Theory of Mind Objectives for Hanabi

Victor Wang **Jonathan Lynn**

Abstract: Hanabi is a cooperative card game that has served as a testbed for exploring theory of mind (ToM) in decision making tasks in multi-agent settings. We design training objectives to facilitate ToM reasoning for self-play Hanabi. Specifically, we train a Reinforcement Learning agent to predict its own cards, and use these predictions to bootstrap training for higher order beliefs.

Our code is available at https://github.com/dubai03nsr/hanabi-learning-environment.

Our video can be accessed at https://youtu.be/0Bl7VyVs3XU.

1 Introduction

Games are some of the most popular and successful applications for Reinforcement Learning (RL). One reason is that, in comparison with real-world scenarios, games tend to have more clearly-defined rules and objectives and can be more easily abstracted into simple, finite environments (i.e. state and action spaces, etc). Moreover, due to the popularity of many of these games, we are able to access large amounts of gameplay as data for planning and learning, as evidenced by progress in boardgames such as Go [1, 2], Backgammon [3, 4, 5, 6], checkers [7, 8], and chess [9]. As a result, the field has achieved remarkable success in learning how to play many of these games. For instance, policies learned by AlphaGo [1] and AlphaGo Zero [2] were able to achieve impressive win rates in Go utilizing Monte Carlo Tree Search, beating the top professional human players at the time [10]. Additionally, AlphaGo Zero even produced novel gameplay strategies that were previously unknown to human players [10].

However, while these formalizations, methods, and results have been instrumental in progressing the field of RL, the games they have been applied to are largely restricted to competitive, zero-sum settings [11]. In contrast, most real-world examples as well as more complex games typically involve, at least partially, elements such as communication, coordination, imperfect information, and joint decision-making. This can introduce additional complexities to learning in the purely competitive setting, as agents can benefit from and thus need ways to process and model other agents' behavior and gameplay strategies.

For instance, Lerer et al. [12] abstracted the cooperative game setting into two distinct problem formulations: (1) by converting it into a regime where all other players play by a predetermined policy except for a single agent, and (2) a multi-agent setting where each player learns their own policy and only falls back to the predetermined policy if it fails to find the optimal policy. The authors proved that both formalizations achieved performance levels at least as high as the predetermined baseline policy. Taking a different approach, work by Schroeder de Witt et al. [13] has shown that in multi-player settings, being able to share common knowledge across players can facilitate learning and improve task performance. By modeling gameplay as a decentralized partially observable Markov decision process (Dec-POMDP), the authors introduced an algorithm (MACKRL) that enabled groups of players to utilize their common knowledge to collectively choose actions while still learning a decentralized policy. They showed that this method not only beats baselines but also the state-of-the-art when applied to a matrix game and StarCraft. Dec-POMDP has now become foundational in how we model similar cooperative environments.

Recently, the concept of Theory of Mind (ToM) has been studied in the context of RL for cooperative gameplay. ToM refers to the ability to infer the mental states – that is, beliefs, intents, knowledge, etc. – of others and how they influence interactions and behavioral outcomes. This is built upon the understanding that other people may hold beliefs, desires, and intentions that are different from one's own, which can lead to different perceptions of shared contexts as well as different action choices [14, 15]. In the context of RL, integrating ToM allows agents to reason about the beliefs and intentions of other agents within a shared environment, which is crucial for effective collaboration and coordination in cooperative settings, something that has not yet been thoroughly explored [16].

We explore this in the game of Hanabi, a cooperative card game where players have limited information about the environment (i.e. hidden cards) and thus must work together (e.g. through hints) to fill in these gaps in order to determine what the optimal action should be. In our work, we design training objectives to support ToM reasoning in self-play Hanabi, training each agent to predict its own cards and bootstrapping these predictions to model beliefs of other players' cards (higher order beliefs). By incorporating ToM reasoning into the learning process, our goal is to evaluate how ToM reasoning influences performance outcomes in this cooperative setting.

2 Hanabi

Hanabi (from Japanese 花火, *fireworks*) is a cooperative card game: the players aim to maximize a common score. Crucially, a player can see everyone's cards but their own, making Hanabi a decentralized partially observable Markov decision process (Dec-POMDP) [13].

2.1 Game Rules

A card (r, c) has a rank $r \in [5]$ and a color $c \in \{\text{red}, \text{yellow}, \text{green}, \text{white}, \text{blue}\}$. For each color, there are 10 cards with respective ranks $[1, 1, 1, 2, 2, 3, 3, 4, 4, 5]$. The game begins with a shuffled deck of the 50 cards face down, 8 hint tokens, and 3 life tokens. The tokens are shared by all players. Each player is dealt 5 cards (if there are 2-3 players) or 4 cards (if there are 4-5 players). As mentioned above, what is unique about Hanabi is the fact that each player cannot see the cards in their own hand, but can only see the cards of the other players. This makes cooperation, through strategies such as providing hints to other players, a core part of Hanabi gameplay.

The players take turns playing in clockwise order, starting with an arbitrary player. On a player's turn, she may play a card, discard a card, or give another player a hint. If she plays or discards a card, it goes to the field or the graveyard, and she draws a card from the deck unless it is empty. Cards in the field and graveyard are visible to all players.

If she plays a card (r, c), it goes to the field (a "successful play") if the cards in the field with color c are exactly the $r - 1$ cards with rank $< r$. Otherwise it goes to the graveyard and a life token is lost. If a rank 5 card was successfully played, a hint token is gained up to a maximum of the initial 8.

If she discards a card, it goes to the graveyard and a hint token is gained; if all 8 are available, she cannot discard.

If a player gives a hint, a hint token is lost. If none is remaining, she cannot give a hint. To give a hint, the player must choose another player and a rank or color, and announce the set of his cards (referred to by their position in his hand) with that rank or color. This set cannot be empty.

The game ends when one of the following occurs: (1) all life tokens are lost, (2) the last card in the deck is drawn and each player goes one more time, (3) a rank 5 of each color is

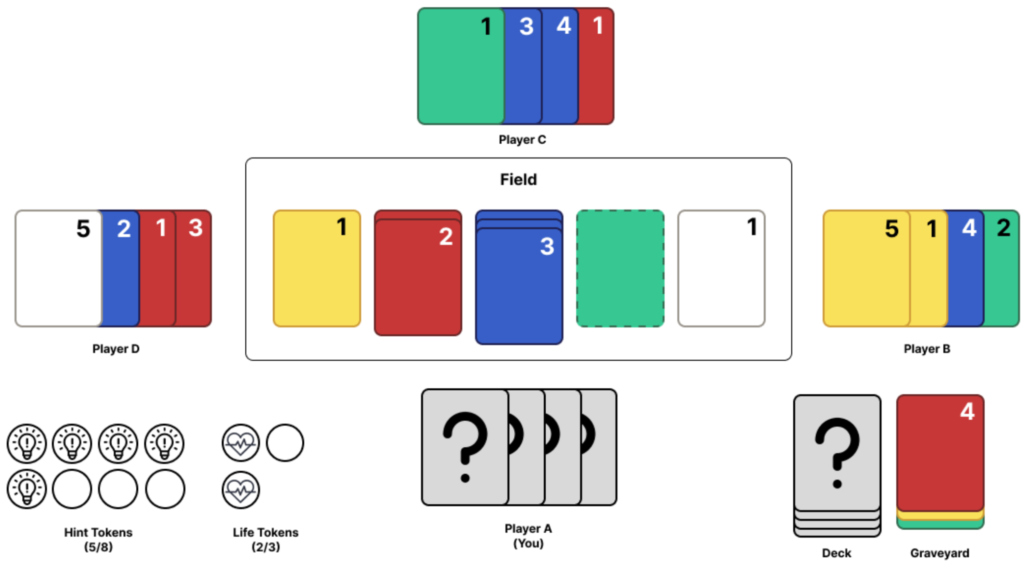

Figure 1: An illustration of Hanabi gameplay with 4 players, from the perspective of Player A. Cards that are grayed out signify those that Player A cannot view. All colors have a card on the field except for green. The players have used up 3 hint tokens and 1 life token.

in the field. In the first case, the score is 0. Otherwise, the score is the number of cards in the field (maximum of 25). See figure 1 for an illustration of a 4-player Hanabi game.

2.2 Reinforcement Learning Formulation

In the reinforcement learning formulation of the game, a player is an agent. In our experiments, we consider a two-player self-play setting, where the agents are copies of each other (illustrated in Figure 2). We discuss the ad-hoc setting in Section 6.

The state consists of the current player index, player hands, field, graveyard, number of life tokens, number of hint tokens, and number of cards in the deck.

The state-derived observation would be the state but without the current player's hand. However, following Bard et al. [17]'s Rainbow agent, we augment the observation with history information. We call the "snapshot" observation, which is the state-derived observation plus the previous player's action and the hint information about the current player's hand. Then the (augmented) observation is the concatenation of the agent's most recent hs snapshot observations, for a history size hs. Note that this augmentation still does not include the sequence of actions beyond the previous action, which could be vital for coordinating strategies. While further augmenting the observation may lead to improvements, this is not the focus of our project, so we retain consistency with Bard et al. [17]'s Rainbow baseline.

The possible actions are playing a card, discarding a card, and giving a hint. Since there are 2 players and thus 5 cards in each hand, and there are 5 ranks and 5 colors, the number of possible actions is at most $5 + 5 + (2 - 1)(5 + 5) = 20$. (Some of the play/discard moves will be invalid if there are no cards left in the deck to replace a missing card. Some of the hint moves will be invalid if the other player does not have cards of certain attributes.)

The transition probabilities obey the game rules above, where the stochasticity of the environment comes exclusively from the shuffle of the deck.

The reward function is deterministic: 1 after a successful play, negative the prior return after losing the last life, and 0 otherwise. Following Bard et al. [17], we use a discount factor

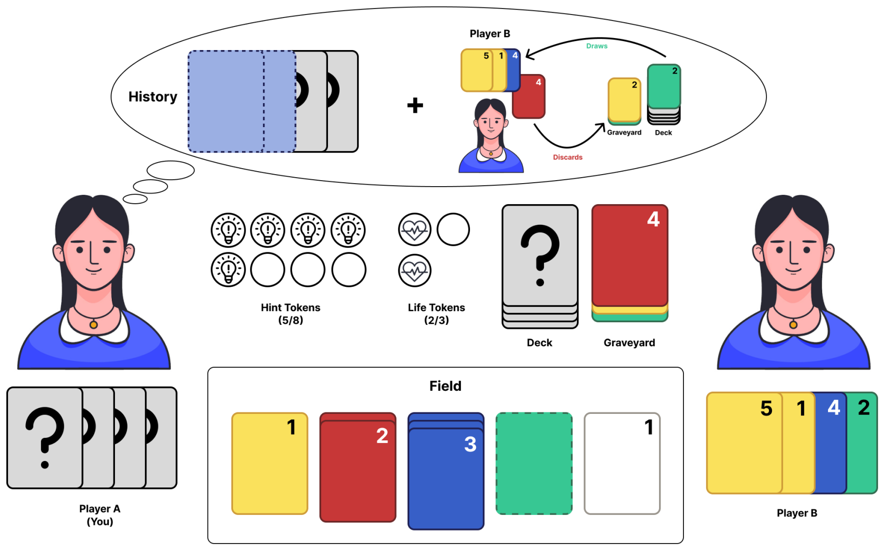

Figure 2: Two player self-play setting. Here, the state space is augmented with the Player B's prior action (discarding Red 4 and drawing Green 2 from the deck) as well as a prior hint that the two left cards in Player A's hand are blue.

of 0.99. However, we note that this choice is inconsistent with the way the game is scored – the final score is simply the total number of successful plays without discounting.

3 Methods

3.1 Baseline

The baseline we use is the Rainbow agent implemented in the Hanabi Learning Environment (HLE) [17]. The Rainbow agent architecture [18] combines improvements to Deep Q-Networks [19], such as distributional Q-learning [20] and prioritized experience replay [21]. The model architecture is a 2-layer feedforward neural network with a hidden dimension of 512 units that inputs the observation and outputs a distribution over the 51 discrete Q values $(-25, \ldots, 25)$ for each action. A replay batch of size 32 is trained on every 4 agent steps. For the observation, we use a history size of $hs = 2$.

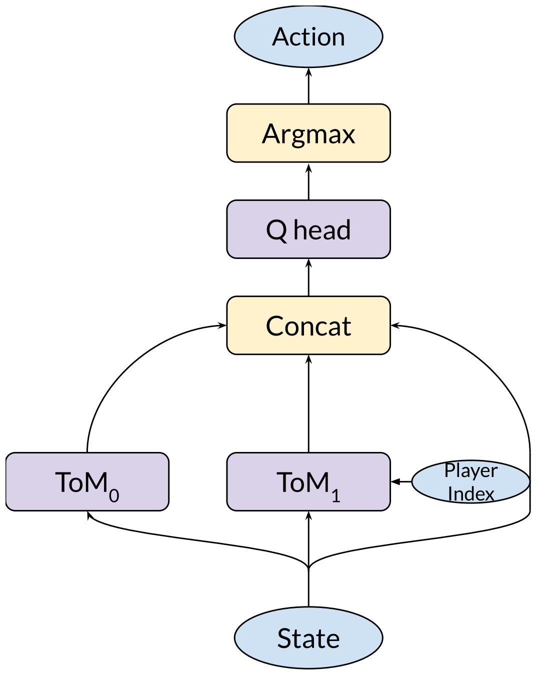

Figure 3: Model architecture at inference.

3.2 Theory of Mind Objectives

Theory of mind (ToM) is a central theme in the game of Hanabi. Especially since communication is limited, players must make moves by inferring the observer's interpretations and interpret moves by inferring the actor's intentions. We formalize this notion recursively with a family of ToM objectives.

Given an agent i and its observation o_i, a model ToM_0 is trained to predict, for each of the agent's cards, a probability distribution over the set of ranks and a probability distribution over the set of colors. Since the agent's true card attributes are known to the creators of the environment, we can train ToM_0 for this task with cross-entropy loss.

For higher orders $k \geq 1$ of ToM, let P be the set of agents, let $p = \{p_i \in P\}_{i \in [k+1]}$ be a sequence of $k+1$ agents, and let o_k and o_{k+1} be the (simultaneous) observations of agents p_k and p_{k+1}. We train $\text{ToM}_k(p_{\leq k+1}, o_{k+1})$ to predict $\text{ToM}_{k-1}(p_{\leq k}, o_k)$. In other words, agent p_{k+1} predicts what agent p_k predicts about what agent p_{k-1} predicts about ... what agent p_1 predicts about his cards. Since we have ToM_{k-1} as computed by the agents, we can train ToM_k again with cross-entropy loss.

As an intuitive compromise between computation and training signal, we use up to order 1 ToM. For example, first order ToM allows for teaching an agent how his hint is interpreted by another agent, if we train ToM_1 on observations before and after he gives the hint.

Figure 3 shows the model architecture at inference. The modules in purple represent a feedforward neural network with a single hidden layer with a dimension of 512, followed by an appropriate softmax. To the ToM_1 module, a player index is given, representing the agent of whom the current agent seeks to predict the order 0 beliefs. (In our 2-player game, there is only one other player.) We refer by the "ToM_k method" to the Figure 3 model where the ToM modules are kept up to order k. The baseline includes none of the ToM modules.

To train the ToM_0 and ToM_1 modules, each replay memory is augmented to include the agent's own cards and the other agent's order 0 beliefs from the previous turn. The outputs of the ToM heads are used to compute their loss functions, and the gradient is stopped between the ToM modules and the Q head.

4 Experiments

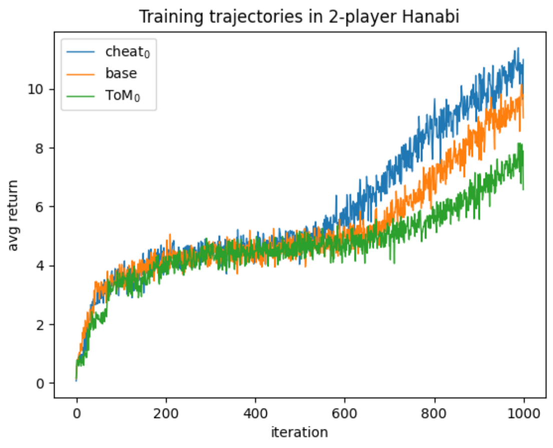

Figure 4: The average of the final 50 points are 10.56 for cheat, 9.26 for base, and 7.39 for ToM_0.

Figure 4 shows the training trajectories for methods built on the Rainbow agent implemented in HLE [17]. Training and experiment runs were executed in an environment we built based on the HLE codebase provided by Bard et al. [17] with the modifications described in sections 2.2 and 3. The agents are trained for 1000 iterations, each with 5000 agent steps. Every 4 agent steps, a replay batch of size 32 is trained on those steps. The base method is the original Rainbow agent, and the cheat_0 method is the ToM_0 method but with the ToM_0 module prediction replaced with the agent's true hand. We consider cheat_0 an upper baseline, since playing with all cards visible is trivial for human players. We find that ToM_0 underperforms the baseline, and provide discussion in Section 6.

Due to compute limitations, we were unable to finish the experiment for ToM_1 and we were unable to run methods many times for significance testing. Running the base Rainbow model for 1000 iterations ($5 \cdot 10^6$ agent steps) takes ~25 GPU hours on an A40, and our models take longer due to the additional model components. Training for at least 500 iterations seems necessary, since the distinction between the baseline and cheat_0 does not become clear until beyond that point.

5 Related Work

Hanabi self-play: Existing work has explored Hanabi as a multiagent problem in the self-play regime, where the RL agent learns by playing against itself [17, 22, 12, 11]. Early work used the online, decision-time planning algorithm Monte Carlo Tree Search (MCTS) to produce optimal gameplay actions [22, 23]. For example, Goodman [22] won the Conference on Games competitions in 2018 and 2019 utilizing IS-MCTS [24] as its base action-selection algorithm and using neural networks to model the policy and value function. On the other hand, Fickinger et al. [25] found that tabular RL methods such as SPARTA (Simultaneous Policies, Actions, Rewards, and Trajectories for Actors) [12] can quickly become intractable when performing multiple-step lookahead searches. Instead, they proposed an inference time policy-tuning method based on Q-Learning, which was able to significantly outperform the tabular SPARTA method in multi-agent settings and perform lookahead searches when computating resources are sufficient. Hu and Foerster [11] developed a *Simplified Action Decoder* (SAD) that allows agents to balance exploration and exploitation in the group setting by observing both the greedy and exploratory actions made by their teammates in a 2-5 player self-play regime. They observed performance improvements over the baselines from Bard et al. [17] and the Bayesian Action Decoder agent [26], a state-of-the-art in the two-player setting.

Ad-hoc play in Hanabi: Subsequent research has also looked at ad-hoc methods [27, 28] to address limitations in the self-play setting. Examples include sub-optimal or incompatible policies when extended to real-world cooperative gameplay, such as an overdependency on initial values or overfitting to policies that are optimized for playing against agents previously encountered (i.e. themselves) [17, 29, 30, 31, 32, 33]. For instance, Canaan et al. [29] utilize behavioral features (MAP-Elites) to generate a set of initial policies as opposed to using a neural network as in Goodman [22]. However, the results have been mixed. Bard et al. [17] found that even using actor-critic deep RL techniques with an Importance Weighted Actor-Learner yielded worse performance than rule-based methods, and in ad-hoc settings outright failed to cooperate.

Proximal Policy Optimization (PPO) methods [34] have become the state-of-the-art reinforcement learning algorithm, yet have not been popular in multi-agent scenarios due to a common critique that it lacks sample efficiency in comparison with off-policy approaches. However, Yu et al. [30] found that applying PPO to the cooperative ad-hoc setting in fact resulted in increased sample efficiency and average returns, particularly in configurations with more than 2 players.

Recent work has also approached the ad-hoc setting through the lens of zero-shot coordination (ZSC) – that is, playing with other players (either AI or human) the agent has never encountered before [32, 31]. As a starting point, Hu et al. [32] extended the self-play setting by explicitly considering the possibility of coordinated symmetry breaking in group settings – where multiple maximal policies can exist simultaneously – into the policy objective function. They found that using this method, which they call *other-play* (OP), agents were able to achieve higher game scores than the self-play state-of-the-art when playing with humans. However, if these symmetries are unknown or cannot be obtained, OP becomes yet another self-play problem, and resulting policies may fail to be optimal in the multi-agent setting. Instead, if we have access to the entire state-action trajectory, generalizing the

Jensen-Shannon divergence across policies can allow us to train agents to produce the *best response* strategy given the strategies of other players in the multi-agent setting, as Lupu et al. [31] have shown.

Other extensions and strategies: Other work has also explored Hanabi gameplay in the human-AI collaboration setting, using natural language instructions provided by humans as a way to align the RL agents to strategies that humans desired [35]. Here, the authors use LLMs to generate a prior policy based on initial human instructions and a natural language description of feedback from policy rollouts in the environment. This is then used as the reference policy to train the RL agent. Hu et al. [33] explore the concept of *self-explaining deviations* (SEDs), or actions that do not follow commonly-accepted behavior, as a strategy of Hanabi gameplay (e.g. hinting strategy) to influence other players to take certain actions.

6 Discussion

6.1 Results

We discuss our hypotheses for the poor performance of ToM_0 compared to the baseline in Figure 4.

Upon inspecting some episodes of the ToM_0 method after training for 1000 iterations, we found that the ToM_0 predictions can largely be explained by the hints explicitly given. In other words, at the stage where the average return is under or about 10, they have not developed complex protocols and thus the most reliable signal for ToM_0 is explicit hints. This is partially explained by not training for long enough and by the exclusion of the ToM_1 objective for predicting how one's actions affect other agents' beliefs, but there is a deeper reason in the observation, discussed next. As for why we see loss in performance rather than mere retention, since the concatenation of the ToM_0 with the state increases dimensionality, learning becomes harder.

As described in Section 2.2, the observation does not include the sequence of actions beyond the previous action, which could be vital for coordinating strategies. For example, with human players, a standard protocol is the following. If a player was earlier told that two of his cards are red, and he is now told that one of them is a 4, he should infer that the other red is playable. For example, if the red stack in the field has rank 1, he infers that the non-4 red is a 2 and can be played. The employment of such protocols is necessary for efficient information conveyance, but it requires considering many moves into the past. Although with human players it is common knowledge [36] what moves have been made, agents cannot act on them, simply because the observation does not encode them. Crucially, including the ToM_1 objective does not resolve this, since giving the above player the hint that one of his reds is a 4 is only appropriate if one knows that he knows that he has two reds, which requires memory of a previous hint.

6.2 Other Methods Tried

We describe some of the other methods we tried. A common approach for training auxiliary objectives thought to be relevant to the desired task is attaching heads (small models) to the end of a shared base model and using the head outputs to compute the loss functions for the auxiliary objectives [37, 38]. A head should be small compared to the base model, so most of the learning occurs in the weights of the base model, and the heads can be discarded for downstream tasks. We tried this approach, where the hidden state of size 512 branches to the different heads for the Q function, ToM_0, and ToM_1. However, it did not work well, which we reasoned to be because the shared model is roughly the same size as, rather than much larger than, each head. Thus, it cannot be assumed that training the ToM heads and discarding them at inference will improve the model for the Q function.

We initially posed the ToM_0 objective as predicting a ternary classification task for each attribute for each card in one's hand, with options (1) attribute present, (2) attribute absent, (3) abstain. For a prediction with confidence c on an option 1 or 2, a correct prediction has loss $-\log_2(1+c)$ and an incorrect prediction has loss $-\log_2(1-c)$. We scale the loss for option 2 by $\frac{1}{5}$ compared to option 1 to account for the label imbalance due to there being 5 colors and 5 ranks. The motivation for these expressions is that, for a correct prediction, the loss decreases roughly linearly with increasing confidence, and for an incorrect prediction, the loss approaches infinity as the confidence approaches 1. However, we derived that this ToM_0 objective reduces to cross-entropy loss on the binary classification task without an abstain option.

With the binary classification ToM_0 task, we found that after training for 1000 iterations, predicted probabilities are quite extremal (close to 0 or 1) and even include near-1 predictions for multiple colors and multiple ranks. Thus, we changed the objective to a 5-way color or rank classification task, trained still with cross-entropy loss.

We tried benchmarking in a smaller environment (with fewer colors and ranks) to compare methods while using less compute, but we found a pattern similar to the full environment, where even the cheating upper baseline does not outperform the baseline until after several hundred iterations.

6.3 Future Work

If the base model is scaled up, it would be more reasonable to adopt the shared base model approach. The advantages of this would be that (1) the number of new parameters to train for auxiliary objectives is small and (2) the inference cost is unaffected since the training signal has been embedded into the base model weights and the heads can be discarded. As a downside, the objectives would no longer be gradient-wise independent, and so jointly optimizing them requires choosing hyperparameters to weight their loss functions. Also, we did not modify the priority replay sampling in the base Rainbow, so the ToM objectives do not affect the priorities, but they should if they are incorporated into a combined loss function.

In this project, we have considered the self-play setting, where the agents are copies of each other. The other setting is ad-hoc play, where agents that were trained separately play together. The ad-hoc setting is difficult for agents that share no training and no significant prior. Although independently trained agents will likely develop incompatible protocols, it is reasonable to aim to train agents that can adapt to each other's protocols by playing a small number of games together. The ToM objectives we proposed have the potential to boost fine-tuning for ad-hoc play, since agents directly learn to predict other agents' beliefs, allowing them to calibrate their intentions with other agents' inferences. Furthermore, such fine-tuning may serve in learning to play with human players too, since humans can be expected to estimate their beliefs up to at least order 1 ToM.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[3] G. Tesauro. Temporal difference learning of backgammon strategy. In *Machine Learning Proceedings 1992*, pages 451–457. Elsevier, 1992.

[4] G. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219, 1994.

[5] G. Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[6] G. Tesauro. Programming backgammon using self-teaching neural nets. *Artificial Intelligence*, 134(1-2):181–199, 2002.

[7] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[8] A. L. Samuel. Some studies in machine learning using the game of checkers. ii—recent progress. *IBM Journal of research and development*, 11(6):601–617, 1967.

[9] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu. Deep blue. *Artificial intelligence*, 134 (1-2):57–83, 2002.

[10] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[11] H. Hu and J. N. Foerster. Simplified action decoder for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1912.02288*, 2019.

[12] A. Lerer, H. Hu, J. Foerster, and N. Brown. Improving policies via search in cooperative partially observable games. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7187–7194, 2020.

[13] C. Schroeder de Witt, J. Foerster, G. Farquhar, P. Torr, W. Boehmer, and S. Whiteson. Multi-agent common knowledge reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[14] S. Baron-Cohen. *Mindblindness: An essay on autism and theory of mind*. MIT press, 1997.

[15] L. J. Byom and B. Mutlu. Theory of mind: Mechanisms, methods, and new directions. *Frontiers in human neuroscience*, 7:413, 2013.

[16] A. Fuchs, M. Walton, T. Chadwick, and D. Lange. Theory of mind for deep reinforcement learning in hanabi. *CoRR*, abs/2101.09328, 2021. URL https://arxiv.org/abs/2101.09328.

[17] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.

[18] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning, 2017.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. URL https://api.semanticscholar.org/CorpusID:205242740.

[20] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

[21] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay, 2016.

[22] J. Goodman. Re-determinizing information set monte carlo tree search in hanabi. *CoRR*, abs/1902.06075, 2019. URL http://arxiv.org/abs/1902.06075.

[23] J. Walton-Rivers, P. R. Williams, R. Bartle, D. Perez-Liebana, and S. M. Lucas. Evaluating and modelling hanabi-playing agents. In *2017 IEEE congress on evolutionary computation (CEC)*, pages 1382–1389. IEEE, 2017.

[24] P. I. Cowling, E. J. Powley, and D. Whitehouse. Information set monte carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2): 120–143, 2012.

[25] A. Fickinger, H. Hu, B. Amos, S. Russell, and N. Brown. Scalable online planning via reinforcement learning fine-tuning. *Advances in Neural Information Processing Systems*, 34:16951–16963, 2021.

[26] J. Foerster, F. Song, E. Hughes, N. Burch, I. Dunning, S. Whiteson, M. Botvinick, and M. Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 1942–1951. PMLR, 2019.

[27] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1504–1509, 2010.

[28] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 567–574, 2011.

[29] R. Canaan, X. Gao, J. Togelius, A. Nealen, and S. Menzel. Generating and adapting to diverse ad-hoc partners in hanabi. *IEEE Transactions on Games*, 2022.

[30] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[31] A. Lupu, B. Cui, H. Hu, and J. Foerster. Trajectory diversity for zero-shot coordination. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7204–7213. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/lupu21a.html.

[32] H. Hu, A. Lerer, A. Peysakhovich, and J. Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

[33] H. Hu, S. Sokota, D. Wu, A. Bakhtin, A. Lupu, B. Cui, and J. Foerster. Self-explaining deviations for coordination. *Advances in Neural Information Processing Systems*, 35: 38400–38410, 2022.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

[35] H. Hu and D. Sadigh. Language instructed reinforcement learning for human-ai coordination. In *International Conference on Machine Learning*, pages 13584–13598. PMLR, 2023.

[36] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*, volume 1 of *MIT Press Books*. The MIT Press, December 1994. ISBN ARRAY(0x5679b980). URL https://ideas.repec.org/b/mtp/titles/0262650401.html.

[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[38] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.