3 agents and their assigned goals spawn at random in a bounded 2-D workspace. Each agent observes its own state, its goal's relative position, and a LiDAR scan of its teammates. In *Buzzwire*, N=2 agents are connected to a ball and must thread it through a narrow corridor to a goal without touching the walls. Each agent observes its own position, velocity and relative position to the goal location. In *Balance*, N=3 agents support a rigid beam to lift a spherical package resting on top to a goal without the package or beam touching the floor. Each agent observes its own position and velocity, its position relative to the package and to the beam, the package's position relative to the goal, the package's velocity, and the beam's velocity, angular velocity and rotation. In *Transport*, N=3 agents must collaborate to push a package onto a goal. Each agent observes its own state and the package's state relative to the goal and to itself. In all four, agents act through continuous velocity commands over a fixed horizon T.

Rewards. The native VMAS rewards of Navigation, Balance, and Buzzwire are dense, which poorly reflects the difficulty of real world reward signals, which are often goal-conditioned and sparse. We replace these environment rewards with sparse variants, $r_{i,t} = \tilde{r}_{i,t}$ if $d_{i,t} < \delta$ and -1 otherwise, where $d_{i,t}$ is agent i's distance to its goal (Navigation) or the ball's distance to the goal, shared by both agents (Buzzwire), and $\tilde{r}_{i,t}$ is the native VMAS reward. Buzzwire episodes run the full horizon and Navigation episodes end early only once every agent is on its goal, so an agent's return counts the steps spent outside the goal region and is bounded below by $-T$. Balance uses a binary success reward: the team receives 1 on the step the package reaches the goal, which ends the episode, and 0 otherwise; the episode also ends if the package or the beam touches the floor, so the return is the success indicator. Transport keeps its native VMAS reward, the per-step decrease of the package's distance to the goal, which is already a difficult exploration signal: the agents must explore an open environment and make contact with the package before any reward is obtained. On both, the difficulty is coordination: one agent cannot lift the beam, and one agent moves the package too slowly to reach the goal within the horizon.

Experts, IL policies and demonstrations. On each task the IL policy is the decentralized policy of one R2BC training session (one small MLP per agent), trained from demonstrations given through R2BC's round-robin protocol by a synthetic demonstrator, the task's ground-truth *expert*: a PPO policy trained with the dense reward on Navigation, a MAPPO [34] policy on Balance, and scripted controllers that hold each agent at a fixed lateral offset while driving toward the goal on Buzzwire and line the agents up behind the package on Transport. The experts themselves are far from perfect: the Transport expert delivers the package in 28% of its episodes and the Balance expert lifts it to the goal in 37%. Their returns are listed in the first two rows of Tab. 1. All demonstration-based methods consume the same transitions and IL policy.

Implementation. Actors and critics are per-agent MLPs with two hidden layers of 256 units with tanh activations. IBMARL uses $E = 3$ critics with $|K| = 2$, arbiter temperature $\beta = 0.05$ and imitation weight of $\alpha = 0.4$. All networks are trained with Adam at a learning rate of 10^{-4}, batch size 1024, $\gamma = 0.99$, Polyak $\tau = 10^{-3}$, exploration noise $\sigma = 0.1$ and gradient norm clipping at 1.0. On Buzzwire the IL policy is executed unconditionally during warm-up. **Baselines** share IBMARL's architectures, DDPG-style updates [15], exploration noise and hyperparameters, with a single critic per agent, so that *how* demonstrations are used is the only variable. **MADDPG** [17] is the demonstration-free reference. **RLfD** [32] keeps the demonstrations in a separate replay buffer from which each minibatch draws a fraction annealed linearly from 50% to 0% over training (over the first two thirds on Balance). **RFT** adds to the policy loss a behavior-cloning regularizer on demonstration states, $\lambda_{BC}\,\lambda(t)\,\|\pi_\theta(o^{\mathcal{D}}) - a^{\mathcal{D}}\|_2^2$ [20, 24], with $\lambda_{BC} = 0.1$ and $\lambda(t)$ annealed from 1 to 0 over the first half of training (the whole run on Transport, the first 27% on Balance): the pull of (Eq. 8) without the gate, on demonstration states and a fixed schedule.

Evaluation Protocol. Policies are evaluated periodically without exploration noise on a dedicated environment. Every method, IBMARL included, is evaluated with its RL actors alone, the decentralized policy each agent would deploy; for IBMARL we also report the *centralized* policy, greedy arbitration between the IL and RL proposals, which needs the centralized critics and the IL policy and is therefore a training-time policy, not a deployable one. Returns are smoothed with a centered window of evaluations; intervals are ±1 s.e.m. over 5 seeds, finals average each seed's last 10

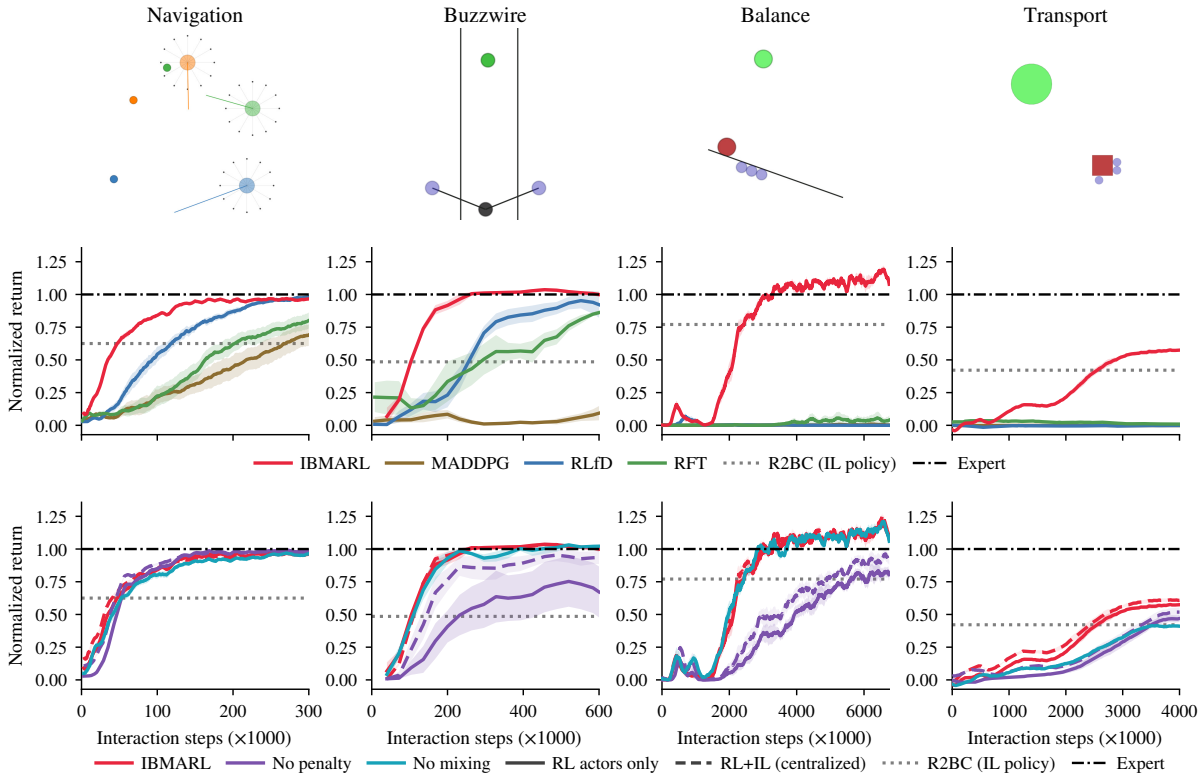


Figure 3: Synthetic demonstrations. Top: the four VMAS tasks (Navigation adapted from [18]). **Middle:** normalized evaluation return of the RL actors (mean ± s.e.m., 5 seeds); dotted line: frozen R2BC IL policy; dash-dotted line: the expert demonstrator. **Bottom:** ablations; solid curves are the RL actors alone, dashed the centralized (arbitrated) policy.

evaluations, and steps to a reference level are the first evaluation whose smoothed return exceeds it, median over seeds, with the number of seeds that ever reach it where fewer than five do. To compare tasks on one axis, Fig. 3 reports the *normalized return* $(R - R_{\text{floor}})/(R_{\text{expert}} - R_{\text{floor}})$, where the floor is the worst attainable return, $-T$ on Navigation and Buzzwire and 0 on Balance and Transport, so that 0 is the floor and 1 is the expert. Per-seed results and Fig. 3 in raw return units are given in Appendix C.

6.2 Comparison with Baselines

Navigation. IBMARL's actors surpass the IL policy after a median of 47k environment steps—2.6× fewer than RLfD (122k) and 4.2× fewer than RFT (198k), while MADDPG reaches it in only 3 of 5 seeds (Fig. 3, Tab. 1). This is the easiest of the four tasks: the goal regions are large enough that undirected exploration eventually finds them, which is why demonstration-free MADDPG makes steady progress and why, given the full budget, replay-based RLfD (-19.6 ± 0.5) ends slightly above IBMARL (-20.9 ± 0.8), both within 0.04 of the expert in normalized return. RFT (-35.0 ± 4.9) and MADDPG (-43.9 ± 6.7) remain well below. The demonstration-based methods therefore separate on speed rather than on the final level: IBMARL reaches the IL level in a sixth of its budget because the arbiter can execute IL proposals from the first step, so the early replay buffer already holds goal-reaching transitions, whereas RLfD must first improve its actors through the critic before its own rollouts reach the goals.

Buzzwire. This task is considerably harder: the ball must be carried through the corridor without contact, so random joint exploration essentially never observes a reward, and MADDPG never reaches the goal in any seed (-194.6 ± 2.5). IBMARL's actors surpass the IL policy after 104k steps, 2.8× fewer than RLfD and RFT (296k each), and finish at -68.9 ± 0.7 against -85.9 ± 8.1 for RLfD and -111.2 ± 4.9 for RFT. The curves in Fig. 3 show where the difference comes from: the baselines stay at the floor for the first 150–200k steps, the time it takes the critics to propagate value from the demonstration transitions back to the start of the corridor and the actors to follow it, whereas IBMARL executes the IL proposals during warm-up and so observes successes from the beginning. IBMARL is also the first to reach the expert's level, after 264k steps in every seed; RLfD reaches it after 552k steps in 4 of 5 seeds and RFT in 1 of 5, and both have a wide spread across seeds where IBMARL's is tight.

Balance. None of the baselines learns the task within 6.75×10^6 steps: RLfD and MADDPG stay at the floor and RFT barely above it (0.014 ± 0.014). The reward is +1 upon success (zero elsewhere), and an episode ends the moment the package or the beam touches the floor, so a team that does not already know how to lift together drops the package within a few steps and never sees a reward; the

Table 1: Synthetic demonstrations: environment steps until the RL actors surpass the IL policy (median over 5 seeds; a fraction gives the seeds that ever surpass it) and final return (mean ± s.e.m. over 5 seeds of each seed's last 10 evaluations).

	Navigation		Buzzwire		Balance		Transport	
Method	steps	final	steps	final	steps	final	steps	final
R2BC (IL policy)	–	−48.7 ± 1.2	–	−137.5 ± 1.8	–	0.287 ± 0.048	–	15.7 ± 1.0
Expert (demonstrator)	–	−17.9 ± 0.4	–	−71.2 ± 1.4	–	0.372 ± 0.009	–	37.2 ± 0.4
MADDPG	3/5	−43.9 ± 6.7	never	−194.6 ± 2.5	never	0.001	never	−0.05 ± 0.01
RLfD	122k	**−19.6 ± 0.5**	296k	−85.9 ± 8.1	never	0.000	never	−0.06 ± 0.10
RFT	198k	−35.0 ± 4.9	296k	−111.2 ± 4.9	never	0.014 ± 0.014	never	0.35 ± 0.09
IBMARL	**47k**	−20.9 ± 0.8	**104k**	**−68.9 ± 0.7**	**2.33M**	**0.423 ± 0.017**	**2.63M**	**21.3 ± 0.3**
IBMARL (centralized)	–	−21.4 ± 0.9	–	−68.8 ± 1.2	–	0.425 ± 0.012	–	22.7 ± 0.6

demonstration transitions give the critics some positive targets, but the actors cannot reproduce the coordinated lift from them. IBMARL's actors instead surpass the IL policy (0.287) after 2.33M steps, reach the expert's level (0.372) after 3.19M steps in every seed, and end at 0.423 ± 0.017, a success rate of 42% and a normalized return of 1.14. The IL policy succeeds in 29% of its episodes, so the arbiter's proposals keep producing successful trajectories on the states the learner actually visits, and the critics learn which of them to keep. The demonstrator is suboptimal but is enough to make the task learnable by IBMARL.

Transport. Within $4{\times}10^6$ steps none of the baselines improves beyond its initial performance: the agents must first find the package in an open workspace and push it in a consistent direction before any reward arrives, and one agent alone moves it too slowly to reach the goal, so no baseline learns to push together. IBMARL's actors surpass the IL policy (15.7) after $2.63{\times}10^6$ steps and end at 21.3 ± 0.3 (centralized policy 22.7 ± 0.6). Learning is slow because the IL policy itself is weak: a normalized return of 0.42 against an expert that delivers in only 28% of its episodes. The arbiter's proposals are often wrong for some agents, but progress comes from the critics gradually learning which agents' IL actions to keep and which actors have surpassed them. Transport is the only task with a large remaining gap to the expert (normalized return 0.57).

Summary. IBMARL is the fastest to the IL level in all cases, the only method that learns Balance and Transport, and the only one that exceeds the expert (Buzzwire, 1.02; Balance, 1.14). The one place a baseline ends ahead is Navigation, where RLfD catches up once exploration is plentiful.

6.3 Ablations

We conduct ablation tests to justify the individual components of IBMARL. Two ablations isolate the two additions IBMARL makes to IBRL: *No mixing* is IBRL's binary all-IL/all-RL arbiter applied to the team, and *No penalty* is IBRL's objective. Together they measure what the multi-agent setting adds. Fig. 3 (bottom) plots both the RL actors alone (solid) and the centralized, arbitrated policy (dashed), and Tab. 2 gives the final returns.

The gated imitation penalty. On Navigation the term is neutral: both ablations match IBMARL throughout, the actors never lag the centralized policy, so the gate has nothing to close, and *No penalty* (−19.4 ± 0.5) in fact ends marginally above IBMARL. On the three harder tasks the term is decisive, in two different ways. On

Table 2: Ablations: final return (mean ± s.e.m., 5 seeds) of the RL actors and, where it differs, of the centralized policy.

	Navigation	Buzzwire	Balance	Transport
IBMARL	−20.9 ± 0.8	−68.9 ± 0.7	0.423 ± 0.017	21.3 ± 0.3
centralized	−21.4 ± 0.9	−68.8 ± 1.2	0.425 ± 0.012	22.7 ± 0.6
No penalty	−19.4 ± 0.5	−111.6 ± 24.0	0.333 ± 0.015	17.3 ± 1.0
centralized	−19.8 ± 0.3	−81.6 ± 6.2	0.347 ± 0.012	19.2 ± 1.4
No mixing	−21.5 ± 0.8	−71.2 ± 3.0	0.422 ± 0.022	15.3 ± 0.6

Buzzwire it closes the *policy lag* of Sec. 5: without it the centralized policy still crosses the IL level and ends at −81.6 ± 6.2, but the RL actors alone cross in only 4 of 5 seeds and end at −111.6 ± 24.0, below RLfD. The arbiter keeps choosing the IL action in the corridor, the critics keep rewarding it, and nothing in the deterministic policy gradient moves the actors toward it, so the deployable policy stays worse than the training-time policy. With the term the policy and centralized curves coincide, and the critics and the IL policy can be discarded at deployment. On Balance and Transport the term improves the centralized policy itself, not only the actors: *No penalty* ends at 0.333 ± 0.015 (centralized 0.347 ± 0.012) on Balance, below the expert, and crosses the IL level later than IBMARL does (Fig. 3); on Transport it ends at 17.3 ± 1.0 (centralized 19.2 ± 1.4) against 21.3 ± 0.3 (22.7 ± 0.6), and the centralized policy stays ahead of the actors throughout. Here the pull toward the IL action on the states the learner visits is what lets the actors reach, and then improve on, the coordinated behavior the arbiter was executing, and better actors in turn give the arbiter better RL proposals to choose from.

Per-agent mixing. Mixing is neutral on Navigation, Buzzwire and Balance and decisive on Transport, where *No mixing* ends at 15.3 ± 0.6, the IL policy's own level, against 21.3 ± 0.3 for IBMARL. On Buzzwire the result is expected: with two agents, mixing adds only two candidates to the pure pair. Balance shows that three agents do not by themselves make mixing useful. The difference is in what the IL policy knows: on Transport the expert succeeds rarely and the IL policy is often right for some agents and wrong for others at the same state, so the arbiter gains from keeping the useful parts of the IL action while letting improved actors act for the rest; on Balance the IL policy tends to be right or wrong for the whole team at once, since all three agents must lift together, and the pure candidates suffice.

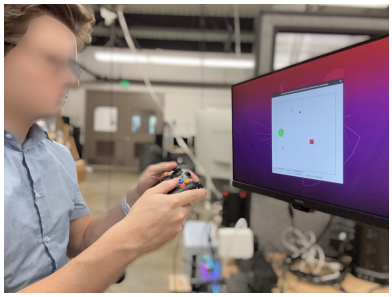


Figure 4: Human demonstrations are given through R2BC's teleoperation interface: the demonstrator drives one agent with a gamepad while the others execute their current policies, and control cycles through the team.

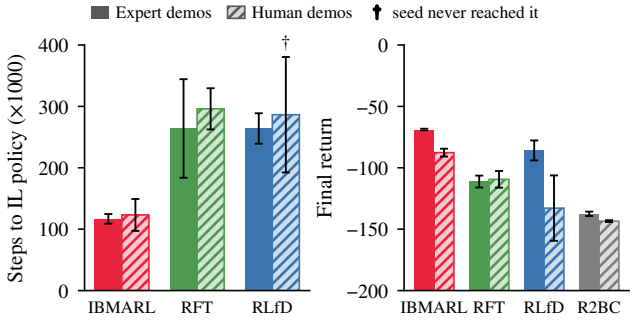


Figure 5: Buzzwire with expert and human demonstrations: environment steps until the RL actors surpass the IL policy (left) and final return (right); mean ± s.e.m. over 5 seeds. †: one RLfD seed never surpasses the IL policy. R2BC is the frozen IL policy trained from each demonstration set.

7 Experiments with Human Demonstrations

The synthetic demonstrators above are consistent and never tire, and the Buzzwire and Transport controllers, in particular, encode a cooperative strategy that a person may not reproduce exactly. R2BC exists to make human demonstrations for teams practical, so we repeat the Buzzwire experiment with demonstrations given by a real human. Using Buzzwire allows us to test human demonstrations for a physically coupled task, where the demonstrator must keep the ball off the walls while a partner they do not control pulls on it, requiring demonstrations that depend on the teammate's behavior.

Setup. A single demonstrator (one of the authors) gives demonstrations through R2BC's teleoperation interface (Fig. 4), which renders the environment, indicates which agent is currently under the demonstrator's control, and runs the remaining agent on its current R2BC policy. The demonstrator commands the controlled agent's velocity with the analog stick of an Xbox gamepad, and control passes to the other agent in round-robin order, as in R2BC; between rounds the R2BC policies are updated on the data collected so far, so later demonstrations are given alongside a progressively better teammate. The budget is 24 episodes (4,800 transitions), twice the scripted budget, of which only 13 reach the goal. The resulting frozen IL policy attains -143.4 ± 0.8, slightly below the expert-trained IL policy's -137.5 (normalized 0.44 against 0.49). All demonstration-based methods consume the same human transitions and the same human-trained IL policy; the baselines, hyperparameters and protocol are unchanged from Sec. 6.1.

Results. The ordering of methods is the same as with synthetic demonstrations (Fig. 5; here steps are means over seeds to match the figure). IBMARL's actors surpass the human IL policy after 123k ± 26k environment steps in every seed, barely later than with synthetic demonstrations (117k±8k), against 296k±34k for RFT and 286k ± 94k for RLfD, which fails to surpass it in one seed; IBMARL is thus 2.3–2.4× faster than the either baseline. At the end of training IBMARL reaches -87.7 ± 3.2, ahead of RFT (-109.4 ± 6.8) and RLfD (-132.9 ± 26.6), and its seeds agree closely, whereas RLfD's spread is wide. Human demonstrations cost every method some final return, which tracks the lower quality of the human demonstrator, but they cost the baselines more: RLfD, which replays the inconsistent demonstration transitions, falls from -85.9 to -132.9, close to the IL policy itself, while IBMARL falls from -68.9 to -87.7 and keeps its lead in both speed and final return; RFT, which clones the demonstration actions on a fixed schedule, is equally poor with either demonstrator. The difference is the gate: inconsistent demonstrations are followed where the critics still rate them above the actors and ignored elsewhere.

8 Discussion

Why IBMARL works. The demonstration-based baselines use the demonstrations only as a stored dataset. IBMARL stores the same dataset but also keeps a cloned IL policy acting throughout the training loop. On Balance and Transport, this policy is the difference between learning and not learning within our budget. The actors are also not bound by the demonstrator: on Buzzwire and Balance they end up outperforming it, so the demonstrator's ceiling, the central limitation of R2BC (Sec. 3.2), is not inherited.

Scaling. IBMARL uses the learned critics to score all 2^N joint action candidates. In practice, this only requires one batched forward pass for $N \leq 3$, but this approach may not scale for large teams. Because every agent's choice is binary, future work should explore using a greedy linear sweep over agents or a random subset of joint action candidates to replace a full enumeration.

Demonstration and deployment cost. Demonstrations cost is only 12 to 500 episodes of single-agent teleoperation, compared with hundreds of thousands to millions of steps for RL. IBMARL bootstraps from this small amount of human data to achieve significant speedups in MARL. Importantly, at deployment the team needs only the per-agent actors: IBMARL requires no communication, no centralized critic and no copy of the IL policy.

Limitations and future work. We highlight three limitations that we believe could lead to exciting avenues for future work. First, all results are in simulation. Future work should explore experiments with both real humans and real robots. Second, we assume the team reward is known. While this is a standard assumption in MARL, we note that a reward learned from human preferences [8] could easily replace the ground-truth reward without changing our method. Importantly, this could let a single person supply both the demonstrations and the reward. Third, the IL policy is frozen

once RL begins. In the future, the demonstrator could keep teaching during RL and leverage the learned critics to find where the team still needs help; closing the loop would allow a single demonstrator to iteratively fine-tune the behavior of a team of robots, with the demonstrator's effort being spent where it matters.

9 Conclusion

Our results demonstrate that realistic, single-human demonstrations can be used to effectively bootstrap multi-agent RL. Crucially, these demonstrations act as a starting point for MARL, rather than a ceiling. Our results also provide evidence that an imitation policy is more useful as a participant than as a data generator. Rather than simply populating a replay buffer, we find value in using an imitation policy to propose actions to be scored by an ensemble of critics. This proposal framework allows them to decide when to imitate and when to diverge from the demonstrations from any state in the environment. We see IBMARL as an important step towards multi-robot teams that one person can effectively teach but that also continue to improve even after the demonstrations end.

References

[1] Ahmed Alagha, Rabeb Mizouni, Jamal Bentahar, Hadi Otrok, and Shakti Singh. 2023. Multiagent Deep Reinforcement Learning with Demonstration Cloning for Target Localization. *IEEE Internet of Things Journal* 10, 15 (2023), 13556–13570.

[2] Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. 2024. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches.* MIT Press.

[3] Lars Ankile, Zhenyu Jiang, Rocky Duan, Guanya Shi, Pieter Abbeel, and Anusha Nagabandi. 2025. Residual Off-Policy RL for Finetuning Behavior Cloning Policies. *arXiv preprint arXiv:2509.19301* (2025).

[4] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research* 27, 4 (2002), 819–840.

[5] Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. 2022. VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning. In *International Symposium on Distributed Autonomous Robotic Systems (DARS).*

[6] The Viet Bui, Tien Mai, and Thanh Hong Nguyen. 2024. Mimicking To Dominate: Imitation Learning Strategies for Success in Multiagent Games. In *Advances in Neural Information Processing Systems (NeurIPS).*

[7] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. 2021. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. In *International Conference on Learning Representations (ICLR).*

[8] Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems (NeurIPS).*

[9] Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. 2019. LIIR: Learning Individual Intrinsic Reward in Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS).*

[10] Scott Fujimoto and Shixiang Shane Gu. 2021. A Minimalist Approach to Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS).*

[11] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. 2018. Reinforcement Learning from Imperfect Demonstrations. *arXiv preprint arXiv:1802.05313* (2018).

[12] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. 2018. Deep Q-Learning from Demonstrations. In *AAAI Conference on Artificial Intelligence.*

[13] Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. 2024. Imitation Bootstrapped Reinforcement Learning. In *Robotics: Science and Systems (RSS).*

[14] Bingyi Kang, Zequn Jie, and Jiashi Feng. 2018. Policy Optimization with Demonstrations. In *International Conference on Machine Learning (ICML).* 2469–2478.

[15] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous Control with Deep Reinforcement Learning. In *International Conference on Learning Representations (ICLR).*

[16] Haochen Liu, Zhiyu Huang, Jingda Wu, and Chen Lv. 2022. Improved Deep Reinforcement Learning with Expert Demonstrations for Urban Autonomous Driving. In *IEEE Intelligent Vehicles Symposium (IV).* 921–928.

[17] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems (NeurIPS).*

[18] Connor Mattson, Varun Raveendra, Ellen Novoseller, Nicholas Waytowich, Vernon J. Lawhern, and Daniel S. Brown. 2026. R2BC: Multi-Agent Imitation Learning from Single-Agent Demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA).*

[19] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. 2020. AWAC: Accelerating Online Reinforcement Learning with Offline Datasets. *arXiv preprint arXiv:2006.09359* (2020).

[20] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming Exploration in Reinforcement Learning with Demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA).* 6292–6299.

[21] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs.* Springer.

[22] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. FACMAC: Factored Multi-Agent Centralised Policy Gradients. In *Advances in Neural Information Processing Systems (NeurIPS).*

[23] Y. Qiu, Y. Jin, L. Yu, J. Wang, Y. Wang, and X. Zhang. 2023. Improving Sample Efficiency of Multiagent Reinforcement Learning With Nonexpert Policy for Flocking Control. *IEEE Internet of Things Journal* 10, 16 (2023), 14014–14027.

[24] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2018. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Robotics: Science and Systems (RSS).*

[25] Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. 2023. PIRLNav: Pretraining with Imitation and RL Finetuning for ObjectNav. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 17896–17906.

[26] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML).* 4292–4301.

[27] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, T. K. Satish Kumar, Sven Koenig, and Howie Choset. 2019. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics and Automation Letters* 4, 3 (2019), 2378–2385.

[28] Yuda Song, Yifei Zhou, Ayush Sekhari, J. Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. 2023. Hybrid RL: Using Both Offline and Online Data Can Make RL Efficient. In *International Conference on Learning Representations (ICLR).*

[29] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks for Cooperative Multi-Agent Learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS).* 2085–2087.

[30] Enrico Turco, Cristian Castellani, Valerio Bo, Claudio Pacchierotti, Domenico Prattichizzo, and Tommaso Lisini Baldi. 2024. Reducing Cognitive Load in Teleoperating Swarms of Robots through a Data-Driven Shared Control Approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

[31] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI Conference on Artificial Intelligence.*

[32] Matej Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. 2017. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. *arXiv preprint arXiv:1707.08817* (2017).

[33] V. L. N. Venkatesh and Byung-Cheol Min. 2024. Learning from Demonstration Framework for Multi-Robot Systems Using Interaction Keypoints and Soft Actor-Critic Methods. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 10754–10761.

[34] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems (NeurIPS).*

[35] Peihong Yu, Manav Mishra, Alec Koppel, Carl Busart, Priya Narayan, Dinesh Manocha, Amrit Bedi, and Pratap Tokekar. 2025. Beyond Joint Demonstrations: Personalized Expert Guidance for Efficient Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2403.08936* (2025).

[36] M. Zhang, S. Su, C. He, and Guillaume Sartoretti. 2024. Hybrid Training for Enhanced Multi-Task Generalization in Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2408.13567* (2024).