HOAD: The Hanabi Open Agent Dataset

Extended Abstract

Aron Sarmasi, Timothy Zhang, Chu-Hung Cheng, Huyen Pham, Xuanchen Zhou, Duong Nguyen, Soumil Shekdar, Joshua McCoy
University of California, Davis
{asarmasi,mtzh,chccheng,hdupham,xuczhou,mdnnguyen,sshekdar,jamccoy}@ucdavis.edu

ABSTRACT

In this work we present the Hanabi Open Agent Dataset (HOAD)—meant to address the current lack of Hanabi datasets, HOAD is an easily extensible, open-sourced, and comprehensive collection of existing Hanabi playing agents, all ported to the Hanabi Learning Environment (HLE). We give a description and analysis of each agent's strategy, and we also show cross-play performance between all the agents, demonstrating both their high quality and diversity of strategy. These properties make HOAD especially well suited to studies involving meta-learning and transfer learning. Finally, we describe in detail an easy way to add new agents to HOAD regardless of the origin codebase of the agent and make our code and dataset publicly available at https://github.com/aronsar/hoad.

KEYWORDS

Hanabi; dataset

ACM Reference Format:
Aron Sarmasi, Timothy Zhang, Chu-Hung Cheng, Huyen Pham, Xuanchen Zhou, Duong Nguyen, Soumil Shekdar, Joshua McCoy. 2021. HOAD: The Hanabi Open Agent Dataset: Extended Abstract. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 3 pages.

1 INTRODUCTION

Hanabi [2] is a tabletop card game for 2-5 players[1], notable for its unique combination of partial observability, cooperation, stochasticity, and implicit communication. Recently proposed as a challenge domain [1] to provide a sophisticated yet well-defined set of challenges for artificial intelligence practitioners, the game is of rising interest to the research community.

In a game of Hanabi, the players work together to assemble five piles of cards, where each pile is a different color, and consists of cards numbered 1 through 5, in that order. The defining feature of the game is that players only see their teammates' cards, and not their own. On their turn, a player may either give a teammate a hint about the color or value of their cards (a limited resource), discard a card (doing so regains a hint, but there are a limited number of copies of each card), or play a card they believe is playable. See [8] for the complete rules. To be successful, a group of players must correctly interpret each others' explicit communication—through hints—as well as each others' implicit communication—through playing cards, discarding, or even hinting (e.g. the finesse play [11]). This results in a multiplicity of optimal and near-optimal strategies, each corresponding to a communication schema.

This aspect of the game is one of the most interesting from a research perspective. Indeed, it recently inspired the Ad-Hoc Challenge, [1], the crux of which is to figure out the communication strategy of a held-out agent given only ten of its games, and then successfully play games with it. However, to the best of our knowledge, there does not exist a dataset of Hanabi playing agents that can facilitate this kind of research. We thus propose the Hanabi Open Agent Dataset (HOAD), a compilation of different agents from different sources, most using conventions typically seen in human play, and all operating within the same environment and using a binarized game state representation specifically designed for neural learning. The dataset allows pairwise play between any two agents, offers an easy way to add additional agents from most codebases, and also gives access to the Dopamine reinforcement learning framework [4], which we plan to use in future work to further improve and extend the HOAD agents.

Table 1: We recorded 500,000 games of each original agent into the HLE representation, and then used them to learn multilayer perceptron (MLP) imitators of each agent.

Agent	Original Self Play Score	Imitator Self Play Score	Imitator Accuracy
Simplebot	16.9	16.8	99.7
Valuebot	19.8	18.0	92.0
Holmesbot	20.8	14.7	90.3
Outer	14.5	14.1	66.7
Iggi	17.0	16.2	90.9
Piers	17.3	15.9	85.8
Rainbow	18.5	18.1	77.5
Van-Den-Bergh	14.0	10.5	81.2

2 THE HANABI OPEN AGENT DATASET

To create HOAD, we conducted a comprehensive review of existing implementations of agents that play Hanabi and ported those that met our criteria. We looked specifically for agents that had codebases available online, scored reasonably well in self-play (above 10 points), and employed strategies sufficiently different from one another, such that when two different agents played one another they performed significantly worse than either did in self-play. As a result, the variation of strategy among the agents is ensured. This disqualified a number of derivate works that were slight modifications on existing agents, as well as the Actor Critic Hanabi Agent

[1] in this work we consider only the two-player scenario for simplicity, leaving 3-5 players to future work

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

(ACHA) [6], which does not have a public codebase, and the agents proposed by [3], which released their codebase concurrently to this work. The FireFlower [10] and BAD [5] agents were also considered, but were not included due to technical difficulties; we plan to add them in future work.

Although some authors of Hanabi playing agents have published multiple agents using the same framework, in general, two arbitrary agents taken from different codebases will not be able to play one another because of differences in implementation in how the agents represent the game state. One naive way to enable agents from multiple different codebases to play one another is to rewrite the game logic of every agent using the same environment and game state representation. This however is prohibitively time-consuming, as some agents are comprised of several thousand lines of game logic. Another method would be to observe the game states of each agent in its native representation and save all the game states to some common representation. In our experience, this was as difficult as the first approach due to the variation in implementation among authors and the size and complexity of the game state (there are 658 binary variables in the game state representation of a 2-player game).

The workaround that made HOAD possible (and which is responsible for its ease of extensibility) is to observe the starting deck order and the actions taken by agents in their native environments. These observations are then used to recreate the games in the Hanabi Learning Environment (HLE). Observing the starting deck order and the actions taken is a much simpler task because the set of legal actions is small (≤ 20 in a 2-player game), and the ordering of the deck is typically known by the respective game engine at the start of the game. Once replay data has been gathered for all the agents in the HLE representation, it is possible to train a neural network to imitate each of the original agents. The accuracy of these imitators is presented in Table 1.

2.1 HOAD Agent Strategy Summaries

We present a summary of each agent in HOAD below, including only necessary detail. A compilation of commonly employed strategies, used both by human players and HOAD agents, can be found at [11].

Simplebot [7] – Plays only cards which it has enough knowledge about to know they are playable, and gives hints only about playable cards, preferring color hints over value hints. It uses an oldest first discarding strategy.

Valuebot [7] – Same as Simplebot, but before playing, checks to see if the next player is about to discard a valuable card (i.e. the last copy of a card).

Holmesbot [7] – Extends Valuebot by including the use of mulligans and by adding additional inference capabilities. Specifically, card knowledge from hints, the discard pile, and other players' hands are used to make deductions about the agent's hand.

Iggi [9] – Similar logic to Simplebot but prefers value hints over color hints and prefers discarding unplayable cards over oldest first.

Outer [9] – Similar to Iggi, but prefers discarding over hinting, and uses more randomness in its hinting and discarding logic; this results in significantly reduced imitation accuracy and is also likely the reason for lower published score.

Table 2: Pairwise play scores are produced by playing each MLP imitator agent with every other for 500 games.

First player	Simplebot	Valuebot	Holmesbot	Outer	Iggi	Piers	Rainbow	Van-Den-Bergh
Simplebot	16.8	15.7	12.8	0.0	4.1	1.1	1.7	0.2
Valuebot	15.2	18.0	17.6	0.0	3.8	1.3	2.0	0.0
Holmesbot	11.2	18.3	14.7	0.0	1.4	0.6	0.5	0.0
Outer	0.0	0.0	0.0	14.1	1.0	4.1	6.2	9.0
Iggi	3.9	3.8	1.8	1.8	16.2	11.8	2.7	6.0
Piers	1.8	0.5	0.2	7.2	10.3	15.9	5.5	9.4
Rainbow	0.3	2.0	0.3	6.6	4.0	5.6	18.1	2.6
Van-Den-Bergh	0.0	0.2	0.0	10.6	4.7	8.3	2.7	10.5

Piers [9] – Extends Iggi by including the use of mulligans (but not as deterministically as Holmesbot), and some additional logic to avoid discarding valuable cards.

Van-der-bergh [9] – Makes some risky plays if they have high likelihood of success and there are remaining mulligans. Prioritizes discarding over hints, gives hints about useless cards, and attempts to maximize transmitted information.

Rainbow [1] – This agent tends to hint for rank instead of color. Conditional action probabilities may be found in the appendix of [1].

2.2 Discussion of Imitator Agent Pairwise Scores

Since the same observation can be passed in to any of the imitator agents, the imitator agents make it possible to play games with agents originating from two different codebases; this is necessary to evaluate how different the players are from each other. Table 2 shows the average score of each imitator agent playing 500 games with every other imitator agent. As expected, when an agent is paired with itself, it typically achieves a much higher score than if it were paired with any other agent—this corresponds to the high scores on the diagonals and the relatively low scores on the off-diagonals. This confirms the intuition that agents must use a similar strategy when paired with one another, else risk miscommunicating, losing all three lives, and scoring zero points.

An interesting feature of Table 2 warranting discussion is the high scores achieved by certain combinations of agents. Two groups of agents that play relatively successfully with each other are Simplebot – Valuebot – Holmesbot, and Iggi – Piers – Outer – Van-der-bergh – Rainbow. All the agents in each of these two groups belong to the same codebase, so the high scores are likely due to the authors of the two codebases reusing logic between their agents. The exception to this is the Rainbow agent, which was produced using reinforcement learning, and so it is surprising that it performs so well with Walton-Rivers' agents. Our best explanation is that both agents reportedly prefer value hints and can presumably also respond well to game states where value hints have been given.

REFERENCES

[1] Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. 2019. *The Hanabi Challenge: A New Frontier for AI Research.* Technical Report. arXiv:1902.00506 http://arxiv.org/abs/1902.00506

[2] Antoine Bauza. 2010. Hanabi. https://www.boardgamegeek.com/boardgame/98778/hanabi

[3] Rodrigo Canaan, Julian Togelius, Andy Nealen, and Stefan Menzel. 2019. Diverse Agents for Ad-Hoc Cooperation in Hanabi. Institute of Electrical and Electronics Engineers (IEEE), 1–8. https://doi.org/10.1109/cig.2019.8847944 arXiv:1907.03840

[4] Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, Marc G Bellemare, and Google Brain. 2018. Dopamine: A Research Framework For Deep Reinforcement Learning. In *unpublished*. arXiv:1812.06110v1 https://github.com/google/dopamine

[5] Jakob N Foerster, H Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew M Botvinick, and Michael Bowling. 2019. Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning. In *Conference on Machine Learning*. arXiv:1811.01458v3

[6] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. *JMLR* 48 (2016). arXiv:1602.01783v2

[7] Arthur O'Dwyer. 2018. Github - quuxplusone/hanabi: Framework for writing bots that play Hanabi. https://github.com/Quuxplusone/Hanabi

[8] Aron Sarmasi, Timothy Zhang, Chu-Hung Cheng, Huyen Pham, Xuanchen Zhou, Duong Nguyen, and Soumil Shekdar. 2020. Hanabi Open Agent Dataset. https://github.com/aronsar/hoad

[9] Joseph Walton-Rivers, Piers R. Williams, Richard Bartle, Diego Perez-Liebana, and Simon M. Lucas. 2017. Evaluating and Modelling Hanabi-Playing Agents. In *CEC*. 1382 – 1389.

[10] D Wu. [n.d.]. GitHub - lightvector/fireflower: A rewrite of hanabi-bot in Scala. https://github.com/lightvector/fireflower

[11] James Zamiell. [n.d.]. GitHub - Zamiell/hanabi-conventions: A list of Hanabi strategies. https://github.com/Zamiell/hanabi-conventions