# Finding Friend and Foe in Multi-Agent Games

**Jack Serrino**[*]
MIT
jserrino@mit.edu

**Max Kleiman-Weiner**[*]
Harvard, MIT, Diffeo
maxkleimanweiner@fas.harvard.edu

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

**Joshua B. Tenenbaum**
MIT, CBMM
jbt@mit.edu

## Abstract

Recent breakthroughs in AI for multi-agent games like Go, Poker, and Dota, have seen great strides in recent years. Yet none of these games address the real-life challenge of cooperation in the presence of unknown and uncertain teammates. This challenge is a key game mechanism in hidden role games. Here we develop the DeepRole algorithm, a multi-agent reinforcement learning agent that we test on *The Resistance: Avalon*, the most popular hidden role game. DeepRole combines counterfactual regret minimization (CFR) with deep value networks trained through self-play. Our algorithm integrates deductive reasoning into vector-form CFR to reason about joint beliefs and deduce partially observable actions. We augment deep value networks with constraints that yield interpretable representations of win probabilities. These innovations enable DeepRole to scale to the full Avalon game. Empirical game-theoretic methods show that DeepRole outperforms other hand-crafted and learned agents in five-player Avalon. DeepRole played with and against human players on the web in hybrid human-agent teams. We find that DeepRole outperforms human players as both a cooperator and a competitor.

## 1 Introduction

Cooperation enables agents to achieve feats together that no individual can achieve on her own [16, 39]. Cooperation is challenging, however, because it is embedded within a competitive world [15]. Many multi-party interactions start off by asking: who is on my team? Who will collaborate with me and who do I need to watch out for? These questions arise whether it is your first day of kindergarten or your first day at the stock exchange. Figuring out who to cooperate with and who to protect oneself against is a fundamental challenge for any agent in a diverse multi-agent world. This has been explored in cognitive science, economics, and computer science [2, 7, 8, 21, 23, 24, 25, 26, 28, 30, 31, 44].

Core to this challenge is that information about who to cooperate with is often noisy and ambiguous. Typically, we only get this information indirectly through others' actions [1, 3, 21, 41]. Since different agents may act in different ways, these inferences must be robust and take into account ad-hoc factors that arise in an interaction. Furthermore, these inferences might be carried out in the presence of a sophisticated adversary with superior knowledge and the intention to deceive. These adversaries could intentionally hide their non-cooperative intentions and try to appear cooperative for their own benefit [36]. The presence of adversaries makes communication challenging— when intent to cooperate is unknown, simple communication is unreliable or "cheap" [14].

This challenge has not been addressed by recent work in multi-agent reinforcement learning (RL). In particular, the impressive results in imperfect-information two-player zero-sum games such as poker

[*]indicates equal contribution

Preprint. Under review.

[4, 6, 27] are not straightforward to apply to problems where cooperation is ambiguous. In heads-up poker, there is no opportunity to actually coordinate or cooperate with others since two-player zero-sum games are strictly adversarial. In contrast, games such as Dota and capture the flag have been used to train Deep RL agents that coordinate with each other to compete against other teams [17, 29]. However, in neither setting was there ambiguity about *who* to cooperate with. Further in real-time games, rapid reflexes and reaction times give an inherent non-strategic advantage to machines [9].

Here we develop *DeepRole*, a multi-agent reinforcement learning algorithm that addresses the challenge of learning who to cooperate with and how. We apply DeepRole to a five-player game of alliances, *The Resistance: Avalon* (Avalon), a popular hidden role game where the challenge of learning *who* to cooperate with is the central focus of play [13]. Hidden role games start with players joining particular teams and adopting roles that are not known to all players of the game. During the course of the game, the players try to infer and deduce the roles of their peers while others simultaneously try to prevent their role from being discovered. As of May 2019, Avalon is the most highly rated hidden role game on boardgamegeek.com. Hidden role games such as Mafia, Werewolf, and Saboteur are widely played around the world.

**Related work** DeepRole builds on the recent success of heuristic search techniques that combine efficient depth-limited lookahead planning with a value function learned through self-play in two-player zero-sum games [27, 33, 34]. In particular, the DeepStack algorithm for no-limit heads up poker combines counterfactual regret minimization (CFR) using a continual re-solving local search strategy with deep neural networks [27, 45]. While DeepStack was developed for games where all actions are public (such as poker), in hidden role games some actions are only observable by some agents and therefore must be deduced. In Avalon, players obtain new private information as the game progresses while in poker the only hidden information is the initial set of cards.

**Contributions.** Our key contributions build on these recent successes. Our algorithm integrates deductive reasoning into vector-form CFR [19] to reason about joint beliefs and partially observable actions based on consistency with observed outcomes, and augments value networks with constraints that yield interpretable representations of win probabilities. This augmented network enables training with better sample efficiency and generalization. We conduct an empirical game-theoretic analysis in five-player Avalon and show that the DeepRole CFR-based algorithm outperforms existing approaches and hand-crafted systems. Finally, we had DeepRole play with a large sample of human players on a popular online Avalon site. DeepRole outperforms people as both a teammate and opponent when playing with and against humans, even though it was only trained through self-play. We conclude by discussing the value of hidden role games as a long-term challenge for multi-agent RL systems.

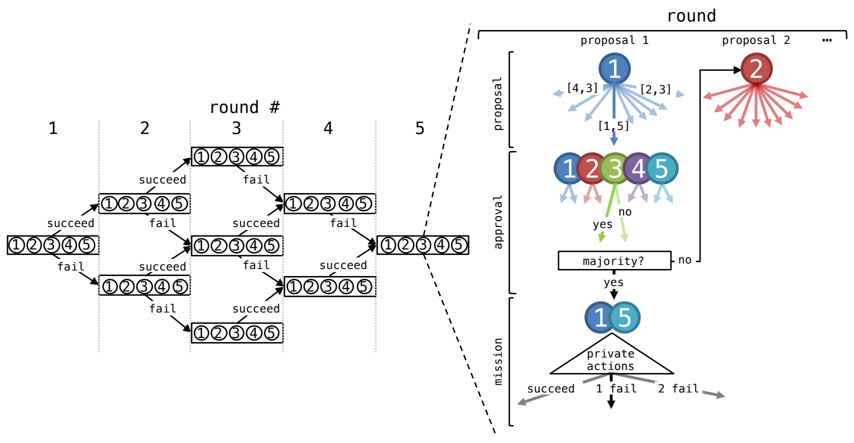


Figure 1: Description of the public game dynamics in The Resistance: Avalon. (left) Each round (rectangle) has up to 5 proposals (white circles) and leads to either a mission that fails or succeeds. (right) Example dynamics within each round. Players (colored circles) alternate proposing subsets of players (2 or 3) to go on a mission which are then put to vote by all 5 players. If the majority approves, those players (1 & 5 in this example) privately and independently decide to succeed or fail the mission. If the majority disapproves, the next player proposes a subset.

## 2 The Resistance: Avalon

We first briefly describe game mechanics of The Resistance: Avalon played with five players. At the beginning of the game, 3 players are randomly assigned to the *Resistance* team and 2 players are assigned to the *Spy* team. The spies know which players are on the Spy team (and hence also know which players are on the Resistance team). One member of the Resistance team is randomly and privately chosen to be the *Merlin* role who also knows all role assignments. One member of the Spy team is randomly chosen to be the *Assassin*. At the end of the game, if the Resistance team has won, the Assassin guesses the identity of Merlin. If the Assassin guesses Merlin correctly then the Spy team wins.

Figure 1 shows a visual description of the public game dynamics. There are five *rounds* in the game. During each round a player *proposes* a subset (two or three depending on the round) of agents to go on a *mission*. All players simultaneously and publicly vote (approve or not approve) of that subset. If a simple majority do not approve, another player is selected to propose a subset to go on the mission. If after five attempts, no proposal receives a simple majority, the Spy team wins. If a simple majority approves, the subset of players privately select whether the mission succeeds or fails. Players on the Resistance team must always choose success but players on the Spy team can choose success or failure. If any of the Spies choose to fail the mission, the mission fails. Otherwise, the mission succeeds. The total number of success and fail votes is made public but the identity of who made those votes is private. If three missions succeed, the Resistance team wins. If three missions fail, the Spy team wins. When people play Avalon, the games are usually rich in "cheap talk," such as defending oneself, accusing others, or debunking others' claims [10]. In this work, we do not consider the strategic implications of natural language communication.

Although Avalon is a simple game to describe, it has a large state space. We compute a lower bound of $10^{56}$ distinct information sets in the 5-player version of Avalon (Appendix D for details). This is larger than the state space of Chess ($10^{47}$) and larger than the number of information sets in heads-up limit poker ($10^{14}$) [18].

## 3 Algorithm: DeepRole

The DeepRole algorithm builds off of recent success in poker by combining DeepStack's innovations of deep value networks and depth-limited solving with deductive reasoning. Unique to DeepRole, our innovations allow the algorithm to play games with simultaneous and hidden actions. In broad strokes, DeepRole is composed of two parts: (1) a CFR planning algorithm augmented with deductive reasoning; and (2) neural value networks that are used to reduce the size of the game tree.

**Background.** Hidden role games like Avalon can be modeled as extensive-form games. We follow the notation of [19]. Briefly, these games have a game tree with nodes that correspond to different histories of actions, $h \in H$, with $Z \subset H$ the set of terminal histories. For each $h \in Z$, let $u_i(h)$ be the utility to player $i$ in terminal history $h$. In extensive-form games, only a single player $P(h)$ can move at any history $h$, but because Avalon's mechanics intimately involve simultaneous action, we extend this definition to let $P'(h)$ be the set of players simultaneously moving at $h$. Histories are partitioned into information sets ($I \in \mathcal{I}_i$) that represent the game states that player $i$ cannot distinguish between. For example, a Resistance player does not know who is on the Spy team and thus all $h$ differing only in the role assignments to the other players are in a single information set. The actions available in a given information set are $a \in A(I)$.

A strategy $\sigma_i$ for player $i$ is a mapping for each $I \in \mathcal{I}_i$ to a probability distribution over $A(I)$. Let $\sigma = (\sigma_1, \ldots, \sigma_p)$ be the joint strategy of all $p$ players. Then, we let $\pi^\sigma(h)$ be the probability of reaching $h$ if all players act according to $\sigma$. We write $\pi^\sigma_i(h)$ to mean the contribution of player $i$ to the joint probability $\pi^\sigma(h) = \prod_{1\ldots p} \pi^\sigma_i(h)$. Finally, let $\pi^\sigma_{-i}(h)$ be the product of strategies for all players except $i$ and let $\pi^\sigma(h, h')$ be the probability of reaching history $h'$ under strategy $\sigma$, given $h$ has occurred.

Counterfactual regret minimization (CFR) iteratively refines $\sigma$ based on the regret accumulated through a self-play like procedure. Specifically, in CFR+, at iteration $T$, the cumulative counterfactual regret is $R_i^{+,T}(I, a) = \max\{\sum_T CFV_i(\sigma^t_{I\to a}, I) - CFV_i(\sigma^t, I), 0\}$ where the counterfactual values for player $i$ are defined as $CFV_i(\sigma, I) = \sum_{z\in Z} u_i(z)\pi^\sigma_{-i}(z[I])\pi^\sigma(z[I], z)$ [38]. At a high-level, CFR iteratively improves $\sigma$ by boosting the probability of actions that would have been beneficial to each player. In two-player zero-sum games, CFR provably converges to a Nash equilibrium. However, it does not necessarily converge to an equilibrium in games with more than two players [37]. We investigate whether CFR can generate strong strategies in a multi-agent hidden role game like Avalon.

## 3.1 CFR with deductive logic

The CFR component of DeepRole is based on the vector-form public chance sampling (PCS) version of CFR introduced in [19], together with CFR+ regret matching [38]. Vector-form versions of CFR can result in faster convergence and take advantage of SIMD instructions, but require a public game tree [20]. In poker-like games, one can construct a public game tree from player actions, since all actions are public (e.g., bets, new cards revealed) except for the initial chance action (giving players cards). In hidden role games, however, key actions after the initial chance action are made privately, breaking the standard construction.

To support hidden role games, we extend the public game tree to be a history of third-person observations, $o \in O(h)$, instead of just actions. This includes both public actions and observable consequences of private actions (lines 22-44 in Alg. 1 in the Appendix). Our extension works when deductive reasoning from these observations reveals the underlying private actions. For instance, if a mission fails and one of the players is known to be a Spy, one can deduce that the Spy failed the mission. `deduceActions`$(h, o)$ carries out this deductive reasoning and returns the actions taken by each player under each information set ($\vec{a}_i[I]$) (line 23). With $\vec{a}_i[I]$ and the player's strategy ($\vec{\sigma_i}$), the player's reach probabilities are updated for the public game state following the observation ($ho$) (lines 24-26).

Using the public game tree, we maintain a human-interpretable joint posterior belief $\mathbf{b}(\rho|h)$ over the initial assignment of roles $\rho$. $\rho$ represents a full assignment of roles to players (the result of the initial chance action) – so our belief $\mathbf{b}(\rho|h)$ represents the joint probability that each player has the role specified in $\rho$, given the observed actions in the public game tree. See Figure 2 for an example belief $\mathbf{b}$ and assignment $\rho$. This joint posterior $\mathbf{b}(\rho|h)$ can be approximated by using the individual players' strategies as the likelihood in Bayes rule:

$$\mathbf{b}(\rho|h) \propto \mathbf{b}(\rho)(1 - \mathbb{1}\{h \vdash \neg\rho\}) \prod_{i \in 1 \ldots p} \pi_i^\sigma(I_i(h, \rho)) \quad (1)$$

where $\mathbf{b}(\rho)$ is the prior over assignments (uniform over the 60 possible assignments), $I_i(h, \rho)$ is the information set implied by public history $h$ and assignment $\rho$, and the product is the likelihood of playing to $h$ given each player's implied information set. A problem is that this likelihood can put positive mass on assignments that are impossible given the history. This arises because vector-form CFR algorithms can only compute likelihoods for each player independently rather than jointly. For instance, consider two players that went on a failing mission. In the information sets implied by the $\rho$ where they are both resistance, each player is assumed to have passed the mission. However, this is logically inconsistent with the history, as one of them must have played fail. To address this, the indicator term $(1 - \mathbb{1}\{h \vdash \neg\rho\})$ zeros the probability of any $\rho$ that is logically inconsistent with the public game tree $h$. This zeroing removes any impact these impossible outcomes would have had on the value and regret calculations in CFR (line 20 in Alg. 2).

## 3.2 Value network

The enhanced vector-form CFR cannot be run on the full public game tree of Avalon (or any real hidden role game). This is also the case for games like poker, so CFR-based poker systems [6, 27] rely on action abstraction and state abstraction to reduce the size of the game tree. However, actions in Avalon are not obviously related to each other. Betting 105 chips in poker is strategically similar to betting 104 chips, but voting up a mission in Avalon is distinct from voting it down. The size of Avalon's game tree does not come from the number of available actions, but rather from the number of players. Furthermore, since until now Avalon has only received limited attention, there are no developed hand-crafted state abstractions available either (although see [5] for how these could be learned). We follow the general approach taken by [27], using deep neural networks to limit the size of the game tree that we traverse (lines 14-16 in Alg. 1 in Appendix A).

We first partition the Avalon public game tree into individually solvable parts, segmented by a proposal for every possible number of succeeded and failed missions (white circles on the left side

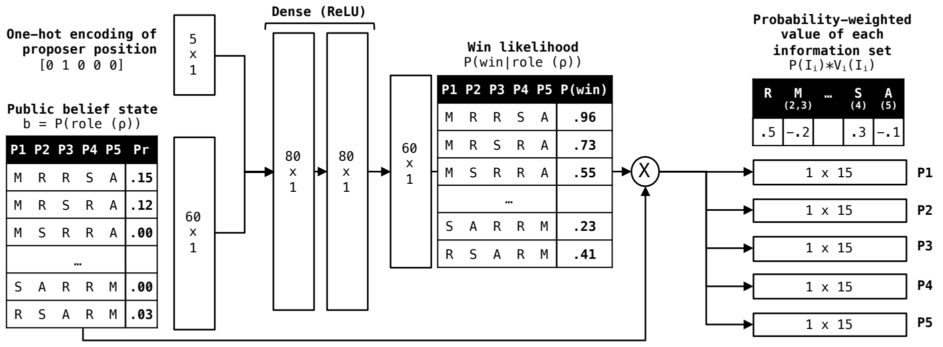


Figure 2: DeepRole neural network architecture used to limit game tree depth. Tables (black headers) show example inputs. The uppercase characters represent the different roles: (R)esistance, (S)py, (M)erlin, (A)ssassin. The outputs are the probability weighted value for each player in each of their information sets. While there is only one information set for Resistance (since they only know their own role), there are multiple for each of the other roles types. "M (2,3)" should be read as Merlin who sees players 2 and 3 as Spy and "S (4)" should be read as Spy who sees player 4 as Assassin.

of Figure 1). This yields 45 neural networks. Each $h$ corresponding to a proposal is mapped to one of these 45 networks. These networks take in a tuple $\theta \in \Theta, \theta = (i, \mathbf{b})$ where $i$ is the proposing player, and $\mathbf{b}$ is the posterior belief at that position in the game tree. $\Theta$ is the set of all possible game situations. The value networks are trained to predict the probability-weighted value of each information set (Figure 2).

Unlike in DeepStack, our networks calculate the *non-counterfactual* (i.e. normal) values for every information set $I$ for each player. This is because our joint belief representation loses the individual contribution of each player's likelihood, making it impossible to calculate a counterfactual. The value $V_i(I)$ for private information $I$ for player $i$ can be written as:

$$V_i(I) = \pi_i^\sigma(I) \sum_{h \in I} \pi_{-i}^\sigma(h) \sum_{z \in Z} \pi^\sigma(h, z) u_i(z) = \pi_i^\sigma(I)\, CFV_i(I)$$

where players play according to a strategy $\sigma$. Since we maintain a $\pi_i^\sigma(I)$ during planning, we can convert the values produced by the network to the counterfactual values needed by CFR (line 15 in Alg. 2).

**Value network architecture** While it's possible to estimate these values using a generic feed-forward architecture, it may cause lower sample efficiency, require longer training time, or fail to achieve a low loss. We design an interpretable custom neural network architecture that takes advantage of restrictions imposed by the structure of many hidden role games. Our network feeds a one-hot encoded vector of the proposer player $i$ and the belief vector $\mathbf{b}$ into two fully-connected hidden layers of 80 ReLU units. These feed into a fully-connected *win probability layer* with sigmoid activation. This layer is designed to take into account the specific structure of $V$, respecting the binary nature of payoffs in Avalon (players can only win or lose). It explicitly represents the probability of a Resistance win ($\vec{\mathbf{w}} = P(\text{win}|\rho)$) for each assignment $\rho$.

Using these probabilities, we then calculate the $V_i(I)$ for each player and information set, constraining the network's outputs to sound values. To do this calculation, for each player $i$, win probabilities are

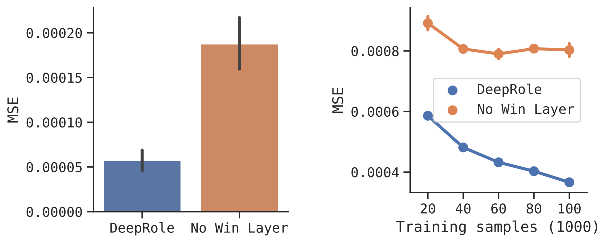


Figure 3: DeepRole generalization and sample efficiency. (left) Generalization error on held out samples averaged across the 45 neural networks. (right) Generalization error as a function of training data for the first deep value network (averaged over N=5 runs, intervals are SE).

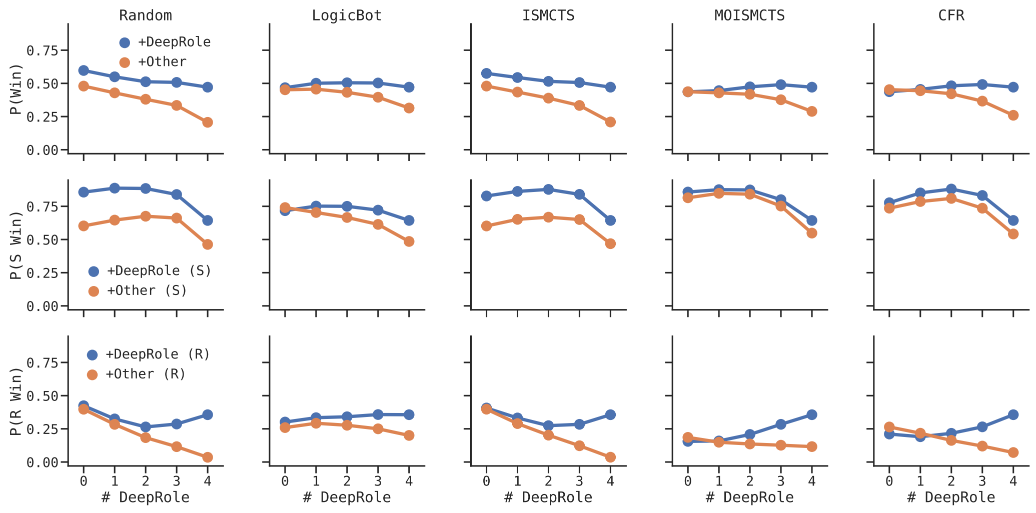


Figure 4: Comparing the expected win rate of DeepRole with other agents. The x-axis shows how many of the first four agents are DeepRole. The y-axis shows the expected win rate for the fifth agent if they played as DeepRole or the benchmark. Standard errors smaller than the size of the dots. (top) Combined expected win rate. (middle) Spy-only win rate. (bottom) Resistance-only win rate.

first converted to expected values ($\vec{u}_i\vec{\mathbf{w}} + \text{-}\vec{u}_i(1 - \vec{\mathbf{w}})$) representing $i$'s payoff in each $\rho$ if resistance win. It is then turned into the probability-weighted value of each information set which is used and produced by CFR: $\vec{V}_i = M_i[(\vec{u}_i\vec{\mathbf{w}} + \text{-}\vec{u}_i(1 - \vec{\mathbf{w}})) \odot \mathbf{b}]$ where $M_i$ is a $(15 \times 60)$ multi-one-hot matrix mapping each $\rho$ to player $i$'s information set, and $\mathbf{b}$ is the belief over roles passed to the network. This architecture is fully differentiable and is trained via back-propagation. A diagram and description of the full network is shown in Figure 2. See Appendix B and Alg. 3 for details of the network training algorithm, procedure, parameters and compute details.

The win probability layer enabled training with less training data and better generalization. When compared to a lesioned neural network that replaced the win probability layer with a zero-sum layer (like DeepStack) the average held-out loss per network was higher and more training data was required (Figure 3).

## 4 Empirical game-theoretic analysis

The possibility of playing with diverse teammates who may be playing conflicting equilibrium strategies, out-of-equilibrium strategies, or even human strategies makes evaluation outside of two-player zero-sum games challenging. In two-player zero-sum games, all Nash equilibria are minimally exploitable, so algorithms that converge to Nash are provably optimal in that sense. However evaluating 3+ player interactions requires considering multiple equilibria and metrics that account for coordinating with teammates. Further, Elo and its variants such as TrueSkill are only good measures of performance when relative skill is intransitive, but have no predictive power in transitive games (e.g., rock-paper-scissors) [40]. Thus, we turn to methods for empirical game-theoretic analysis which require running agents against a wide variety of benchmark opponents [40, 42].

We compare the performance of DeepRole to 5 alternative baseline agents: CFR – an agent trained with MCCFR [22] over a hand-crafted game abstraction; LogicBot – a hand-crafted strategy that uses logical deduction; RandomBot - plays randomly; ISMCTS - a single-observer ISMCTS algorithm found in [11, 12, 35]; MOISMCTS - a multiple-observer variant of ISMCTS [43]. Details for these agents are found in Appendix C.

We first investigated the conditional win rates for each baseline agent playing against DeepRole. We consider the effect of adding a 5th agent to a preset group of agents and compare DeepRole's win rate as the 5th agent with the win rate of a baseline strategy as the 5th agent in that same preset group. For each preset group (0-4 DeepRole agents) we simulated >20K games.

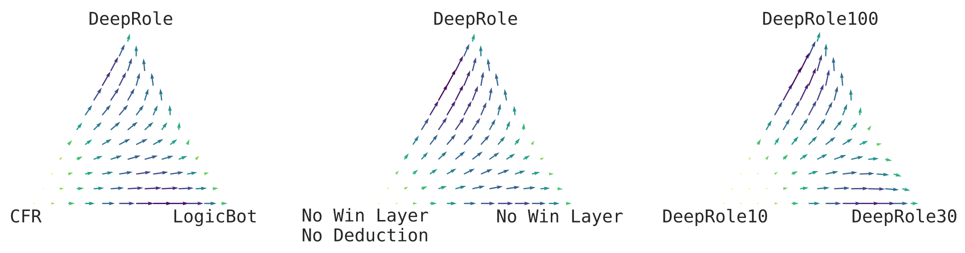


Figure 5: Empirical game-theoretic evaluation. Arrow size and darkness are proportional to the size of the gradient. (left) DeepRole against hand-coded agents. (center) DeepRole compared to systems without our algorithmic improvements. (right) DeepRole against itself but with CFR iterations equal to the number next to the game.

Figure 4 shows the win probability of each of these bots when playing DeepRole both overall and when conditioning on the role (Spy or Resistance). In most cases adding a 5th DeepRole player yielded a higher win rate than adding any of the other bots. This was true in every case we tested when there were at least two other DeepRole agents playing. Thus from an evolutionary perspective, DeepRole is robust to invasion from all of these agent types and in almost all cases outperforms the baselines even when it is the minority.

To formalize these intuitions we construct a meta-game where players select a mixed meta-strategy over agent types rather than actions. Figure 5 shows the gradient of the replicator dynamic in these meta-games [40, 42]. First, we compare DeepRole to the two hand-crafted strategies (LogicBot and CFR), and show that purely playing DeepRole is the equilibrium with the largest basin of attraction. The ISMCTS agents are too computationally demanding to run in these contests, but in a pairwise evaluation, playing DeepRole is the sole equilibrium.

Next, we test whether our innovations make DeepRole a stronger agent. We compare DeepRole to two lesioned alternatives. The first, DeepRole (No Win Layer), uses a zero-sum sum layer instead of our win probability layer in the neural network. Otherwise it is identical to DeepRole. In Figure 3, we saw that this neural network architecture did not generalize as well. We also compare to a version of DeepRole that does not include the logical deduction step in equation 1, and also uses the zero-sum layer instead of the probability win layer (No Win Layer, No Deduction). The agent without deduction is the weakest, and the full DeepRole agent is the strongest, showing that our innovations lead to enhanced performance.

Finally, we looked at the impact of CFR solving iterations during play (thinking time). More iterations make each move slower but may yield a better strategy. When playing DeepRole variants with 10, 30, and 100 iterations against each other, each variant was robust to invasion by the others but the more iterations used, the larger the basin of attraction (Figure 5).

| | Adding DeepRole or a Human | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | to 4 DeepRole | | | | to 4 Human | | | |
| | +DeepRole | | +Human | | +DeepRole | | +Human | |
| | Win Rate (%) | (N) | Win Rate (%) | (N) | Win Rate (%) | (N) | Win Rate (%) | (N) |
| Overall | **46.9 ± 0.6** | (7500) | 38.8 ± 1.3 | (1451) | **60.0 ± 5.5** | (80) | 48.1 ± 1.2 | (1675) |
| Resistance | **34.4 ± 0.7** | (4500) | 25.6 ± 1.5 | (856) | **51.4 ± 8.2** | (37) | 40.3 ± 1.5 | (1005) |
| Spy | **65.6 ± 0.9** | (3000) | 57.8 ± 2.0 | (595) | **67.4 ± 7.1** | (43) | 59.7 ± 1.9 | (670) |

Table 1: Win rates for humans playing with and against the DeepRole agent. When a human replaces a DeepRole agent in a group of 5 DeepRole agents, the win rate goes down for the team that the human joins. When a DeepRole agent replaces a human in a group of 5 humans, the win rate goes up for the team the DeepRole agent joins. Averages ± standard errors.

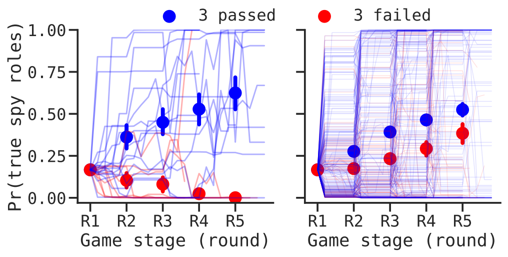


Figure 6: Belief dynamics over the course of the game. (left) DeepRole's posterior belief in the ground truth Spy role assignments as a Resistance player with four humans. (right) DeepRole's posterior belief of the true spy team while observing all-human games from the perspective a Resistance player.

## 5 Human evaluation

Playing with and against human players is a strong test of generalization. First, humans are likely to play a diverse set of strategies that will be challenging for DeepRole to respond to. During training time, it never learns from any human data and so its abilities to play with people must be the result of playing a strategy that generalizes to human play. Importantly, even if human players take the DeepRole neural networks "out of distribution", the online CFR iterations can still enable smart play in novel situations (as with MCTS in AlphaGo).

Humans played with DeepRole on the popular online platform ProAvalon.com (see Appendix F for commentated games and brief descriptions of DeepRole's "play style"). In the 2189 mixed human/agent games we collected, all humans knew which players were human and which were DeepRole. There were no restrictions on chat usage for the human players, but DeepRole did not say anything and did not process sent messages. Table 1 shows the win rate of DeepRole compared to humans. On the left, we can see that DeepRole is robust; when four of the players were DeepRole, a player would do better playing the DeepRole strategy than playing as an average human, regardless of team. More interestingly, when considering a game of four humans, the humans were better off playing with the DeepRole agent as a teammate than another human, again regardless of team. Although we have no way quantifying the absolute skill level of these players, among this pool of avid Avalon players, DeepRole acted as both a superior cooperator and competitor – it cooperated with its teammates to compete against the others.

Finally, DeepRole's interpretable belief state can be used to gain insights into play. In Figure 6 we show DeepRole's posterior probability estimate of the true set of Spies when playing as a Resistance player. When DeepRole played as the sole agent among four humans (left plot), the belief state rapidly converged to the ground truth in the situations where three missions passed, even though it had never been trained on human data. If three missions failed, it was often because it failed to learn correctly. Next, we analyze the belief state when fed actions and observations from the perspective of a human resistance player playing against a group of humans (yoked actions). As shown in Figure 6, the belief estimates increase as the game progresses, indicating DeepRole can make correct inferences even while just observing the game. The belief estimate converges to the correct state faster in games with three passes, presumably because the data in these games was more informative to all players.

## 6 Discussion

We developed a new algorithm for multi-agent games called DeepRole which effectively collaborates and competes with a diverse set of agents in The Resistance: Avalon. DeepRole surpassed both humans and existing machines in both simulated contests against other agents and a real-world evaluation with human Avalon players. These results are achieved through the addition of a deductive reasoning system to vector-based CFR and a win probability layer in deep value networks for depth-limited search. Taken together, these innovations allow DeepRole to scale to the full game of Avalon allowing CFR agents to play hidden role games for the first time. In future work, we will investigate whether the interpretable belief state of DeepRole could also be used to ground language, enabling better coordination through communication.

Looking forward, hidden role games are an exciting opportunity for developing AI agents. They capture the ambiguous nature of day-to-day interactions with others and go beyond the strictly adversarial nature of two-player zero-sum games. Only by studying 3+ player environments can we start to capture some of the richness of human social interactions including alliances, relationships, teams, and friendships [32].

**Acknowledgments**

We thank Victor Kuo and ProAvalon.com for help integrating DeepRole with human players online. We also thank Noam Brown and Murray Campbell for helpful discussions and comments. This work was supported in part by The Future of Life Institute, DARPA Ground Truth, the Center for Brains, Minds and Machines (NSF STC award CCF-1231216), and the Templeton World Charity Foundation.

## References

[1] Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *arXiv preprint arXiv:1709.08071*, 2017.

[2] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, 1985.

[3] Chris L Baker, Julian Jara-Ettinger, Rebecca Saxe, and Joshua B Tenenbaum. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, 1:0064, 2017.

[4] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

[5] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. *arXiv preprint arXiv:1811.00164*, 2018.

[6] Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 2017.

[7] Colin Camerer. *Behavioral game theory: Experiments in strategic interaction*. Princeton University Press, 2003.

[8] Colin F Camerer, Teck-Hua Ho, and Juin-Kuan Chong. A cognitive hierarchy model of games. *The Quarterly Journal of Economics*, pages 861–898, 2004.

[9] Rodrigo Canaan, Christoph Salge, Julian Togelius, and Andy Nealen. Leveling the playing field-fairness in ai versus human game benchmarks. *arXiv preprint arXiv:1903.07008*, 2019.

[10] Gokul Chittaranjan and Hayley Hung. Are you a werewolf? detecting deceptive roles and outcomes in a conversational role-playing game. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5334–5337. IEEE, 2010.

[11] Peter I Cowling, Edward J Powley, and Daniel Whitehouse. Information set Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):120–143, 2012.

[12] Peter I Cowling, Daniel Whitehouse, and Edward J Powley. Emergent bluffing and inference with Monte Carlo tree search. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 114–121. IEEE, 2015.

[13] Don Eskridge. The Resistance: Avalon, 2012.

[14] Joseph Farrell and Matthew Rabin. Cheap talk. *Journal of Economic perspectives*, 10(3):103–118, 1996.

[15] Adam Galinsky and Maurice Schweitzer. *Friend and Foe: When to Cooperate, when to Compete, and how to Succeed at Both*. Random House, 2015.

[16] Joseph Henrich. *The secret of our success: how culture is driving human evolution, domesticating our species, and making us smarter*. Princeton University Press, 2015.

[17] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*, 2018.

[18] Michael Johanson. Measuring the size of large no-limit poker games. *arXiv preprint arXiv:1302.7008*, 2013.

[19] Michael Johanson, Nolan Bard, Marc Lanctot, Richard Gibson, and Michael Bowling. Efficient nash equilibrium approximation through monte carlo counterfactual regret minimization. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 837–846. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[20] Michael Johanson, Kevin Waugh, Michael Bowling, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[21] Max Kleiman-Weiner, Mark K Ho, Joseph L Austerweil, Michael L Littman, and Joshua B Tenenbaum. Coordinate to cooperate or compete: abstract goals and joint intentions in social interaction. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society*, 2016.

[22] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, pages 1078–1086, 2009.

[23] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Julien Perolat, David Silver, Thore Graepel, et al. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4193–4206, 2017.

[24] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[25] Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *arXiv preprint arXiv:1707.01068*, 2017.

[26] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, volume 94, pages 157–163, 1994.

[27] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[28] Martin A Nowak. Five rules for the evolution of cooperation. *Science*, 314(5805):1560–1563, 2006.

[29] OpenAI. OpenAI Five. `https://blog.openai.com/openai-five/`, 2018.

[30] Julien Perolat, Joel Z Leibo, Vinicius Zambaldi, Charles Beattie, Karl Tuyls, and Thore Graepel. A multi-agent reinforcement learning model of common-pool resource appropriation. In *Advances in Neural Information Processing Systems*, pages 3646–3655, 2017.

[31] David G Rand and Martin A Nowak. Human cooperation. *Trends in cognitive sciences*, 17(8):413, 2013.

[32] Michael Shum, Max Kleiman-Weiner, Michael L Littman, and Joshua B Tenenbaum. Theory of minds: Understanding behavior in groups through inverse planning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.

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

[34] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[35] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[36] DJ Strouse, Max Kleiman-Weiner, Josh Tenenbaum, Matt Botvinick, and David J Schwab. Learning to share and hide intentions using information regularization. In *Advances in Neural Information Processing Systems*, pages 10270–10281, 2018.

[37] Duane Szafron, Richard Gibson, and Nathan Sturtevant. A parameterized family of equilibrium profiles for three-player kuhn poker. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 247–254. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[38] Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit texas hold'em. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[39] Michael Tomasello. *A natural history of human thinking*. Harvard University Press, 2014.

[40] Karl Tuyls, Julien Perolat, Marc Lanctot, Joel Z Leibo, and Thore Graepel. A generalised method for empirical game theoretic analysis. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 77–85. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[41] Tomer Ullman, Chris Baker, Owen Macindoe, Owain Evans, Noah Goodman, and Joshua B Tenenbaum. Help or hinder: Bayesian models of social goal inference. In *Advances in neural information processing systems*, pages 1874–1882, 2009.

[42] Michael P Wellman. Methods for empirical game-theoretic analysis. In *AAAI*, pages 1552–1556, 2006.

[43] Daniel Whitehouse. *Monte Carlo tree search for games with hidden information and uncertainty*. PhD thesis, University of York, 2014.

[44] Michael Wunder, Michael Kaisers, John Robert Yaros, and Michael Littman. Using iterated reasoning to predict opponent strategies. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 593–600. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[45] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2008.

## A DeepRole depth-limited CFR

**Algorithm 1** DeepRole depth-limited CFR

```
1: INPUT $h$ (root public game history); $\mathbf{b}$ (root public belief); $n$ (# iterations); $d$ (averaging delay);
   NN$[h]$ (neural networks that approximate CFVs from $h$)

   Init regrets $\forall I, r_I[a] \leftarrow 0$, Init cumulative strategies $\forall I, s_I[a] \leftarrow 0$

2: procedure SolveSituation($h, \mathbf{b}, n, d$)
3:     $\vec{u}_{1...p} \leftarrow \vec{0}$
4:     for $i = 1$ to $n$ do
5:         $w_i \leftarrow \max(i - d, 0)$
6:         $\vec{u}_{1...p} \leftarrow \vec{u}_{1...p}$+ModifiedCFR+($h, \mathbf{b}, w_i, \vec{1}_{1...p}$)
7:     end for
8:     return $\vec{u}_{1...p}$ / $\sum w_i$
9: end procedure

10: procedure ModifiedCFR+($h, \mathbf{b}, w, \vec{\pi}_{1...p}$)
11:     if $h \in Z$ then
12:         return TerminalCFVs($h, \mathbf{b}, \vec{\pi}_{1...p}$)
13:     end if
14:     if $h \in$ NN then
15:         return NeuralCFVs($h, \mathbf{b}, \vec{\pi}_{1...p}$)
16:     end if
17:     $\vec{u}_{1...p} \leftarrow \vec{0}$
18:     for $i \in P'(h)$ do  ▷ A strategy must be calculated for all moving players at public history $h$
19:         $\vec{I}_i \leftarrow$ lookupInfosets$_i(h)$
20:         $\vec{\sigma}_i \leftarrow$ regretMatching+($\vec{I}_i$)
21:     end for
22:     for public observation $o \in O(h)$ do
23:         $\vec{a}_{1...p} \leftarrow$ deduceActions$(h, o)$
24:         for $i \in P'(h)$ do
25:             $\vec{\pi}_i \leftarrow \vec{\sigma}_i[\vec{a}_i] \odot \vec{\pi}_i$
26:         end for
27:         $\vec{u}'_{1...p} \leftarrow$ ModifiedCFR+($ho, \mathbf{b}, w, \vec{\pi}_{1...p}$)
28:         for each player $i$ do
29:             if $i \in P'(h)$ then
30:                 $\vec{m}_i[\vec{a}_i] \leftarrow \vec{m}_i[\vec{a}_i] + \vec{u}_i$
31:                 $\vec{u}_i \leftarrow \vec{u}_i + \vec{\sigma}_i[\vec{a}_i] \odot \vec{u}'_i$
32:             else
33:                 $\vec{u}_i \leftarrow \vec{u}_i + \vec{u}'_i$
34:             end if
35:         end for
36:     end for
37:     for $i \in P'(h)$ do ▷ Similar to line 18, we must perform these updates for all moving players
38:         for $I \in \vec{I}_i$ do
39:             for $a \in A(I)$ do
40:                 $r_I[a] \leftarrow \max(r_I[a] + \vec{m}_i[a][I] - \vec{u}_i[I], 0)$
41:                 $s_I[a] \leftarrow s_I[a] + \vec{\pi}_i[I]\vec{\sigma}_i[I][a]w$
42:             end for
43:         end for
44:     end for
45:     return $\vec{u}_{1...p}$
46: end procedure
```

**Algorithm 2** Terminal value calculation

```
 1: procedure TerminalCFVs($h, \mathbf{b}, \vec{\pi}_{1...p}$)
 2:     $\vec{v}_{1...p}[\cdot] \leftarrow 0$                                   ▷ Initialize factual values
 3:     $\mathbf{b}_{\text{term}} \leftarrow$ CalcTerminalBelief($h, \mathbf{b}, \vec{\pi}_{1...p}$)
 4:     for $i = 1$ to $p$ do
 5:         for $\rho \in \mathbf{b}_{\text{term}}$ do
 6:             $\vec{v}_i[I_i(h,\rho)] \leftarrow \vec{v}_i[I_i(h,\rho)] + \mathbf{b}_{\text{term}}[\rho]u_i(h,\rho)$
 7:         end for
 8:     end for
 9:     return $\vec{v}_{1...p}/\vec{\pi}_{1...p}$                  ▷ Convert factual to counterfactual
10: end procedure

11: procedure NeuralCFVs($h, \mathbf{b}, \vec{\pi}_{1...p}$)
12:     $\mathbf{b}_{\text{term}} \leftarrow$ CalcTerminalBelief($h, \mathbf{b}, \vec{\pi}_{1...p}$)
13:     $w \leftarrow \sum_\rho \mathbf{b}_{\text{term}}[\rho]$
14:     $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_p \leftarrow \text{NN}[h](h, \mathbf{b}_{term}/w)$     ▷ Call NN with normalized belief
15:     return $w\vec{v}_{1...p}/\vec{\pi}_{1...p}$                 ▷ Convert factual to counterfactual
16: end procedure

17: procedure CalcTerminalBelief($h, \mathbf{b}, \vec{\pi}_{1...p}$)
18:     for $\rho \in \mathbf{b}$ do
19:         $\mathbf{b}_{\text{term}}[\rho] \leftarrow \mathbf{b}[\rho]\prod_i \vec{\pi}_i(I_i(h,\rho))$
20:         $\mathbf{b}_{\text{term}}[\rho] \leftarrow \mathbf{b}_{\text{term}}[\rho](1 - \mathbb{1}\{h \vdash \neg\rho\})$     ▷ Zero beliefs that are logically inconsistent
21:     end for
22:     return $\mathbf{b}_{\text{term}}$
23: end procedure
```

## B Value network training

We generate training data for the deep value networks by using CFR to solve each part of the game from a random sample of starting beliefs. By working backwards from the end of the game, trained networks from later stages enable data generation using CFR at progressively earlier stages. This progressive back-chaining follows the dependency graph of proposals shown on the left side of Figure 1. This generalizes the procedure used to generate DeepStack's value networks [27].

For each network, we sampled $120,000$ game situations ($\theta \in \Theta$) to be used for training and validation. For each sample, CFR ran for 1500 iterations, skipping the first 500 during averaging. The neural networks were each trained for 3000 epochs (batch size of 4096) using the Adam optimizer with a mean squared error loss on $\vec{V}$. Training hyperparameters and weight initializations used Keras defaults. 10% of the data was set aside for validation. Training on 480 CPU cores, 480 GB of memory, and 1 GPU took roughly 48 hours to produce the networks for every stage in the game.

## C Comparison Agents

**CFR** CFR denotes an agent using a strategy trained by external sampling MCCFR with a hand-built imperfect-recall abstraction, used to reduce the size of Avalon's immense game tree. We bucket information sets for players based on their initial information set (their role and who they see) and a set of hand-chosen game metrics: the round number, the number of failed missions each player has participated in, and the number of times a player has proposed a failing mission. We trained the bot until we observed decayed performance of the bot in self-play. In total, CFRBot was trained for 6,000,000 iterations.

**LogicBot** LogicBot is an agent that plays a hand-crafted pre-set strategy derived from our intuition of playing Avalon with real people. During play, LogicBot keeps a list of possible role assignments that are logically consistent with the observations it has made. As resistance, it randomly samples an assignment and proposes a mission using the resistance players in that assignment. It votes up proposals if and only if the proposed players and the proposer are resistance in a randomly sampled assignment or if it is the final proposal in the round. As spy, it proposes randomly, votes opposite to resistance players, and selects merlin randomly.

**Random** Our random agent selects an action uniformly from the available actions.

**Algorithm 3** Backwards training

```
1: INPUT $P_{1...n}$: Dependency-ordered list of game parts.
2: INPUT $\Theta_{1...n}$: For each game part, a distribution over game situations.
3: INPUT $d$: The number of training datapoints generated per game partition.
4: OUTPUT $N_{1...n}$: $n$ trained neural value networks, one for each game part.

5: procedure EndToEndTrain($P_{1...n}, \Theta_{1...n}, d$)    ▷ Train a neural network for each game partition
6:     for $i = 1$ to $n$ do
7:         $\mathbf{x}, \mathbf{y} \leftarrow$ GenerateDatapoints($P_i, \Theta_i, N_{1...i-1}$)
8:         $N_i \leftarrow$ TrainNN($\mathbf{x}, \mathbf{y}$)
9:     end for
10:    return $N_{1...n}$
11: end procedure

12: procedure GenerateDatapoints($d, S, \Theta, N_{1...k}$)  ▷ Given a game partition, it's distribution over game situations, and the NNs needed to limit solution depth, generate $d$ datapoints.
13:     for $i = 1$ to $d$ do
14:         $\theta_i \sim \Theta$    ▷ Sample a game situation from the distribution
15:         $\mathbf{v}_i \leftarrow$ SolveSituation($S, \theta_i, N_{1...k}$)    ▷ Solve that game situation for every player's values, using previously trained neural networks to solution depth.
16:     end for
17:     return $\theta_{1...d}, \mathbf{v}_{1...d}$    ▷ Return all training datapoints
18: end procedure
```

**Algorithm 4** Game Situation Sampler

```
1: INPUT $s$: The number of succeeds.
2: INPUT $f$: The number of fails.
3: OUTPUT $p, \mathbf{b}$: A random game situation from this game part, consisting of a proposer and a belief over the roles.

4: procedure SampleSituation($s, f$)
5:     $I \leftarrow$ SampleFailedMissions($s, f$)    ▷ Uniformly sample $f$ failed missions
6:     $E \leftarrow$ EvilPlayers($I$)    ▷ Calculate evil teams consistent with the missions
7:     $P(E) \sim \text{Dir}(\vec{1}_{|E|})$    ▷ Sample probability of each evil team
8:     $P(M) \sim \text{Dir}(\vec{1}_n)$    ▷ Sample probability of being Merlin for all players
9:     $\mathbf{b} \leftarrow P(E) \bigotimes P(M)$    ▷ Create a belief distribution using $P(E)$ and $P(M)$
10:    $p \sim \text{unif}\{1, n\}$    ▷ Sample a proposer uniformly over all the players
11:    return $p, \mathbf{b}$
12: end procedure
```

**ISMCTS & MOISMCTS** We also evaluate our bot against opponents playing using the ISMCTS family of algorithms. Specifically, we evaluate our bot against the single-observer ISMCTS (ISMCTS) algorithm shown in [11, 12, 43], as well as the improved multiple-observer version of the algorithm (MOISMCTS). Each variant used 10,000 iterations per move.

## D State space calculation

Unlike large two-player games like Poker, Go, or Chess, Avalon's complexity lies in the combinatorial explosion that comes with having 5 players, four role types (Spy, Resistance, Merlin, Assassin), and a large number observable moves. We lower bound the number of information sets by just considering the longest possible game. The longest possible game lasts five rounds with each round requiring five proposals. Each proposal can made in 10 different ways by choosing which 2 or 3 players out of 5 should go on the mission. From there, there are 16 ways proposals 1-4 can be voted down and 16 ways proposal 5 can be voted up. Thus, a lower bound on the number of information sets is $(10 * 16)^{5*5} \approx 10^{56}$ which does not consider shorter games or any of the private information.

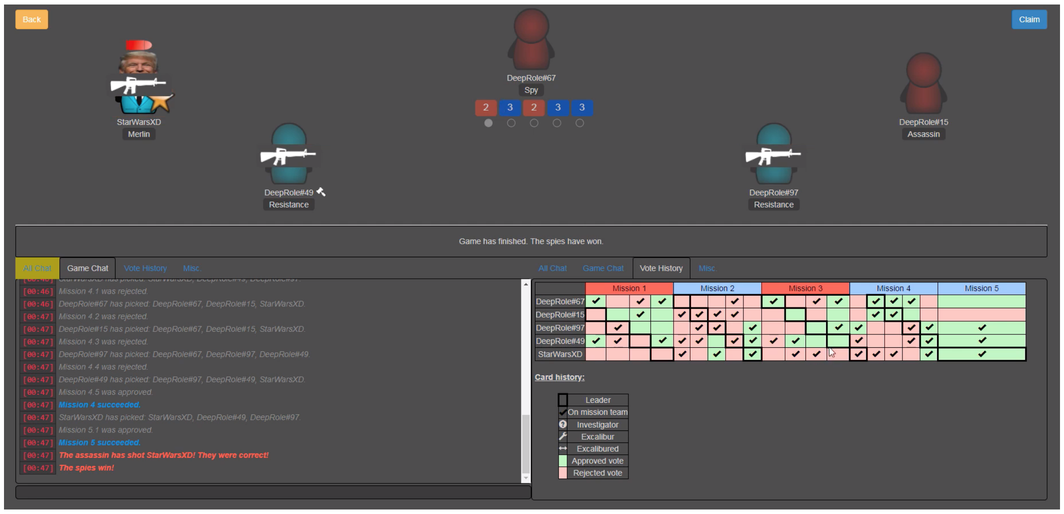
## E ProAvalon.com

`ProAvalon.com` is a website where players can play Avalon online in groups of 5 to 10. We've integrated DeepRole in to this website, allowing humans from all around the world to play against 0-4 DeepRole agents. Fig. 7 shows the game interface for human players on ProAvalon.com. Natural language communication is done via a publicly visible chat. See the website for more details about the specific interface.

## F Human commentary of DeepRole v. Human games.

Some players on ProAvalon.com have uploaded commentary that qualitatively examine the style of play the bots have. We examine two of these games to show DeepRole effectively cooperating and competing with a human player.

In the first game we examine (`https://www.youtube.com/watch?v=LKdY4Us0Ci4`), the human player is playing as "VT" ("Vanilla Town", i.e. non-Merlin resistance). After the first two missions fail and the third one passes, the human player is able to accurately deduce the identities of the spy players. During proposals for the 4th and 5th missions, however, his fellow resistance teammates (including Merlin), seem to be rejecting missions that he knows to be "clean" (do not contain a spy). While he expresses exasperation that one of his teammates doesn't seem to deduce the obvious, these clean missions are eventually approved and succeed. At the end of the



Game has finished. The spies have won.

```
[00:46] StarWarsXD has picked: StarWarsXD, DeepRole#49, DeepRole#97.
[00:46] Mission 4.1 was rejected.
[00:46] DeepRole#67 has picked: DeepRole#67, DeepRole#15, StarWarsXD.
[00:47] Mission 4.2 was rejected.
[00:47] DeepRole#15 has picked: DeepRole#67, DeepRole#15, StarWarsXD.
[00:47] Mission 4.3 was rejected.
[00:47] DeepRole#97 has picked: DeepRole#67, DeepRole#97, DeepRole#49.
[00:47] Mission 4.4 was rejected.
[00:47] DeepRole#49 has picked: DeepRole#97, DeepRole#49, StarWarsXD.
[00:47] Mission 4.5 was approved.
[00:47] Mission 4 succeeded.
[00:47] StarWarsXD has picked: StarWarsXD, DeepRole#49, DeepRole#97.
[00:47] Mission 5.1 was approved.
[00:47] Mission 5 succeeded.
[00:47] The assassin has shot StarWarsXD! They were correct!
[00:47] The spies win!
```

| | Mission 1 | | | | Mission 2 | | | | | Mission 3 | | | | Mission 4 | | | | | Mission 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DeepRole#67 | 🟩✓ | 🟥 | 🟥✓ | 🟩✓ | 🟥 | 🟥 | 🟥 | 🟥✓ | 🟥 | 🟩✓ | 🟥 | 🟥✓ | 🟩✓ | 🟥 | 🟩✓ | 🟩✓ | 🟩✓ | 🟥 | 🟩 |
| DeepRole#15 | 🟥 | 🟩 | 🟩✓ | 🟩 | 🟥✓ | 🟥✓ | 🟥✓ | 🟥✓ | 🟥 | 🟥 | 🟩 | 🟥 | 🟩 | 🟥 | 🟩✓ | 🟩✓ | 🟩 | 🟥 | 🟥 |
| DeepRole#97 | 🟥 | 🟥✓ | 🟩 | 🟩 | 🟥 | 🟥✓ | 🟥✓ | 🟥 | 🟩✓ | 🟥 | 🟥 | 🟩 | 🟩✓ | 🟩✓ | 🟥 | 🟥 | 🟥✓ | 🟩✓ | 🟩✓ |
| DeepRole#49 | 🟩✓ | 🟥✓ | 🟥 | 🟩✓ | 🟥✓ | 🟥✓ | 🟩 | 🟥✓ | 🟩✓ | 🟥✓ | 🟩✓ | 🟩 | 🟩 | 🟥✓ | 🟥 | 🟥 | 🟥✓ | 🟩✓ | 🟩✓ |
| StarWarsXD | 🟥 | 🟥 | 🟥 | 🟥 | 🟥✓ | 🟥 | 🟩✓ | 🟥 | 🟩✓ | 🟥 | 🟥✓ | 🟥✓ | 🟥 | 🟥✓ | 🟥✓ | 🟥✓ | 🟥 | 🟩✓ | 🟩✓ |

**Card history:**

| | |
|---|---|
| ☐ | Leader |
| ✓ | On mission team |
| ? | Investigator |
| | Excalibur |
| ↔ | Excalibured |
| 🟩 | Approved vote |
| 🟥 | Rejected vote |

Figure 7: The ProAvalon.com game interface. This shows a completed game between 4 DeepRole agents and a human player (no affiliation to this work's authors). The interface consists of a visualization of a "round table" of players (top), a public chat for each game (bottom left), and the public game history (bottom right).

game, resistance win, revealing that the rejecting player was Merlin all along – purposefully rejecting missions to seem ignorant.

In the second game we examine (`https://www.youtube.com/watch?v=9RkUFHYTo_s`), the human player is playing as Merlin. During multiple rounds of the game, the human player "slams clean", proposing a mission containing no spies – generally an obvious indicator of Merlin-like knowledge of the spy players. While these missions are ultimately approved and succeeded, the DeepRole Assassin correctly identifies the human player due to their obvious play, resulting in a spy victory.

There are more examples of DeepRole v. human games on YouTube, and we encourage readers to check out other videos with qualitative analysis of DeepRole.