Investigating the Effect of Human-in-the-Loop Teaching on Model Performance

Prifysgol Abertawe
Swansea University

Megan Ford

Faculty of Science and Engineering
Swansea University

This dissertation is submitted for the degree of
Master

September 2023

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 40,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 100 figures.

Megan Ford
September 2023

Acknowledgements

I would like to acknowledge my supervisor, Dr Muneeb Ahmad, for his invaluable guidance and input throughout our discussions.

Abstract

Interactive Reinforcement Learning (IRL) is a machine learning technique which incorporates a human-in-the-loop training approach. The IRL agent learns from human provided feedback, which can be delivered through several different forms, for example, scalar value rewards. IRL has been proven to be a valuable training method for agents, demonstrating faster learning times and decreased exploration. However, much of the current research has been constrained to limited, discrete environments, where agents have few possible actions to choose from in a given state. It is also known that there are certain limitations or hindrances associated with IRL; for example, agents are susceptible to learning human biases. This project investigates the effects of IRL in a larger environment, formalised as a Partially Observable Markov Decision Process (POMDP). A pre-trained RL agent was subject to an extra layer of training, in which participants of a user study were asked to observe and provide feedback to the agent following each action that was taken. Once training was complete, users interacted with both the RL and IRL agent whilst performing the same task. Results show that the IRL model outperforms the RL agent, confirming that a combination of both human and environmental rewards may be of value for complex, real-world environments. There is also evidence that the IRL training effected the state-action space of the model, improving overall task strategy. Future work intends to investigate the effects of human biases.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Project overview

Interactive Reinforcement Learning (IRL) is a subsection of Machine Learning algorithms that feature a human-in-the-loop dynamic to their learning processes. Similarly to Reinforcement Learning, IRL architecture includes an environment, an agent, states and actions and finally a reward function. Additionally, IRL integrates feedback provided by a human, often an expert in the context of the environment that the agent is learning within. There are number of ways in which a human can provide feedback to the agent in an IRL environment. These come in forms such as; binary critiques, scalar values, guidance and action advice [1].

Research has found relevant applications for IRL such as robotics [1,2] and gaming [3,4]. IRL is also thought to be highly applicable to the HCI field, with research in areas such as social robotics [5]. Cruz et al found that much of the IRL research carried out in HCI focuses on the improvement of algorithm performance, whilst the quality of the interactions are rarely evaluated [1]. The transfer of human knowledge to an agent, through the means of rewards, or other forms of guidance, is an important part of learning; particularly in complex environments, where the human interactions come from experts of that environment [6]. IRL has also been used in an attempt to speed up training times; RL agents often work well but require large amounts of time and data to begin showing good performance levels [6]. Through an evaluation of Interactive Machine Learning systems (IML), Boukhelifa et al. found that human-centred design and evaluation were also imperative in the addressing of black box issues that are often present in machine learning models [7].

Agents that learn in an environment which utilises human feedback, such as IRL, are susceptible to learning a human's inherent bias. This bias is caused by a person's mental model of the task or environment that they are providing feedback in. A mental model in this context, is a person's internal representation of the task based on their experiences of it in the real world [8]. Human bias can be transferred to an agent in a number of different forms, such as reward bias and preferential bias. Reward bias is the tendency for a human teacher to provide an agent with an inflated (more positive) reward, in an attempt to motivate an agent to learn [2], much like how teachers motivate human learners. In this project, preferential bias is defined by a human's preferred strategy to complete a task. For example, if an agent learns how to play a game from a human teacher who plays the game one specific way each

time, this is a preferential bias to that strategy.

1.2 Problem

The use of Reinforcement Learning (RL) in domains such as robotics has seen success [6]. The problem is that the process of training an RL model is often lengthy; requiring huge amounts of time and data before reaching a good performance level. The introduction of human feedback has been shown to enhance, and often quicken, the training process. This is ideal for agents learning in a larger, potentially complex, environment, as its training can be supplemented by human expert knowledge through a series of interactions. The success of IRL has been proven, but often within constrained environments [6], such as when there are a total of four possible decisions (or actions) to be taken by the agent at any one time.

1.3 Project Aims

This project aims to:

- Present the reader with an understanding of the different types of bias which may be present within a machine learning model.
- Establish the effect of human feedback on an agent's performance within a larger learning environment; directly addressing one of the current limitations in research in IRL.
- Provide foundational work with the notion that this will then be carried forward into future work.

1.4 Research Questions

This project will address the following research question:

RQ1: Does the addition of interactive reinforcement learning improve training and model performance?

1.5 Approach

In order to address the research questions outlined in the previous section, this project will:

- Discuss the effect of bias in ML models through a comprehensive literature review around the topic of bias and current work in the effort to mitigate the effects. Particular attention is paid to reward and preferential bias which can be a result of human feedback in IRL models.

- Design and implement a system in which a Reinforcement Learning model is trained to play the game of Battleship against a human opponent. This is done prior to any human interaction and is trained to play an optimal game (to sink all opponent ships in the least amount of moves). During a lab-based study, users are asked to play against this model, on a simple interactive GUI. Following this, the users are then asked to participate in an extra training process for the model. Whereby, they observe the model's moves and supply it with scalar value rewards; based on how they believe it performs at each step. Once the model has been trained by all those participating in the study, the users then play a series of complete games against the new model. The performance of the model prior to training with human rewards (referred to as the RL model) is then compared to the one which has trained with them (referred to as the IRL model). The results are then analysed in order to determine the effects of human feedback.

1.6 Outcomes

A total of 100 complete games were played against both models; which involved 20 users playing five games per model each. The IRL model received training from each user across a complete game (with no opponent) and received on average 95 rewards per training iteration. A brief comparison of model performance metrics show that the IRL model achieved the lowest average moves and the highest win-loss strategy. Meaning, on average, the model with human-inputted feedback, won more games in less moves. A more in depth analysis of these results can be found in Section 7 of this paper.

This study is intended to be used as the foundational work for the purpose of testing designs and interfaces with users; future work will then seek to analyse the effect of different models on the user experience. Including an investigation into how bias may effect the user experience.

1.7 Document Overview

The remainder of this document is divided into the following sections:

- **Section 2** Literature Review. Discussion and analysis of current research material in the area.
- **Section 3** Project Plan. Outlines the timeline of the project.
- **Section 5** Design. System description.
- **Section 6** Implementation. Description of implementation.
- **Section 7** Results. Discussion and analysis of results.
- **Section 8** Future Work. Presents potential direction of future work.

2 Literature Review

2.1 Reinforcement Learning

2.1.1 What is RL?

Reinforcement Learning is a machine learning technique in which an agent learns how to optimally complete a task within a given environment. Learning is achieved through a series of interactions between the agent and the environment. The agent performs actions (A), which cause transitions in the environment's state (S). Following each action, a reward (R) is given to the agent. The reward represents the quality of the interaction, if the agent performed a "good" action it will be rewarded positively, thus reinforcing this kind of action in similar states in the future. Put simply, the agent learns which actions result in the highest rewards. RL was first defined in research originating from psychology and operations research [9], the computational techniques used today are a product of that work.

The RL framework is displayed in Figure 1 which describes how the agent and environment interact through the medium of actions and rewards [10].

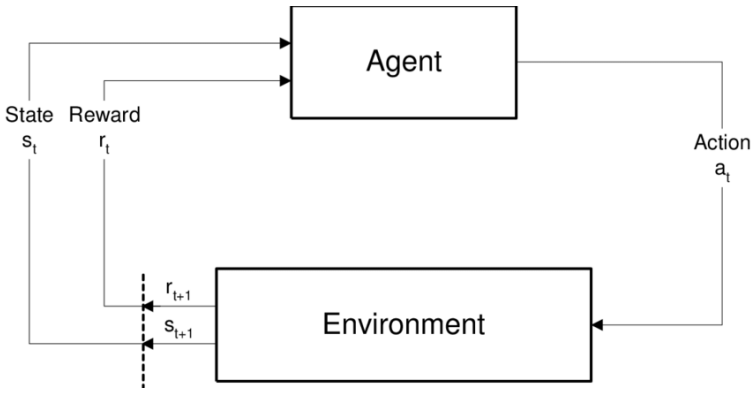


Figure 1: Reinforcement Learning Framework

2.1.2 Current Applications and Challenges

In recent years, reinforcement learning has achieved much success across a plethora of applications; from domains such as robotics [11–13] and gaming [14]. These applications are often limited to a small state-space. However, with advancements being made in Deep Reinforcement Learning (DRL), this machine learning method is beginning to show success within larger, more complex environments [14].

Mnih et al presented the first DRL model that successfully learned a policy to competitively play Atari, the model would go on to outperform human experts [14]. Lillicrap et al explored the application of DRL in continuous action spaces, showing their work successfully learned how to solve multiple physics based problems, sometimes only from pixel inputs [15]. These two papers were the first to see a significant breakthrough in DRL research, showcasing the versatility of these algorithms. However, there is one major setback that is restricting reinforcement learning from being employed in wider domains and that is the lack of safety-criticality. Safe reinforcement is a subsection of research in its own right, that is beginning to explore methods and techniques to ensure safety within systems [16].

Brunke et al outlines in depth the work that has been produced in this space, providing summaries of promising work in the attempt to achieve safe RL. One of the main road blocks is finding a way for the RL agent to safely explore its environment [17]. Exploration is a huge part of the learning process, choosing actions with unknown consequences is where the risk comes from, but also where the learning takes place. Current examples include restricting exploration to groups of policies that are guaranteed to be safe [18]; this work is confined to discrete state-action spaces. Proving again the challenge of RL in real-world applications. Other issues preventing the extension of RL uses is the black box issue of the models [19]. The black box phenomena leads to unpredictable behaviour, which is not ideal in a safety critical system.

2.1.3 Biases in RL

The previous section outlined current successes and challenges associated with implementing reinforcement learning methods in real-world applications. There are many more setbacks to address on the topic, however, this review aims to highlight the main ones, which leads ultimately to bias. Bias in reinforcement learning methods often negatively affect algorithm performance and as previously mentioned, prevents RL from being safety critical. The current work addressing this issue will be discussed.

Villaflor et al conducted a study which aimed to address the issue of optimism bias by creating methods that allowed, at test time, a search for policies that appear robust to multiple possible futures. Optimism bias is a product of an agent's incapacity to understand the effects of a policy and world dynamics due to the relationship it learns between actions and

rewards. This optimistic behaviour can be dangerous in safety critical environments [20]. Their approach was an offline RL method and results from a simulation of an autonomous vehicle proved superior to other techniques.

Primary bias is another significant challenge that arises during the training process. Nikishin et al describe this bias type as the agent's tendency to begin overfitting on early experiences [21], which effectively damages the rest of the learning process. Overfitting at an early stage is an issue that primarily arises when a model learns from a large dataset. Given that the advancement of reinforcement learning depends upon the ability to tackle increasingly larger problem sets, mitigating primary bias is going to be a necessity. The proposed solution in this paper is a simple one; to intermittently reset part of the agent during training. It is demonstrated through a number of experiments (and thus, domains) to be an effective method, showing improved performance. This technique does come with potentially unwanted effects though, such as a brief drop in performance after the reset. The technique itself is simple, but drops in performance are likely to result in learning inefficiencies. Reinforcement learning is already renowned to be extremely time-consuming, however, it is pointed out that a potential future for this method could lie with using RL feedback to optimise the resetting parameter.

2.2 Interactive Reinforcement Learning

2.2.1 What is IRL?

The main goal of IRL is to offer a method in which humans are able to be kept in the loop during the training process of machine learning agents. As previously mentioned, results have shown that the inclusion of humans during the learning stage offers a number of benefits, namely accelerated learning and increased efficiency. IRL follows a framework [22] which is depicted in Figure 2. Similarly to RL, IRL still works on the basis of interactions between an agent and an environment, only now there is a third member, the human. The figure visualises the interactions between each member as follows:

- The agent takes an action, which is then observed by the human and the environment.
- The environment updates the state and returns this to the agent and the human. The reader can think of this as the result or consequence of the action.

- The human then supplies feedback to the agent. Feedback is often a measure of how "good" or "bad" the result of the action was.

The above steps are repeated until the training run is complete.

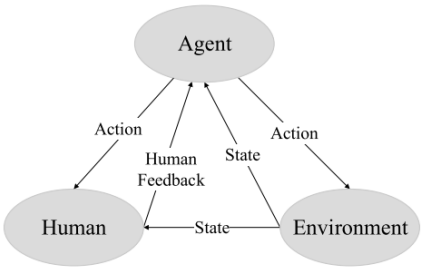


Figure 2: Interactive Reinforcement Learning Framework

A closer look into IRL architecture can be seen in Figure 3, here different reward methods are highlighted, in addition to human knowledge integration methods. Each method and type of feedback will have their uses across a number of different applications. The most commonly used feedback type in reinforcement learning is scalar-valued, this is due to the seamless transition from environmental rewards (which are also scalar values).

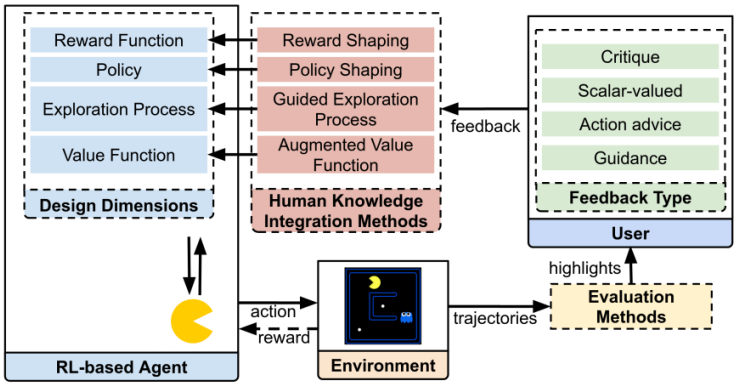


Figure 3: Human Feedback Methods

2.2.2 Current Applications and Challenges

The ability to integrate human knowledge into a machine learning agent is an exciting and useful prospect. Particular use cases would be those in which humans have an abundance of expertise and so in turn would be able to guide the agent's learning effectively. IRL has been utilised across many domains, similarly to RL, the main fields of research like within

robotics and autonomous vehicles [23, 24].

Koert et al explore the impact of multi-channel IRL; learning from both environmental and human feedback. The idea is that the human feedback will assist in accelerating the learning process of the agent (in this case a robot agent) in sequential tasks. The novelty of this study was the offering of multiple input channels for the users to provide feedback, across two experiments. The study also provided the robot with a self-confidence parameter, once this was high enough it would begin to question any human inputs that did not align with how it was about to act. Conclusions from this paper again call for more thorough work into how an agent might communicate with human teachers so that trust between the two can be established [23].

Zhang et al found similar results, showing that both environmental and human rewards improved performance beyond a single type of reward function [24]. It is important to note though that these results came from simulations only, future work in extending this into real world scenarios are required.

2.2.3 Biases in IRL

Biases that arise in human-in-the-loop systems can also impact an agent's overall performance, sometimes with unexpected consequences [25]. Whilst IRL has proven benefits, designing systems that are resilient to biases that stem from human interactions remains a real challenge.

The most prominent bias type that has been discovered in IRL is positive reward bias. The tendency for a human teacher to use positive feedback as a way to encourage the agent casts certain doubts on the usefulness of human feedback. Thomaz et al found that 69.8% of the rewards given in their user study were postive. Whilst they believe more work is needed to investigate this trend, they speculate that it is due to the participants treating the agent as a social entity that requires encouragement [26].

Chao et al demonstrate the importance of investigating the effects of human-in-the-loop training. Their study involved the transfer of human knowledge into a robot agent to carry out certain (simple) tasks. Even though the tasks were within small, defined spaces, a simple

bias demonstrated by the human teachers effected the behaviour of robot agent as a result. In one task the agent was taught to move objects from the left side of a table to the right side. Since the human teachers showed a bias to select the left most object each time, the agent learned that to complete this task successfully it was to choose the left most object [27]. This was likely an overlooked consequence and relatively harmless in this scenario, however, with the expansion of environments it is a wonder what long term effects a bias such as this could cause.

Krening et al explore the effect of interaction design on how humans experience IRL systems. Their conclusions show that interaction design has quite a substantial bearing on user experience, which in turn has an effect on the learning process. Since RL algorithms are mostly probabilistic, effects of feedback are not instantaneous, this led to 20% of their users feeling frustrated. It was found that an agent who seemed to respond more obviously to feedback was favoured and perceived as "more intelligent" [28]. This is an important finding since it appears there is room here for a human to develop a "likeness" bias for certain agents that react in a way they like (even though it may not be the most suitable model). Taking this further, it may also have some bearing on trust in the system.

To conclude, Cruz et al discuss the idea of safe interactive reinforcement learning, which may help to solve some of the issues associated with bias in RL systems as a whole. Ensuring that not only interactions, but outcomes of interactions, are safe for the human is paramount [1]. It is suggested that this a promising direction of research and potentially a solution lies here in the quest to employ RL in real-world, safety-critical settings.

2.3 Related Work

Sheidlower et al present CAIR (Continuous Action-space Interactive Reinforcement learning), the first IRL algorithm capable of outperforming state-of-the-art RL models. The CAIR algorithm is validated through two simulated robotics tasks, it utilises both environmental and human feedback to achieve the improved performance results [29]. Sheidlower's work is of particular interest as it explores similar techniques that are employed in this project, solidifying the relevance of this work. In contrast to CAIR, this project looks to question and explore what direct effects the human interactions cause within the agent's behaviour.

Kancko addressed the application of solving the game of Battleship with reinforcement

learning [30]. The paper employs and compares a number of different algorithms (including those that are not RL models) before concluding that the RL algorithm out-performed all other techniques. This paper was a good foundation to lay, proving the power of RL within a POMDP environment.

The following project, although inspired by numerous studies and other external motivations, utilises Sheidlower et al and Kanko's promising work as direct motivation.

3 Project Plan and Time Management

In order to manage the timeline and monitor progress of this research, the below Gantt Chart (Figure 4) was created and adhered to. As the project developed over time, a further literature review was required in order to keep up to date with the relevant research in this space. This was the only adjustment, leaving the remaining amount of the available time to complete the implementation, testing and writing stages. In addition to this, meetings were held once weekly with the project supervisor.

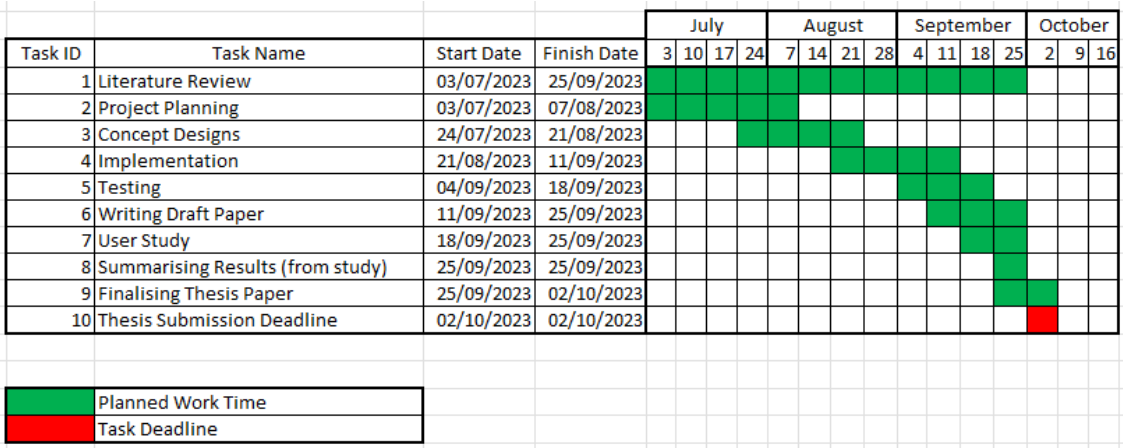


Figure 4: Gantt Chart

4 Design

4.1 Introduction to Battleship

Battleship is a two-player strategy game, in which players are tasked to destroy their opponents fleet. The game can be played across a number of mediums; from paper and pen, to mobile applications. The game board is made up of four grids of pre-defined size, each player has two grids each. Player's ships are distributed across the first grid (referred to as the *Ships Grid*), these grids must be hidden from the opponent. Enemy fire is also recorded on the first grid - this allows a player to keep track of how much (or little) of their fleet has been hit. The second grid (referred to as the *Search Grid*) is used to track shots fired at the enemy, allowing a player to track how well they are doing at hitting opponent ships.

Once both players are satisfied with the positioning of their fleet, game play begins. Each player takes it in turn to fire a shot at the opponent's grid, at one specific location. The outcome of the shot is then returned to the player; hit, miss or sunk. The game is deemed over once one of the players has successfully sunk their opponents entire fleet. There is always one winner at the end of a game.

There are a number of rules that must be followed before and during game play:

- Ships cannot overlap, or go beyond grid boundaries.
- Ships must be placed horizontally or vertically (not diagonally).
- Ships cannot be moved once game play begins.
- Ships must be placed so that there is at least one grid space between each other.
- Battleship is a turn-taking game. Specifically for this paper, no player should have consecutive turns, even when the current turn returns a positive (hit).

4.2 System Description

4.2.1 Environment: Board Setup and Game Play

In order to be consistent with the usual setup of Battleship, the grids used are two-dimensional matrices of size 10×10. As mentioned previously, players are assigned a search grid and a ship grid. The human player's view in the context of this project can be seen below in Figure

5. The left hand side grid is their search grid and to the right is the ships grid, where the player can see their ships and the opponent's moves.

In the usual setup, Battleship players have five ships each. The sizes range from one grid square to five, a player has one of each size. In this design, players have ten ships each and are denoted by green squares on the grid. The fleet consists of:

- four ships of size one,
- three ships of size two,
- two ships of size three and
- one ship of size four

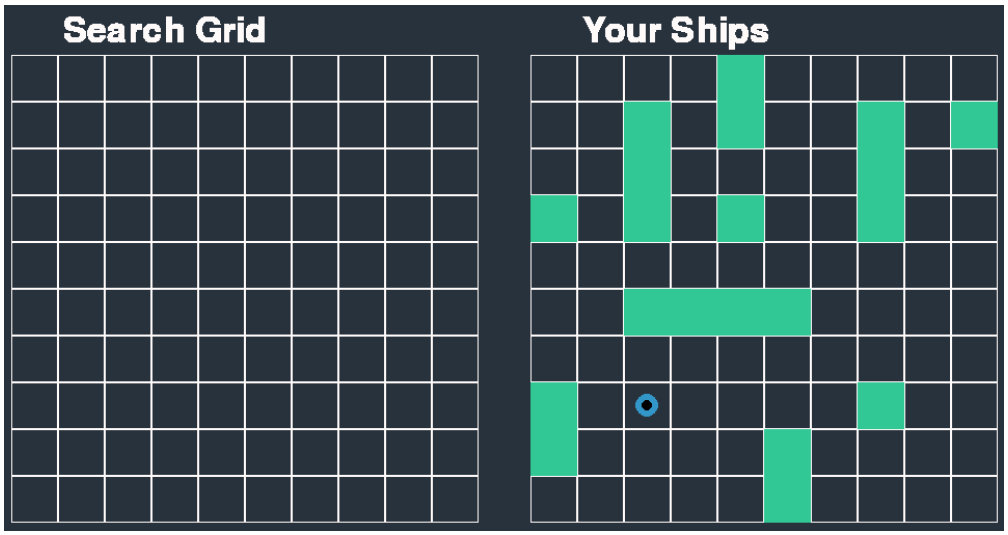


Figure 5: Battleship GUI

The board consists of three types of grid squares; ship squares, ocean squares and unknown squares. On the ship grid, the player knows that the green squares are their ships and the grey is the ocean. At the beginning of a game, before any shots are fired, the player does not know anything about the search grid - it consists of unknown squares only.

As game play progresses, the search grid will return information to the player. A successful shot (the player has hit an opponents ship) would return an orange circle at that location. A miss shot (the player's shot landed on an ocean square) would return a blue

circle at that location. Any remaining grey squares are those that have not been fired at yet and so do not reveal any information to the player. It is possible to win the game without searching every grid location, once all ships are sunk, it is accepted that any remaining grey squares are ocean squares.

4.2.2 Actions and States

Since the grid size used in this system is 10×10, the number of possible moves is 100. However, upon making a move, the result can return one of two pieces of information; whether the shot was a hit or a miss. Therefore, the number of possible actions is 200. The number of possible actions reduces by two every time a move is made.

The number of states that are possible is finite, due to the defined (and fixed) grid size. Prior to making any moves, the probability of any and all ship placements are possible. These only begin to change (become more or less likely) upon taking shots (performing actions). As each shot, or action, taken reveals more information about the grid, some ship configurations can also become impossible.

4.2.3 Rewards

In both the RL and IRL models, rewards are given per action performed. Rewards are simple scalar values.

In regards to the RL model, the reward values are as follows:

-1.0 ; given in the event of an invalid action. An example of an invalid action would be a repeated move (taking a shot at the same grid location more than once). This is not needed in a game of Battleship, since everything a player needs to know about a position is revealed within one shot.

-0.1 ; given if the shot is a miss.

0.0 ; given in the event of game over and the result is a loss.

1.0 ; given if the shot is a hit.

The reward values used in the training process for the IRL model are still scalar values, as previously mentioned, however users are encouraged to use the entire range of numbers

(to 1.d.p.) between -1.0 and +1.0. This allows the human to provide a more customised set of rewards. For example, a near miss (a shot fired one square away from an opponent ship) may be rewarded slightly higher than a miss which was not this close.

4.2.4 Markov Decision Process

In Reinforcement Learning, problems are often formalised as a Markov Decision Process. An MDP is a finite, stochastic model of a system made up States, Actions, Transition States and Rewards. They can be defined as a tuple:

$$< S, A, T, R >$$

Where:

- S is a set of all possible states (describes the environment).
- A is a set of all possible actions with which an agent can take.
- $T(s, a)$ is a stochastic transition function. Maps probability of arriving at new state, s_{t+1}, from current state s_t due to taking action a
- $R(s, a)$ is a reward function. An immediate reward, representing an evaluation of the quality of the selected action.

The above model is only accurate for a system in which the outcome of an action is known before taking it. Considering this, it is found that Battleship cannot be formalised as an MDP. This is due to the fact that, until a result is returned after an action is performed, there is no information about the new state. More explicitly, before taking a shot at the opponent's fleet, it is unknown whether it will be a hit or a miss. However, it is possible to instead formalise Battleship as a Partially Observable MDP (POMDP).

4.2.5 Partially Observable Markov Decision Process

Partially Observable Markov Decision Processes (POMDPs) can be used to model a system where the outcome of an action is non-deterministic [31]. Figure 6 represents how an agent interacts with its environment in the context of a POMDP. In order to correctly formalise the game of Battleship as a POMDP, the introduction of Observations and an Observation function is required. This results in the 6-tuple:

$$< S, A, O, T, Z, R >$$

Where:

- O is a set of all observations with which an agent may perceive.
- $Z(o, s)$ is the observation function. Maps the probability of perceiving observation o, in state s.

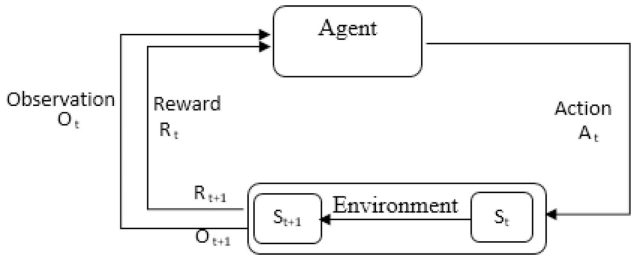


Figure 6: Partially Observable Markov Decision Process

5 Implementation

5.1 Reinforcement Learning

5.1.1 Environment

OpenAI Gym is an open source toolkit which offers a standardised framework for developing and testing RL agents [32]. It is designed with researchers and enthusiasts in mind with many pre-developed environments to test agents within, saving time on designing the specifics of an environment. Having said this, the Gym framework is also designed to be extensible; allowing the creation of custom environments. The gym framework primarily runs on two methods:

- `reset()`; this method resets the environment, ready for a new episode.
- `step()`; this method passes the action that the agent selected through to the environment. Normally returning a reward, an updated state and other information important to the environment.

Using this framework [33], a two-player Battleship environment was built. This meant that the `step()` method was called every time any player made a move, regardless of whether it was made by an RL agent or a human player. Of course, the human player does not require rewards, so based on what type of payer was making the move, the values returned from this method were different. The `reset()` not only resets the Battleship environment, but also randomly places ships for both the human and agent player. Placing ships is normally a manual task in Battleship, players would complete this prior to game play. However, for the purpose of training the RL agent, this was automated and each time is completed randomly (keeping it fair for both the agent and human players).

Two main classes run the majority of the Battleship environment:

- `Model model`; defines and trains the RL model
- `Game game`; initiates game play and utilises the modified `reset()` and `step()` methods.

Important parameters to note are:

- `rl_ships`; keeps track of the status of the ships that the RL agent is firing at (whether they have been hit or sunk).

- `rl_state`; keeps track of the current state of the game board, gets updated after every step the RL agent takes. Helps the RL agent choose next steps based on probabilities that are calculated due to the current state.
- `terminal`; checks whether the criteria for game over is met after every step taken by either a human or RL agent player.

5.1.2 Algorithm

In regards to finding a reinforcement learning algorithm that would be suitable for this context, there are many available. As previously mentioned, Battleship is a highly stochastic, and non-deterministic, problem set, therefore an algorithm that utilises probabilities is required. Policy Gradient (PG) methods are a type of reinforcement learning technique which seek to optimise the policy directly, that is, by maximising the expected outcome (reward). There is currently an abundance of policy gradient algorithms available to use, in this work a simple Tensorflow reward-to-go method was selected for use, given that OpenAI offers well documented open source algorithms [34]. A reward-to-go policy makes sure to only reinforce when the consequences of an agent's action is known, else the reward has no bearing on whether the action was "good" or not. The initial expression for a policy gradient is as follows:

$$\nabla_\theta J(\pi_\theta) = E[\sum_{t=0}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) R(\tau)]$$

Updating the expression to the following makes sure that rewards are only ever given after actions (and their consequences) are known. This is how the policy gradient agent is trained to play Battleship:

$$\nabla_\theta J(\pi_\theta) = E[\sum_{t=o}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1})]$$

The process of setting up this policy gradient algorithm is as follows:

1. Initialise the policy network; in this case a feed-forward neural network.
2. Generate a trajectory using one sample.

3. Estimate the return.

4. Update the policy parameters.

5. Use optimisation technique and repeat steps 2-4 until optimal policy is achieved.

The goal of this algorithm is to calculate the probabilities (which are updated per step) and, using gradient descent, optimise the weights in the neural network to achieve an optimal policy gradient algorithm.

5.2 Interactive Reinforcement Learning

Once the RL model reaches optimal performance, it is then used as "base knowledge" for the IRL model. Some IRL models may train on human feedback only, however, this project is to investigate the outcome of a model that trains with both. Twenty users were involved in an extra layer of training, in which they were asked to observe the agent's steps and return scalar value rewards based on how they thought the agent was performing. The users were asked to supply rewards between the values of -1.0 and +1.0 to one decimal point. These values were chosen in order to keep consistency with the lower and upper boundaries of the environmental rewards given to the RL agent. The study was to encourage maximum customisabilty and since the reward system was setup to receive scalar values only this was already very limited. Due to this, the users were not given any extra details about how the environmental rewards were calculated (i.e. +1.0 for a hit), so as to not influence them in copying that system. They are told, however, that positive value rewards should be used for "good" actions and negative value rewards should be used for "bad" actions.

5.3 GUI

The GUI is a simple design using the `pygame` library. It features two grids as previously described in Section 4.2.1. The left grid is used to fire shots at the agent's fleet. The right grid displays the player's fleet and enemy (agent) shots. The interface invites users to make moves using mouse clicks during game play. User's receive encouraging feedback throughout, an example of this can be seen in Figure 7. Also demonstrated here are each of the possible markers used on the game board:

- Blue represents a missed shot, i.e. the shot landed on an ocean square.

- Orange represents a hit.

- Red represents a sunken ship.

It should also be noted that the most recent shot fired by the agent is marked with a black spot; this is to assist the human player in keeping track of their opponents shots.

During training, the display alters to include a textbox field in which users can input rewards using keyboard inputs, an example of this can be seen in Figure 8a. In order to submit a reward, users simply have to press the enter key. Numerical values only are accepted at this stage of development. An example of what a user sees during the lab is shown in 8b.

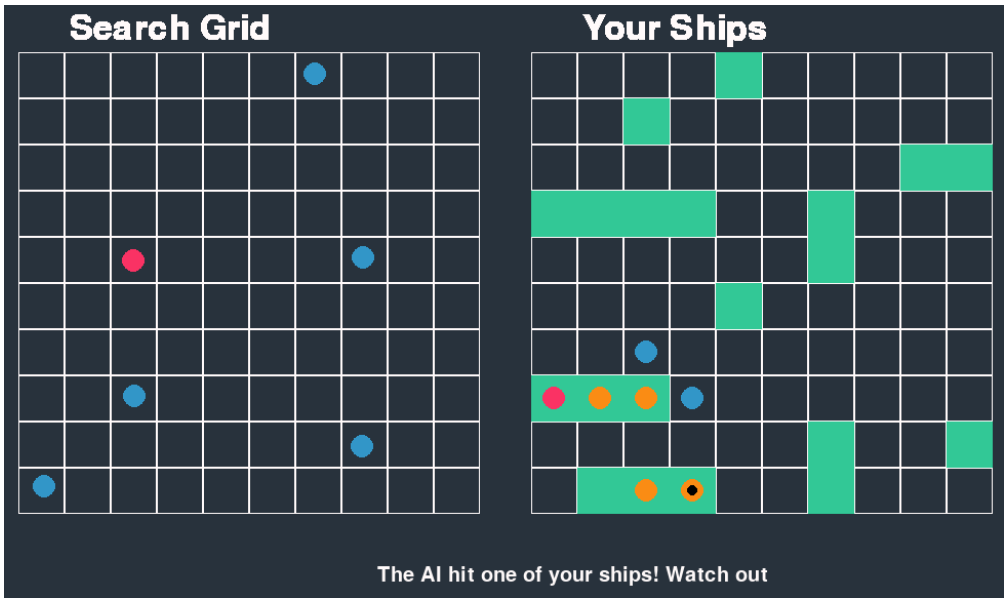


Figure 7: GUI: In-Game Feedback

5.4 Game Play

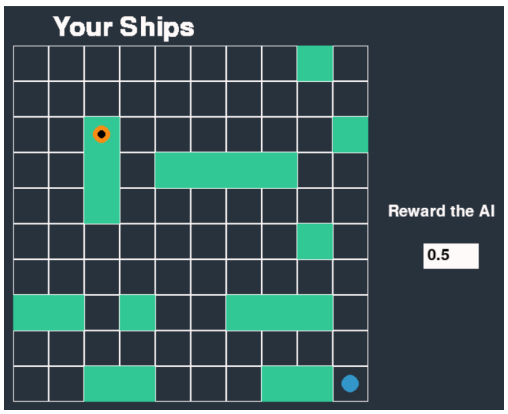
Upon installing all correct dependencies, instructions on how to initiate game play are as follows:

5.4.1 Play Against RL model

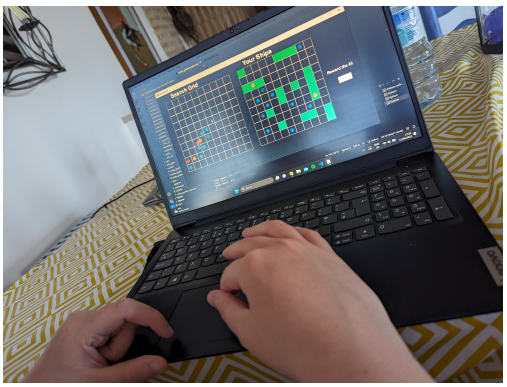
On line 624 within policy_gradient_agent.py, set:

```
default_model_name = "policy_gradient-3299008"
```

Once the default model is set, enter the following in a terminal:



(a) Example Training Run



(b) Example User Training IRL Agent

Figure 8: Training Examples

```
python policy_gradient_agent.py
```

5.4.2 Play Against IRL model

```
default_model_name = "policy_gradient-2"
```

Once the default model is set, enter the following in a terminal:

```
python policy_gradient_agent.py
```

5.4.3 Begin Interactive Training

Enter the following in a terminal:

```
python policy_gradient_agent.py --train
```

6 Results & Discussion

6.1 Results

Both the RL and IRL agent played a total of 100 times against 20 user study participants; five times each per user, per model. Upon completion, the results were recorded and can be seen in the following visualisations. Figure 9 shows the total number of steps taken per game.

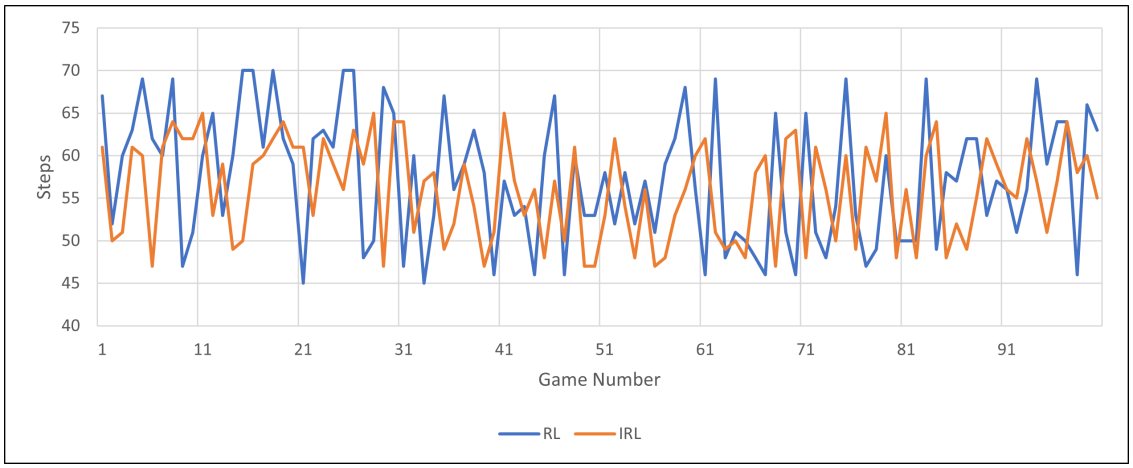


Figure 9: Comparison of Number of Steps per Game

A comparison of the first twenty games is shown in Figure 10 to show the results more clearly. It is easier to see here that the mean number of steps is across these twenty games are lower for the IRL model.

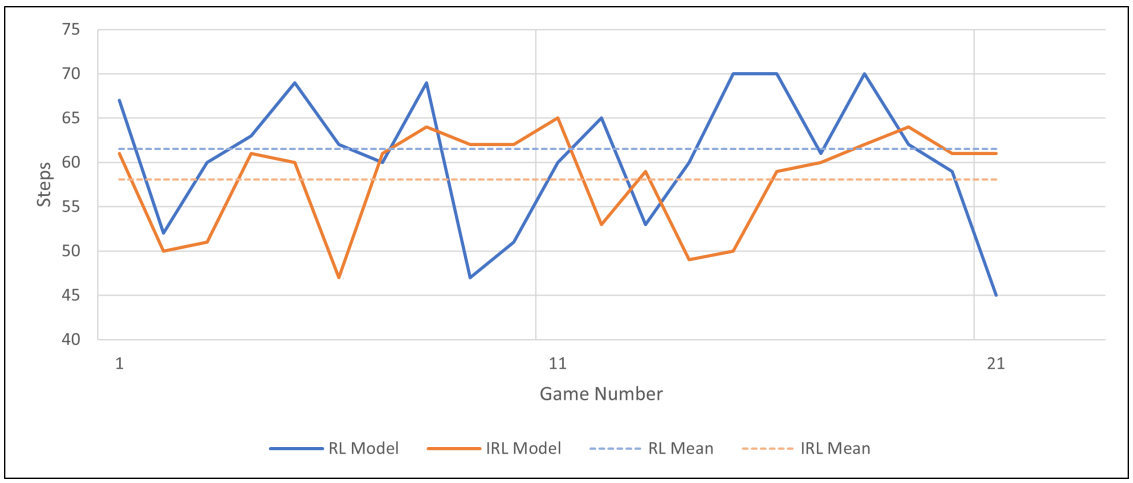


Figure 10: Comparison of First 20 Games

Figure 11 shows the results from each game played by the IRL model and indicates which games the agent won or lost. There is no real trend between less total moves and the agent winning; the results are fairly scattered. This will be discussed further in Section 6.2.

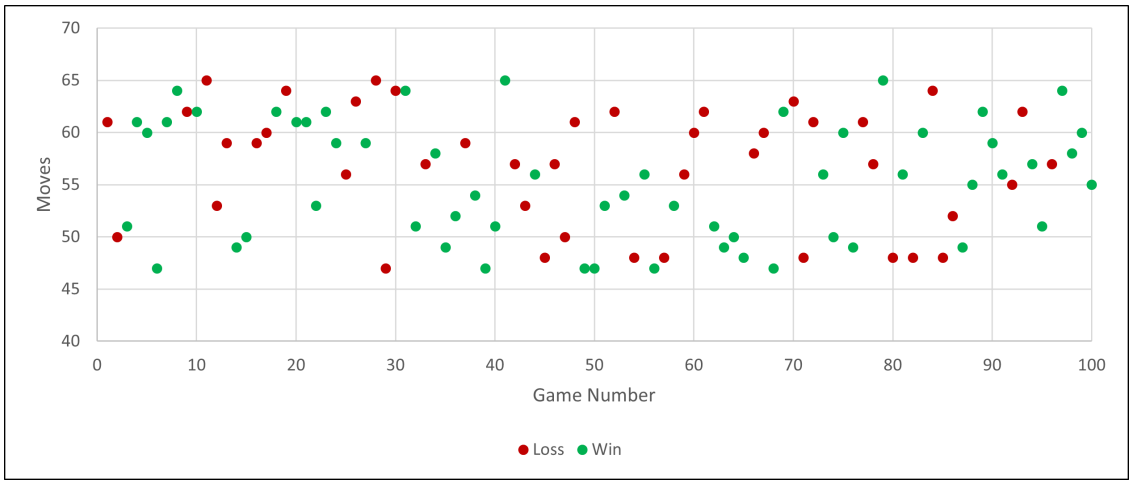


Figure 11: Win-Loss Comparison: IRL Model

A number of important model performance parameters are recorded in Table 1, offering easy comparison between the two models.

Model	Min Steps	Max Steps	Mean Steps	Win-Lose Ratio
RL	46.0	70.0	59.1	1.1
IRL	47.0	65.0	56.3	1.3

Table 1: Comparison of Model Performance

Prior to IRL training, Figure 12 shows an example of the agent making an unnecessary move. The agent has taken a shot at a location directly next to a previously sunken ship; this is a wasted move as following the rules outlined in Section 4.1, ships are not placed directly next to each other; there is always a minimum of one grid square between each ship.

Figure 13 provides a visualisation of the rewards given to the RL and IRL agents over the extent of one training epoch. The human-provided rewards were significantly more positive in this example, with a larger variety of values compared with that of the environmental rewards. A summary of this data is provided in Table 2. 76.86% of the RL agent's rewards

were below zero, in comparison only 15.05% of the IRL agent's rewards were below this value. Showing that the human teacher, in this example, positively reinforced the agent. This is often the case with human-in-the-loop training.

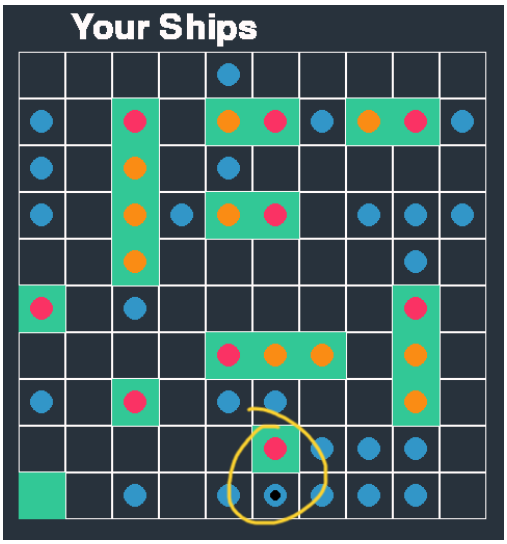


Figure 12: Example of RL Agent Strategy

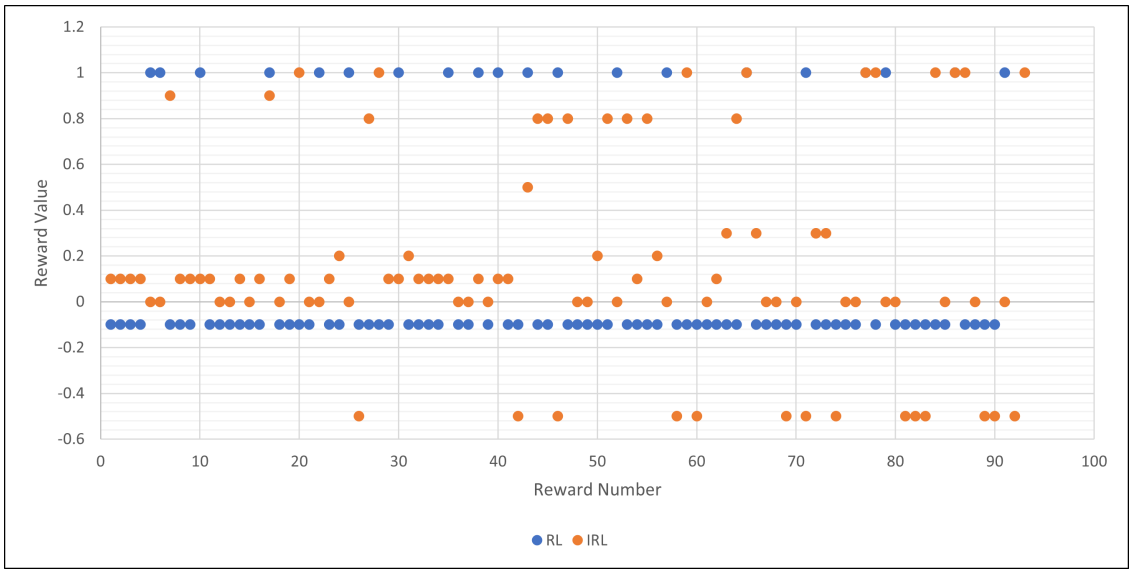


Figure 13: Comparison of Rewards (one epoch)

Model	Mean Reward	% Below 0
RL	0.14	76.86
IRL	0.17	15.05

Table 2: Comparison of Rewards Over One Epoch

6.2 Analysis & Discussion

From the information displayed in Table 1, it is clear that the IRL slightly outperforms the RL model in all but one metric, minimum steps. The IRL model won 18% more games overall and achieved an average 4% lower than the RL agent.

Since the IRL agent was only trained by 20 participants, as opposed to the millions of training runs that the RL agent completed, it is not so simple to deduce from these results that the IRL agent performs better due to the interactive training.

There are first a number of environmental factors to consider in this case. Firstly, participants played 5 games each against the RL agent before completing interactive training. There is likely chance that the users had an impaired attention span by the time they played any games against the IRL model. This may have lead to more wins for the IRL agent, not because of an increase in model performance but due to a decrease in performance on behalf of the human players.

Additionally, there is no real trend between less total moves in a game and the end result (winner of the game), shown in Figure 11. For example, the IRL agent won a game in 51 moves but lost a game in 47. This scattered distribution of wins and losses will have been in part, a result of the human player's ability. Since twenty participants were used, it is fair to assume that some will have been more skilled than others, causing the skew of results. It is also important to remember that Battleship is in part, a guessing game, therefore some of the wins and losses will have been down to a few good or bad guesses on behalf of both players.

Having said this, there is evidence that the IRL training did somewhat improve the agent's strategy. As previously mentioned, there were multiple occasions where the RL agent searched in locations directly next to sunken ships, an example of which is shown in Figure 12. These shots are unnecessary due to the rules of the game. The IRL agent did not make any moves like this across the 100 games it played, suggesting the human feedback helped to develop the state-action space. The participants were made aware of the game

rules and likely began to give the agent negative value rewards for actions like this.

A final, interesting point to consider is the pronounced difference in reward values provided by the environment compared to those from human participants. Whilst the RL algorithm trained on mostly negative feedback, the IRL agent trained on mostly positive. This is somewhat expected, as studies have found that humans tend to being teacher agents similar to how they would teach humans, with positive reinforcement. It is also reassuring to see that in the example training epoch, displayed in Figure 13 this particular participant used a nice range of numbers between the defined upper and lower boundaries.

7 Limitations

The implementation and user study that has been presented in this thesis were subject to a number of limitations, namely, a short time period in which to carry out development and testing.

The RL model was able to train extensively in comparison to the IRL model, this is solely due to the short timescale that was on offer to complete the project. Since the IRL model trained on real-time human feedback, it was time consuming and therefore only completed twenty full training runs (in comparison to the millions that the RL model was able to run through). This means it is quite difficult to conclude whether the IRL training had much of an affect on the overall model behaviour.

In addition, the IRL model was trained using a limited number of users. Each of whom were asked to play multiple games against both agents in one sitting. This meant that each user played a total of 10 games of Battleship and observed an additional game played solely by the IRL agent during the training phase. It is fair to assume that some of the users may have become disinterested in the process, or their attention may have changed throughout the study (this is just what happens when a human is asked to complete a repetitive task, their attention span can become limited). Ultimately, this would have somewhat affected the results that have been presented here. With a more generous timescale, the user study could have benefited from a larger number of participants. Suay et al discuss ways in which to combat the short attention span of a human completing repetitive tasks. The most interesting approach being to allow the agent to essentially rush through parts of the task that it knows well, until it reaches unknown parts, where the state-action space is not well defined [6]. This approach, however interesting, would not be applicable to the game of Battlehip, but it is something to keep in mind during future user studies.

Finally, the GUI was built using the `pygame` library and whilst the interface displays everything correctly and simply to the user, it is very limited. Since this was a small-scale project with certain time limitations, the GUI was acceptable. However, the limitations that comes with this library certainly restricted the amount of user engagement features that could be added to the interface. For example, simply displaying messages began to effect the smooth-running of game play - too many messages caused time delays between making a move and printing the shot on the grid. Additionally, in order to hold a successfully

larger user study, it would be wise to develop on a platform that is easy to distribute. This study was limited to the laptop that had the source code on it to execute the program.

8 Conclusion

8.1 Conclusion

The main objective of this thesis was to explore and investigate the impact of introducing human-in-the-loop training on model performance. A small user study was undertaken in order to facilitate this work. Twenty participants were able to interact with two different models, one of which they collectively trained over the course of the study which resulted in the agent receiving approximately 2000 human-provided rewards. Analysis of the results show that the IRL agent outperformed the RL agent in all but one of the measured metrics. It is believed that due to the participants having a wider range of reward values, the IRL training was able to expand the state-action space, which in effect improved the strategy of the agent.

The positive results also showed that IRL is effective in larger environments and may therefore its uses into real-world, complex environments. It is important to note however, that this was a small scale study which was constrained by time and consequently there may have been other environmental factors at play in the presented results. The main concern being that the participants were completing many repetitive tasks, which humans are known for not being good at without having a reduced attention span.

In addition to this, the topic of bias, particularly that which originates from humans, was examined. IRL models in particular are vulnerable to these bias types, as they are trained solely on human feedback. Examples of this can be reward bias, humans have a tendency to positively reward agents in an attempt to encourage (reinforce) good behaviour. This tendency was evident in the user study. Whilst the rewards supplied by the participants did have a positive trend, there was insufficient examples to deduce whether this had a potential impact on the model's overall performance.

8.2 Future Work

The results observed from the user study highlight the advantages of employing a human-in-the-loop approach in the training of ML agents. Furthermore, this project was designed with the intentions of being used as foundational work for future projects working with RACE / UK Atomic Energy Authority (UKAEA). The focus of this future work will be

directed towards aiding their primary domain; robotics and remote applications in extreme settings, notably fusion energy. The particular focus will be to explore the enhancement of the current tele-operated robotics system, known as MASCOT, that is used to carry out maintenance tasks inside fusion reactors. Currently, the maintenance tasks are carried out solely by highly trained human operators who likely experience a consistently high cognitive load throughout the work day. Future endeavours will involve the exploration of automating some of these tasks, so that the human operators may benefit from the use of an agent's assistance. The intention is to collaborate with their industry experts and robotic operators in order to achieve designs and implementations that prioritise a human-centric focus.

References

[1] C. Arzate Cruz and T. Igarashi, "A survey on interactive reinforcement learning: Design principles and open challenges," in *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, (New York, NY, USA), p. 1195–1209, Association for Computing Machinery, 2020.

[2] A. L. Thomaz, G. Hoffman, and C. Breazeal, "Reinforcement learning with human teachers: Understanding how people want to teach robots," in *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 352–357, 2006.

[3] P. Barros, A. Tanevska, and A. Sciutti, "Learning from learners: Adapting reinforcement learning agents to be competitive in a card game," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2716–2723, 2021.

[4] S. Krening and K. M. Feigh, "Effect of interaction design on the human experience with interactive reinforcement learning," in *Proceedings of the 2019 on Designing Interactive Systems Conference*, DIS '19, (New York, NY, USA), p. 1089–1100, Association for Computing Machinery, 2019.

[5] N. Akalin and A. Loutfi, "Reinforcement learning approaches in social robotics," *Sensors*, vol. 21, no. 4, 2021.

[6] H. B. Suay and S. Chernova, "Effect of human guidance and state space size on interactive reinforcement learning," in *2011 RO-MAN*, pp. 1–6, 2011.

[7] N. Boukhelifa, A. Bezerianos, and E. Lutton, *Evaluation of Interactive Machine Learning Systems*, pp. 341–360. Cham: Springer International Publishing, 2018.

[8] A. Anderson, J. Dodge, A. Sadarangani, Z. Juozapaitis, E. Newman, J. Irvine, S. Chattopadhyay, M. Olson, A. Fern, and M. Burnett, "Mental models of mere mortals with explanations of reinforcement learning," *ACM Trans. Interact. Intell. Syst.*, vol. 10, may 2020.

[9] P. Dayan and Y. Niv, "Reinforcement learning: The good, the bad and the ugly," *Current Opinion in Neurobiology*, vol. 18, no. 2, pp. 185–196, 2008. Cognitive neuroscience.

[10] A. Eriksson, G. Capi, and K. Doya, "Evolution of meta-parameters in reinforcement learning algorithm," pp. 412 – 417 vol.1, 11 2003.

[11] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," 2019.

[12] A. S. Polydoros, "Survey of model-based reinforcement learning: Applications on robotics," 2017.

[13] P. Kormushev, S. Calinon, and D. G. Caldwell, "Reinforcement learning in robotics: Applications and real-world challenges," *Robotics*, vol. 2, no. 3, pp. 122–148, 2013.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[16] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll, "A review of safe reinforcement learning: Methods, theory and applications," 2023.

[17] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," 2021.

[18] T. M. Moldovan and P. Abbeel, "Safe exploration in markov decision processes," 2012.

[19] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers Chemical Engineering*, vol. 139, p. 106886, 2020.

[20] A. R. Villaflor, Z. Huang, S. Pande, J. M. Dolan, and J. Schneider, "Addressing optimism bias in sequence modeling for reinforcement learning," in *Proceedings of the 39th International Conference on Machine Learning* (K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, eds.), vol. 162 of *Proceedings of Machine Learning Research*, pp. 22270–22283, PMLR, 17–23 Jul 2022.

[21] E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. Courville, "The primacy bias in deep reinforcement learning," in *Proceedings of the 39th International Conference on Machine Learning* (K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and

S. Sabato, eds.), vol. 162 of *Proceedings of Machine Learning Research*, pp. 16828–16847, PMLR, 17–23 Jul 2022.

[22] J. Lin, Z. Ma, R. Gomez, K. Nakamura, B. He, and G. Li, "A review on interactive reinforcement learning from human social feedback," *IEEE Access*, vol. 8, pp. 120757–120765, 2020.

[23] D. Koert, M. Kircher, V. Salikutluk, C. D'Eramo, and J. Peters, "Multi-channel interactive reinforcement learning for sequential tasks," *Frontiers in Robotics and AI*, vol. 7, 2020.

[24] Q. Zhang, J. Lin, Q. Sha, B. He, and G. Li, "Deep interactive reinforcement learning for path following of autonomous underwater vehicle," *IEEE Access*, vol. 8, pp. 24258–24268, 2020.

[25] C. Chao, M. Cakmak, and A. L. Thomaz, "Towards grounding concepts for transfer in goal learning from demonstration," in *2011 IEEE International Conference on Development and Learning (ICDL)*, vol. 2, pp. 1–6, 2011.

[26] A. L. Thomaz, G. Hoffman, and C. Breazeal, "Reinforcement learning with human teachers: Understanding how people want to teach robots," in *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 352–357, 2006.

[27] C. Chao, M. Cakmak, and A. L. Thomaz, "Towards grounding concepts for transfer in goal learning from demonstration," in *2011 IEEE International Conference on Development and Learning (ICDL)*, vol. 2, pp. 1–6, 2011.

[28] S. Krening and K. M. Feigh, "Effect of interaction design on the human experience with interactive reinforcement learning," in *Proceedings of the 2019 on Designing Interactive Systems Conference*, DIS '19, (New York, NY, USA), p. 1089–1100, Association for Computing Machinery, 2019.

[29] I. Sheidlower, A. Moore, and E. Short, "Keeping humans in the loop: Teaching via feedback in continuous action space environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 863–870, 2022.

[30] T. Kancko, "Reinforcement learning for the game of battleship," *Masaryk University, Faculty of Informatics*, 2020.

[31] H. Kurniawati, "Partially observable markov decision processes and robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 253–277, 2022.

[32] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[33] V. Anklin, "Playing battleships with deep reinforcement learning," 2019.

[34] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.