Learning Robust Helpful Behaviors in Two-Player Cooperative Atari Environments

Paul Tylkin
Harvard University
ptylkin@g.harvard.edu

Goran Radanovic
MPI-SWS*
gradanovic@mpi-sws.org

David C. Parkes
Harvard University
parkes@eecs.harvard.edu

Abstract

We initiate the study of helpful behavior in the setting of two-player Atari games, suitably modified to provide cooperative incentives. Our main interest is to understand whether reinforcement learning can be used to achieve robust, helpful behavior— where one agent is trained to help a second, partner agent. Robustness requires the helpful AI to be able to cooperate effectively with a diverse set of partners. We study this question with both artificial partner agents as well as human participants (introducing a new, web-based framework for the study of human-with-AI behavior). We achieve positive results in both Space Invaders and Fall Down, as well as successful transfer to human partners, including with people who are asked to deliberately follow unexpected behaviors.

1 Introduction

As we anticipate a future of systems of AIs, interacting in ever-increasing ways and with each other as well as with people, it is important to develop methods to promote cooperation. This need for cooperation is relevant, for example, in settings with automated vehicles [1], home robotics [14], as well as in military domains, where UAVs assist teams of soldiers [31].

In this paper, we seek to advance the study of cooperative behavior through suitably modified, two-player Atari games. Although closed and relatively simple environments, there is a rich, recent tradition of using Atari to drive advances in AI [23, 24]. Atari games are designed to be fun for people to play, are challenging enough to test AI methods, and also provide a rich landscape of domains that has been well-studied in single-player settings [36, e.g.]. To the best of our knowledge, Atari games have not been used so far as a test-bed for the study of cooperative behavior. Here, we modify two-player Atari games to provide them with cooperative incentives. For two-player Space Invaders, for example, we reconfigure the game dynamics so that players maximize the joint score and so there is no bonus for loss of life of the other player.[2]

Our main interest is to understand whether reinforcement learning can be used to achieve *helpful behavior*— where one agent is trained to help a second, partner agent. We seek robust helpful behavior: the helpful AI should be able to cooperate effectively with a diverse set of partners. We study this question with both artificial partner agents as well as human participants (introducing a new, web-based framework for the study of human-AI behavior in the context of Atari games).[3]

We use *ACKTR* [36] for reinforcement learning (as provided as part of *OpenAI Baselines* [8]), together with *OpenAI Gym* [4] and *Arcade Learning Environment (ALE)* [3, 21]. ALE is built around the *Stella* Atari 2600 emulator. We modify OpenAI Gym and ALE to work with two players, and modify

*Max Planck Institute for Software Systems

[2]We also develop a modified, two-player Fall Down, reconfigured so that players maximize the joint score and the game ends on either player's loss of life, as with Space Invaders. For a complete list of Atari 2600 game, we refer the reader to `https://en.wikipedia.org/wiki/List_of_Atari_2600_games`.

[3]All experiments are conducted subject to oversight by an Institutional Review Board (IRB).

Cooperative AI Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020).

OpenAI Gym to allow for deploying frozen policies alongside policies that are training. To ALE, we add functionality to write to the Atari emulator's RAM; e.g., to check robustness, and modulate play in various ways, for example using random start positions. We also introduce the *Javatari Learning Environment*, which makes use of a modified version of Javatari [26] to allow for in-browser Atari play by humans together with an AI player and support the efficient crowdsourcing of trajectories.

To explain our results, we need to introduce a few different concepts. First, to train regular, non-helper AI behaviors, we use ACKTR with a double-headed policy (controlling both players), and for different points along a training curve we extract and freeze single-agent policies. We refer to these as agent policies S_1 through S_4, corresponding to increasing skills (S_1 is novice, and S_4 is expert level, representing training ACKTR until converged). Through reward modifications we also develop agents with diverse behaviors; e.g., agents that prefer to be close to each other, or further apart. Trained agents can be *paired with self*, e.g., $S_2 - S_2$, or paired with another type of agent, e.g., $S_2 - S_3$. We also train Helper-AIs to best-respond to specific, target behaviors, for example $H(S_2)$ is a Helper-AI that is trained to best-respond to S_2. $H(S_2) - S_3$ represents the configuration in which this Helper-AI is deployed along with S_3. Whereas Helper-AIs such as $H(S_2)$ are trained to convergence, we also train helper agents for a smaller number of episodes— a *bounded-helper-AI* such as $bH(S_2)$ is trained to best-respond to S_2 for the same number of episodes as are used to train the S_2 agent (so that $bH(S_2) - S_2$ is comparable in training effort to $S_2 - S_2$).

The experimental results, stated here for two-player Space Invaders, show the following:

1. (Helpful behavior vs. expert behavior) Whereas pairing a non-expert AI with an expert-level AI leads to performance that is worse than when paired with self (e.g., $S_4 - S_2$ is worse than $S_2 - S_2$), Helper-AIs provide a substantial boost in joint score (e.g., $H(S_2) - S_2$ is better than $S_2 - S_2$).
2. (Robust helpful behavior) For all non-expert AIs (S_1 through S_3), pairing this agent with a Helper-AI that is trained for an off-target behavior, say $H(S_2) - S_3$, provides an improvement in performance relative to paired with self.
3. (Robust helpful behavior, bounded helpers) For all non-expert AIs (S_1 through S_3), pairing this agent with a bounded-Helper-AI that is trained for an off-target behavior, say $bH(S_2) - S_3$, provides an improvement in performance relative to paired with self (and similarly for a bounded-Helper-AI trained for on-target behavior).
4. (Robust human transfer) Helper-AIs (including bounded-helper-AIs) improve performance when paired with people, relative to pairing people with medium-skill or high-skill non-helper AIs (e.g., $H(S_2) - Human$ is better than $S_2 - Human$). This performance superiority is maintained even when people follow unexpected behaviors, either through random teleporting to a different location or asking players to "do something unusual".

We also study a special kind of helper agent, referred to as an *Intervention-AI*. In addition to taking actions as one of the two players, an Intervention-AI can also take over for the second, partner agent for some period of time (while incurring a per-action cost), seeking to prevent catastrophic mistakes. We use two-player Fall Down to support a comparative study of Intervention-AIs, showing that even small amounts of intervention are especially beneficial in Fall Down.[4]

These results show that robust, helpful behavior can be achieved in the Space Invaders environment by using RL to learn to best respond to the behavior of another agent, and that this provides far better performance than pairing the partner with an expert agent. The success with Helper-AIs does not come from over-fitting to the behavior of a particular partner— we show robust performance when paired with partners not encountered during training, including artificial agents as well as players controlled by people. Looking forward, and given the richness of the family of Atari 2600 games, we anticipate that this Atari framework will support further study into how to bring humans and AIs together in diverse settings, and we plan to make the two-player Atari framework and Javatari Learning Environment available for use by other researchers.

Related Work. Prior work has modeled a helperful agent as a leader in a Stackelberg leader-follower model, with the second, partner agent assumed to respond to the leader (perhaps with an

[4]We also confirmed result (1) above for Fall Down, but have not confirmed (2) through (4) because of computational budget constraints.

incorrect model of the world) [9]. Our model turns this around, studying Helper-AIs that adapt to a target behavior, and we test these Helper-AIs when matched with other, off-target behaviors. This reframing is appropriate when it is the AI that should adapt to the human actor, rather than the other way around.

Nikolaidis et al. [25] and Crandall et al. [7] study human-AI collaboration in a repeated setting, the latter paper also introducing mechanisms for communication to facilitate cooperative behavior. Carroll et al. [6] demonstrate the gains from deriving faithful models of human behavior in the collaborative two player game, *Overcooked*. We differ in studying the robustness of helper behaviors to misspecifications of partner agents. Although technically quite different, work on shared autonomy (e.g., [18, 30]) relates to the Intervention-AI concept in the present paper.

Other work studies a setting where the AI needs to infer the reward function by observing a human policy, with the human giving demonstrations [15]. Similar steering and teaching settings have been analyzed in environment design (e.g., [37]), machine teaching (e.g., [34]), and advice giving (e.g., [2]). In contrast to these works, we consider an AI that is well-informed about the world, but needs to work effectively with another actor who might not be perfectly rational. There has also been work on cooperative AI that makes use of online learning to adapt to another agent [12, 28, e.g.]. Other work has consider strategic behavior in multi-agent systems and recognized the importance of modelling other actors [10, 13, 22, 27, 29, e.g.], as well as the importance of responding to a diverse set of behaviors in the context of *ad hoc* teaming [5, 33]. There is also broader body of loosely related work on topics such as cooperation in social dilemmas [20, e.g.], learning to communicate [11, e.g.], and social influence [17, e.g.].

2 Preliminaries

Two agents (players), $\mathcal{P}_1$ and $\mathcal{P}_2$, act in a world described by a two-agent Markov Decision Process. The MDP is defined by tuple $\langle \mathcal{S}, \mathcal{A}_1 \times \mathcal{A}_2, \kappa, R, s_0, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}_1$ and $\mathcal{A}_2$ are the action spaces for $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, actions denoted by a_1 and a_2, κ is the transition kernel and provides the probability $\kappa(s'|s,(a_1,a_2))$ of joint action (a_1,a_2) leading to s', conditioned on state s, $R : \mathcal{S} \to \mathbb{R}$ is the reward function, s_0 is the starting state, and γ is the discount factor. We denote by T the end of a game, in our context, a random variable for the time when a life is lost. We focus on stationary Markov policies.

2.1 Differently-skilled AIs

A joint policy, denoted by π, defines the probability $\pi(a_1,a_2|s)$ that joint action (a_1,a_2) is taken in state s. The objective is to maximize the discounted sum of the obtained rewards, with utility $u(\pi) = \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} R(s_t)|s_0,\pi\right]$, where s_t is obtained by executing policy π for t steps, starting from the initial state s_0. We also write $\pi = (\pi_1,\pi_2)$, where the individual policies $\pi_1(a_1|s)$ and $\pi_2(a_2|s)$ are defined in the natural way to reflect masking the action of the other player.

An optimal joint policy is denoted by $\pi^* \in \arg\max_\pi u(\pi)$. We use the term *skill* to refer to the expected reward of a joint policy. Apart from the optimal joint policy, we also consider a set of joint policies $\Pi^\Lambda = \{\pi^\lambda | \lambda \in \{1,...,\Lambda\}\}$, whose elements satisfy $u(\pi^{\lambda_1}) > u(\pi^{\lambda_2}), \forall \lambda_1 > \lambda_2$. Parameter $\lambda > 0$ represents the skill level.

Training. We train joint policies for differing amounts of experience and denote as S_1 through S_4 the (single player) non-helper AI agents that correspond to the actions of player 2 at different snapshots.[5] We choose S_1 through S_4 to correspond to joint policies that are reasonably spaced in score, where S_4 is an expert player and obtained through joint training until performance is no longer improving (200,000 parameter updates, where an update occurs every 80 frames, in each of 32 parallel environments).[6] To obtain agents S_1, S_2, and S_3, we train for 10,000, 30,000, and 50,000 parameter updates respectively.

[5] This is player 2 defined by the emulator, however, this choice is arbitrary and we could choose either player.

[6] All training is done with ACKTR [36], as implemented in the OpenAI baselines [8], and keeping the hyperparameters at their default values (see the Appendix for details). We use 32 CPU cores together with one GPU. When training to control a Helper-AI we modify the ACKTR implementation to allow for a second, frozen policy corresponding to a particular behavior of a partner agent.

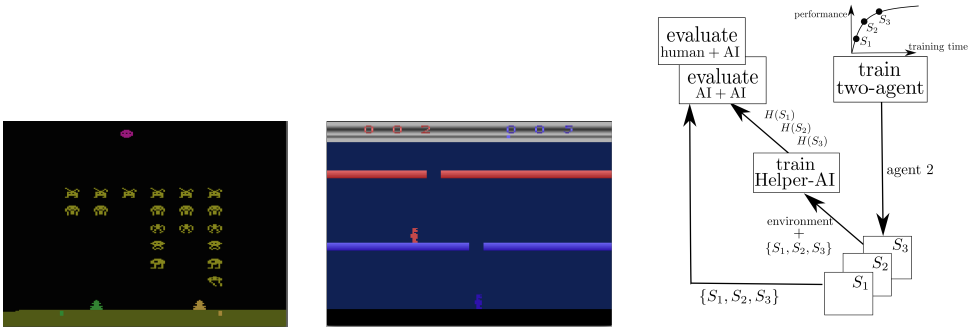

Figure 1: We train a joint policy in the two-player modes of *Space Invaders* and *Fall Down* to obtain a set of differently-skilled, single-actor AIs, i.e., S_1, S_2, and S_3, in increasing levels of skill. We train a Helper-AI agent, $H(S_1)$, $H(S_2)$, and $H(S_3)$, considering each of S_1, S_2 and S_3 as a target behavior, respectively. We test these Helper-AIs in pairings with various off-target AIs as well as with people.

2.2 Behaviorally-Diverse AIs

We also train two, behaviorally-diverse AIs, as a way to provide additional off-target behaviors and test the robustness of Helper-AIs. For this, we use reward shaping, and we construct AIs under a preference to be distant from each other and under a preference to be close to each other, modifying the reward during training (training only; we still report the total score without "distancing rewards"). In particular, we train an S_2-close and S_2-distant agent, for a number of parameter updates that achieve a similar performance in self-play as S_2.[7] We defer the details to the Appendix.

3 Learning Cooperative Behaviors

3.1 Helper-AIs

Helper-AIs are obtained by considering different target behaviors, i.e., fixing the policy of one player and training a policy for the other player. For example, we can consider the target behavior as π_2^λ, for some skill λ. The Helper-AI for this target behavior, denoted by $\pi_1^{*|\lambda}$, is given by

$$\pi_1^{*|\lambda} \in \arg\max_{\pi_1} u(\pi_1, \pi_2^\lambda). \tag{1}$$

From the optimality of $\pi_1^{*|\lambda}$, we have $u(\pi_1^{*|\lambda}, \pi_2^\lambda) \geq u(\pi_1^\lambda, \pi_2^\lambda)$ and $u(\pi_1^{*|\lambda}, \pi_2^\lambda) \geq u(\pi_1^*, \pi_2^\lambda)$, where π_1^* is corresponds to an expert-skill policy. We see in our experiments that this performance advantage can be large, and even when paired with off-target behaviors. Note that $u(\pi_1^\lambda, \pi_2^{\lambda_1})$ need not be an increasing function in λ for $\lambda > \lambda_1$: higher skill need not imply better collaboration. This is another question that we will study in our experiments. In the following, we will denote the Helper-AI for target behavior S_i by $H(S_i)$. Helper-AIs are trained against different target behaviors for 100,000 parameter updates.

3.2 Intervention-AIs

An *Intervention-AI* is a helpful agent that can act in the world and also take actions for the other player (overruling that player). The action space of an Intervention-AI that represents player $\mathcal{P}_1$ is $\mathcal{A}_1 \times (\mathcal{A}_2 \cup \{\texttt{no-op}\})$. The action it suggests for player $\mathcal{P}_2$ is adopted when it is not `no-op`. Intervention-AIs can be defined for different target behaviors. They are also parameterized by a cost c_{int}, representing the cost incurred for overriding the other player; e.g., resulting from the additional burden or effort caused by the intervention. As an example, the Intervention-AI $\pi^{*|\lambda,\mathcal{I}}$ for a target-behavior with skill λ is composed of the policy π_1 for actions of player $\mathcal{P}_1$ and policy π_2' for

[7] For Space Invaders, these AIs achieve average scores in self-play of 1,111, 1,141 and 1,134, for S_2-close and S_2-distant, and S_2 respectively.

actions of player $\mathcal{P}_2$, and

$$\pi^{*|\lambda,\mathcal{I}} \in \arg\max_{\pi_1,\pi_2'} \big[u\left(\pi_1, \mathcal{I}(\pi_2', \pi_2^\lambda)\right) - C\left(\pi_1, \mathcal{I}(\pi_2', \pi_2^\lambda); c_{\text{int}}\right) \big], \quad (2)$$

where $\mathcal{I}$ is an *intervention operator* that defines the induced, joint policy, and $C(\cdot)$ is the expected value of the discounted sum of the intervention costs. The intervention operator is defined to provide the action of π_2^λ when π_2' chooses `no-op` and the action provided by π_2' otherwise. Let $u(\pi^{*|\lambda,\mathcal{I}}, \pi_2^\lambda) = u(\pi_1^{*|\lambda,\mathcal{I}}, \mathcal{I}(\pi_2^{*|\lambda,\mathcal{I}}, \pi_2^\lambda))$, and similarly $C(\pi^{*|\lambda,\mathcal{I}}, \pi_2^\lambda; c_{\text{int}}) = C\left(\pi_1, \mathcal{I}(\pi_2', \pi_2^\lambda); c_{\text{int}}\right)$. We have

$$u(\pi^{*|\lambda,\mathcal{I}}, \pi_2^\lambda) \geq u(\pi^{*|\lambda,\mathcal{I}}, \pi_2^\lambda) - C(\pi^{*|\lambda,\mathcal{I}}, \pi_2^\lambda; c_{\text{int}})$$
$$u(\pi^{*|\lambda,\mathcal{I}}, \pi_2^\lambda) - C(\pi^{*|\lambda,\mathcal{I}}, \pi_2^\lambda; c_{\text{int}}) \geq u(\pi_1^{*|\lambda}, \pi_2^\lambda),$$

where the first inequality uses $C(\cdot) \geq 0$ and the second inequality follows because Helper-AI $\pi_1^{*|\lambda}$ is a feasible Intervention-AI (setting $a_2 =$`no-op` always) with zero cost. As a result, an Intervention-AI will dominate a Helper-AI in terms of on-target performance.

Training. For the Intervention-AI, after clipping rewards to $[0, 1]$, we consider per-action costs in the range $[0.01, 0.05]$ for Space Invaders and in the range $[0.005, 0.05]$ for Fall Down. This range of costs provides a range in the rate of per-action intervention from 1% to 60% for Space Invaders, and 1% to 63% for Fall Down.[8] We report the total score, without intervention costs, so that the scores with intervention-AIs are directly comparable to settings without an Intervention-AI. The Intervention-AIs are trained for 100,000 parameter updates.

3.3 Bounded Helper-AIs

We also train Bounded-Helper-AIs, which are Helper-AIs that are trained against a target behavior for a limited number of parameter updates. In particular, we fix the number of updates to match up against a baseline of interest. For target behavior S_2, for example, we consider the Bounded-Helper-AI, denoted $bH(S_2)$, that is trained for the same number of parameter updates as S_2, i.e., 30,000 parameter updates, and analogously for the other Bounded-Helper-AIs.

3.4 Helper-AIs trained with Random Position Initialization

For the human experiments we also consider Helper-AIs that are trained for randomly initialized starting positions of the players. The hypothesis is that this provides additional robustness through exposure to new states; e.g., the players having switched sides, or the players near the middle of the screen with no aliens killed. We first train agents, analogously to S_1, S_2, and S_3 with randomized starting positions every new game and wave of aliens and including a reward penalty for crossing from the randomly generated side. We denote these agents as R_1, R_2, ..., R_3. We then train a Helper-AI with randomized starting positions and the same reward penalty; e.g., $RH(R_2)$ is a randomized Helper-AI trained with R_2 as the target.

4 Experiments: Helper-AIs with AIs

In two-player Space Invaders, the players score points by hitting invading aliens or the command ship. The aliens also shoot, and the game ends when either player is hit or the aliens reach the bottom of the screen (landing on Earth). We modify the ROM to make it cooperative: (1) we remove the bonus for the other agent being killed, and (2) we make the points scored for hitting the enemy command ship the same in 1- and 2-player modes. We also make use of being able to read and write to RAM.[9] We provide the experimental results in Table 1. We observe the following:

[8] At 60 frames per second, 1% corresponds to 1 intervention/minute.

[9] We read from RAM to calculate joint reward and gain state information used to modify rewards in order to train the diverse behaviors (close and distance preferences). We write to RAM to randomize the start position of players during training.

	The Behavior of the Matched AI				
	S_1	S_2	S_2-close	S_2-distant	S_3
Performance with self	878	1,134	1,111	1,141	2,141
... with expert skill agent	694	963	457	711	1,826
with Helper-AI trained for					
S_1	$\boxed{1,701}$	2,294	1,185	1,449	3,538
S_2	1,587	$\boxed{2,434}$	1,227	1,548	3,792
S_2-close	1,254	1,836	$\boxed{1,932}$	1,405	2,733
S_2-distant	1,414	2,197	1,210	$\boxed{2,375}$	3,838
S_3	1,282	2,204	1,220	1,670	$\boxed{3,844}$
S_2 (bounded helper)	1,337	2,148	1,193	1,550	3,009

Table 1: Two Player, Cooperative Space Invaders. Game score, averaged over 100 games, of pairing an agent (columns) with different agents (rows): whether another copy of itself, a higher-skilled agent, or Helper-AIs, both on-target and off-target. While the strongest performance is by Helper-AIs paired with the agents that they have been trained with, the benefit to using a Helper-AI is positive for all of the pairings. The Bounded-Helper-AI also provides a uniform advantage related to self-play and matching with the expert skill agent.

	agent 2			
	S_1	S_2	S_3	S_4
with self	46.0	77.4	120.2	248.1
with S_4	32.7	44.3	79.1	–
with Helper-AI	63.8	93.7	151.9	–

Table 2: Two-player, Cooperative Fall Down. The game score for different skill levels of agent two from low (S_1) to high (S_4), averaged over 100 games. Agent two is paired with itself, with the high skill agent S_4, and with Helper-AI. The collaboration of an AI with its on-target Helper-AI achieves the highest score, while pairing an AI with S_4 can degrade the joint performance.

(1) There is a substantial and uniform improvement in terms of score from pairing an AI with its on-target Helper-AI compared to pairing the AI agent with another copy of itself. When expressed as the percentage of score increase, the improvement has the average of 94% across the AIs tested, at it ranges from 74% for S_2-close to 115% for S_2.

(2) There is always some benefit for pairing an AI with an off-target Helper-AI, with some behaviorally-diverse AIs appearing to require more specialized on-target training (the S_2-close agent only receives a substantial benefit from an on-target Helper-AI, although off-target Helper-AIs are still able to achieve comparable performance to a pairing with another S_2-close agent without experiencing the modified rewards used to train S_2-close).

(3) The Helper-AI with limited training, $bH(S_2)$ is still able to improve performance for partner agents, although its effect is less than a fully-trained Helper-AI. This suggests that effective helpful behavior can be learned quickly and also still transfer to environments with off-target AIs.

We have also tested the performance of Helper-AIs in Fall Down. The results closely follow what we see for Space Invaders (Table 2). Pairing an AI with its on-target Helper-AI leads to a significant increase in the joint performance, with the improvement ranging from 21% to 39%. Pairing an AI with S_4 leads to degraded performance, except when the AI is of the same type (S_4). The goal in Fall Down is for players to fall through gaps in the platforms, while the screen scrolls up. A player loses their life if they fail to fall through a gap when the platforms reach the top of the screen.[10]

[10]We make a couple of modifications to make Fall Down more suitable for our study. First, we terminate the game when the first life is lost, providing cooperative incentives. Second, we modify the RAM to fix the scrolling speed to stop it increasing during the game, because this allows for a greater range of skill by agents

4.1 Understanding Helper-AI Behavior

We also collect data to understand how the Helper-AIs are able to help less-skilled agents. One piece of information is to consider the reasons for episode termination. For this, we focus on agents S_2, S_4 and the Helper-AI agent trained with S_2, i.e., $H(S_2)$. There are four reasons that an episode may end (Player 1 hit, Player 2 hit, Both players hit, Aliens Land). We extract these from emulator RAM. To survive, a player must both avoid getting hit and prevent the invaders from landing by hitting them. In the following we summarize what we find. A more detailed version of the results can be found in the Appendix. When S_2 is paired with itself, the observed probability of either player getting hit is approximately equal, and the probability of the aliens landing is 18%. However, when player 1 is replaced with S_4, while that player is individually less likely to get hit (presumably due to its higher skill), the miscoordination of letting the aliens land goes up to 25%. On the other hand, when player 1 is replaced with $H(S2)$, not only is it much less likely to get hit (presumably by being aware of which invaders its partner agent, S_2, is able to reliably hit), but overall miscoordination goes down to 15%.

4.2 Intervention AIs

Intervention-AIs are able to improve the score relative to on-target Helper-AIs even with about 1% interventions, suggesting that they are able to learn to intervene effectively. We are especially interested to understand the different effect of intervention in Fall Down vs. Space Invaders. In Space Invaders, an Intervention-AI that intervenes 1% of the time increases the average score to 2,534 for pairing with S_2 (a 4% benefit relative to a Helper-AI). In Fall Down, the Intervention-AI increases the average score for pairing with S_2 to 104.1 (an 11% benefit relative to a Helper-AI). The interventions are qualitatively different. For Space Invaders and a small intervention cost (and 1% interventions), there is an average of 17% interventions directly after a previous intervention (meaning that the agent chooses to intervene for multiple actions in a row). In Fall Down, the analogous number is 58% while the total number of interventions, normalized by game length, in the two games is essentially the same. Intervention-AIs in Fall Down learn that a longer sequence of interventions is necessary to improve performance. This can be understood from the differing dynamics in the two environments: survival in Space Invaders is as simple as dodging a missile, but in Fall Down requires navigating to the platform below while avoiding interference from the other player. We provide a more detailed set of results on Intervention AIs in the Appendix.

5 Helper-AI Transfer to Human Partners

In this section, we describe the results from deploying different Helper-AIs together with humans. For this, we use a new crowdsourcing environment, which we refer to as the *Javatari Learning Environment (JLE)*. JLE supports running Atari games in a web browser, including with the various affordances we consider and with the game populated by an AI agent (e.g., S_2 or $H(S_2)$). JLE is described in the Appendix.

We run a number of experiments, each with 10 distinct human subjects recruited from Amazon Mechanical Turk, paired with different AI agents. We limit participation to those based in the U.S., who had previously completed 10,000 tasks with a 98% approval rate, i.e., relatively experienced workers. The humans are unaware which AI they were playing together with, and whether it was supposed to be helpful or not. In the Appendix, we provide additional details on instructions given to participants and our quality control mechanisms.

The human participants were paired with (1) the high-skill AI S_4, (2) the medium-skill AI S_2, (3) the Helper-AI trained with S_2, $H(S_2)$, and (4) the limited training Helper-AI trained with S_2, i.e., $bH(S_2)$. We also wanted to study whether designing Helper-AIs that are exposed to more diverse starting states will make them more robust to unexpected changes in the environment. To test this, we also consider the performance of $H(S_2)$ and $RH(R_2)$ in the human experiments ($RH(R_2)$ and R_2 are agents with random position initialization, see Section 3.4). We also promote unexpected behavior by the human subjects. We introduce two different kinds of shocks, each repeated twice

(the unmodified game increases the scrolling speed to arbitrarily high values, which effectively truncates the episode length).

AI Agent	Paired with S_2	Paired with Humans
S_2	1,134	704
S_4	963	545
$H(S_2)$	2,434	1,547
$bH(S_2)$	2,148	1,083
$H(S_2)$	-	950 (shock environment)
$RH(R_2)$	-	1,260 (shock environment)

Table 3: Comparing the performance of different AI agents, both standard AIs such as S_2 and S_4 as well as Helper-AIs, including Bounded-Helper-AIs, when paired with either S_2 or with human subjects. In the bottom half of the table we report the results for comparing the performance of Helper-AI $H(S_2)$ and a Helper-AI trained with randomization $RH(R_2)$ in human experiments where the players were sometimes randomly teleported to different positions and sometimes asked to do something unexpected for a period of time.

during the experiment: (1) randomly teleporting each player to a different location,[11] and (2) asking the human to "do something unusual" for 20 seconds.[12] This second treatment is designed as a proxy for humans acting at least somewhat adversarially to their partner, and we observed behaviors such as the human participant moving rapidly back and forth, stopping shooting, and trying to get very close to the AI.

Helper-AIs are able to improve performance when paired with humans, relative to using either the medium-skill agent S_2 or the expert skill agent S_4 (Table 3). In addition, we observe the same pattern with human subjects as with AIs, in that pairing with the expert-level AI actually hurts performance. Also, the Helper-AI trained with randomization, $RH(R_2)$, was able to perform better in the experiments in which shocks are introduced into the environment, suggesting a useful direction in training Helper-AIs that are robust to unexpected shifts in the environment.

In the Appendix, we provide additional results that qualitatively compare the strategies of different agents, and indicate that agents of different types (humans, Non-Helper-AIs, and Helper-AIs) have qualitatively different behavior.

6 Conclusion

We have presented the first study of cooperation in the setting of two-player Atari games (suitably modified to provide cooperative incentives). Through a simple application of reinforcement learning, we have demonstrated success in the design of Helper-AIs with robust performance in cooperative AI settings, both with off-target behaviors of other AIs and with people.

The JLE framework is quite flexible, with its ability to support AI-AI as well as human-AI collaboration, competition, and even mixed cooperative-competitive dynamics (by choosing the appropriate reward functions). Atari games are extremely well documented, and the ability to write to emulator RAM also allows essentially arbitrary changes to the game environments. In future work, we plan to extend our results to different Atari games, and study variations with Helper-AIs that have only partial information regarding the joint goal (e.g., in the context of Atari games, the joint score). It will be interesting to understand which types of environments support robust helpful behavior through simple RL behaviors, as studied here, and where more sophisticated Helper-AI designs will be needed. Also of interest is to employ *imitation learning* (building from [16, 32, 35]), applied to crowdsourced human trajectories, in order to model human behavior and train Helper-AIs with human models as targets. We envision a loop between crowdsourcing (and imitation learning) to model human behavior, and reinforcement learning for the design of Helper-AI policies.

[11] The teleportation treatment appeared after three minutes and seven minutes of game play (with the following warning to the human player: *You and your partner are about to be teleported!*).

[12] The unexpected action treatment appeared after five minutes and nine minutes of game play, with the message: *Now do something unexpected!*. After 20 more seconds, the message *Back to normal! Keep playing as you were before.* would appear.

Acknowledgements

The project or effort depicted was or is sponsored, in part, by the Defense Advanced Research Projects Agency under cooperative agreement number HR00111920029, the content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. Approved for public release; distribution is unlimited. This work was also supported, in part, by a SNSF Early Postdoc Mobility fellowship.

References

[1] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus. Learning Robust Control Policies for End-to-End Autonomous Driving From Data-Driven Simulation. *IEEE Robotics and Automation Letters*, 5(2):1143–1150, 2020.

[2] Ofra Amir, Ece Kamar, Andrey Kolobov, and Barbara Grosz. Interactive Teaching Strategies for Agent Training. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, page 804–811, 2016.

[3] M. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *JAIR*, 47:253–279, 2013.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv:1606.01540*, 2016.

[5] Rodrigo Canaan, Julian Togelius, Andy Nealen, and Stefan Menzel. Diverse Agents for Ad-Hoc Cooperation in Hanabi. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

[6] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the Utility of Learning about Humans for Human-AI Coordination. In *Advances in Neural Information Processing Systems*, pages 5175–5186, 2019.

[7] Jacob W Crandall, Mayada Oudah, Fatimah Ishowo-Oloko, Sherief Abdallah, Jean-François Bonnefon, Manuel Cebrian, Azim Shariff, Michael A Goodrich, Iyad Rahwan, et al. Cooperating with Machines. *Nature Communications*, 9(1):1–12, 2018.

[8] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov. OpenAI Baselines, 2017.

[9] C. Dimitrakakis, D. Parkes, G. Radanovic, and P. Tylkin. Multi-View Decision Processes: The Helper-AI Problem. *NIPS*, 30:5449–5458, 2017.

[10] J. Foerster, R. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch. Learning with Opponent-Learning Awareness. *AAMAS*, 17:122–130, 2018.

[11] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to Communicate with Deep Multi-agent Reinforcement Learning. In *Advances in neural information processing systems*, pages 2137–2145, 2016.

[12] Ahana Ghosh, Sebastian Tschiatschek, Hamed Mahdavi, and Adish Singla. Towards Deployment of Robust Cooperative AI Agents: An Algorithmic Framework for Learning Adaptive Policies. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 447–455, 2020.

[13] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *International Conference on Machine Learning*, pages 1802–1811, 2018.

[14] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot Learning in Homes: Improving Generalization and Reducing Dataset Bias. pages 9094–9104, 2018.

[15] D. Hadfield-Menell, S. Russell, P. Abbeel, and A. Dragan. Cooperative Inverse Reinforcement Learning. *NIPS*, 29:3909–3917, 2016.

[16] J. Ho and S. Ermon. Generative Adversarial Imitation Learning. *NIPS*, 29:4565–4573, 2016.

[17] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z Leibo, and Nando De Freitas. Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning. In *International Conference on Machine Learning*, pages 3040–3049, 2019.

[18] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared autonomy via hindsight optimization. *Robotics science and systems: online proceedings*, 2015, 2015.

[19] Vitaly Kurin, Sebastian Nowozin, Katja Hofmann, Lucas Beyer, and Bastian Leibe. The atari grand challenge dataset. *arXiv preprint arXiv:1705.10998*, 2017.

[20] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent Reinforcement Learning in Sequential Social Dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473, 2017.

[21] M. Machado, M. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *JAIR*, 61:523–562, 2018.

[22] Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai and human behavior: Chess as a model system. *arXiv preprint arXiv:2006.01855*, 2020.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602*, 2013.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[25] S. Nikolaidis, S. Nath, A. Procaccia, and S. Srinivasa. Game-Theoretic Modeling of Human Adaptation in Human-Robot Collaboration. *HRI*, 12:323–331, 2017.

[26] P. Peccin. Javatari - Online Atari 2600 Emulator. `https://github.com/ppeccin/javatari.js`, 2015.

[27] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine Theory of Mind. In *International Conference on Machine Learning*, pages 4218–4227, 2018.

[28] Goran Radanovic, Rati Devidze, David Parkes, and Adish Singla. Learning to Collaborate in Markov Decision Processes. In *International Conference on Machine Learning*, pages 5261–5270, 2019.

[29] R. Raileanu, E. Denton, A. Szlam, and R. Fergus. Modeling Others using Oneself in Multi-Agent Reinforcement Learning. 35:4254–4263, 2018.

[30] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Shared autonomy via deep reinforcement learning. *arXiv preprint arXiv:1802.01744*, 2018.

[31] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fdelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics*, 2017.

[32] J. Song, H. Ren, D. Sadigh, and S. Ermon. Multi-Agent Generative Adversarial Imitation Learning. *NeurIPS*, 31:7472–7483, 2018.

[33] Peter Stone, Gal A Kaminka, Sarit Kraus, and Jeffrey S Rosenschein. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[34] Sebastian Tschiatschek, Ahana Ghosh, Luis Haug, Rati Devidze, and Adish Singla. Learner-aware Teaching: Inverse Reinforcement Learning with Preferences and Constraints. In *Advances in Neural Information Processing Systems*, pages 4147–4157, 2019.

[35] Z. Wang, J. Merel, S. Reed, N. de Freitas, G. Wayne, and N. Heess. Robust Imitation of Diverse Behaviors. *NIPS*, 30:5326–5335, 2017.

[36] Y. Wu, E. Mansimov, R. Grosse, S. Liao, and J. Ba. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. *NIPS*, 30:5279–5288, 2017.

[37] H. Zhang and D. Parkes. Value-based Policy Teaching with Active Indirect Elicitation. *AAAI*, 23:208–214, 2008.

Appendix A: Atari 2600 Platform Architecture and Reading/Writing RAM with the Modified Arcade Learning Environment

The Atari 2600, which uses a MOS Technology 6507 microprocessor[13], has 128 bytes of RAM. The RAM completely encodes a game state. In addition, there is a ROM which dictates the game dynamics, and is a binary file compiled from MOS 6502 assembly. The original ALE framework [3, 21] allows researchers to read from the RAM of the Atari emulator, corresponding to various in-game elements, including score, positions of the players, and enemies. We use this throughout our experiments, including the ability to determine the position of the agents and thus create *close* and *distant* variants, as described in Section 2.2.

In the framework that we have built, we also added the capability for writing to the RAM, which allows us to modify any in-game variables, including the game dynamics. There are many reasons for finding this capability as interesting and useful: (1) it allows us to replay game trajectories and compare the behavior of different agents given a fixed starting game state; (2) it allows us to modify the world in ways that may differ from the original game logic (e.g., teleporting agents between different parts of the screen, making the shields in Space Invaders indestructible or removing them, etc.); (3) it allows us to directly modify the in-game rewards in accordance with any of these aspects, which makes them indistinguishable from the original game logic from a player's perspective, and is especially relevant when pairing AIs with human players.

There are two common types of ROMs for the Atari 2600: 4 kilobyte ROMs, and 8 kilobyte ROMs that use bank switching since the Atari's processor can not address this amount of external memory. The 4KB games are much easier to disassemble and reassemble, since the memory addresses do not change during program execution. Space Invaders is a 4 KB game, while Fall Down is an 8 KB game; this is largely why we chose to make all changes to Fall Down only through the emulator. In Fall Down, as in many bank-switched games, one of the banks is used only for a game mode selection screen (as seen in Figure 2).

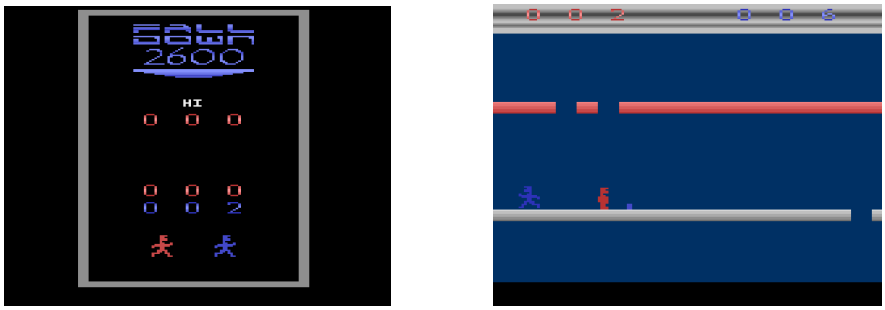

Figure 2: Screenshots from Banks 1 and 2 in *Fall Down*. The memory addresses in Tables 4 and 5 all refer to Bank 2.

As described in Tables 4 and 5, there are some key variables within the games. Notably, these can be completely different game-to-game. The memory addressing is of the form `00xy`, where $x \in \{8, 9, A, B, C, D, E, F\}$ and $y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$. Each address can hold one byte. Thus, we obtain the 128 bytes of addressable RAM, since $|x| \times |y| = 8 \cdot 16 = 128$.

As is common in bank-switched games, there is a special indicator byte in Fall Down for whether we are in Bank 1 (the game mode selection screen) or in Bank 2 (the game). Specifically, if memory location F9 contains the hexademical value 18, we are in Bank 1, and if it contains hexadecimal value AF, then we are in Bank 2. Once a game has started, the value in this memory location is also a proxy for being in the terminal state in bank-switched games, given that the ROM has a special instruction to switch banks when the game is over, going from Bank 2 to Bank 1.

Most Atari games, including Space Invaders and Fall Down, have Game Modes (as mentioned in Table 4), controlled by particular locations in the RAM. The different game modes available in Space Invaders are illustrated in Figure 3. There are 112 game modes in total, which have various features,

[13] A slightly more powerful cousin of the MOS 6507, the MOS 6502, was used in many of the most famous personal computer systems, including the Commodore 64, the Apple I and][, and others.

Variable	Space Invaders	Fall Down
Game Mode	DC	F6
Lives	C9	-
Player 1 x-position	9C	D5
Player 2 x-position	9D	D6
Player 1 y-position	-	D7
Player 2 y-position	-	D8

Table 4: RAM locations of environment variables for *Space Invaders* and *Fall Down*.

such as moving shields, attacker shots that don't move in a straight line, attacker shots that move quickly, and invisible attackers; the game modes can be grouped into the following (with all possible combinations of game features possible in each mode):

1. [game modes 1 - 16] single player
2. [game modes 17 - 32] opposing players, where each player interacts in a separate world (fighting their own set of invaders), alternating turns/worlds
3. [game modes 33 - 48] opposing players playing at the same time; a player gets 200 points if the other player dies,
4. [game modes 49 - 64] same as #3, but ability to shoot alternates between the players
5. [game modes 65 - 80] partnership game, where one player can only move the cannon left and the other player can only move the cannon right; who can fire the cannon alternates
6. [game modes 81 - 96] partnership game, where the players alternate control of the cannon until a shot is fired, and then control switches
7. [game modes 97 - 112] partnership game, where one player can move the cannon and the other can fire it

Without modification, none of these game modes are quite appropriate for a cooperative game setting and designing Helper-AIs. #2 does not feature players even interacting with the same environment, #4 has temporal dependence between the action set available to the player, in that part of the action set is disabled for one player until the other player takes an action, #6 is similar except that a player can't act at all until the other player has acted (and the resulting state is entirely the effect of the other player's actions), and #7 does not have the same action set for the two players. The closest to the desired setting for a cooperative game is #3, but this mode has the problem that a player gets 200 points if the other player is killed.

Thus, we use `DiStella`[14] to disassemble the Space Invaders ROM, and modify the appropriate lines of assembly to remove this point bonus.[15] We also modify the ROM to make the points awarded for hitting the command ship consistent across the one-player and two-player games, so that performance can be readily compared. We reassemble the modified assembly with `dasm`[16]. All of our experiments with two-player Space Invaders use game mode 33, along with these two modifications to the ROM.

Fall Down has nine game modes, selected from the mode selection screen in Bank 1 or by writing directly to RAM. Some modes have the human play against an Atari provided "AI". This is a heuristic agent provided as part of the game ROM, and that gets information about the location of the next platform gap and thus the direction in which to go. The different game modes are:

1. Human vs. AI
2. Human vs. AI, easy mode (the AI is not as effective, in that it does not move in the air, until it has landed on the next platform)
3. Human vs. AI, advanced mode (the scroll speed is initialized at a higher value)

[14]Presently offline, but an archived version is available at `https://web.archive.org/web/20180618000759/https://sourceforge.net/projects/distella`.

[15]Note that changes like this could instead be accomplished via writing to the RAM and immediately subtracting the point at the appropriate time.

[16]Available at `https://dasm-assembler.github.io`

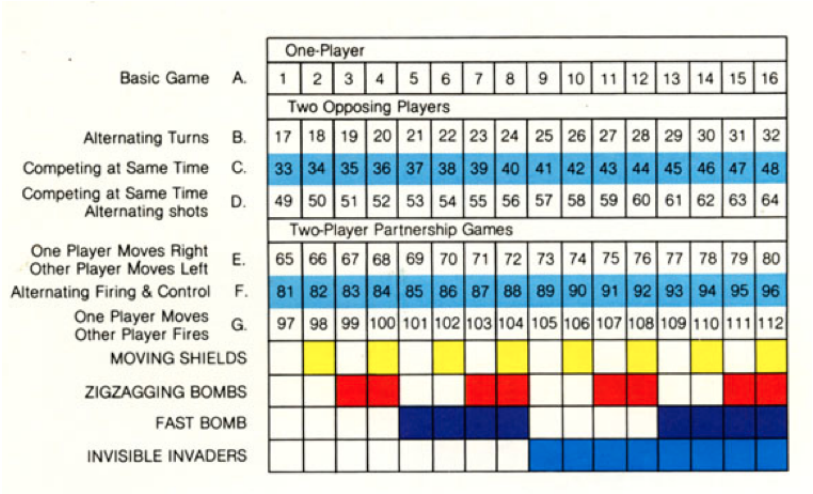

Figure 3: An illustration of the different game modes available in Space Invaders, excerpted from the Atari 2600 Space Invaders manual.

4. Human vs. AI, pass-by mode (the players can pass by each each, instead of interfering with each other's movements and "bouncing off each other")
5. Human vs. AI, invisibility mode (the background color is alternated so that one of the players is temporarily invisible during game play)
6. Human vs. Human
7. Human vs. Human, advanced mode (the scroll speed is initialized at a higher value)
8. Human vs. Human, pass-by mode (the players can pass by each each, instead of interfering with each other's movements and "bouncing off each other")
9. Human vs. Human, invisibility mode (the background color is alternated so that one of the players is temporarily invisible during game play)

For all of our experiments with Fall Down, we use game mode #6, with either our RL-based AIs or people controlling the players.

Difficulty modes are not part of the RAM but controlled via an emulated hardware switch, with a difficulty switch for each player. This is controlled in ALE via the `setDifficulty(difficulty)` function.[17] The difficulty modes can be set separately for each player. In Space Invaders, the higher difficulty corresponds to wider players, meaning that it is easier to get hit, while the capacity to fire remains the same. Here, we adopt the "easy mode" for both players.

In Fall Down, the higher difficulty level disables screen wrap-around, where walking off the screen to the left brings the player back to the right and vice-versa. We adopt the "hard mode" for both players. The effect is that the players can more easily interfere with each other's actions by physically blocking the other agent, or can choose to jump to make room for the other agent, and thus there is a greater coordination aspect and more of an opportunity to study the use of Helper-AIs. Playing in different difficulty modes can also be used as a limited in-game resource.

Appendix B: Training Agents in Multi-player Atari

Our technical stack for training agents in multi-player Atari is illustrated on the left side of Figure 7. It consists of modifications to ALE to allow access to controlling the second player (in the original version of ALE, only single-player modes of the games are supported). In addition, we provide hooks to the score representation for the second player, which allows us to specify the reward to the agents

[17] From the ALE code: "If the first bit is 1, then it will put the left difficulty switch to A (otherwise leave it on B). If the second bit is 1, then it will put the right difficulty switch to A (otherwise leave it on B)."

Variable	Space Invaders	Fall Down
Player 1 Score (hundreds)	E6	F3
Player 2 Score (hundreds)	E7	F4
Player 1 Score (ones)	E8	F1
Player 2 Score (ones)	E9	F2
Scroll Speed	-	8D
# Aliens/Animation Speed	91	-
Wave Mode	AA	-
Shields	AB, AC, ..., C5	-

Table 5: RAM locations of other important environment variables for *Space Invaders* and *Fall Down*. (Unclipped) scores are determined by adding 100 times the hundreds score to the ones score for each player. For example, in Space Invaders, Player 1's score is $100 \cdot val(E6) + val(E8)$, and player 2's score is $100 \cdot val(E7) + val(E9)$. When Space Invaders is not yet in active game mode (i.e., the demo or attract mode, which precedes the start of the game and is used for selecting the game modes), the memory locations for score will hold different values, corresponding to the game mode and number of players, so these registers are only in use during actual game play. The Wave Mode variable refers to how far from the bottom of the screen new waves of aliens start (decreasing over time, up to a certain limit). Shields are represented by 27 RAM locations (meaning that 21% of in-game RAM was used just for displaying the shields' state), with each location corresponding to part of each of the three shields. Shields are automatically hidden in later wave modes.

Figure 4: An illustration of the points awarded for hitting each space invader, excerpted from the Atari 2600 Space Invaders manual.

as being the joint reward (i.e., the sum of the rewards for Players 1 and 2) in the fully cooperative setting. We also introduce support for Fall Down, which was not previously included in ALE. It is important to note that while making changes to a ROM (as described in the previous section for Space Invaders) can also be accomplished by appropriate modifications via the emulator, the reverse is not true; there are changes to the game dynamics, such as introducing intervention AIs or rewarding certain behavioral patterns, that can be made only via the emulator.

Reward clipping is useful when using reinforcement learning in games such as Space Invaders, to stabilize the learning procss and to allow sharing hyperparameters, such as the learning rate, to be shared across games [23, 24]. There is no reward clipping necessary in Fall Down, because the rewards are already in $\{0, 1\}$. A result of reward clipping is that the AI is unaware of the actual points scored for hitting enemies, as depicted in Figure 4, both during training and testing. Humans playing the game, however, observe the actual points given. Exploring the potential differences between human and AI game play due to reward clipping is left to future work.

Diverse Agent Behaviors

As depicted in Figures 5 and 6, the two-player training procedure that we use can also train behaviorally-diverse AIs when suitable auxiliary rewards are introduced into the training process. The amount of the reward must be large enough to encourage the desired behavior, but also not so large that it disrupts the process of learning effective performance in the environment.

For Space Invaders, we provide an additional reward of 0.1 (with normal in-game rewards clipped, as usual) when the inter-player distance is in the range $[5, 25]$ (for mode *close*) and at least 45 (for mode *distant*), where the distance can range from 0 to 82, as determined by RAM. For Fall Down,

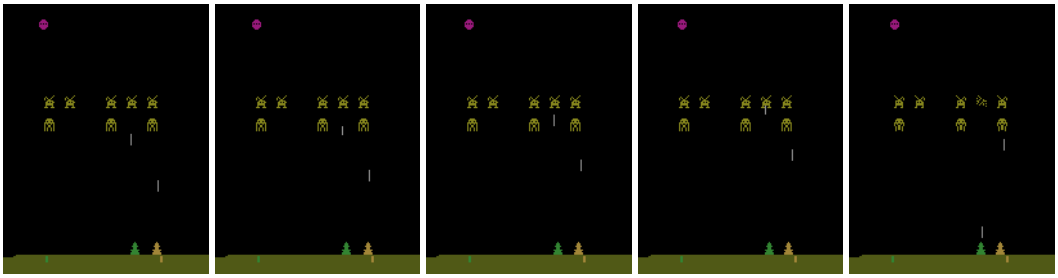

Figure 5: Frames from *close* agents playing 2-Player Cooperative Space Invaders. The players (green and yellow) remain close to each other throughout the game.

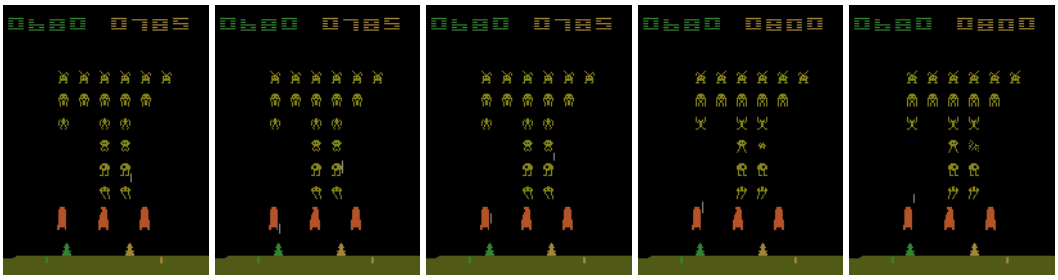

Figure 6: Frames from *distant* agents playing 2-Player Cooperative Space Invaders. The players (green and yellow) remain quite far from each other throughout the game.

we provide an additional reward of 0.025 for an inter-player distance at most 70 (for mode *close*) and at least 150 (for mode *distant*)[18].

Appendix C: Javatari Learning Environment

The Javatari Learning Environment (JLE) consists of a server running our modified multi-player Atari framework based around ALE and a lightweight front-end using Javatari. JLE allows for trained AI agents to play alongside humans in a web browser, built around a modified version of the Javatari emulator, communicating continuously with a server running a technical stack based around our modified Atari framework. The key idea here is to send only the emulator RAM to the web browser, and return the actions taken by the human. This allows for complete compatibility between ALE and Javatari, because the transitions between states are generated by ALE's Stella emulator.[19] The design of our Javatari Learning Environment is illustrated in Figure 7.

Appendix D: Intervention-AI Experiments

Here, we expand on the discussion in Section *Intervention AIs*. Tables 6 and 7 show the results of using Intervention-AIs that intervene infrequently across different skill levels. Notably, they are able to improve the score relative to on-target Helper-AIs even with about 1% interventions, suggesting that they are able to learn to intervene effectively.

[18]The positions of the agents are encoded in the emulator RAM state. This set-up is able to generate agents with qualitatively different behavior. In Space Invaders, the *close* agents have on average 1/8 of the screen distance between them while the *distant* agents have on average 3/4 of the screen distance between them, compared with a typical distance of 1/2 without these modifications. In Fall Down, we use the Euclidean distance in (x, y) coordinates. In Space Invaders, we use [5,25] as the distance threshold for training close agents to prevent the players from favoring occupying exactly the same position, which is a pathological behavior that results from rewarding them for any distance ≤ 25.

[19]The emulated game dynamics are different between Stella and Javatari at the trajectory level. Similar incompatibility at the frame level was observed by [19].

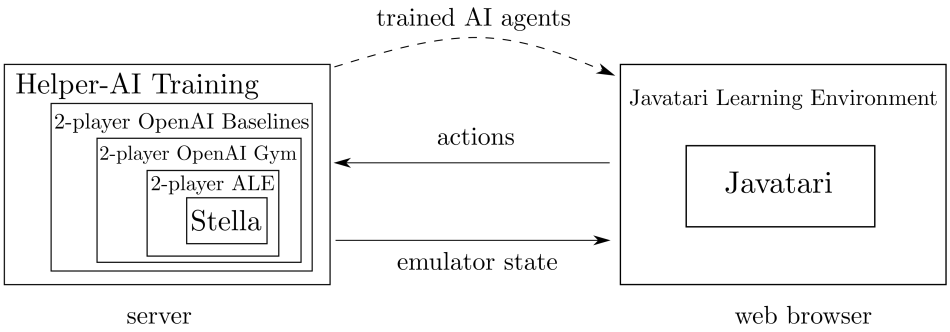

Figure 7: Technical framework: We use 2-player modifications of ALE, OpenAI Gym, and the OpenAI Baselines, modify the Atari emulator *Stella* to allow us to write to the RAM, including during game play. The Javatari Learning Environment allows for trained AI agents to play alongside humans in a web browser, built around a modified version of the Javatari emulator, and communicating continuously with a server running a technical stack based around our modified Atari framework. This allows for complete compatibility between ALE and Javatari because the transitions between states are generated by ALE's Stella emulator.

agent 2	cost	% intervene	score
S_1	0.05	2%	1,772
S_1	0.025	20%	1,927
S_1	0.01	58%	3,680
S_2	0.05	1%	2,534
S_2	0.025	19%	3,234
S_2	0.01	59%	4,787
S_3	0.05	1%	3,985
S_3	0.025	19%	4,367
S_3	0.01	60%	5,029

Table 6: Two-player, Cooperative Space Invaders. Game score with Intervention-AI, varying the cost for intervention, averaged over 100 games, and with each of S_1, S_2 and S_3 as agent two. As the cost for intervention decreases, the Intervention-AI takes more actions, and the average score improves.

Another measure of the effectiveness of an Intervention-AI is how frequently the action that it takes is different from the action that its partner *would have taken* had it not intervened. The results from studying this effect are in Tables 8 and 9. The fraction of effective interventions increases with lower-skilled partners, and with fewer total interventions. Of course, the counterfactual action is never exposed to the Intervention-AI, although a more sophisticated form of Intervention-AI training might include this in the state, to prevent it from intervening when its partner is likely to take the given action anyway.

Different kinds of interventions are left to future work, including changes in the fabric of the games themselves, such as adjusting a game-specific variable like scroll speed, changing the difficulty switch for the other player, changing positions with the other player, and so on. It is also left to future work to consider trajectory-level counterfactuals, i.e., to what extent the interventions are able to shift the distribution of states, and whether they effectively alter the occupancy measures of low-quality states. The Intervention-AI framework also has the benefit of being able to make any single-player game into a 2-player game, where the Intervention-AI is modified so that it does not typically act, but can act when it chooses to intervene on behalf of the other player.

Appendix E: Understanding Helper-AI Behavior

Here we provide more detailed results on behavioral differences among agents. As an illustrative example of Helper-AI Behavior, we consider S_3, S_4, and the on-target Helper-AI agents $H(S_3)$ in Space Invaders. In cooperative self-play, S_3 achieves an average score of 2,141 over 100 games,

agent 2	cost	% intervene	score
S_1	0.05	2%	91.5
S_1	0.02	27%	111.8
S_1	0.01	52%	128.0
S_1	0.005	63%	170.3
S_2	0.05	1%	104.1
S_2	0.02	25%	151.6
S_2	0.01	50%	160.3
S_2	0.005	61%	171.7
S_3	0.05	1%	182.9
S_3	0.02	19%	203.8
S_3	0.01	47%	210.7
S_3	0.005	57%	214.0

Table 7: Two-player, Cooperative Fall Down. Game score with Intervention-AI, varying cost for intervention, averaged over 100 games, and with each of S_1, S_2 and S_3 as agent two. As the cost for intervention decreases, the Intervention-AI takes more actions, and the average score improves.

agent 2	cost	% effective
S_1	0.05	85%
S_1	0.025	81%
S_1	0.01	82%
S_2	0.05	82%
S_2	0.025	80%
S_2	0.01	79%
S_3	0.05	81%
S_3	0.025	80%
S_3	0.01	77%

Table 8: Two-player, Cooperative Space Invaders. Percent effective interventions, varying the cost for intervention, averaged over 100 games, and with each of S_1, S_2 and S_3 as agent two.

agent 2	cost	% effective
S_1	0.05	76%
S_1	0.02	72%
S_1	0.01	66%
S_1	0.005	64%
S_2	0.05	74%
S_2	0.02	70%
S_2	0.01	65%
S_2	0.005	65%
S_3	0.05	72%
S_3	0.02	71%
S_3	0.01	65%
S_3	0.005	65%

Table 9: Two-player, Cooperative Fall Down. Percent effective interventions, varying the cost for intervention, averaged over 100 games, and with each of S_1, S_2 and S_3 as agent two.

with about 49% of the points scored by $\mathcal{P}_1$ and 51% of the points scored by $\mathcal{P}_2$. When paired with Helper-AI, they achieve an average score of 3,844, and S_3 scores about 40% of the points (1,538 points on average). In other words, the S_3 agent actually plays substantially better with Helper-AI, scoring 40% more points on average, compared to the performance of one of the agents in cooperative self-play[20]. Since scoring points comes from hitting aliens – which appear in waves of 36 – an increased score corresponds to surviving more waves of attackers. Thus, we can directly observe the benefits of pairing Helper-AI agents with weaker agents: they actually make the weaker agents themselves perform better, as opposed to doing all of the work for them. [21]

Table 10 summarizes the results that indicate the reasons for episode termination. These results are explained in the main text, and they indicate that coordination increases if on of the agents in two-agent interaction S_i-S_i is replaced by Helper-AIs $H(S_i)$, and decreases if it is S_i is replaced by S_4.

[20]Looking at the raw number of enemies hit (as opposed to game score), the results are essentially unchanged.

[21]Surprisingly, when S_3 is paired with the more-skilled agent S_4, we see the opposite effect. Together, they achieve an average score of 1,826 points, with S_4 scoring about 54% of the points. This translates to S_3 scoring 840 points on average, i.e., playing about 30% worse when paired with S_4 than in cooperative self-play. In addition to being killed sooner, the miscoordination results in allowing the aliens to land more frequently, thus ending the game.

Reason for Episode Ending	Player 1	Player 2	Observed Probability
Player 1 Hit	S_2	S_2	40%
Player 2 Hit	S_2	S_2	38%
Both Players Hit	S_2	S_2	4%
Aliens Land	S_2	S_2	18%
Player 1 Hit	S_4	S_2	33%
Player 2 Hit	S_4	S_2	42%
Both Players Hit	S_4	S_2	0%
Aliens Land	S_4	S_2	25%
Player 1 Hit	$H(S_2)$	S_2	19%
Player 2 Hit	$H(S_2)$	S_2	60%
Both Players Hit	$H(S_2)$	S_2	6%
Aliens Land	$H(S_2)$	S_2	15%

Table 10: Comparing the reasons for episode termination in two-player, Cooperative Space Invaders over 100 games, with S2 as Player 2, and varying Player 1.

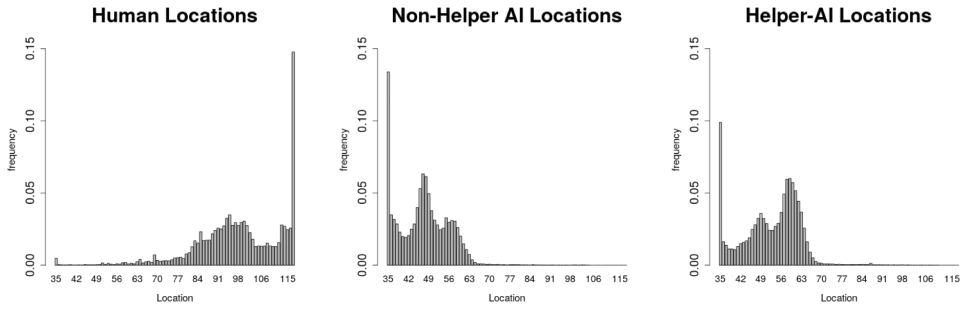

Figure 8: Location distribution of humans, Non-Helper AIs, and Helper-AIs in Space Invaders, over approximately 800,000 total locations. The possible locations range from 35 (left-most) to 117 (right-most). The human player starts at location 117 and the AIs start at location 35, and are automatically moved to these locations at the beginning of each new wave of aliens. Helper-AIs tend to spend less time at their initial location and play more in the center of the screen than non-Helper AIs.

Qualitative assessment of behavioral differences.

Interestingly, agents of different types (humans, Non-Helper-AIs, and Helper-AIs) have qualitatively different behavior. This can be seen from the location distributions of Non-Helper-AIs, Helper-AIs, and humans in Figure 8. Here, the possible locations range from 35 (the left-most location on the screen) to 117 (the right-most location on the screen). The human player starts each game at location 117, and the AIs start each game at location 35, and are automatically moved to these locations at the beginning of each new wave of aliens (i.e., when 36 aliens are hit). We observe that human players have a wider range than AIs, and spend somewhat more time at their initial location. Some human players also spend a non-trivial amount of time at the left-most location, suggesting that they are exploring the possible locations during the game, which we do not observe in AIs. Helper-AIs tend to spend less time at their initial location and play more in the center of the screen than non-Helper AIs.

The action distribution for Non-Helper AIs, Helper-AIs, and humans are provided in Figure 9, and the observed probabilities for action bigrams are in Tables 11, 12, 13. We observe that humans are much more likely not to act at all, and that Helper-AIs are less likely to move left (away from the human player) than Non-Helper AIs. Human players are less likely to take actions that require simultaneous key-presses (right-fire and left-fire). In addition, all players appear to prefer to take the same action repeatedly, but humans appear not to switch actions that would be tricky to execute physically (for instance, almost never taking the left action immediately after the right action). Also, Helper-AIs appear to be less likely to repeat the same action.

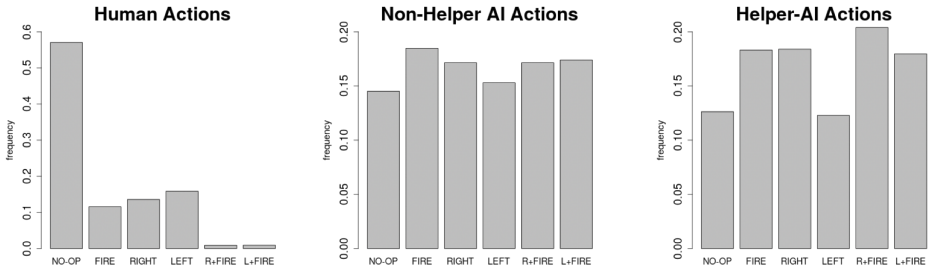

Figure 9: Action distribution of humans, Non-Helper AIs, and Helper-AIs in Space Invaders, over approximately 800,000 total actions. The possible actions are NO-OP (do nothing), fire, move left, move right, move right and fire, or move left and fire. Helper-AIs are less likely to move left (away from the human player) than Non-Helper AIs.

Human	NO-OP	FIRE	RIGHT	LEFT	R+FIRE	L+FIRE
NO-OP	85%	8%	3%	4%	0%	0%
FIRE	41%	55%	1%	1%	1%	1%
RIGHT	12%	1%	84%	0%	2%	0%
LEFT	12%	1%	0%	84%	0%	2%
R+FIRE	10%	13%	32%	0%	45%	0%
L+FIRE	10%	11%	0%	31%	0%	48%

Table 11: Observed probabilities for action bigrams in Space Invaders, for humans in human experiments. These represent the row-conditioned observed probabilities of action pairs for humans.

Non-HelperAI	NO-OP	FIRE	RIGHT	LEFT	R+FIRE	L+FIRE
NO-OP	41%	11%	14%	14%	10%	10%
FIRE	9%	41%	9%	8%	15%	19%
RIGHT	11%	10%	53%	7%	14%	5%
LEFT	13%	9%	8%	46%	7%	17%
R+FIRE	8%	16%	13%	5%	47%	10%
L+FIRE	8%	20%	5%	15%	9%	42%

Table 12: Observed probabilities for action bigrams in Space Invaders, using Non-Helper AI in human experiments. These represent the row-conditioned observed probabilities of action pairs for Non-Helper AIs.

Helper-AI	NO-OP	FIRE	RIGHT	LEFT	R+FIRE	L+FIRE
NO-OP	20%	18%	20%	14%	13%	16%
FIRE	11%	28%	13%	9%	16%	23%
RIGHT	14%	13%	39%	8%	18%	9%
LEFT	14%	13%	11%	29%	9%	23%
R+FIRE	10%	16%	18%	6%	39%	11%
L+FIRE	10%	20%	8%	14%	10%	38%

Table 13: Observed probabilities for action bigrams in Space Invaders, using Helper-AI in human experiments. These represent the row-conditioned observed probabilities of action pairs for Helper-AIs.

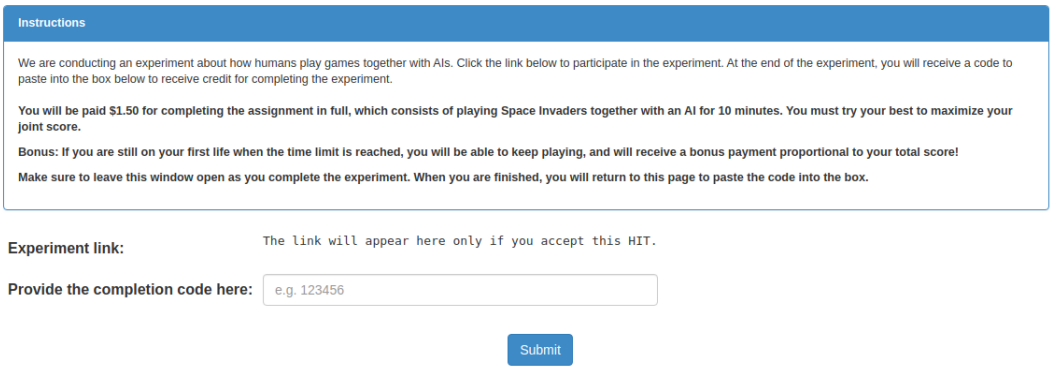

Figure 10: Instructions given to human subjects on Amazon Mechanical Turk.

Appendix F: Amazon Mechanical Turk Experiments

In this section we provide additional informtion on our AMT experiments. Figure 10 shows the instructions given to the participants before they choose whether to accept the task (Human Intelligence Task, abbreviated as HIT). As we metnioned in the main text, we limited participation to those based in the U.S., who had previously completed 10,000 tasks with a 98% approval rate, i.e., relatively experienced workers. We paid workers 1.50 USD for 10 minutes of gameplay, regardless of the number of games that they played, and ran experiments on weekdays from 4 - 8 PM EDT / 1 - 5 PM PDT. Games from subjects that did not interact with the system or that left before the entire 10 minute experiment was complete were dropped from the dataset. The humans were unaware which AI they were playing together with, and whether it was supposed to be helpful or not. To help incentivize the participants to play their best, we added the possibility of a bonus payment of up to 1.50 USD if they were still on their first game during the 10 minute experiment (i.e., neither the participants nor the AI partner had lost a life). At this point they could chose to extend game play. None of the subjects chose to do so.