WIP: Human-AI interactions in real-world complex environments using a comprehensive reinforcement learning framework

Md Saiful Islam[1], Srijita Das[1], Sai Krishna Gottipati[2], William Duguay[2], Clodéric Mars[2], Jalal Arabneydi[3], Antoine Fagette[4], Matthew Guzdial[1,5], Matthew E. Taylor[1,2,5]
[1]University of Alberta, [2]AI Redefined Inc, [3]JACOBB, [4]Thales, [5]Amii
{mdsaifu1, srijita1, guzdial, matthew.e.taylor}@ualberta.ca, {sai, william, cloderic}@ai-r.com, jalal.arabneydi@jacobb.ai, antoine.fagette.e@thalesdigital.io

ABSTRACT

Deep reinforcement learning (RL) has successfully tackled many real-world tasks. However, these algorithms suffer from the well-known sample-inefficiency problem. Deep RL systems usually require millions of environment interactions to learn and have stable performance. In this work, we show that human-AI teams outperform human-only controlled and fully autonomous teams for complex tasks. We develop a novel simulator for a critical infrastructure scenario and a user interface for humans to effectively advise AI agents. We show that humans can provide useful advice to the RL agents, allowing them to improve learning in a multi-agent setting.

KEYWORDS

Multi Agent RL, Human-AI Teaming, System

1 INTRODUCTION

Protecting a critical infrastructure such as an airport against threats is a complex, sensitive, and expensive task, leading to a history of exploring automated solutions [11]. However, complete automation of such a defense system is inadvisable due to its importance. Conversely, the task of continuously monitoring such systems and quickly assessing and handling potential threats would benefit from AI capabilities. One potential solution would then be to build hybrid systems that rely on the strengths of both human operators and autonomous systems to decrease costs and increase performance, while maintaining meaningful human control in dangerous or critical operations [26].

We developed a system where humans and AI agents can collaborate to defend an airport's airspace from drone incursions. In this work, we demonstrate experimental evidence that indicates that AI drone training can be improved by using human-like demonstrations. The developed platform also provides opportunities for further experimentation, such as: 1) increasing the confidence in AI agents by their human operators, 2) accelerating the training process by injecting human expertise, and 3) investigating various collaboration and advice modalities between humans and AI agents. In recent times, reinforcement learning (RL) has had many successes in solving many decision-making problems ranging from the game of Go [28] to deploying a super-pressure balloon in stratosphere [5]. While several domains like Atari and Mujoco exist to benchmark current state-of-the-art RL research [19, 33], simulators specific to real-world scenarios are rare. We develop a novel simulator for the aforementioned airspace and airportś restricted zone protection system. The use case consists of a fleet of ally (blue) drones trying to protect a restricted airspace from multiple enemy (red) drones. Following recommendations from experts in the field of air defense, the simulator is designed to mimic with reasonable simplifications real-world dynamics with respect to the speed of the drones, their flight dynamics as well as the specifications of the ground radar sensor, the sensing payloads (radar and electro-optical) embedded on the blue drones and the neutralization payloads embedded on the blue drones. Real-world dynamics make the environment complex. The complexity of the environment means that a naive RL application would require numerous environment interactions. Given the expense and risk associated with these interactions in a real environment, we would like to minimize them. We demonstrate that agent demonstrations can minimize the number of required environment interactions.

To investigate the effects of human and agent demonstrations, we compare policy networks trained in our simulator using human demonstrations, agent demonstrations, an agent's own experiences, and the experiences of blue drones. We use non-expert human and agent demonstrations to showcase the robustness of our approach to address limited availability of human experts.

The key contributions of this work include the following:

(1) developing a novel multi-agent simulator for a defense-specific use case modeling real-world dynamics;
(2) using state-of-the-art RL algorithms to train agents in our simulator;
(3) developing a user interface for our simulator, which enables human operators to dynamically take the place of an agent to produce in-context demonstrations; and
(4) demonstrating empirically that trained agent demonstrations or human-again mix demonstrations help agents to learn faster.

2 RELATED WORK

There is a large amount of existing literature on using external knowledge from different sources to make RL agents sample-efficient [2] [6], where this external knowledge can originate from humans or other agents. Early examples of using human knowledge in decision making relied on collecting demonstrations from experts, popularly known as imitation learning [27]. However, these methods usually incur human costs in terms of attention and availability of experts. Although imitation learning methods are still popular, a significant amount of existing work uses human preferences and advice, rather than demonstrations, from experts to better guide RL agents [7, 23, 29].

Advice in teacher-student frameworks: In a teacher-student framework [31], a more knowledgeable or skilled teacher provides advice to a student agent to improve its sample efficiency. Torrey et al. [34] proposed various heuristics, such as early advising and importance advising, to decide when to provide advice to the student. In addition to multi-agent reinforcement learning, Silva et al. [8] proposed a framework for agents to learn simultaneously from each other based on their roles (teacher or student). Omidshafiei et al. [22] introduced a general "learning to teach" framework in cooperative settings where each agent can be a teacher or student and can learn when and what to advise within a fixed budget. Kim et al. [14] proposed a scalable learning-to-teach framework, which addresses the teacher credit assignment problem and considers the impact of teacher's advice on the student's learning. While many teacher-student frameworks concentrate on a single teacher and student, or multiple teachers [13], our focus is on a single teacher with multiple students. In our research, the teacher is either a trained agent or a real human operator, while the students are learning agents. This kind of teacher-student framework had not been examined in a defense-specific environment.

Demonstration to guide RL: As discussed above, learning from demonstrations [32] has been a common approach with Deep RL. Hester et al. [10] first leveraged demonstrations inside an existing deep RL algorithm (DQN) by using a supervised loss function to model demonstrations. Later, demonstrations were used to speed up DDPG in complex robotics tasks [21, 36] by using specific techniques like behavior cloning loss and prioritized replay buffers. Goecks et al. [9] provided a unified loss function by integrating loss function components from prior works. We use some of these loss functions in our approach, as explained in detail in Sections 3 and 5.3.

In addition to demonstrations, human trainers have used other modalities such as feedback to build an explicit model of a human's reward model inside the **TAMER** framework [15]. Knox et al. initially trained a classifier on explicit human signals and used this model to infer the best action in guiding an agent. The human feedback is provided in TAMER in the form of agreement or disagreement within a certain window while watching the agent in action. Later, this framework was combined with RL [16, 17] where the human reward model acts as a shaping function to guide the RL agent. This was later extended to deep learning with respect to the reward model [39] and Q-function [1]. Similarly, Arumugam et al. [3] propose **COACH** framework, a method for training RL agents using feedback from human operators. The approach involves training a deep neural network to predict human feedback and using this predicted feedback to optimize the agent's policy. MacGlashan et al. [18] propose a framework that enables an agent to learn from the feedback provided by a human expert who provides feedback in the form of "preference queries," which ask the expert to compare two policies and indicate which one is better. Our simulator is also an experimentation platform for benchmarking all these different kinds of modalities. Tambe et al. [30] demonstrated plans for Stackelberg games can be executed with human-agent teams (HATs), but did not explore real-time coordination with humans. Our work focus on human demonstration to gain significant performance improvements utilizing both human and agent knowledge in real-life applications.

Securing critical national infrastructure, such as airports, is a significant challenge for security agencies worldwide due to the threat of enemy attacks, and only a few research studies have addressed it. Pita et al. [24] proposed randomizing security schedules using a game-theoretic approach to enhance airport security. Khalil et al. [12] used a federated RL-based framework for unmanned aerial vehicles to operate in a dynamic defence system. In contrast, our focus is on using an autonomous multi-agent and efficiently incorporating human knowledge. Başpınar et al. [4] used a cooperative DRL model to maximize survival considering both individual and collective actions of unmanned aerial vehicles. Additionally, Venugopal et al. [37] incorporated uncertainty into their DQN model, which combines drone signalling, notification, and defender allocation to protect valuable resources. However, our work primarily concentrates on improving the human-agent team's performance by using human knowledge in a more complex and realistic defence-specific environment.

3 BACKGROUND

Markov decision process: RL, multi-agent RL, or other human-in-the-loop learning algorithms are often built on the Markov decision process (MDP) formulation. An MDP is defined by $\langle \mathcal{S}, \mathcal{A}, R, \mathcal{P}, \gamma \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, R is the reward function, $\mathcal{P}$ is the transition probability function, and $\gamma \in [0, 1)$ is the discount factor. At any time step t of the episode, the agent is in state $s_t \in \mathcal{S}$, executes an action $a_t \in \mathcal{A}$ based on its policy π and current observation o_t (i.e., $a_t \sim \pi(\cdot|o_t)$). The agent then transitions to the next state $s_{t+1} \in \mathcal{S}$ according to the transition probability function $\mathcal{P}$ and receives a reward $r_t \in R$. The goal of an RL algorithm is to maximize the expected discounted sum of rewards by optimizing its policy π.

The Q value function $Q^{\pi}(s, a)$ of a given state-action pair (s, a) determines the expected future reward starting from the given (s,a) and following the policy π. The optimal value function $Q^*(s, a)$ provides the maximal values in all states and is determined by the Bellman equation:

$$Q^*(s, a) = \mathbb{E}\left[R(s, a) + \gamma \sum_{s'} P\left(s' \mid s, a\right) \max_{a'} Q^*\left(s', a'\right)\right]$$

Deep Q networks: Deep Q networks (DQNs) were introduced in [20], where neural networks were used to predict the Q values and are updated using the Bellman equation. Further improvements were shown in double Q networks [35] by using two Q networks instead of just one and in dueling Q networks [38] where the Q network consisted of seperate heads for estimating value and advantage functions. The combination of these approaches was commonly referred to as D3QN (double dueling DQN) and is our baseline algorithm in this work.

Following the notation in [10], the double DQN loss is given by:

$$J_{DQ}(Q) = \left(R(s, a) + \gamma Q\left(s_{t+1}, a_{t+1}^{\max}; \theta'\right) - Q(s, a; \theta)\right)^2$$

where $\theta\hat{}\prime$ are the parameters of the target network and $a_{t+1}^{\max} = \operatorname{argmax}_a Q\left(s_{t+1}, a; \theta\right)$.

Deep Q networks from demonstration: DQN from demonstrations (a.k.a. DQfD) was introduced [10] to learn from human demonstrations in addition to agent's own experience. Firstly, for pretraining using demonstrations in a supervised fashion, margin classification loss was used.

$$J_E(Q) = \max_{a \in A} \left[Q(s,a) + l\left(a_E, a\right) \right] - Q\left(s, a_E\right)$$

where a_E is the action taken by the demonstrator in state s. $l(a_E, a)$ is a margin function that is 0 when $a = a_E$ and positive otherwise. To improve pretraining, the authors also proposed adding the n-step return to help propagate the values of the demonstrator's trajectory to earlier states. The n-step return is:

$$r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \max_a \gamma^n Q\left(s_{t+n}, a\right)$$

The subsequent n-step loss, which takes into account the n-step return, is denoted as $J_n(Q)$. Furthermore, an L2 regularization loss was added to the parameters of the network to prevent it from over-fitting to the relatively small demonstration dataset. Thus, the overall loss used to update the network is:

$$J(Q) = J_{DQ}(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q) + \lambda_3 J_{L2}(Q) \tag{1}$$

Hester et al. used DQfD in the context of a single-agent RL setting. In this paper, we extend it to a multi-agent RL setting where we train all five ally drones based on demonstrations and combined experiences of all the agents. We elaborate further on these details in Sections 4 and 5.

4 PROBLEM FORMULATION

This section describes the problem setting, its formalization as an MDP, and the architecture for the simulator and user interface.

4.1 Environment design

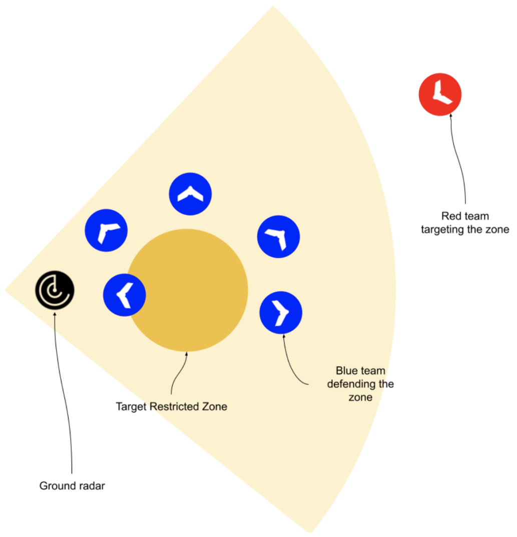


Figure 1: Environment

In this paper, we introduced an airport defense simulator and explored the impact of human feedback on multi-agent RL problems. The environment used for our problem formulation is shown in Figure 1. There are two teams, the ally (blue) and the enemy (red). The blue drones can be autonomous or controlled by a human. Each drone is equipped with electro-optics. The blue team consists of five drones, a ground radar sensor and a ground control station (GCS). Each ally drone also has several neutralization payloads (i.e., devices capable of neutralizing enemy drones when they are within a certain range). The red team comprises a single drone equipped with its radar sensor and a potentially hazardous payload. The goal of the blue team is to protect the restricted zone of the airport from the red team by detecting, localizing and neutralizing the enemy drones.

In this environment, the blue and red drones has a partially observed view of the environment. The detection and localization of the red drone provided by the radar and EO sensors embeds noise and uncertainty. As per the defense expert's suggestions, we intentionally introduce errors into the system to simulate these real-world factors. Specifically, the detection probability of the radar is 95% and there is a 5% probability of the radar failing to detect the red drone. Moreover, the radar fails to detect the enemy drone if it is outside the radar range. The detection frequency is set to 1 Hz and the maximum speed of the drones is ten meters per second. The range of the neutralization payloads embedded on the blue drones is set to ten meters. All these dynamics make the scenario complex and require the blue drones to anticipate the trajectory of the red drone.

The experimental platform is built around a simplified airspace simulator operating in 2D. Although simplified, several aspects have been modelled following real-world specifications based on feedback from domain experts, such as the detection capabilities of the drone sensors and the radar, as well as the dynamics of the rotary-wing drones.

4.2 MDP formulation

We define the MDP for this domain as follows.

(1) **Observation space:** The observation space consists of the relative positions of the red drone, the blue drones, and the restricted airspace over 3 time-steps. Each agent has a partial observation of the environment. Each blue agent has an observation that includes 1) the relative x and y distance to the red drone (in meters) and 2) the relative x and y distance to the center of the restricted zone (in meters). To implicitly account for velocity, we stack three time steps together, for a total of $(2+2) \times 3 = 12$ features.
(2) **Action-space:** The action-space is continuous, varying from [-1,1]. We discretize the action-space into 2 discrete actions: positive and negative angles of rotation.
(3) **Reward function:** The blue drones receive a positive reward if they successfully neutralize the red drone and a negative reward if red drone enters the target area.
The team's reward is defined as follows

$$R(s) = \begin{cases} +1 & \text{if any blue drone neutralizes the red drone} \\ -1 & \text{if the red drone enters the restricted zone} \end{cases}$$

Additionally, at every time step, the blue drones receive a shaping reward $R_I(s)$ proportional to their relative distance from the red drone in consecutive time steps; $R_I(s) = (d_{t-1}(b,r) - d_t(b,r))$ where $d_t(b,r)$ and $d_{t-1}(b,r)$refers to the relative distance between the blue and red drone at time-step t and $(t-1)$ respectively.

We train the agents in a multi-agent centralized training and execution setting where each agent has it's own observation-space as defined in the MDP formulation. Each agent has the same reward function and action space. We train these agents in parallel using Cogment [25]. Cogment is an open-source platform enabling training and operating various kinds of multi-agent RL and human-in-the-loop learning algorithms in a distributed way, due to its microservice architecture.

4.3 User Interface and Architecture for HIL Interactions

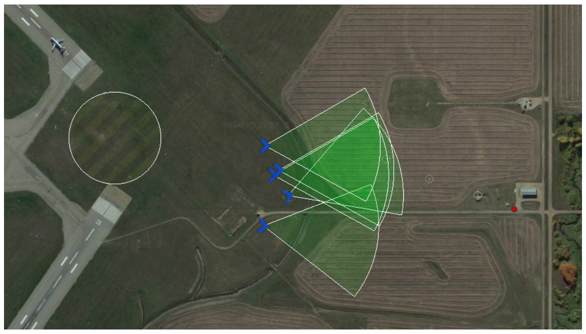


Figure 2: User interface for human operators to control the agents

In order for the human operator to control the ally drones, we have developed a user interface as shown in Figure 2. The blue drones can be either fully autonomous, fully human operated, or hybrid (i.e., control is shared by the human-agent team). This is implemented by two Cogment actor implementations (drone and human actors), shown in Figure-3. For each drone agent in the simulation, Cogment instantiates two actors using those implementations, it can then dynamically assign the control of the drone entity to one of them. The human operator can select drones and set specific waypoints for them by predicting the enemy drone's anticipated trajectory. In Figure 2, the waypoints are denoted by grey circles on the map. Once a specific waypoint has been defined, Cogment dynamically gives control of the associated ally drone to the operated agent causing it to move towards the defined waypoint via the shortest path under the standard physical dynamics constraints. Similarly, the human operator can delete existing waypoints through the interface to rectify any errors in predicting the enemy drone trajectory. Conversely, when no waypoints are defined, the control is given back to the autonomous agent. The interface also allows the human to operate at 3 simulation speeds (1x, 2x, 5x), according to their preferences.

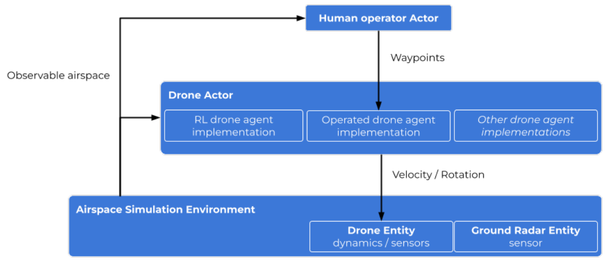

Human operator Actor
Waypoints
Observable airspace
Drone Actor
RL drone agent implementation
Operated drone agent implementation
Other drone agent implementations
Velocity / Rotation
Airspace Simulation Environment
Drone Entity dynamics / sensors
Ground Radar Entity sensor

Figure 3: Airspace simulation and hierarchical multi-agent modeling

5 EXPERIMENTS

In this section, we report the experimental settings and the results of our comparative analysis of different ways forRL approaches to receive guidance in our simulator.

5.1 Environment configuration

Our study focuses on a simulated environment created specifically for this research, as described in Section 4.1. At the start of each episode, all drones start from random positions , with the blue and red drones having the same speed. To simplify the environment, we sampled the starting position of the blue drones from a circular region with a radius of 1000 meters located near the restricted zone. In contrast, the red drone's starting position is sampled from a similar circular region located atleast 1000 meters to the right of the restricted zone. Additionally, we added three waypoints between the blue and red drones to facilitate the simulation. These waypoints guide the red drone to reach the restricted space within a fixed time-limit.

5.2 Performance metric

We evaluated the performance of the trained agent using the *success rate* as the performance metric. The success rate is defined as the percentage of times the blue team wins over all evaluation trials. This metric was chosen as it provides a clear and intuitive measure of the agents' ability to defeat the red drone and is directly proportional to the average reward. We execute thirty evaluation episodes per hundred training episodes to compute the success rate. During the evaluation episodes, agents do not learn or explore. Our learning curves show the performance metric reported in the evaluation episodes. We also averaged the results over five runs with different seed values to account training and environmental stochasticity.

5.3 Experimental setup

For our experiments, we use a fully trained D3QN agent for generating 2, 500 agent demonstration. We collect 250 human demonstrations from seven humans that completed at least 30 episodes. Our baseline was D3QN, training from scratch without additional guidance. We adapt prior work [10] to leverage demonstrations inside D3QN from either another agent or from a mix of human and agent demonstrations[1] We use D3QN-PH to represent D3QN agents trained with additional agent demonstrations and D3QN-MH to represent D3QN agents trained with a mix of agent and human demonstrations. For experiments with D3QN-PH and D3QN-MH, we sampled thirty percent demonstration samples from real human or trained agent replay memory and seventy percent from agent replay memory consisting of past agent experiences.

Algorithm abbreviation	Human Demo	Agent Demo
D3QN	0	0
D3QN-PH	0	2,500
D3QN-MH	500	2,000

Table 1: Algorithm abbreviation with # of human and agent demo for each algorithm.

After testing our models with various proportions of demonstration data, we determined that the performance of the learning process was not significantly affected by the agent and demo proportions. Hence, we set these proportions to 70% agent experience and 30% demonstration experience to get adequate learning. To update the network, the training algorithm sampled mini-batches from the demonstration data and applied three loss functions: the double Q-learning loss and the n-step double Q-learning loss as described in Section 3. In initial experiments, the $L2$ regularization loss and margin classification loss failed to improve learning and these losses were not used. Moreover, we did not use any pre-training for any of the reported algorithms. The Q-function of DQN is updated by Equation 3, D3QN-PH and D3QN-MH are updated using Equation 1. The expert margin in Equation 3 was set to $M = 0.8$ in alignment with prior work [10]. A heuristic agent was used as a basic benchmark, which is a manually coded agent that relies on the distance between the blue and red drones. The heuristic agent always directs the closest blue drone towards the red drone, and when it reaches the neutralization range, this blue drone immediately neutralizes the red drone.

The human-demo in Figure 4 refers to the average winning percentage of actual human demonstrators. The winning percentage for average human demonstration is 62% with standard deviation of 17%. Human participants are researchers, software engineers, and graduate students from the University of Alberta and partner organizations. We only considered demonstrations where either the agent or human won the game to account for good quality of demonstrations. The results in Figure 4 report the mean and standard deviation over 5 runs, with the y-axis indicating the winning % and the x-axis denoting the number of training episodes.

We perform hyper-parameter tuning to find reasonable parameters for the algorithms. We considered learning rate values of [0.4, 0.04, 0.004, 0.0004, 0.00004], epsilon decay values of [0.99, 0.995, 0.9995, 0.99995, 0.999995], and discount factor values of [0.9, 0.99, 0.999]. The supervised loss coefficient weight (λ_2 in Equation 1) varied between 10^5 to 1, and we set λ_3 to 0. The same network structure was used across D3QN, D3QN-PH, and D3QN-MH, consisting

[1]Agent demonstrations refer to demonstrations generated by a D3QN agent that was trained to an average performance of 85% +/- 10%.

Parameter	Value
Training Episodes	10,000
Replay Memory size	100,000
Batch size	64
Learning rate	0.0004
Discount factor	0.99
Target network update frequency	10
Initial ϵ	1.0
Final ϵ	0.05
ϵ decay per episode	0.999995

Table 2: Model hyper-parameters.

of two hidden layers with 64 fully connected neurons. A final fully connected layer was added to represent each action's Q-values. The non-linearity function used in all layers was rectified linear units (ReLU). During training, we use the Adam optimizer and applied an epsilon-greedy policy, gradually reducing epsilon from 1 to 0.05. The batch size was 64 and the replay memory size was $1,00,000$.

5.4 Results

We aim to investigate the following two research questions:

(RQ1) How well does a trained agent perform in this specific environment?

(RQ2) Do agent or human demonstrations help make the RL agent more sample efficient?

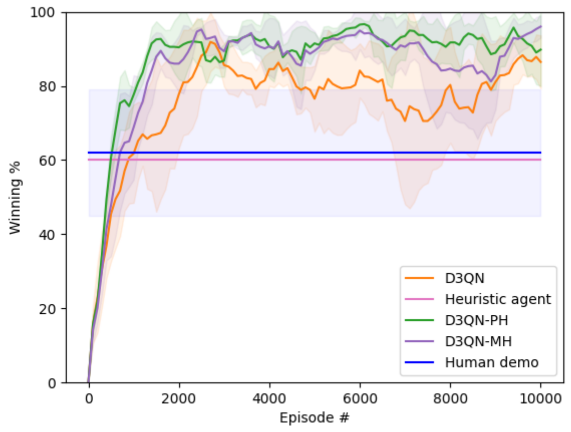
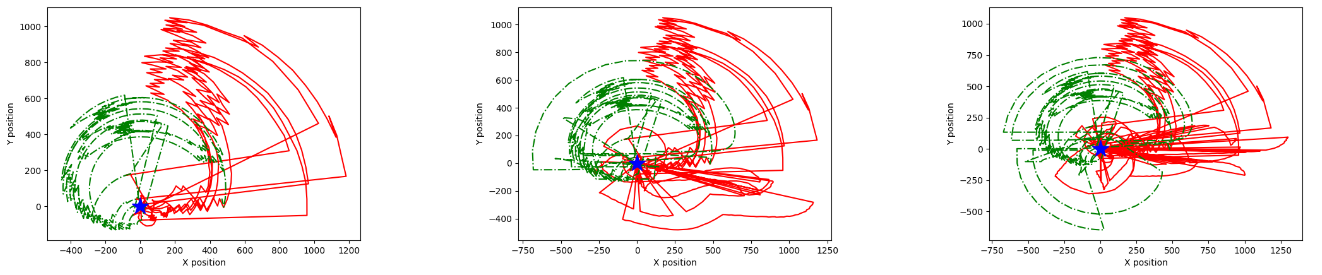


Figure 4: The success rate comparison for D3QN, D3QN with trained agent demonstrations, D3QN with real human and trained agent mix demonstrations, and a heuristic baseline. Here, the suffix -PH represents demonstrations from a trained agent and -MH indicates a mixture of real and trained agent demonstrations.

To answer RQ1, we train a D3QN and plot its performance in Figure 4. The agent reaches a success rate of roughly 90% in 3500 episodes. The trained agent outperforms the baseline heuristic



(a) trained agent demonstration

(b) a human user (User 1)

(c) a human user (User 2)

Figure 5: Visual representation of five episodes from trained agent and two different real human users

agent, which has a success rate of 60%. The average human performance is around 63%, which is almost equal to the heuristic agent. This is because of two reasons: (1) there is a relatively small number of human demonstrations (250 winning episodes) and (2) the game has a learning curve and our human participants require some time to get used to the interface and the dynamics of the game.

To answer RQ2, we identify that D3QN-PH reaches a success rate of more than 90% in 1600 episodes, outpacing D3QN as shown in Figure 4. We performed an unpaired t-test to examine the significant difference between the performance of the D3QN-PH and D3QN agents. The D3QN-PH agents exhibit a statistically significant performance as compared to D3QN ($p < 0.0001$) and the effect size is 1.43. At the end of learning, both algorithms converge to the same final performance level (around 90%). This provides support for our claim that trained agent demonstrations make the RL agent more sample efficient in our environment, consistent with existing results in the literature [10, 21].

5.5 Pilot study using human demonstration

We also trained the learning agent with a mix of agent and actual human demonstrations, denoted as D3QN-MH as shown in Figure 4. We sampled an equal proportion of human and trained agent demonstrations in every mini-batch used for training. We used a mix of both types of demonstrations because of the lack of human demonstrations collected in our ongoing pilot study. The results do not suggest a significant learning improvement when compared to D3QN-PH; however, the performance is still statistically significant than the baseline D3QN.

Since, this simulation has a learning curve for humans because of the interface and the dynamics, we intend to collect more demonstrations from humans and to include a burn-in period for humans to understand the environment and learn to play before collecting demonstrations. We visualized five trajectories of all blue and red drones from trained agent demonstrations and two actual human demonstrations from different users who played more than 30 games, as shown in Figure 5. In this figure, the blue star denotes one of the ally drones that neutralized the enemy drone and is the frame of reference (located at $(0, 0)$). The red and green lines represent the relative position of the enemy drone and the restricted airspace (with respect to the blue drone's position). Figure 5(a) shows five trajectories of ally drones generated from the trained D3QN agent. We note that across all the figures, the red drone starts moving towards the restricted zone while being chased by the blue drones until it is neutralized. The low density of red lines around the blue star indicates that the blue drones quickly neutralizes the red drone without following it for a long time. From Figure 5(b) and 5(c), which depicts trials by two different human participants, we notice that there is more movement (high-density) around the blue star, suggesting that the human tries setting waypoints in different areas (using the whole team of five blue drones) of the map to neutralize the red drone. These trajectories are sub-optimal (longer trajectory length) as compared to the trajectories from trained agent demonstrations. However, these might be helpful to neutralize the red drones in difficult environment configurations where the trained RL agents fails to catch the enemy drone (trained agents have a failure rate of around 10% in this task).

6 ETHICS STATEMENT

Our human subject study was approved by the University's ethics board (REB number: Pro00107555). We have designed the simulator environment so that drones are not equipped with weapons that can directly endanger the lives of humans. We are also focusing on a defensive task to minimize the risk of our work being used by bad actors.

7 CONCLUSION

To conclude, we developed a novel defense-specific simulator that reflects read-world dynamics. We further showed that in such a complex domain, a trained agent and AI combination is effective compared to fully human or fully autonomous operations. As part of an ongoing effort, we want to show that actual humans can be effective in such domains where many interactions and specificity are involved. As part of future work, we want to customize the user interface to seamlessly handle different modalities of human advice. We also want to experiment with humans having different expertise over different tasks in this domain. Finally, we want to handle different modalities of advice depending on when they are most effective for agents and convenient for humans.

8 ACKNOWLEDGEMENTS

Part of this work has taken place in the Intelligent Robot Learning (IRL) Lab at the University of Alberta, which is supported in part by research grants from the Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, Amii; Compute Canada; Huawei; Mitacs; and NSERC.

REFERENCES

[1] Riku Arakawa, Sosuke Kobayashi, Yuya Unno, Yuta Tsuboi, and Shin-ichi Maeda. 2018. DQN-TAMER: Human-in-the-Loop Reinforcement Learning with Intractable Feedback. *arXiv preprint arXiv:1810.11748* (2018).

[2] Mauricio Fadel Argerich, Jonathan Fürst, and Bin Cheng. 2020. Tutor4RL: Guiding Reinforcement Learning with External Knowledge.. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering (1)*.

[3] Dilip Arumugam, Jun Ki Lee, Sophie Saskin, and Michael L Littman. 2019. Deep reinforcement learning from policy-dependent human feedback. *arXiv preprint arXiv:1902.04257* (2019).

[4] Barış Başpınar. 2022. Deep Reinforcement Learning-Based Cooperative Survivability Maximization for a UAV Fleet on an Air-to-Ground Mission. *Journal of Aeronautics and Space Technologies* 15, 2 (2022), 94–107.

[5] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. 2020. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature* 588, 7836 (2020), 77–82.

[6] Adam Bignold, Francisco Cruz, Matthew E Taylor, Tim Brys, Richard Dazeley, Peter Vamplew, and Cameron Foale. 2021. A conceptual framework for externally-influenced agents: An assisted reinforcement learning review. *Journal of Ambient Intelligence and Humanized Computing* (2021), 1–24.

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

[8] Felipe Leno Da Silva, Ruben Glatt, and Anna Helena Reali Costa. 2017. Simultaneously learning and advising in multiagent reinforcement learning. In *Proceedings of the 16th conference on autonomous agents and multiagent systems*. 1100–1108.

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

[10] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. 2018. Deep q-learning from demonstrations. In *AAAI*.

[11] Robert H Kewley and Mark J. Embrechts. 2002. Computational military tactical planning system. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 32, 2 (2002), 161–171.

[12] Alvi Ataur Khalil and Mohammad Ashiqur Rahman. 2022. FED-UP: Federated Deep Reinforcement Learning-based UAV Path Planning against Hostile Defense System. In *2022 18th International Conference on Network and Service Management (CNSM)*. IEEE, 268–274.

[13] Chaitanya Kharyal, Tanmay Kumar Sinha, SaiKrishna Gottipati, Srijita Das, and Matthew E. Taylor. 2022. Do As You Teach: A Multi-Teacher Approach to Self-Play in Deep Reinforcement Learning. In *Deep Reinforcement Learning Workshop NeurIPS 2022*. https://openreview.net/forum?id=KEH4KSoJh2W

[14] Dong-Ki Kim, Miao Liu, Shayegan Omidshafiei, Sebastian Lopez-Cot, Matthew Riemer, Golnaz Habibi, Gerald Tesauro, Sami Mourad, Murray Campbell, and Jonathan P How. 2011. Learning hierarchical teaching policies for cooperative agents. In *AAMAS*.

[15] W Bradley Knox and Peter Stone. 2009. Interactively shaping agents via human reinforcement: The TAMER framework. In *KCAP*.

[16] W Bradley Knox and Peter Stone. 2010. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *AAMAS*.

[17] W Bradley Knox and Peter Stone. 2012. Reinforcement learning from simultaneous human and MDP reward. In *AAMAS*.

[18] James MacGlashan, Mark Ho, Robert Loftin, Bei Peng, Guan Wang, David L. Roberts, Matthew E.Taylor, and Michael L. Littman. 2017. Interactive Learning from Policy-Dependent Human Feedback. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[21] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 6292–6299.

[22] Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P How. 2019. Learning to teach in cooperative multiagent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6128–6136.

[23] Jongjin Park, Younggyo Seo, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. 2022. SURF: Semi-supervised reward learning with data augmentation for feedback-efficient preference-based reinforcement learning. *arXiv preprint arXiv:2203.10050* (2022).

[24] James Pita, Manish Jain, C Western, P Paruchuri, J Marecki, M Tambe, F Ordonez, and Sarit Kraus. 2009. Armor software: A game theoretic approach to airport security. *Protecting Airline Passengers in the Age of Terrorism* (2009), 163.

[25] A. I. Redefined, Sai Krishna Gottipati, Sagar Kurandwad, Clodéric Mars, Gregory Szriftgiser, and François Chabot. 2021. Cogment: Open Source Framework For Distributed Multi-actor Training, Deployment & Operations. *CoRR* abs/2106.11345 (2021). arXiv:2106.11345 https://arxiv.org/abs/2106.11345

[26] Filippo Santoni de Sio and Jeroen Van den Hoven. 2018. Meaningful human control over autonomous systems: A philosophical account. *Frontiers in Robotics and AI* 5 (2018), 15.

[27] Stefan Schaal. 1999. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences* (1999).

[28] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

[29] Theodore R Sumers, Mark K Ho, Robert D Hawkins, Karthik Narasimhan, and Thomas L Griffiths. 2021. Learning rewards from linguistic feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 6002–6010.

[30] Milind Tambe. 2011. *Security and game theory: algorithms, deployed systems, lessons learned.* Cambridge university press.

[31] Matthew E Taylor, Nicholas Carboni, Anestis Fachantidis, Ioannis Vlahavas, and Lisa Torrey. 2014. Reinforcement learning agents providing advice in complex video games. *Connection Science* 26, 1 (2014), 45–63.

[32] Matthew E. Taylor, Halit Bener Suay, and Sonia Chernova. 2011. Integrating Reinforcement Learning with Human Demonstrations of Varying Ability. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS).*

[33] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

[34] Lisa Torrey and Matthew Taylor. 2013. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. 1053–1060.

[35] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[36] Mel 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).

[37] Aravind Venugopal, Elizabeth Bondi, Harshavardhan Kamarthi, Keval Dholakia, Balaraman Ravindran, and Milind Tambe. 2021. Reinforcement Learning for Unified Allocation and Patrolling in Signaling Games with Uncertainty. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 1353–1361.

[38] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1995–2003.

[39] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. 2018. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *AAAI*.