# The Yōkai Learning Environment: Tracking Beliefs Over Space and Time

**Constantin Ruhdorfer**[1] , **Matteo Bortoletto**[1] , **Andreas Bulling**[1]
[1]Collaborative Artificial Intelligence, University of Stuttgart, Germany
constantin.ruhdorfer@vis.uni-stuttgart.de

## Abstract

Developing collaborative AI hinges on Theory of Mind (ToM) – the ability to reason about the beliefs of others to build and maintain common ground. Existing ToM benchmarks, however, are restricted to passive observer settings or lack an assessment of how agents establish and maintain common ground over time. To address these gaps, we introduce the Yōkai Learning Environment (YLE) – a multi-agent reinforcement learning (RL) environment based on the cooperative card game Yōkai. In the YLE, agents take turns peeking at hidden cards and moving them to form clusters based on colour. Success requires tracking evolving beliefs, remembering past observations, using hints as grounded communication, and maintaining common ground with teammates. Our evaluation yields two key findings: First, current RL agents struggle to solve the YLE, even when given access to perfect memory. Second, while belief modelling improves performance, agents are still unable to effectively generalise to unseen partners or form accurate beliefs over longer games, exposing a reliance on brittle conventions rather than robust belief tracking. We use the YLE to investigate research questions in belief modelling, memory, partner generalisation, and scaling to higher-order ToM.

## 1 Introduction

Effective collaboration among partners requires *common ground* – the shared knowledge, beliefs, and assumptions that enable them to interpret each other's actions and respond appropriately [Clark, 1996]. When common ground cannot be reliably maintained, collaboration often breaks down [Klein *et al.*, 2005]. To establish common ground, humans use Theory of Mind (ToM) [Premack and Woodruff, 1978] to infer others' mental states. Consequently, enabling computational agents to perform ToM reasoning has been identified as a critical step toward effective human-AI collaboration [Klien *et al.*, 2004; Dafoe *et al.*, 2020; Dafoe *et al.*, 2021].

Research in cognitive science suggests that human ToM integrates multiple reasoning processes – including memory, spatial navigation, and mentalising [Suddendorf and Corballis, 2007; Mitchell, 2009]. Prior work in computational ToM has focused on the latter, for example, by predicting others' behaviour from an omniscient observer's perspective [Baker *et al.*, 2011; Rabinowitz *et al.*, 2018; Gandhi *et al.*, 2021]. However, humans are rarely passive observers of a scene where they solely predict behaviour [Sclar *et al.*, 2022]. Instead, they are active participants and must adjust their ToM reasoning to the interactive setting in which their own actions change the environment, influence the behaviour of others, and vice versa. This interactive structure necessitates tracking how common ground – the mutually recognised shared experience built through interaction [Tomasello, 2010] – evolves during collaboration.

The goal of this paper is thus to

> *propose an environment in which effective collaboration between computational agents requires building common ground by tracking beliefs over space and time.*

This is the *Yōkai Learning Environment* (YLE)[1] – a novel multi-agent reinforcement learning (RL) environment based on the collaborative card game Yōkai. In the YLE, players must collaborate to group face-down cards by colour over multiple rounds without direct communication (see Figure 1, left). Each round, one player privately observes the colour of two cards, moves one of them to a new position on the table, and may place a hint card to communicate information about unobserved card colours to others (see Figure 2). Importantly, players may end the game early for a higher reward, but at the risk of losing the game. This makes it a high-stakes, all-or-nothing decision that requires strong ToM reasoning to infer card colours from partners' actions rather than first-hand observation.

The design of YLE creates a uniquely challenging coordination task. Agents must: (1) remember the colours of moving cards to avoid redundant checks, (2) infer the beliefs and knowledge of others to establish and maintain common ground, (3) interpret teammates' actions and hints as implicit belief updates, (4) use hint cards as grounded communication, and (5) assess when to end early based on inferred shared knowledge. These challenges require belief tracking over both space (shifting card locations) and time (multi-round

[1]Code available at https://git.hcics.simtech.uni-stuttgart.de/public-projects/Yokai_env.

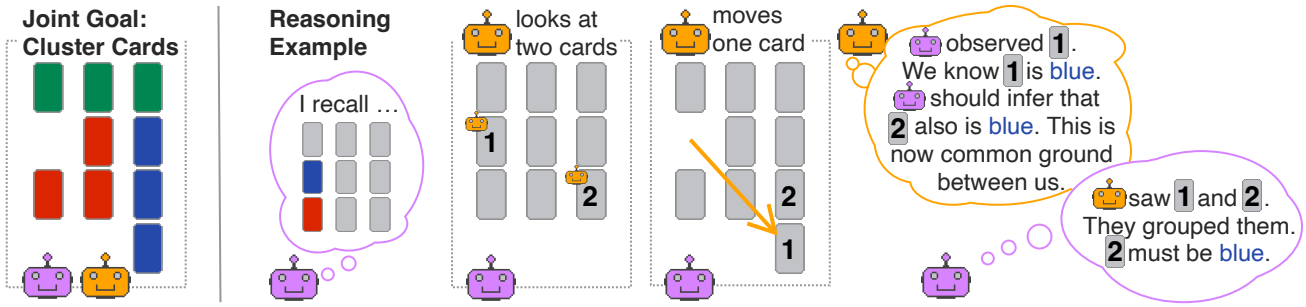


Figure 1: The YLE poses a challenging ToM reasoning task for cooperative agents in a dynamic environment. Agents cannot observe all cards in a single game. Successful play requires agents to remember and reason about others and their own mental states. In the example, it's 🤖 's turn. 🤖 observes the colour of two cards (1 and 2) privately. 🤖 recalls from earlier that card 1 is blue. When 🤖 moves card 1 next to card 2, 🤖 can infer that the second card is likely blue as well. 🤖 and 🤖 can now assume this is common ground going forward.

gameplay with limited observability), and together reflect key components of ToM [Premack and Woodruff, 1978; Clark, 1996; Pillow, 1989; Southgate *et al.*, 2007; Tomasello, 2010] in a dynamic, interactive setting. Effectively establishing common ground requires aligning on ad hoc conventions for grouping cards with partners. Prior environments, such as Hanabi [Bard *et al.*, 2020], were used to study zero-shot coordination and ToM, but lack the spatial-temporal structure, evolving belief dynamics, and the challenge of ending early based on inferred knowledge.

We evaluate a range of agents on YLE and highlight key limitations in their ability to track common ground in this novel spatial-temporal belief reasoning task. Our experiments show that even with access to perfect memory, agents struggle to collaborate effectively. Among agents without perfect memory, those with explicit memory modules perform best, and performance drops further in four-player settings. While belief representation learning improves performance, agents fail to generalise to new partners or maintain accurate beliefs over longer time horizons. Instead, they rely on brittle conventions rather than robust and generalisable belief tracking. Our contributions are:

1. A fast JAX-based YLE – a new cooperative multi-agent RL benchmark for belief tracking, common ground, and ToM, training at hundreds of thousands of steps per second.
2. An empirical analysis showing that agents fail to coordinate effectively in YLE, and in particular fail to end games early – a behaviour humans achieve reliably.
3. A demonstration that belief representation learning improves performance, but agents still fail to generalise to novel partners or maintain accurate beliefs over time.
4. An evaluation of YLE in a four-player setting, revealing the increased difficulty of maintaining common ground when high-order ToM is required.

## 2 Yōkai

Yōkai[2] is a turn-based cooperative game in which players must sort face-down cards into same-colour clusters to win. A key requirement is that all cards must be connected to at least one other card on their sides. Classically, the game is played by two to four players with 16 cards of four colours each. Complementing these 16 Yōkai cards, additional *hint cards* feature one or multiple colours and are used to hint at the colour of a face-down card underneath them. They are the only allowed communication mechanism in the game. The number of players determines the number of hint cards, which varies between seven and 10 (see Appendix B for further details). Hint cards are initially face-down and must be revealed before being placed face-up on a Yōkai card. Importantly, this Yōkai card is locked in place and can no longer be moved or observed. Since most hints can be multi-coloured, their interpretation is left to the other agents, depending on the context and history.

A typical turn in Yōkai consists of four steps (see Figure 2). The current player first observes two cards *privately*, then moves one card, and then either (a) reveals or (b) places a face-up hint card. Consequently, the available legal moves depend on the specific card to be moved. Figure 11, while Figure 10 show legal moves and card configurations. The game ends when all hint cards have been revealed and placed. Optionally, players can end the game prematurely instead of observing cards at the beginning of their turn. The final game score is based on the final card configuration:

$$S = 5N_{\text{hints\_face\_down}} + 2N_{\text{hints\_not\_placed}} + N_{\text{hints\_correct}} - N_{\text{hints\_wrong}}. \quad (1)$$

Players are encouraged to end the game early to achieve a higher score, since unused hint cards contribute most to the reward. However, the earlier a player ends a game, the less information they can acquire first-hand and thus must infer knowledge from others' behaviour (see Figure 12 for an example). As such, *the final game score measures players' joint ToM reasoning abilities*. In particular, *the rate at which players successfully end the game early reflects how much*

[2]Detailed rules for Yōkai are available at http://boardgame.bg/yokai%20rules.pdf.

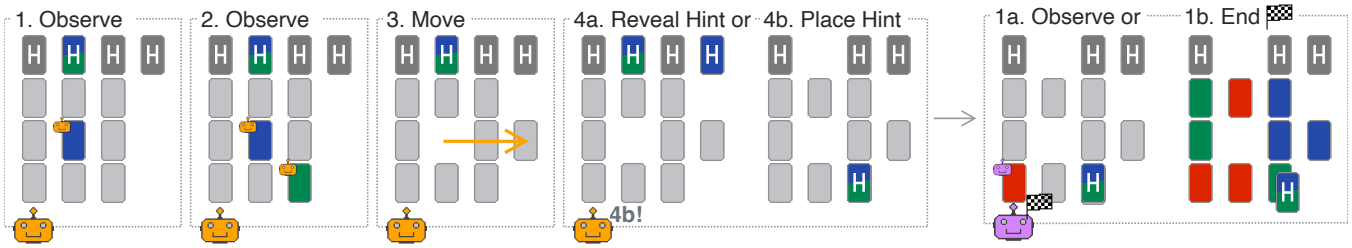


Figure 2: A sample round in nine-card YLE. Agent first observes two cards (one blue, one green), moves the blue card, and finishes by either (4a) revealing or (4b) placing a hint. places a hint. Then, can start its turn. chooses to end the game. Ending the game will return the final reward based on the outcome. This game is lost because while blue cards are clustered, green and red ones are not.

*they rely on belief inference*, making it a powerful new implicit measure of ToM reasoning under uncertainty.

Yōkai poses three key challenges to cooperative agents: First, achieving a high score requires agents to end a game as early as possible, resulting in a higher risk of failure. Second, ending the game early requires interpreting others' actions and tracking their beliefs to form correct own beliefs over the colours of unobserved cards (see Figure 1). Third, agents must use hints efficiently as grounded communication, but a single misplaced hint can render the game unwinnable.

## 3 Preliminaries

We model Yōkai as a *decentralised partially observable Markov decision process* (Dec-POMDP) [Oliehoek and Amato, 2016], formally described as a tuple $\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \Omega, O, A, \mathcal{R}, \mathcal{T}, \gamma \rangle$. $\mathcal{N}$ is the set of agents where $i \in \mathcal{N}$, $1 \leq i \leq |\mathcal{N}|$ denotes one specific agent and $-i$ denotes all other agents. Set $\mathcal{S}$ denotes the possible true states of the environment, with $s \in \mathcal{S}$ being a single state. At timestep $t$ an agent in state $s_t$ receives a partial observation $o_t^i \in \Omega$ through the observation function $O(s_t, i)$. The action-observation history per agent is denoted by $\tau^i$. Following Sclar et al. [2022], an agent with no ToM chooses an action $a^i \in A$ only based on their observation of the environment $\pi^i(a^i|o_t^i)$. Agents with zeroth-order ToM incorporate memory, conditioning their policy on a hidden state $h_t^i$ in addition to their current observation $\pi^i(a^i|o_t^i, h_t^i)$. Furthermore, agents that reason about the mental states of others estimate their hidden states $\hat{h}_t^{(j)}$ for $j \neq i$ and condition their action on all $\hat{h}_t^{(-i)}$: $\pi^i(a^i|o_t^i, h_t^i, \hat{h}_t^{-i})$. Based on the joint action performed by all agents $\boldsymbol{a} = (a_1, \ldots, a_n)$ the environment returns a shared reward $r_t = R(s_t, \boldsymbol{a})$ and then produces a new state based on the state transition function $\mathcal{T}$. Finally, using the discount factor $\gamma^t$, the agents optimise the discounted return of a trajectory given by $R(\tau) = \sum_t \gamma^t r_t$.

## 4 The Yōkai Learning Environment

The Yōkai Learning Environment implements the game logic with two exceptions: First, we support configurable game sizes (e.g., a **3x3** version with nine cards and three colours). We refer to it as **3x3** while the original game is referred to as **4x4**. The **3x3** two-player game has four hint cards: one one-colour hint and three two-colour hints. Second, our game is restricted to a $g \times g$ grid to efficiently compute legal moves. We pick $g$ large enough not to limit gameplay in practice and confirm this experimentally, see Appendix B. Our implementation is based on JaxMARL [Rutherford *et al.*, 2023] and is fully GPU-accelerated. It allows end-to-end learning on a single GPU at hundreds of thousands of steps-per-second, see Table 3.

**Yōkai as a Graph** We opted to model Yōkai as a graph since it is a natural formulation of the relational nature of the environment and allows for an efficient implementation in JAX. At time $t$, the state is a graph $G_t(V, A)$ with cards and hints as vertices $V = Y \cup H$ and adjacency matrix $A$. Within $G_t$, cards are vertices $V = Y \cup H$ and connected to neighbours via edges represented by an adjacency matrix $A$. Cards and hints are identified by their index $0 \leq i < |Y|$ or $0 \leq j < |H|$. Hint cards are not connected to any card. For example, in the 3x3 version with nine cards and four hints, colours (e.g., blue, green, red) and hints are randomly assigned each episode, and the graph $A \in \{0,1\}^{13\times13}$ is updated when cards move. With this setup, moving a card $i$ boils down to obtaining $A'$ from $A$ via $A' = \text{move}(A, i, n)$ where in move the row and column vector at index $i$ of $A$ are replaced with a binary encoding of the new neighbours $n$. Additionally, a configuration is legal if all cards are connected, computed via a reachability matrix over $A$. Determining whether the game has been won then boils down to computing

$$\bigwedge_{c \in C} \texttt{isConnected}(A_c) \tag{2}$$

where $A_c$ denotes the submatrix of $A$ obtained by deleting all column and row vectors not associated with colour $c$, i.e. by finding the set of indices $\{j | 0 \leq j < |Y| \wedge Y[j] \neq c\}$ where $Y[j]$ denotes accessing the tuple $Y$ at index $j$. To compute the state-dependent set of legal move actions $A(s) \subseteq A$, we run $\texttt{isConnected}(A')$ on all $A'$ obtainable by moving a card to a position in the $g \times g$ field.

**Observation Space** Agents receive graph-based or image-like observations encoding card properties (e.g., colour, position, lock state) and hint features. Observations include both *public* (topology and hints) and *private* (card features) information, leading to asymmetric knowledge. Graph observations encode cards as an adjacency matrix $A_t \in \{0,1\}^{|Y|\times|Y|}$, and their properties as node features $F_t \in \mathbb{R}^{|Y|+|H|\times f}$, where $f$ is the feature dimension. In image-like observations, card features $F$ are embedded into

a $g \times (g+1) \times f$ tensor based on their positions, while hint card features are embedded into column $g+1$. Agents observe which cards others inspect, but not their content. Partial observability forces agents to seek information and track shared knowledge.

**Action Space** Actions in the YLE are represented by a categorical action space. Additional players increase the number of hints and, thereby, the number of possible actions. The action space includes observing pairs of cards, moving cards, placing or revealing hints, ending the game, and a no-op. The action space size is the sum of the number of cards to look at $|Y|$, the number of target positions any card can be moved to $N_{moves} = g^2|Y|$, and the number of hints $|H|$. The number of cards a hint can be placed onto $N_{hint\_moves} = |H| \cdot |Y|$, 1 action for no operation and 1 action to end the game. The exact size of the action space depends on the setup but ranges from $\sim 1,000$ to $\sim 3,000$ in practice, all details are in Appendix B.

**Reward** The score in Equation 1 is only given when the game ends successfully, and sometimes can be negative due to hint misplacements. Agents, however, may learn that it is better to avoid grouping cards altogether to avoid penalties. To address this, we modify the reward function. If the game ends without a win, we give a small negative reward to discourage this strategy:

$$R = \begin{cases} S & \text{if game won} \\ -d_a - (|C| - c_a) - i_h & \text{otherwise.} \end{cases} \quad (3)$$

where $c_a$ is the number of correctly formed clusters, $i_h$ is the number of incorrectly placed hints, and $d_a \in \{0, 1\}$ indicates whether the game was ended early. Since rewards are sparse, we shaped only the training reward to accelerate learning. We additionally reward observing a previously unobserved card $(+1)$, grouping a single colour, and placing a correct hint (both $+1$) or an incorrect hint $(-1)$.

## 5 Baselines, Experiments, and Results

To evaluate the spatio-temporal belief tracking capabilities of agents in the YLE, we explore five complementary research questions (RQs). We begin by comparing agents with different neural architectures to establish baselines, then progressively test their capacity for belief modelling, generalisation, and reasoning in increasingly complex scenarios:

**RQ1** How well do different policies reason over their memory (zero-order ToM)?

**RQ2** Does learning belief representations and explicit action encoding improve agents' performance?

**RQ3** Do agents learn belief-tracking strategies that let them coordinate with new partners?

**RQ4** Do agents represent beliefs in their internal activations?

**RQ5** Can agents scale to higher-order ToM scenarios involving four players?

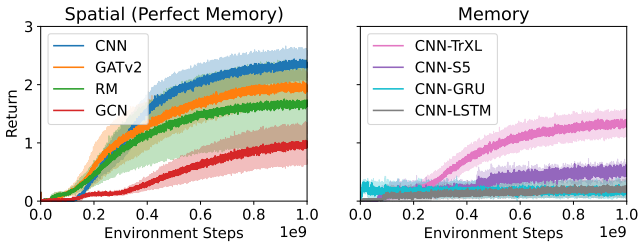


Figure 3: Training curves for the **3x3** YLE, with bands showing the standard error over three seeds. **Left**: comparison of encoders with perfect memory. **Right**: comparison of memory architectures.

**Setup** We use the two-player **3x3** YLE version, which already presents a significant challenge for RL agents. Agents are trained using MAPPO [Yu *et al.*, 2022] for $10^9$ steps and evaluated on 1,000 games, across three random seeds. The maximum episode length was 32 for this setting. We report four metrics: return (**R**), success rate (**SR**), number of colours clustered (**NC**) and the rate of successfully ending early (**SEE**). **SEE** captures how often agents end the game early and still win – a decision that requires strong belief inference under uncertainty that comes with significant risk of failure. We tuned key hyperparameters for our models (e.g., encoder size, depth, etc.); Training details are in Appendix C.

### 5.1 RQ1a: Perfect memory is not sufficient to solve the YLE

To evaluate if agents can reason about their own memory (zero-order ToM), we first compared different encoder architectures in a **perfect memory setting**: In this setting, agents were given perfect memory by including all previously observed cards in their observations. This experiment also served to set a performance upper bound. Given the YLEs spatial reasoning task and graph-based nature, we compared CNN-, GCN- [Kipf and Welling, 2016], GATv2- [Brody *et al.*, 2021], and Relation Module (RM)-based encoders [Zambaldi *et al.*, 2018]. Architectural details are provided in Appendix C.2. To anchor results, we additionally establish a human performance baseline based on results from 25 games played by 5 participants (ages 24 - 28; see Appendix F).

As shown in Table 1 (top), all agents severely underperform the human baseline. Among agents, despite YLEs graph-based nature, the CNN encoder outperformed both GNNs and the RM in all metrics except SEE (see Figure 3, left). We hypothesise this is due to YLE's dynamic spatial layout: card positions and relationships change frequently, which may be better captured by CNNs than by message-passing. In contrast to humans (SEE 65%), agents failed to solve the task reliably or finish early. This suggests that the YLE requires more than memory: It demands the ability to reason about asymmetric knowledge to maintain common ground over time and to act under uncertainty.

### 5.2 RQ1b: Explicit memory modules show stronger zero-order ToM than implicit ones

While the perfect memory setting establishes an upper bound, it assumes agents retain all past observations without

| Agent | Memory | R ↑ | SR in % ↑ | NC ↑ | SEE in % ↑ |
|---|---|---|---|---|---|
| Human | - | $5.7 \pm 2.5$ | $95.0 \pm 25.8$ | $3.0 \pm 0.0$ | $65.0 \pm 48.8$ |
| GCN-M | perfect | $0.9 \pm 0.5$ | $26.8 \pm 13.4$ | $1.6 \pm 0.3$ | $0.0 \pm 0.0$ |
| GATv2-M | perfect | $1.9 \pm 0.4$ | $58.2 \pm 13.5$ | $2.3 \pm 0.2$ | $0.0 \pm 0.0$ |
| RM-M | perfect | $1.6 \pm 1.1$ | $50.7 \pm 35.0$ | $2.2 \pm 0.6$ | $0.0 \pm 0.0$ |
| CNN-M | perfect | $\mathbf{2.3 \pm 0.3}$ | $\mathbf{71.8 \pm 8.8}$ | $\mathbf{2.6 \pm 0.1}$ | $0.0 \pm 0.0$ |
| CNN-LSTM | implicit | $0.2 \pm 0.1$ | $3.9 \pm 4.1$ | $0.8 \pm 0.3$ | $1.0 \pm 1.8$ |
| CNN-GRU | implicit | $0.3 \pm 0.2$ | $2.3 \pm 0.3$ | $0.7 \pm 0.1$ | $\mathbf{2.1 \pm 1.8}$ |
| CNN-S5 | implicit | $0.1 \pm 0.1$ | $3.0 \pm 3.8$ | $0.7 \pm 0.2$ | $0.1 \pm 0.2$ |
| CNN-TrXL | explicit | $\mathbf{1.5 \pm 0.3}$ | $\mathbf{43.5 \pm 9.7}$ | $\mathbf{2.1 \pm 0.2}$ | $0.0 \pm 0.0$ |
| ToMTrack | explicit | $\mathbf{1.6 \pm 0.3}$ | $\mathbf{48.3 \pm 10.7}$ | $\mathbf{2.2 \pm 0.2}$ | $0.0 \pm 0.0$ |

Table 1: Agent performance on the **3x3** YLE. The first block compares encoder architectures with access to perfect memory (-M). The second block evaluates different temporal architectures with the best-performing CNN encoder. There, we compare implicit and explicit memory modules. Best results are in **bold**. ToMTrack (final block) is the best method overall when not given perfect memory.

limitation. To better reflect realistic constraints and to asses whether agents can reason over their own internal knowledge representations, we evaluated models with different temporal architectures. This tests their capacity to track and use information over time – a key aspect of zero-order ToM. We tested LSTM- [Hochreiter and Schmidhuber, 1997], GRU [Cho *et al.*, 2014], S5- [Smith *et al.*, 2023], and TransformerXL-based [Dai *et al.*, 2019, TrXL] architectures, paired with the best-performing CNN encoder. S5 has recently excelled on memory tasks in [Samsami *et al.*, 2024], while TrXL includes an explicit memory mechanism shown effective in memory-intensive RL [Parisotto *et al.*, 2020].

As shown in Table 1 (middle) and Figure 3 (right), TrXL outperformed all other temporal models ($p < 0.05$, one-sided paired t-test). Nonetheless, there is still a performance gap between TrXL and the best perfect memory baseline CNN-M. Contrary to expectations from prior work [Samsami *et al.*, 2024], S5 offered no clear advantage over traditional RNNs. LSTM, GRU, and S5 agents occasionally ended games early. This behaviour appears to reflect a form of *gambling*, as agents terminate the game after very few actions and without sufficient information-gathering. This suggests a failure to track beliefs over time, as agents that end early do so without sufficient inference – contrasting with human players, whose early-ending behaviour reflects confident ToM reasoning. This suggests a failure to reliably recall past observations, leaving guessing as the primary strategy. This tendency is further reflected in the short average game length ($22.9 \pm 15.5$, $14.0 \pm 15.6$ and $30.4 \pm 1.42$, respectively). TrXL agents never finish early.

Taken together, these results indicate that explicit memory architectures, such as TrXL, are better suited for zero-order ToM reasoning. Nonetheless, the gap in performance relative to the perfect memory baseline demonstrates the difficulty of learning robust belief-tracking strategies in YLE.

### 5.3 RQ2: Modelling beliefs over cards and distilling partner policies improves results

Given the partial observability of the YLE, agents must maintain a belief $b$ over the hidden colours $c$ of all cards.

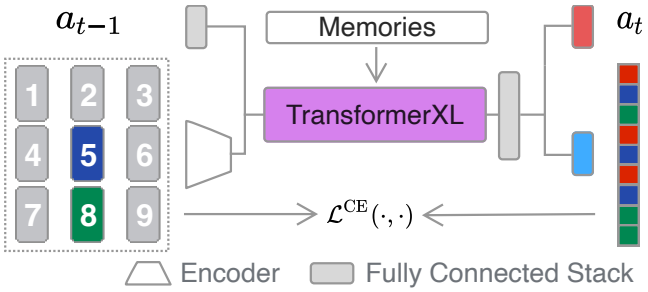


Figure 4: Our ToMTrack agent generates beliefs over card colours using an auxiliary prediction head.

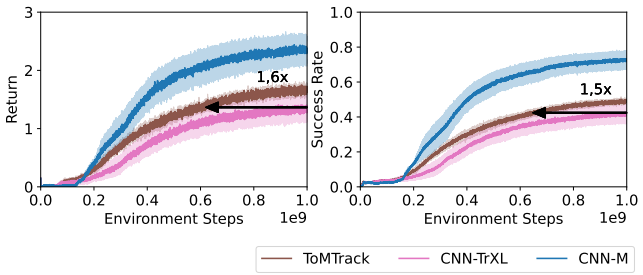


Figure 5: **Left**: compares return. **Right**: shows success rate. ToMTrack trains at least $1.5\times$ faster.

To support this, we designed ToMTrack, a method for learning an implicit model over cards' colour (see Figure 4). ToMTrack is an online representation learning RL approach inspired by previous work on offline belief representation learning [Wang *et al.*, 2023]. At each timestep $s_t$, the policy $\pi$ should learn useful representations that contain information about the cards' colours, no matter if they have been previously observed or inferred (first-order ToM, recall Figure 1). We thus want our policy to model $p(c_i|s_t)$, $\forall\, 0 < i < |Y|$. To achieve this, we added an auxiliary belief classification head $f_{\text{belief}}$ to the policy network. This head maps the intermediate representation learned by the policy to a distribution over possible card colours. During training, we used the

ground truth card colours as labels. We used an auxiliary belief prediction task [Jaderberg *et al.*, 2017] to train $f_{\text{belief}}$. Specifically, we added a cross-entropy loss $\mathcal{L}^{\text{CE}}$ to the actor, $\mathcal{L'}_{\text{a}} = \mathcal{L}_{\text{a}} + \lambda\mathcal{L}_{a}^{\text{CE}}$, and critic loss, $\mathcal{L'}_{\text{c}} = \mathcal{L}_{\text{c}} + \lambda\mathcal{L}_{c}^{\text{CE}}$; where $\lambda \geq 0$ is a hyperparameter.

In addition, since information is often implicitly communicated using actions, we add a second input encoder to explicitly process the last action taken by the active agent. This enables the policy to explicitly interpret observed behaviour and update its beliefs about card colours accordingly, see Figure 1.

We also evaluated other baselines based on the simplified action decoder (SAD) [Hu and Foerster, 2020] or intrinsic motivation, but found that they underperformed, see Appendix H. In contrast, `ToMTrack` outperforms CNN-TrXL ($48.3$ vs $43.5$ SR) and decreases the gap to CNN-M ($71.8$ SR) (Table 1). These results demonstrate that modelling latent beliefs and interpreting others' actions improves agent coordination.

### 5.4 RQ3: Agents do not exhibit transferable models of partner belief inference

Effective strategies for tracking one's own and others' beliefs should generalise to novel partners. We test this via zero-shot coordination [Hu *et al.*, 2020, ZSC], where agents must cooperate with independently trained partners at test time. Success in ZSC indicates that agents do not rely on partner-specific conventions but instead learn to track beliefs, interpret actions, and reason about others' knowledge in a generalisable way.

We first evaluate ZSC performance of the previously trained CNN-M and `ToMTrack` agents using cross-play (XP), where agents are paired with separately trained partners. Figure 6 shows that XP SR drops by $88.5\%$ on average, indicating a strong reliance on learned conventions.

To address this, we retrain agents using other-play [Hu *et al.*, 2020, OP], which prevents overfitting to arbitrary conventions by exploiting environment symmetries. In Hanabi [Bard *et al.*, 2020], colour symmetry alone (OP+C) suffices for strong ZSC [Hu *et al.*, 2020]. We find that this is not the case in YLE. As shown in Figure 6, while both CNN-M and `ToMTrack` trained with OP+C improve XP performance, a significant SP-XP performance gap remains. Unlike Hanabi, YLE introduces spatial-temporal complexity and thus additional symmetry classes (see Appendix C.4), e.g. agents could form conventions around locations. We therefore also test OP+C+R, using rotational symmetries to break spatial conventions. This improves XP further for CNN-M and `ToMTrack`. For CNN-M it nearly closes the SP-XP gap ($33.2$ vs $27.4$).

However, while OP+C+R improves XP, this comes at a significant gap in overall SR (e.g. $86.5$ vs $41.5$ SP in CNN-M), suggesting that non-OP agents rely heavily on spatial and colour conventions. When these are unavailable, coordination suffers – implying that belief tracking and action interpretation remain difficult to learn directly. This suggests that CNN-M and `ToMTrack` do not acquire general belief strategies out-of-the-box, though OP-trained variants may begin to.

To further test agents' ability to perform first-order ToM reasoning, i. e, to infer a partner's beliefs and intentions, we designed a diagnostic task (see Figure 7). In this test, only agents with first-order ToM will correctly identify that they can group all blue cards with one move, despite not being able to observe all blue cards first-hand. Grouping blue cards is optimal as it increases the number of card clusters by one, increasing the chance of finishing successfully or possibly early.

As shown in Table 2, we find that even the best-performing ZSC agents fail to select optimal actions and instead make moves consistent with either zero-order or no ToM reasoning. This shows that, despite OP improvements, agents lack transferable models of belief inference.

In summary, *agents in YLE struggle to track beliefs of novel partners*. Unlike Hanabi, success in YLE likely cannot be achieved by symmetry-breaking alone: as additional symmetries are removed, performance declines. This suggests the bottleneck is not symmetry identification but rather the absence of learning mechanisms that foster generalisable ToM reasoning.

### 5.5 RQ4: Agents struggle to represent knowledge of self and others over longer timespans

To succeed in the YLE, agents must maintain a belief over all cards' colours. However, our results so far suggest that models are not using beliefs correctly. To identify whether agents fail to represent beliefs or instead only fail to use them, we apply linear probes [Alain and Bengio, 2016] to the agents' hidden activations at each timestep of the game, to predict the colour of each card individually [Matiisen *et al.*, 2022; Bortoletto *et al.*, 2024a; Bortoletto *et al.*, 2024b; Zhu *et al.*, 2024]. This approach assesses whether belief-relevant information is linearly decodable from the learned embeddings – whether the card's colour was directly observed, inferred, or remains uncertain (see Appendix D for details).

Figure 8 shows that agents start to form internal representations accurately in the early-to-mid-game. However, performance degrades in the later stages. This suggests that agents struggle to retain and update beliefs over longer timespans, especially as cards move and the spatial arrangement grows more complex. Together, this indicates a challenge in representing dynamic beliefs in YLE.

### 5.6 RQ5: Agents struggle to scale belief reasoning in four-player YLE

To test whether agents can scale to higher-order Theory of Mind (ToM) reasoning, we evaluate them in the four-player version of YLE. In this setting, each player observes fewer cards directly, and knowledge is distributed across more teammates. As a result, coordination increasingly depends on inferring what others know and what others believe about each other's knowledge. For example, reasoning about whether agent 1 understands what agent 2 knows about agent 3's interpretation of agent 4's hint involves third-order ToM.

This demand is reflected in performance: as shown in Figure 9, `ToMTrack` exhibits increased variance in both SR and R, and mean SR drops by over $10$ percentage points compared to the two-player setting. Note that the four-player

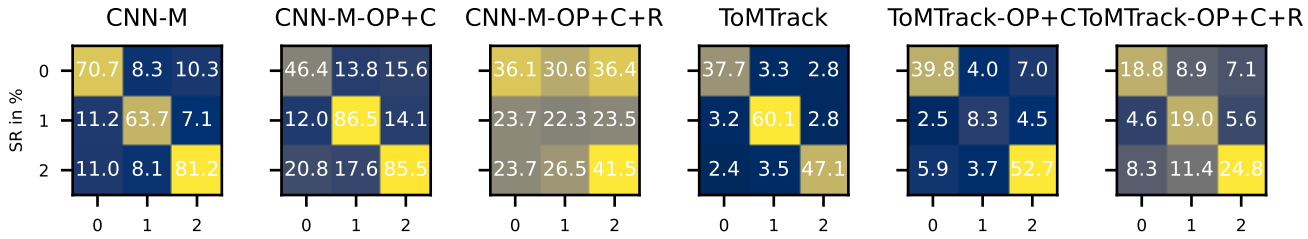


Figure 6: Zero-shot coordination performance measured by SR. We report additional metrics in Appendix E.2. Self-play agents do not learn to coordinate with others. Other-Play agents (OP) have some success at partner generalisation. However, the additional complexity of YLE requires more symmetries as input to OP (+C+R) compared to Hanabi, which only requires +C.

| Agent | Memory | T0 ↓ | T1 ↓ | Wrong ↑ |
|---|---|---|---|---|
| CNN-M-OP+C+R | Perfect | $13.8 \pm 0.5$ | $86.5 \pm 14.2$ | $30.3 \pm 3.1$ |
| ToMTrack-OP+C+R | Explicit | $12.6 \pm 0.9$ | $98.8 \pm 28.3$ | $28.2 \pm 2.7$ |

Table 2: Results from the diagnostic ToM test in Figure 7. We report the average rank (based on action probabilities) of zero-order (T0), first-order (T1), and incorrect actions (e.g., mismatched card colours). We find that agents fail to make zero- or first-order inferences in this simple test.

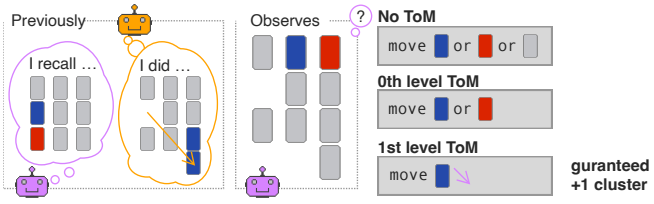


Figure 7: ToM reasoning influences 🤖 's decision. Without ToM reasoning, no move is optimal. A zero-order agent (memory-based) may move either the red or blue card, as both would group two cards. A first-order agent that correctly interprets 🤖 's action (see Figure 1) will recognise that moving the blue card completes a full cluster. Agents are rewarded during training for forming complete clusters. An example of second-order ToM reasoning can be found in Appendix B.3.

game allows higher absolute returns due to more hint cards (see Figure 9, left). However, as reflected in the success rate, this does not suggest better coordination. Our results suggest that scaling ToM reasoning in multi-agent RL remains an open challenge, especially in settings that require maintaining shared beliefs across multiple agents over time.

### 5.7 Key takeaways & Limitations

Across all RQs, agents, even with explicit memory or belief modelling, fail to maintain accurate beliefs or coordinate reliably over time. They overfit to partner-specific conventions and struggle with belief tracking over longer horizons. Notably, humans successfully end games early in 65% of cases, while agents very rarely do so – even with perfect memory or belief modelling (see Table 1). This finding *positions SEE as a compelling metric for assessing belief reasoning*, as it reflects agents' ability to infer shared beliefs and identify coordination success. We present two main takeaways: **First**, ToM reasoning must be evaluated with novel partners to avoid mistaking brittle conventions for genuine ToM — ToM tests must acknowledge the generality constraint [Rakoczy, 2012]. **Second**, YLE offers a challenging environment for assessing common ground reasoning under uncertainty, even in its simplest 2-player 3×3 setting. It supports future work to explore scaling to more agents and the original game size. Both enable an investigation of increasingly complex ToM settings.

While the YLE significantly expands our ability to evaluate key elements of ToM and advance cooperative agents, it also has limitations: It currently abstracts away modalities used in human ToM, such as language or gaze, and focuses on isolating belief-based reasoning. Additionally, it focuses solely on cooperation and does not address belief modelling in competitive or mixed-motive settings. These simplifications allow for focused evaluation of core ToM challenges before addressing the full complexity of human ToM in more real-world settings.

## 6 Related Work

Current benchmarks study ToM reasoning in AI mostly from a static observer's perspective [Baker *et al.*, 2011; Baker *et al.*, 2017; Rabinowitz *et al.*, 2018; Bara *et al.*, 2021; Bara *et al.*, 2023; Fan *et al.*, 2021; Duan *et al.*, 2022; Kim *et al.*, 2023; Bortoletto *et al.*, 2024c; Gandhi *et al.*, 2023], framed as a supervised learning task – an approach with notable limitations [Aru *et al.*, 2023; Bortoletto *et al.*, 2024a]. Other benchmarks evaluated agents in interactive environments, such as the Hanabi Learning Environment (HLE) [Bard *et al.*, 2020] and SymmToM [Sclar *et al.*, 2022]. In contrast to Hanabi, Yōkai requires spatial-temporal belief reasoning, requiring agents to bind beliefs to moving cards [Fernandez *et al.*, 2023]. Compared to SymmToM, YLE offers greater strategic depth and stronger pressure to develop ToM. In SymmToM, heuristics outperform all baselines and communication is unrestricted, while in YLE, misused hints can irreversibly lock cards in

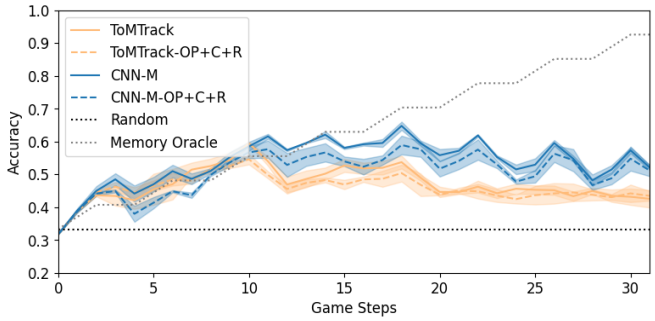


Figure 8: Probing accuracies for each step. *Memory Oracle* observes new cards whenever possible and recalls them perfectly. Agents fail to track colours over time, especially after the mid-game.

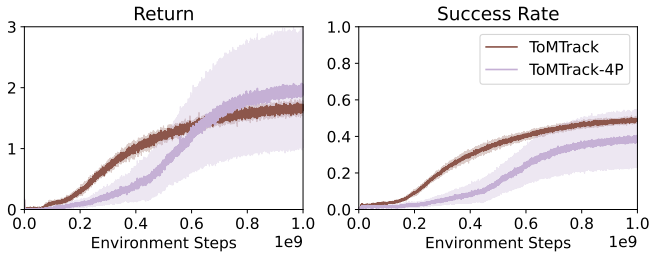


Figure 9: Performance comparison of the two-player (`ToMTrack`) and four-player (`ToMTrack-4P`) **3x3** YLE. 4-player YLE requires tracking common ground between more players, and results in lower performance with higher variance.

losing positions. Unlike Hanabi and SymmToM, YLE also introduces a high-stakes belief-based decision: agents may end the game early for a higher payoff, sacrificing future information and relying more on their beliefs.

A second line of work investigated whether neural networks represent mental states internally [Bortoletto *et al.*, 2024a; Zhu *et al.*, 2024]. For example, Matiisen et al. [2022] probed DRL agents for intention representations in cooperative navigation. We extend probing to YLE, where beliefs evolve dynamically and asymmetric knowledge emerges over time – showing the value of probing in a more complex environment.

Finally, YLE contributes to the broader family of cooperative multi-agent RL (MARL) benchmarks [Samvelyan *et al.*, 2019; Carroll *et al.*, 2019; Bard *et al.*, 2020; Ruhdorfer *et al.*, 2024], but with a strong emphasis on ToM and common ground reasoning. While many techniques were developed for Hanabi, our results indicate they do not transfer directly to the YLE, thus highlighting the new challenges posed by our benchmark and requiring new modelling approaches.

## 7 Conclusion & Future Work

We introduced the Yōkai Learning Environment – a multi-agent RL benchmark where effective collaboration requires belief tracking over space and time. Implemented in JAX, YLE supports end-to-end GPU training at hundreds of thousands of steps per second. We show that RL agents on the YLE struggle to retain knowledge, to generalise across partners, or to identify when coordination has succeeded – a high-stakes decision reflected in their inability to end games early. YLE thus offers a scalable, diagnostic testbed for advancing common ground reasoning in collaborative AI.

While our work focused on evaluating RL agents, future work might use this benchmark to evaluate other systems, like language agents [Xi *et al.*, 2023; Wang *et al.*, 2024]. We envision that Yōkai can play a major role in isolating and assessing critical aspects of these systems that realte to memory, spatial reasoning and common ground tracking.

## Acknowledgements

The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting C. Ruhdorfer. We especially thank R. Yang for implementing a prototype of the environment as part of her master's thesis, and A. Penzkofer and L. Shi for numerous insightful discussions.

## References

[Abdessaied *et al.*, 2024] Adnen Abdessaied, Lei Shi, and Andreas Bulling. Vd-gr: Boosting visual dialog with cascaded spatial-temporal multi-modal graphs. In *Proc. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5805–5814, 2024.

[Alain and Bengio, 2016] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[Aru *et al.*, 2023] Jaan Aru, Aqeel Labash, Oriol Corcoll, and Raul Vicente. Mind the gap: challenges of deep learning approaches to theory of mind. *Artificial Intelligence Review*, 56(9):9141–9156, January 2023.

[Ba *et al.*, 2016] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[Baker *et al.*, 2011] Chris Baker, Rebecca Saxe, and Joshua Tenenbaum. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[Baker *et al.*, 2017] 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(4):0064, 2017.

[Bara *et al.*, 2021] Cristian-Paul Bara, Sky CH-Wang, and Joyce Chai. MindCraft: Theory of mind modeling for situated dialogue in collaborative tasks. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1112–1125, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[Bara *et al.*, 2023] Cristian-Paul Bara, Ziqiao Ma, Yingzhuo Yu, Julie Shah, and Joyce Chai. Towards collaborative

plan acquisition through theory of mind modeling in situated dialogue. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 2958–2966, 2023.

[Bard *et al.*, 2020] 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, 2020.

[Bortoletto *et al.*, 2024a] Matteo Bortoletto, Constantin Ruhdorfer, Adnen Abdessaied, Lei Shi, and Andreas Bulling. Limits of theory of mind modelling in dialogue-based collaborative plan acquisition. In *Proc. 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–16, 2024.

[Bortoletto *et al.*, 2024b] Matteo Bortoletto, Constantin Ruhdorfer, Lei Shi, and Andreas Bulling. Benchmarking mental state representations in language models. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024.

[Bortoletto *et al.*, 2024c] Matteo Bortoletto, Constantin Ruhdorfer, Lei Shi, and Andreas Bulling. Explicit modelling of theory of mind for belief prediction in nonverbal social interactions. In *Proc. 27th European Conference on Artificial Intelligence (ECAI)*, 2024.

[Bortoletto *et al.*, 2024d] Matteo Bortoletto, Lei Shi, and Andreas Bulling. Neural reasoning about agents' goals, preferences, and actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 456–464, 2024.

[Bradbury *et al.*, 2018] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[Brody *et al.*, 2021] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[Carroll *et al.*, 2019] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Clark, 1996] Herbert H Clark. *Using language*. Cambridge university press, 1996.

[Dafoe *et al.*, 2020] Allan Dafoe, Edward Hughes, Yoram Bachrach, Tantum Collins, Kevin R. McKee, Joel Z. Leibo, Kate Larson, and Thore Graepel. Open Problems in Cooperative AI, December 2020.

[Dafoe *et al.*, 2021] Allan Dafoe, Yoram Bachrach, Gillian Hadfield, Eric Horvitz, Kate Larson, and Thore Graepel. Cooperative ai: machines must learn to find common ground, 2021.

[Dai *et al.*, 2019] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[DeepMind *et al.*, 2020] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

[Duan *et al.*, 2022] Jiafei Duan, Samson Yu, Nicholas Tan, Li Yi, and Cheston Tan. Boss: A benchmark for human belief prediction in object-context scenarios. *arXiv preprint arXiv:2206.10665*, 2022.

[Fan *et al.*, 2021] Lifeng Fan, Shuwen Qiu, Zilong Zheng, Tao Gao, Song-Chun Zhu, and Yixin Zhu. Learning triadic belief dynamics in nonverbal communication from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7312–7321, 2021.

[Fernandez *et al.*, 2023] Jorge Fernandez, Dominique Longin, Emiliano Lorini, and Frédéric Maris. A logical modeling of the yōkai board game. *AI Communications*, (Preprint):1–34, 2023.

[Foerster *et al.*, 2016] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

[Foerster *et al.*, 2019] Jakob Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1942–1951. PMLR, 09–15 Jun 2019.

[Fukushima, 1975] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20:121–136, 1975.

[Gandhi *et al.*, 2021] Kanishk Gandhi, Gala Stojnic, Brenden M Lake, and Moira R Dillon. Baby intuitions benchmark (bib): Discerning the goals, preferences, and actions of others. *Advances in neural information processing systems*, 34:9963–9976, 2021.

[Gandhi *et al.*, 2023] Kanishk Gandhi, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah Goodman. Understanding social reasoning in language models with language models. *Advances in Neural Information Processing Systems*, 37, 2023.

[Hamon, 2024] Gautier Hamon. transformerXL_PPO_JAX, July 2024.

[Harris *et al.*, 2020] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[Heek *et al.*, 2023] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[Hu and Foerster, 2020] Hengyuan Hu and Jakob N Foerster. Simplified action decoder for deep multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2020.

[Hu *et al.*, 2020] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "Other-play" for zero-shot coordination. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4399–4410. PMLR, 13–18 Jul 2020.

[Hunter, 2007] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[Jaderberg *et al.*, 2017] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017.

[Kim *et al.*, 2023] Hyunwoo Kim, Melanie Sclar, Xuhui Zhou, Ronan Le Bras, Gunhee Kim, Yejin Choi, and Maarten Sap. Fantom: A benchmark for stress-testing machine theory of mind in interactions. *arXiv preprint arXiv:2310.15421*, 2023.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Klein *et al.*, 2005] Gary Klein, Paul J Feltovich, Jeffrey M Bradshaw, and David D Woods. Common ground and coordination in joint activity. *Organizational simulation*, 53:139–184, 2005.

[Klien *et al.*, 2004] G. Klien, D.D. Woods, J.M. Bradshaw, R.R. Hoffman, and P.J. Feltovich. Ten challenges for making automation a "team player" in joint human-agent activity. *IEEE Intelligent Systems*, 19(6):91–95, 2004.

[Matiisen *et al.*, 2022] Tambet Matiisen, Aqeel Labash, Daniel Majoral, Jaan Aru, and Raul Vicente. Do deep reinforcement learning agents model intentions? *Stats*, 6(1):50–66, December 2022.

[Matthews *et al.*, 2024] Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Thomas Jackson, Samuel Coward, and Jakob N. Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. *CoRR*, abs/2402.16801, 2024.

[Mitchell, 2009] Jason P Mitchell. Inferences about mental states. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1309–1316, 2009.

[Nikulin *et al.*, 2023] Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Viacheslav Sinii, Artem Agarkov, and Sergey Kolesnikov. XLand-minigrid: Scalable meta-reinforcement learning environments in JAX. In *Intrinsically-Motivated and Open-Ended Learning Workshop @NeurIPS2023*, 2023.

[Oguntola *et al.*, 2023] Ini Oguntola, Joseph Campbell, Simon Stepputtis, and Katia Sycara. Theory of mind as intrinsic motivation for multi-agent reinforcement learning, 2023.

[Oliehoek and Amato, 2016] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer International Publishing, 2016.

[Parisotto *et al.*, 2020] Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M. Jayakumar, Max Jaderberg, Raphaël Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[Pillow, 1989] Bradford H Pillow. Early understanding of perception as a source of knowledge. *Journal of experimental child psychology*, 47(1):116–129, 1989.

[Premack and Woodruff, 1978] David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526, 1978.

[Rabinowitz *et al.*, 2018] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International conference on machine learning*, pages 4218–4227. PMLR, 2018.

[Rakoczy, 2012] Hannes Rakoczy. Do infants have a theory of mind? *British Journal of Developmental Psychology*, 30(1):59–74, 2012.

[Ruhdorfer *et al.*, 2024] Constantin Ruhdorfer, Matteo Bortoletto, Anna Penzkofer, and Andreas Bulling. The overcooked generalisation challenge. *arXiv preprint arXiv:2406.17949*, 2024.

[Rutherford *et al.*, 2023] Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, et al. Jaxmarl: Multi-agent rl environments in jax. In *Second Agent Learning in Open-Endedness Workshop*, 2023.

[Samsami *et al.*, 2024] Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering memory tasks with world models. In *The Twelfth International Conference on Learning Representations*, 2024.

[Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, page 2186–2188, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Sclar *et al.*, 2022] Melanie Sclar, Graham Neubig, and Yonatan Bisk. Symmetric machine theory of mind. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 19450–19466. PMLR, 17–23 Jul 2022.

[Smith *et al.*, 2023] Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[Southgate *et al.*, 2007] V. Southgate, A. Senju, and G. Csibra. Action anticipation through attribution of false belief by 2-year-olds. *Psychological Science*, 18(7):587–592, 2007. PMID: 17614866.

[Suddendorf and Corballis, 2007] Thomas Suddendorf and Michael C. Corballis. The evolution of foresight: What is mental time travel, and is it unique to humans? *Behavioral and Brain Sciences*, 30(3):299–313, June 2007.

[Team, 2020] The Pandas Development Team. pandas-dev/pandas: Pandas. February 2020.

[Tomasello, 2010] Michael Tomasello. *Origins of human communication*. MIT press, 2010.

[Treutlein *et al.*, 2021] Johannes Treutlein, Michael Dennis, Caspar Oesterheld, and Jakob Foerster. A new formalism, method and open issues for zero-shot coordination. In *International Conference on Machine Learning*, pages 10413–10423. PMLR, 2021.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Virtanen *et al.*, 2020] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[Wang *et al.*, 2023] Andrew Wang, Andrew C. Li, Toryn Q. Klassen, Rodrigo Toro Icarte, and Sheila A. McIlraith. Learning belief representations for partially observable deep rl. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[Wang *et al.*, 2024] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024.

[Xi *et al.*, 2023] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng

Qiu, Xuanjing Huang, and Tao Gui. The rise and potential of large language model based agents: A survey, 2023.

[Ying *et al.*, 2021] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[Yu *et al.*, 2022] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[Zambaldi *et al.*, 2018] Vinícius Flores Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David P. Reichert, Timothy P. Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew M. Botvinick, Oriol Vinyals, and Peter W. Battaglia. Relational deep reinforcement learning. *CoRR*, abs/1806.01830, 2018.

[Zhu *et al.*, 2024] Wentao Zhu, Zhining Zhang, and Yizhou Wang. Language models represent beliefs of self and others. *arXiv preprint arXiv:2402.18496*, 2024.

# A Appendix: Infrastructure & Tools

**Servers and software** We ran our experiments on two different server systems. Both are running Ubuntu 24.04. One is equipped with NVIDIA Tesla V100-SXM2 GPUs with 32GB of memory and Intel Xeon Platinum 8260 CPUs. The other is equipped with NVIDIA H100-NVL GPUs with 94GB of memory and AMD EPYC 9454 CPUs. All training runs are executed on a single GPU only. We trained our models using JAX [Bradbury *et al.*, 2018], Flax [Heek *et al.*, 2023] and Optax [DeepMind *et al.*, 2020]. In terms of tools, we did our analysis using NumPy [Harris *et al.*, 2020], Pandas [Team, 2020], SciPy [Virtanen *et al.*, 2020] and Matplotlib [Hunter, 2007]. Our TransformerXL implementation is based on the one provided by [Hamon, 2024]. Our single-file implementation of MAPPO is based on the one provided by JaxMARL [Rutherford *et al.*, 2023] with additional inspiration taken from [Hamon, 2024].

**Accesability and reusability** The Yōkai Learning Environment (YLE) integrates directly into JaxMARL and implements its standard environment interface. As a result, any baseline method implementation of JaxMARL can be adapted to our environment quickly and with minimal effort. This makes our open-source environment easily accessible to future researchers who will benefit from our reference implementations and those provided by the wider community. It also enables inter-environment comparison, facilitating broader benchmarking across cooperative multi-agent tasks. The project is accessible under https://git.hcics.simtech.uni-stuttgart.de/public-projects/Yokai_env.

**Training times and compute requirements** Training times vary between server setup and agent. Our longest training agent is `ToMTrack`-OP+C+R since it needs to compute memory updates for the TransformerXL and the asymmetric domain randomisation used in Other-Play. Experiments with `ToMTrack`-OP+C+R take 8 hours to complete per seed on our H100 setup and around 24 hours on our V100 setup. The CNN-M baselines train quickest: 6 hours on the H100 server and 14 hours on a V100 server. All our experiments are runnable on a single V100 with 32GB VRAM.

# B Fast, Scalable, and Parallel Yōkai in Graph-based JAX

We first describe in detail how our environment operates and then share additional details on reinforcement learning in the YLE.

## B.1 JAX-based Implementation

We carefully engineer our environment to be end-to-end usable on a GPU directly and make it just-in-time compilable using JAX. This design enables both simulation and learning to occur entirely on the GPU, avoiding costly GPU–CPU communication and significantly improving efficiency. As a result, the environment supports training at hundreds of thousands of steps per second on modern hardware, enabling fast experimentation with relatively modest compute resources.

This efficiency and scalability stem primarily from how we represent and update the environment's state using adjacency matrices $A_t$ at each timestep $s_t$. In the following, we describe two practical examples of how this is achieved.

**Efficiently computing legal moves** We always mask illegal actions and thus need to compute the set of legal actions at every step. While identifying observable cards and placable hint cards is computationally inexpensive, computing legal next moves is by far the most costly computation our environment performs.

To determine which card moves are legal in a given state $s_t$, the environment must simulate each possible action $a \in A$ and assess whether the resulting state $s_{t+1}$ is valid. A state is considered legal if (1) all cards are connected into a single, fully-connected graph via their neighbours, and (2) no cards overlap spatially.

To make this efficient, we compute all potential next states in parallel:

$$S_{t+1} = \{s_{t+1} | s_{t+1} = \mathcal{T}(s_t, a) \forall a \in \mathcal{A}\}. \tag{4}$$

Each state $s_{t+1}$ is associated with an adjacency matrix $A_t$ that reflects the card layout after a move. For each candidate action (i.e., moving a card $y \in Y$ to a new grid position $(x, y)$), we first verify that the position is within bounds and the card is movable.

We then check whether each resulting state is legal by evaluating its adjacency matrix. Specifically, we compute the reachability matrix:

$$R = (I + A)^{|Y|-1} \tag{5}$$

The state is legal if all entries in $R$ are positive, i.e., $R[i][j] > 0$ for all $i, j$. This test, implemented as a parallel matrix power operation, allows us to efficiently filter legal moves at each step.

While designing environments for JAX requires careful consideration, these efforts potentially yield significant

| Num Envs $n$ | 512 | 1,024 | 2,048 |
|---|---|---|---|
| 2-player **3x3** | 139,107 | 274,669 | 559,251 |
| 2-player **4x4** | 135,662 | 144,636 | 148,269 |

Table 3: We evaluate steps-per-second (SPS) of our environment on a single Nvidia H100 by taking random actions in $n$ parallel environments. Our environment scales to hundreds of thousands of SPS.

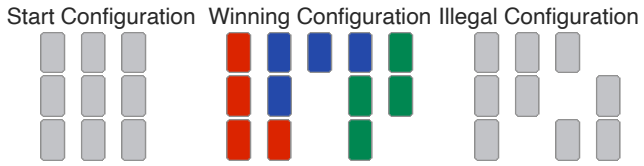


Figure 10: Examples of start, winning, and illegal card configurations for a Yōkai game with nine cards and three colours.

speedups that make RL research accessible to a broad range of researchers.

**Efficiently executing moves** In our graph-based setup, executing environment transitions is highly efficient. Observing a card simply involves unmasking its colour in the node feature matrix $F_t$. To execute a move, we update the adjacency matrix $A_t$ to reflect the new spatial configuration. Specifically, when a card with index $i \in [0, |Y|)$ is moved, we first remove its current edges by setting:

$$A_t[j, i] = 0 \text{ and } A_t[i, j] = 0 \quad \forall j \in [0, |Y|) \tag{6}$$

Next, we compute a binary vector $v$ such that $v_k = 1$ if card $k$ is a spatial neighbour of the new position of card $i$, and 0 otherwise. We then update the adjacency matrix by setting:

$$A_{:,i} = v \text{ and } A_{i,:} = v. \tag{7}$$

**Resulting Performance** We present how our environment performs in terms of steps-per-second across many parallel environments. All experiments were run by taking 1000 random consecutive actions in $n$ parallel environments. Table 3 shows results. Our environment scales similarly to other recently proposed JAX-based RL environments, e.g. [Rutherford *et al.*, 2023; Nikulin *et al.*, 2023; Ruhdorfer *et al.*, 2024; Matthews *et al.*, 2024].

## B.2 Reinforcement Learning in the YLE

We want to continue discussing additional aspects of reinforcement learning in the YLE.

**Playing Yōkai on a Grid** While in the original game, cards can be placed in any legal configuration, our environment limits the configuration to be in a predefined field of size $g \times g$ where $g = 9$ in the 3x3 and $g = 10$ in the 4x4 version. This enables the efficient computation of legal moves but prohibits some rare card configurations. While computing the exact number of disallowed configurations is infeasible, we find them to be extremely uncommon in practice.

For example, let's assume a 2-player 3x3 game on a 9x9 grid. In such a setting, there a 4 hints and thus a total of 8 move actions for both players together. Cards are initially placed in a square in the centre of the board, leaving 3 empty spaces in the grid towards either side. In theory, thus, if agents agree to start placing cards in a row towards one of these sides, they would run out of space towards the border after 3 moves. In practice, this is seldom a suitable strategy and thus does not usually occur in games of Yōkai. This is because players want to group cards and are limited by the number of moves. They thus do not tend to start grouping cards in random directions, but rather group cards where they initially find a certain colour.

To verify that this is also true for our agents, we count the number of times when, in an evaluation game, an agent places a card on a border position. Note this is an overestimation as it does not account for the cases where agents place a card on the border but do not intend to place further cards in the same direction. We find that policies rarely to never use to border region of the board. Our best-performing policy `ToMTrack` uses the border $0.0 \pm 0.0\%$ of the time when evaluated over all three seeds in 1000 games each. We also discuss this in section 5.7.

**The action space in detail** To observe two cards, agents first pick from $|Y|$ and then from $N_{cards} - 1$ cards. Next, an agent moves a card by picking one of $N_{moves}$ actions, which can be understood to take card $0 \leq i < |Y|$ and place it at $(x, y) \in g \times g$. Finally, the active agent reveals a hint from the hint pile or places a hint onto a card by picking from $|H| + |H| \cdot |Y|$ actions. Consequently, the maximum episode length is $8|H|$ while the minimum is 1. The number of actions is therefore

$$N_a = 1 + |Y| + N_{moves} + |H| + N_{hint\_moves} + 1. \tag{8}$$

For example, the action space is of size $1,068$ for 2-player **3x3** games and $2,322$ for the **4x4** version.

**YLE is a symmetric ToM environment** [Sclar *et al.*, 2022] defines symmetric ToM environments as environments that (1) have a symmetric action space, (2) feature imperfect information, (3) allow the observation of others, and (4) require information-seeking behaviour. YLE features all four and thus qualifies. First, the action space is symmetric (see above). Second, information is imperfect since agents can never observe all cards first-hand and, for most of the game, only know a very limited subset of all cards. Third, it allows the observation of others both using explicitly encoded actions, as in `ToMTrack` or by observing their moves indirectly via their observations. Fourth, information needs to be acquired by observing cards first-hand or, second-hand, by interpreting moves and reasoning about knowledge. Agents that do not seek information and, for instance, repentantly observe the same card, will fail the game.

**Number of Players, Hints and the Difficulty** How many and which hint cards are available depend on the environment version and the number of players. For a 2-player game, there are 2 hints of one, 3 hints of two and 2 hints of three colours, i.e. 7 in total. Since there are 4 possible one-, 6 possible two- and 4 possible three-colour hints, players get a random subset of these hints at the start of the game per category. We give a full overview of all possible variations in Table 4. The number of hints directly influences the maximum number of turns in the game. This is because the game ends when

| | 4x4 | | | 3x3 | | |
|---|---|---|---|---|---|---|
| **Number of Players** | **2** | **3** | **4** | **2** | **3** | **4** |
| Number of 1-colour hint cards | 2 | 2 | 3 | 1 | 2 | 3 |
| Number of 2-colours hint cards | 3 | 4 | 4 | 3 | 3 | 3 |
| Number of 3-colours hint cards | 2 | 3 | 3 | - | - | - |
| Sum | 7 | 9 | 10 | 4 | 5 | 6 |

Table 4: Number of hint cards according to the number of players and the environment version. The 3x3 version features three colours and thus has no three-colour hints.

all hints have been used. This means that games take longer if more players are added. Note, though, that this usually does not make the game easier. Instead, the number of cards one player can directly observe themself decreases with the number of players. For 4x4 YLE in the 2-player version, any agent can observe up to 14 of the 16 cars themselves (8 of 9 in 3x3 Yōkai). This shrinks to 10 of 16 for the 4x4 version and 6 of 9 for the 3x3 version when 4 players take part. Note how in the 3- and 4-player versions, agents are thus required to both work with less first-hand information and need to rely more on hints and interpreting the moves made by those around them. Not only this, but agents are tasked to do this tracking for more players as well. Thus, adding players to the YLE increases the difficulty substantially in two ways.

**Additional Details on Legal Positions** We display several examples of legal target positions given a card to be moved in Figure 11. These only display a subset of all legal moves since the set of legal moves is different for each card on the board. For some cards, there are no legal moves whatsoever. This typically occurs when a card occupies a position in a narrow band that connects two clusters, such as in example 3. In example 3, there is no conceivable target position for both cards marked with an orange dot that would keep the cluster of cards connected, except their current position. Note that cards might be movable in the future again, i.e. if other cards are moved such that cards are freed again. In example 3, for instance, if the card with the blue dot is placed above any of the cards marked with an orange dot, both cards gain one additional legal target position, which makes them movable. Moving cards in a way that keeps option value (i.e. that keeps many cards moveable) is thus recommended. Observe example 4, for instance, here players placed cards in such a way that for most cards, very few target positions are still legal. This will usually highly restrict the agents' ability to sort cards.

### B.3 Second Order ToM Reasoning Example

Figure 12 displays an example for second-order ToM reasoning in 2-player Yokai. It shows an agent ( ) making inferences about what they assume the other agents ( ) knowledge is due to their own previous action. In the example infers the colour of the card due to the behaviour of the other agent. reasons that because of how played, must have interpreted his move correctly. thus reasons that thinks that knew that cards 1 and 2 have the same colour and must go together. This is the case, made this inference correctly, and card 3 indeed is also blue. This reasoning allows agents to require less first-hand observations to get an overview of the board, potentially enabling them to end early.

Note that an interesting strategic aspect of Yokai is the amount of shared information agents acquire together (both and choose to observe card 2) versus how much information they try to acquire asymmetrically to then use to hint knowledge to others via hints or actions (cards 1 and 3). Agents thus effectively control how big their shared knowledge is and when to strategically expand it. This has interesting strategic depth. Agents will differ in their strategic preference here, and developing strategies that are suitable for a wide range of partners is therefore a challenge.

This is just one illustrative example of ToM reasoning in YLE. Belief reasoning in the YLE can range from recalling previously observed cards (zero-order ToM), to tracking what another player knows (first-order), to inferring what one player believes about another's beliefs, including their beliefs about oneself (second-order and beyond).

Note that such higher-order reasoning steps are even likelier to occur in settings involving more partner agents.

## C Training Details

We share all the details of the training here, including all hyperparameters used in training and all details on the network architectures and how we tuned them. In general, we have tuned many aspects of our policy to arrive at the strongest agents we could for our experiments, see below. Finally, the number of parameters per model is shown in Table 6. We took extra care in making these equal to make our results comparable.

### C.1 Reinforcement Learning Details

Our work employs a standard MAPPO algorithm [Yu *et al.*, 2022] for all experiments. MAPPO is an extension of PPO [Schulman *et al.*, 2017] to the multi-agent setting. It separates the actor and the critic network. Since the actor and critic are separated and the critic is discarded after training, MAPPO allows giving the critic access to privileged information during training. This leads to the good performance of MAPPO in cooperative multi-agent tasks, compare [Yu *et al.*, 2022], which is why we choose it in our work. We present all parameters of the learning process in Table 5.

### C.2 Neural Network Architectures

Our work employs several encoder architectures. We present all the details on them below. In general, we aimed to make

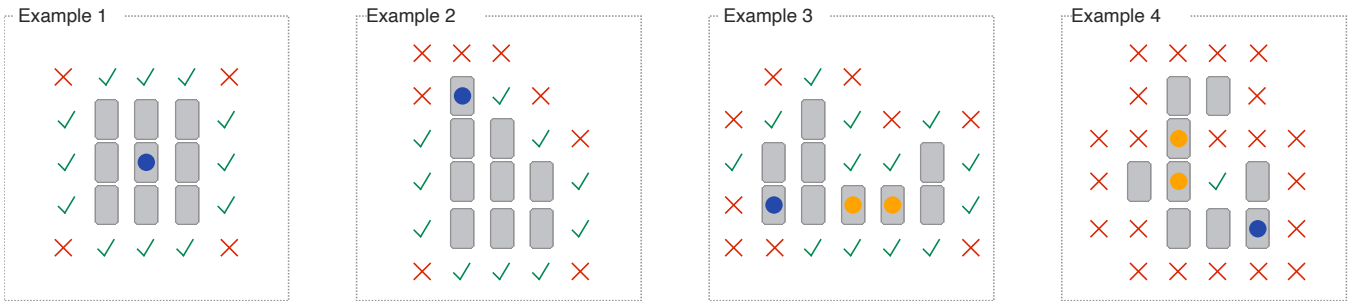


Figure 11: Valid target positions for the card with the blue dot. Cards can only be moved so that all cards remain connected via their sides. Cards with an orange dot cannot be moved at all legally. Example 1 shows the initial position. All cards can be moved in the initial configuration. Example 2 stems from the early game. Examples 3 and 4 display configurations that might be encountered during the mid or late game.

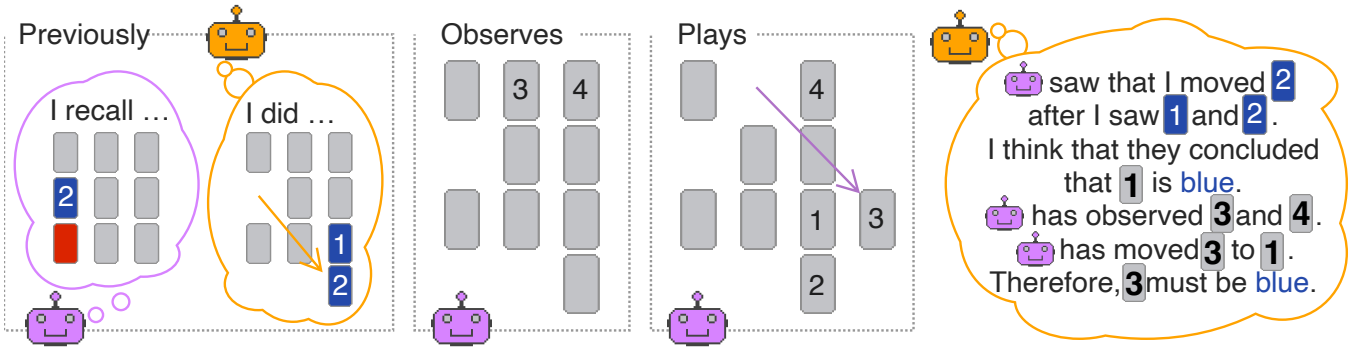


Figure 12: A second-order ToM reasoning example over multiple timesteps in 2-player YLE. beliefs that beliefs that he ( ) knew that cards 1 and 2 are of the same colour. Even though never saw card 1 and never observed card 3, both now know where all blue cards are, that they are grouped and that both share this knowledge as part of their common ground. In the future, they can now both observe and move different cards to finish early as they have one less card that needs to be observed.

comparisons fair by giving all compared models a similar number of parameters, also see Table 6. The biggest advantage in terms of parameter size was RM-M for the spatial reasoning and CNN-LSTM for the temporal reasoning experiment, but neither performed the best.

### General Policy Architecture

In general, in our work, no matter the encoder, the final policy network always has the same structure. That is given an embedding $e_{\text{encoder}}$ produced by an encoder network $f_{\text{encoder}}$ and possibly a memory architecture, we pass it to a stack of dense layers and either a value or action head. We always use $4$ dense layers before the heads with ReLU [Fukushima, 1975; **?**] as the activation function between the layers and a hidden dimension of $128$. The layers are initialised using orthogonal initialisation with a scale of $\sqrt{2}$ and bias $0$. We have determined the number of policy layers and their size using hyperparameter tuning. We compared $2$, $3$ and $4$ layers with hidden dimension $32$, $64$ and $128$. We found that the largest configuration outperforms the others. Additionally, for all neural network architectures detailed below, we also varied and explored: The number of encoder blocks ($2$, $3$ and **4**), the hidden dimension of the encoder blocks, the aggregation function after the encoder (mean, sum, multiply and **concatenate**) and whether to anneal the dense rewards (False and **True**). Finally, the value head produces a singular scalar output and is initialised using orthogonal initialisation with a scale of $0.00$ instead of $\sqrt{2}$. We initialise the action head in the same way but it produces logits for all possible actions instead. The critic network is always parameterised the same as the actor network, but the input is different. Since we are using MAPPO our critic receives as input a so-called *world state* that combines the individual observations of all agents. This is possible in MAPPO due to centralised training and decentralised execution [Foerster *et al.*, 2016]. As the world state, we concatenate the individual observations along the feature dimension.

### GCN, GATv2 and the pre-embedding of Nodes

We use two graph neural networks in our work, one based on graph convolutions [Kipf and Welling, 2016] and one based on the attention mechanism [Brody *et al.*, 2021]. We pick these two as they stem from two different families of graph learning architectures and they thus are representative of a larger body of work. In our work, both methods observe the graph directly as described in the paper. There are two additional implementation details to note, though. First, both graph architectures observe an input graph that features a virtual node, similar to the hub-nodes of [Abdessaied *et al.*, 2024] or the `[VNode]` token of [Ying *et al.*, 2021]. A vir-

| Description | Value |
|---|---|
| Optimiser | Adam [Kingma and Ba, 2015] |
| Adam $\beta_1$ | 0.9 |
| Adam $\beta_2$ | 0.999 |
| Adam $\epsilon$ | $1 \cdot 10^{-5}$ |
| Learning Rate $\eta$ | $3 \cdot 10^{-4}$ |
| Learning Rate Annealing | Yes (linear) |
| Maximum Grad Norm | 0.5 |
| # Simulators | 1024 |
| Discount Rate $\gamma$ | 0.99 |
| GAE $\lambda$ | 0.95 |
| Entropy Coefficient | 0.01 |
| Value Loss Coefficient | 0.5 |
| # PPO Epochs | 4 |
| # PPO Minibatches | 4 |
| # PPO Steps | 128 |
| PPO Value Loss | Clipped |
| PPO Value Loss: Clip Value | 0.2 |
| PPO Value Loss: Scale Clip Value | No |
| Reward Shaping | Yes (linearly annealed) |

Table 5: Hyperparameters of the learning process.

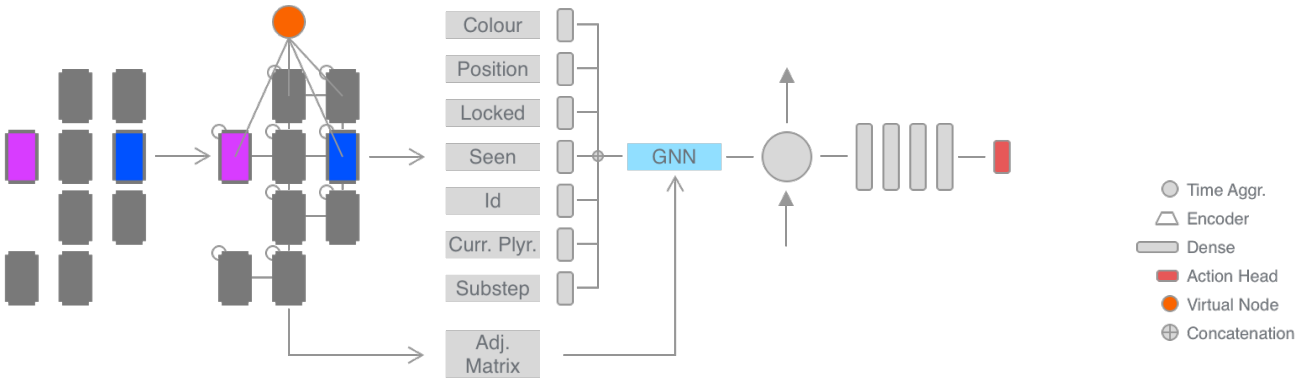


Figure 13: A sketch of the GNN and pre-embedding architecture used in this work.

| Model | # Parameters |
|---|---|
| GCN-M | 317,048 |
| GATv2-M | 317,560 |
| RM-M | 354,168 |
| CNN-M | 322,156 |
| CNN-LSTM | 470,252 |
| CNN-GRU | 437,484 |
| CNN-S5 | 388,524 |
| CNN-TrXL | 400,940 |
| `ToMTrack` | 404423 |
| `ToMTrack-SAD` | 417863 |

Table 6: Number of parameters of all models used in this work.

tual node is an empty node that connects to all other nodes in the graph. It solely serves the purpose of reducing hop distances between nodes. The motivation for employing them in the YLE is that since cards can be distant from each other in terms of hop distance, information from one card might not propagate to the next during message passing. I.e. imagine the GNN observes two blue cards on opposite sides of the current card cluster. If the number of message-passing rounds is too low, this information might never be aggregated during message-passing. In RL, neural networks tend to be comparatively small, which is also true in our work. This also limits the number of message-passing steps to be small (2 in our case). With a virtual node, however, this is not an issue since the virtual node aggregates information from all other nodes in one single message-passing step. Second, our work uses pre-embeddings of node-feature modalities as described in [Bortoletto *et al.*, 2024d]. We take the node features of size $n \times f$ and embed them per modality via a separate dense

encoder before then concatenating all of the resulting vectors into new node features shaped $n \times f'$. $f'$ depends on the size of these dense layers. We sketch the overall approach in Figure 13. In general, we found that embedding the node features per modality before using them in the GNN improved performance following earlier work, i.e. see [Bortoletto *et al.*, 2024d]. Specifically, for each modality $m \in M$ we train a dense layer $f_m$ that embeds the modality per node into an embedding $e_m \in \mathbb{R}^{b \times t \times n \times 2}$ for the current player embedding or $e_m \in \mathbb{R}^{b \times t \times n \times 4}$ for all other modalities where $b$ is the batch size, $t$ the time and $n$ the number of nodes. All embeddings are then concatenated into a node embedding $e_{\text{node}} = e_{\text{colour}} \oplus e_{\text{position}} \oplus e_{\text{locked}} \oplus e_{\text{seen}} \oplus e_{\text{id}} \oplus e_{\text{is\_current\_player}} \oplus e_{\text{substep}}$ with $e_{\text{node}} \in \mathbb{R}^{b \times t \times n \times f'}$ with $f' = 26$ using the configuration described. We then add self-connections, the virtual node using a zero-vector $e_{\text{virt}} \in \{0\}^{f'}$ and connect the virtual node to all real nodes before passing them to our graph encoder $f_{\text{GNN}}$. The output dimension of the graph encoder is $64$. For GATv2, we use $4$ heads. We apply $4$ graph layers in the relational encoder comparison. The resulting embedding is then concatenated along the node dimension and fed to the rest of the policy.

**Relation Module**

Our work adapts the relation module of [Zambaldi *et al.*, 2018]. We only use the self-attention mechanism of their proposed relation module and skip the CNN encoder they propose alongside the self-attention layers. This is because our setting deals with an environment where agents observe the position of cards directly as opposed to the setting of [Zambaldi *et al.*, 2018] that deals with image inputs directly. Thus, in our relation module, non-local interactions between all cards are simply computed using self-attention directly. Similarly to the GNN encoders, we also pre-embed the modalities of all nodes using the same mechanism as discussed above. The resulting embedding is then also passed to the remaining policy. Our relational module is also applied $4$ times, features $4$ heads, and its output dimension per layer is $64$.

**CNN**

Our work compares a CNN encoder and determines it to be the best-performing model. As described earlier, the YLE returns an image-like observation to be processed by our CNN. The exact size of the input tensor to the CNN depends on the YLE settings, but in 2-player 3x3 YLE, the input dimension is $9 \times 10 \times 13$, which we will use as a running example. It includes the same features as discussed above, with some minor changes. The first three channels include a one-hot encoding of cards' colours, the next three an encoding of hint colours, followed by two channels for the position, one for whether the card is locked, one for whether the observing agent is the current player, one for the number of the current substep (1-4) and one that identifies the card via an identifier number. Our convolutional stack uses $4$ 2d convolutions with $64$ filters each and ReLU applied between them. The kernel size is $(3, 3)$, we apply a stride $1$ and valid padding. The resulting embedding is flattened to produce the final embedding to be passed on to later layers.

**LSTM and GRU**

Next to the encoder architectures, our work compares several memory architectures. Here, memory architecture is used as a term to denote classical RNNs as well as structured state space or transformer [Vaswani *et al.*, 2017] architectures. LSTMs [Hochreiter and Schmidhuber, 1997] and GRUs [Cho *et al.*, 2014] are commonly used in RL and such are natural first approaches for our environment. They provide an implicit memory through their hidden state that is computed at every step from the current observation encoding and the previous hidden state. In both cases, we take the embedding generated by the encoder, $e_{\text{encoder}}$, pass it through a dense layer $f_{\text{pre\_rnn}}$ before finally applying either the LSTM or GRU on the resulting embedding. $f_{\text{pre\_rnn}}$ embeds $e_{\text{encoder}}$ to the hidden dimension of the RNN. As the hidden dimension of the RNN we pick $128$ in our work. We also experimented with using $256$ but found no difference in performance. We use orthogonal initialisation for both the GRU and the LSTM.

**S5**

As already described, we use S5 [Smith *et al.*, 2023] layers because of the great performance of structured state-space models on memory tasks in deep model-based RL [Samsami *et al.*, 2024]. Similar to LSTM and GRU, S5 only provides an implicit memory mechanism. We use a similar setup as with both the LSTM and GRU layers, i.e. we also use a $f_{\text{pre\_s5}}$ embedding layer that embeds $e_{\text{encoder}}$ to the hidden dimension of our S5 layer. We use $2$ S5 blocks, $1$ layer, place the LayerNorm [Ba *et al.*, 2016] before the S5 layer and use the activation function proposed in the original work [Smith *et al.*, 2023]. As before we use $128$ as the hidden dimension of S5.

**TransformerXL**

Finally, as already introduced TransformerXL [Dai *et al.*, 2019] is an explicit memory mechanism that shows great performance in memory-intensive RL tasks when used correctly [Parisotto *et al.*, 2020]. Our usage strictly follows the one described by [Parisotto *et al.*, 2020] as implemented by [Hamon, 2024]. Similar to the other memory architectures, we also first pass the embedding received by the encoder through a dense layer $f_{\text{pre\_trxl}}$ before handing the resulting embedding to the transformer. While LSTM, GRU and S5 use $128$ as their hidden dimension, we only use a hidden dimension of $64$ for both the dense layers within the transformer as well as the projection layers in multi-head attention. This is to keep the number of parameters consistent between the architectures. We furthermore configure the transformer to have $8$ heads and to use gating with a gating bias of $2$. We use a single transformer layer. Finally, we set the length memory window to $32$. $32$ is long enough to recall all of the previous moves in the 3x3 YLE. Memories from a past game – as marked by the `done` signal – are masked. We apply standard positional encoding based on sinus waves.

## C.3 ToMTrack

`ToMTrack` is a direct extension of the CNN-TrXL baselines and shares all the parameters mentioned earlier with both the CNN as well as the TransformerXL work. There are two extensions we add to arrive at `ToMTrack`. The

| Agent | Memory | R ↑ | SR in % ↑ | NC ↑ | SEE in % ↑ |
|---|---|---|---|---|---|
| ToMTrack w/o $f_{\text{belief}}$ | explicit | $0.9 \pm 0.5$ | $31.2 \pm 14.8$ | $1.8 \pm 0.4$ | $0.0 \pm 0.0$ |
| ToMTrack w/o action | explicit | $1.4 \pm 0.4$ | $41.2 \pm 11.7$ | $2.0 \pm 0.3$ | $0.0 \pm 0.0$ |
| ToMTrack | explicit | $\mathbf{1.6 \pm 0.3}$ | $\mathbf{48.3 \pm 10.7}$ | $\mathbf{2.2 \pm 0.2}$ | $0.0 \pm 0.0$ |

Table 7: ToMTrack ablations. We evaluate ToMTrack without the additional belief representation learning and the additional action encoding and find that only both together improve performance.

first one – the auxiliary belief prediction task – is already mostly described in the main text. To generate the distribution over possible card colours, we add a colour prediction head $f_{\text{belief}}$ to our policy. $f_{\text{belief}}$ takes as input the embedding generated by the stack of dense layers and produces an output $e_{\text{belief}} \in \mathbb{R}^{b \times t \times |Y| c}$ that resembles the prediction over all cards and colours. For the **3x3** version $|Y|c = 9 \cdot 3 = 27$. To obtain the final predictions, we reshape $e_{\text{belief}}$ to be of shape $b \times t \times |Y| \times c$ and apply the Softmax function of the last dimension to get a prediction per card. The method only has one hyperparameter $\lambda$ – which balances the losses - and we pick $\lambda = 0.1$ in this work. The second extension – the active action decoding – is realised via an additional input stream. Before passing the previous action to this stream, we first decode it into a reduced dimension that serves as a dense representation of that action of shape $4 + |Y| + 2 + |H| + 1$. The first $4$ bits encode the move type alias the substep. The next $|Y|$ bits encode the Yōkai card affected by the current action, if no card is affected, all are set to $0$. The next $2$ bits encode the position affected by the action. For instance, if the previous active action was a move action, the resulting coordinates are encoded there. The next $|H|$ bits encode whether and if so which hint card is used in an action, and finally, the last bit encodes whether the previous action was to terminate the game. For a 2-player **3x3** YLE game $4 + |Y| + 2 + |H| + 1 = 20$. Note that this is a much denser representation compared to just encoding the raw number. The resulting encoding $e_{\text{action}}$ is processed by its own dense layer and then concatenated with the output of the encoder before being passed to the rest of the policy network. Note that we have ignored any hidden or memory state in the above formulation for notational convenience.

### C.4 Other-Play in Yōkai

The core idea of Other-Play (OP) is that during self-play training, agents learn to break symmetries of the environment in arbitrary ways that do not generalise to novel partners. In Hanabi, for instance, agents might form conventions around certain colours. OP presents agents with a special asymmetric domain randomisation that makes it impossible for agents to coordinate on breaking symmetries in certain ways. In Hanabi, this is achieved by a recolouring operation that shows each agent a different recolouring of the game state [Hu *et al.*, 2020].

Therefore, to train agents via OP for the YLE, one first needs to find all classes of symmetries that exist in an environment, and then, second, find an appropriate domain randomisation operation. Generally, since agents also observe coloured cards in the YLE, we can apply the same recolouring operation as Hu *et al.*. However, since the YLE features a spatial grid, additional symmetries exist around the location of cards. Even if agents can not coordinate on specific colours when we apply recolouring, they are, for example, able to coordinate on sorting the first colour they find in a specific place. Agents, for instance, might choose to observe the same two cards at the start and place one of them in the top left corner. This is an arbitrary convention (top left vs bottom right etc.). To prevent this, we apply asymmetric rotation to the agents' observations. This way, when one agent places a card next to the top-left corner in their environment, their partner agents observe this as placing the card next to potentially any corner in different episodes. Training with rotated observations requires to also requires rotating the actions of the agent appropriately. We show an example of this process in Figure 14.

Note that these are not the only ways for agents to learn to break symmetries. Recent work [Treutlein *et al.*, 2021] showed that agents also might coordinate to break symmetries in arbitrary ways over multiple timesteps. It is impossible to list all possible classes of symmetries for an environment with enough complexity. This is a drawback of the OP algorithm as it requires them as input. In our work, we thus focus on showing that while Hanabi only requires breaking one symmetry for efficient ZSC while the YLE – due to its additional complexities – requires additional effort to improve ZSC performance, which is still far from optimal.

## D Additional Probing Details

Over the duration of $1000$ games, we collect a probing dataset $\mathcal{D} = \{x^{(i)}, y^{(i)}\}$. The probing datasets map the activations of an agent to a label $z$. A *probe* then is a function $g : f_l(x) \rightarrow \hat{z}$ and trained on $\mathcal{D}$ where $f$ is a neural network and $f_l(x)$ is the intermediate representation of $x$ at layer $l$. In our case, we train simple logistic regressions. We always probe before the action head of our agent. As highlighted in the main text, during all experiments, we train one and evaluate it on a hold-out test set $\mathcal{D}_{\text{test}} \subset \mathcal{D}$. Additionally, note that many of these tasks feature imbalanced datasets since the does-not-know label is far more likely to occur than any other label. We correct for that by weighing the loss per class and subsampling the does-not-know label to the largest other class. During all experiments, we train one probe per card – 9 in total – and average the scores computed on a hold-out test over all 9.

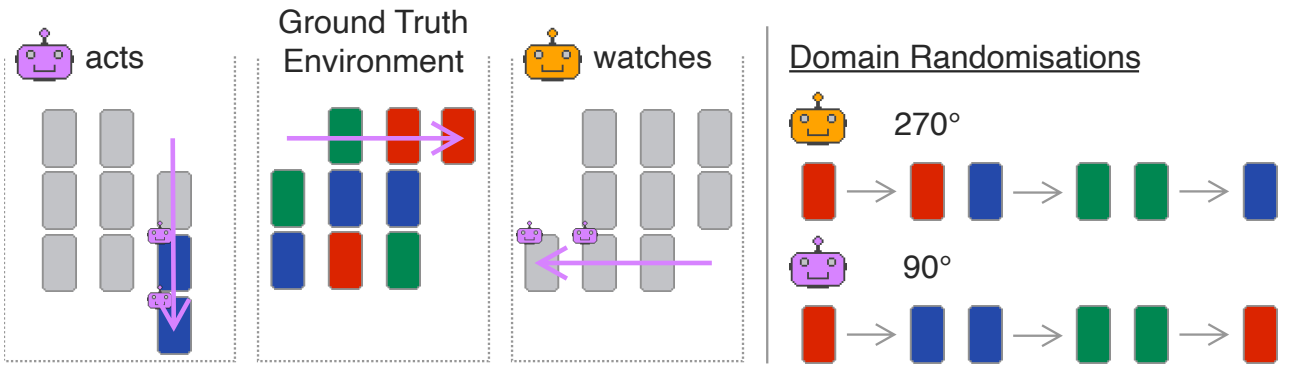


Figure 14: An example of how both agents act in rotated and recoloured environments to break symmetries for zero-shot coordination.

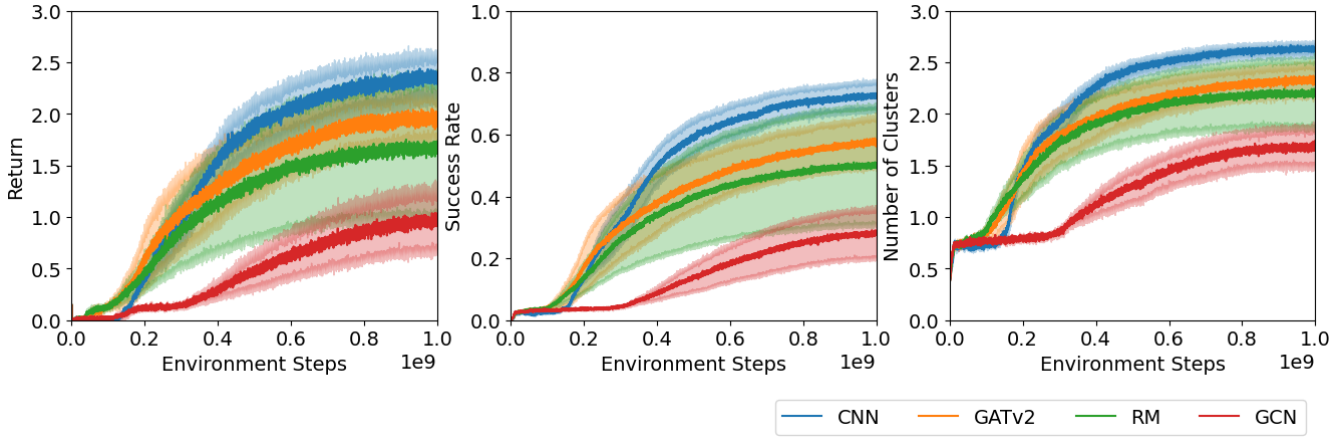


Figure 15: Return, success rate and numbers of clusters over time per encoder while given access to perfect memory.

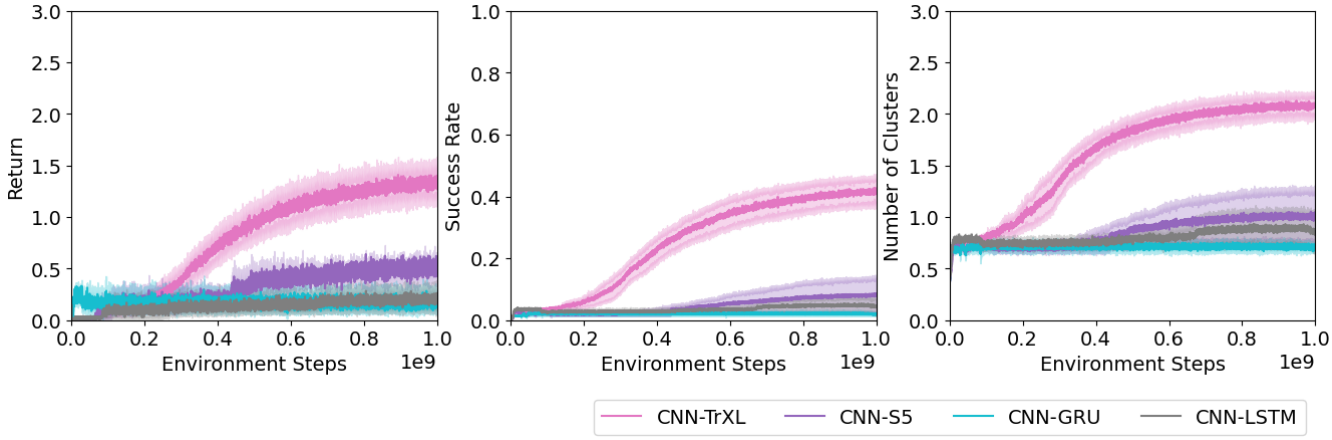


Figure 16: Return, success rate and numbers of clusters over time per memory architecture.

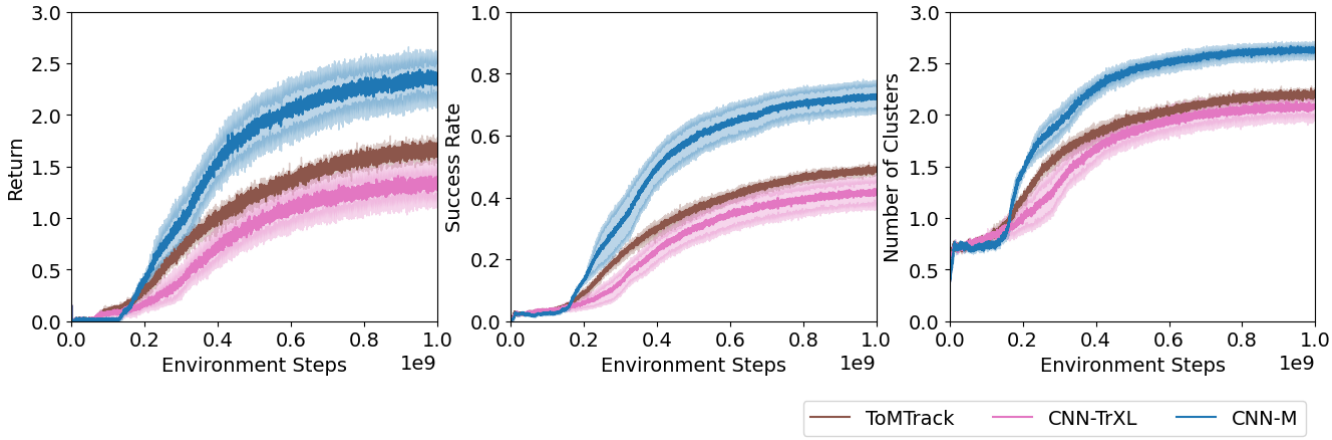


Figure 17: Return, success rate and numbers of clusters over time for `ToMTrack`, `ToMTrack-SAD`, CNN-TrXL and the CNN-M architecture.

# E Additional Experimental Results

## E.1 Additional Training Curves

For completeness, we show how the most important metrics of our agents developed over time. Figure 15 shows the curves of the spatial-reasoning experiment, Figure 16 of the memory-reasoning experiment and lastly Figure 17 reports the same for `ToMTrack`.

## E.2 Additional Zero-Shot Coordination Metrics

In Figure 18 we show additional metrics for the ZSC evaluation in our main body of work. Training curves for our other-play agents are shown in Figure 19 to show convergence behaviour.

## E.3 2- versus 4-player Games

In Figure 20 we show how two- and four-player settings compare for our best model, `ToMTrack`.

# F Human Performance in Yōkai

We collected human game performance in a setting that adheres to the same setting that artificial agents face, i.e. participants played with 9 cards on a 9x9 grid in a two-player setting. Humans played in person. A pair of subjects was instructed on the rules verbally. Each pair of subjects was given a single training round to make sure that all participants were familiar with the rules. To make results as comparable as possible, humans were instructed to avoid any form of non-verbal communication. The study was conducted under an umbrella agreement of the relevant institutional review board. Compensation was also handled under the rules of the relevant institution.

# G Example Games

We show an example game in which `ToMTrack` wins in self-play in Figure 21 and one where it loses in Figure 22.

# H Details on Other Attempts

While `ToMTrack` provides a mechanism for learning representations that represent card colours, it does not necessarily encourage the agents to maximise their shared or individual knowledge. Consequently, we were inspired to also experiment with a variant of `ToMTrack` that is not trained via an auxiliary task but rather uses $\mathcal{L}_{\text{CE}}$ as intrinsic motivation for both agents; `ToMTrack-I`. `ToMTrack-I` agents are encouraged to form an implicit belief model that aligns well with the true state of the environment by adding an intrinsic reward $r_i$ to the reward based on the negative cross-entropy loss of the actor classification head:

$$r = r + \lambda_{\text{s}} * r_{\text{s}} + r_{\text{i}}, \text{ where } r_{\text{i}} = -\mathcal{L}_a^{\text{CE}}. \tag{9}$$

In this formulation, the agents thereby receive a higher reward when the belief model's prediction loss $\mathcal{L}_a^{\text{CE}}$ is low. This approach is loosely inspired by [Oguntola *et al.*, 2023]. We originally assumed that this would achieve superior results but found that the intrinsic reward confused the agents to a point where they barely learned a useful policy altogether, falling below baseline performance (i.e. CNN-TrXL). Future work should study more sophisticated ideas based on this concept.

Previous work on Hanabi has proposed the simplified action decoder (SAD) [Hu and Foerster, 2020]. SAD agents can take two actions during training that other agents can observe: a normal environment action and a greedy action. Other agents are allowed to observe both. The intuition is that during exploration, the true intentions of agents are blurred with explorative actions and thus less informative for others. [Hu and Foerster, 2020] implements their idea on top of epsilon-greedy Q-learning. Since during evaluation $\epsilon = 0$, all actions are greedy and are given as input to both action inputs heads of other agents. During evaluation, $\epsilon = 0$ and both actions are greedy. SAD showed improved performance in Hanabi, including the Bayesian action decoder [Foerster *et al.*, 2019]. We have adapted this idea as `ToMTrack-SAD` to our policy learning approach. Since we do not have an $\epsilon$ to set, we adopt their idea by also allowing agents to take two actions during

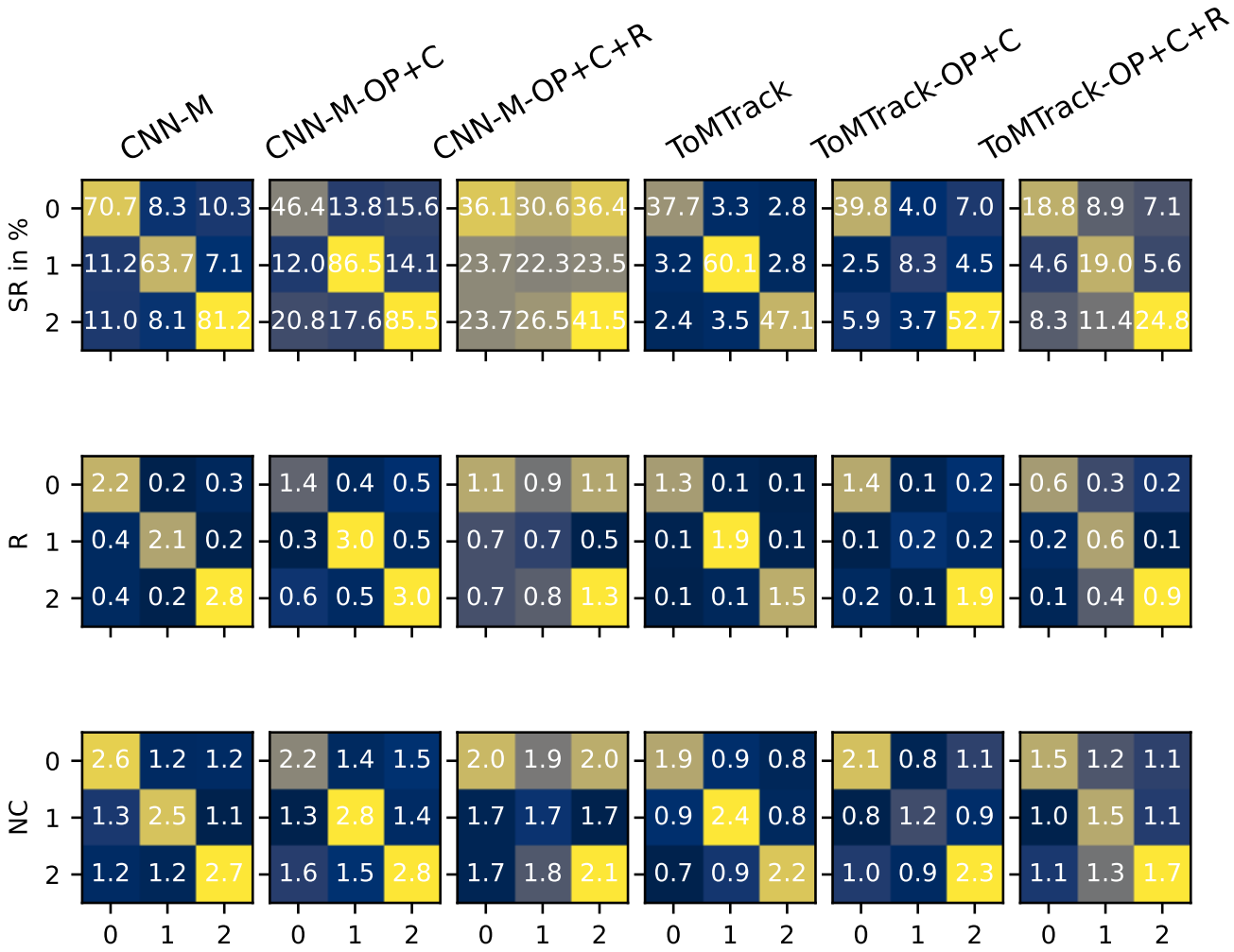


Figure 18: Zero-shot coordination performance measured by success-rate, average return and number of cards clustered. SP and OP+C are consistently outperformed by OP+C+R, showing that beyond cards' colour, agents need to focus on other reasoning types.

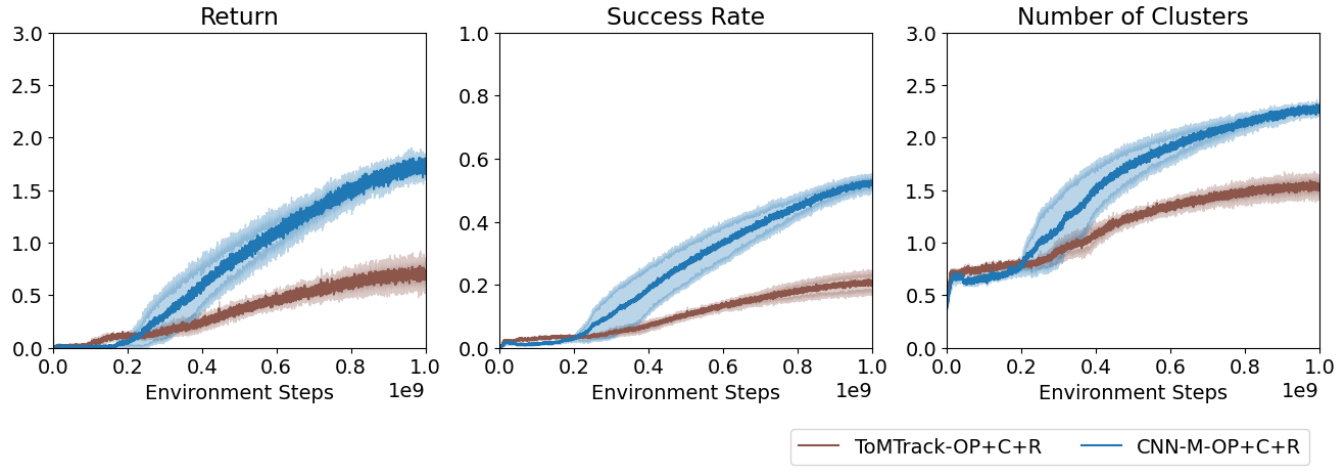


Figure 19: We show training curve metrics for our agents trained with other-play to demonstrate convergence. Recall that these metrics measure self-play performance rather than ZSC. ZSC metrics are shown in Figure 6 and Figure 18.

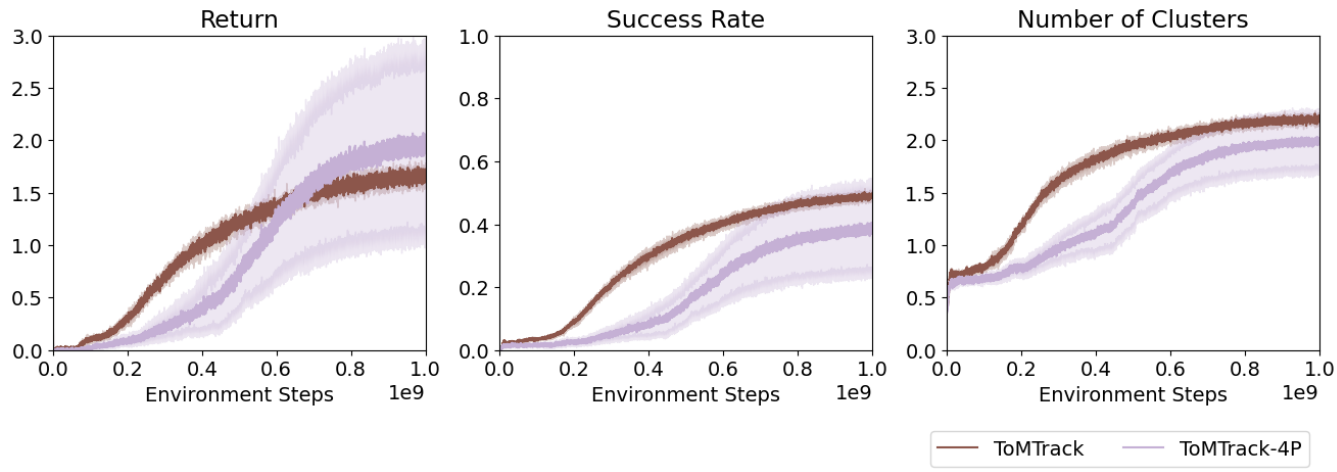


Figure 20: `ToMTrack` in the two- and four-player **3x3** YLE. In four-player YLE more hints are available to be placed, and as such, we observe higher total rewards collected by the `ToMTrack-4P` agent, even though it wins fewer games and clusters fewer colours.

| **Agent** | **Memory** | **R** ↑ | **SR in %** ↑ | **NC** ↑ | **SEE in %** ↑ |
|---|---|---|---|---|---|
| `ToMTrack` (SAD) | explicit | $0.7 \pm 0.4$ | $21.5 \pm 13.4$ | $1.5 \pm 0.4$ | $0.0 \pm 0.0$ |

Table 8: Additional `ToMTrack-SAD` results. This approach underperforms our baselines from Table 1 considerably and was thus excluded from the main paper.

training: One sampled from the agents' produced action distribution is used as the environment action that gets executed, and one greedy action is determined as the legal action with the highest associated logit. The true environment action is duplicated during evaluation with similar reasoning.

`ToMTrack-SAD` is our extension of `ToMTrack`. In `ToMTrack-SAD` both actions are encoded as described earlier and concatenated before being fed to the same dense layer. Since we use MAPPO and no Q-learning approach as in the original work [Hu and Foerster, 2020], our agents pick their normal environment action given their current set of parameters $\theta_a$ in state $s_t$ via

$$a_{\text{env}} \sim \pi(\cdot|s_t, \theta_a) \tag{10}$$

and their second greedy action as

$$a_{\text{greedy}} = \arg\max_i \pi(a_i|s_t, \theta_a). \tag{11}$$

Both are encoded and processed as described above when discussing `ToMTrack`. Note that we have ignored any hidden or memory state in the above formulation for notational convenience.

Results in Table 8 show that `ToMTrack-SAD` performs worse than CNN-TrXL or other baselines. We speculate that including greedy actions as input is uninformative in an environment with many actions like the YLE.

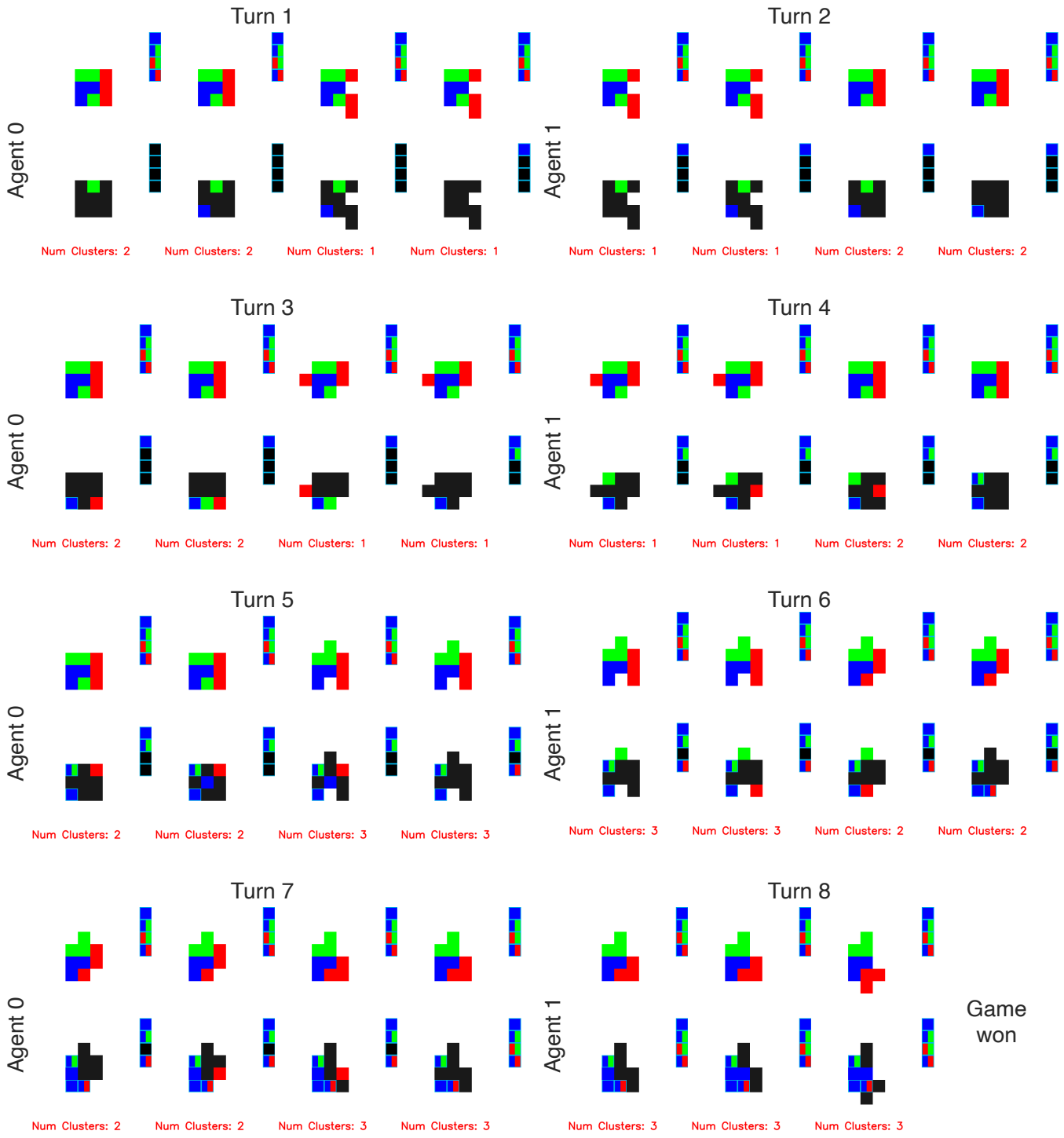


Figure 21: We show an example game in which `ToMTrack` **wins** in self-play. For each turn, we display the ground truth information at the top and the current observable information – essentially the agents' observation – at the bottom. We also show how many card clusters exist at this point per turn. Hint cards are highlighted by a light-blue border. In turn 7, the agent finishes the final cluster of cards by correctly inferring that the card under the *rb*-hint must be red and not blue. In turn 8, agent 1 should have finished early for more reward, but did not recognise that fact and instead used the last hint card, thus finishing the game. Note that the agents tend to play quite conservatively. In turns 1 and 3, for instance, it gains knowledge of two separate green cards but only groups them in turn 5 after agent 1 places a corresponding *gb*-hint in turn 4.

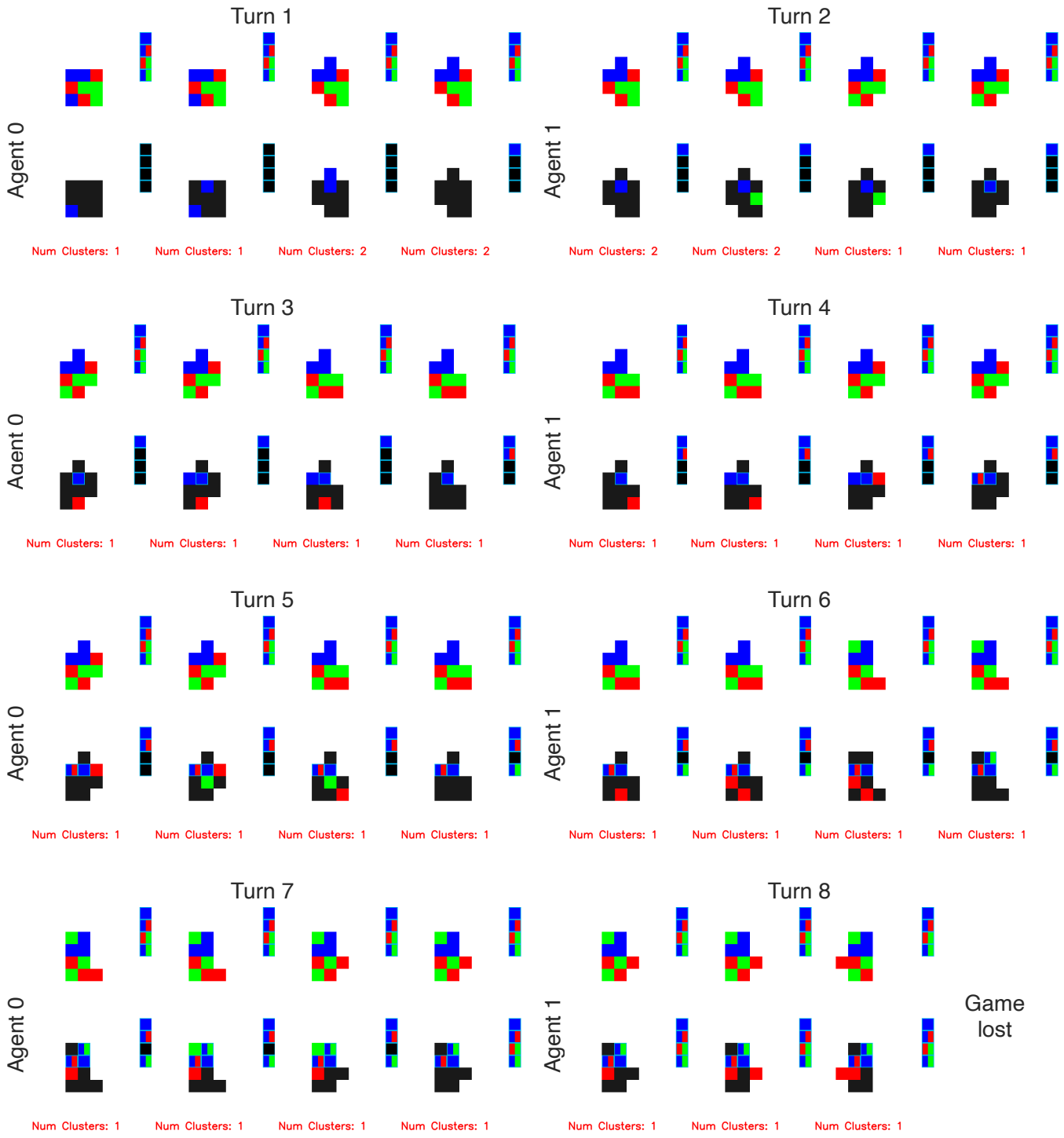


Figure 22: We show an example game in which `ToMTrack` **loses** in self-play. For each turn, we display the ground truth information at the top and the current observable information – essentially the agents' observation – at the bottom. We also show how many card clusters exist at this point per turn. Hint cards are highlighted by a light-blue border. Note that agent 0 in turn 1 finds a blue group early and manages to form a cluster quickly. Agent 0 confirms this by turn 3. In turn 6, the game is still winnable. However, because both agents over-focused on establishing a common ground around the blue cluster, they fail to realise this and in turn 7, agent 0 then blunders and misplaces the green card. From there, the game is lost under all circumstances. Additionally, note that in turns 4 and 6, agent 1 establishes the identity of all red cards but fails to recall that in turn 6, therefore also misplaying.