Partner Modelling Emerges in Recurrent Agents (But Only When It Matters)

Ruaridh Mon-Williams[1]* **Max Taylor-Davies**[1]* **Elizabeth Mieczkowski**[2] **Natalia Vélez**[2]
Neil R. Bramley[1] **Yanwei Wang**[3]† **Thomas L. Griffiths**[2]† **Christopher G. Lucas**[1]†
[1]University of Edinburgh [2]Princeton University [3]Massachusetts Institute of Technology

Abstract

Humans are remarkably adept at collaboration, able to infer the strengths and weaknesses of new partners in order to work successfully towards shared goals. To build AI systems with this capability, we must first understand its building blocks: does such flexibility require explicit, dedicated mechanisms for modelling others—or can it emerge spontaneously from the pressures of open-ended cooperative interaction? To investigate this question, we train simple model-free RNN agents to collaborate with a population of diverse partners. Using the 'Overcooked-AI' environment, we collect data from thousands of collaborative teams, and analyse agents' internal hidden states. Despite a lack of additional architectural features, inductive biases, or auxiliary objectives, the agents nevertheless develop structured internal representations of their partners' task abilities, enabling rapid adaptation and generalisation to novel collaborators. We investigated these internal models through probing techniques, and large-scale behavioural analysis. Notably, we find that structured partner modelling emerges when agents can influence partner behaviour by controlling task allocation. Our results show that partner modelling can arise spontaneously in model-free agents—but only under environmental conditions that impose the right kind of social pressure.

1 Introduction

While humans are certainly impressive 'solo' learners and problem-solvers, our capacity for cooperation and collaboration is even more remarkable—enabling us to achieve goals beyond the reach of any single individual, and leverage the complementary abilities of others while sharing or mitigating the costs of action. In particular, humans display exceptional *flexibility* in adapting to unfamiliar partners and task contexts. Indeed, it could be argued that this capacity for flexible collaboration is one of the primary contributors to the success of our species—without it, it is hard to see how our ancestors could have developed the culture and civilisation that persist to this day [1, 2, 3]. As we develop artificial agents that will operate alongside us, occupying our homes and workplaces, it is crucial that they too can share in this collaborative process [4].

One explanation for the powerful flexibility of human collaboration lies in our well-developed 'Theory of Mind' (ToM)—our general faculty for inferring and representing the latent mental properties (such as goals, beliefs, desires or intentions) that drive others' behaviour [5]. As with many other social contexts, effective collaboration with diverse partners requires an individual to not only react to the actions of others, but also attempt to *predict*, and where appropriate to *influence* them—processes which rely on the formation of predictive representations ('mental models') of other agents' decision-making [6, 7, 8]. In collaborative contexts, we can use our ToM to infer and represent the varying

*Equal contribution. Correspondence to `ruaridh.mw@ed.ac.uk` or `m.taylor-davies@sms.ed.ac.uk`
†Equal senior authorship

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

strengths and weaknesses of different partners [9, 10, 11]. For example, imagine being assigned to work with unfamiliar classmates on a group project for a machine learning class. Effectively dividing up the different tasks will require representations of each contributor's abilities: who will be best suited to implement the code, write the report, and deliver the presentation.

In this paper, we examine whether artificial agents, when trained to collaborate with different partners but without explicit mechanisms for agent modelling, spontaneously develop internal representations of their partners' abilities. We investigate this question in a fully cooperative setting, where agents optimise a shared goal (i.e., a single reward function), but have no prior knowledge of each other's attributes or action policies. Crucially, we train reinforcement learning agents with generic recurrent architectures and only task reward supervision—there are no auxiliary objectives or architectural priors pushing agents to model one another. This stands in contrast to prior work, such as Rabinowitz et al.'s 'Machine Theory of Mind' framework, which relies on specialised components optimised explicitly to infer other agents' internal states [12]. We find that despite these minimal inductive biases, agents develop structured, internal representations that **(i)** encode the different competencies of their partners; **(ii)** generalise to previously unseen collaborators; and **(iii)** emerge selectively, depending on agents' ability to control task allocation. Together, our findings suggest that partner modelling can arise within artificial agents solely from the demands of flexible cooperation, *without* explicit incentives or specialised architectures.

2 Related work

2.1 Ad hoc teamwork

The field of ad hoc teamwork (AHT) deals with the problem of developing agents that learn to collaborate 'on the fly' with previously unseen 'teammates', without any prior coordination [13]. AHT shares some basic elements with the field of multi-agent reinforcement learning (MARL); but where MARL typically assumes control of all agents in the environment, in AHT we control only a single agent (often called the 'AHT agent' or 'learner'; we will use 'ego agent' throughout), with teammates' actions governed by either simple heuristics or pre-trained (frozen) RL policies. AHT also considers only settings where all agents share a common cooperative objective—while individual agents might have additional goals or small differences in reward function, they are never in conflict with one another. The focus of research in AHT has mainly been on producing agents that can adapt to the varying 'play styles' (policies) of different partners or human collaborators [14]. In contrast to self-play training (where agents co-adapt to each other), ad hoc agents must generalise to novel partners under zero shot (no prior interaction) or few shot (minimal adaptation rounds) conditions. AHT can thus be viewed in large part as the problem of rapidly inferring a novel teammate's underlying parameters or characteristics. Accordingly, many approaches have involved explicit inference and representation of these characteristics; traditionally via forms of Bayesian belief-updating over a discrete teammate space [15, 16, 17, 18], or more recently using neural-network-based encoders to learn latent representations of teammate policies [12, 19, 20]. In contrast, our work uses the AHT setting to study *implicit*, emergent partner modelling in simple RNN agents without additional architectural components or auxiliary objectives.

2.2 Agent modelling

While many AHT approaches leverage some method for representing different teammates, the problem of modelling other agents in a shared environment is not unqiue to the AHT setting [21]. Recent work on agent modelling has primarily employed deep neural network-based approaches, where a dedicated module is optimised explicitly to produce useful representations of other agents' properties via some auxiliary objective. These representations can then be used to condition the controlled agent's own action policy [21, 22], allowing them to adapt their behaviour directly to the properties of the different agents with whom they must interact. For example, He et al. [23] extend the DQN architecture with an additional network that produces representations of the opponent policy. In contrast, Raileanu et al. [24] avoid having to maintain a separate model of other agents by using the learner's own current policy to infer others' goals via maximum likelihood. Numerous other works have employed some form of encoder-decoder architecture, typically trained via a reconstruction loss to learn latent embeddings that facilitate behaviour prediction [25, 12, 19, 26, 27, 28].

Beyond these explicit modelling approaches, a parallel line of work has examined social influence and coordination in multi-agent RL. For example, Jaques et al. [29] show that giving agents intrinsic motivation to shape others' behaviour improves cooperative outcomes, while work on zero-shot coordination demonstrates that agents trained with diverse partners can adapt to unseen teammates without additional training [30, 31]. These studies highlight the general idea that social prediction and adaptation are key to successful collaboration.

2.3 Emergent representations in model-free RL

In contrast to the explicit approach common across most agent modelling research, a different line of work has explored the *implicit* representations that emerge spontaneously in model-free RL agents trained only to achieve a particular high-level task. For example, multiple works have investigated the representations that develop within RL agents trained on simple navigation tasks, finding for example that agents encode target distance, reachability, and progress from their starting location [32], and that structured 'mental maps' of the environment emerge in the memories of 'blind' RNN agents [33]. Other research has explored the information encoded by model-free puzzle-solving or game-playing agents, isolating goal representations in maze-solving networks [34], humanlike chess concepts in AlphaZero [35], and planning-like abilities in RNN agents trained to play sokoban [36].

Our work is also closely connected to the meta-learning literature. Recurrent policies trained across many tasks or partners can act as implicit meta-learners, encoding past experience in hidden states to support rapid adaptation [37]. This view frames recurrence as a way for agents to learn about collaborators, not just task features. Our findings extend this perspective by showing that such implicit meta-learning can give rise to partner-specific internal models under the right collaborative pressures.

2.4 Cognitive and economic perspectives

Work in cognitive science, anthropology, and economics converges on the idea that prediction underpins intelligence and collaboration. Clark [38] argues that perception and action are driven by hierarchical prediction. Byrne and Whiten's 'Machiavellian Intelligence' hypothesis [**?**] proposes that human intelligence evolved to anticipate others' behaviour. Harsanyi [39] formalised this idea in economics through Bayesian games, where agents reason about hidden partner traits. Grosz and Kraus [40] further emphasise the need for shared predictive structures to coordinate group plans. Our results align with these perspectives, showing that predictive partner models can arise spontaneously in simple recurrent agents under collaborative pressure.

2.5 Attention

Recent work uses attention-based architectures to study social reasoning. Long et al. [41] analyse collaboration via attention weights, while Decision Transformer [42] frames RL as sequence modelling. In contrast, we use a simple RNN to show that structured partner representations can emerge without strong inductive biases. Future work could apply attention mechanisms to probe whether agents implicitly track partner positions or trajectories.

3 Problem Formulation

As is standard in the AHT literature, we formulate the problem as a two-agent partially observable Markov decision process (POMDP). This is defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}_1, \mathcal{O}_2, \mathcal{A}_1, \mathcal{A}_2, P, r, \gamma \rangle$, where $\mathcal{O}_i$ and $\mathcal{A}_i$ denote the observation and action spaces of agent $i \in \{1, 2\}$ (with $\vec{\mathcal{O}} = \mathcal{O}_1 \times \mathcal{O}_2$, $\vec{\mathcal{A}} = \mathcal{A}_1 \times \mathcal{A}_2$), $\mathcal{S}$ is the environment state space, $P : \mathcal{S} \times \vec{\mathcal{A}} \mapsto \Delta(\mathcal{S})$ denotes the state transition function, $r : \mathcal{S} \times \vec{\mathcal{A}} \mapsto \mathbb{R}$ is the shared reward function, and γ is the discount factor. At each timestep t the ego agent (agent 1) receives an snapshot of the environment $o_t^1 \in \mathcal{O}_1$—information from which may be retained in future timesteps as part of internal memory state h_t with a recurrent function $h_t = f(h_{t-1}, o_t^1)$. The ego agent acts according to its learned policy $\pi(a_t^1 \mid h_t)$ and the partner (agent 2) acts according to its (fixed) pre-trained policy π_2, governed by latent parameters $z^* \in \mathcal{Z}$, sampled from distribution $p(z^*)$ at the beginning of each episode. These traits, such as how quickly a given partner can perform each task, are not directly observable to the ego agent—rather, as in

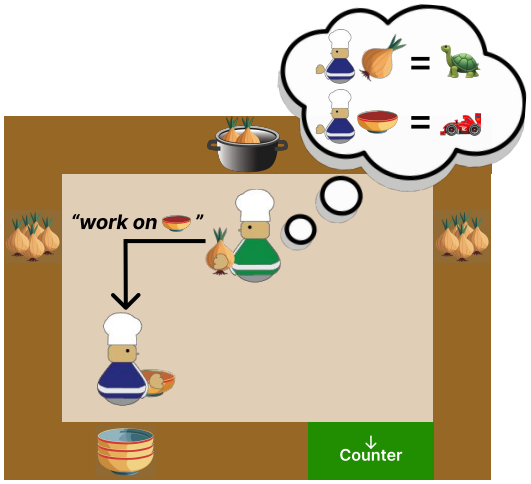

Figure 1: An illustration of our task setting, based on the 'cramped room' layout of Overcooked-AI. The ego agent (green) has a learned internal representation of how competent (fast) their partner (blue) is at each of the two subtasks; it uses this representation to determine which subtask the partner should work on.

the context of human theory of mind, they must be inferred (explicitly or implicitly) from observed behaviour.

The ego agent is trained *only* to maximise the expected cumulative reward across episodes, $\max_\pi \ \mathbb{E}_{z^* \sim p(z^*)} \ \mathbb{E}_{\tau \sim \pi, \pi_2(z^*)} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$, where τ denotes a trajectory of states, observations, and actions sampled from the ego policy π and partner policy $\pi_2(z^*)$ under the transition dynamics P. Each episode involves a different partner sampled from a distribution over latent traits $z^* \in \mathcal{Z}$. Importantly, no explicit architectural mechanisms or auxiliary objectives encourage modelling of these latent traits. We are interested in whether, under these minimal conditions, internal partner models nevertheless emerge *implicitly* within the ego agent's recurrent state h_t.

4 Methods

4.1 Environment

Critical to any test of our hypothesis is that the environment imposes *collaborative pressure*; i.e. the ego agent's optimal policy depends on the latent characteristics of their partner, and so modelling those characteristics is conducive to achieving high joint reward. We provide this through Overcooked-AI [43], a fully cooperative environment where agents work together to prepare soups, tasked with maximising the throughput $r = \frac{\Delta \text{Soup}}{\Delta \text{Time}}$ (for additional results in a second cooperative environment, see Appendix A). Each agent must navigate a shared kitchen to gather ingredients, cook them, and serve the completed soups — making success heavily dependent on coordination and division of labour. The environment difficulty can be modified via different recipes, which vary in complexity and number of ingredients, and different kitchen layouts, which introduce specific constraints (e.g. encouraging agents to pass items to perform the task successfully, or forcing agents to navigate around each other in cramped spaces). Figure 1 illustrates one such layout.

4.2 Agent Architecture

The ego agent's policy is implemented as a gated recurrent unit (GRU) [44] recurrent neural network (RNN). At each timestep t, the agent processes an observation o_t and updates its hidden state via $h_t = GRU(o_t, h_{t-1})$. The hidden state h_t acts as a general dynamic memory, evolving as the interaction unfolds. The ego policy is trained using Proximal Policy Optimisation (PPO) [45], implemented in JAX [46] to facilitate efficient parallel training.

4.3 Experimental Design

Achieving high reward in the Overcooked-AI environment requires agents to successfully coordinate two different subtasks: preparing ingredients ('task 1'), and serving soup ('task 2'). We train our ego agent alongside a distribution of partners who vary in how competent they are at the two subtasks (operationalised as how frequently they can take actions towards each task). To introduce a direct connection between partner properties and optimal ego agent behaviour, we grant the ego agent control over which subtask its partner is working towards at any given time. Our hypothesis is that effective task allocation will require the ego agent to learn how to recognise and represent the abilities of different partners—that is, after training, the RNN hidden state dynamics should be optimised to encode how fast a given partner is at tasks 1 and 2 (as illustrated in Figure 1).

Within this framework, we carry out three experiments to probe different dimensions of partner modelling:

4.3.1 Experiment 1: Does the pressure to allocate subtasks drive the emergence of partner modelling?

This experiments tests whether placing the responsibility for subtask allocation to the ego agent encourages it to develop internal representations of its partner's capabilities. In particular, we ask whether the ego agent can learn to reliably allocate tasks effectively by representing different partners' ability profiles? To test this, we generate a distribution of partners, each characterised by a two-dimensional vector $\mathbf{v} = [v_1, v_2]$ denoting the cooldown interval (i.e, the number of timesteps between consecutive actions) for each subtask. These cooldowns can be interpreted as inverse proxies for subtask skill: a lower v_i reflects higher proficiency in subtask i, allowing the partner to act more frequently. The ego agent is trained alongside a population of partners with v_i sampled independently from the set $\{1, 2, 3, 4, 7, 9\}$. During evaluation, the ego agent is paired with partners whose cooldowns are sampled from a different set, $\{0, 2, 3, 8, 10\}$. While some individual cooldown values overlap (e.g. 2 and 3), the *pairs* of cooldowns (v_1, v_2) used for training and evaluation are disjoint, ensuring that no test partner configuration was encountered during training. The ego agent has a constant cooldown of 2 for both tasks, allowing for a balance between dominating the tasks (if too fast) and slowing learning (if too slow), and is equipped with an additional action that allows it to dictate which subtask the partner contributes to at each timestep.

4.3.2 Experiment 2: Can agents adapt online to new partners within an episode?

Successful collaboration in the real world often requires us to deal with sudden, unexpected changes. In this experiment, we examine whether our ego agent can adapt dynamically to different partners *within* a single episode. During each training episode, there is a 50% probability that the partner is 'switched' at a random timestep between 30 and 70% of the 600-timestep duration (the random dynamics ensure that the agent cannot memorise a fixed timing pattern). To ensure non-triviality, the post-switch partner is always sampled with a *mirrored* ability profile (i.e. if the initial partner is faster at task 1, the new partner is faster at task 2, and vice versa). When evaluating the agent after training, we simplify the analysis by performing the switch in every episode, always at exactly $t = 300$, and only in a single direction (faster at task 1 $\rightarrow$ faster at task 2).

4.3.3 Experiment 3: Can blind agents develop partner models from task reward alone?

Inspired by the findings of Wijmans et al. [33], who showed that 'blind' agents trained for PointGoal navigation develop internal map-like representations of their environment despite having access only to proprioceptive feedback, we ask a related question—can blind versions of our Overcooked agents, trained purely to maximise cooperative task reward, still develop internal models of their partners' capabilities?

To investigate this, we remove all visual input from our ego agents and restrict them to receiving only egocentric signals (information only about their current grid cell, including their location, orientation, and any objects they are holding). Importantly, they are unable to perceive their partner's location, or directly observe their behaviour.

To generate variation in partner competence, we first train a high-performance onion-preparing agent in self-play, and then inject controlled levels of noise into its policy. This produces a range of partner

behaviours from reliably competent to frequently erratic. The ego agent must infer where on this spectrum each partner lies purely from its own direct experiences and task rewards. Full details of the partner generation and evaluation procedures are provided in the Appendix.

4.4 Evaluation Overview

To investigate how and when internal partner models emerge, we analyse agent behaviour across five Overcooked-AI layouts (see Appendix). Our evaluation is motivated by three key questions: (i) Can the ego agent collaborate effectively with previously unseen partners? (ii) Do hidden states encode partner traits such as speed or competence (iii) Does the agent update its internal model in response to mid-episode changes in partner attributes? To address these questions, we analyse overall reward (total number of soups delivered), adaptation curves (how quickly reward accumulates over time), linear probe accuracy (how well partner traits can be 'decoded' from RNN hidden states by optimising a single linear layer with input size $\dim(h)$ and output size 1) and UMAP projections [47] (capturing the structure of hidden states). We also compare against three baselines, each designed to isolate a different factor affecting partner modelling: a *feedforward MLP*, which lacks memory; a *single-partner RNN*, trained on a fixed partner to assess the role of training diversity; and a *non-influential RNN*, trained across the full distribution of partners but without the ability to influence their behaviour. A more detailed description of each baseline is provided in the Appendix.

5 Results

5.1 Collaborative agents adapt to unseen partners

To establish the importance of modelling different partners to our chosen environment, we compare the performance of our RNN ego agent policies against two simpler baselines: a purely feed-forward MLP policy, and an RNN policy trained alongside only a single constant partner. Figure 2 shows the results of this comparison across five different layouts. We find that the RNN policy trained against diverse partners outperforms both the MLP policy and the single-partner-trained RNN variant; presumably by virtue of a learned ability to model partner parameters. The single exception to this is *fivebyfive_v1*, in which the MLP agent achieved the highest reward—possibly due to the fact that *fivebyfive_v1* is a simpler layout, requiring less spatial coordination and allowing simpler behavioural strategies to perform well. Importantly, the higher performance of the multi-partner-RNN with respect to the single-partner RNN does not reflect simple memorisation of different partners, since the partners encountered in evaluation were unseen during training. In addition to superior performance, we find that evaluation episodes with the (multi-partner-trained) RNN ego agent yield a higher correlation between which task the partner spends most time on, and which task they're fastest at (Figure 2C)—suggesting that the ego agent's task allocation decisions are informed by a representation of different partners' ability profiles.

5.2 Agent memory encodes partners' abilities (when there is pressure to do so)

A key idea behind our experimental design is that having the ability to influence which subtask a partner performs will pressure the ego agent to learn hidden representations that encode the task speeds of different partners. To investigate the hypothesis further, we trained RNN policies under three different conditions: our 'full' setup including both partner diversity and task influence, plus the 'single-partner' and 'non-influential' variants described in Section 4.4. For each condition, we then extracted the trained agent's hidden states during evaluation episodes alongisde 46 different unique partners in 5 different layouts (with 20 seeds per condition/layout/partner).

Figure 3A shows, for each condition and layout, a 2D UMAP projection of these hidden states (averaged over the final 50 timesteps of each rollout), coloured by the difference in task speeds for each partner (task 1 speed $-$ task 2 speed). We can see that for the multi-partner-trained RNN, there is a high degree of structure, with less for the single-partner-trained variant, and less still for the non-influential variant. For each condition and layout, we also trained single-layer linear probes at different values of t to predict partner task speeds from hidden states averaged over $(0, t]$. Figure 3B shows the accuracy of these probes on a test set of held-out partners: we see that, for the initial hidden states at $t = 0$, our linear probes perform no better than a baseline trained on random data. As the episodes progress and the ego agent is able to interact with and observe each partner, probe

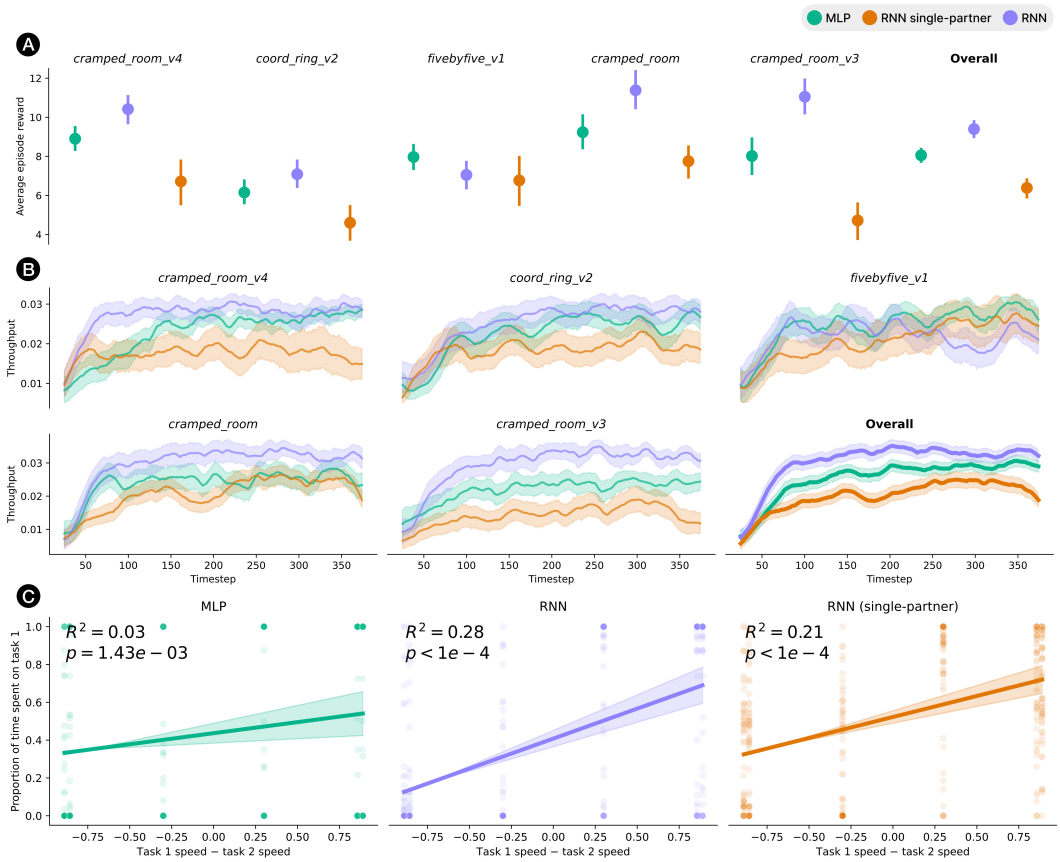

Figure 2: Comparison between different ego agent policies in the overcooked environment. Each policy was evaluated in five different layouts against 8 different combinations of partner speed parameters, over 10 seeds. **(A)** average episode reward, per-layout and overall **(B)** throughput (rate of soup production), per-layout and overall, with shaded areas giving bootstrapped 95% confidence intervals **(C)** correlation between how much faster the partner agent was at task 1 vs task 2 and the proportion of time the partner spent performing task 1 (over all layouts). Shaded areas show 95% confidence intervals over the slope; a higher correlation indicates more efficient task allocation.

accuracy increases across all three conditions—but is highest for the multi-partner-trained RNN, and significantly impaired for the non-influential RNN.

These results offer convincing evidence, first of all, that our ego agent has learned to encode meaningful information about different partners in memory, *without being explicitly trained to do so*. They also demonstrate the importance of environmental pressure to the emergence of these representations. In particular, when the ego agent is stripped of its ability to influence partners' behaviour directly, its hidden states contain significantly less information about partner task abilities (as measured by linear decodeability)—strikingly, even less than those of the ego agent that only ever encountered a single partner during training! For a replication of these results in a second environment, see Appendix A.

5.3 Agents can adapt online to new partners

So far, we have established that our RNN ego agents, once trained, can adapt to different partners across different episodes. A stronger version of flexible cooperation involves adapting to new partners 'online', without the environment or internal agent state being reset. To test this, we train an ego agent alongside partners whose task speeds may change up to once per episode. Across three layouts, we then evaluate this agent over a number of 600-timestep rollouts where they are paired initially with a

Figure 3: A comparison of the hidden states of RNN ego agents trained under different conditions. **(A)** UMAP embeddings of RNN hidden states averaged over the final 50 timesteps of each episode, coloured by the partner's difference in task speeds (speed 1 − speed 2), for five different layouts of the Overcooked environment. **(B)** Mean test accuracy of linear probes trained to recover partner speeds from sets of RNN hidden states accumulated up to different timesteps (with shaded areas giving bootstraped 95% confidence intervals).

partner that is fast at task 1 and slow at task 2, then switched at $t = 300$ to a partner with the opposite profile.

To measure how well our agent copes with this scenario, we track the average soup throughput over time. From the results in Figure 2B, we expect that the throughput should initially increase to a steady state; we anticipate that it will then drop sharply at $t = 300$ as the partner's task speeds are reversed. After this point, if the ego agent is capable of adapting online to the new partner, the throughput should increase once more to a new steady state; if not, it should remain low. Figure 4A shows that for all three tested layouts, the throughput does indeed increase again after $t = 300$, demonstrating the presence of online adaptation [3]. As further evidence, Figure 4B shows that, on average, the partner is directed to allocate their time mostly to task 1 for $t < 300$ and mostly to task 2 for $t > 300$. Finally, we also visualise in Figure 4C how the ego agent's hidden states are affected by the switch. UMAP projections of hidden states averaged over $t < 300$ align roughly with the distribution of embeddings for 'baseline' episodes with (constant) partners fast at task 1 and slow at task 2; over

[3]We note that the throughput does not fully recover post-switch due to an asymmetry in the subtask complexities: because task 1 requires more steps than task 2 to execute, the overall team efficiency is higher when the partner is fast at task 1 and slow at task 2 than the reverse.

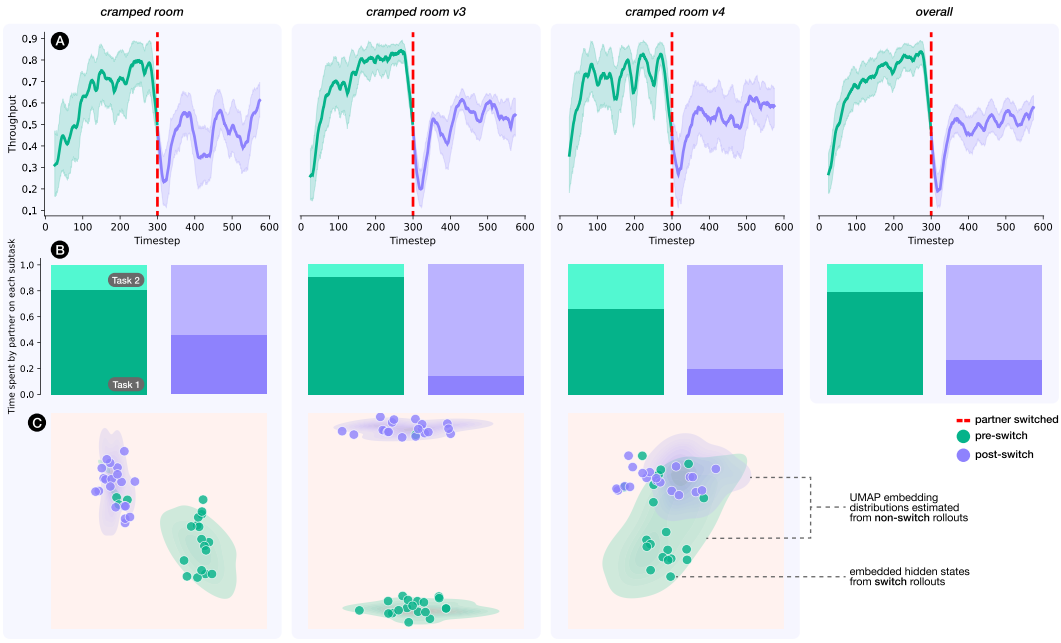

Figure 4: A demonstration of online adaptation. **(A)** Average throughput (rate of soup delivery) during episodes where the partner is switched halfway through from being faster at task 1 to faster at task 2. **(B)** From the same episodes, the average proportion of time spent by the partner performing each subtask before and after the switch. **(C)** UMAP embeddings of the average pre-switch and post-switch RNN hidden states from each episode. Also shown are the distributions (approximated via KDE) for embeddings of hidden states from *non-switch* baseline episodes with partners matching the pre- and post-switch speeds respectively.

$t > 300$, they move to match the distribution for episodes with partners slow at task 1 and fast at task 2 (see supplementary videos for visual examples of this adaptation).

5.4 Partner modelling emerges even in blind AI agents

We find that structured partner modelling also emerges in blind agents – trained without any architectural biases toward modelling their partner. Despite relying purely on egocentric observations and scalar task reward, these agents generalise to new partners and outperform both a recurrent agent trained on a single partner and a memoryless MLP baseline. Evaluated across five Overcooked layouts (with six novel partners per layout and 10 random seeds per permutation), the blind RNN agents trained with collaborators with a diverse range of competencies achieved an average throughput of 9.43 soups per episode. This compares to just 5.8 for the single-partner RNN and 1.08 for the MLP. These results show that the interaction structure and memory can enable adaptive behaviour to the capabilities of partner agents – even under harsh observational conditions. Further details, including corresponding videos and a breakdown of the results, are included in the supplementary materials.

6 Discussion

In this paper, we have studied the question of whether representations of other agents' relevant attributes can emerge simply as a result of environmental pressure to collaborate effectively with diverse partners. In the absence of dedicated architectural features or auxiliary objectives, we found that RNN agents trained to play a version of the cooperative game 'Overcooked' nevertheless developed structured internal representations of their partners' task abilities. In an additional experiment, we showed that these representations enable agents to adapt online to new partners within the same episode. Finally, we also demonstrated that the development of structured representations is significantly weakened when agents are denied the ability to influence partner behaviour. Taken as a whole, our results serve to illustrate the idea that social intelligence can emerge from specific environmental

pressures acting in concert with general mechanisms for learning and memory, rather than necessarily relying on unique architectures. We believe that this is important to bear in mind as we seek to develop artificial agents that bridge the gap towards humanlike social cognition and behaviour.

It is notable that structured partner modelling also emerged in blind agents trained without visual input, relying solely on egocentric signals and task reward. Despite having no explicit access to their partner's actions or state, these "blind" agents developed internal representations that enabled them to effectively collaborate with partners displaying a diverse range of competencies. This emerged despite minimal inductive biases and limited observational input. This indicates that the structure of the interaction itself is sufficient to drive the emergence of partner-aware policies.

While we feel our work represents a valuable contribution to the study of emergent partner modelling, we highlight various limitations that might serve as starting points for future research. First and foremost, our experiments used only a single cooperative environment (Overcooked-AI)—while we are confident that our results will generalise to other environments and task settings, an obvious target for future work is to confirm this empirically. Of particular interest would be *open-ended* environments that allow us to study how partner representations evolve over time in a more continuous setting; or those with more complex or overlapping subtasks. Relatedly, future work might study whether our findings scale to more than one teammate—we believe that they should, provided that the environment imposes sufficient pressure (i.e. all teammates' behaviour is relevant to task completion). That said, inference will become more demanding with additional agents, and so the ego agent may learn to rely on shortcuts such as modelling the average behaviour of its teammates.

A further limitation is that we have restricted ourselves to studying relatively simple forms of representation. In the real world, people engage in highly complex modelling of their social partners, including through hierarchical representations that deal with how others are perceiving them in turn. It would be interesting to investigate whether these simple, general agent architectures are capable of acquiring such sophisticated capabilities, and under what environmental conditions. Related to this is the fact that our implementation of 'influence' was very strong, essentially taking the form of direct control. Humans typically influence their collaborators in much more nuanced ways—future work might reduce this gap via some form of communication system, where the partner learns to follow (or ignore) high-level instructions from the ego agent.

Aside from these limitations, a further avenue is to explore transformer-based agents, where attention may offer a natural probe of whether agents implicitly track their partners' positions or strategies. Another direction is to test whether internal partner representations can be transferred between agents and tasks via hidden state initialisation. For example, one could evaluate whether seeding an agent's memory with the final hidden state from a previous rollout improves adaptation when encountering the same partner; probing the portability and generality of the learned representations. Comparing 'transplanted' and 'cold-start' agents would provide insight into the extent to which internal partner models support efficient reuse and generalisation. Another possible avenue would involve direct comparisons between the implicit partner modelling approach we study here and various explicit methods: we expect that the latter would perform better in the specific modelling contexts they were trained for, at the cost of reduced flexibility to changes in task environment or partner attribute space.

Finally, while our primary motivation is the desire for artificial agents capable of collaborating flexibly with humans, we believe that a version of our approach might also be used to shed light on the evolutionary differences in collaborative and cooperative behaviour observed across different animal species. To this end, future work might explore using large-scale evolutionary simulations to further study the interplay of environmental pressures and agent architectures; an approach which has recently proven fruitful for investigating other social behaviours such as altruism [48].

7 Acknowledgements

This work was supported by the EPSRC CDT in RAS (EP/L016834/1) and the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program awarded to E.M. We thank J. Shah and many others for their invaluable support and expertise.

References

[1] Michael Tomasello, Alicia P Melis, Claudio Tennie, Emily Wyman, and Esther Herrmann. Two key steps in the evolution of human cooperation: The interdependence hypothesis. *Current Anthropology*, 53(6):673–692, 2012.

[2] Sarah Blaffer Hrdy. *Mothers and Others: The Evolutionary Origins of Mutual Understanding*. Harvard University Press, Cambridge, MA, 2009.

[3] Joseph Henrich. *The Secret of Our Success: How Culture Is Driving Human Evolution, Domesticating Our Species, and Making Us Smarter*. Princeton University Press, 2018.

[4] Ruaridh Mon-Williams, Gen Li, Ran Long, Wenqian Du, and Christopher G Lucas. Embodied large language models enable robots to complete complex tasks in unpredictable environments. *Nature Machine Intelligence*, pages 1–10, 2025.

[5] Ian A Apperly and Stephen A Butterfill. Do humans have two systems to track beliefs and belief-like states? *Psychological review*, 116(4):953, 2009.

[6] 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.

[7] Rebecca Saxe and Nancy Kanwisher. People thinking about thinking people: the role of the temporo-parietal junction in "theory of mind". In *Social neuroscience*, pages 171–182. Psychology Press, 2013.

[8] Mark K Ho, Rebecca Saxe, and Fiery Cushman. Planning with theory of mind. *Trends in Cognitive Sciences*, 26(11):959–971, 2022.

[9] Yang Xiang, Natalia Vélez, and Samuel J Gershman. Collaborative decision making is grounded in representations of other people's competence and effort. *Journal of Experimental Psychology: General*, 152(6):1565, 2023.

[10] Yang Xiang, Natalia Vélez, and Samuel J Gershman. Optimizing competence in the service of collaboration. *Cognitive Psychology*, 150:101653, 2024.

[11] Carolyn Baer and Darko Odic. Mini managers: Children strategically divide cognitive labor among collaborators, but with a self-serving bias. *Child Development*, 93(2):437–450, 2022.

[12] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, S. M. Ali Eslami, and Matthew Botvinick. Machine theory of mind. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4218–4227. PMLR, 10–15 Jul 2018.

[13] Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V Albrecht. A survey of ad hoc teamwork research. In *European conference on multi-agent systems*, pages 275–293. Springer, 2022.

[14] Yancheng Liang, Daphne Chen, Abhishek Gupta, Simon S Du, and Natasha Jaques. Learning to cooperate with humans using generative agents. *arXiv preprint arXiv:2411.13934*, 2024.

[15] Piotr J. Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *J. Artif. Int. Res.*, 24(1):49–79, July 2005.

[16] Samuel Barrett, Peter Stone, and Sarit Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '11, page 567–574. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[17] Stefano V. Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '13, page 1155–1156, 2013.

[18] Stefano V. Albrecht, Jacob W. Crandall, and Subramanian Ramamoorthy. Belief and truth in hypothesised behaviours. *Artificial Intelligence*, 235:63–94, 2016.

[19] Georgios Papoudakis and Stefano V Albrecht. Variational autoencoders for opponent modeling in multi-agent systems. *arXiv preprint arXiv:2001.10829*, 2020.

[20] Arrasy Rahman, Ignacio Carlucho, Niklas HÃ¶pner, and Stefano V. Albrecht. A general learning framework for open ad hoc teamwork using graph-based policy learning. *Journal of Machine Learning Research*, 24(298):1–74, 2023.

[21] Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

[22] Ruaridh Mon-Williams, Theodoros Stouraitis, and Sethu Vijayakumar. A behavioural transformer for effective collaboration between a robot and a non-stationary human. In *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1150–1157. IEEE, 2023.

[23] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé, III. Opponent modeling in deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1804–1813. PMLR, 20–22 Jun 2016.

[24] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4257–4266. PMLR, 10–15 Jul 2018.

[25] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1802–1811. PMLR, 10–15 Jul 2018.

[26] Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann. Deep interactive bayesian reinforcement learning via meta-learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 1712–1714, 2021.

[27] Annie Xie, Dylan Losey, Ryan Tolsma, Chelsea Finn, and Dorsa Sadigh. Learning latent representations to influence multi-agent interaction. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 575–588. PMLR, 16–18 Nov 2021.

[28] Georgios Papoudakis, Filippos Christianos, and Stefano V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, 2021.

[29] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*, pages 3040–3049. PMLR, 2019.

[30] Adam Lerer, Hengyuan Hu, Jakob Foerster, and Noam Brown. Improving policies via search in cooperative partially observable games. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7187–7194, 2020.

[31] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

[32] Kshitij Dwivedi, Gemma Roig, Aniruddha Kembhavi, and Roozbeh Mottaghi. What do navigation agents learn about their environment? . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10266–10275, 2022.

[33] Erik Wijmans, Manolis Savva, Irfan Essa, Stefan Lee, Ari S. Morcos, and Dhruv Batra. Emergence of maps in the memories of blind navigation agents. In *The Eleventh International Conference on Learning Representations*, 2023.

[34] Ulisse Mini, Peli Grietzer, Mrinank Sharma, Austin Meek, Monte MacDiarmid, and Alexander Matt Turner. Understanding and controlling a maze-solving policy network, 2023.

[35] Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47):e2206625119, 2022.

[36] Arthur Guez, Mehdi Mirza, Karol Gregor, Rishabh Kabra, Sebastien Racaniere, Theophane Weber, David Raposo, Adam Santoro, Laurent Orseau, Tom Eccles, Greg Wayne, David Silver, and Timothy Lillicrap. An investigation of model-free planning. 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 2464–2473. PMLR, 09–15 Jun 2019.

[37] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[38] Andy Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and brain sciences*, 36(3):181–204, 2013.

[39] John C Harsanyi. Games with incomplete information played by "bayesian" players, i–iii part i. the basic model. *Management science*, 14(3):159–182, 1967.

[40] Barbara J Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[41] Qian Long, Ruoyan Li, Minglu Zhao, Tao Gao, and Demetri Terzopoulos. Inverse attention agents for multi-agent systems. *arXiv preprint arXiv:2410.21794*, 2024.

[42] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[43] Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Tjarko Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktäschel, Chris Lu, and Jakob Nicolaus Foerster. Jaxmarl: Multi-agent rl environments and algorithms in jax. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[44] 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.

[45] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[46] 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.

[47] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

[48] Max Taylor-Davies, Gautier Hamon, Timothé Boulet, and Clément Moulin-Frier. Emergent kin selection of altruistic feeding via non-episodic neuroevolution. In Pablo García-Sánchez, Emma Hart, and Sarah L. Thomson, editors, *Applications of Evolutionary Computation*, pages 496–509, 2025.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction accurately represent the paper's empirical contributions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Our Discussion section includes some limitations of the current work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: Our paper does not include any theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Detailed information for reproduceability will be provided in the (supplementary) technical appendices.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Links to the experiment and analysis code will be provided in the (supplementary) technical appendices.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: These details will be provided in the (supplementary) technical appendices.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All results figures include confidence intervals where appropriate.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: This information will be provided in the (supplementary) technical appendices.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We are not aware of any particular societal impacts of this work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not pose any such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited the authors of the Overcooked-AI environment, which is the only existing asset we use.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Our work does not include the release or introduction of any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not use any experiments with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not use any experiments with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We have not used LLMs for any core or non-standard component of our work.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix

A CoinGame results

To ensure that our findings generalise beyond the specific environment of Overcooked, we conducted some preliminary experiments using a modified version of the JaxMARL suite's CoinGame environment [43].

A.1 Environment

In our version of CoinGame, two agents (the ego agent and one partner) must work together to collect red and blue coins within a small gridworld (see Figure 5A). At any given timestep, the partner is in either 'red mode' (tries only to collect red coins while ignoring blue) or 'blue mode' (vice versa). Different partners are characterised by a two-dimensional 'skill profile' $[s_r, s_b]$, which controls their probability of successfully executing actions when in each mode (the ego agent has $s_r = s_b = 1$). As in our Overcooked experiments, the ego agent can exert influence by switching their partner's mode. The ego agent is rewarded based on the total number of coins collected ($n_{\text{red}} + n_{\text{blue}}$) over an episode; they are thus incentivised to find the most efficient 'division of labour' between themselves and their partner.

A.2 Experimental procedure

As in our main Experiment 1 (5.1) we trained ego agents for 1e7 timesteps under four conditions:

1. MLP (ego agent uses a simple MLP in place of an RNN)
2. No-influence RNN (ego agent has no control over which coin type the teammate pursues)
3. Single-partner RNN (ego agent only exposed to a single partner type during training)
4. Multi-partner RNN (ego agent paired with multiple partner types during training and has influence)

During training, the single-agent RNN was always paired with a teammate with skill profile $[0.2, 0.8]$; in all other cases partners were sampled uniformly from the set $\big\{[x, 1-x] \;\forall\; x \in \{0.2, 0.4, 0.6, 0.8\}\big\}$. During evaluation, partners were always sampled uniformly from the set $\big\{[x, 1-x] \;\forall\; x \in \{0.1, 0.3, 0.5, 0.7, 0.9\}\big\}$. The ego agent thus never encountered during evaluation a partner they had previously seen in training. From these evaluation episodes we recorded the total number of coins collected, the number of timesteps where the teammate was pursuing the 'correct' coin colour (based on their skill profile), and the hidden states of the ego agent RNN (where applicable). As with our experiments in Overcooked, the hidden states were analysed via UMAP projections and linear probe accuracy.

A.3 Results

Looking at Figure 5B, we see that the multi-partner-RNN ego agent outperforms the single-partner-RNN agent, which in turn outperforms the no-influence-RNN and the MLP agents—replicating the trend we observed in our Overcooked results. Also corroborating our previous results is Figure 6, which shows that hidden states extracted from the multi-partner-RNN are more structured with respect to partner skill profiles than those from the single-partner or no-influence variants. Finally, we computed the correlation to partner skill profile of *individual* RNN hidden unit activations. As Figure 7 shows, we found multiple units with strong positive or negative correlations (16/32 with $|\text{corr}| \geq 0.5$); as well as some units with close-to-zero correlation (likely encoding task/environment information unrelated to partner properties).

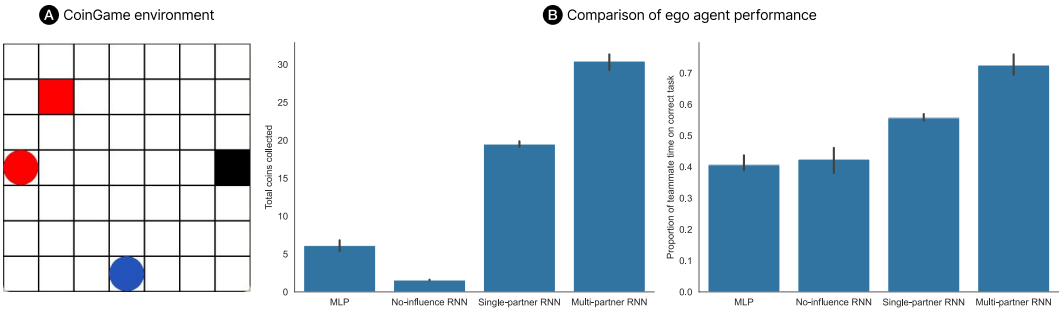

Figure 5: (A) An example CoinGame environment layout at episode start. The two coloured circles are coins, the red square is a teammate currently in 'collect red coins' mode and the black square represents the ego agent. (B) A comparison of the evaluation performance of different ego agents trained in the CoinGame environment, showing the mean total number of coins collected by ego agent and teammate (left) and the mean proportion of time spent by the teammate pursuing the 'correct' coin colour (i.e. the one matched to their highest skill level). Error bars give bootstrapped 95% confidence intervals over 5 random training seeds.

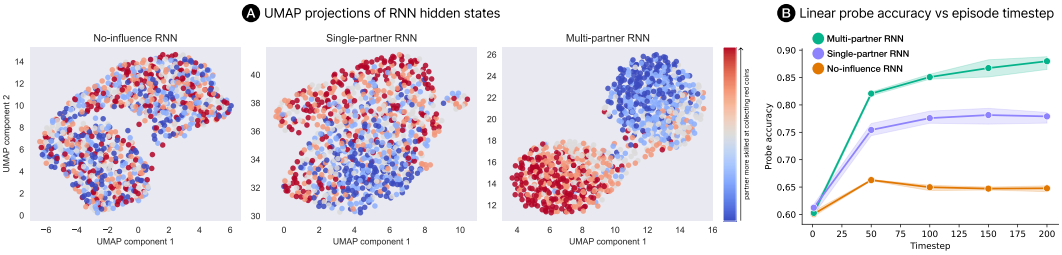

Figure 6: (A) UMAP embeddings of RNN hidden states averaged over the final 50 timesteps of each episode, coloured by partner skill profile. (B) Mean test accuracy of linear probes trained to recover partner skill from sets of RNN hidden states accumulated up to different timesteps (with shaded areas giving bootstraped 95% confidence intervals).

B Additional details

B.1 Training setup

B.1.1 Ego agent training

The ego policy is trained on a single GPU using Proximal Policy Optimisation (PPO), running synchronously across 256 parallel Overcooked-AI environment instances. Both agent and environment are implemented in JAX with `jax.jit` for accelerated gradient updates and rollout collection. Each rollout lasts 400 timesteps in Experiments 1 and 3, and 600 timesteps in Experiment 2, and agents are rewarded for every successful soup delivery. Training runs for 15 million timesteps against a distribution of partner agents. For the first 5 million timesteps, a decayed reward shaping is used to aid policy learning (rewarding the agent for putting onions in the pot and for cooking the soup when it contains the correct ingredients). For experiments 1 and 2, partner agents could at any given timestep perform one of two subtasks: (1) placing ingredients into a pot, or (2) serving soup (which involved picking up a bowl, ladling the soup, and delivering it). Each subtask is handled by a separate, pre-trained neural network that specialises in the specific behaviour. The RNN ego agent has an additional action that allows it to set the partner's current subtask. For experiment 3 (with the blind agent), the ego policy is trained against a distribution of partners who perform subtask 1 to varying competencies. We randomised the starting state of each episode during both training and testing. The RNN hyperparameters used for the experiments are shown in Table 1.

B.1.2 Partner agent training

Each partner policy consists of two feed-forward subtask networks trained independently in self-play using PPO with reward shaping. To obtain subtask-specific policies, each network was paired with a

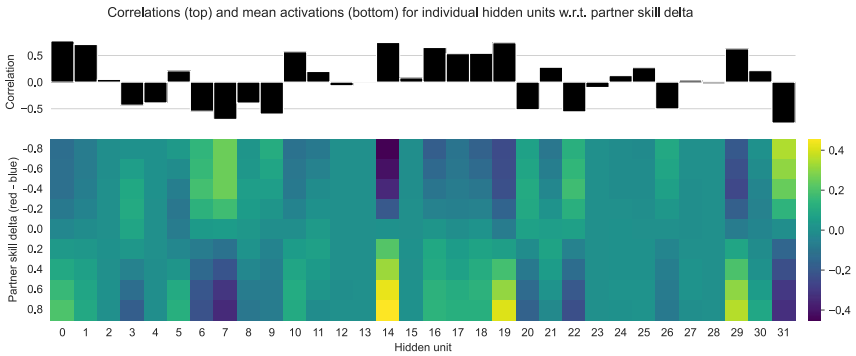

Figure 7: Top: correlation coefficients between individual hidden unit values and partner skill profile computed over 1000 evaluation episodes. Bottom: normalised average value of each hidden unit for each distinct partner.

Table 1: RNN training hyperparameters

Parameter	**Value**
FC_DIM_SIZE	128
GRU_HIDDEN_DIM	128
ACTIVATION	relu
LR	5e-4
ANNEAL_LR	True
LR_WARMUP	0.05
NUM_ENVS	256
NUM_STEPS	256
UPDATE_EPOCHS	4
NUM_MINIBATCHES	64
TOTAL_TIMESTEPS	1e7
CLIP_EPS	0.2
ENT_COEF	0.01
GAMMA	0.99
GAE_LAMBDA	0.95
SCALE_CLIP_EPS	False
VF_COEF	1.0
MAX_GRAD_NORM	0.25

partner that always performed the *other* subtask. To ensure robustness we introduced randomness during training: the subtask network had a 30% probability of waiting on any given step and a 30% probability that its partner would take a random action. This prevented over-fitting to specific interaction patterns or timings and ensured that the learned policies could operate independently. We found that this approach allowed us to produce policies for the partner agent faster than relying on manually specified rules or heuristics.

After training, these subtask networks were combined to form the full partner agents capable of performing both subtasks. To create a distribution of partner behaviours with varying speeds, we controlled the speed at which each subtask policy could be executed when interacting with the ego agent. For example, a partner agent might be able to execute the ingredient placement policy every four timesteps and the serving policy every two timesteps. This enables us to simulate a range of partners with different speeds and proficiency across the two tasks, while utilising the same robust underlying subtask policies.

For experiment 3, we created a distribution of partners with different competencies by adding a certain probability of the partner taking a random action at any given timestep (e.g, a competent partners might have a 5% chance of acting randomly; an incompetent partner might have a 95% chance).

B.2 Agent architectural design

The architecture includes both input and output fully connected (FC) layers. After the convolutional neural network (CNN) processes the observation, an FC layer maps the resulting embedding to the hidden size of the GRU. The GRU's output is then passed through separate two-layer multilayer perceptrons (MLPs) for the actor and critic heads, which generate action logits and a scalar value estimate, respectively. Observations are pre-processed to ensure each agent has a local, self-centred view of its environment. Furthermore, the CNN output is normalised before being fed into the GRU to stabilise training and improve performance.

B.3 Evaluation details

B.3.1 Throughput

The throughput (rate of soup production, used in Figures 2 and 4) is given by $\frac{\Delta \text{Soup}}{\Delta \text{Time}}$. To obtain a point estimate of the throughput at time t, we fit the slope of the cumulative reward curve over the sliding window $[t-25, t+25]$ by the method of least squares.

B.3.2 UMAP projections

To obtain 2D embeddings of RNN hidden states (as seen in Figures 3 and 4), we use the official Python implementation of the UMAP algorithm (https://umap-learn.readthedocs.io/en/latest/), with hyperparameters `min_dist=1.0` and `n_neighbors=N-1` where N is the number of hidden state datapoints being projected. These hyperparameters were selected to ensure focus on the 'global', high-level structure of the embedding space wrt the partner parameters, rather than small-scale localised clusterings. For all other hyperparameters we use the library default values.

B.3.3 Linear probe analysis

For our linear probe analysis (used in Figure 3), we optimise a linear layer to perform the classification task $\vec{x}_t \mapsto y$ where $\vec{x}$ is the averaged RNN hidden states up to time t from a single episode, and y is the partner's speed at *either* task 1 or 2 for that episode. We train probes separately for different values of t as well as ego agents trained under different conditions. Each probe is trained for a total of 1e3 steps using the Adam optimiser with a learning rate of 1e-2. We use a train-test split of 80-20 over rollout seeds (i.e. where we have 20 seeds per partner speed combination, we randomly assign 16 of those seeds to the train set and 4 to the test set). We report the distance-aware classification accuracy over the test set. For comparison, we train a 'baseline' probe under identical conditions using random $\vec{x}$ (sampled from the standard Normal distribution with the same shape as the hidden states) and the true y labels.

B.4 Other experiment details

B.4.1 Experiments 1-2

To reduce the chance of poor policy convergence due to a random seed, we ran the entire training process over five random seeds per environment layout, and selected the ego agent policy that achieved the highest average episode return at the end of training. During testing, we paired the ego agent with 24 different partners not encountered during training and collected 20 episode rollouts per partner (over 20 random seeds, which randomised the starting state), for a total of 2400 different rollouts. The seeds that achieved the highest return are as follows:

Table 2: Best performing seed (1–5) for each layout and model

	MLP	**RNN**	**RNN Online**	**RNN Single Partner**
Cramped Room	2	3	5	2
FiveByFive	4	2	4	5
Coord Ring	2	3	5	1
Cramped Room V3	1	2	2	5
Cramped Room V4	2	2	2	5

For experiments 1 and 2, we used an ego agent with a fixed cooldown interval of 2 (across both subtasks). We found that if the speed of the ego agent was higher than this, then it would experience less pressure to model the partner agent, converging to a policy that was independent of partner properties. Conversely, at lower speeds, the ego agent struggled to converge to an effective policy.

B.4.2 Experiment 3

In Experiment 3, we test if a 'blind' ego agent that only receives local observations of its own square (position, orientation, and held object) can implicitly model the competence of its partner. For the partner policy, we use the ingredient-preparation subnetwork (subtask 1). During training, the ego agent is paired with a distribution of partners that vary in their probability of taking random actions (versus optimal actions); representing different levels of competence. For testing, we introduce six novel partners with fixed randomness levels $\in \{0, 0.05, 0.1, 0.9, 0.95, 1\}$. This allows us to investigate whether the ego agent adapts its behaviour to both highly competent and highly erratic partners. We hypothesise that the ego will collaborate efficiently with skilled partners, while taking on more of the task when paired with less capable ones.

B.5 Environment layouts

For both training and evaluation, we use five environment layouts with different spatial constraints: *cramped room*, *coord ring*, *fivebyfive*, *cramped_room_v3*, and *cramped_room_v4*. All layouts were chosen on the basis that they incentivise agents to work together rather than independently, and are no larger than 5x5 tiles (to ensure convergence to successful policies). The five layouts are depicted in Figure 8.

(a) Cramped Room (b) FiveByFive (c) Coord Ring (d) Cramped Room V3 (e) Cramped Room V4

Figure 8: Layouts used in the Experiments

B.6 Baselines

To isolate the conditions under which partner modelling emerges, we utilise three **baselines** that allow us to disentangle the effects of training diversity, memory, partner information and architecture on collaborative performance. First, to see if memory plays a role, we include a **feedforward network (FFN)** MLP baseline. This tests whether memory is necessary for adapting to diverse partners (poor performance would suggest that memory is important in partner modelling). We next consider a **single partner specialist**, which is a GRU recurrent agent trained with one partner. This baseline probes whether exposure to diverse partners is necessary to develop a generalisable partner model - poor performance in this baseline would outline that training diversity is cruical for emergent modelling. Finally, we used a **non-influential** - an RNN trained over a distribution of partners, but without the ability to influence them. During rollout with the non-influential RNN, the partner mode was switched halfway through training to allow the ego agent to retain memory of the partner's ability in both subtasks.

B.7 Illustrative videos of agent–partner interactions

To illustrate the experiments and interaction dynamics, we include example videos from the RNN policy trained on a distribution of partners. Although many policies are trained throughout the experiments, these examples are used to highlight the kind of adaptations (and occasional failures to adapt) that emerge under novel test conditions. Videos are included to show the ego agent interacting with two previously unseen partners (using one random seed per layout) in each experiment. In Experiment 1, the agent interacts with a fixed partner throughout the episode — one that is competent

(i.e., speed 1) at ingredient preparation and one at serving. In Experiment 2, the partner switches partway through the episode (at time step 300), from a skilled ingredient preparation partner to a skilled serving partner, or vice versa. During training, partner switches occur at variable times or not at all, so the agent cannot anticipate when or whether a change will occur. In Experiment 3, the blind ego agent is paired with both an unskilled partner and a competent one (at ingredient preparation) - to visualise the large variations in partner ability. These videos are provided in the supplementary materials and demonstrate how the trained policy generalises to new human collaborators.

B.8 Compute

All experiments were run on A100 GPUs, totalling 462 GPU-hours. Policy training used 225 GPU hours for all three experiments. A total of 37,400 rollouts were simulated, totalling 207 GPU-hours.

B.9 Code availability

The full code for this paper is available at: `https://github.com/ruaridhmon/emergent_partner_modelling`