ENTITY-WISE FACTORIZATION ENABLES GENERALIST AGENTS IN HANABI

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-play lets multi-agent reinforcement learning agents cooperate well with copies of themselves, but the arbitrary conventions they invent along the way rarely transfer to a new partner, human or otherwise. This is exemplified in Hanabi, a cooperative card game for 2 to 5 players widely used to probe theory of mind. Until now, n-player game has been considered separately, with multiple works focusing only on 2- or 3-player games. In this work, we propose a generalist Hanabi agent that can play and be trained on any n-player game jointly. To enable this, we require a common state representation agnostic to the number of players. Inspired by UPDeT (Hu et al., 2021b), a successful example of factorizing an observation into per-entity tokens so that one set of parameters can handle a variable number of entities, we propose FHA (factorized Hanabi agent), a simplified and fast factorization architecture. We then perform a comprehensive analysis of self-play, other-play and off-belief learning trained with our proposed architecture, jointly and independently across players. In doing so, we observe that factorization and joint training can improve cross-play performance of self-play agents, because the agent learns a general policy that it applies regardless of the number of players. Thus, by avoiding the overspecialization issues encountered by self-play agents, self-play can be a viable training method for cross-play agents. Finally, the learned policy is transferable, as FHA is able to train on a subset of settings and zero-shot transfer at deployment to unseen settings.

1 INTRODUCTION

Cooperative multi-agent reinforcement learning is usually trained and evaluated in a single fixed setting: a fixed number of agents, trained together, tested together. Self-play makes this convenient, since an agent only has to coordinate with copies of itself, but it also lets the agent settle on arbitrary conventions that are not shared by any other agent. This gap between self-play competence and coordination with unfamiliar partners has been formalized as the zero-shot coordination (ZSC) problem: training a policy that coordinates well with independently trained partners it has never seen, without further fine-tuning (Treutlein et al., 2023). Existing ZSC methods address this at varying cost: Other-Play (Hu et al., 2020) requires known symmetries of the game to be identified in advance, hierarchical methods such as K-level reasoning (Cui et al., 2021) and Off-Belief Learning (Hu et al., 2021a) require multiple rounds of iterative training and, for OBL, an explicit belief model at each level, and population-based methods (Lupu et al., 2021; Nekoei et al., 2021) require training against a large, diverse pool of partners.

Hanabi is a natural place to study this (Bard et al., 2020). It is a fully cooperative card game played by 2 to 5 players under partial observability, and it is widely used as a benchmark for cooperative MARL and theory of mind. Yet existing Hanabi agents are built and evaluated for one fixed player count and one fixed set of teammates. The joint action and belief space grows combinatorially with each additional player, and training separate agents per player count multiplies this cost further, so most Hanabi agents, including Off-Belief Learning and Other-Play, are trained and evaluated on 2 players (occasionally 3), while 4 and 5 players are rarely trained on at all. Sharing one set of parameters across all player counts would reduce this cost, but it is far from trivial. The architectures commonly used in cooperative MARL, and in Hanabi in particular, stack feed-forward and recurrent layers that expect an input of fixed dimension and produce an output of fixed dimension. In Hanabi, both depend on the number of players. A network built for the 2-player game therefore cannot even

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

be evaluated on the 5-player game, let alone play it well. The broader Dec-POMDP literature, most prominently StarCraft micromanagement, has addressed variable numbers of agents with architectures that factorize the network across players or entities, so that its size does not depend on how many there are. Examples include pooling player-specific embeddings into a permutation-invariant representation (Hao et al., 2023), using attention mechanisms to aggregate information from all players (Park et al., 2025; Hu et al., 2021b), and (partially) sharing parameters across agents, e.g., through hypernetworks (Tessera et al., 2025). These architectures have been shown to transfer across team sizes, but they have not been brought to Hanabi. To the best of our knowledge, every Hanabi agent to date is trained for a single, fixed player count.

In this work, we propose FHA, a factorized Hanabi agent and, to our knowledge, the first Hanabi agent that can be trained jointly on all game settings, from 2 to 5 players, with a single set of parameters. We restructure the observation to separate information that is invariant to the number of players from information specific to each partner. The same encoder then processes every partner-specific block and every partner-specific action. The number of parameters is therefore independent of the number of players, and the same network can be deployed in any setting, including ones never seen during training. Unlike UPDeT, which routes recurrence through self-attention over all entities at every timestep, FHA handles temporal recurrence and cross-player mixing separately. This keeps the computation simple and lets it use fused recurrent kernels, making it considerably faster in wall-clock time (Appendix A). Trained with self-play jointly across all player counts, FHA matches the self-play score of specialist agents trained separately for each setting, and consistently outperforms UPDeT trained in the same multi-setting fashion. The learned policy also transfers: FHA trained on a subset of player counts plays zero-shot in unseen ones.

Beyond self-play performance, we find that training across all game settings makes FHA substantially better at cross-play. This matters because, in Hanabi, cross-play is notoriously difficult. Players cannot see their own cards and can only communicate through a small budget of hints, so success depends on modeling what teammates know, what they intend, and how they will interpret a hint, which is a form of theory of mind. A self-play agent can meet this requirement trivially by agreeing with its copies on arbitrary signalling conventions, such as "a color hint on the leftmost card means *discard it*." Such conventions are uninterpretable to any agent not trained alongside it, so cross-play performance collapses even between agents that each achieve near-perfect self-play scores (Hu et al., 2020; 2021a). The factorization literature has not examined this question either. Its learning algorithms, typically centralized training with value decomposition, maximize the return of a team of co-trained agents, and the resulting policies are only evaluated in self-play. We find that, holding the architecture fixed, FHA agents trained across all player counts coordinate with independently trained partners substantially better than FHA agents trained on a single player count, and they do so consistently across every setting we evaluate. We attribute this to the diversity of the training distribution. An agent that must play every player count with the same parameters cannot tailor its conventions to the quirks of any single setting, and is instead pushed towards strategies that hold regardless of how many teammates it has and where they sit. Even plain self-play can thus yield agents capable of meaningful cross-play, provided the agent is prevented from over-specializing to a single setting.

A residual gap between self-play and cross-play nevertheless remains, even between independently trained seeds of the same FHA configuration. We therefore combine FHA with two zero-shot coordination methods, Other-Play and Off-Belief Learning. However, we observe that, even though these methods can be combined, Other-Play does not improve cross-play, and Off-Belief Learning improves the cross-play of FHA at the first level of its hierarchy, but not at the next. This shows that factorization can interfere with zero-shot coordination methods.

Our contributions are threefold. First, we introduce a player-agnostic architecture that can be trained across every Hanabi player-count setting with a single model, matching the self-play performance of specialists trained separately per setting. Second, we study how multi-task training across player counts affects cross-play and transfer between game settings, a question not examined in the entity-factorization literature we build on. Third, we combine this architecture with Off-Belief Learning, the state-of-the-art zero-shot-coordination method, show that it improves cross-play at the first level of the OBL hierarchy, and identify why this gain does not carry over to subsequent levels.

2 Related work

Entity-wise Factorization and Permutation Structure. A primary challenge in generalist multi-agent architectures is that conventional policies assume fixed-dimensional observations and action spaces, which tie the architecture to a particular number of agents or entities. UPDeT addresses this problem through policy decoupling (Hu et al., 2021b), where observations are decomposed into embedded entity tokens and actions are partitioned into action groups. This allows a single Transformer-based policy to operate across tasks with different team sizes. Iqbal et al. (2021) also exploit entity-wise structure for multi-task generalization, but in a different way. They randomly partition entities and construct an auxiliary value estimate from factors over the resulting subgroups, which encourages recurring interaction patterns to transfer across tasks. Hao et al. (2023) design a permutation-aware policy in which, when the entities are permuted, the outputs for entity-independent actions remain unchanged while those for entity-targeted actions are permuted with their corresponding entities. We apply entity-wise factorization to Hanabi and study both transfer across player counts and cross-play. Our model (FHA) factorizes the observation into one fixed-size token per teammate and processes each token with the same recurrent network, while maintaining a separate hidden state for each teammate. Play and discard values are averaged over teammates and clue values are concatenated across teammates. As a consequence, the parameter count is independent of the number of players, and the network is equivariant to teammate ordering.

Symmetry and Zero-Shot Coordination. Self-play can produce high-performing policies that rely on arbitrary conventions shared only by co-trained copies, which leads to poor coordination with independently trained agents or humans at test time (Hu et al., 2021a; 2020). Previous works addressed this through symmetry-based methods, partner reasoning, or training diversity (Cui et al., 2021; Muglich et al., 2022; Lupu et al., 2021; Jha et al., 2025). We discuss these approaches and their relationship to our model in Appendix F.

3 Background

The game of Hanabi. In Hanabi, 2 to 5 players work cooperatively to arrange cards in ascending order (from 1 to 5) for each color (typically five colors: red, blue, green, white, and yellow). Each player holds a hand of cards that it cannot see, but that all of its teammates can. On each turn, a player performs one of three actions. Either it gives a clue, plays a card, or discards a card. If the player *gives a clue*, it can provide one piece of information to another player about their hand (e.g., "These two cards are blue"). Giving a clue costs a *hint token*, and there are 8 hint tokens available. Alternatively, the player can decide to *play a card*, hoping it can be added to the sequence of cards on the table. If the card is correct (i.e., it extends one of the color sequences in ascending order), it is successfully played. If the card is wrong, it is discarded, and the team loses one of its 3 *life tokens*. Finally, the player can *discard a card* from their hand to gain a hint token back. This action is necessary when hint tokens run out, but it comes with the risk of discarding a crucial card. Each correctly arranged card results in 1 point, for a maximum of 25 points (all 5 cards of all 5 colors have been arranged correctly). The game ends either when all cards are arranged, all life tokens have been used, or the deck is empty.

Multi-Agent Reinforcement Learning. Hanabi is thus partially observable (players cannot see their own cards), decentralized (each player acts alone), and fully cooperative (points are shared by the team). In the MARL literature, such a setting is modeled as a Decentralized Partially-Observable Markov Decision Process (Dec-POMDP) (Bernstein et al., 2002; Nair et al., 2003), defined by the tuple $G = \{S, A, P, R, \Omega, O, N, \gamma\}$, with the set of states S, the set of actions A, the transition function P, the reward function R, the set of observations for each agent Ω, the observation function O, the number of agents N, and the discount factor γ. Agent i receives an observation $o^i \sim O(o \mid i, s)$ of the global state s, and all agents share the reward $r = R(s, \boldsymbol{a})$ of the joint action $\boldsymbol{a} = \left[a^i\right]_{i=1}^N$. At each timestep t, each agent selects a_t^i from a stochastic policy $\pi_\theta^i\left(a^i \mid \tau_t^i\right)$ conditioned on its action-observation history (AOH) $\tau_t^i = \left\{o_0^i, a_0^i, r_0^i, \ldots, o_t^i\right\}$, and the environment transitions to $s_{t+1} \sim P(\cdot \mid s_t, \boldsymbol{a}_t)$. Under the joint policy π, the Q-value $Q^\pi(s, a) = \mathbb{E}\left[\sum_t \gamma^t r_t \mid \pi, s_t = s, a_t = a\right]$ is the expected return, and we write $Q^* = \max_\pi Q^\pi$ for the optimal Q-value.

In single-agent RL, Deep Q-Networks (DQN) (Mnih et al., 2015) learn Q_θ, an approximation of Q^* with a neural network parametrized by θ. Q_θ is learned by minimizing $(Q_\theta(s,a) - (r + \gamma \max_{a'} Q_{\theta'}(s',a')))^2$, where $Q_{\theta'}$ is a copy of Q_θ updated regularly to stabilize learning. To extend DQN to the partially observable setting, Hausknecht & Stone (2015) propose Deep Recurrent Q-Network (DRQN), which uses recurrent neural networks to estimate Q-values based on the AOH, i.e., $Q^\pi(\tau, a) = \mathbb{E}\left[\sum_t \gamma^t r_t \mid \pi, \tau_t = \tau, a_t = a\right]$. This, combined with several modern best practices on top of DQN, including double-DQN (van Hasselt et al., 2016), a dueling network architecture (Wang et al., 2016), prioritized experience replay (Schaul et al., 2016), and a distributed training setup with parallel running environments (Horgan et al., 2018), results in Recurrent Replay Distributed Deep Q-Networks (R2D2) (Kapturowski et al., 2019), which we build on in Section 4. In MARL, a straightforward strategy is to represent each agent as an independent single-agent learner, e.g., using R2D2. An agent can then be trained through self-play (SP) with copies of itself (Tan, 1993; Tampuu et al., 2017). Our proposed algorithm, FHA, also uses SP to learn cooperative strategies.

UPDeT. UPDeT (Hu et al., 2021b) is a transformer-based agent architecture that handles a variable number of entities and a variable action space with a single set of parameters. The observation is split into M per-entity tokens; a shared transformer processes them jointly through self-attention and returns one output token per entity, each decoded into that entity's own action logits by a small *adapter* network (needed because, e.g., an ally supports fewer actions than an enemy in UPDeT's original StarCraft setting). To handle partial observability without a separate recurrent module, UPDeT appends one extra memory token to the M entity tokens and feeds its output back in as input at the next step: self-attention itself is reused as the recurrent unit (Figure 1a). Concretely, $\{h_t, y_t\} = \mathrm{TF}(h_{t-1}, x_t)$, where TF is the self-attention block and x_t, h_t, y_t are the entity tokens, memory token, and outputs at time t. Because self-attention is all-to-all, every output y_t^i depends on every input $x_{t'}^j$, for every entity j and every $t' \leq t$, i.e., on the entire trajectory so far: cross-entity mixing is repeated at every timestep rather than incurred once, and h_t must be computed strictly after h_{t-1}, ruling out the fused kernels available for simpler recurrent units.

4 Algorithm

FHA builds on R2D2, which learns a recurrent Q-network for each agent i,

$$\{h_t^i, Q_t^i\} = f_\theta(h_{t-1}^i, o_t^i),$$

where f_θ is typically an LSTM, o_t^i is agent i's observation at time t, and the hidden state h_t^i summarizes the AOH τ_t^i, so that Q_t^i approximates $Q(\tau_t^i, \cdot)$. In Hanabi, the sizes of both the input o_t^i and the output Q_t^i depend on the number of players N, so a network trained for one player count cannot be evaluated on another. FHA removes this dependency in two steps. First, it factorizes the observation into one token per teammate, all of the same size, and pairs each token with the actions that concern that teammate (Section 4.1). Second, it processes every token with the same parameters and assembles the resulting Q-values with a fixed, symmetric rule (Section 4.2). As a result, the number of parameters is independent of N, and the network is equivariant to the ordering of teammates.

4.1 Factorized observation and action spaces

Observation parts The Hanabi observation of agent i (Bard et al., 2020) contains public information about the game (fireworks, discard pile, remaining hint and life tokens), the hands of all teammates, which i can see but they cannot, and the actions taken by the other players.

We now define the parts from which both our implementation of UPDeT and FHA build their observations. We denote the observing player by i and the observed player by j. All quantities carry an additional time index t, which we omit for readability. The parts are: h_j, the hand of player j, as visible to player i; b, the board, i.e., the fireworks assembled so far; d, the discard pile; c_j, the clues that player j received this round; cc_j, the clued cards, i.e., cards of j whose rank or color has been revealed by the received clues; and id_j, the index of player j, mapped to the interval $[0, 1]$.

UPDeT observations The UPDeT applied to Hanabi (Figure 1a) combines these parts into tokens of different sizes, grouped by their semantic role: the global state $g = [b, d]$, the self-observation

$s = [c_i, cc_i]$, and the per-partner observation $p_j = [h_j, c_j, cc_j, id_j]$. To handle partial observability, UPDeT also maintains a special token h that serves as a memory cell. Each part is first mapped into a common hidden space by its own encoder; the resulting tokens are then processed by a transformer encoder and finally mapped by separate heads into the corresponding pieces of the Q-function.

Hanabi actions are either *self-directed* (play or discard one of one's own cards), which do not target a teammate, or *partner-directed* (give teammate j a color or rank clue). Token s is mapped to Q_{self}, which holds the values of the *play* and *discard* actions, and each per-partner token p_j is mapped to Q_j, which holds the values of the *clue* actions directed at that player. Concatenating these pieces yields the full Q-function, $Q_t^i = [Q_{\text{self}}, Q_1, \ldots, Q_{N-1}]$, which is then used for Q-learning.

4.2 Factorized Hanabi Agent

The Factorized Hanabi Agent (Figure 1b) is built to work on uniform tokens, each of which contains all the information pertaining to the observer, the observed teammate, and the global state:

$$o^{i,j} = [b, d, h_j, c_j, cc_j, id_j, c_i, cc_i]$$

These tokens are passed through the equivariant FHA model (described below), which outputs one piece of the Q-function per teammate, $\hat{Q}_j = [Q^j_{\text{self}}, Q_j]$. The self-directed values are then mean-pooled across teammates,

$$Q_{\text{self}} = \frac{1}{N-1} \sum_{k=1}^{N-1} Q^k_{\text{self}},$$

The partial Q-functions are combined with the fixed rule above.

FHA-H: per-player LSTM, heuristic aggregation. The simplest variant applies the same recurrence independently to every token,

$$\{h_t^j, \hat{Q}_t^j\} = f_\theta(h_{t-1}^j, o_t^{i,j}), \qquad j \neq i,$$

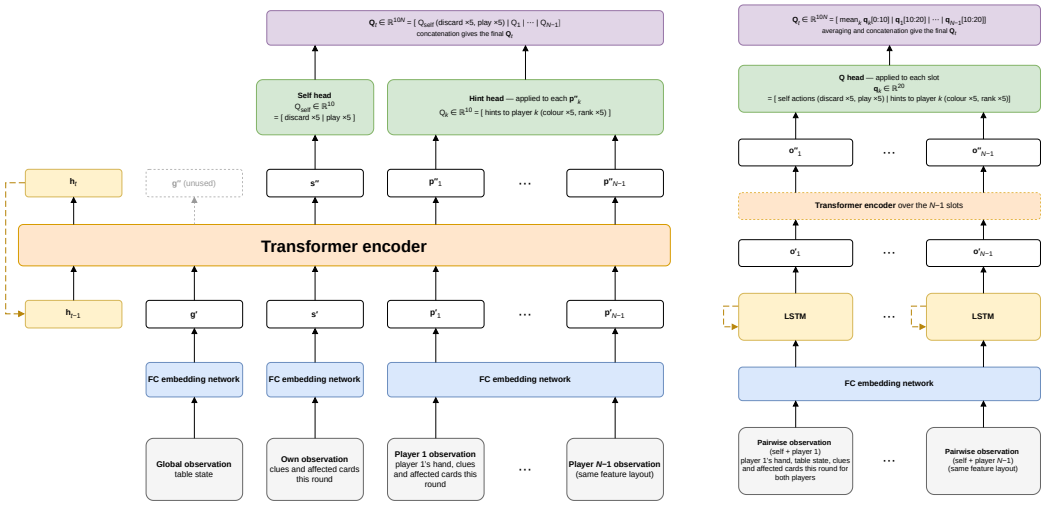
with weights shared across teammates and no communication between the recurrences. Agent i thus maintains one hidden state per teammate. The outputs are combined with the fixed rule above. Since the weights are shared, FHA-H is permutation-equivariant by construction. Teammates interact only through the averaging of self-directed values, which acts as a strong inductive bias: each recurrence must estimate the value of playing or discarding from the global information and the hand of a single partner alone.

FHA-T: per-player LSTM, cross-player attention. FHA-T uses the same per-player LSTM, but before the Q-values are produced, the embeddings of all teammates are mixed by a Transformer encoder. Without positional encodings, self-attention is permutation-equivariant, and hence so is FHA-T. The learned mixing lets the agent condition the value of an action concerning one teammate on the state of the others, e.g., to decide which teammate is most in need of a clue.

Comparison with UPDeT. Both variants separate the two roles that UPDeT fuses into a single self-attention block. Temporal recurrence is handled by one LSTM per teammate, and cross-player mixing happens once per timestep, outside the recurrent loop. This has two practical benefits. First, the recurrence can use fused LSTM kernels, and the per-player recurrences can be batched along the player dimension, which makes FHA an order of magnitude faster than UPDeT in wall-clock time (Table 1). Second, temporal credit assignment propagates through LSTM gates rather than through repeated self-attention, and FHA-T achieves a higher score than UPDeT (Section 5.1).

5 Experiments

We study four questions. First, can a single factorized agent, trained jointly on all player counts, match specialists trained separately for each (Section 5.1)? Second, how does the set of player counts seen during training affect cross-play and zero-shot transfer to unseen player counts (Section 5.2)? Third, how does compatibility between independently trained agents evolve during training, and does it carry over across player counts (Section 5.3)? Fourth, can FHA be combined with dedicated zero-shot coordination methods (Section 5.4)?



(a) UPDeT (b) FHA

Figure 1: Architectures compared in this work. (a) UPDeT: a single transformer encoder attends jointly over the global, own, and per-player observation slots, with separate self and hint heads producing Q_t. (b) FHA: each pairwise observation is encoded by a per-player LSTM. Both FHA variants merge the per-player outputs into Q_t with the same fixed rule; the Transformer variant additionally routes the LSTM outputs through a transformer encoder (dashed box) for learned cross-player communication before this merge, which the Heuristic variant omits.

Tasks. We use Hanabi and treat each player count $N \in \{2, 3, 4, 5\}$ as a separate task. Factorized agents can be trained on any subset of tasks: we consider all tasks jointly (*multitask*), pairs $\{2, 3\}$ and $\{4, 5\}$, and each task in isolation (*specialists*). Every factorized agent is evaluated on all tasks, which measures in-distribution performance on its training tasks and zero-shot transfer on the others.

Methods. We evaluate the two variants of FHA, FHA-H and FHA-T (Section 4.2), and UPDeT (Hu et al., 2021b) as a factorized baseline. As a non-factorized reference, we train an IQL-LSTM agent separately on each task, which, since its input size depends on N, can only be evaluated on the task it was trained on (Appendix C). Unless stated otherwise, all agents are trained in self-play with independent Q-learning (IQL), share the same R2D2 setup (Kapturowski et al., 2019), and differ only in their observation representation and Q-network. Hyperparameters, network sizes and training details are given in Appendix G.

Metrics. We report the *self-play* score, obtained when an agent plays with copies of itself, and the *cross-play* score, obtained when it plays with agents of the same configuration trained from different random seeds. Cross-play is our proxy for coordination with unfamiliar partners: an agent relying on arbitrary, seed-specific conventions reaches a high self-play score but a low cross-play score. We train 15 seeds per configuration, and 10 for UPDeT due to its computational cost (Table 1). For cross-play, at each evaluation step we sample 15 teams of distinct seeds, average each team's score over 500 games, and report the mean and standard deviation across teams.

5.1 Comparison of multitask methods with specialist baselines

Figure 2 compares multitask FHA-H, FHA-T and UPDeT, each trained jointly on all player counts, with IQL-LSTM specialists trained separately for each.

Self-play. Given enough training, IQL-LSTM specialists reach the highest self-play scores, but they converge more slowly for $N > 3$, where factorized agents reach good scores much earlier despite playing all four tasks with a single set of parameters. Both variants of FHA lose little performance compared to the specialists, and FHA-T obtains higher self-play than FHA-H, which we attribute to the additional expressivity of its learned cross-player attention. UPDeT reaches

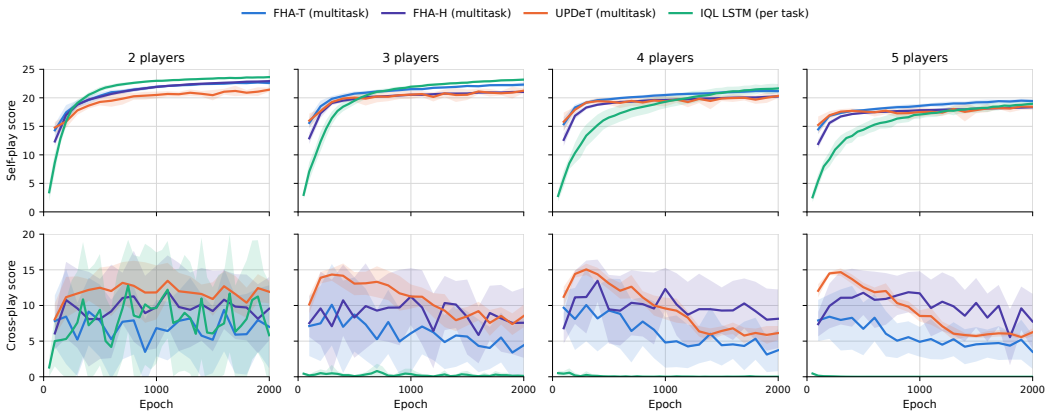


Figure 2: Self-play and cross-play scores during training, for each player count. IQL-LSTM is trained separately for every player count, while the factorized agents are trained on all player counts at once. UPDeT curves are averaged over 10 seeds, all other curves over 15.

comparable to FHA-H self-play scores and worse than FHA-T, but at a much higher computational cost: an epoch of multitask training takes 507 seconds for UPDeT, against 45 seconds for FHA-T and 29 seconds for FHA-H, i.e., UPDeT is 11 to 18 times slower (Table 1).

Cross-play. Cross-play shows a very different picture. IQL-LSTM specialists have essentially no cross-play for $N > 2$: independently trained seeds settle on mutually incompatible conventions. In contrast, all factorized agents retain a substantial degree of cross-play at every player count. Since this holds for three different network structures, it points to a property of factorized training rather than of a particular architecture. Section 5.2 disentangles the respective roles of the architecture and of the training tasks. UPDeT obtains the highest cross-play values, but its cross-play degrades as training progresses while its self-play keeps increasing, a sign that its seeds gradually overfit to their own conventions. Given its computational cost, we do not use UPDeT in the more extensive studies below. Finally, FHA-H obtains better cross-play than FHA-T, at the price of slightly lower self-play. We interpret this as a regularization effect: without learned mixing between teammates, the heuristic aggregation restricts the conventions the agent can express, and independent seeds converge to simpler solutions that are closer to each other. We therefore use FHA-H as our main agent in the remainder of this section, and report the corresponding results for FHA-T in the appendix.

5.2 Cross-play impact of training on other player counts

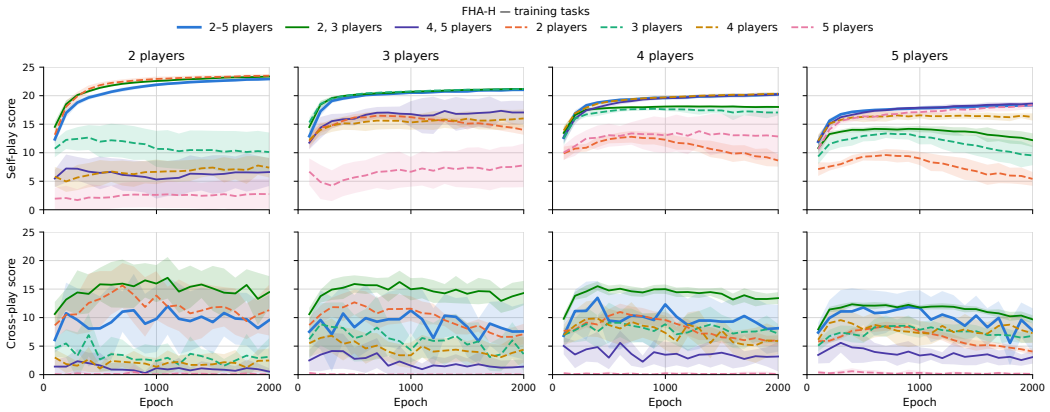
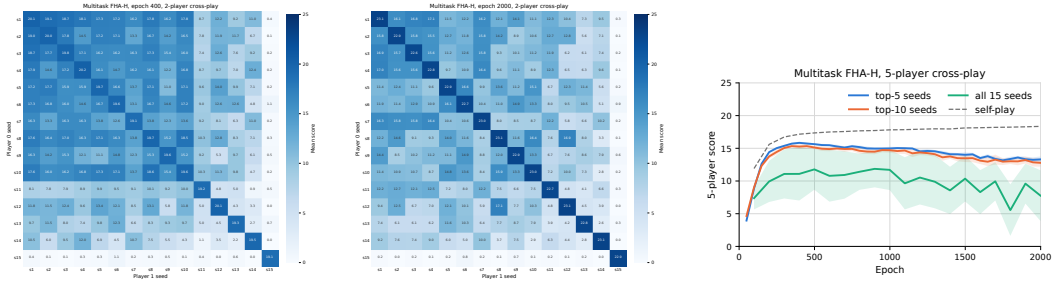


Figure 3: Self-play (top row) and cross-play (bottom row) of FHA-H trained on different subsets of player counts, evaluated on every player count.



(a) 2-players, epoch 400 (b) 2-players, epoch 2000 (c) Top M seeds for 5-players

Figure 4: Cross-play among the 15 multitask seeds. **(a, b)** 2-player cross-play matrices at epochs 400 and 2000, 1000 games per cell. Rows and columns are ordered by mean cross-play at epoch 400; the diagonal is self-play. **(c)** Evolution of the cross-play with $N = 5$ players among the top M most compatible seeds, compared to self-play and the cross-play of the whole population of agents.

To separate the effect of the training tasks from that of the architecture, we train FHA-H on different subsets of player counts and evaluate every agent on all four tasks (Figure 3).

Self-play and transfer. On every task it was trained on, the multitask agent matches the self-play of the corresponding specialist. Thus, generalizing across player counts comes at no cost in performance. Agents also transfer zero-shot to player counts they were not trained on, although this transfer fades as the difference between the training and evaluation player counts grows.

Cross-play. The multitask agent maintains robust cross-play across all player counts, whereas specialists exhibit a much larger gap between self-play and cross-play. A notable exception is the agent trained on $\{2, 3\}$, which reaches even higher cross-play, including on the 4- and 5-player tasks it was never trained on. This is not specific to FHA-H, as the same holds for FHA-T (Figure 6 in the appendix). Similarly, the agent trained on 2 players only has good cross-play on the smaller player counts. We hypothesize that, with few partners, the agent learns a robust strategy of clues and reactions between pairs of players, and falls back on it when playing with more partners, effectively decomposing the game into a set of pairwise interactions, while the more sophisticated strategies learned with more partners do not transfer as well. Overall, these results show that the diversity of training tasks is by itself a source of cross-play robustness: an agent that must play every player count with the same parameters cannot tailor its conventions to any single one of them.

5.3 Evolution of agents' compatibility during training across different tasks

We now examine how compatibility between seeds evolves during training. Figure 4 shows the 2-player cross-play matrix of the 15 multitask FHA-H seeds at epochs 400 and 2000. The matrix does not exhibit separate clusters of mutually compatible seeds, as would be expected if seeds converged to a few distinct sets of conventions. Instead, compatibility is ordered along a single axis: seeds that cross-play well do so with all other good seeds, and weaker seeds are uniformly weaker. This suggests that seeds converge, with varying degrees of success, towards a single shared strategy rather than towards competing ones. At epoch 2000, the self-play diagonal is more pronounced, i.e., every seed has specialized further to itself, but the ordering is preserved: seeds that were compatible at epoch 400 remain the most compatible.

Compatibility also carries over across player counts. We select the $M \in \{5, 10\}$ seeds with the highest mean 2-player cross-play and evaluate their cross-play in the 5-player game (Figure 4c). Seeds that are compatible at $N = 2$ remain compatible at $N = 5$. We see the cross-play of the top seeds stays close to their self-play, and most of the variance of the full population comes from its bottom third. Hence, 2-player cross-play, which is much cheaper to evaluate, is a good predictor of cross-play in larger games. Nonetheless, a gap between self-play and cross-play remains, even among the best seeds (see also Appendix D for FHA-T), which motivates combining FHA with methods designed specifically for zero-shot coordination.

5.4 COMBINING FHA WITH ZERO-SHOT-COORDINATION METHODS

FHA is complementary to zero-shot coordination (ZSC) methods, and Section 5.3 shows that a residual gap remains for them to close. We combine FHA with two ZSC methods developed specifically for Hanabi: Other-Play (OP) (Hu et al., 2020) and Off-Belief Learning (OBL) (Hu et al., 2021a).

Other-Play. In our setup, OP did not improve cross-play, neither with FHA nor with the IQL-LSTM baseline (Appendix E). This does not contradict the results of Hu et al. (2020), who combine OP with additional techniques that we do not use, namely the Simplified Action Decoder (Hu & Foerster, 2020) and an auxiliary loss.

Off-Belief Learning. OBL trains a hierarchy of policies, in which the policy at level k acts optimally assuming that past actions were taken by the policy at level $k-1$, level 1 being uniformly random, and uses a learned belief model over the agent's own hand to sample fictitious transitions (Hu et al., 2021a). Since OBL only changes the learning targets, FHA can directly replace the underlying Q-network. We first validate our implementation with IQL-LSTM. As expected, it reaches much higher cross-play than IQL alone (Figure 5), and cross-play increases between level $k = 1$ and $k = 2$. For FHA-T, we see a consistent increase of cross-play performance for $k = 1$ over IQL FHA-T. Moreover, cross-play is higher than the LSTM OBL-1 baseline for 5-players. However, this does not propagate to $k = 2$, where cross-play performance for FHA-T drops. Strictly speaking, FHA does not satisfy one of the assumptions of OBL. A fictitious transition resamples the acting agent's hand from the belief model, which changes what its teammates observe, while their recurrent states were computed from the true hand. OBL avoids this inconsistency with a public-LSTM, whose recurrence only processes public information, private information being injected after the recurrence. In FHA, the per-player LSTMs process teammates' hands, and thus private information. Thus, we see that, even though technically we can perfectly combine FHA with zero-shot coordination methods, in practice they can interfere with one another. Further combining factorization with zero-shot is thus an interesting research direction, which we leave for future work.

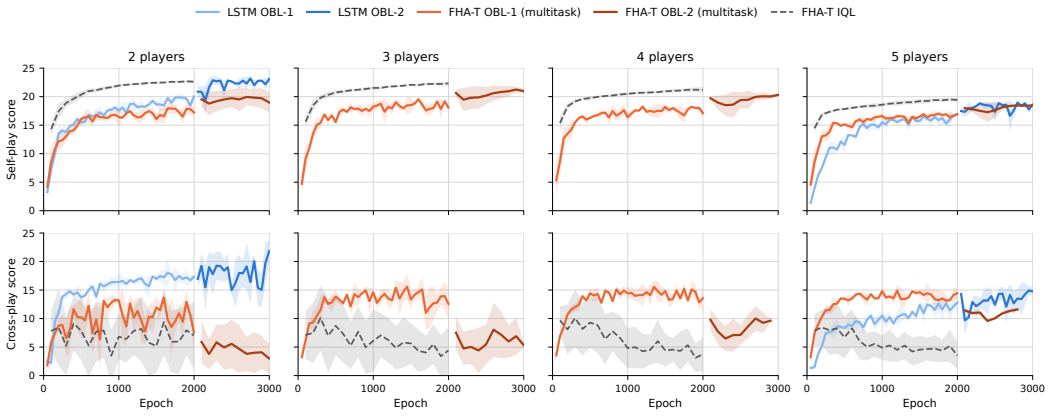


Figure 5: Self-play and cross-play of FHA-T and IQL-LSTM trained with OBL, for two successive levels of the OBL hierarchy. As a reference, we also provide FHA-T trained with self-play.

6 CONCLUSION

We propose FHA, a simple and fast factorized architecture that plays Hanabi with 2 to 5 players using a single set of parameters. Trained jointly on all player counts, FHA matches the self-play of single-setting specialists, transfers zero-shot to unseen player counts, and reaches substantially higher cross-play, especially with heuristic aggregation, which acts as a regularizer. With this aggregation, independently trained seeds converge towards a single shared strategy, and their 2-player cross-play predicts cross-play in games of up to 5 players. Future work includes reducing the variance across seeds, designing regularizers less drastic than removing cross-player attention, and a "public" variant of FHA compatible with Off-Belief Learning.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods, creating or editing software code, editing the research paper to improve readability, creating or modifying scientific figures or images.

We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, suggesting experimental parameters, formatting references. Generating synthetic data sets, formulating mathematical claims, assisting in the writing of proofs, assisting with translation, cleaning and reformatting datasets, are not applicable to this work.

We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. Reformulated sections were proofread and verified. Rendered figures were compared with the Weights and Biases[1] dashboard used to log experiments. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

[1] https://wandb.ai/site

REFERENCES

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. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, March 2020. ISSN 0004-3702. doi: 10.1016/j.artint.2019.103216. URL http://dx.doi.org/10.1016/j.artint.2019.103216.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4): 819–840, 2002.

Brandon Cui, Hengyuan Hu, Luis Pineda, and Jakob Foerster. K-level reasoning for zero-shot coordination in hanabi. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Jianye Hao, Xiaotian Hao, Hangyu Mao, Weixun Wang, Yaodong Yang, Dong Li, Yan Zheng, and Zhen Wang. Boosting multiagent reinforcement learning via permutation invariant and permutation equivariant networks. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=OxNQXyZK-K8.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2018.

Hengyuan Hu and Jakob N Foerster. Simplified action decoder for deep multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1xm3RVtwB.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "Other-play" for zero-shot coordination. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 4399–4410. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/hu20a.html.

Hengyuan Hu, Adam Lerer, Brandon Cui, Luis Pineda, Noam Brown, and Jakob Foerster. Off-belief learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 4369–4379. PMLR, 18–24 Jul 2021a. URL https://proceedings.mlr.press/v139/hu21c.html.

Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. Updet: Universal multi-agent reinforcement learning via policy decoupling with transformers, 2021b. URL https://arxiv.org/abs/2101.08001.

Shariq Iqbal, Christian A. Schroeder de Witt, Bei Peng, Wendelin Bohmer, Shimon Whiteson, and Fei Sha. Randomized entity-wise factorization for multi-agent reinforcement learning. In *Proceedings of the 38 th International Conference on Machine Learning*, 2021.

Kunal Jha, Wilka Carvalho, Yancheng Liang, Simon Shaolei Du, Max Kleiman-Weiner, and Natasha Jaques. Cross-environment cooperation enables zero-shot multi-agent coordination. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 27198–27220. PMLR, 2025.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7204–7213. PMLR, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Darius Muglich, Christian Schroeder de Witt, Elise van der Pol, Shimon Whiteson, and Jakob Foerster. Equivariant networks for zero-shot coordination. In *Advances in Neural Information Processing Systems*, volume 35, pp. 6410–6423. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/29e4b51d45dc8f534260adc45b587363-Paper-Conference.pdf.

Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 705–711, 2003.

Hadi Nekoei, Akilesh Badrinaaraayanan, Aaron Courville, and Sarath Chandar. Continuous coordination as a realistic scenario for lifelong learning. In *International Conference on Machine Learning*, pp. 8016–8024. PMLR, 2021.

Hyunwoo Park, Baekryun Seong, and Sang-Ki Ko. Spectra: Scalable multi-agent reinforcement learning with permutation-free networks, 2025. URL https://arxiv.org/abs/2503.11726.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE*, 12(4):e0172395, 2017.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning (ICML)*, pp. 330–337, 1993.

Kale-ab Abebe Tessera, Arrasy Rahman, Amos Storkey, and Stefano V Albrecht. Hypermarl: Adaptive hypernetworks for multi-agent rl. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=56CqYnf9Dr.

Johannes Treutlein, Michael Dennis, Caspar Oesterheld, and Jakob Foerster. A new formalism, method and open issues for zero-shot coordination, 2023. URL https://arxiv.org/abs/2106.06613.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, pp. 2094–2100, 2016.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48, pp. 1995–2003, 2016.

R Zhao, J. Song, Y. Yuan, H. Hu, Y. Gao, Y. Wu, Z. Sun, and W. Yang. Maximum entropy population-based training for zero-shot human-ai coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6145–6153, 2023.