Teaching AI to Cooperate with Human Partners through Rule-based Play

Andrew Bean

Blackfriars Hall, University of Oxford

Submitted to the Oxford Internet Institute in partial fulfilment of the requirements for the degree of

Master of Science in Social Data Science

Trinity Term 2022

Abstract

With the ongoing expansion of artificial intelligence into tasks which interact directly with people, (driving cars, processing images, translation, etc.), it is important to develop agents which not only work near people, but cooperate with them. In this thesis, I begin by drawing on existing literature to build a framework for types of human-AI cooperation along axes of core competencies and levels of social interaction. Within this framework, the remaining work focuses on the particular role of understanding in helping independent agents to work together in multi-agent environments. Most reinforcement learning algorithms do not have a mechanism to teach an agent to cooperate with agents who are dissimilar to itself, especially not humans. To cooperate, agents benefit from understanding how people behave, which requires experience working with human-like partners. This thesis proposes that such experience could be generated using rule-based agents designed to imitate humans. The imitators are used as training partners within the reinforcement learning environment to introduce human-like behaviour and increase the likelihood of familiarity and successful cooperation with human partners. To test this, I create a novel environment in which pairs of agents work together to score points in a card game, using separate information and resources to promote reliance on cooperation. Using this environment, I implement and train four models using three existing training methods and the new approach. To compare the agents, I conducted an online experiment with human partners, which showed promising results for the potential usefulness of rule-based partners. The average performance of the methods surpasses the benchmarks, and although the sample size is too small to conclusively compare between models, the average performance of the new method exceeds the others. These results recommend the continued study of rule-based training partners for their potential to improve human compatibility in AI agents.

Contents

List of Figures

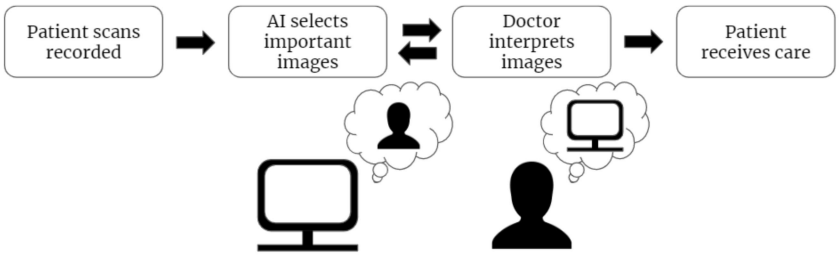
List of Tables

1

Introduction

The use of artificial intelligence (AI) is rapidly proliferating across disciplines, as computers are learning to solve new problems ranging from driving cars to supporting medical diagnosis and language translation (Bojarski et al. 2016; Steiner et al. 2018; Ghoshal et al. 2021; Y. Wu et al. 2016). With this expansion, there is also a growing emphasis on the importance of AI as a tool for human enhancement and partnership rather than as a replacement (Diamond 2020; Wilder et al. 2020). In many fields, including those such as chess where AI are thought to be superior, results from human-AI pairings continue to exceed the standard set by AI operating independently (Baraniuk 2015; Steiner et al. 2018). Collaboration between human and AI is therefore an important area for study, which has spawned new research in fields such as human-computer interaction as well as in the proposed creation of a new sub-field called 'Cooperative AI' (Liang et al. 2019; Traeger et al. 2020; Dafoe

Figure 1.1: Cooperative AI in Medical Diagnosis. The doctor and AI can cooperate to achieve better results together. First, patient scans are taken. Then the AI uses an understanding of the task at hand to select images with tailoring to the doctor's own expertise. Next, the doctor interprets the selected scans, keeping in mind the strengths and weaknesses of the AI to know how to use it best. This process is repeated until the doctor is satisfied with the result, and then informs the patient of the diagnosis. Each agent relies on comparative advantages (scalable image processing vs medical understanding) and benefits from the strengths of the other.

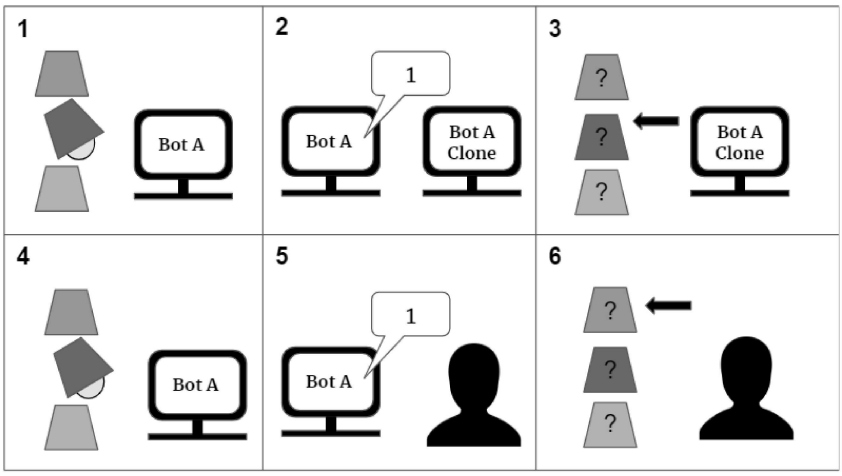


et al. 2020; Onnasch and Roesler 2021). This thesis considers a particular problem in human-AI partnership, the need for mutual understanding in cooperation, and the challenge of training an AI to 'understand' human partners.

Looking first at a human example provides a motivation for the need for mutual understanding in cooperation. Suppose that two people are going to eat lunch together, and each is bringing food to share. They must first understand their context; in the middle of summer, bringing ice cream is a better idea than hot chocolate. They must also understand each other; if one is an excellent cook and the other a baker, they would mutually prefer to bring the things they are best at preparing. Transferring these ideas to human-AI cooperation, a doctor might be using an AI to accelerate the reading of diagnostic tests. In this case, an agent with a better understanding of the doctor can provide more helpful answers. Perhaps the doctor is a specialist in one particular disease, and only needs to see a few annotated results to understand the situation, but would need more detail when the results are more indicative of something else. Or, the doctor is more proficient than the AI in certain scenarios, and the best outcomes could result from highlighting uncertainty and deferring to the doctor. In each case, the doctor stands to benefit if the AI can effectively leverage understanding to cooperate better.

AI agents with this sort of understanding could be developed by manually specifying a list of cooperative opportunities and teaching suitable behaviors for each one, but it would clearly be preferable to develop techniques for learning cooperation more generally. Since the goal is an emergent behaviour for an agent acting in the world, reinforcement learning is the natural approach to consider. However, the standard techniques in reinforcement learning which have been highly successful in independent tasks are not necessarily well-suited to multi-agent cooperative scenarios (Carroll et al. 2019; Strouse et al. 2021). In particular, self-play, the technique of repeatedly attempting a task in cooperation with copies of the same agent, can lead to insular behaviours which do not adapt well to cooperation with humans, or even other AIs (Hu et al. 2021). In broad strokes, agents trained with self-play can lack a sufficient understanding of what to expect from novel partners. Figure 1.2 shows an example of this where the bot has learned a communication system, but it only works with identical partners. Two main approaches have been suggested to teach understanding through the training process: introducing humans into the training environment, and broadening the diversity of AI training partners to better anticipate what humans might do (Carroll et al. 2019; Strouse et al. 2021). This thesis looks to human self-explanation as rule-based actors to offer a third approach which approximates human involvement in the training process using rule-based agents.

Figure 1.2: Communication Failure Example. An AI is playing a communication game where one partner knows where the ball is hidden, and must tell the other. *Top:* The bot learns where the ball is hidden (1). It then tells an identical clone by sending the message '1' (2). Since the bot is identical, it knows that '1' means the middle cup and guesses correctly (3). *Bottom:* The bot learns where the ball is hidden (4). It tells a human partner using the learned message '1' (5). The human does not understand and guesses the wrong cup (6). If the bot had learned a more generalizable approach, such as using the colors, it could have worked with the human. This is a toy example, but reflects the problem which arises in self-play.



This thesis offers four main contributions:

- A theoretical discussion of cooperative AI, including a categorization of the levels of social structure at which AI agents could operate, in order to situate different forms of cooperation, and particularly partnership
- Motivation for a new approach to teaching AI agents to understand human behaviours by using rule-based partners
- A new training environment for cooperative multi-agent reinforcement learning under incomplete information, as well as an online testing platform for human experiments
- Initial testing of the new training methodology alongside previously successful methods

2

Methods

The methods are divided into two major sections, the theoretical framework (2.1), and the practical methods (2.2). The theoretical section considers the opportunities for cooperative AI and locates this thesis in spectrum of social organizational levels. Following on, I use a framework drawn from Dafoe et al. 2020 to highlight the types of cooperative behaviours which might be learned. Within this framework, the area of 'Understanding' emerges as a focus, and motivates the use of a rule-based training partner to teach AI agents to understand human behaviours.

The practical methods describes the approach taken to testing the new training approach. It begins with a description of the new environment, followed by reinforcement learning techniques. As a baseline, three methods used in other

literature are discussed before moving to the implementation of all four deep learning agents.

2.1 Theoretical Framework

Cooperative opportunities for AI arise in a broad range of circumstances, reflecting the diversity of social scenarios in which cooperation takes place. Previous work by Dafoe et al. helpfully identifies major dimensions of difference which categorize opportunities: alignment of incentives, nature of agents (human, AI, or combination), and perspective (cooperative individual or social planner) (Dafoe et al. 2020). Along the 'perspective' dimension, a greater level of detail may be found by focusing on the level of sociality at which an AI agent would operate. This draws out the conceptual framework underlying the identified dimension to facilitate the development of theory and reasoning about differences along this axis. In particular, I highlight that more specificity is possible in the different ways an individual AI might be expected to act. By grouping research in this way, it is easier to identify both how existing developments tie together and where further opportunities exist.

Visions for cooperative AI agents range in scale, from assistants, or akin to pets, to super-intelligences which coordinate and (benevolently) manipulate humans (Müller 2022; Hamada and Kanai 2022). In Figure 2.1 I order the spectrum based upon the levels of social organization at which the AI agents operate, along with a few examples. At the lowest end would be non-social AIs, and at each step the form of social engagement shifts, first to 'Assistive', with a close relationship to single partner, then 'Independent', able to operate as a social agent within an environment of other agents, and then what I have termed 'Orchestrative', operating as an actor which shapes social environments to change the nature of social interactions. Further steps along the spectrum may be imagined, but I have not identified work towards those steps in the existing literature. Many of the abilities required of

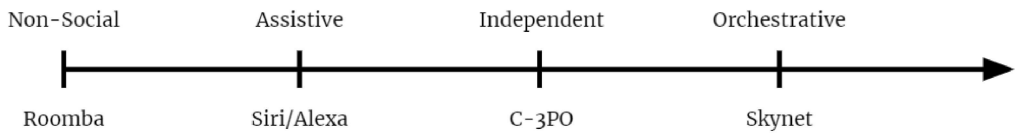
a cooperative AI are shared across visions, but different abilities are emphasized at different points on the spectrum. While humans are able to move between the levels of engagement, and a general intelligence or 'Strong AI' might be expected to as well, it would be possible to train specialized AI agents for particular tasks along this spectrum (Searle 1980). Unlike what would be expected of a general intelligence, the capabilities required to operate at the higher end of the spectrum do not necessarily include the abilities at the lower end.

Using this expanded spectrum facilitates a more precise approach to artificial agents especially from the perspective of their environments. Framing agents based on the sort of environments in which they act allows for expectations based on human behaviors in similar environments and thus an intuitive connection between theory and examples. The next few sections demonstrate this by showing how current research efforts fit into the framework, as well as locating the practical work done in this thesis. The training environment used in this paper is most comparable to the 'Independent' case, where an agent acts within a group of other agents which may or may not have aligned goals.

2.1.1 AI Social Perspectives

AI as Assistive. At the micro level, AI agents can vary in agency ranging from passive 'collector' to 'creator' (Cila et al. 2017). Several authors have argued for aiming towards a midpoint in which AI agents act as personal assistants. These agents would require only a minimal amount of social engagement, interacting with the person they are supporting in the context of a task in which they may specialize. Virtual assistants such as Siri and Alexa are predecessors of this approach, but AI has the potential to greatly enhance the usefulness of such partnerships. De Peuter et al. argue that supportive AIs are better suited than independent expert AIs for

Figure 2.1: Examples of AIs with different levels of social engagement. From left to right, agents are organized by the nature of the social interactions they are expected to have. At the bottom are 'Non-Social' agents, who make little or no distinction between people and inanimate objects. Roomba is an robot vacuum cleaner for household use, which tries to avoid collisions, but does not care whether it has avoided a person or a piece of furniture. One step above this are 'Assistive' agents, which form a close working relationship with a single person. The assistive software programs Siri and Alexa are early versions of this concept. Proceeding further, 'Independent' agents can interact with a variety of people and hold goals which may be more complex. These agents do not currently exist in practice, but could be like C-3PO from *Star Wars*, filling a human-like role. The last level indicated on this spectrum is 'Orchestrative', where agents influence the social environment to promote cooperative outcomes. The fictional examples of these agents tend to be malevolent, like Skynet from the *Terminator* series. A more helpful application could be a route planning agent which helps coordinate commuters in a city.



contexts where the objectives change between tasks and flexibility is key (De Peuter et al. 2021). This result holds empirically even in the highly structured context of video games, where Tylkin et al. found that helping agents trained to assist people lead to better scores than cooperating with an expert AI in Atari games, with similar results shown by Zhang in another game (Tylkin et al. 2021; S. Zhang 2021).

Practically, the assistant model resolves a number of current challenges. From an ethical perspective, AI agents complicate the assignment of moral actions, but assigning full responsibility to human partners clarifies (Müller 2022). The assistant model is also a natural starting point for cooperative AI because it allows shared responsibility, with the AI agents providing whatever value they can while relying upon the help of their human partners to cover gaps where capabilities are not yet available (Xiong et al. 2022). This is precisely what occurs under 'algorithmic triage', which uses confidence estimates to decide whether human intervention is required (Straitouri et al. 2021).

AI as Independent. At a higher level of social interaction, AI agents could be expected to function in human-like roles, either independently or on behalf of a

person. Agents of this sort are currently most common in fictional media, with examples like C-3PO (*Star Wars*) or Lieutenant Commander Data (*Star Trek: The Next Generation*), who function almost identically to humans. These agents would be expected to have social interactions with a variety of different people, and to maintain and act towards complex ends which required varied subtasks (Cila 2022). Potential examples would include a personal shopper AI which goes to stores, selects, and purchases desirable items. Unlike assistants, these agents would cooperate in mixed-motive settings, seeking out other agents with sufficiently similar goals, and requiring concepts of compromise and negotiation. Risks of dual-use technology are more likely at this level. For example, a useful agent would likely need to understand deception well enough to avoid making bad choices, but might therefore also learn to deceive. Research into human interaction with AI often implicitly works towards this standard by choosing specific scenarios and studying the dynamics (Suh et al. 2021; Bennett et al. 2022). One design for this approach is offered by Battistoni et al., who envision AI agents as first-class participants in human-AI interaction, whose terms of engagement are given equal weight as humans' (Battistoni et al. 2021). They particularly highlight the need for human adoption of computer-friendly forms of interaction which occur in specific media such as text, and minimize ambiguity (Battistoni et al. 2021). At this level, it is important to consider whether such AI agents should be made in the image of humans, or as something different. One possible option is to create AI agents interested in collective rather than personal welfare, which may induce cooperation between people as well (Shirado and Christakis 2017).

The training environment used in this paper is also best suited to AIs as independent actors, with application for assistive AI as well. The challenge of learning to cooperate with human partners in multi-agent environments naturally involves agents which choose actions separately and are each impacted by the choices of the others, as would be expected of independent AI agents. Card games like the one used in this paper involve multiple agents working both with and against each other, reflecting the desired scenario. The perfect alignment of incentives

between partners simplifies the finding of friendly and opposed agents, which is out of scope for this task, but would be a logical extension. In partnership, the agents each need to learn how to respond well to the other, and to the variety of possible partners which could occur.

AI as Orchestrative. Within cooperative AI research, a third strand of thought advocates for the use of AI as a tool for social organization. Such an AI would be able to influence social interactions between people as a separate actor. In popular imagination, AIs in this function include Skynet and HAL 9000, which are manipulative and antagonistic, but Hamada and Kanai envision interventions which promote wellbeing (Hamada and Kanai 2022). This is often done by changing the payoffs in social dilemmas to promote cooperation (Baumann 2022; McAleer et al. 2021). Baumann 2022 and McAleer et al. 2021 each use an AI tasked with increasing cooperation and given coercive tools to do so, but it has been shown that simpler cooperative bots can suffice to induce cooperation as well (Traeger et al. 2020). An early practical example here might be a route-planning system which uses traffic data and the desired trips it has been given to help a city of people coordinate their commutes.

2.1.2 Core Aspects of Cooperative AI

The notion of 'Cooperative AI' provides a unifying framework for areas of research which would be required to develop an AI with cooperative capabilities. I use the categories laid out by Dafoe et al., namely Communication, Commitment, Institutions, and Understanding (Dafoe et al. 2020). While Cooperative AI is a broad area and includes things such as using AI agents to promote human cooperation, I focus particularly on the nature of the cooperation between humans and AIs. Research often involves more than one of the four categories at a time, due both to their interconnectedness and conceptual overlap, but the framing is helpful for highlighting common themes between developments.

Communication. The first major area of research in cooperative AI is communication. Aside from the unlikely case of agents whose actions randomly happen to complement one another, cooperation depends upon communication to create understanding, coordinate actions, and accomplish goals. Even in cases where AI agents do not explicitly communicate while undertaking a task, coordination through shared training experience is itself a form of communication (Ritz et al. 2021). The type of communication necessary for any particular cooperation will vary, and notably, cooperation between AI agents involves substantially different challenges than cooperation between AI agents and humans. Inter-AI communication can use technical protocols which send large amounts of structured information (Y. Wang and Sartoretti 2022; Zhou et al. 2021). These messages do not need to resemble human language, and can even include raw observational states or policy gradients almost like a stream of consciousness (Lo and Sengupta 2022). Such messaging is especially valuable in allowing the distribution of decision-making tasks to accelerate coordination between separate agents (Balachandar et al. 2019).

Human-AI communication, by contrast, relies either upon humans learning to operate in machine-friendly terms, or vice versa. Plenty of effort is put into the first option, especially in the history of computing as people have learned to work with punched cards and low-level programming languages. These have been replaced over time with more human-friendly UI such as screens and mice, but human-computer interaction still takes place on different terms than human-human interaction. Researchers have focused on studying how humans perceive communication from computers, looking both at the most helpful forms of text and non-verbal communication through movement (Feng and Boyd-Graber 2019; Dragan et al. 2015). Emulating human appearance with avatars and in speech has also shown promise as a tool for making communication feel more natural (Suh et al. 2021; Bennett et al. 2022). In the environment used in this thesis, communication can take place only through choice of actions taken. As such, any communication learned is likely to be a drawback of the agents, since it will be suited to other AI rather than humans (Hu et al. 2021). The rules-based agent

designed to represent human behaviour does not communicate, so no form of human communication could be learned.

Commitment. In addition to communication, cooperation requires the ability to make credible commitments and to have mutual trust. A major vein of commitment research focuses on so-called 'social dilemmas', scenarios in game theory where pro-social actions are best for the success of the group, but individuals are not incentivized to take them (Macy and Flache 2002). The most classic examples of these dilemmas are the Prisoner's Dilemma and the Snowdrift game, where cooperation may be possible in iterated forms of the games (Doebeli and Hauert 2005). In such a scenario, cooperation often relies on both participants being able to credibly commit to their actions. Examples of each of these games are shown as a normal form game in Figure 2.2. Machine learning techniques have been able to train agents which cooperate in these games, but a number of challenges have been raised (Yang 2021). In the first place, it isn't clear that humans would prefer to work with an AI which has better cooperative outcomes if that AI is not personable (McKee, Bai, et al. 2022). Even if an AI is chosen as a partner, training algorithms which take note of the preferences of others can be exploited by adversarial third-parties to learn unhelpful behaviors (Chelarescu 2021; Fujimoto and Pedersen 2022). Finally, the choice of games has itself been criticized, since games with only one pro-social solution do not require the AI agents to make value judgements about which type of cooperation to prefer (Stastny et al. 2021).

Figure 2.2: Commitment Games. *Snowdrift Game:* This game features a commitment problem because each player only wants to cooperate if the other defects. Therefore, if one player could credibly commit to defecting, the other would cooperate, and the committed player would have their ideal outcome. *Prisoner's Dilemma:* The iterated prisoner's dilemma, which repeats the game shown below, features a commitment problem because neither player can credibly commit to cooperating. Each player would prefer the other to cooperate, but can only incentivise cooperation by conditionally committing to cooperation themselves if the other cooperates.

Snowdrift Game

	Cooperate	Defect
Cooperate	2,2	1,3
Defect	3,1	0,0

Prisoner's Dilemma

	Cooperate	Defect
Cooperate	2,2	-1,3
Defect	3,-1	0,0

Game theoretical approaches have also been used to study commitment more directly. DiGiovanni and Clifton particularly consider the problem of hidden information, since it is easier to trust a commitment from an agent incapable of keeping secrets, and show that conditional information exchange can be used to promote trust (DiGiovanni and Clifton 2022). In this paper, the problem of commitment is deliberately avoided by the choice of testing environment. Since the partners are entirely aligned in their preferences, there is no risk that an AI agent would choose to work towards an outcome which the human would dislike unless it was due to lack of skill in playing the game.

Institutions. Cooperation often involves a variety of possibilities, and strong institutions provide a the social structure necessary to create clear expectations. Building these institutions is a crucial aspect of planning for the future of cooperative AI (Barbierato and Zamponi 2022). Particularly complex is the challenge of teaching normative behaviors, since there is often not a consensus among people what the right choices are to teach (Stastny et al. 2021). The well-known example of this is the case of self-driving cars, where an global online experiment revealed substantial disagreements in who self-driving cars ought to protect in the case of unavoidable accidents (Awad et al. 2018). Rather than choosing any particular view, Koster et al. 2022 propose using a democratic approach to aggregate the normative views

of the impacted parties, but this approach not always tenable (Koster et al. 2022). Entirely opposed views will not be suitable for aggregation in this way, but this is rare, and agents cooperating do not need to be entirely aligned in order to cooperate (Radke et al. 2022).

A variety of other challenges arise from AI which can be ameliorated with the establishment of institutions. Cooperative AI can also pose a threat if it is deceptive, making it important to create insurances of truthfulness (Evans et al. 2021). Further danger arises from the possibility that AI agents could be unaware of the secondary consequences of their actions, though the agents can be taught to consider this issue (Alamdari et al. 2022). In both examples, institutions can require that developers of AI take appropriate care in handling these issues for their agents and use cases.

Understanding. Finally, cooperation depends upon mutual understanding to share goals and coordinate actions. This understanding is a two-directional problem, as AI agents can benefit from a better understanding of humans, but humans will also need to learn to understand the AIs with which they cooperate (Carroll et al. 2019). Observational research suggests that humans do not like working with current AIs, but that this perception may be a preconceived bias rather than reflective of outcomes (Gero et al. 2020). Further education about how AI agents function is also known to result in better outcomes from partnership (Ashktorab et al. 2020). To improve understanding, several authors emphasize the importance of explainability, for both relational and performance reasons (Jovanović and Schmitz 2022; Boggess et al. 2022).

On the other side, substantial effort is being put into teaching AIs to understand humans. A subset of efforts for learning about human partners focus on adaptive agents, which learn about and tailor their behaviors to each specific person (Ghosh et al. 2020; McKee, Leibo, et al. 2022). Adaptive methods are especially valuable when coordination is more important than absolute skill level or when adaptation is itself the goal, as in educational applications (Kabudi et al. 2021). A second

approach has been to use the features of possible actions to coordinate without previous communication, but this is done in very specialized situations and may not generalize well (Ma et al. 2022).

The other major vein of research follows typical reinforcement learning patterns in tasks where success benefits from emergent theory of mind behaviors (Kopparapu et al. 2022). Part of the difficulty of teaching AI about humans is the cost of creating human data. Since common AI training techniques involve learning over thousands or millions of repetitions, finding enough human data to suffice can be prohibitive. However, reinforcement learning techniques without human data can easily learn behaviors which are tailored to other AI agents (Hu et al. 2021). Through repeated play, the agents adapt to their partners, effectively developing understanding of the other agents. Therefore, to understand and cooperate with human partners, some human experience is required.

To resolve this, 'behavioral cloning play' uses imitation learning, which duplicates an expert human demonstration, to create a human-like proxy agent with which to train (Carroll et al. 2019). Alternatively, 'fictitious co-play' trains a wider variety of behaviors in other agents in the expectation that the wide population diversity will reflect the range of behaviors a human might choose (Strouse et al. 2021). Each of these approaches represents human play by another agent, either through imitation or randomness, in order to allow the AI to understand what human partners could be like. This thesis proposes an alternative approach, using rules-based agents to represent human behaviour. Rules-based approaches have historically been used to build AI which contain existing human knowledge, often under the term 'expert systems' (Seirawan et al. 1997; Liebowitz 2019). In cases where rules effectively describe human thought process and behaviour, they offer another perspective which could be used to teach AI agents to understand humans.

Relative to previous methods, using rule-based models to develop understanding of humans has both practical and theoretical benefits. Practically, behavioral cloning play, which imitates human behaviour, is limited by the need to acquire

actual human experience (Carroll et al. 2019). The quality of the final results will depend on the quality of the imitation as a representation of human behaviour. However, to make a more representative agent requires larger amounts of more diverse experience, increasing the cost in terms of human involvement. Imitation learning also lacks a mechanism to identify which aspects of the imitated behaviour are essential rather than incidental. By comparison, a rules-based AI could be built following a typical user experience design process, where an expert plans for the important interaction scenarios. This will not be universally superior, but takes advantage of existing expertise in what people are like to limit the human cost to the time required to build the rules-based agent or agents. This approach can be expected to be most useful in scenarios where human behaviour is well-ordered, but complex enough to be expensive to learn via imitation. The environment used in this game is expected to be one of these, since there are conceptual shortcuts people can make easily which an imitation agent would have to learn. For example, people will recognize the symmetry between suits and follow a single policy in symmetrical positions. As compared to 'fictitious co-play', rules-based play offers an approach to capture actual human behaviour, rather than approximating with randomness. Since randomness works by covering a broad enough field of options to probabalistically ensure representation, there will likely also be a large number of unrepresentative behaviours which reduce the quality of cooperation. Instead, rules-based agents may more closely approximate human actions, resulting in agents which expect a narrower range of more likely behaviours.

2.2 Practical Methods

The practical aspects of the method can be divided into four majors components which each produced a substantial artefact. A partial summary of each is provided in Figure 2.3 below in chronological order.

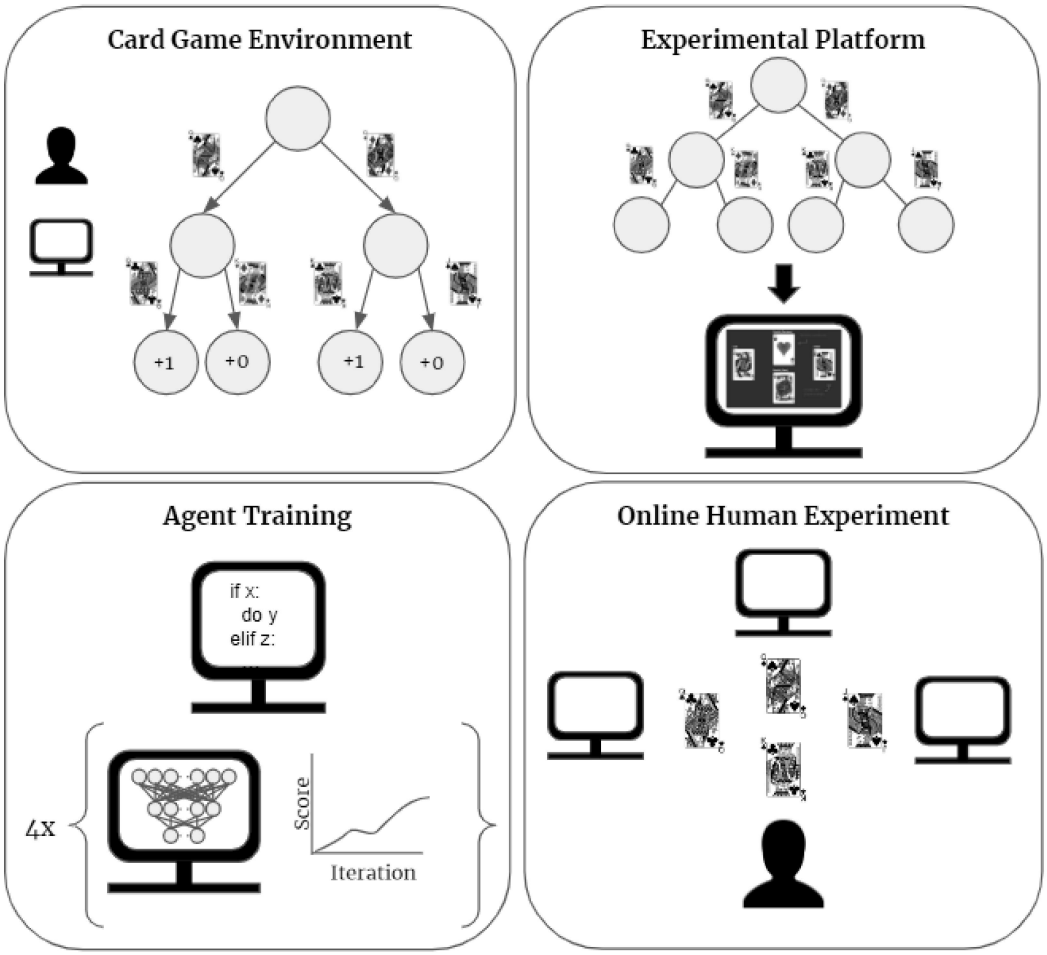
Detailed descriptions for each section may be found:

- Card Game Environment: the details of how the environment was designed and implemented in Section 2.2.1
- Reinforcement Learning Agents: the motivations and training methods for each of the agents are in Section 2.2.2
- Experimental Platform and Online Human Experiment: the platform and some of the experiment are described in Section 2.2.5. The majority of the discussion of the experiment is in the Results, Section 3.

2.2.1 Card Game Environment

In reinforcement learning, games are often used as models for more complex problems. Games are highly-controlled environments where the choice of game allows control over the particular challenges that an AI will have to face. For example, recent papers have used No-Limit Texas Hold'em and Hanabi as scenarios for operating under incomplete information and interpreting the actions of other players (Brown and Sandholm 2017; Hu et al. 2021). The environment in this experiment is based on the card game Spades. It is a four-player partnership game, where the players take turns choosing cards from their hands to play in order to collect points. The environment involves incomplete information about what each other player is able to play, and is purely cooperative within the partnerships, but purely competitive between them. Spades is in the family of trick-taking card games, similar to Bridge,

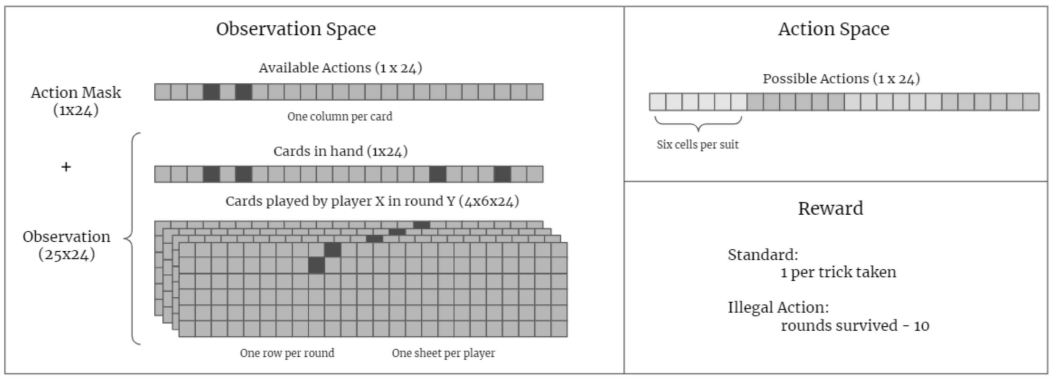
Figure 2.3: Major Methodological Elements. *Card Game Environment:* A simple game was designed to include cooperation and implemented as an environment following the standard set by OpenAI Gym. *Experimental Platform:* The environment was translated to a web-based format using the Empirica platform. An experiment flow was built to include instructions, game playing, and survey questions. *Agent Training:* A rule-based agent was constructed to play the game based on game theory and prior observed behaviour. Four deep Q agents were trained, three based on existing literature and one novel in this thesis. The novel agent uses the rule-based agent as a training partner. All of them were trained in parallel on Amazon Web Services compute resources. *Online Human Experiment:* Human participants were recruited for online play with the AI agents.



which have been previously studied for the purpose of developing artificial agents (Niklaus et al. 2019; Rong et al. 2019; Cohensius et al. 2020). These games feature a few challenges which distinguish them from games where AI has already surpassed human abilities. In the first place, card games tend to feature large amounts of hidden information and stochasticity. These two issues also appear in Poker, and a particularly difficult form of Poker was played at a superhuman level by Brown and Sandholm in 2017 (Brown and Sandholm 2017). Partnership games add another crucial complexity in the need to coordinate actions with another player. This coordination is particularly difficult in the bidding phase of games, when a large number of conventions are used to communicate hidden information (Rong et al. 2019; Cohensius et al. 2020). To emphasize the role of coordination rather than conventions, I use a variant of Spades where rather than bidding and trying to take the targeted number of tricks, players are simply rewarded for the total number of tricks taken. To increase the likelihood that human players are willing to play several games, the hand size is also limited to only six cards each.

The environment is constructed in Python, following the standards set by OpenAI Gym and PettingZoo (Brockman et al. 2016; Terry et al. 2021). The observations available to the agent at each step contain the full information of the environment which has been available up to that stage. This means that the agents, unlike many human players, will have perfect recall. The observation is encoded as a one-hot array, with rows corresponding to when each card was played, and columns corresponding to each card. The actions available are presented as a mask within the action space. Since the legal actions change greatly from turn to turn, this allows the shape of the action space to remain constant. Using these standards means that the game can easily be connected to agents trained by others, as well as being shareable for re-use. Figure 2.4 shows the particular details of the environment.

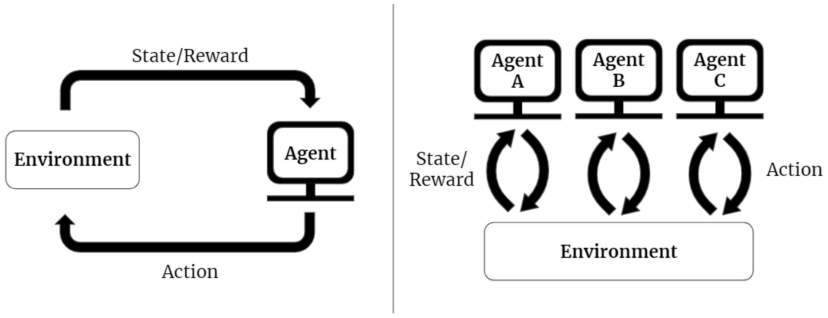
Figure 2.4: Card Game Environment Specifications *Observation space:* the observation space consists of a dictionary with two elements, the action mask and the observation. Each has 24 columns, one per card in the game, which are one-hot encoded. The action mask is a filter for legal actions based on the state of the game. The observation has 25 rows, with the first row corresponding to the cards the agent is holding, and the other 24 rows corresponding to cards which have been played. Rows 2-7 indicate what the first player played in rounds 1-6, and similarly in sets of six for rows 8-13, 14-19, and 20-25. *Action space:* the action space consists of all possible actions during the game, which is playing each card. The cards are encoded as numbers 0-23, just as in the observation data, ordered from low to high and grouped in suits. *Reward:* the reward in most cases is simply the number of tricks taken by the agent during the game. In the event that a rule is broken, the rule-breaking agent receives a penalty of -10, but earns 1 point per round survived, to distinguish between illegal play in earlier versus later rounds. The penalty reward is useful if the environment were used without taking advantage of the action masking.



2.2.2 Reinforcement Learning

Machine learning approaches can be broadly categorized into three major areas: supervised learning, unsupervised learning, and reinforcement learning. Of these, reinforcement learning is concerned primarily with an independent agent learning to perform a task by repeatedly generating experiences from its environment. Figure 2.5 shows this cycle with the typical terms for each step. For a single agent, this is generally treated as a Markov Decision Process (MDP), which is a formalization of the process in game-theoretical terms (Kiran et al. 2022). A MDP is comprised of a set of states, $s \in S$, actions, $a \in A$, a transition function moving between states based on actions $T : (S, A) \Rightarrow S$, and a reward function $R : (S, A) \Rightarrow \mathbb{R}$, and is so named because it has the Markov property that the possible future states

Figure 2.5: Reinforcement Learning. *Left:* In reinforcement learning, an agent chooses actions which are sent to the environment. The environment changes in response, and returns its new state, along with any rewards which may have been obtained. This is repeated many times as the agent learns to choose actions which optimize the rewards. *Right:* In multi-agent reinforcement learning, the same process occurs with many agents at once. Due to the impact of the other agents, sending the same action to the environment does not necessarily produce the same result, making learning more difficult.



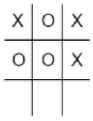
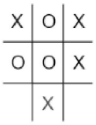
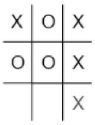
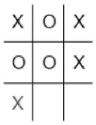
depend only on the current state and actions (Puterman 1994). Several variants of MDPs have been studied which add properties such as 'partial observability', where the agent has limited information about the current state, which applies to games where there is hidden or stochastic information including the one used in this study (Kaelbling et al. 1998).

Q-Learning. An intuitive approach to reinforcement learning is to learn a function which maps from states to the expected reward of choosing each possible action in that state. This approach was originally developed by Watkins and Dayan, who called it a 'Q-function', defined in terms of the MDP variables as

$$Q(s,a) := \mathbb{E}\left(\sum_{t=0}^{T} \gamma^t r_{t+1} | s_0 = s, a_0 = a\right),$$

where γ is a discount factor between current and future rewards, and T is the total number of timesteps in the environment (Watkins and Dayan 1992). They similarly showed that Q-learning would "converge to optimal action-values with probability 1 so long as all actions are repeatedly sampled in all states and the action-values are represented discretely" Watkins and Dayan 1992. This result is only for a single-agent environment. Claus and Boutilier extend to the multi-agent case and show that

Figure 2.6: Partial Q-Table. In a game of Tic Tac Toe, the X's are next to play with the current board state as shown in the left column. For each possible action indicated in red, the value of the resulting state is listed at the bottom. An agent can then act by choosing the actions which maximize the Q-function.

State	Actions		
			
Q-value	0	1	-1

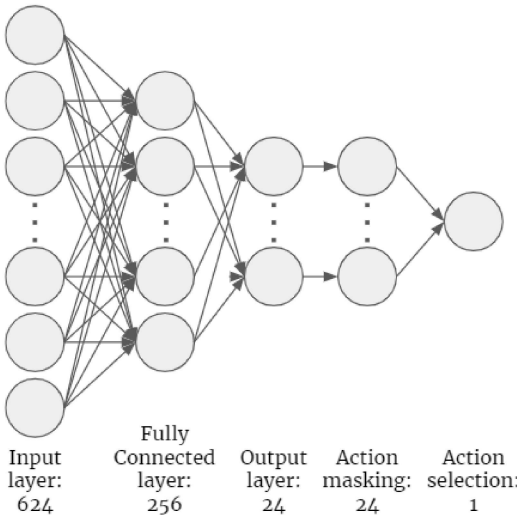
although good results are often achieved, the same assurances do not apply (Claus and Boutilier 1998). An example of a partial Q-table is shown below in Figure 2.6.

Similar to other reinforcement learning methods, there is an inherent trade-off between exploration of new possible actions and exploitation of best current beliefs to maximize rewards. Similar to other areas of machine learning, a typical approach is to create a tunable hyperparameter $0 < \epsilon < 1$, such that each time an action is taken it is chosen uniformly at random with probability $1 - \epsilon$ and is otherwise chosen by maximizing the value of the Q function (Kiran et al. 2022). This value can be made to approach 1 as the learning continues, so that the expected depth of traversal in the state tree before selecting a random action increases, effectively shifting exploration from a broad range of actions initially towards exploration deeper in the tree as more confidence is built in the best initial actions. In the case of this paper, sampling at random from the action space is not suitable, since only a small subset of actions may be chosen at any particular point in time. To deal with this, an alternative randomization is used, with chooses with a probability proportionate to the Q-values for each action. The actions which are ineligible are set to always be zero so that they are not selected. This approach favors more exploration to contrast between actions with similar expected rewards, which is

expected to be helpful in card games, where there are likely to be a number of obviously bad choices, but several viable options between which to choose.

Deep Reinforcement Learning. In the simplest form, Q-learning can be performed using a lookup table, storing the expected values for each state-action pair and updating them as learning progresses. Since common environments are often combinatorially large, this is not always a feasible approach. Chess, as a famous example, is thought to have on the order of 10^{120} possible positions. Instead, modern approaches use function approximation, replacing the lookup table with a neural network which estimates the value of Q for any state-action pair (Mnih et al. 2015). More advanced variants such as Double Deep Q and Dueling Deep Q networks extend the core approach by splitting the task into two networks in different ways (Z. Wang et al. 2016; Hasselt et al. 2016). To save computational cost, this thesis uses a single Deep-Q network based on the Mnih et al. 2015 as implemented in RLLib.

Figure 2.7: Deep Q Network Specifications. The layers of the deep Q network are shown above. The first two layers are fully connected, followed by action masking which removes the possibility of choosing illegal actions. The final selection is made based on the maximum Q value in rollout. Exploration is done with probabilities proportionate to the Q values during training.



2.2.3 Training to Cooperate

In studies of top performing human-AI partnerships, a contrast can be made between AI agents which are being used in parallel with human experts, and AIs trained specifically as helpers. In the first case, emphasis is placed on topic expertise, whereas the second focuses on complementarity. In theory, complementary expertise is more useful, since it can cover the gaps where a human would have made a mistake, and thus improve overall outcomes, and this holds up in experimental testing (Q. Zhang et al. 2022). However, complementary expertise is difficult to acquire, since it requires understanding what humans can be expected to do well. The obvious approach here is to train the agents with humans, and this has shown success, but is also prohibitively expensive in terms of human time required (Swamy et al. 2019). Alternative approaches which show promise include using imitation learning to replicate experience with humans, and training a wide variety of agents to broaden the compatibility of the final AI (Julien Dossa et al. 2019; Carroll et al. 2019; Fontaine et al. 2021; Strouse et al. 2021). This work is inspired by the imitation approach, but compares to the second approach, as it was more successful in at least one direct comparison, (Strouse et al. 2021). Strouse et al. designed an approach called 'Ficitious Co-Play', which trains a pool of agents to perform well with both each other and all previous versions of themselves. By including previous agents in the training set, the overall pool is diversified, which both reduces the risk of learning behaviors tailored to the particular population, and also increases the number of poorly performing agents, which better reflects the expected skill level of most humans on the task used in the study (Strouse et al. 2021). By contrast, this paper will consider the problem of simulating human experience when human expertise is more competitive with that of the AI agents, and potentially stronger.

For the purposes of constructing rule-based agents, it is convenient to choose a task where human-behaviour has been previously studied. The classic approach to modeling strategic behaviour would be to use solution concepts from game theory, but very few experiments have claimed that people actually use the strategies

predicted by game theory, and the quality of those results is contested (Palacios-Huerta and Volij 2008; Wooders 2010; Levitt et al. 2010). These papers, however, focus on games where the min-max solution involves mixed-strategies, and the authors suggest that the ability to reliably randomize may be a limiting factor, rather than a lack of intention to use min-max strategies. In light of this, rather than a full-game policy, I focus on using rules to remove weakly-dominated strategies, and randomize between options where multiple behaviours could be rationalized. This approach focuses the rules on particular cases where game theoretical policies are easier to recognize and use, since in the game chosen the dominated strategies can mostly be described as 'wasting good cards', a straightforward concept. Helpfully, one of the key instances of this dynamic is cooperative, when a partner has already won the round, and playing a card to beat the partner would be 'wasteful'. A more expansive set of solution concepts was considered, but this simple set of rules already brought the performance of the rules-based agent in line with the trained agents. To avoid making the rules-based agent a superior training partner simply by virtue of having better strategies, it was limited to this level of play.

2.2.4 Agent Types

Four different agents are tested, three of which are drawn from existing literature as comparisons, and the fourth of which is novel to this thesis. For this experiment, eight agents were trained via self play for four repetitions of 40,000 games. The agents were then finished with each of the methods below over another 160,000 games. All of the training was performed using Amazon Web Services t3-large instances and took approximately 280 hours. The full state space of the game is exponentially large, as with many games, so this training is only a very small fraction of the overall, and the models will only succeed if they generalize from the training. Each is motivated and described in turn below:

Self Play. Self play is the process of learning to perform a task via repetition in an environment where the other agents are all copies of one another. In environments with a single agent, self play follows the standard reinforcement learning paradigm described previously. This method has been successful in zero-sum cases with only two agents as well, where the agent essentially treats the other player as part of the environment (Silver et al. 2018). In those competitive environments, the agent is incentivized to exploit weaknesses in itself and gradually converge towards mutual best responses (Claus and Boutilier 1998). With larger numbers of agents and in cooperative environments, self play has been shown to have difficulty generalizing to new partners in what is termed 'zero-shot' cooperation (Foerster et al. 2019; Hu et al. 2021). Many of the guarantees of convergence which can be shown for single agent reinforcement learning do not hold for multiple agents because the environment is now learning and not stable (Lanctot et al. 2017). From a theoretical perspective, learning only via self play can be expected to produce agents which only 'understand' how they themselves would act. Therefore, it should be expected that self play is most useful in environments where there are a small number of agents, a unique best policy, and minimal need for conventions, which together minimize the need to understand other agents. In social and cooperative settings, these are relatively rare occurrences, since any sort of communication will rely upon conventions, and social settings definitionally involve multiple agents. However, as the simplest approach, self play makes a suitable benchmark against which to measure the performance of more sophisticated methods.

Pool Play. Pool play is a variation upon self play, in which several agents are trained independently then combined into a 'pool' in which they all train by playing with one another (Lanctot et al. 2017). This approach helps to expand the variety of actions taken in training, which in turn requires the agents to each learn policies which are best responses to a wider range of possible partners. In practice, when agents are learning simultaneously through pool play they can converge towards similar policies to one another, reducing the benefits of diversification (Qin et al.

2019). This could occur whenever a plurality of agents take actions which share a best response. Over time, the agents will all prefer to respond well to those actions, thus converging their behaviours towards uniformity. In this case, there are a small enough number of possible actions at each step that the variety provided by pool play may provide a useful diversification. In working with humans, this diversity may make the model more flexible, but does not target an understanding of behaviour, potentially leaving an opportunity for even better techniques.

Fictitious Co-play. Fictitious co-play extends pool play to consider the challenge of working with partners at a variety of skill levels (Strouse et al. 2021). In many environments, AI agents can achieve superhuman performance, so this question begins to consider the particulars of understanding human behaviour rather than just diversity of behaviours. The paper which introduced it focused on the game *Overcooked*, where human play is not obviously optimal, and a variety of behaviours could be shown even from the same person in the same game. Fictitious co-play follows the pool play model of training several agents separately and then with one another to broaden the range of training partners. In the group training phase, previous checkpoints from each self play agent are included in the pool, which adds in experience with lower skill partners for no additional cost. The method draws inspiration from fictitious self play, which is a variant of self play that includes past versions of the self agent in order to prevent repeated cycling through behaviours (Heinrich et al. 2015). In the *Overcooked* environment, a few experiments have shown fictitious co-play to have state-of-the-art performances with human partners. However, success in the *Overcooked* environment can often be achieved by a single highly-skilled agent simply working around a partner. This makes it particularly suitable to fictitious co-play and other methods which focus on training with low skill partners. The bar for 'understanding' a human partner can therefore be low. By contrast, limited information in card games increases the value of good partnership and will test whether the agents are learning interactive forms of cooperation.

Behavioural Cloning. Behavioural cloning is an additional method which focuses on understanding human behaviour, although it is not used in this thesis. The previous methods have all focused on learning generalised behaviour through random variation in training partners. By contrast, behavioural cloning specifically aims to include human behaviour in the training so that the AI agents will learn targeted cooperative actions. To do this, human-imitating agents are trained using imitation learning to replicate recorded human actions in the environment, which then form a pool of training partners for the final agent (Carroll et al. 2019). Using a wider variety of human partners also helps to capture differences between people (Knott et al. 2021). This approach can increase the effectiveness of training by spending more time learning how to coordinate with high probability actions, improving the focus relative to randomness. The main issue with behavioural cloning play is the need to include human data, which is expensive to collect, and the reason it is not included in this thesis. It is also challenging to ensure that the human training data is sufficiently representative and does not bias the agent towards working better with certain people. The importance of this risk will vary with the environment, including whether there are meaningfully different human approaches, and whether the agent will need to be broadly useful, or whether it can be custom tailored.

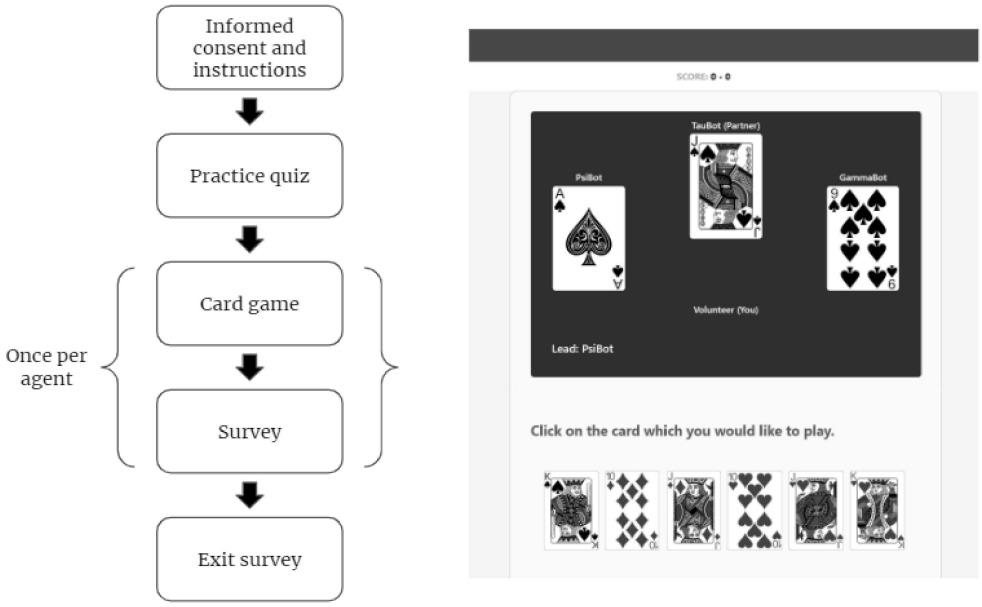
Rule-based Play. The final method, and one of the novelties of this thesis, is rule-based play. Inspired by behavioural cloning, the goal of rule-based play is to simulate much of the benefit of using human training data without actually involving human training partners. To accomplish this, rule-based play follows a similar pattern of creating human-imitating training partners for the reinforcement learning agent. Instead of using imitation learning, the training partners are hand-coded rule-based systems which reflect existing knowledge about how people act in the environment, as well as game-theoretical approaches. For card games in particular, previous studies have shown that people will follow partial solutions predicted by game theory such as avoiding plays which are weakly dominated (Teófilo et al. 2014). Psychological studies also argue that people leverage rules to learn concepts, and

historically people have used rules to build AI 'expert-systems', both suggesting that rule-based approaches may be able to approximate human behaviour better than randomness (Nosofsky et al. 1994; Goodman et al. 2008; Rabi et al. 2015; J. Wu et al. 2020; Seirawan et al. 1997). Relative to collecting training examples, writing rules allows a more precise targeting of what behaviours ought to be included as 'human' and which are incidental. It also shifts the burden of generalization away from the imitation learner and back onto the designers of the model. This takes advantage of pre-existing understanding and makes game-theoretical approaches useful when there is reason to believe people might follow them. In the particular case of this card game environment, a few simple rules are used which people might be expected to follow, and which can be justified on the grounds of rationality. First, the agent does not expend stronger cards scoring points when a weaker one would certainly sufficed. Second, the agent does not discard strong cards when weaker cards are available to be discarded. Finally, if the agent can score a point, it does so. The first two rules can be clearly justified on the basis of 'weakly-dominated actions', since the immediate outcomes are the same, and the potential future uses of the stronger card are no worse than those of the weaker card. The same cannot be said of the third rule, since in rare cases it could be better to allow an opponent to score now in order to score more later. Based on the nature of the environment, the value of earlier points is higher than later ones, making it likely that people do follow this rule. The applicability of this rule is assessed empirically in the study.

2.2.5 Online Environment

The card game environment was launched as an online experimental platform using Empirica to collect human interaction data (Almaatouq et al. 2021). Four pre-trained agents, as well as a random agent, saved as networks and loaded as static players in the online game. Volunteers were recruited via social media (Reddit and Facebook) to play eight games, two with each tested agent, against a pair of random agents. The experiment flow is shown at left in Figure 2.8. To begin, they are given

Figure 2.8: Experiment Flow. *Left:* Each participant is given instructions on how the experiment works while requesting consent. They then complete a practice quiz to verify that they understood the rules. Once the quiz is complete, they play two games with one of the agents, followed by a survey. The game-survey pairing is repeated for each pre-trained agent. At the end, a final exit survey is completed. *Right:* An image of what the experimental setup looks like.



the rules, along with labelled example images from the games to explain. They are then required to successfully complete a quiz about the rules before beginning to play. Games are played in pairs with the same agent to give volunteers more time to learn about their partners. An example of a player's turn is shown on the right in Figure 2.8. After each pair of games, players were asked to rate the quality of their partner and to compare their current partner to the previous one. Additional images of the experimental setup are included in Appendix A

3

Results

Results are broken into three sections. First, general descriptive statistics are provided for the whole experiment. Next, realized performance is compared both across the models in the human-AI games, and between the human-AI pairings and benchmarks with other partners. Finally, results from the survey are summarized and analyzed in the context of performance and decisions taken.

3.1 Descriptive Statistics and Benchmarks

The tables below show high-level descriptive statistics regarding the experimental data collection. These descriptors pertain to the gameplay and scores from the games. Benchmarks collected from running simulations with the AI agents are also included here. Survey data is described in subsequent sections.

Table 3.1 below shows the number of participants, grouped by the numbers of games they played. The complete experiment includes eight games, but a few participants left early. Scores are measured in terms of tricks taken per game, which takes whole number values ranging from 0 to 6 inclusive. The last column converts this to a winning percentage of tricks. Based on a pairwise t-test, there is not a statistically significant difference between the number of tricks taken per game by players who left early versus those who played all eight games. The games are also conceptually distinct from each other, so the tricks taken data from incomplete trials results can be safely included in the analysis.

Table 3.1: Participant Descriptive Statistics. The experiment contained eight games, which most participants completed. Of those who finished less, the number of games played and mean scores are shown. There is no statistically significant difference in the scores of participants who left the study early.

Games Played	Number of Participants	Mean Score	Winning Percentage
8	40	3.29	54.8%
7	1	3.17	52.8%
6	3	3.27	54.5%
3	4	3.25	54.2%
2	1	3.00	50.0%
Total	49	3.29	54.8%

Table 3.2 shows the number of games played with each pre-trained agent. The numbers are not identical due to the early quitting mentioned above. The order of agents is randomized, so the dropouts do not consistently impact any single agent. A total of 350 games were played across the four agents, with the agents on average taking a bit more than half of the six tricks per game.

Table 3.2: Artificial Agent Descriptive Statistics. The number of games and the mean number of points scored are shown for each type of agent. The number of games played is not identical due to participants leaving the study early.

Model	Number of Games Played	Mean Score	Winning Percentage
Self Play	86	3.31	55.2%
Pool Play	93	3.17	52.8%
Fictitious Co-play	83	3.32	55.3%
Rule-based Play	88	3.35	55.8%
Mean	87.5	3.29	54.8%

Each agent was also benchmarked in a set of games with an identical agent and with a random partner as a comparison to working with a human. The models were run for 20000 games with the same set of random seeds, and their scores are shown in Table 3.3. The standard deviations of the distributions are also shown, and consistently fall in the range of 1.4.

Table 3.3: Artificial Agent Random Benchmarking. The baseline distribution of scores is shown for each agent based on 20000 simulated games with a random partner.

Model	Mean Score	Standard Deviation	Winning Percentage
Self Play	3.05	1.41	50.8%
Pool Play	3.03	1.40	50.6%
Fictitious Co-play	3.04	1.40	50.7%
Rule-based Play	3.05	1.41	50.8%
Mean	3.04	1.40	50.7%

Similarly, Table 3.4 below show a benchmark where each model is tested with a copy of itself as a partner. The models were run for 20000 games with the same set of random seeds while rotating which player goes first. The means are slightly higher than with random partners, and the standard deviations are similar. The differences in means are statistically significant only for self play and fictitious co-play, with p-values shown in the last column. Since all of the games for both benchmarks are played from the same set of seeds, all of the differences indicate

better performance in identical scenarios even though some of the distributions are not shifted enough for significance.

Table 3.4: Artificial Agent Self Benchmarking. The baseline distribution of scores is shown for each agent based on 20000 simulated games with an identical partner.

Model	Mean Score	Standard Deviation	Winning Percentage	Difference to Random	p-value
Self Play	3.07	1.43	51.3%	0.02	0.03*
Pool Play	3.06	1.40	50.9%	0.03	0.12
Fictitious Co-play	3.06	1.39	51.1%	0.02	0.04*
Rule-based Play	3.07	1.43	51.0%	0.02	0.24

3.2 Performance Comparison

To compare the relative ability of the agents to cooperate with humans, Table 3.5 shows the means and standard deviations in each set of samples. Table 3.5 also shows the comparisons between each model and the AI-random and AI-AI team benchmarks, with p-values for the differences between them shown in the final columns. The difference between the benchmarks and human collaboration is statistically significant for fictitious co-play relative to the random benchmark, and for rule-based play relative to both benchmarks. The other p-values for self play and fictitious co-play are also less than 0.10, but do not reach the 0.05 significance level.

Table 3.5: Human-Agent Performance Means and Standard Deviations. For each model, the mean scores are shown with a human partner, a random partner, and a identical partner. The differences between the scores of the human-AI partnership and the two benchmarks are tested for significance, with the p-values shown.

Model	Games Played	Mean Score	Std Dev	AI-Random B'mark	Diff p-value	AI-Self B'mark	Diff p-value
Self Play	86	3.31	1.33	3.05	0.07	3.07	0.10
Pool Play	93	3.17	1.32	3.03	0.31	3.06	0.39
Fictitious Co-play	83	3.32	1.30	3.04	0.05*	3.06	0.07
Rule-based Play	88	3.35	1.26	3.05	0.03*	3.07	0.04*

Table 3.6 shows the p-values for pairwise t-tests between the models, adjusted using Tukey's honestly significant difference method to account for repeated pairwise testing. The point estimates for the newer methods are higher than the older methods, but none of the differences are significant with the small sample size.

Table 3.6: Pairwise t-tests for Difference of Means between Models. The score distributions of the human-AI partnerships are compared for significance testing. The p-values for a two-sided t test are shown below.

p-values $\mu_1 \neq \mu_2$	Self Play	Pool Play	Fictitious Co-play
Pool Play	.88	-	-
Fictitious Co-play	.90	.85	-
Rule-based Play	.90	.76	.90

The goal of the new method is to use understanding to improve performance, so an effort was also made to assess the quality of the models' understanding at its potential relevance. The most direct measure of understanding would be the compatibility of decisions made with human partners as done above. As a second measure, this paper uses the ability of the model to predict human decisions based on their inputs. As the models were trained in large part via practice with themselves and other similar agents, a higher ability to predict human actions increases the chances that the agent has also learned behaviours adaptive to working with those human actions. For each participant and agent, I compute below a measure of the AI's ability to predict the human decisions using cross entropy, derived from Shannon Entropy (Shannon 1948), which is given as

$$H(X) := -\sum_{x \in X} q(x) \log(p(x)),$$

where $p(x)$ is the probability of choosing action x with the AI model, and $q(x)$ is the probability of choosing action x by the human (either 1 or 0). For each model, the possible actions are chosen from a distribution, with the weights on each action being the values learned via training. To compute the entropy of a human action, I take the product $-q(x)\log(p(x))$ for the value of x corresponding to the

human action, with the values $p(x)$ computed by the neural network based on the observation which was available to the human player. This is simply summed across all the choices made in a game to produce the total entropy of that game for that agent and human player. To standardize across different games, I also subtract out the entropy of a uniform random process predicting the same actions, so that agents do not score better if the game in question happened to involve less decisions than another game. The first table below, Table 3.7, shows statistics regarding the average ability of each agent to predict human decisions. Positive values indicate that the agent is better than random guessing, so self play has the highest ability to predict human play, and all four agents are on average better than random.

Table 3.7: Entropy of Predicting Human Plays. The Shannon Entropy between the distribution of actions chosen by each model and those chosen by human players is given below. The entropy values are normalized by subtracting the entropy of a random uniform distribution across the action space, so that positive values indicate more similarity in distribution than random.

Model	Mean	Standard Deviation
Self Play	0.140	0.65
Pool Play	0.130	0.60
Fictitious Co-play	0.128	0.62
Rule-based Play	0.133	0.62

To provide further intuition about these numbers, the distributions are plotted below in Figure 3.1. All of the distributions are centered slightly above zero and are broadly symmetrical. Figure 3.2 shows the distribution of entropy values for the random uniform predictions. Unlike the others, the random distribution is not re-centered, and therefore has a much wider range, reflecting the variation in the number of options available.

Figure 3.1: Normalized Entropy Distributions. The realized entropy scores of each model are shown for all of the human actions taken in the experiment.

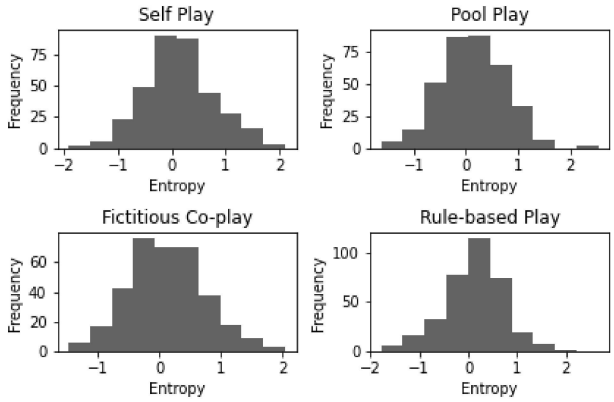
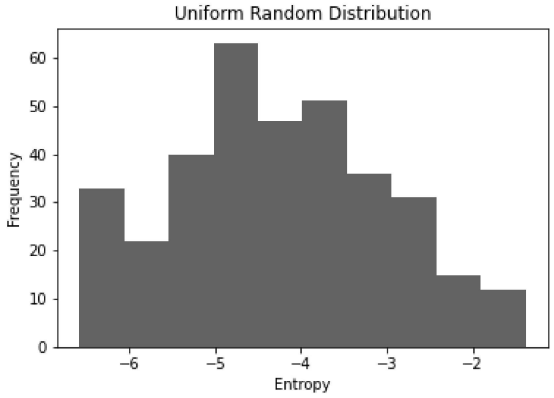


Figure 3.2: Random Entropy Distribution. The distribution of entropy scores between a uniform random distribution and human actions taken in the game is shown. The sign is inverted to align with the convention used in other tables and figures of more positive values meaning higher similarity.

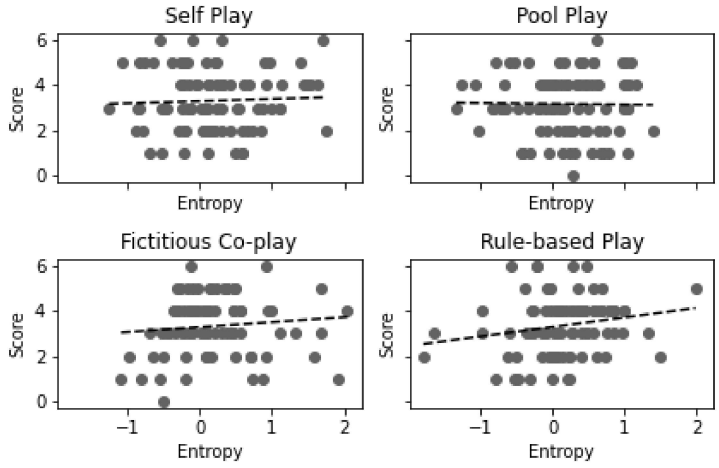


This measure of model fit is then compared on a per-game basis to the scores achieved in each game. The correlations and significance testing are shown in Table 3.8. The relationship is strongest for the newer models, and the p-values are smaller, but do not quite reach statistical significance. Figure 3.3 also shows scatter plots, which show generally positive relationships for fictitious co-play and rule-based play, but no little relationship for self play or pool play.

Table 3.8: Pearson r for Prediction Entropy and Scores. Correlation coefficients between the entropy scores for similarity to human actions and survey preference scores are shown, along with p-values.

Model	r	p-value
Self Play	0.05	0.66
Pool Play	-0.02	0.88
Fictitious Co-play	0.11	0.34
Rule-based Play	0.20	0.07

Figure 3.3: Scores vs Model Predictiveness. The survey-based preference scores and the similarity-based entropy scores are compared using a scatter plot. They are statistically uncorrelated. Rule-based play may show a relationship, but it is not visually clear that there should be one.



For the rules-based model, the training partner is either deterministic or chooses moves uniformly at random, so entropy does not provide a useful measure of the representativeness of the training partner. Instead, the predictiveness has to be assessed based on how often the moves selected are actually played by human players. Similarly to the entropy calculations, I consider only moves where there was a choice to be made, but in this case I simply count how many of the chosen moves match with human plays. Table 3.9 shows these counts considering only the cases where the rules-based training partner chooses an action. Note that there can never be six choices made, since the final card is always the only option

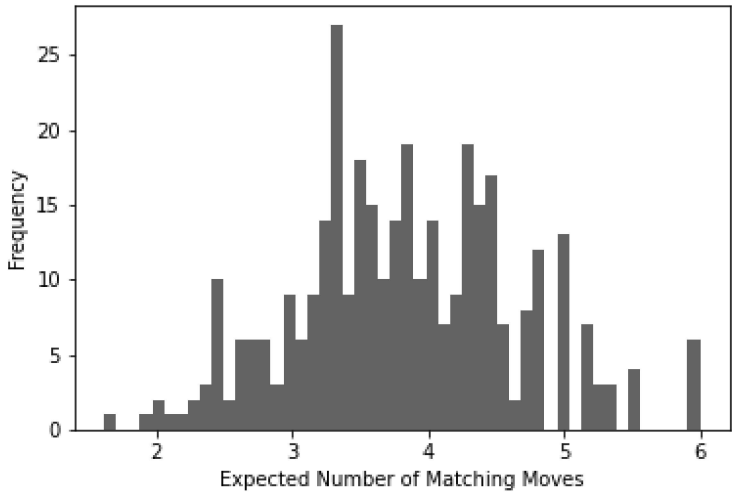
Table 3.9: Rule-based Training Partner Similarity to Human Play. The frequency with which the rule-based training partner accurately predicted human actions is shown out of the total number of times a rule was applicable. Cases where no rule applied or where only one choice was available are not counted towards the total number of decisions made.

Matching	Number of Decisions Made					
Decisions	0	1	2	3	4	5
0	129	25	11	1	1	0
1	-	38	37	12	5	1
2	-	-	38	25	8	2
3	-	-	-	11	5	1

available, and no values can fall below the diagonal. In nearly a third of games, the rules-based training partner did not apply any of the rules. On average, there were 1.4 cases per round where a rule applied. When a rule was applied, it matched the human decision in 61% of cases.

Figure 3.4 shows an all-in calculation of the expected number of plays per game which would match, including both only-move choices and random choices. The minimum possible value is 1, which never occurs. There are 6 games out of 350 where the rules-based training partner correctly predicted every human move. On average, the training partner was correct on 3.84 moves per game, moderately more than half after accounting for the last move always being correct.

Figure 3.4: Expected Human-Similarity of Training Partner Predictions. For each game played, the rules-based training partner was scored on how many human actions were predicted correctly. When the rule-based agent provided a distribution of possible actions, the total was increased by the probability of choosing the correct action, so that the total score is the expected number of matching actions in each game.



As a comparison, a random agent would be expected to match 3.54 moves per game, essentially half of the total, as seen in Figure 3.5. The expected number of predicted moves per game by each agent is shown below in Table 3.10. Of the agents, only the rule-based partner was designed with the goal of being similar to human play, and it is also the only one which is clearly more similar to human play than a random agent by this metric.

Figure 3.5: Expected Human-Similarity of Random Predictions. The chart shows the number of human actions which a uniformly random agent would be expected to accurately guess in each of the games played in the experiment.

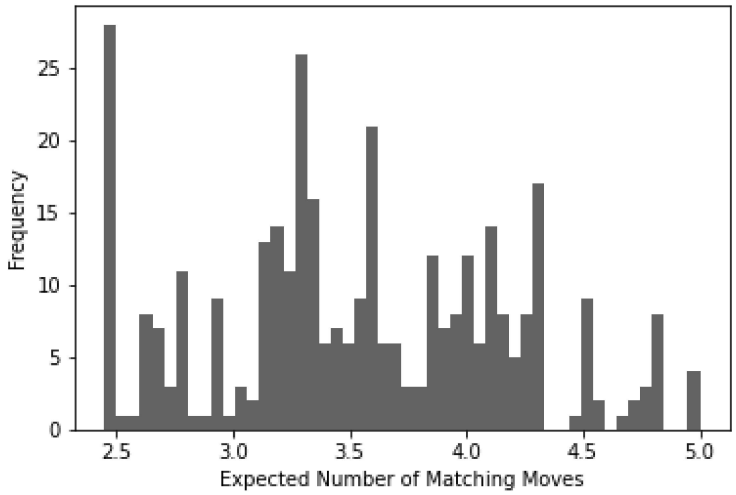


Table 3.10: Expected Number of Human-matching Moves per Game. The expected number of matching moves is shown for each model used in the experiment. The deep Q models are shown above, with the random and rule-based models below.

Model	Mean	Std Dev
Self Play	3.54	0.67
Pool Play	3.53	0.65
Fictitious Co-play	3.53	0.68
Rule-based Play	3.53	0.67
Random Play	3.54	0.64
Rule-based Partner	3.84	0.82

3.3 Survey Results

Alongside the games, players were also asked about their views on the cooperative abilities of the agents with whom they were paired. Table 3.11 below shows the total votes counted for each comparison.Neutral votes are noted in parentheses. Close to half of all votes cast were neutral, reducing the power of the subsequent comparisons.

Table 3.12 shows the net votes in each category, counting neutral votes as zero. The numbers are shown as net preference for the row entry over the column entry.

Table 3.11: Preference Survey Votes per Model. The total votes counted in the surveys are recorded indicating which pair of agents they were asked to compare. The number of neutral votes is shown in parenthesis.

Total Votes Row vs Column	Self Play	Pool Play	Fictitious Co-play	Rule-based Play
Self Play	-	20 (8)	19 (10)	18 (9)
Pool Play	20 (8)	-	24 (12)	12 (6)
Fictitious Co-play	19 (10)	24 (12)	-	18 (5)
Rule-based Play	18 (9)	12 (6)	18 (5)	-

Table 3.12: Preference Survey Net Votes per Model. The net votes counted in the surveys are recorded for each pair of agents. Positive numbers indicate that the row model was preferred to the column model.

Net Votes Row > Column	Self Play	Pool Play	Fictitious Co-play	Rule-based Play
Self Play	-	0	-1	-3
Pool Play	0	-	-4	2
Fictitious Co-play	1	4	-	3
Rule-based Play	3	-2	-3	-

For each pair, the score can be tested as though it were binomially distributed, and the p-values for this test are shown below in Table 3.13. All of the values are above .10, with most being even larger. This test is not adjusted for the number of comparisons being made, since the results are already not significant.

Table 3.13: Significance Testing of Participant Preferences between Models. Using a binomial distribution, the net preference scores for the survey were tested for significance. The unadjusted p-values for each are given below.

Net Votes Row > Column	Self Play	Pool Play	Fictitious Co-play	Rule-based Play
Self Play	-	.61	.50	.25
Pool Play	.61	-	.19	.34
Fictitious Co-play	.50	.19	-	.13
Rule-based Play	.25	.34	.13	-

To check whether the survey responses are providing new information beyond recent performance, the responses can be compared to the game scores of the preceding games. Figure 3.6 shows boxplots of the distribution of scores when participants indicated that they preferred or did not prefer a particular agent. At left, preferences are compared by difference in scores to the previous agent. At right, only the most recent agents scores are used. In each case, the median score is higher when the recent agent scored well. Table 3.14 below shows the coefficients which result from fitting a mutinomial logistic regression for each score to predict the survey response. A positive relationship indicates that a higher score is related to a greater likelihood of preferring the current agent. The current score is highly correlated and significant when predicting a preference for the current agent, and also close to significant for predicting neutrality rather than a preference for the previous agent. The previous score is only slightly correlated, and the relationships are likely the result of randomness.

Figure 3.6: Preferred Agents by Score. The score of the most recently played agent, and the difference in score between the most recent and second most recent agent are each shown separated by whether the participant preferred the recent or previous agent in the survey.

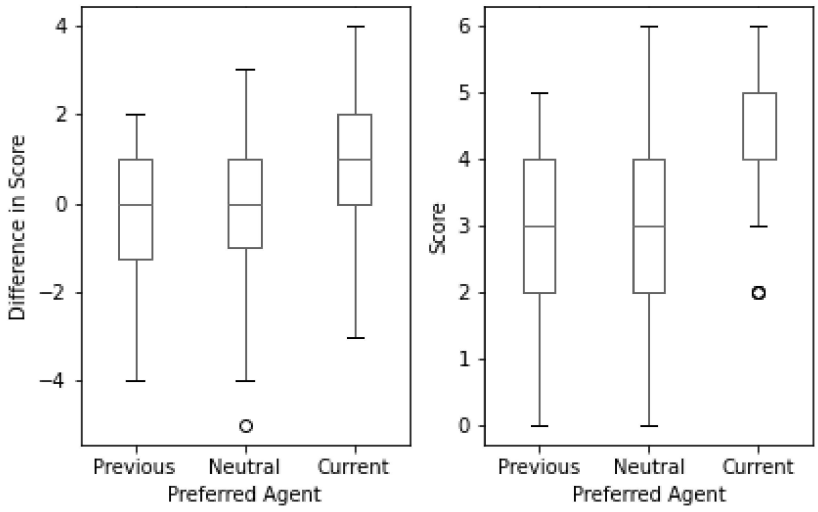


Table 3.14: Predictiveness of Scores for Survey Results. The scores of the most recent partner agent and the second most recent partner agent are used to predict the preference rating given in the survey. The most recent score was correlated with a preference towards the most recent agent and highly significant. The previous score was not a significant predictor.

	Survey Response			
	Neutral	p-value	Current	p-value
Constant	-1.15	-	-3.23	-
Current Score	.26	.11	.86	.00*
Previous Score	.22	.21	.05	.79

Finally, a multinomial logistic regression was fit to predict the survey scores based on the previously computed entropy scores which measure similarity between agent and human choices. A positive coefficient indicates that higher predictiveness results in higher likelihood of preferring the model. Separate regressions were fit for each model since the selection of preferences was based only on two models at a time. The scores of the round are also included as a control. Table 3.15 shows the coefficients and p-values. As before, the scores are significant, but the addition of the entropy measure is not statistically significant.

Table 3.15: Predictiveness of Entropy for Survey Results. Entropy scores from recent games are used to predict which agents are preferred in the survey. After controlling for recent scores, none of the results are significant.

		Survey Response			
		Neutral	p-value	Current	p-value
Self Play	Constant	-1.12	-	-2.26	-
	Cross Entropy	.65	.11	.40	.70
	Current Score	.44	.21	.21	.13
Pool Play	Constant	-0.49	-	-2.75	-
	Cross Entropy	.05	.94	1.00	.32
	Current Score	.27	.34	.63	.10
Fictitious Co-play	Constant	-1.06	-	-3.43	-
	Cross Entropy	.70	.44	-0.32	.74
	Current Score	.12	.78	1.10	.08
Rule-based Play	Constant	-1.23	-	-4.50	-
	Cross Entropy	-0.85	.40	.03	.97
	Current Score	.30	.49	1.20	.03*

4

Discussion

The main contributions of this thesis center around human-AI partnership, and teaching AI agents about human behaviour. First, a discussion of the scale of human-AI cooperation and the differences that occur depending on the nature of the environment in which the AI agents are expected to act. Secondly, motivation for focusing on 'Understanding' as a means of improving coordination between agents trained via reinforcement learning and humans. Finally, a new cooperative environment based on the popular card game Spades, which is available both as a automated environment and an online experimental platform for human experiments. Initial testing of agents in this environment is provided, which suggests that the new training method may be beneficial, but does not have the sample size required to be conclusive.

4.1 Principle Findings

As far as the testing, several questions are considered in turn based on the data collected. To begin with, each agent is benchmarked in games played with a random partner. This provides a baseline measure to establish the difference in skill between the trained agents and random ones, as well as a comparison to the improvements when human players are added. Table 3.3 shows these results. The win rates are generally in the range of 51%, slightly better than random. With the large number of games played, this difference is significant, though playing better than random is a rather low bar. Of note, each of the agents is of similar quality when playing with a random partner. As such, any later differences in performance should be considered as a function of the collaboration, not merely a difference in the baseline abilities of one of the agents. The second comparison, which has agents tested with a copy of themselves as their partner, outperforms the first, further emphasizing that the agents are better than random ones.

The trials with human participants, shown in Table 3.5, all outperformed the benchmark trials. For the two most complex approaches, Fictitious Co-Play and Rule-based Play, the comparison between human and benchmark play was statistically significant. For self play, the mean score was only modestly lower, and the p-value was still quite small. For pool play, the performance was much lower than the other models, though still better than the benchmarks. Although two of the models did not reach statistical significance, the interpretation is similar. In general, the human participants were better than the random partners in the benchmarking. This indicates that the human players were playing strategically and not simply at random trying to complete the experiment. Since participants were anonymous volunteers, there are also few incentives to complete the study other than interest, further suggesting that the results can be interpreted as coming from strategic play.

Comparing between the training methods, none of the differences are large enough to reach statistical significance. As such, nothing can be said conclusively about

their relative quality. Promisingly, the highest estimates for the means belong to the new rule-based play method and the previous top method, fictitious co-play. Based on this, the new method shows potential to be a new technique for training AI agents, and additional research should be done to verify this result with a larger sample.

An important assertion of the rule-based approach is that rules are actually useful for capturing human behaviour. Based on the entropy measurements of the agents, any one of the deep Q networks was more similar to human behaviour than random. When compared with the 'expected number of matching moves' metric, this difference disappears, and the neural models fall back in line with randomness. This is reflective of the difference between the entropy and expected value approaches. Entropy, as a logarithmic function, effectively takes into account the shape of the distribution of the expected values. While the means are similar, the number of matching moves chosen by the neural network agents is roughly normally distributed around the mean, while the random selection is much more spread out. The entropy approach penalizes this difference because it cares about similarity in distribution as well as mean.

More importantly, the rule-based partner had a substantially higher similarity to human moves than either the random play or the reinforcement learning agents. This provides evidence that rule-based agents can be designed which mimic human behaviour, potentially even based only on a simplistic set of principles. The rule-based partner does not capture the entirety of human play, but is remarkably accurate considering that the rules only apply in about 25% of tricks. After accounting for the final trick where the agents have no choice to make, the random uniform distribution has about 50% accuracy, matching 2.5 out of 5 tricks. This implies that the weighted average number of options when a choice is made is 2, since the random agent is correct half the time. Therefore, the 61% accuracy over 478 tricks is enough to clearly establish that the rule performs statistically better than random.

Even if the action choices are more similar to human behaviour, this does not necessarily imply that the agents have a better understanding of human partners. However, when the training partners behave similarly to humans, the agents have the closest thing to human experience possible. For the cases of self-play and rule-based play, the partners are tested directly, so the above results do indicate increased experience with human-like behaviours. For pool play and fictitious co-play, there are a variety of partners, so the measure captured is only a part of the total experience, although all of the non-tested agents are developed with the same methods and should not be expected to be more or less human-like.

The final major aspect of the experiment was the survey questions. Participants were asked to rate which agents they preferred to work with throughout the experiment. None of the comparisons were significant, and half of the people surveyed reported having no preference between them. To better understand the lack of response, I considered the extent to which responses were driven by recent performance, by the difference in performance between previous agents, and by similarity to human play. The most predictive factor was simply the performance of the agent in the previous game. Neither the performance in earlier games nor similarity to human play were statistically significant factors. This suggests that people who did respond to the survey with a preference were generally driven by events immediately prior. Since the earlier games do not have an effect, the comparisons may not be capturing much information at all about relative preferences. Instead, people seem to simply prefer agents who score well, and forget about agents they haven't played with recently.

4.2 Relation to Other Studies

The central idea in this study is that AI agents can better understand people if their training data is more representative of what people are like. This is followed

by the prediction that human-like behaviour might be achievable via means which minimize the actual collection of human data. The new rules-based method does not entirely remove the use of human data; developing rules requires observational experience of people. In the right circumstances, it does have the potential to dramatically reduce the amount of data relative to previous work which attempts to integrate human behaviour in reinforcement learning. The two most relevant previous studies are Carroll et al. 2019 and Strouse et al. 2021, which deal with introducing human data into multi-agent reinforcement learning environments, and collaborating without human data respectively. This paper falls somewhere between the two, attempting to keep human data in the reinforcement learning process while minimizing the amount required.

In Carroll et al. 2019, the introduction of human data is done using a form of imitation learning called behavioral cloning. The original paper focuses on cloning the behaviour of a single person, but a follow up, Knott et al. 2021, also shows the usefulness in having a variety of human partners. Including human-like training partners can be thought of as a form of targeted exploration, which believes that certain actions are more valuable, in the vein of ϵ-greedy search or Monte Carlo Tree Search. In a multi-agent environment, the other partners in the environment change the possible states to be explored. Rather than the usual balance between exploration and exploitation, including human-like partners shifts the balance between different areas of exploration. Rather than a random distribution, the idea is to move towards a more targeted distribution. Carroll et al. 2019 use imitation learning to generate training partners which create this targeted learning. The subsequent addition of multiple people to the training expands the range of actions which are seen in training, which is beneficial when there are a variety of approaches which would be common among people, but human actions are still a small subset of possible actions. Targeting training in this way may allow AI agents to learn behaviours that are more adaptive to human partners.

This study follows the concept of using human-like training partners to target the training data. The primary difference is in the approach to developing the agents. Imitation learning requires the collection of large amounts of human data to train the agent. By contrast, the rule-based approach allows a human to abstract the targeting to a chosen set of rules. The rules shift the task of synthesizing the relevant aspects of behaviour from the imitation learner onto the human. This likely means that the rules will require less data collection. The rules may also be better targeted, as an imitation learner will learn to imitate every part of the recorded behaviours, whereas a rule-based agent would only teach the behaviours chosen for encoding in the rules.

The second study, Strouse et al. 2021, learns to collaborate with human partners with no human data whatsoever. Their approach is fictitious co-play, which uses a wider variety of training partners drawn from checkpoints during the learning process. In comparison to Carroll et al. 2019, fictitious co-play does not make an attempt to model human behaviour. Their stated focus is on helping the agent to adapt to different levels of skill demonstrated by the human partner, and as such they meet this by introducing different levels of skill in the training partners. The addition of varied skill training partners follows the premise of wanting to guide training towards particular types of scenario, but avoids human experience by relying on variation from randomness. This implicitly assumes one of three things. Either low-skill AI agents are similar in behaviour to low-skill humans, neither one is critically different from randomness, or the low-skill AI agents will suffice by virtue of their sheer numbers and variety. By contrast, my approach expects that rules will do a better job representing human behaviour than either of these. The environment in question will be the deciding factor in which assumptions should be correct, with each likely to do well when the assumptions are met.

4.3 Practical Implications

The practical implications of this new methodology are largely speculative given the inconclusive results of this small study. The core targeting of human-like behaviour in training partners is broadly applicable to cooperative scenarios between humans and AI. Supposing that specific rule-based training partner method also works as expected, many of the same cooperative scenarios could be approached this way.

The entire process of rule-based training can be used in cases where the set of possible actions is much larger than the set of likely human actions. This training method can provide tailoring to human behaviour while minimizing the need to collect human training data. An example outside of games, some grocery stores are now using robots to patrol the aisles for spills. These bots have only a basic sense of collision avoidance, and often get in the way of shoppers by not adequately understanding the situation and moving. A cooperative bot could be trained with a set of rule-based shoppers to learn to choose navigational paths which let people move around more easily. Coordinating motion is a general category where people often follow rules, sometimes in written form like driving cars, but also simpler rules such as minimizing the total distance travelled. A route-planning method is likely to be necessary as a component of any embodied AI system, and coordinating motions with other agents in the environment must be a part of that.

Rule-based training may also apply in situations with a smaller action set if human behaviour is unlikely to match the behaviours learned by gradient descent. These are difficult to identify *a priori*, but may be found after training an agent via other methods if the agent itself acts in an inhuman manner. If so, the trained agent could continue training with a new rule-based partner in order to target cooperation skills after the basic task-oriented skills have been learned. This could occur in cases where humans are processing information which is difficult to encode into the environment, such as planning a shopping list. An AI which needs to be told all of the changing meal preferences is much less useful than one which can figure them

out as needed through discussion. In that example, the human side of the task is unlikely to be well-imitated by an AI, but could be built out using rules.

4.4 Limitations

This study has a few key weaknesses which need to be highlighted, centered mainly around the challenges of the training environment. The card game used in this experiment has a high degree of stochasticity. Randomness arises from a number of sources: the cards dealt during the testing, the training experiences of each agent, and the human participants' behaviours and preferences. Cards games by nature involve randomly dealing a set of cards. This initial deal can dramatically impact the game, with it being possible, albeit unlikely, to have deals which are entirely impossible to win. Because of this high degree of variance, the performance of the human-AI pairings displays a similarly high variance even though their opponents are quite weak. Combined with this, the randomness and hidden information in the environment mean that the agents are unlikely to be strictly better than one another. If, for example, two agents reached identical information states in their training, but the hidden information in each state was different, the learnings for that state will differ. Over large amounts of training this becomes less extreme, but there are likely still idiosyncratic differences due to the training experience. These differences increase the randomness present in the final results, since even in otherwise equivalent situations one agent may random have a better expectation of the hidden information remaining.

High amounts of randomness are particularly undermining for the survey, since it appears that a large portion of the decision was based on recent results. With the variance in any given trick being larger than the difference between models, not much information can be extracted from the surveys. A similar problem arises for the comparison in performance. In pre-experiment testing, a power analysis

was run predicting that a sample size of about 80 participants would be enough to differentiate between the two top methods and the two others. In the actual experiment, the estimated effect size was not as large as expected, so that the differences in means were not significant.

Variation in the human participants is mostly dealt with by randomizing treatments as much as possible. On average, the human players were clearly playing to win, but it is difficult to assess whether any particular person was playing well. Poor game scores may come from unlucky deals rather than bad play. These sorts of differences would be expected to be minimized since every player was paired with every agent. The order of the agents is also randomized to avoid any effects resulting from either increased experience playing this game, fatigue, or any other impacts of testing the agents in sequence.

The fact that all of the players were volunteers is also a potential issue for generalization. People who are interested in playing a card game with AIs are not necessarily representative of the general population. They may have higher than average experience with card games, which could make them more likely to recognize and use strategies like the ones used by the rule-based training partner. They likely have an interest in either card games or AI, or have a sympathy towards researchers seeking volunteers. This must necessarily be considered in balance with the problems arising in any other recruitment method, and does not jeopardize the validity of the experiment. Since the data collected here is most useful as an early demonstration of a concept, drawing broad conclusions would already be ill advised, but the sampling method limits the scope further.

4.5 Future Research

The concept of rule-based training partners raises a number of natural extensions which could be studied in future research. First and foremost, the results of this study

need to be expanded to a sample size which verifies that they are reliable. Based on the effect size in this study, finding an effect by running the same experiment at larger scale would require many thousands of participants. Instead, it would be simpler to apply the same method in environments with less noise in the final results. Once the principle was more firmly established, variations on the same idea could be tested.

Rule-based partners can also be considered as a type of synthetic data, created with a particular goal in mind. While this paper focuses on using rules to replicate human behaviour, it would also be possible to choose them to suit a variety of ends. If, for example, the most common human behaviours in some environment are sub-optimal, it may be better to teach an AI to cooperate with more ideal behaviour, and highlight to people when they could improve. This example is only likely to work for simple cases, as people might be willing to be taught a more efficient algorithm, but are unlikely to take moral coaching from an AI, and may find an AI trying to teach overly complex behaviours to be quite frustrating.

In the Carroll et al. 2019 study which tried to replicate human behaviour, follow ups determined that a variety of imitation agents yielded better results. Similarly, this study only uses one agent, which follows every rule in a small set. Since people are not monolithic, it could be better to build a family of agents which each follow some subset of the prepared rules. This would result in agents which have more variety in behaviour for relatively minimal additional cost.

Along the same line of thought, following deterministic rules may be too limiting, and result in a partner which is too rigid when humans make small variations around the predicted behaviour. To mitigate this, leaky rules could be used, which are follow with some probability ϵ, to re-introduce some variation. The rules could also follow a more Bayesian approach, producing probabilities on each action rather than being deterministic, which would allow the same sort of targeted exploration with flexibility.

References

Alamdari, Parand Alizadeh et al. (May 2022). "Be Considerate: Avoiding Negative Side Effects in Reinforcement Learning". en. In: *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems*, p. 9.

Almaatouq, Abdullah et al. (Oct. 2021). "Empirica: a virtual lab for high-throughput macro-level experiments". en. In: *Behavior Research Methods* 53.5, pp. 2158–2171. URL: https://doi.org/10.3758/s13428-020-01535-9 (visited on 08/01/2022).

Ashktorab, Zahra et al. (Oct. 2020). "Human-AI Collaboration in a Cooperative Game Setting: Measuring Social Perception and Outcomes". In: *Proceedings of the ACM on Human-Computer Interaction* 4.CSCW2, 96:1–96:20. URL: https://doi.org/10.1145/3415167 (visited on 05/04/2022).

Awad, Edmond et al. (Nov. 2018). "The Moral Machine experiment". en. In: *Nature* 563.7729. Number: 7729 Publisher: Nature Publishing Group, pp. 59–64. URL: https://www.nature.com/articles/s41586-018-0637-6 (visited on 05/07/2022).

Balachandar, Niranjan, Justin Dieter, and Govardana Sachithanandam Ramachandran (June 2019). "Collaboration of AI Agents via Cooperative Multi-Agent Deep Reinforcement Learning". In: *arXiv:1907.00327 [cs]*. arXiv: 1907.00327. URL: http://arxiv.org/abs/1907.00327 (visited on 05/04/2022).

Baraniuk, Chris (2015). *The cyborg chess players that can't be beaten*. en. URL: https://www.bbc.com/future/article/20151201-the-cyborg-chess-players-that-cant-be-beaten (visited on 03/20/2022).

Barbierato, Enrico and Maria Enrica Zamponi (June 2022). "Shifting Perspectives on AI Evaluation: The Increasing Role of Ethics in Cooperation". en. In: *AI* 3.2. Number: 2

Publisher: Multidisciplinary Digital Publishing Institute, pp. 331–352. URL: https://www.mdpi.com/2673-2688/3/2/21 (visited on 05/03/2022).

Battistoni, Pietro et al. (2021). "A Change in Perspective About Artificial Intelligence Interactive Systems Design: Human Centric, Yes, But Not Limited to". en. In: *HCI International 2021 - Late Breaking Papers: Multimodality, eXtended Reality, and Artificial Intelligence*. Ed. by Constantine Stephanidis et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 381–390.

Baumann, Tobias (Feb. 2022). "Cooperative Artificial Intelligence". In: *arXiv:2202.09859 [cs]*. arXiv: 2202.09859. URL: http://arxiv.org/abs/2202.09859 (visited on 05/04/2022).

Bennett, Casey et al. (Feb. 2022). "Exploring Data-Driven Components of Socially Intelligent AI through Cooperative Game Paradigms". en. In: *Multimodal Technologies and Interaction* 6.2. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 16. URL: https://www.mdpi.com/2414-4088/6/2/16 (visited on 05/04/2022).

Boggess, Kayla, Sarit Kraus, and Lu Feng (Apr. 2022). "Toward Policy Explanations for Multi-Agent Reinforcement Learning". In: *arXiv:2204.12568 [cs]*. arXiv: 2204.12568. URL: http://arxiv.org/abs/2204.12568 (visited on 05/04/2022).

Bojarski, Mariusz et al. (Apr. 2016). *End to End Learning for Self-Driving Cars*. arXiv:1604.07316 [cs]. URL: http://arxiv.org/abs/1604.07316 (visited on 08/11/2022).

Brockman, Greg et al. (June 2016). *OpenAI Gym*. arXiv:1606.01540 [cs]. URL: http://arxiv.org/abs/1606.01540 (visited on 08/04/2022).

Brown, Noam and Tuomas Sandholm (2017). *Superhuman AI for heads-up no-limit poker: Libratus beats top professionals*. URL: https://www.science.org/doi/10.1126/science.aao1733 (visited on 03/20/2022).

Carroll, Micah et al. (2019). "On the Utility of Learning about Humans for Human-AI Coordination". In: *Advances in Neural Information Processing Systems*. Vol. 32.

Curran Associates, Inc. URL: https://papers.nips.cc/paper/2019/hash/f5b1b89d98b7286673128a5fb112cb9a-Abstract.html (visited on 12/15/2021).

Chelarescu, Paul (June 2021). "Deception in Social Learning: A Multi-Agent Reinforcement Learning Perspective". In: *arXiv:2106.05402 [cs]*. arXiv: 2106.05402. URL: http://arxiv.org/abs/2106.05402 (visited on 05/04/2022).

Cila, Nazli (Apr. 2022). "Designing Human-Agent Collaborations: Commitment, responsiveness, and support". In: *CHI Conference on Human Factors in Computing Systems*. CHI '22. New York, NY, USA: Association for Computing Machinery, pp. 1–18. URL: https://doi.org/10.1145/3491102.3517500 (visited on 05/03/2022).

Cila, Nazli et al. (May 2017). "Products as Agents: Metaphors for Designing the Products of the IoT Age". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. New York, NY, USA: Association for Computing Machinery, pp. 448–459. URL: https://doi.org/10.1145/3025453.3025797 (visited on 05/04/2022).

Claus, C. and Craig Boutilier (1998). "The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems". en. In: *undefined*. URL: https://www.semanticscholar.org/paper/The-Dynamics-of-Reinforcement-Learning-in-Systems-Claus-Boutilier/38d35d5581e58dca4e9458501e65c1f85ca754d5 (visited on 07/20/2022).

Cohensius, Gal et al. (Feb. 2020). "Bidding in Spades". In: *arXiv:1912.11323 [cs]*. arXiv: 1912.11323. URL: http://arxiv.org/abs/1912.11323 (visited on 12/19/2021).

Dafoe, Allan et al. (Dec. 2020). "Open Problems in Cooperative AI". In: *arXiv:2012.08630 [cs]*. arXiv: 2012.08630. URL: http://arxiv.org/abs/2012.08630 (visited on 12/14/2021).

De Peuter, Sebastiaan, Antti Oulasvirta, and Samuel Kaski (July 2021). "Toward AI Assistants That Let Designers Design". In: *arXiv:2107.13074 [cs]*. arXiv: 2107.13074. URL: http://arxiv.org/abs/2107.13074 (visited on 05/03/2022).

Diamond, Arthur M. (Mar. 2020). "Robots and Computers Enhance Us More Than They Replace Us". en. In: *The American Economist* 65.1. Publisher: SAGE Publications Inc, pp. 4–10. URL: https://doi.org/10.1177/0569434518792674 (visited on 06/01/2022).

DiGiovanni, Anthony and Jesse Clifton (Apr. 2022). "Commitment games with conditional information revelation". In: *arXiv:2204.03484 [cs]*. arXiv: 2204.03484. URL: http://arxiv.org/abs/2204.03484 (visited on 05/04/2022).

Doebeli, Michael and Christoph Hauert (2005). "Models of cooperation based on the Prisoner's Dilemma and the Snowdrift game". en. In: *Ecology Letters* 8.7. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1461-0248.2005.00773.x, pp. 748–766. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1461-0248.2005.00773.x (visited on 05/07/2022).

Dragan, Anca D. et al. (Mar. 2015). "Effects of Robot Motion on Human-Robot Collaboration". In: *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. ISSN: 2167-2121, pp. 51–58.

Evans, Owain et al. (Oct. 2021). "Truthful AI: Developing and governing AI that does not lie". In: *arXiv:2110.06674 [cs]*. arXiv: 2110.06674. URL: http://arxiv.org/abs/2110.06674 (visited on 05/04/2022).

Feng, Shi and Jordan Boyd-Graber (Mar. 2019). "What can AI do for me? evaluating machine learning interpretations in cooperative play". In: *Proceedings of the 24th International Conference on Intelligent User Interfaces*. IUI '19. New York, NY, USA: Association for Computing Machinery, pp. 229–239. URL: https://doi.org/10.1145/3301275.3302265 (visited on 05/04/2022).

Foerster, Jakob N. et al. (Sept. 2019). "Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning". In: *arXiv:1811.01458 [cs]*. arXiv: 1811.01458. URL: http://arxiv.org/abs/1811.01458 (visited on 12/15/2021).

Fontaine, Matthew C. et al. (2021). "On the Importance of Environments in Human-Robot Coordination". In: *35th Conference on Neural Information Processing Systems*. Sydney, Australia. URL: https://drive.google.com/file/d/1PWNMMsS6ejCZpk6uVgCETFvuRNvqYpY5/view?usp=embed_facebook (visited on 12/15/2021).

Fujimoto, Ted and Arthur Paul Pedersen (Mar. 2022). "Adversarial Attacks in Cooperative AI". In: *arXiv:2111.14833 [cs]*. arXiv: 2111.14833. URL: http://arxiv.org/abs/2111.14833 (visited on 05/04/2022).

Gero, Katy Ilonka et al. (Apr. 2020). "Mental Models of AI Agents in a Cooperative Game Setting". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, pp. 1–12. URL: https://doi.org/10.1145/3313831.3376316 (visited on 05/04/2022).

Ghosh, Ahana et al. (May 2020). "Towards Deployment of Robust Cooperative AI Agents: An Algorithmic Framework for Learning Adaptive Policies". en. In: Auckland (NZ); Virtual (Covid), pp. 447–455. URL: http://eprints.cs.univie.ac.at/6587/ (visited on 05/04/2022).

Ghoshal, Biraja et al. (2021). "Estimating uncertainty in deep learning for reporting confidence to clinicians in medical image segmentation and diseases detection". en. In: *Computational Intelligence* 37.2. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/coin.12411, pp. 701–734. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12411 (visited on 06/01/2022).

Goodman, Noah D. et al. (2008). "A Rational Analysis of Rule-Based Concept Learning". en. In: *Cognitive Science* 32.1. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1080/03640210701802071, pp. 108–154. URL: https://onlinelibrary.wiley.com/doi/abs/10.1080/03640210701802071 (visited on 07/01/2022).

Hamada, Hiro Taiyo and Ryota Kanai (Feb. 2022). “AI agents for facilitating social interactions and wellbeing”. In: *arXiv:2203.06244 [cs]*. arXiv: 2203.06244. URL: http://arxiv.org/abs/2203.06244 (visited on 05/04/2022).

Hasselt, Hado van, Arthur Guez, and David Silver (Mar. 2016). “Deep Reinforcement Learning with Double Q-Learning”. en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 30.1. Number: 1. URL: https://ojs.aaai.org/index.php/AAAI/article/view/10295 (visited on 07/22/2022).

Heinrich, Johannes, Marc Lanctot, and David Silver (June 2015). “Fictitious Self-Play in Extensive-Form Games”. en. In: *Proceedings of the 32nd International Conference on Machine Learning*. ISSN: 1938-7228. PMLR, pp. 805–813. URL: https://proceedings.mlr.press/v37/heinrich15.html (visited on 07/30/2022).

Hu, Hengyuan et al. (May 2021). “"Other-Play" for Zero-Shot Coordination”. In: *arXiv:2003.02979 [cs]*. arXiv: 2003.02979. URL: http://arxiv.org/abs/2003.02979 (visited on 12/15/2021).

Jovanović, Mlađan and Mia Schmitz (Feb. 2022). “Explainability as a User Requirement for Artificial Intelligence Systems”. In: *Computer* 55.2. Conference Name: Computer, pp. 90–94.

Julien Dossa, Rousslan Fernand et al. (July 2019). “A Human-Like Agent Based on a Hybrid of Reinforcement and Imitation Learning”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. ISSN: 2161-4407, pp. 1–8.

Kabudi, Tumaini, Ilias Pappas, and Dag Håkon Olsen (Jan. 2021). “AI-enabled adaptive learning systems: A systematic mapping of the literature”. en. In: *Computers and Education: Artificial Intelligence* 2, p. 100017. URL: https://www.sciencedirect.com/science/article/pii/S2666920X21000114 (visited on 05/07/2022).

Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra (May 1998). “Planning and acting in partially observable stochastic domains”. en. In: *Artificial Intelligence* 101.1, pp. 99–134. URL:

https://www.sciencedirect.com/science/article/pii/S000437029800023X (visited on 07/20/2022).

Kiran, B Ravi et al. (June 2022). "Deep Reinforcement Learning for Autonomous Driving: A Survey". In: *IEEE Transactions on Intelligent Transportation Systems* 23.6. Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 4909–4926.

Knott, Paul et al. (May 2021). "Evaluating the Robustness of Collaborative Agents". In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '21. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 1560–1562. (Visited on 07/30/2022).

Kopparapu, Kavya et al. (Jan. 2022). "Hidden Agenda: a Social Deduction Game with Diverse Learned Equilibria". In: *arXiv:2201.01816 [cs]*. arXiv: 2201.01816. URL: http://arxiv.org/abs/2201.01816 (visited on 05/04/2022).

Koster, Raphael et al. (Jan. 2022). "Human-centered mechanism design with Democratic AI". In: *arXiv:2201.11441 [cs, econ, q-fin]*. arXiv: 2201.11441. URL: http://arxiv.org/abs/2201.11441 (visited on 05/04/2022).

Lanctot, Marc et al. (Nov. 2017). *A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning*. arXiv:1711.00832 [cs]. URL: http://arxiv.org/abs/1711.00832 (visited on 07/30/2022).

Levitt, Steven D., John A. List, and David H. Reiley (2010). "What Happens in the Field Stays in the Field: Exploring Whether Professionals Play Minimax in Laboratory Experiments". en. In: *Econometrica* 78.4. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.3982/ECTA7405, pp. 1413–1434. URL: https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA7405 (visited on 07/01/2022).

Liang, Claire et al. (May 2019). "Implicit Communication of Actionable Information in Human-AI teams". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. New York, NY, USA: Association for Computing

Machinery, pp. 1–13. URL: https://doi.org/10.1145/3290605.3300325 (visited on 05/25/2022).

Liebowitz, Jay (July 2019). *The Handbook of Applied Expert Systems*. en. Google-Books-ID: WzX3DwAAQBAJ. CRC Press.

Lo, Yat Long and Biswa Sengupta (Mar. 2022). "Learning to Ground Decentralized Multi-Agent Communication with Contrastive Learning". In: *arXiv:2203.03344 [cs]*. arXiv: 2203.03344. URL: http://arxiv.org/abs/2203.03344 (visited on 05/04/2022).

Ma, Mingwei et al. (Jan. 2022). "Learning to Coordinate with Humans using Action Features". In: *arXiv:2201.12658 [cs]*. arXiv: 2201.12658. URL: http://arxiv.org/abs/2201.12658 (visited on 05/06/2022).

Macy, Michael W. and Andreas Flache (May 2002). "Learning dynamics in social dilemmas". In: *Proceedings of the National Academy of Sciences* 99.suppl_3. Publisher: Proceedings of the National Academy of Sciences, pp. 7229–7236. URL: https://www.pnas.org/doi/abs/10.1073/pnas.092080099 (visited on 05/07/2022).

McAleer, Stephen et al. (June 2021). "Improving Social Welfare While Preserving Autonomy via a Pareto Mediator". In: *arXiv:2106.03927 [cs]*. arXiv: 2106.03927. URL: http://arxiv.org/abs/2106.03927 (visited on 05/04/2022).

McKee, Kevin R., Xuechunzi Bai, and Susan T. Fiske (Jan. 2022). "Warmth and competence in human-agent cooperation". In: *arXiv:2201.13448 [cs]*. arXiv: 2201.13448. URL: http://arxiv.org/abs/2201.13448 (visited on 05/04/2022).

McKee, Kevin R., Joel Z. Leibo, et al. (Mar. 2022). "Quantifying the effects of environment and population diversity in multi-agent reinforcement learning". en. In: *Autonomous Agents and Multi-Agent Systems* 36.1, p. 21. URL: https://doi.org/10.1007/s10458-022-09548-8 (visited on 05/04/2022).

Mnih, Volodymyr et al. (Feb. 2015). "Human-level control through deep reinforcement learning". en. In: *Nature* 518.7540. Number: 7540 Publisher: Nature Publishing

Group, pp. 529–533. URL: https://www.nature.com/articles/nature14236 (visited on 07/21/2022).

Müller, Luise (Feb. 2022). "Domesticating Artificial Intelligence". en. In: *Moral Philosophy and Politics*. Publisher: De Gruyter. URL: https://www.degruyter.com/document/doi/10.1515/mopp-2020-0054/html (visited on 05/04/2022).

Niklaus, Joel et al. (June 2019). "Survey of Artificial Intelligence for Card Games and Its Application to the Swiss Game Jass". In: *2019 6th Swiss Conference on Data Science (SDS)*, pp. 25–30.

Nosofsky, Robert M., Thomas J. Palmeri, and Stephen C. McKinley (1994). "Rule-plus-exception model of classification learning". In: *Psychological Review* 101.1. Place: US Publisher: American Psychological Association, pp. 53–79.

Onnasch, Linda and Eileen Roesler (July 2021). "A Taxonomy to Structure and Analyze Human–Robot Interaction". en. In: *International Journal of Social Robotics* 13.4, pp. 833–849. URL: https://doi.org/10.1007/s12369-020-00666-5 (visited on 02/10/2022).

Palacios-Huerta, Ignacio and Oscar Volij (2008). "Experientia Docet: Professionals Play Minimax in Laboratory Experiments". en. In: *Econometrica* 76.1. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.0012-9682.2008.00818.x, pp. 71–115. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.0012-9682.2008.00818.x (visited on 07/17/2022).

Puterman, Martin L. (1994). *Markov Decision Processes*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470316887 (visited on 07/20/2022).

Qin, Rong-Jun, Jing-Cheng Pang, and Yang Yu (Nov. 2019). *Improving Fictitious Play Reinforcement Learning with Expanding Models*. arXiv:1911.11928 [cs, stat]. URL: http://arxiv.org/abs/1911.11928 (visited on 07/30/2022).

Rabi, Rahel, Sarah J. Miles, and John Paul Minda (Mar. 2015). "Learning categories via rules and similarity: Comparing adults and children". en. In: *Journal of Experimental Child Psychology* 131, pp. 149–169. URL: https://www.sciencedirect.com/science/article/pii/S0022096514002008 (visited on 07/01/2022).

Radke, David, Kate Larson, and Tim Brecht (Apr. 2022). "The Importance of Credo in Multiagent Learning". In: *arXiv:2204.07471 [cs]*. arXiv: 2204.07471. URL: http://arxiv.org/abs/2204.07471 (visited on 05/04/2022).

Ritz, Fabian et al. (July 2021). "A Sustainable Ecosystem through Emergent Cooperation in Multi-Agent Reinforcement Learning". en. In: MIT Press. URL: https://direct.mit.edu/isal/article/doi/10.1162/isal_a_00399/102891/A-Sustainable-Ecosystem-through-Emergent (visited on 05/04/2022).

Rong, Jiang, Tao Qin, and Bo An (Mar. 2019). "Competitive Bridge Bidding with Deep Neural Networks". In: *arXiv:1903.00900 [cs]*. arXiv: 1903.00900. URL: http://arxiv.org/abs/1903.00900 (visited on 12/19/2021).

Searle, John R. (Sept. 1980). "Minds, brains, and programs". en. In: *Behavioral and Brain Sciences* 3.3. Publisher: Cambridge University Press, pp. 417–424. URL: https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/abs/minds-brains-and-programs/DC644B47A4299C637C89772FACC2706A (visited on 04/18/2022).

Seirawan, Yasser, Herbert A. Simon, and Toshinori Munakata (Aug. 1997). "The implications of Kasparov vs. Deep Blue". In: *Communications of the ACM* 40.8, pp. 21–25. URL: https://doi.org/10.1145/257874.257878 (visited on 12/16/2021).

Shannon, C. E. (July 1948). "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3. Conference Name: The Bell System Technical Journal, pp. 379–423.

Shirado, Hirokazu and Nicholas A. Christakis (May 2017). "Locally noisy autonomous agents improve global human coordination in network experiments". en. In: *Nature*

545.7654. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 7654 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Human behaviour;Network topology;Social evolution;Sociology Subject_term_id: human-behaviour;network-topology;social-evolution;sociology, pp. 370–374. URL: https://www.nature.com/articles/nature22332 (visited on 12/15/2021).

Silver, David et al. (2018). "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419. _eprint: https://www.science.org/doi/pdf/10.1126/science.aar6404, pp. 1140–1144. URL: https://www.science.org/doi/abs/10.1126/science.aar6404.

Stastny, Julian et al. (Nov. 2021). "Normative Disagreement as a Challenge for Cooperative AI". In: *arXiv:2111.13872 [cs]*. arXiv: 2111.13872. URL: http://arxiv.org/abs/2111.13872 (visited on 05/04/2022).

Steiner, David F. et al. (Dec. 2018). "Impact of Deep Learning Assistance on the Histopathologic Review of Lymph Nodes for Metastatic Breast Cancer". eng. In: *The American Journal of Surgical Pathology* 42.12, pp. 1636–1646.

Straitouri, Eleni et al. (Sept. 2021). "Reinforcement Learning Under Algorithmic Triage". In: *arXiv:2109.11328 [cs]*. arXiv: 2109.11328. URL: http://arxiv.org/abs/2109.11328 (visited on 05/03/2022).

Strouse, DJ et al. (2021). "Collaborating with Humans without Human Data". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 14502–14515. URL: https://proceedings.neurips.cc/paper/2021/hash/797134c3e42371bb4979a462eb2f042a-Abstract.html (visited on 05/04/2022).

Suh, Jaeyoung et al. (Aug. 2021). "Development of Speech Dialogue Systems for Social AI in Cooperative Game Environments". In: *2021 IEEE Region 10 Symposium (TENSYMP)*. ISSN: 2642-6102, pp. 1–4.

Swamy, Gokul et al. (Jan. 2019). *On the Utility of Model Learning in HRI*. Tech. rep. arXiv:1901.01291. arXiv:1901.01291 [cs, stat] version: 1 type: article. arXiv. URL: http://arxiv.org/abs/1901.01291 (visited on 06/01/2022).

Teófilo, Luís et al. (2014). "Rule based strategies for large extensive-form games: A specification language for No-Limit Texas Hold'em agents". en. In: *Computer Science and Information Systems* 11.4, pp. 1249–1269. URL: http://www.doiserbia.nb.rs/Article.aspx?ID=1820-02141400029T (visited on 07/01/2022).

Terry, J et al. (2021). "PettingZoo: Gym for Multi-Agent Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 15032–15043. URL: https://papers.neurips.cc/paper/2021/hash/7ed2d3454c5eea71148b11d0c25104ff-Abstract.html (visited on 08/04/2022).

Traeger, Margaret L. et al. (Mar. 2020). "Vulnerable robots positively shape human conversational dynamics in a human–robot team". en. In: *Proceedings of the National Academy of Sciences* 117.12. ISBN: 9781910402115 Publisher: National Academy of Sciences Section: Physical Sciences, pp. 6370–6375. URL: https://www.pnas.org/content/117/12/6370 (visited on 12/15/2021).

Tylkin, Paul, Goran Radanovic, and David C Parkes (2021). "Learning Robust Helpful Behaviors in Two-Player Cooperative Atari Environments". en. In: *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems*, p. 3.

Wang, Yutong and Guillaume Sartoretti (Jan. 2022). "FCMNet: Full Communication Memory Net for Team-Level Cooperation in Multi-Agent Systems". In: *arXiv:2201.11994 [cs]*. arXiv: 2201.11994. URL: http://arxiv.org/abs/2201.11994 (visited on 05/08/2022).

Wang, Ziyu et al. (Apr. 2016). *Dueling Network Architectures for Deep Reinforcement Learning*. arXiv:1511.06581 [cs]. URL: http://arxiv.org/abs/1511.06581 (visited on 07/22/2022).

Watkins, Christopher J. C. H. and Peter Dayan (May 1992). "Q-learning". en. In: *Machine Learning* 8.3, pp. 279–292. URL: https://doi.org/10.1007/BF00992698 (visited on 07/20/2022).

Wilder, Bryan, Eric Horvitz, and Ece Kamar (May 2020). *Learning to Complement Humans*. Tech. rep. arXiv:2005.00582. arXiv:2005.00582 [cs] type: article. arXiv. URL: http://arxiv.org/abs/2005.00582 (visited on 05/25/2022).

Wooders, John (2010). "Does Experience Teach? Professionals and Minimax Play in the Lab". en. In: *Econometrica* 78.3. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.3982/ECTA7970, pp. 1143–1154. URL: https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA7970 (visited on 07/17/2022).

Wu, Jie, Qiufang Fu, and Michael Rose (Feb. 2020). "Stimulus modality influences the acquisition and use of the rule-based strategy and the similarity-based strategy in category learning". en. In: *Neurobiology of Learning and Memory* 168, p. 107152. URL: https://www.sciencedirect.com/science/article/pii/S1074742719302199 (visited on 07/01/2022).

Wu, Yonghui et al. (Oct. 2016). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. arXiv:1609.08144 [cs]. URL: http://arxiv.org/abs/1609.08144 (visited on 08/13/2022).

Xiong, Wei et al. (Mar. 2022). "Challenges of human—machine collaboration in risky decision-making". en. In: *Frontiers of Engineering Management* 9.1, pp. 89–103. URL: https://doi.org/10.1007/s42524-021-0182-0 (visited on 05/03/2022).

Yang, Jiachen (Dec. 2021). "Cooperation in Multi-Agent Reinforcement Learning". In: *Georgia Tech Theses and Dissertations*. Accepted: 2022-01-14T16:11:34Z Publisher: Georgia Institute of Technology. URL: http://hdl.handle.net/1853/66146 (visited on 05/04/2022).

Zhang, Qiaoning, Matthew L Lee, and Scott Carter (Apr. 2022). "You Complete Me: Human-AI Teams and Complementary Expertise". In: *CHI Conference on Human Factors in Computing Systems*. CHI '22. New York, NY, USA: Association for Computing Machinery, pp. 1–28. URL: https://doi.org/10.1145/3491102.3517791 (visited on 06/01/2022).

Zhang, Shubao (Dec. 2021). "Towards Controllable Agent in MOBA Games with Generative Modeling". In: *arXiv:2112.08093 [cs]*. arXiv: 2112.08093. URL: http://arxiv.org/abs/2112.08093 (visited on 05/04/2022).

Zhou, Jiaying et al. (June 2021). "Assisted Learning: Cooperative AI with Autonomy". In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X, pp. 3130–3134.

Appendices

A

Gameplay Images

The following pages provide additional documentation of the experimental design through images of the website. Figure A.1 shows the annotated images used with the instructions of how scoring works. Figure A.2 is a question from the comprehension check quiz. Figure A.3 is an image of the view presented to participants when they need to choose an action.

Figure A.1: Scoring Instructions. In the example below, the East player went first, so the highest heart wins. However, Spades always trump any other suit, so West wins instead. This example is used to explain how scoring and trump suits work.

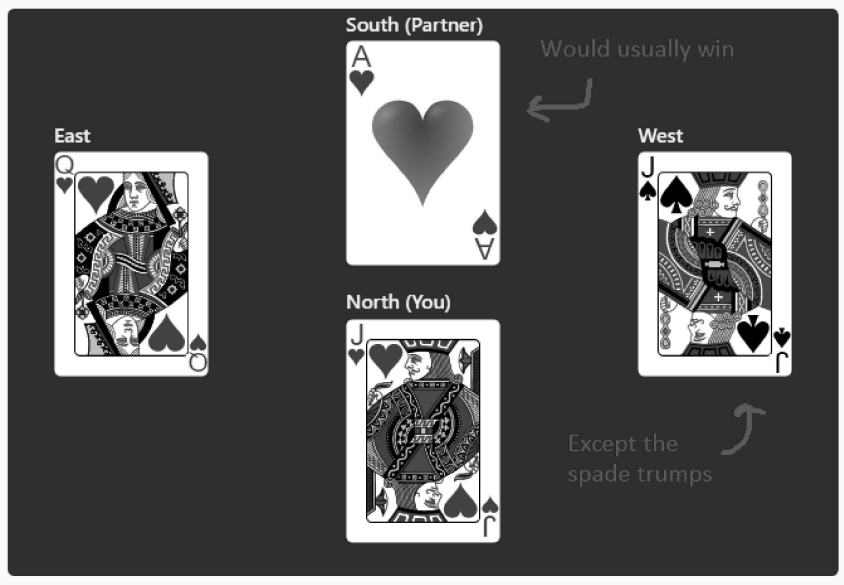


Figure A.2: Example Quiz Question. This question checks that participants know which card is highest, and that high cards win. The later questions become more complicated.

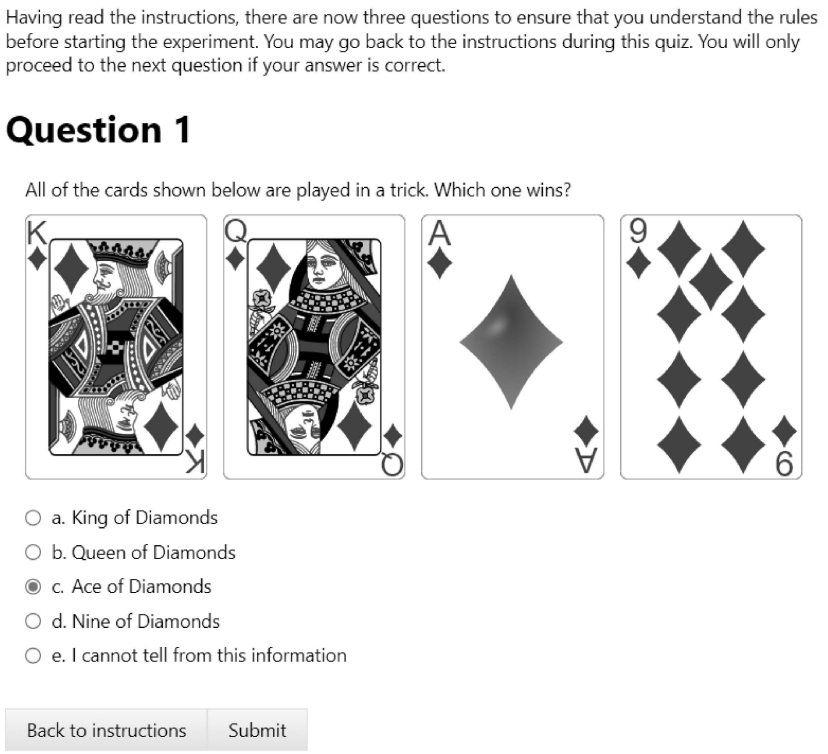
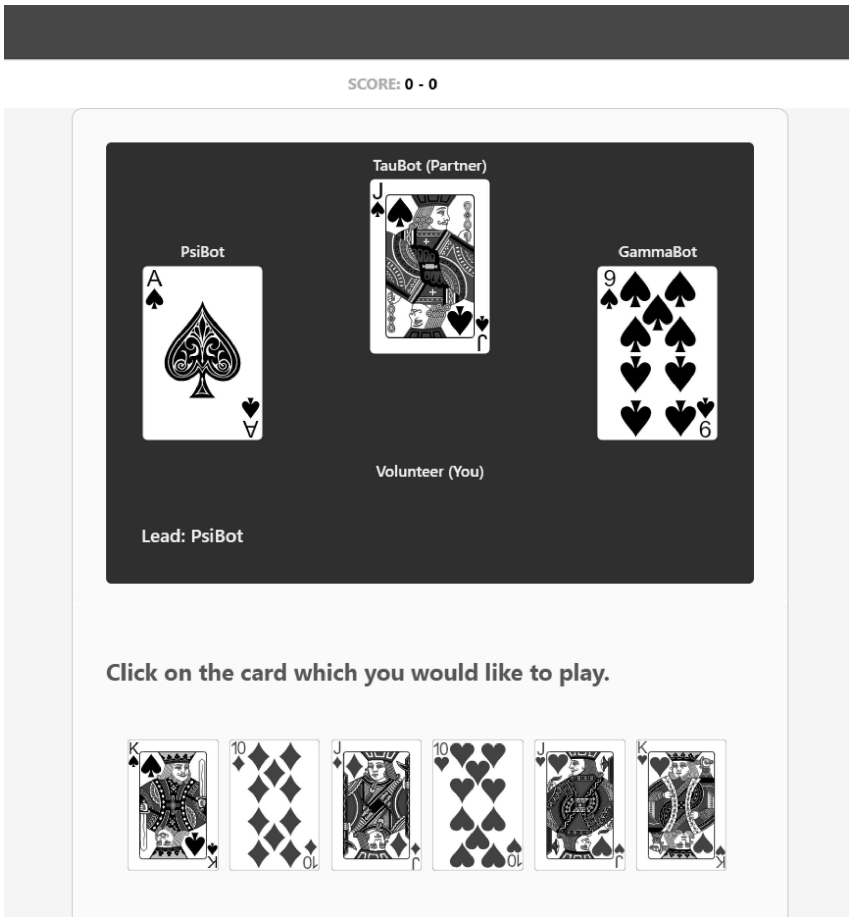


Figure A.3: Example Gameplay. Players are shown an image of the game board on which to play cards, and any previous cards played by other agents. The hand of available cards is at the bottom, and cards are selected by clicking on them. If the rules prohibit a card from being played, a pop-up will instead warn the player and explain the rules before allowing them to choose a different card.



B

Code Excerpts

The following sections provide an excerpt from the overall codebase used in this thesis, reduced in size to minimize the length of the final product. The Python files highlight the environment and the agent designs, and are standalone files. The JavaScript files show the basics of the interface but would not run without the remainder of the app. The totality of the code will be available on GitHub.

B.1 Python

The Python code is used for environment construction and model training.

B.1.1 Environment

The following code is an environment following the standard created by OpenAI's Gym.

```
from gym.spaces import MultiBinary, Discrete, Dict
import numpy as np
import functools
from pettingzoo import AECEnv
from pettingzoo.utils import agent_selector
from pettingzoo.utils import wrappers

N_CARDS = 24
N_AGENTS = 4
N_ITERS = N_CARDS // N_AGENTS

TRUMP = 'S'
SUITS = ['S', 'C', 'D', 'H']
RANKS = ['9', '10', 'J', 'Q', 'K', 'A']
NONE = np.zeros([1, N_CARDS])

SUIT_LOOKUP = {}
SUIT_REVERSE = {}
for i, suit in enumerate(SUITS):
    SUIT_LOOKUP[suit] = i
    SUIT_REVERSE[i] = suit

RANK_LOOKUP = {}
RANK_REVERSE = {}
for i, rank in enumerate(RANKS):
    RANK_LOOKUP[rank] = i
    RANK_REVERSE[i] = rank


```

```
30 def env():
31     """
32     The env function often wraps the environment in wrappers by ↩
       default.
33     You can find full documentation for these methods
34     elsewhere in the developer documentation.
35     """
36     env = raw_env()
37     # This wrapper is only for environments which write results ↩
       to the terminal
38     env = wrappers.CaptureStdoutWrapper(env)
39     # Provides a wide vareity of helpful user errors
40     # Strongly recommended
41     env = wrappers.OrderEnforcingWrapper(env)
42     return env
43
44
45 class raw_env(AECEnv):
46     """
47     The metadata holds environment constants. From gym, we ↩
       inherit the "render_modes",
48     metadata which specifies which modes can be put into the ↩
       render() method.
49     At least human mode should be supported.
50     The "name" metadata allows the environment to be pretty ↩
       printed.
51     """
52
53     metadata = {"render_modes": ["human"], "name": "cards"}
54
55     def __init__(self):
56         """
57         The init method takes in environment arguments and
```

```
58          should define the following attributes:
59          - possible_agents
60          - action_spaces
61          - observation_spaces
62
63          These attributes should not be changed after initialization.
64          """
65          self.state_seed = None
66          self.possible_agents = ["player_" + str(r) for r in range(N_AGENTS)]
67          self.partners = {"player_" + str(int(r + N_AGENTS / 2) % N_AGENTS): "player_" + str(r) for r in range(N_AGENTS)}
68          self.agent_name_mapping = dict(
69              zip(self.possible_agents, list(range(len(self.possible_agents))))
70          )
71
72          # Gym spaces are defined and documented here: https://gym.openai.com/docs/#spaces
73          self._action_spaces = {agent: Discrete(N_CARDS) for agent in self.possible_agents}
74          self._observation_spaces = {
75              agent: Dict(
76                  {"observation": MultiBinary([N_CARDS + 1, N_CARDS]),
77                   "action_mask": Discrete(N_CARDS),
78                   }
79              ) for agent in self.possible_agents
80          }
81
82      # this cache ensures that same space object is returned for the same agent
```

```
    # allows action space seeding to work as expected
    @functools.lru_cache(maxsize=None)
    def action_space(self, agent):
        # Gym spaces are defined and documented here: https://gym.openai.com/docs/#spaces
        return Discrete(N_CARDS)

    @functools.lru_cache(maxsize=None)
    def observation_space(self, agent):
        return Dict(
            {"observation": MultiBinary([N_CARDS + 1, N_CARDS]),
             "action_mask": MultiBinary(N_CARDS),
             }
        )

    def render(self, mode="human"):
        """
        Renders the environment. In human mode, it can write to terminal, open
        up a graphical window, or open up some other display that a human can see and understand.
        """
        if all(self.dones.values()):
            print("Game over")
        else:
            print(f"Current round: {self.current_round}")
            print(f"Tricks taken: {self.tricks_taken}")
            print(f"Current play: {self.cards_played}")
            print(f"Current hand: {self.hands[self.agent_selection]}")

    def observe(self, agent):
        """
```

```
112         Observe should return the observation of the specified ↩
    agent. This function
113         should return a sane observation (though not necessarily ↩
    the most up to date possible)
114         at any time after reset() is called.
115         """
116         # observation of one agent is the previous state of the ↩
    other
117         return self.observations[agent]
118
119     def close(self):
120         """
121         Close should release any graphical displays, subprocesses↩
    , network connections
122         or any other environment data which should not be kept ↩
    around after the
123         user is no longer using the environment.
124         """
125         pass
126
127     def seed(self, state_seed):
128         self.state_seed = state_seed
129
130     def reset(self, state_seed=None):
131         """
132         Reset needs to initialize the following attributes
133         - agents
134         - rewards
135         - _cumulative_rewards
136         - dones
137         - infos
138         - agent_selection
```

```
139         And must set up the environment so that render(), step(),↩
     and observe()
140         can be called without issues.
141
142         Here it sets up the state dictionary which is used by ↩
    step() and the observations dictionary which is used by step↩
    () and observe()
143         """
144         # Use the same seed for sets of four seats, but rotate ↩
    who goes first
145         if state_seed:
146             self.state_seed = state_seed
147             np.random.seed(self.state_seed - self.state_seed % 4)
148             first_index = self.state_seed % 4
149         elif self.state_seed:
150             np.random.seed(self.state_seed - self.state_seed % 4)
151             first_index = self.state_seed % 4
152         else:
153             first_index = np.random.choice([0, 1, 2, 3], 1)[0]
154         self.agents = self.possible_agents[first_index:]
155         self.agents.extend(self.possible_agents[:first_index])
156         self.rewards = {agent: 0 for agent in self.agents}
157         self._cumulative_rewards = {agent: 0 for agent in self.↩
    agents}
158         self.dones = {agent: False for agent in self.agents}
159         self.illegal_move = False
160
161         self.hands = {agent: {} for agent in self.agents}
162         self.private_info = {agent: [] for agent in self.agents}
163         self.private_observations = {agent: NONE.copy() for agent↩
     in self.agents}
164         self.__deal_cards()
165         self.public_info = []
```

```
166         self.public_observation = np.zeros([N_CARDS, N_CARDS])
167 
168         self.action_masks = {agent: self.private_observations[↩
    agent].flatten() for agent in self.agents}
169         self.infos = {agent: {'public': self.public_info, '↩
    private': self.private_info[agent]} for agent in self.agents}
170         self.state = {agent: NONE.copy() for agent in self.agents↩
    }
171 
172         self.observations = {
173             agent: {'observation': np.append(self.↩
    private_observations[agent], self.public_observation, axis=0)↩
    ,
174                     'action_mask': self.action_masks[agent]}
175             for agent in self.agents}
176         self.current_round = 0
177         self.tricks_taken = {agent: 0 for agent in self.agents}
178 
179         self.cards_played = {agent: '' for agent in self.agents}
180         self.has_lead = self.agents[0]
181         """
182         Our agent_selector utility allows easy cyclic stepping ↩
    through the agents list.
183         """
184         self._agent_selector = agent_selector(self.agents)
185         self.agent_selection = self._agent_selector.next()
186 
187         # Ensure relevant information is updated
188         for i in self.agents:
189             self.__update_action_mask(i)
190             self.__update_observation(i)
191             # obs = self.observe(i)
192         # tensorboardprint('finished reset')
```

```
193
194     def step(self, action):
195         """
196         step(action) takes in an action for the current agent (↩
    specified by
197         agent_selection) and needs to update
198         - rewards
199         - _cumulative_rewards (accumulating the rewards)
200         - dones
201         - infos
202         - agent_selection (to the next agent)
203         And any internal state used by observe() or render()
204         """
205
206         one_hot_action = NONE.copy()
207         one_hot_action[0, action] = 1
208         agent = self.agent_selection
209         this_round = self.current_round
210
211         if self.dones[self.agent_selection]:
212             # handles stepping an agent which is already done
213             # accepts a None action for the one agent, and moves ↩
    the agent_selection to
214             # the next done agent,  or if there are no more done ↩
    agents, to the next live agent
215             # Updates the results of the trick
216             if self._agent_selector.is_last() and not self.↩
    illegal_move:
217                 winning_agent = self.__choose_winner()
218                 # Increments the winner's tricks taken
219                 self.tricks_taken[winning_agent] += 1
220                 self.tricks_taken[self.partners[winning_agent]] ↩
    += 1
```

```
221                 self._accumulate_rewards()
222             return self._was_done_step(None)
223 
224         # Check for valid moves
225         if (sum(self.action_masks[agent]) == 0) and not self.↩
    dones[agent]:
226             print('No legal moves')
227             self.dones = {agent: True for agent in self.agents}
228             for i in self.agents:
229                 self.rewards[i] = 0
230             self.rewards[agent] = 0
231             self._accumulate_rewards()
232             return
233 
234         if self.action_masks[agent][action] == 0:
235             self.dones = {agent: True for agent in self.agents}
236             for i in self.agents:
237                 self.rewards[i] = self.tricks_taken[i] + self.↩
    current_round / 10
238             self.rewards[agent] = -10 + self.current_round
239             self._accumulate_rewards()
240             self.illegal_move = True
241             return
242 
243             # the agent which stepped last had its ↩
    _cumulative_rewards accounted for
244         # (because it was returned by last()), so the ↩
    _cumulative_rewards for this
245         # agent should start again at 0
246         # NOT SURE WHAT THIS STEP IS FOR
247         # self._cumulative_rewards[agent] = 0
248 
249         # stores action of current agent
```

```
250         self.state[self.agent_selection] = one_hot_action
251         self.cards_played[agent] = self.__decode_card(action)
252         self.hands[agent].remove(self.__decode_card(action))
253 
254         # collect reward if it is the last agent to act per cycle
255         if self._agent_selector.is_last():
256             # Updates the results of the trick
257             winning_agent = self.__choose_winner()
258             # Increments the winner's tricks taken
259             self.tricks_taken[winning_agent] += 1
260             self.tricks_taken[self.partners[winning_agent]] += 1
261             # Sets the lead to the winner
262             winning_index = self.possible_agents.index(↩
    winning_agent)
263             self.has_lead = winning_agent
264             self.agents = self.possible_agents[winning_index:]
265             self.agents.extend(self.possible_agents[:↩
    winning_index])
266             self._agent_selector.reinit(self.agents)
267             # Resets the cards played
268             self.cards_played = {i: '' for i in self.agents}
269 
270             self.current_round += 1
271             # rewards for all agents are placed in the .rewards ↩
    dictionary if in the final round
272             if self.current_round == N_ITERS:
273                 for i in self.agents:
274                     self.rewards[i] = self.tricks_taken[i]
275 
276                     # The dones dictionary must be updated for ↩
    all players.
277             self.dones = {i: self.current_round >= N_ITERS for i ↩
    in self.agents}
```

```
278
279         # observe the current state
280         self.public_info.append(self.cards_played)
281         observation_index = self.agent_name_mapping[agent] * ↩
    N_ITERS + this_round
282         self.public_observation[observation_index, action] = 1
283         self.private_observations[agent][0, action] = 0
284         for i in self.agents:
285             self.__update_observation(i)
286
287         for i in self.agents:
288             self.__update_action_mask(i)
289
290         # selects the next agent.
291         self.agent_selection = self._agent_selector.next()
292         # Adds .rewards to ._cumulative_rewards
293         self._accumulate_rewards()
294
295     def __encode_card(self, card):
296         rank = RANK_LOOKUP[card[:-1]]
297         suit = SUIT_LOOKUP[card[-1]] * N_ITERS
298         return rank + suit
299
300     def __decode_card(self, code):
301         rank = RANK_REVERSE[code % N_ITERS]
302         suit = SUIT_REVERSE[code // N_ITERS]
303         return rank + suit
304
305     def __deal_cards(self):
306         deck = [str(rank) + suit for rank in RANKS for suit in ↩
    SUITS]
307         deck = np.random.permutation(deck)
308         for i, agent in enumerate(self.agents):
```

```
309             hand = list(deck[N_ITERS * i:N_ITERS * (i + 1)])
310             self.hands[agent] = hand
311             for card in hand:
312                 self.private_info[agent].append(card)
313                 self.private_observations[agent][0, self.↩
    __encode_card(card)] = 1
314         return
315
316     def __update_info(self, agent):
317         self.infos[agent] = {'public': self.public_info, 'private↩
    ': self.private_info[agent]}
318         return
319
320     def __update_observation(self, agent):
321         self.observations[agent] = {
322             'observation': np.append(self.private_observations[↩
    agent], self.public_observation, axis=0),
323             'action_mask': self.action_masks[agent]}
324         return
325
326     def __update_action_mask(self, agent):
327         self.action_masks[agent] = self.private_observations[↩
    agent].flatten()
328         lead = self.cards_played[self.has_lead]
329         if lead != '':
330             lead_suit = lead[-1]
331             hand = self.hands[agent]
332             suits = [card[-1] for card in hand]
333             if lead_suit in suits:
334                 for card in hand:
335                     if card[-1] != lead_suit:
336                         self.action_masks[agent][self.↩
    __encode_card(card)] = 0
```

```
337         self.__update_observation(agent)
338         return
339 
340     def __choose_winner(self):
341         winning_card = self.cards_played[self.has_lead]
342         winning_player = self.has_lead
343         for player, card in self.cards_played.items():
344             if self.__higher_card(winning_card, card):
345                 winning_card = card
346                 winning_player = player
347 
348         return winning_player
349 
350     def __higher_card(self, winning_card, card):
351         suit = winning_card[-1]
352         rank = RANK_LOOKUP[winning_card[:-1]]
353         new_suit = card[-1]
354         new_rank = RANK_LOOKUP[card[:-1]]
355         if new_suit == suit:
356             if new_rank > rank:
357                 return True
358             else:
359                 return False
360         elif new_suit == TRUMP:
361             return True
362         return False
```

B.1.2 Deep Q Agents

The following code extends RLLib Deep Q networks to allow for the use of action masking.

```
1 import numpy as np
```

```
2
3 from copy import deepcopy
4
5 from ray.rllib.agents.dqn.dqn_torch_model import DQNTorchModel
6 from ray.rllib.agents.registry import get_trainer_class
7 from ray.rllib.examples.policy.random_policy import RandomPolicy
8 from ray.rllib.models.torch.fcnet import FullyConnectedNetwork as↩
      TorchFC
9 from ray.rllib.utils.framework import try_import_torch
10 from ray.rllib.utils.torch_utils import FLOAT_MAX
11 from ray.rllib.utils.annotations import override
12
13
14 from dqn_agents import cards_env
15
16 torch, nn = try_import_torch()
17
18 class TorchMaskedActions(DQNTorchModel):
19     """PyTorch version of above ParametricActionsModel."""
20
21     def __init__(self, obs_space, action_space, num_outputs, ↩
    model_config, name, **kw):
22         DQNTorchModel.__init__(
23             self, obs_space, action_space, action_space.n, ↩
    model_config, name, **kw
24         )
25
26         self.action_embed_model = TorchFC(
27             obs_space,  # obs_space: gym.spaces.space.Space
28             action_space,  # action_space: gym.spaces.space.Space
29             action_space.n,  # num_outputs: int
30             model_config,  # model_config: dict
31             name + "_action_embed",
```

```
        )
        self.obs_space = obs_space

        self.action_shape = action_space.n

    @override(DQNTorchModel)
    def forward(self, input_dict, state, seq_lens):
        # Extract the available actions tensor from the observation.
        action_mask = input_dict["obs"]["action_mask"]
        obs = input_dict["obs"]["observation"]

        obs = torch.reshape(obs, (-1, 600))

        combined_obs = torch.cat([action_mask, obs], dim=1)

        # Compute the predicted action embedding
        action_logits, _ = self.action_embed_model(
            {"obs": combined_obs}
        )

        # turns probit action mask into logit action mask
        inf_mask = torch.clamp(torch.log(action_mask), -1e10, FLOAT_MAX)
        masked_actions = action_logits + inf_mask
        return masked_actions, state

    def value_function(self):
        return self.action_embed_model.value_function()


class MaskedRandomPolicy(RandomPolicy):
```

```
62      """Hand-coded policy that returns random actions within the ↩
    action mask."""
63
64      def __init__(self, *args, **kwargs):
65          super().__init__(*args, **kwargs)
66
67      @override(RandomPolicy)
68      def compute_actions(
69              self,
70              obs_batch,
71              state_batches=None,
72              prev_action_batch=None,
73              prev_reward_batch=None,
74              **kwargs
75      ):
76          # Alternatively, a numpy array would work here as well.
77          # e.g.: np.array([random.choice([0, 1])] * len(obs_batch)↩
    )
78
79          return [np.random.choice(obs[:cards_env.N_CARDS].nonzero↩
    ()[0]) for obs in obs_batch], [], {}
80
81
82  def default_config():
83      config = deepcopy(get_trainer_class('DQN')._default_config)
84
85      config["v_min"] = -10.0
86      config["v_max"] = 10.0
87      config["lr"] = 0.001
88      config["num_gpus"] = 1 if torch.cuda.is_available() else 0
89
90      config["num_workers"] = 1
91      config["rollout_fragment_length"] = 30
```

```
    config["train_batch_size"] = 200
    config["horizon"] = 200
    config["no_done_at_end"] = False
    config["framework"] = "torch"
    config["log_level"] = "WARN"

    config["n_step"] = 1

    config["exploration_config"] = {
        # The Exploration class to use.
        "type": "StochasticSampling", #EpsilonGreedy
        # Config for the Exploration class' constructor:
        #"initial_epsilon": 0.1,
        #"final_epsilon": 0.0,
        #"epsilon_timesteps": 100000,  # Timesteps over which to anneal epsilon.
    }
    config["hiddens"] = []
    config["dueling"] = False

    return config
```

B.1.3 Rule-based Agent

The following code implements a rule-based agent compatible with the environment.

```
"""Hand-coded policy that returns rule-based actions.

The commented out bits of code result in a much stronger agent which would have given the rule-based training an
advantage compared to other methods."""

import numpy as np
```

```
7
8  from ray.rllib.examples.policy.random_policy import RandomPolicy
9  from ray.rllib.utils.annotations import override
10
11 from dqn_agents import cards_env
12
13
14 class RuleBasedPolicy(RandomPolicy):
15
16     def __init__(self, *args, **kwargs):
17         super().__init__(*args, **kwargs)
18         (obs_space, action_space, _) = args
19
20         self.obs_space = obs_space
21
22         self.action_shape = action_space.n
23
24     @override(RandomPolicy)
25     def compute_actions(
26             self,
27             obs_batch,
28             state_batches=None,
29             prev_action_batch=None,
30             prev_reward_batch=None,
31             **kwargs
32     ):
33
34         def trick_winner(played_this_trick):
35             played = list(np.array([np.sign(o.size) for o in ↩
   played_this_trick]).nonzero()[0])
36             lead = played.copy()
37             while len(lead) > 1:
38                 temp = lead.pop(0)
```

```
39                if ((temp + 3) % 4) not in played:
40                    lead.append(temp)
41            lead = lead[0]
42            winner = lead
43            winning_card = played_this_trick[lead][0]
44            for i, card in enumerate(played_this_trick):
45                if not card.size > 0:
46                    continue
47                # Same suit
48                if (card[0] // 6) == (winning_card // 6):
49                    if card[0] > winning_card:
50                        winning_card = card[0]
51                        winner = i
52                # Or spades
53                elif (card[0] // 6) == 0:
54                    winning_card = card[0]
55                    winner = i
56            return winner, winning_card
57
58        def identify_current_player(player_obs):
59            max_rows = np.array([o.sum(axis=1).nonzero()[0].max(↩
    initial=-1) for o in player_obs])
60            current_round = max_rows.max()
61            possible_current_players = list(
62                np.array([1 if m == current_round - 1 else 0 for ↩
    m in max_rows]).nonzero()[0])
63            if len(possible_current_players) > 0:
64                current_player = possible_current_players.copy()
65                while len(current_player) > 1:
66                    temp = current_player.pop(0)
67                    if ((temp + 3) % 4) not in ↩
    possible_current_players:
68                        current_player.append(temp)
```

```
69                current_player = current_player[0]
70            else:
71                current_player = None
72                current_round += 1
73            return current_player, current_round
74
75        def identify_lead_player(player_obs):
76            max_rows = np.array([o.sum(axis=1).nonzero()[0].max(↩
    initial=-1) for o in player_obs])
77            current_round = max_rows.max()
78            possible_lead_players = list(np.array([1 if m == ↩
    current_round else 0 for m in max_rows]).nonzero()[0])
79            if len(possible_lead_players) < 4:
80                lead_player = possible_lead_players.copy()
81                while len(lead_player) > 1:
82                    temp = lead_player.pop(0)
83                    if ((temp + 1) % 4) not in ↩
    possible_lead_players:
84                        lead_player.append(temp)
85                lead_player = lead_player[0]
86            else:
87                lead_player = None
88            return lead_player
89
90        def choose_single_action(obs):
91
92            obs = np.reshape(obs, (-1, 24))
93            # RLLib appends action mask to observation array
94            if obs.shape == (26, 24):
95                action_mask = obs[0, :]
96                obs = obs[1:, :]
97            legal_actions = action_mask.nonzero()[0] #obs[0, :].↩
    nonzero()[0]
```

```
98            played_cards = obs[1:, :].sum(axis=0).nonzero()[0]
99            player_obs = [obs[1:7, :], obs[7:13, :], obs[13:19, ↩
    :], obs[19:, :]]
100           current_player, current_round = ↩
    identify_current_player(player_obs)
101           lead_player = identify_lead_player(player_obs)
102
103           # If there is only one legal move, make it and stop ↩
    checking rules
104           if len(legal_actions) == 1:
105               return legal_actions[0]
106
107           # Lead will not be returned if the agent itself has ↩
    the lead
108           if not lead_player:
109               # Choose a random action
110               # Notably, leading aces performs much better than↩
     random actions, but
111               return np.random.choice(legal_actions)
112               # If holding any aces
113               #if any((legal_actions % 6) == 5):
114               #    aces = legal_actions[(legal_actions % 6 == ↩
    5)]
115               #    same_suits = [((legal_actions // 6) == (suit↩
     // 6)).sum() for suit in aces]
116                   # Choose the ace with the least cards beneath↩
     it to reduce chance of getting trumped
117               #    return aces[np.argmin(same_suits)]
118               # If no aces, play the non-trump card with the ↩
    most cards in play under it
119               #else:
120                   #undermined_cards = np.array([len(
```

```
121                 #    played_cards[np.all([(played_cards < ↩
    card), ((played_cards // 6) == (card // 6))], axis=0)]) for
122                 #                                card in ↩
    legal_actions])
123                 #self_undermined_cards = np.array([len(
124                 #    legal_actions[np.all([(legal_actions < ↩
    card), ((legal_actions // 6) == (card // 6))], axis=0)])
125                 #                                    for card ↩
    in legal_actions])
126                 #undermined_cards += self_undermined_cards
127                 #card_values = np.array([card % 6 for card in↩
     legal_actions])
128                 #cards_beneath = card_values - ↩
    undermined_cards
129                 #return legal_actions[cards_beneath.argmax()]
130                 #return np.random.choice(legal_actions)
131             else:
132                 played_this_trick = [o[current_round, :].nonzero↩
    ()[0] for o in player_obs]
133                 winning, winning_card = trick_winner(↩
    played_this_trick)
134                 n_played = sum([o.size > 0 for o in ↩
    played_this_trick])
135 
136             # If all the playable cards are the same suit
137             if len(set(legal_actions // 6)) == 1:
138                 # If that suit is spades
139                 if (legal_actions // 6)[0] == 0:
140                     # If going last play the lowest winning card
141                     if n_played == 3:
142                         if winning_card // 6 == 0:
143                             if len(legal_actions[legal_actions > ↩
    winning_card]) > 0:
```

```
144                             return legal_actions[↩
    legal_actions > winning_card].min()
145                         else:
146                             return legal_actions.min()
147                     else:
148                         return legal_actions.min()
149                 # If not last, play the lowest winning card
150                 elif winning_card // 6 == 0:
151                     if len(legal_actions[legal_actions > ↩
    winning_card]) > 0:
152                         return legal_actions[legal_actions > ↩
    winning_card].min()
153                     else:
154                         return legal_actions.min()
155                 else:
156                     return legal_actions.min()
157             else:
158                 # If it is possible to win the trick, do so
159                 if all((legal_actions // 6) == (winning // 6)↩
    ):
160                     if len(legal_actions[legal_actions > ↩
    winning_card]) > 0:
161                         # If agent is going last, play lowest↩
     winning card
162                         if n_played == 3:
163                             return legal_actions[↩
    legal_actions > winning_card].min()
164                         # Otherwise play a winning card at ↩
    random
165                         else:
166                             return np.random.choice(↩
    legal_actions[legal_actions > winning_card]) #legal_actions.↩
    max()
```

```
                        # If winning is not possible, play the lowest possible card
                        else:
                            return legal_actions.min()
                    # If following suit is not possible, play the lowest card
                    else:
                        return legal_actions.min()
            # Otherwise, if a trump can be played
            elif 0 in set(legal_actions // 6):
                trumps = legal_actions[legal_actions // 6 == 0]
                # Play the lowest winning trump
                if winning_card // 6 == 0:
                    if len(trumps[trumps > winning_card]) > 0:
                        return trumps[trumps > winning_card].min()
                    else:
                        return legal_actions.min()
                else:
                    return trumps.min()
            # Otherwise play the lowest card that won't win
            else:
                return legal_actions.min()

            # Otherwise no rules apply, so choose at random
            return np.random.choice(legal_actions)

        return np.array([choose_single_action(obs) for obs in obs_batch]), [], {}
```

B.2 JavaScript

The JavaScript code is used for the online experiment.

B.2.1 Interactive Behaviour

The following code governs most of the interactive behaviour of the web app.

```
import Empirica from "meteor/empirica:core";

// onGameStart is triggered once per game before the game starts, and before
// the first onRoundStart. It receives the game and list of all the players in
// the game.
Empirica.onGameStart((game) => {
  game.players.forEach((player) => {
    player.set("cumulativeScore", 0);
  });
});

// onRoundStart is triggered before each round starts, and before onStageStart.
// It receives the same options as onGameStart, and the round that is starting.
Empirica.onRoundStart((game, round) => {

  const ranks = ['9','10','jack','queen','king','ace']
  const suits = ['clubs','hearts','diamonds','spades']
  const deck = suits.flatMap((s) => (ranks.map((r) => ({rank: r, suit: s}))));

  const shuffled = _.shuffle(deck)
```

```
21
22   game.players.forEach((player, i) => {
23     const suitValues = {spades: 0, clubs: 1, diamonds: 2, hearts:↩
       3};
24     const rankValues = {9: 0, 10: 1, jack: 2, queen: 3, king: 4, ↩
     ace: 5};
25
26     let shuffledHand = shuffled.slice(i*6, (i+1)*6)
27     let sortedHand = shuffledHand.sort((a, b) => (suitValues[a['↩
     suit']]*6 + rankValues[a['rank']] > suitValues[b['suit']]*6 +↩
       rankValues[b['rank']]) ? 1 : -1)
28     player.round.set("hand", sortedHand);
29     player.round.set("score", 0)
30   })
31
32   const human = game.players.filter((player) => {return !player.↩
     bot;})[0];
33   round.set("humanPartner", human.get("partner"))
34
35   const partner = game.players.filter((p) => {return p.get("seat"↩
     ) == human.get("partner")})[0]
36   const partnerNames = game.get("partnerNames")
37   partner.set("name", partnerNames[Math.floor(round.get("↩
     effectiveIndex")/ 2)])
38
39   round.set("winner", "East");
40   let nullObs = new Float32Array(600);
41   round.set('cumulative-obs', nullObs);
42   round.set('current-stage', 0);
43
44   console.log("onRoundStart start");
45
46 });
```

```
47
48 // onStageStart is triggered before each stage starts.
49 // It receives the same options as onRoundStart, and the stage ↩
       that is starting.
50 Empirica.onStageStart((game, round, stage) => {
51   console.log(`onStageStart ${stage.get("type")} start`)
52
53   if (stage.get("type") === "outcome"){
54     game.players.forEach((player) => {
55       let card = round.get(`played-${player.get("seat")}`)
56       stage.set(`played-${player.get("seat")}`, card)
57     })
58   } else if (stage.get("type") === "play") {
59     game.players.forEach((player) => {
60       stage.set(`played-${player.get("seat")}`, null)
61       round.set(`played-${player.get("seat")}`, null)
62       round.set(`submitted-${player.get("seat")}`, false)
63     })
64     round.set('current-stage', round.get('current-stage') + 1)
65   } else {
66
67   }
68
69 });
70
71 // onStageEnd is triggered after each stage.
72 // It receives the same options as onRoundEnd, and the stage that↩
       just ended.
73 Empirica.onStageEnd((game, round, stage) => {
74
75   if (stage.get("type") === "play"){
76
77     // Wait a brief moment for server lag
```

```
78    let i = 0;
79    while ((i < 1000) && (game.players.some((player) => {return ↩
    round.get(`played-${player.get("seat")}`) !== undefined}))) {
80      i++;
81    }
82    if (game.players.every((player) => {return (round.get(`played↩
    -${player.get("seat")}`) !== undefined) && (round.get(`played↩
    -${player.get("seat")}`) !== null)})){
83
84      round.set("lead", round.get("winner"))
85      const leader = round.get("lead");
86      const leadCard = round.get(`played-${leader}`);
87      const rankValues = {"9":9, "10":10, "jack":11, "queen":12, ↩
    "king":13, "ace":14};
88
89      // Waiting here briefly because the servers can be slow to ↩
    update values
90      stage.set("winningSuit", leadCard["suit"]);
91      stage.set("winningRank", rankValues[leadCard["rank"]]);
92      stage.set("winningSeat", leader);
93
94      game.players.forEach((player) => {
95        let card = round.get(`played-${player.get("seat")}`)
96        if ((card['suit'] === stage.get("winningSuit")) && (↩
    rankValues[card["rank"]] > stage.get("winningRank"))) {
97          stage.set("winningRank", rankValues[card["rank"]]);
98          stage.set("winningSeat", player.get("seat"));
99        } else if ((card['suit'] === "spades") && (stage.get("↩
    winningSuit") !== "spades")) {
100         stage.set("winningSuit", "spades");
101         stage.set("winningRank", rankValues[card["rank"]]);
102         stage.set("winningSeat", player.get("seat"));
103       }
```

```
104       });
105
106       round.set("winner", stage.get("winningSeat"))
107
108       game.players.forEach((p) => {
109         if ((round.get("winner") === p.get("seat")) || (round.get↩
    ("winner") === p.get("partner"))){
110           p.round.set("score", p.round.get("score") + 1)
111         }
112       })
113     } else {
114       console.log("Timed out")
115       game.players.forEach((player) => {player.exit("timedOut")})
116     }
117   }
118 });
119
120 // onRoundEnd is triggered after each round.
121 // It receives the same options as onGameEnd, and the round that ↩
    just ended.
122 Empirica.onRoundEnd((game, round) => {
123 });
124
125 // onGameEnd is triggered when the game ends.
126 // It receives the same options as onGameStart.
127 Empirica.onGameEnd((game) => {
128   console.log("Game", game._id, "has ended");
129 });
```

B.2.2 Player Interface

The following code controls the interface when cards are being played by the participant.

```
import React from "react";

const PlayingCard = ({ cardDetails, seat, player, round, game }) =>
   => {

  const name = game.players.filter((p) => p.get("seat") == seat)
   [0].get("name")
  const isNull = cardDetails === null
  const self = player.get("seat") === seat ? "(You)" : ""
  const partner = player.get("partner") === seat ? "(Partner)" :
   ""
  const isLeader = round.get("winner") === seat
  const cssSeat = (player.get("seat") === seat) ? "self" : (
   player.get("partner") === seat) ? "partner" : (player.get("
   follows") === seat) ? "after" : "before"
  const cssClass = "card-thumb " +  cssSeat
  return (
    <div className={cssClass}>

        <strong className="card-label"> {name} {self} {partner}
   </strong>
        {!isNull && (
          <img
          src={`/cards/${cardDetails["rank"]}_of_${cardDetails["
   suit"]}.svg`}
          alt={`2_of_clubs`}
        />
        )}
    </div>
```

```
  );
};


export default class TaskStimulus extends React.Component {
  state = { interestValue: 0.08 };

  getName = (seat, game) => {
    const name = game.players.filter((p) => p.get("seat") == seat)[0].get("name")
    return name
  }

  render() {
    const {game, player, stage, round } = this.props;
    const task = round.get("task") || {};
    const pairData = task.features || {};
    const seats = ["North", "East", "South", "West"]
    let index = seats.findIndex( element => {
      if ((element === round.get("lead")) && (stage.get("type") === "outcome")) {
        return true;
      }
      if ((element === round.get("winner")) && (stage.get("type") === "play")) {
        return true;
      }
    });
    let displayOrder = []
    seats.forEach((s,i) => displayOrder[(i - index + 4) % 4] = s)

    return (stage.get("type") === "round_outcome" ? (<br/>) : (
      <div className="cards-table">
```

```
53          <div className="cards">
54            {displayOrder.map((seat) =>
55            <PlayingCard cardDetails={stage.get(`played-${seat}`)} ↩
    key={seat} seat={seat} player={player} round={round} game={↩
    game}/>
56              )
57              }
58            <div className="leader"><strong>{(stage.get("type") ===↩
     "play") ? "Lead: " : "Winner: "} {this.getName(round.get("↩
    winner"),game)}</strong></div>
59          </div>
60        </div>
61      ));
62    }
63  }
```

Listing B.1: Stimulus

```
1  import { Position, Toaster } from "@blueprintjs/core";
2  import React from "react";
3
4  const WarningToaster = Toaster.create({
5    className: "warning-toaster",
6    position: Position.TOP,
7  });
8
9  //timed button
10 const TimedButton = (props) => {
11   const { onClick, remainingSeconds, stage, cardDetails, disabled↩
      } = props;
12
13   return (
14     <input type="image"
15     className="card-button"
```

```
16      id='buttonname'
17      src={`/cards/${cardDetails["rank"]}_of_${cardDetails["suit"]}.svg`}
18      alt={`${cardDetails["rank"]} of ${cardDetails["suit"]}`}
19      onClick={evt => {onClick(evt, cardDetails)}}
20      disabled={disabled}
21      >
22
23      </input>
24      );
25  };
26
27
28  export default class TaskResponse extends React.Component {
29
30    encodeCard(card) {
31      const suitValues = {spades: 0, clubs: 1, diamonds: 2, hearts: 3};
32      const rankValues = {9: 0, 10: 1, jack: 2, queen: 3, king: 4, ace: 5};
33      return suitValues[card['suit']]*6 + rankValues[card['rank']];
34    }
35
36    getName = (seat, game) => {
37      const name = game.players.filter((p) => p.get("seat") == seat)[0].get("name")
38      return name
39    }
40
41    handleSubmit = (event, cardDetails) => {
42      event.preventDefault();
43
44      const { player, round, stage } = this.props;
```

```
45    const playersTurn =  ((player.get("seat") === round.get("winner")) || (round.get(`submitted-${player.get("follows")}`)))
46
47    if (!playersTurn){
48      WarningToaster.show({ message: "Currently waiting on the bots to play"});
49      return;
50    }
51    if (player.stage.submitted){
52      WarningToaster.show({ message: "Currently waiting on the bots to play"});
53      return;
54    }
55    if (playersTurn && (player.get("seat") !== round.get("winner"))){
56      const leadSuit = stage.get(`played-${round.get("winner")}`)['suit']
57      const hasSuit = player.round.get("hand").some((item) => {return item['suit'] === leadSuit;})
58      const followedSuit = cardDetails['suit'] == leadSuit
59      if (hasSuit && !followedSuit){
60        WarningToaster.show({ message: "You must follow suit when possible."});
61        return;
62      }
63    }
64    if (playersTurn && (!player.stage.submitted)){
65      stage.set(`played-${player.get("seat")}`, cardDetails);
66      round.set(`played-${player.get("seat")}`, cardDetails);
67      let hand = player.round.get("hand").filter((item) => {return item !== cardDetails;});
68      player.round.set("hand", hand);
```

```
69      player.stage.submit();
70
71      const agent_mapping = {East: 0, South: 1, West: 2, North: ↩
    3}
72      let stageIndex = round.get('current-stage');
73      const cumObs = round.get('cumulative-obs')
74      console.log(`Human play: ${cardDetails}`)
75      if (cardDetails) {
76        let obsIndex = agent_mapping[player.get("seat")]*6*24 + ↩
    this.encodeCard(cardDetails) + (stageIndex)*24;
77        console.log(obsIndex)
78        if (!isNaN(obsIndex)) {cumObs[obsIndex] = 1;}
79      }
80
81      round.set('cumulative-obs', cumObs);
82      round.set(`submitted-${player.get("seat")}`, true);
83      return;
84    }
85  };
86
87  handleNext = (event) => {
88    event.preventDefault();
89    const { player, stage, round } = this.props;
90    round.set(`submitted-${player.get("seat")}`, true);
91    player.stage.submit();
92    return;
93
94  };
95
96  renderResult() {
97    const { game, player, round, stage } = this.props;
98    const winner = ((round.get("winner") === player.get("seat")) ↩
    || (round.get("winner") === player.get("partner")));
```

```
99    const opponent = game.players.filter((p) => {return p.get("↩
    seat") === player.get("follows");})[0]
100
101
102
103    return (
104      <div className="result">
105        {winner ? (
106          <div className="alert alert-error">
107            <div className="alert-content">
108              <strong>Outcome</strong> Your team won the trick
109            </div>
110          </div>
111        ) : (
112          <div className="alert">
113            <div className="alert-content">
114              <strong>Outcome</strong> The opposing team won the ↩
    trick
115            </div>
116          </div>
117        )}
118        <div className="result-score">
119        <div className="result-item">
120            <div className="result-entry label">Played Lead</div>
121            <div className="result-entry value">
122              {this.getName(round.get("lead"), game)} {round.get(↩
    "lead") === player.get("seat") ? "(You)" : ""} {round.get("↩
    lead") === player.get("partner") ? "(Partner)" : ""}
123            </div>
124          </div>
125          <div className="result-item">
126            <div className="result-entry label">Winning Player</↩
    div>
```

```
127             <div className="result-entry value">
128               {this.getName(round.get("winner"),game)} {round.get↩
    ("winner") === player.get("seat") ? "(You)" : ""} {round.get(↩
    "winner") === player.get("partner") ? "(Partner)" : ""}
129             </div>
130           </div>
131           <div className="result-item last-item">
132             <div className="result-entry label">Cumulative Score↩
    </div>
133             <div className="result-entry value">
134               {player.round.get("score")} - {opponent.round.get("↩
    score")}
135             </div>
136           </div>
137         </div>
138         <div className="next-button-box">
139         <input type="button"
140                className="next-button"
141                onClick={this.handleNext}
142                value="Next"
143         ></input>
144         </div>
145       </div>
146     );
147   }
148
149   renderGameResult() {
150     const { game, player, round, stage } = this.props;
151     const opponent = game.players.filter((p) => {return p.get("↩
    seat") === player.get("follows");})[0]
152     const winner = player.round.get("score") > opponent.round.get↩
    ("score") ;
153     const tie = player.round.get("score") === opponent.round.get(↩
```

```
"score");

 return (
   <div className="result">
     {winner ? (
       <div className="alert alert-error">
         <div className="alert-content">
           <strong>Outcome</strong> Your team won the game!
         </div>
       </div>
     ) : tie ? (
       <div className="alert alert-neutral">
         <div className="alert-content">
           <strong>Outcome</strong> The game resulted in a tie.
         </div>
       </div>
     ) : (
       <div className="alert">
         <div className="alert-content">
           <strong>Outcome</strong> Your team was defeated!
         </div>
       </div>
     )}
     <div className="result-final-score">
       <div className="result-final-item">
         <div className="result-final-entry label">Cumulative Score</div>
         <div className="result-final-entry value">
           {player.round.get("score")} - {opponent.round.get("score")}
         </div>
       </div>
```