Adaptively Coordinating with Novel Partners via Learned Latent Strategies

Benjamin Li[1] **Shuyang Shi**[1] **Lucia Romero**[2] **Huao Li**[2] **Yaqi Xie**[1] **Woojun Kim**[1]
Stefanos Nikolaidis[3] **Michael Lewis**[2] **Katia Sycara**[1] **Simon Stepputtis**[4]

[1]Carnegie Mellon University [2]University of Pittsburgh
[3]University of Southern California [4]Virginia Tech
{benjil, shuyangs, yaqix, woojunk, sycara}@andrew.cmu.edu
{luciaromero, hul52, cmlewis}@pitt.edu
nikolaid@usc.edu, stepputtis@vt.edu

Abstract

Adaptation is the cornerstone of effective collaboration among heterogeneous team members. In human-agent teams, artificial agents need to adapt to their human partners in real time, as individuals often have unique preferences and policies that may change dynamically throughout interactions. This becomes particularly challenging in tasks with time pressure and complex strategic spaces, where identifying partner behaviors and selecting suitable responses is difficult. In this work, we introduce a strategy-conditioned cooperator framework that learns to represent, categorize, and adapt to a broad range of potential partner strategies in real-time. Our approach encodes strategies with a variational autoencoder to learn a latent strategy space from agent trajectory data, identifies distinct strategy types through clustering, and trains a cooperator agent conditioned on these clusters by generating partners of each strategy type. For online adaptation to novel partners, we leverage a fixed-share regret minimization algorithm that dynamically infers and adjusts the partner's strategy estimation during interaction. We evaluate our method in a modified version of the Overcooked domain, a complex collaborative cooking environment that requires effective coordination among two players with a diverse potential strategy space. Through these experiments and an online user study, we demonstrate that our proposed agent achieves state of the art performance compared to existing baselines when paired with novel human, and agent teammates.

1 Introduction

As an increasing number of AI agents and robots enter our daily lives, it is becoming increasingly important to explore methods of effective collaboration between agent and human. When an agent attempts to collaborate with an unknown partner (whether human or artificial), the challenge can typically be divided into two steps: (1) accurately predicting the partner's behavior, and (2) choosing actions that advance the team toward its common goal. Our approach focuses on understanding and responding to behavioral patterns that distinguish different types of collaborators. Equipping an agent with the ability to identify partner type can enable it to exploit knowledge about that type when choosing its own actions, enabling adaptation for coherent and fluid collaboration. This type of adaptation is essential, as misaligned actions between teammates inevitably decrease overall performance. The problem of adapting to a previously unknown human is an instance of the more general problem of ad hoc teamwork, cooperating with any previously unknown teammate [38]. The difficulties of the problem arise if the agent begins without prior knowledge of its teammate's behavioral patterns, yet must select its own actions to match the teammate's intended strategy

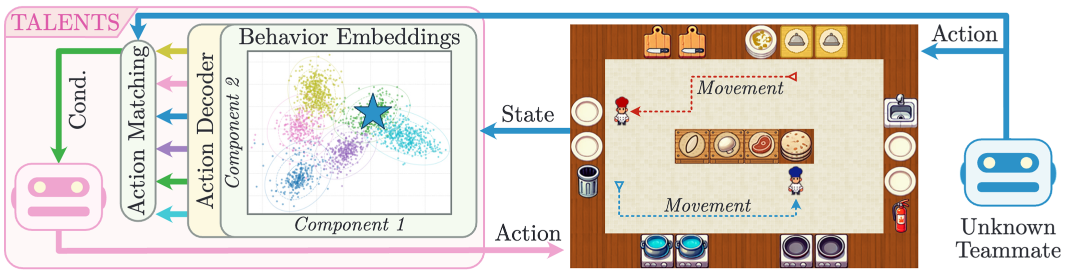


Figure 1: Overview of **TALENTS**: Provided an observation of a teammate, the VAE's latent strategy clusters are queried to generate action predictions. At subsequent timesteps, the teammate's actual actions are compared to these predictions, and the belief over the teammate's latent strategy is progressively updated.

for completing the collective goal. This presents significant challenges given the multiplicity of possible strategies for any collaborative task. The difficulty intensifies in human-agent partnerships, where humans typically employ diverse, non-stationary, and often suboptimal approaches that can shift rapidly during interaction. Consequently, effective collaborative agents must be capable of interpreting their partners' intentions and behaviors in real-time while demonstrating rapid adaptive responses.

In coordinated team settings, roles and specializations can be learned by agents in a centralized manner [23; 43]. However, human-agent teams are often *decentralized*, in that roles and strategies are often implicit and need to be inferred [24]. To account for the diverse set of teammates an agent may encounter at test-time, an effective cooperator agent in this setting must be able to successfully learn and play the best response to each potential teammate's strategy. A popular class of methods that seeks to address this is population-based training, in which a population of agent policies is generated, from which agents are drawn to serve as teammates to train a cooperator [18; 41]. At their core, these methods seek to construct agent populations with sufficient behavioral diversity to span the distribution of strategies exhibited by human players, often aided by a secondary objective during population training that enforces diversity in the set of training partners [4; 25; 34; 39; 51]. To further increase the strategic diversity of training partners, these agent populations can be used to train a generative model, which enables sampling from a continuous latent strategy distribution to simulate a much wider range of novel teammates [22]. Complementary to population-based approaches, agent modeling methods focus on online adaptation by inferring teammate behaviors and conditioning agent actions on these inferred strategies [29; 46; 50; 52]. However, these approaches often rely on a limited or hand-defined collection of strategy types to operate over, or lack the diverse synthetic training data provided by population-based methods. We argue that unifying these two paradigms can enable more effective team collaboration: leveraging diverse population data to learn a latent strategy space from which unique partners can be generated during training, while explicitly reasoning over teammate types at test-time to enable rapid online adaptation and best-response action selection.

In this work, we introduce **T**eam **A**daptation via **L**at**E**nt **N**o-regre**T** **S**trategies (**TALENTS**) (see Fig. 1), a novel method for zero-shot coordination that models partner behaviors in a semantic strategy space. At the core of our **TALENTS** agent is a latent behavior embedding created by a variational autoencoder. This autoencoder is trained from offline trajectory data in the respective environment, collected from a population of baseline agents. With this dataset, we train a variational auto-encoder (VAE) with the goal of predicting each agent's next action and utilize its latent space as behavior representation. Next, we determine an appropriate number and location of behavior clusters utilizing K-Means with silhouette analysis [32]. At test time, the type of a new partner, with respect to the behavior clusters, is determined by assessing each latent cluster mean's decoded prediction accuracy with respect to the partner's actual action (observed at the next timestep). Our agent's belief of its partner is updated according to this per-step loss, and determines the most likely partner type which is then used as a conditioning term for its own action.

We evaluate our agent in a modified Overcooked environment that extends prior work [2; 10; 45] with additional temporal pressure through order timers and bonus rewards for fast deliveries. This setting requires the coordinated use of three cooking stations to prepare two distinct recipes, significantly increasing the demands for effective teamwork and strategic coordination. We conduct experiments in both agent-agent and human-agent zero-shot coordination settings. In agent-agent evaluations, our method demonstrates overall superior performance compared to existing baselines. Furthermore, we deploy these same agents in human studies, demonstrating their ability to collaborate effectively with diverse unseen human partners. In summary, the work introduces the following contributions:

- We introduce a novel method for learning strategy-specific best responses for human-agent collaboration using a generative model that characterizes agents based on latent strategies.
- Our method learns a strategy-conditioned cooperator that assesses teammate type and adapts its behavior through an online regret-minimization framework.
- In extensive teaming experiments, including a human-agent subject study, we show that our agent achieves state-of-the-art performance in a challenging Overcooked environment, adapting to new partners in a similar strategy space in a zero-shot manner.

2 Related Work

Zero-Shot Coordination Collaborating with partners that were unseen by agents during training is the central goal of Zero-Shot Coordination (ZSC) [14; 38]. Self-play (SP) is a common approach to address this problem, in which agents establish a joint strategy by pairing with a copy of itself during training [33].These approaches have been shown to surpass human performance in zero-sum games in which agents can exploit sub-optimalities of opponents at test time [1; 37], but are often unable to achieve comparable performance in common-payoff cooperative settings since these agents tend to develop rigid behavior patterns [2]. Population-based training (PBT) methods combat this rigid convention formation by optimizing over diverse partner sets [18; 41]. Fictitious Co-Play (FCP) [39] leverages intermediate checkpoints of partner policies to simulate teammates of different skill levels, while other methods increase the strategy coverage of the population by means of diversity or entropy related objectives [25; 30; 51], cross-play performance [4; 34], or reward shaping [40; 44; 48]. E3T [47] applies the entropy objectives of PBT methods to the more efficient SP training paradigm, defining the trainer partner as a mixture between the ego policy (maximizing coordination) and a random policy (maximizing entropy). GAMMA [22] further enhances the breadth of PBT strategy coverage by learning a generative model over the population of policies, encoding trajectories into latent representations. They are then able to perform targeted sampling over these latents to generate novel agents spanning the strategy space of the original population. Our proposed **TALENTS** method applies the same approach of GAMMA by training a generative model from PBT agent trajectory rollouts, but differs in that we then cluster the learned latent space and perform controlled sampling over the clusters. We treat these clusters as a population of strategy types, training a cooperator agent conditioned on these clusters to explicitly learn role-compatible behaviors.

Strategy Inference Inferring the partner type allows agents to form beliefs about other agents in the environment and optimize its own behavior in response to those beliefs, whether in adversarial or cooperative environments. A common strategy is to learn latent embeddings of partner type, role, or skill from a dataset of trajectories [13; 28; 36; 46; 42]. Other methods directly estimate the opponent's policy updates [6], perform Bayesian inference of partner type [3; 7; 9; 15; 52], or learn a partner model through interaction during training [16; 26; 47]. In MeLIBA [52], a Bayesian filtering objective is used to model confidence in the partner's type as the trajectory rolls out, allowing a best response agent to respond optimally given its knowledge about its opponent. Grover et al. [11] proposes a triplet loss term to encourage distinguishability between different types of opponents, and Papoudakis and Albrecht [29] expands on this by using reinforcement learning to train an agent conditioned on this latent embedding. Zhao et al. [50] use apprenticeship learning to model the different types of observed strategies, then online mixture of experts to select compatible policies over the course of an episode. Li et al. [19] trains a library of teammate types, each with a different role that, when faced with an unknown human, use cross-entropy to choose strategies that are highly compatible with the human's inferred current strategy. A separate but adjacent line of work learns Theory of Mind models to predict partner preferences, intentions, or beliefs [20; 21; 31; 49]. Our work proposes to use the same latent space that generated training partners to perform strategy inference. At train-time, the cooperator agent is conditioned on the latent cluster its teammate is generated from in order to learn strategy-specific best responses. At test-time, we utilize a tracking-regret minimized approach to enable *intra*-episodic adaptation, improving our agent's ability when paired with teammates that may change their strategy several times over the course of a single episode.

3 Preliminaries

In this section, we establish the mathematical framework for our multi-agent reinforcement learning setup conditioned on latent strategies.

3.1 Two-Player Markov Decision Process

A Markov Decision Process (MDP) is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}_1, \mathcal{A}_2, P, R, \gamma, H)$. While we present the two-agent formulation for clarity, this framework naturally extends to an arbitrary number of agents. In this formulation, ($\mathcal{S}$) represents a set of states, and ($\mathcal{A}_i$) denotes the set of actions available to agent (i), where ($i \in 1, 2$). The transition probability function $P : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{S} \rightarrow [0, 1]$ defines the probability $P(s'|s, a)$ of transitioning to state s' where a joint action a is taken in state s. The team reward function $R : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{S} \rightarrow \mathbb{R})$ specifies the reward $R(s, a_1, a_2, s')$ received by both agents when transitioning from state s to state s' after executing the joint actions a_1 and a_2. The discount factor $\gamma \in [0, 1)$ determines the significance of future rewards, influencing how current decisions are evaluated relative to potential long-term outcomes. Lastly, H represents the horizon of the problem, which is typically omitted in infinite horizon problems.

The goal in a multi-agent MDP is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}_1, \mathcal{A}_2$ that maximizes the expected discounted sum of rewards, defined as $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_{1,t}, a_{2,t}, s_{t+1})]$.

3.2 Hidden Parameter Markov Decision Process

A Hidden Parameter Markov Decision Process (HiP-MDP) extends the standard MDP framework to incorporate unobserved parameters that influence the transition and reward dynamics. Formally, a HiP-MDP is defined as a tuple $\mathcal{M}(\mathcal{Z}) = (\mathcal{S}, \mathcal{A}_1, \mathcal{A}_2, \mathcal{Z}, P, P^z, R_1, R_2, \gamma, H)$ where $\mathcal{S}$, $\mathcal{A}$, γ, and H are defined as in the standard MDP. $\mathcal{Z}$ is a set of hidden parameters and $P : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{S} \times \mathcal{Z} \rightarrow [0, 1]$ is a transition probability function that depends on the hidden parameter $z \in \mathcal{Z}$. $P^z : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{S} \times \mathcal{Z}$ is a transition probability function for the hidden parameter $z \in \mathcal{Z}$, representing how z may evolve over the course of an episode. R is defined as $\mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{S} \times \mathcal{Z} \rightarrow \mathbb{R}$, representing the reward function that depends on the hidden parameter $z \in \mathcal{Z}$.

For a specific, constant value of the hidden parameter $z \in \mathcal{Z}$, the HiP-MDP reduces to a standard MDP $\mathcal{M}_z = (\mathcal{S}, \mathcal{A}_1, \mathcal{A}_2, P(\cdot|z), R(\cdot|z), \gamma, H)$. Given the two-player HiP-MDP framework, the learning objective for each agent i can be formulated as:

$$\max_{\pi_i} \mathbb{E}_{\pi_1, \pi_2, z}[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_{1,t}, a_{2,t}, s_{t+1}, z)] \tag{1}$$

3.3 Tracking Regret Minimization

To address the strategic adaptations and non-stationarity inherent in our HiP-MDP setting, where teammates may shift between different strategies during an episode, we incorporate tracking regret minimization. This permits a limited number of optimal response policy changes over the course of an episode while maintaining performance guarantees. In online learning, at each time step t, an agent selects a decision x_t from a convex set $\mathcal{X}$ and subsequently observes a loss function $f_t : \mathcal{X} \rightarrow \mathbb{R}$. Unlike static regret, which compares against a single fixed decision, tracking regret compares the agent's performance against a sequence of decisions that can change a limited number of times.

Formally, for a partition $\mathcal{I} = \{I_1, I_2, \ldots, I_m\}$ of the time horizon $[T]$ into m intervals, and a sequence of comparators $\mathbf{u} = (u_1, u_2, \ldots, u_m)$ where each $u_j \in \mathcal{X}$, the tracking regret is defined as:

$$Reg_T^s(\mathcal{I}, \mathbf{u}) = \sum_{t=1}^{T} f_t(x_t) - \sum_{j=1}^{m} \sum_{t \in I_j} f_t(u_j). \tag{2}$$

The worst-case tracking regret against any sequence with at most m expert policies (i.e. $m-1$ expert changes) is

$$Reg_T^s(m) = \max_{\mathcal{I},\mathbf{u}} Reg_T^s(\mathcal{I}, \mathbf{u}). \tag{3}$$

Fixed Share [12] with optimally-selected switching rate α achieves the following bound for any comparator sequence of m segments:

$$Reg_T^s(m) = O\Big(\sqrt{T\,(m \ln N + m \ln \frac{T}{m})}\Big). \tag{4}$$

4 Team Adaptation via No-Regret Strategies

We introduce **TALENTS**, a method of training a general cooperator agent that is conditioned on a wide range of strategies and behaviors. In the online setting, the cooperator agent can assess the strategy of its partner and adapt to it by playing the corresponding best response policy. Given a dataset of offline trajectories exhibiting a wide range of strategies, we first use a variational autoencoder (VAE) [17] to learn a latent space of strategy vectors. We then partition the latent space into discrete clusters, with each cluster representing a unique high-level strategy. Next, we train a best response agent over the different strategy clusters, generating a strategic partner agent for every episode by selecting a strategy cluster and sampling from its latent mean. We condition the cooperator agent on its training partner's mean in order to learn a strategy-dependent best response. At test time, we use the fixed share algorithm to select a latent mean to condition our cooperator agent on, allowing intra-episodic adaptation to the new partner.

4.1 Strategy Learning

To form the latent strategy space, we utilize an offline dataset of trajectories generated by agent-agent rollouts. Note that this data can alternatively come from human gameplay [13; 50], or a mix of human and agent data [22]. However, this work focuses on exploring ad hoc collaboration with humans without seeing human data at train time. The dataset, $\mathcal{D}_{traj} = \{\tau_i\}_{i=1}^N$, is composed of observation-action pairs for each agent collected from a pre-trained set of population agents, $\{o_t^{(i)}, a_t^{(i)}\}_{t=0}^T, i \in \{1, 2\}$. This dataset contains a diverse set of agent strategies and behaviors increasing the amount of covered latent space over the set of possible agent types. With this dataset, we subsequently train a VAE to approximate the posterior distribution of the trajectories. The VAE architecture consists of an encoder network $q_\phi(z|\tau)$ that maps trajectories τ to a distribution over latent strategy variables z, and a sequential decoder network $p_\theta(a_{t:t+H}|z, o_t)$ that predicts the next H actions given the latent variable and current observation, insight that follows from Zintgraf et al. [52]. Specifically, the encoder parameterized by ϕ compresses the trajectory information into a low-dimensional latent space, producing parameters of a multivariate Gaussian distribution $\mathcal{N}(\mu_\phi(\tau), \Sigma_\phi(\tau))$. By predicting the H future actions of the agent, we learn a representation of the agent's long-term intent. Importantly, we will later be able to leverage the decoder as a method for generating agents of specific behavior types.

The objective of the sequential VAE is to maximize the log-likelihood of the future actions given past observations and actions: $\log p_\theta(a_{t:t+H}|o_t, \tau_{t-h:t})$. However, directly computing this is intractable as it requires marginalizing over all possible latent variables: $\log p_\theta(a_{t:t+H}|o_t, \tau_{t-h:t}) = \log \int p_\theta(a_{t:t+H}|z, o_t)p(z|\tau_{t-h:t})dz$. Instead, we optimize the Evidence Lower Bound (ELBO):

$$\mathcal{L}(\theta, \phi; \tau) = \mathbb{E}_{z\sim q_\phi(z|\tau_{t-h:t})}[\log p_\theta(a_{t:t+H}|z, o_t)] - \beta D_{KL}(q_\phi(z|\tau_{t-h:t})||p(z)) \tag{5}$$

Once we have learned our VAE, we cluster the latent space. To this end, we perform K-means clustering with silhouette analysis [32] to determine the optimal clusters and the number of clusters in an unsupervised fashion. Each cluster represents a unique agent type, with which we utilize to train our best response cooperator agent.

4.2 Learning a Strategy-Conditioned Cooperator Agent

To train our agent, we utilize the strategy clusters and leverage the generative capability of our trained VAE. We retrieve actions for each agent type by sampling from the latent mean of each strategy cluster and decoding them, conditioned on the observed environment state. For each episode in the training process, we randomly select one such cluster to sample from, using the priority-based sampling technique proposed in Zhao et al. [51]. Using a categorical variable corresponding to the partner cluster, we learn a bias vector corresponding to the cooperator's action space. The actor network's output logits are then biased by this vector, explicitly encouraging or dissuading the cooperator from taking certain actions depending on the partner type (see algorithm 1). This motivates the cooperator to learn the unique conventions needed to best respond to its partner (Note that we utilize a high-level action space in our experiments). We train our cooperator agent using independent PPO [5; 35].

Algorithm 1 Strategy-Conditioned Cooperator Training

```
 1: **Input:** Trained VAE encoder $q_\phi(z|\tau)$, decoder $p_\theta(a_{t:t+H}|z, o_t)$, strategy clusters
    $\{\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2), \ldots, \mathcal{N}(\mu_K, \sigma_K^2)\}$, priority distribution $p(c)$
 2: Initialize cooperator policy $\pi_\theta(\cdot|o, c)$
 3: **for** each training episode **do**
 4:     Sample a strategy cluster $c \sim p(c)$
 5:     Sample latent strategy $z \sim \mathcal{N}(\mu_c, \sigma_c^2 I)$
 6:     **for** each timestep $t$ in episode **do**
 7:         Observe state $o_t$ for cooperator agent
 8:         Compute action bias vector $b_c$ from cluster embedding matrix $E$, where $b_c = E[c]$
 9:         Compute action logits from actor network $l_t = f_\theta(o_t)$
10:         Apply bias to logits: $\tilde{l}_t = l_t + b_c$
11:         Sample action from biased distribution $a_t \sim \text{softmax}(\tilde{l}_t)$
12:         Sample generated action for partner $a_{c,t} \sim p_\theta(\cdot \mid z, o_t)$
13:         Execute actions and collect rewards
14:     **end for**
15:     Update $\pi_\theta$ using PPO with collected trajectory
16:     Update priority-based sampling weights using total episodic return
17: **end for**
18: **Return:** Strategy-conditioned cooperator policy $\pi_\theta$
```

4.3 Online Adaptation to Novel Partners

The trained cooperator agent will be able to best respond to any partner strategy in the VAE distribution, provided that it has knowledge on which latent strategy the partner is playing. This adds the additional challenge of needing to infer the partner's type in the online setting. An approach that has been explored in previous research is to encode the novel partner's trajectory segment into the latent space and conditioning the cooperator on that [13; 46], but in practice distribution shift between train-time and test-time trajectories can lead to brittleness or suboptimal encoded representations, particularly when playing with humans. Instead, we utilize regret minimization and a no-regret algorithm to represent the partner's behavior.

We implement a variant of the fixed-share algorithm to infer the partner type, setting each strategy cluster as an "expert". At test time, we sample a latent representation from each cluster and decode into the predicted action, conditioned on the current observation. At the next timestep, we calculate the incurred loss by comparing the previous timestep's prediction with the actual action of the partner, then update the weights of each expert accordingly. Importantly, the fixed-share algorithm differs from standard static no-regret methods (such as Hedge [8], FTRL [27]), in that it minimizes the regret given that the expert may switch policies $m - 1$ times during an episode (algorithm 2). This is to account for the fact that policies at test-time may be non-stationary and might change latent strategy z over the course of the episode, due to influence from its partner or continuous adaptation and learning from the task. We examine the effect of fixed-share versus static-regret minimization in Figure 4.

Fixed-share allows the agent to model these strategy changes and adapt its behavior during the episode. It also provides the regret bound that if the agent encounters a truly novel strategy at test time, the agent will only perform as poorly as the best-response to the closest observed strategy during training.

Algorithm 2 Online Adaptation to Novel Partners with Fixed-Share

```
1: Input: Latent clusters $\{\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2), \ldots, \mathcal{N}(\mu_K, \sigma_K^2)\}$, trained cooperator policy
   $\pi_\theta(\cdot|o, c)$, VAE decoder $p_\theta$, switching parameter $\alpha \in (0, 1)$, learning rate $\eta > 0$
2: Initialize weight vector $w^1 = (1/K, \ldots, 1/K)$ uniformly over $K$ experts
3: for $t = 1, 2, \ldots$ do
4:     Observe current state $o_t$
5:     for each cluster $c \in \{1, 2, \ldots, K\}$ do
6:         Sample latent strategy $z_c \sim \mathcal{N}(\mu_c, \sigma_c^2 I)$
7:         Predict partner action $\hat{a}_t^c = \arg\max_a p_\theta(a|z_c, o_t)$
8:     end for
9:     Compute leading expert $c^* = \arg\max_c w_c^t$
10:    Execute cooperator action $a_t \sim \pi_\theta(\cdot|o_t, c^*)$
11:    Observe partner's actual action $a_t^p$
12:    Compute loss for each expert: $\ell_c^t = -\log p_\theta(a_t^p|z_c, o_t)$
13:    Update pre-sharing weights: $\tilde{w}_c^{t+1} = w_c^t \exp(-\eta \ell_c^t)$
14:    Normalize: $\tilde{w}^{t+1} = \tilde{w}^{t+1} / \sum_{c=1}^{K} \tilde{w}_c^{t+1}$
15:    Apply fixed-share update: $w_c^{t+1} = (1 - \alpha)\tilde{w}_c^{t+1} + \alpha \sum_{j=1}^{K} \tilde{w}_j^{t+1} / K$
16: end for
```

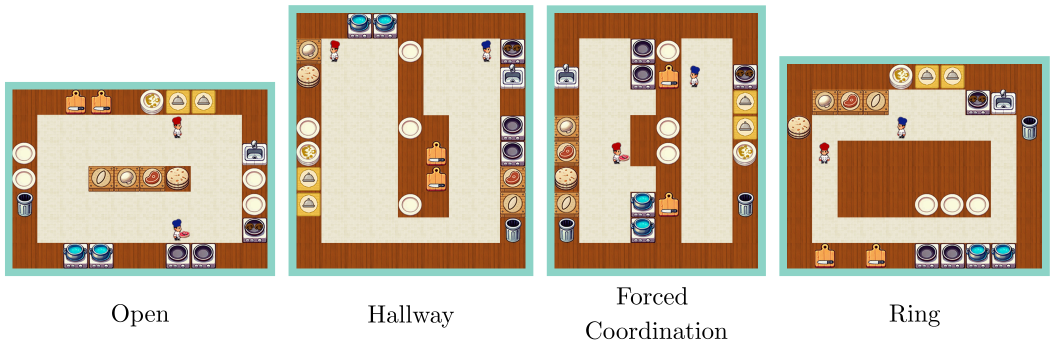


Figure 2: The four Overcooked layouts used in experiments.

5 Experiments

We evaluate our algorithm in a modified version of the Overcooked-ai environment [2] across agent-agent and human-agent gameplay. In this domain, we evaluate across four layouts (see Fig. 2) for agent-agent teams, and three for human-agent teams. These layouts are conceptually similar to those proposed in [2], but feature additional task complexity and timing constraints to further highlight the need for effective teamwork.

5.1 Agent-Agent Zero-Shot Coordination

We evaluate TALENTS in agent-agent settings using three different agent populations: Fictitious Co-Play (FCP) [39], Maximum Entropy Population (MEP) [51], and Behavior Preference (BP) Agents [44]. For the FCP and MEP populations, eight self-play agents are initialized with different seeds and trained. The MEP population is trained with an additional population entropy loss [51]. The final FCP and MEP populations consist of the final policies as well as two intermediary policy checkpoints per policy. As stated by Wang et al. [44], a well-designed population of behavior preference presents behavior diversity that is more similar to that of human teams, compared to other population generation methods. Therefore, we select it as another training population and train 24 behavior preference (BP) agents to cover a wide spectrum of partner strategies. The BP is represented by a linear combination of event-based shaped reward terms that encourage the agent to learn various behaviors. Using these populations, we compare our method to two baselines: the population-trained best response cooperator (**BR**), and a **GAMMA** cooperator. Notably, we select **GAMMA** as a

Table 1: Resulting scores of games played with a held-out set of 12 behavior-preferenced SP agents (mean ± standard error).

Pop	Agent	Open	Hallway	Forced-Coord	Ring
FCP	TALENTS	$\mathbf{710.36 \pm 88.75}$	$\mathbf{635.59 \pm 107.54}$	34.38 ± 6.59	$\mathbf{596.19 \pm 33.34}$
	GAMMA	616.67 ± 14.99	537.60 ± 26.75	38.36 ± 6.65	395.03 ± 10.09
	BR	427.07 ± 14.17	366.14 ± 70.50	$\mathbf{56.09 \pm 12.33}$	288.61 ± 31.85
MEP	TALENTS	$\mathbf{720.84 \pm 72.63}$	$\mathbf{640.51 \pm 27.36}$	36.53 ± 2.58	$\mathbf{568.11 \pm 8.99}$
	GAMMA	682.38 ± 14.99	575.23 ± 21.17	31.89 ± 4.33	369.15 ± 12.47
	BR	437.00 ± 27.76	440.48 ± 113.03	$\mathbf{50.40 \pm 14.29}$	335.61 ± 28.48
BP	TALENTS	$\mathbf{842.39 \pm 36.47}$	$\mathbf{642.94 \pm 81.00}$	56.55 ± 9.09	$\mathbf{647.62 \pm 16.96}$
	GAMMA	573.93 ± 52.69	513.47 ± 34.06	47.52 ± 2.61	387.58 ± 17.41
	BR	469.88 ± 52.01	454.26 ± 86.5	$\mathbf{78.53 \pm 8.55}$	640.43 ± 28.06

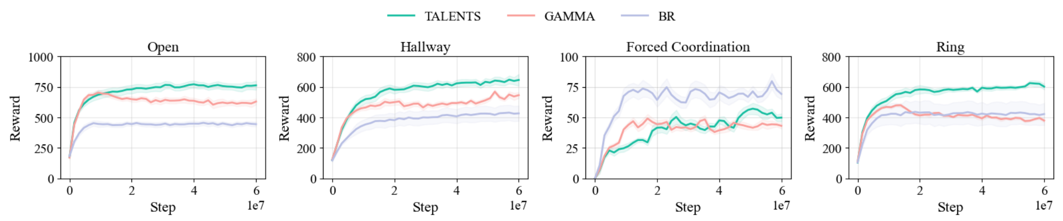


Figure 3: Evaluation performance during training for each layout, averaged across agents trained with FCP, MEP, and BP populations.

baseline due to the method also leveraging a generative model in its cooperator training process. Through our experiments, we wish to evaluate if our strategy-specific agent generation and cooperator conditioning will result in a more robust adaptive cooperator agent.

To generate the trajectories for VAE training, we select agents from the population to form a team and perform several joint rollouts, repeating until joint trajectories between all the agents in the population is achieved. This set of trajectories is used to train VAEs for both **TALENTS** and **GAMMA**.

To compare the performance of the methods with unseen agent partners, we evaluate using a held-out set of behavior preference agents representing a diverse span of strategies and competencies (see Tab. 1). We find that with the exception of the *Forced Coordination* layout, **TALENTS** is able to achieve the highest reward out of the evaluated methods, regardless of the population it was trained on. We hypothesize that both **TALENTS** and **GAMMA** are less effective than the population **BR** agent in *Forced Coordination* due to there being a very clear partition of duties between the teammates. **GAMMA** and **TALENTS**, which focus on exploring the diverse space of possible partners (many of which exhibit ineffective or uncooperative behavior), receive less reward signal when paired with agents of lower skill and are, as a result of the sparser training, less proficient in the overall task than the **BR** agent.

To evaluate the efficacy of the tracking-regret minimization framework, we ablate **TALENTS** by replacing the fixed-share algorithm with a static-regret minimizer. We use exponential weights to keep the weight update structure constant, only ablating the weight-sharing component. We then evaluate our cooperator with a partner policy randomly selected from a held-out behavior preference set, but replace the partner agent with a policy with a different behavior preference halfway through the episode. We see in Fig. 4 that the static-regret ablation is unable to update its belief to the new partner, and suffers lowered reward in the latter half of the episode as a result.

All models were trained on a server with 2× AMD EPYC 7713 64-core processors, 1.08 TB of system memory, and 5× Nvidia RTX 6000 Ada GPUs. In our experiments, VAE train time is typically 4-6 hours, while a cooperator agent can be trained in approximately 24 hours.

5.2 Human-Agent Zero-Shot Coordination

In this section, we evaluate our proposed method against state-of-the-art multi-agent coordination baselines with real human participants. Data collection is done on online crowd-sourcing platforms

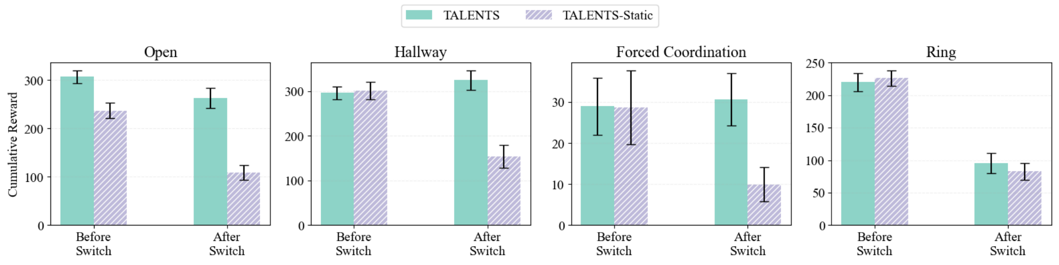


Figure 4: Accumulated reward with a partner policy swap midway through the episode ($t = 1200$). Error bars are one standard error from the mean.

Prolific and Cloud Research. Each participant completed three rounds of gameplay, each lasting for four minutes, with different agents as the partner. The entire session took around 30 minutes to complete and participants received a $7 base payment plus up to a $3 bonus depending on their performance. For these experiments, we limited the action frequency of agents to match that of human capability. Team score and subjective ratings were recorded as the performance metrics.

5.2.1 Experiment Design

A mixed experiment design is used for the human-agent evaluation, where agent type is the within subject variable (three levels: **TALENTS**, **GAMMA**, **BR**) and the map layout is the between subject variable (three levels: hallway, open, forced coordination). At the end of each round, participants are asked to complete a set of questionnaires measuring their subjective preference for the agent partner they just interacted with. The questionnaire consists of five surveys, each measuring their workload (NASA-TLX), perceived team fluency, trust, coordination, and satisfaction on 7-point Likert scales.

5.2.2 Results

We collected data from a total of 119 participants. To control data quality, we filtered out team trajectories from inactive participants and survey responses from those who incorrectly answered the trap questions. As shown in Fig 5, our agent significantly outperforms both baselines in team score and subjective ratings. Mixed ANOVA tests show significant main effects for team score ($F(2, 166) = 5.76, p = .003$), with our agent achieving significantly higher score than both baselines in post-hoc comparisons ($p < .05$). Similarly, our agent receives higher subjective ratings in team fluency ($F(2, 122) = 4.31, p = .02$) and perceived trust ($F(2, 122) = 3.23, p = .04$) compared to than the BR baseline ($ps < .05$).

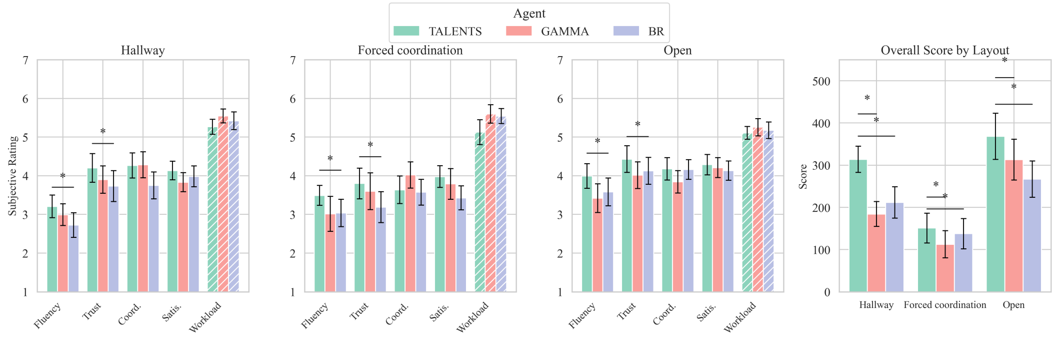


Figure 5: Human-agent teamwork evaluation comprises team scores and participants' subjective ratings of agent teammates. Perceived workload (shaded) is lower if better. Statistically significant differences between agents are marked by asterisks. Error bars are one standard error from the mean.

6 Conclusion

In this paper, we proposed TALENTS, a method for training a strategy-conditioned intra-episodic adaptive cooperator via generative teammates. Our approach combines a variational autoencoder to learn a latent strategy space, unsupervised clustering to identify distinct strategy types, and a fixed-share algorithm for online adaptation. The experimental results in the Overcooked environment demonstrate that our strategy-conditioned cooperator effectively identifies and adapts to partner strategies, achieving better performance than naive cooperators and significantly outperforming self-play agents trained with different partners. Our approach addresses a key limitation of existing methods by enabling intra-episodic adaptation to previously unseen and continuously changing partner strategies at test time. By representing strategies in a continuous latent space and clustering them into discrete types, we create a best-response mapping that allows for flexible adaptation. The fixed-share algorithm's ability to account for non-stationary partner behavior further enhances our method's robustness in dynamic collaborative settings.

6.1 Limitations

Our approach shows promising performance in the Overcooked environment but has limitations. It struggles on the Forced Coordination layout due to sparse reward signals when paired with ineffective or low-skill partners, which additionally causes the priority sampling method to bias toward choosing those partners in future episodes. Generalization to novel teammates is limited by the VAE's interpolation capabilities, meaning our agent can only truly generalize to new teammates if they exhibit similar behaviors to the training data. However, we demonstrate that our agent can successfully play with humans without having been trained on real human data. Furthermore, to better align with human gameplay, which requires real-time interactions, we artificially slowed down agent actions to match human actions-per-second rates. While agents act at every step, with the game designed for ten steps per second, humans typically only take 3–4 actions in the same timeframe. This slowdown may hinder agents' long-term planning capabilities.

References

[1] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365 (6456):885–890, 2019.

[2] 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.

[3] Georgios Chalkiadakis and Craig Boutilier. Coordination in multiagent reinforcement learning: A bayesian approach. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 709–716, 2003.

[4] Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. Generating diverse cooperative agents by learning incompatible policies. In *The Eleventh International Conference on Learning Representations*, 2023.

[5] Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[6] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130, 2018.

[7] 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 *International Conference on Machine Learning*, pages 1942–1951. PMLR, 2019.

[8] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[9] Haobo Fu, Ye Tian, Hongxiang Yu, Weiming Liu, Shuang Wu, Jiechao Xiong, Ying Wen, Kai Li, Junliang Xing, Qiang Fu, et al. Greedy when sure and conservative when uncertain about the opponents. In *International Conference on Machine Learning*, pages 6829–6848. PMLR, 2022.

[10] Tobias Gessler, Tin Dizdarevic, Ani Calinescu, Benjamin Ellis, Andrei Lupu, and Jakob Nicolaus Foerster. Overcookedv2: Rethinking overcooked for zero-shot coordination. In *The Thirteenth International Conference on Learning Representations*.

[11] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *International conference on machine learning*, pages 1802–1811. PMLR, 2018.

[12] Mark Herbster and Manfred K Warmuth. Tracking the best expert. *Machine learning*, 32(2): 151–178, 1998.

[13] Joey Hong, Sergey Levine, and Anca Dragan. Learning to influence human behavior with offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:36094–36105, 2023.

[14] 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.

[15] Dana Hughes, Akshat Agarwal, Yue Guo, and Katia Sycara. Inferring non-stationary human preferences for human-agent teams. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1178–1185. IEEE, 2020.

[16] 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.

[17] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[18] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[19] Huao Li, Tianwei Ni, Siddharth Agrawal, Fan Jia, Suhas Raja, Yikang Gui, Dana Hughes, Michael Lewis, and Katia Sycara. Individualized mutual adaptation in human-agent teams. *IEEE Transactions on Human-Machine Systems*, 51(6):706–714, 2021. doi: 10.1109/THMS. 2021.3107675.

[20] Huao Li, Ini Oguntola, Dana Hughes, Michael Lewis, and Katia Sycara. Theory of mind modeling in search and rescue teams. In *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 483–489, 2022. doi: 10.1109/ RO-MAN53752.2022.9900572.

[21] Huao Li, Yu Chong, Simon Stepputtis, Joseph P Campbell, Dana Hughes, Charles Lewis, and Katia Sycara. Theory of mind for multi-agent collaboration via large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 180–192, 2023.

[22] Yancheng Liang, Daphne Chen, Abhishek Gupta, Simon S Du, and Natasha Jaques. Learning to cooperate with humans using generative agents. *Advances in Neural Information Processing Systems*, 37:60061–60087, 2024.

[23] Yuntao Liu, Yuan Li, Xinhai Xu, Yong Dou, and Donghong Liu. Heterogeneous skill learning for multi-agent tasks. *Advances in neural information processing systems*, 35:37011–37023, 2022.

[24] Dylan P Losey, Mengxi Li, Jeannette Bohg, and Dorsa Sadigh. Learning from my partner's actions: Roles in decentralized robot teams. In *Conference on robot learning*, pages 752–765. PMLR, 2020.

[25] Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In *International conference on machine learning*, pages 7204–7213. PMLR, 2021.

[26] Long Ma, Yuanfei Wang, Fangwei Zhong, Song-Chun Zhu, and Yizhou Wang. Fast peer adaptation with context-aware exploration. In *International Conference on Machine Learning*, pages 33963–33982. PMLR, 2024.

[27] Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 525–533. JMLR Workshop and Conference Proceedings, 2011.

[28] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*, pages 189–196, 2015.

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

[30] Jack Parker-Holder, Aldo Pacchiano, Krzysztof M Choromanski, and Stephen J Roberts. Effective diversity in population based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18050–18062, 2020.

[31] 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.

[32] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[33] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. doi: 10.1147/rd.33.0210.

[34] Bidipta Sarkar, Andy Shih, and Dorsa Sadigh. Diverse conventions for human-ai collaboration. *Advances in Neural Information Processing Systems*, 36:23115–23139, 2023.

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

[36] Andy Shih, Stefano Ermon, and Dorsa Sadigh. Conditional imitation learning for multi-agent games. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 166–175. IEEE, 2022.

[37] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[38] Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1504–1509, 2010.

[39] DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. *Advances in Neural Information Processing Systems*, 34: 14502–14515, 2021.

[40] Zhenggang Tang, Chao Yu, Boyuan Chen, Huazhe Xu, Xiaolong Wang, Fei Fang, Simon Shaolei Du, Yu Wang, and Yi Wu. Discovering diverse multi-agent strategic behavior via reward randomization. In *International Conference on Learning Representations*.

[41] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

[42] Chen Wang, Claudia Pérez-D'Arpino, Danfei Xu, Li Fei-Fei, Karen Liu, and Silvio Savarese. Co-gail: Learning diverse strategies for human-robot collaboration. In *Conference on Robot Learning*, pages 1279–1290. PMLR, 2022.

[43] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. ROMA: Multi-agent reinforcement learning with emergent roles. 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 9876–9886. PMLR, 13–18 Jul 2020.

[44] Xihuai Wang, Shao Zhang, Wenhao Zhang, Wentao Dong, Jingxiao Chen, Ying Wen, and Weinan Zhang. Zsc-eval: An evaluation toolkit and benchmark for multi-agent zero-shot coordination. *Advances in Neural Information Processing Systems*, 37:47344–47377, 2024.

[45] Sarah A Wu, Rose E Wang, James A Evans, Joshua B Tenenbaum, David C Parkes, and Max Kleiman-Weiner. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science*, 13(2):414–432, 2021.

[46] Annie Xie, Dylan Losey, Ryan Tolsma, Chelsea Finn, and Dorsa Sadigh. Learning latent representations to influence multi-agent interaction. In *Conference on robot learning*, pages 575–588. PMLR, 2021.

[47] Xue Yan, Jiaxian Guo, Xingzhou Lou, Jun Wang, Haifeng Zhang, and Yali Du. An efficient end-to-end training approach for zero-shot human-ai coordination. *Advances in Neural Information Processing Systems*, 36:2636–2658, 2023.

[48] 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.

[49] Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, et al. Proagent: building proactive cooperative agents with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17591–17599, 2024.

[50] Michelle Zhao, Reid Simmons, and Henny Admoni. Coordination with humans via strategy matching. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9116–9123. IEEE, 2022.

[51] Rui Zhao, Jinming Song, Yufeng Yuan, Haifeng Hu, Yang Gao, Yi Wu, Zhongqian Sun, and Wei Yang. Maximum entropy population-based training for zero-shot human-ai coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6145–6153, 2023.

[52] 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, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.

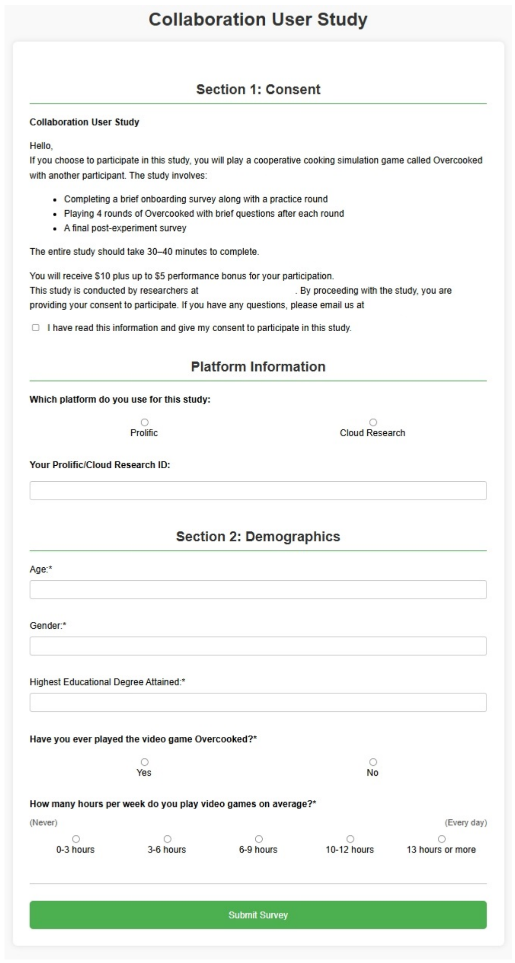
A Human Experiment Details

A.1 Experimental Design

Human study participants were initially acquainted with the premise of the domain via a series of instruction screens (Figs A.1, A.2, A.3), then had the opportunity to test the game mechanics through a single-player tutorial session (Fig A.4a). In the gameplay phase of the experiment (Fig A.4b), participants were randomly assigned to one of the three human-study layouts (open, hallway, forced coordination). Each participant then played three games, one with each evaluated agent (TALENTS, GAMMA, and Population Best Response), presented in a randomized order to control for sequence effects. Episode length was $T = 2400$ environment timesteps (4 minutes). Many previous works in the Overcooked domain use a game length of $T = 400$ or $T = 600$ [34; 22; 39; 2]. We selected a longer game-length both due to the longer-horizon tasks of the modified Overcooked environment as well as to better compare the intra-episodic team adaptation capability of the algorithms and reduce the amount of randomness in our results.

Upon completion of each game, participants were asked to complete a post-game survey, consisting of NASA-TLX, team fluency, team trust, coordination, and satisfaction surveys, shown in Figure A.5.

A.2 Online Data Collection Instructions and Interface



Collaboration User Study

Section 1: Consent

Collaboration User Study

Hello,
If you choose to participate in this study, you will play a cooperative cooking simulation game called Overcooked with another participant. The study involves:

- Completing a brief onboarding survey along with a practice round
- Playing 4 rounds of Overcooked with brief questions after each round
- A final post-experiment survey

The entire study should take 30–40 minutes to complete.

You will receive $10 plus up to $5 performance bonus for your participation.
This study is conducted by researchers at . By proceeding with the study, you are providing your consent to participate. If you have any questions, please email us at

☐ I have read this information and give my consent to participate in this study.

Platform Information

Which platform do you use for this study:

○ Prolific ○ Cloud Research

Your Prolific/Cloud Research ID:

Section 2: Demographics

Age:*

Gender:*

Highest Educational Degree Attained:*

Have you ever played the video game Overcooked?*

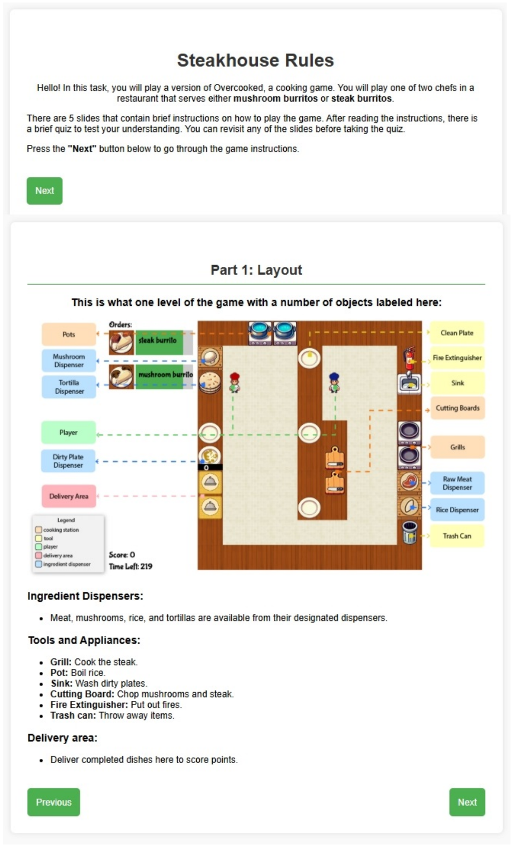
○ Yes ○ No

How many hours per week do you play video games on average?*

(Never) (Every day)

○ 0-3 hours ○ 3-6 hours ○ 6-9 hours ○ 10-12 hours ○ 13 hours or more

(a)



Steakhouse Rules

Hello! In this task, you will play a version of Overcooked, a cooking game. You will play one of two chefs in a restaurant that serves either **mushroom burritos** or **steak burritos**.

There are 5 slides that contain brief instructions on how to play the game. After reading the instructions, there is a brief quiz to test your understanding. You can revisit any of the slides before taking the quiz.

Press the **"Next"** button below to go through the game instructions.

Part 1: Layout

This is what one level of the game with a number of objects labeled here:

Ingredient Dispensers:

- Meat, mushrooms, rice, and tortillas are available from their designated dispensers.

Tools and Appliances:

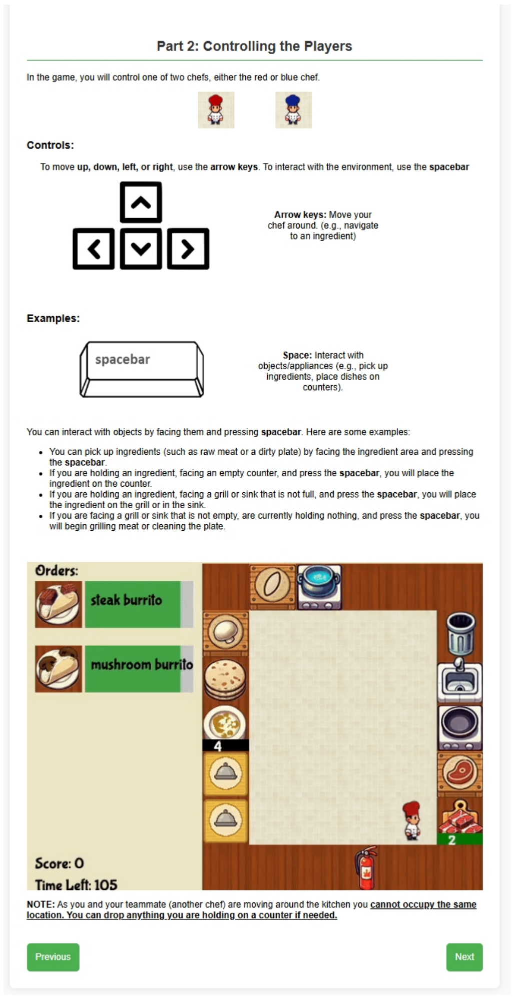
- **Grill:** Cook the steak.
- **Pot:** Boil rice.
- **Sink:** Wash dirty plates.
- **Cutting Board:** Chop mushrooms and steak.
- **Fire Extinguisher:** Put out fires.
- **Trash can:** Throw away items.

Delivery area:

- Deliver completed dishes here to score points.

(b)

Figure A.1: Online data collection instructions.



Part 2: Controlling the Players

In the game, you will control one of two chefs, either the red or blue chef.

Controls:

To move **up, down, left, or right**, use the **arrow keys**. To interact with the environment, use the **spacebar**

Arrow keys: Move your chef around. (e.g., navigate to an ingredient)

Examples:

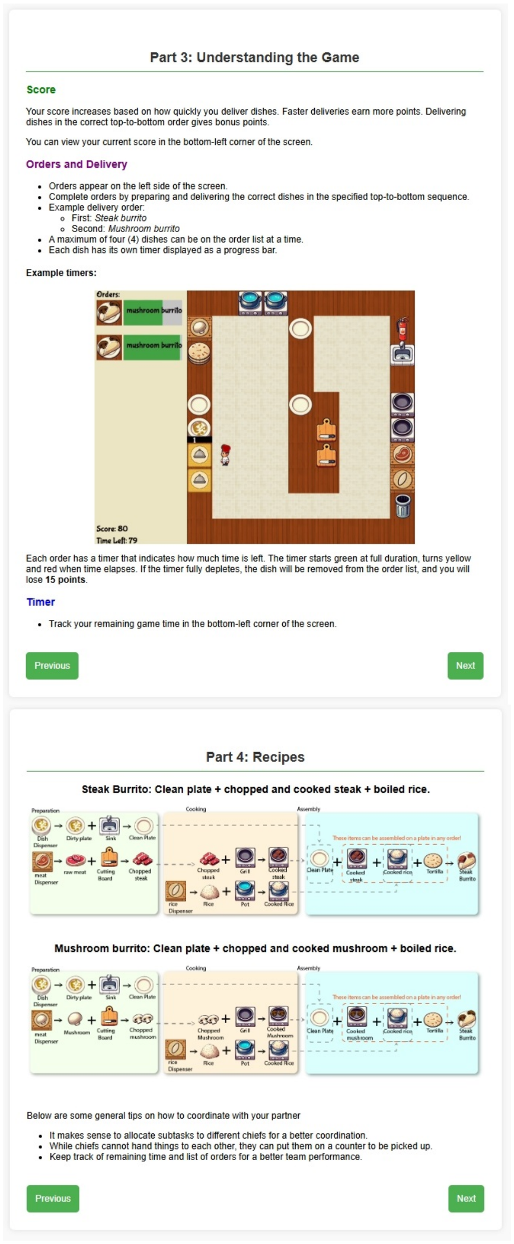
Space: Interact with objects/appliances (e.g., pick up ingredients, place dishes on counters).

You can interact with objects by facing them and pressing **spacebar**. Here are some examples:

- You can pick up ingredients (such as raw meat or a dirty plate) by facing the ingredient area and pressing the **spacebar**.
- If you are holding an ingredient, facing an empty counter, and press the **spacebar**, you will place the ingredient on the counter.
- If you are holding an ingredient, facing a grill or sink that is not full, and press the **spacebar**, you will place the ingredient on the grill or in the sink.
- If you are facing a grill or sink that is not empty, are currently holding nothing, and press the **spacebar**, you will begin grilling meat or cleaning the plate.

NOTE: As you and your teammate (another chef) are moving around the kitchen you **cannot occupy the same location. You can drop anything you are holding on a counter if needed.**

(a)



Part 3: Understanding the Game

Score

Your score increases based on how quickly you deliver dishes. Faster deliveries earn more points. Delivering dishes in the correct top-to-bottom order gives bonus points.

You can view your current score in the bottom-left corner of the screen.

Orders and Delivery

- Orders appear on the left side of the screen.
- Complete orders by preparing and delivering the correct dishes in the specified top-to-bottom sequence.
- Example delivery order:
    - First: *Steak burrito*
    - Second: *Mushroom burrito*
- A maximum of four (4) dishes can be on the order list at a time.
- Each dish has its own timer displayed as a progress bar.

Example timers:

Each order has a timer that indicates how much time is left. The timer starts green at full duration, turns yellow and red when time elapses. If the timer fully depletes, the dish will be removed from the order list, and you will lose **15 points**.

Timer

- Track your remaining game time in the bottom-left corner of the screen.

Part 4: Recipes

Steak Burrito: Clean plate + chopped and cooked steak + boiled rice.

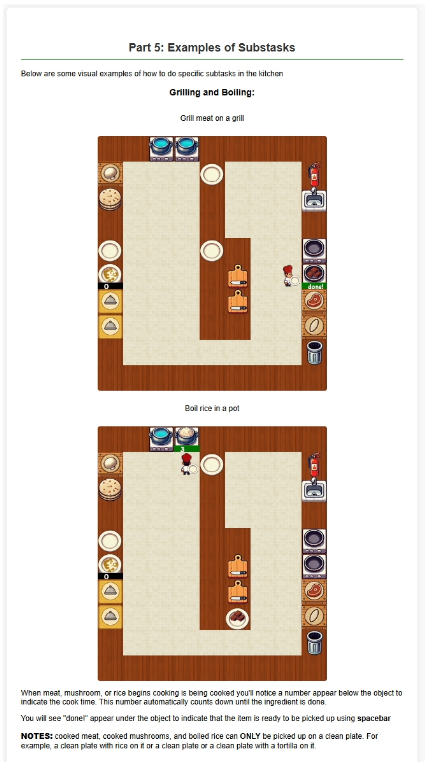
Mushroom burrito: Clean plate + chopped and cooked mushroom + boiled rice.

Below are some general tips on how to coordinate with your partner

- It makes sense to allocate subtasks to different chiefs for a better coordination.
- While chiefs cannot hand things to each other, they can put them on a counter to be picked up.
- Keep track of remaining time and list of orders for a better team performance.

(b)

Figure A.2: Online data collection instructions.



Part 5: Examples of Substasks

Below are some visual examples of how to do specific subtasks in the kitchen

Grilling and Boiling:

Grill meat on a grill

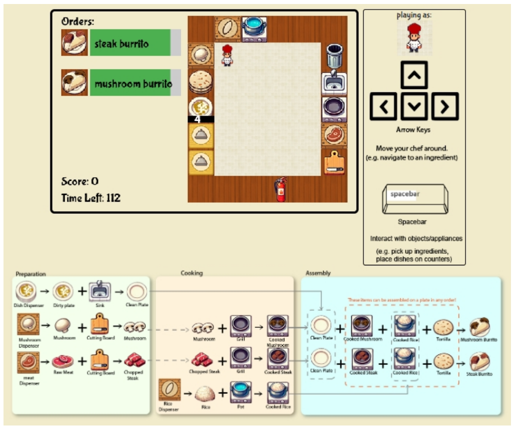
Boil rice in a pot

When meat, mushroom, or rice begins cooking is being cooked you'll notice a number appear below the object to indicate the cook time. This number automatically counts down until the ingredient is done.

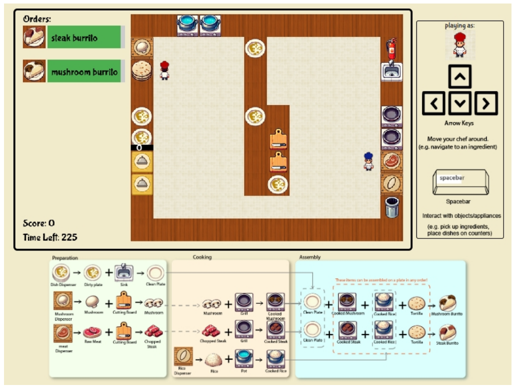
You will see "done!" appear under the object to indicate that the item is ready to be picked up using **spacebar**

NOTES: cooked meat, cooked mushrooms, and boiled rice can **ONLY** be picked up on a clean plate. For example, a clean plate with rice on it or a clean plate or a clean plate with a tortilla on it.

Figure A.3: Online data collection instructions.

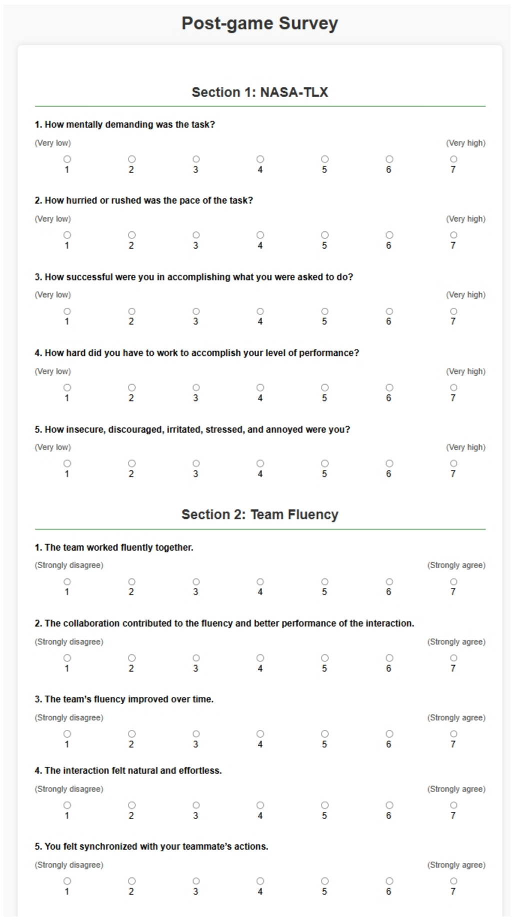


(a) Single-player tutorial session.



(b) Two-player (agent in blue, human in red) actual data collection session (hallway layout).

Figure A.4: Online data collection gameplay interfaces.



Post-game Survey

Section 1: NASA-TLX

1. How mentally demanding was the task?

(Very low) (Very high)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

2. How hurried or rushed was the pace of the task?

(Very low) (Very high)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

3. How successful were you in accomplishing what you were asked to do?

(Very low) (Very high)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

4. How hard did you have to work to accomplish your level of performance?

(Very low) (Very high)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

5. How insecure, discouraged, irritated, stressed, and annoyed were you?

(Very low) (Very high)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

Section 2: Team Fluency

1. The team worked fluently together.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

2. The collaboration contributed to the fluency and better performance of the interaction.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

3. The team's fluency improved over time.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

4. The interaction felt natural and effortless.

(Strongly disagree) (Strongly agree)

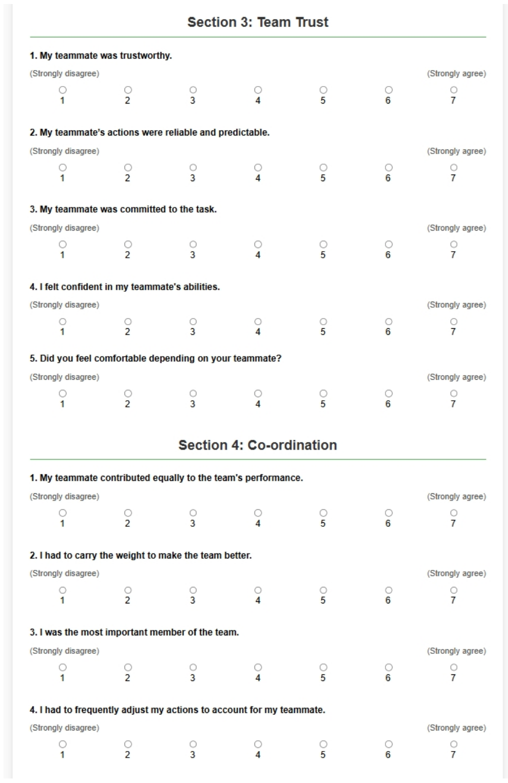
○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

5. You felt synchronized with your teammate's actions.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

(a)



Section 3: Team Trust

1. My teammate was trustworthy.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

2. My teammate's actions were reliable and predictable.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

3. My teammate was committed to the task.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

4. I felt confident in my teammate's abilities.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

5. Did you feel comfortable depending on your teammate?

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

Section 4: Co-ordination

1. My teammate contributed equally to the team's performance.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

2. I had to carry the weight to make the team better.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

3. I was the most important member of the team.

(Strongly disagree) (Strongly agree)

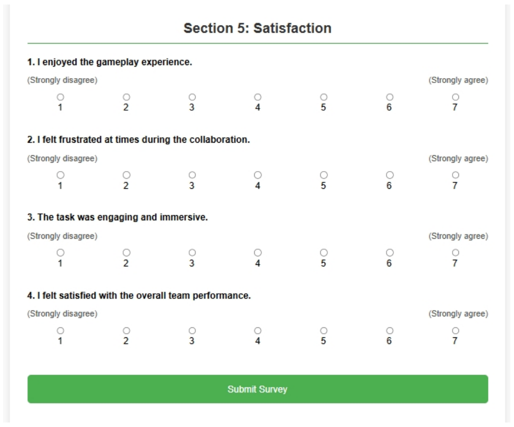
○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

4. I had to frequently adjust my actions to account for my teammate.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

(b)



Section 5: Satisfaction

1. I enjoyed the gameplay experience.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

2. I felt frustrated at times during the collaboration.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

3. The task was engaging and immersive.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

4. I felt satisfied with the overall team performance.

(Strongly disagree) (Strongly agree)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

(c)

Figure A.5: Post-game survey questions.

B Training Details

B.1 Population Training

For the Behavior Preference [44] agent population, we define a set of nine tunable preferences, separated into three distinct categories. Each preference has the option of an "encouraged" reward shaping term as well as a "discouraged" term. For each category, we train agents with every permutation of the preferences (8 agents per category, 24 agents total). A best response agent is trained in self-play with each of the behavior-preferenced agents. We use these agents as the base population in human experiments following Wang et al. [44]'s observation that these agents cover a more diverse strategy space than other population methods (MEP, FCP).

Preference	**Reward Shape**
Plating Ingredients	$-30, 20$
Washing Plates	$-30, 20$
Delivering Dishes	$-30, 20$
Chopping Ingredients	$-30, 20$
Potting Rice	$-30, 20$
Grilling Meat/Mushroom	$-30, 20$
Taking Mushroom From Dispenser	$-15, 10$
Taking Rice From Dispenser	$-15, 10$
Taking Meat From Dispenser	$-15, 10$

Table A.1: Event-based BP features and corresponding reward design.

For MEP [51] and FCP [39] agent populations, we train eight unique agents, saving two intermediate policy checkpoints ($\frac{1}{3}, \frac{2}{3}$ of total training iterations) and the final policy to form a total population of 24 policies.

B.2 Encoder Training

As mentioned in the main text, we pair agents in the population and perform several joint rollouts to form the trajectory dataset. For FCP and MEP populations, we collect trajectories from every combination of policies in the population. For the BP population, we play each preference agent with its self-play best response. In total, each dataset of trajectories contained between 500 and 700 agent games with 2400 timesteps each. Encoder training hyperparameters are listed in table A.2, and were tuned via a simple linesearch. Encoder input observations were in the form of a 26-channel egocentric lossless state representation, one of the standard options of the Overcooked domain. KL weight β was linearly annealed over the course of training.

B.3 Cooperator Training

Cooperator agents were trained via Independent PPO, though other RL methods would likely also work. Action masking was applied during both train and test time to ensure agents chose valid actions. To encode the cluster information in the TALENTS agent, an action biasing approach was used. We experimented with using an embedding layer to encode the cluster parameters and appending that to the agent's observation, but found that approach to be less effective at learning distinguishably different behaviors for different types of partners, and as a result, had worse performance. Training hyperparameters can be found in table A.3. To test the belief update and expert switching mechanism of fixed share, we tracked the distribution of weights over time when switching the cooperator's partner's policy for one with a different strategy during an episode (same premise as the ablation study shown in 4). As shown in A.6, we see the best predicted expert change over time.

Table A.2: Role Encoder Training Parameters and Model Configuration

Parameter	Default Value	Description
Model Architecture		
Latent Dimension	8	Dimension of the latent role space
Window Length	50	Length of trajectory input sequence
Prediction Horizon	50	Future action prediction steps
CNN Feature Extractor		
Input Channels	26	Channels in state representation
Conv Layers	[16, 32, 32]	Convolutional layer filter counts
Kernel Size	3×3	Convolution kernel dimensions
Activation	ReLU	Convolutional layer activation
Sequence Encoding		
Action Embedding Dim	8	Discrete action embedding size
Temporal Encoder	GRU	Sequence processing architecture
GRU Hidden Size	256	Temporal encoder hidden dimension
GRU Input Size	CNN + Action	Combined feature input
Encoder Network		
Hidden Layer Size	256	Encoder hidden dimension
Output Layer	$2\times$ Latent Dim	Mean and log-variance outputs
Encoder Activation	ReLU	Hidden layer activation
Decoder Network		
Decoder RNN	GRU	Future sequence decoder
Decoder Hidden Size	256	Decoder RNN hidden dimension
Decoder Input	Latent + CNN	Combined latent and observation features
Action Predictor	Linear	Final action classification layer
Training Hyperparameters		
Batch Size	512	Training batch size
Number of Epochs	100	Total training epochs
Learning Rate	5×10^{-4}	Adam optimizer learning rate
β Start	0.0	Initial β-VAE regularization weight
β End	0.05	Final β-VAE regularization weight
Train/Validation Split	80/20	Data split ratio (%)

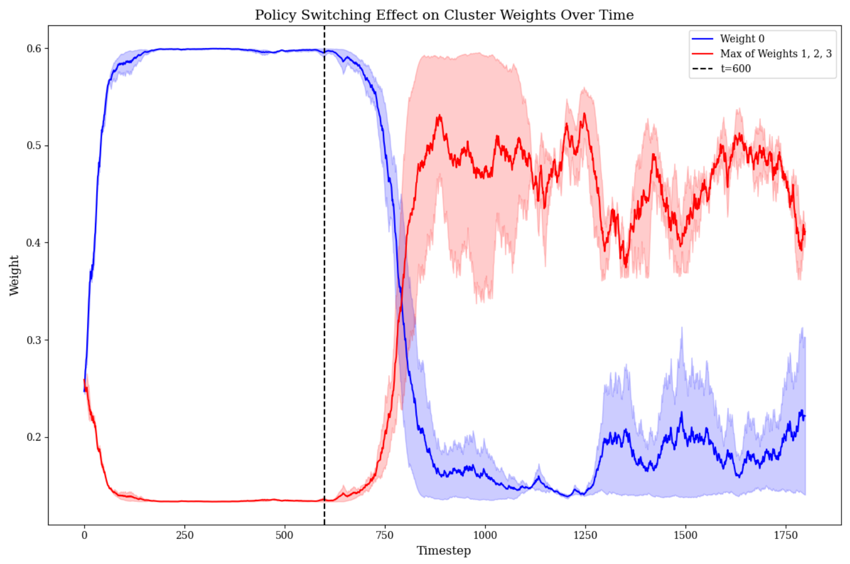


Figure A.6: Cluster Weight Distribution With a Partner Policy Swap Over Time

Table A.3: Cooperator Policy Training Parameters

Parameter	Value
Optimization Parameters	
Total Training Steps	4e7
Training Batch Size	38,400
Parallel Environments	16
Learning Rate Schedule	$[10^{-3}, 10^{-4}, 10^{-5}]$
GAE Lambda (λ)	0.99
KL Coefficient	0.5
PPO Clip Parameter	0.2
Value Function Loss Coefficient	0.5
Entropy Coefficient	0.02
Gradient Clipping	1.0
Discount Factor (γ)	0.995
Fixed Share Parameters	
Learning Rate	0.2
Sharing Parameter (α)	0.4
Regret Clipping	1.0
Network Architecture	
Actor Embedding Dimension	392
Critic Embedding Dimension	392
Cluster Embedding Dimension	32
Action Space Size	27
Observation Channels	12
Action Bias Weight	2.0

C Additional Experiments

C.1 Additional Baselines

The primary motivation of this work was to develop a state-of-the-art method for training a cooperator using simulated data generated by agent populations. As such, the baseline algorithms used for evaluation (GAMMA, BR) were selected as they are representative SOTA methods for population-based training, and enable us to isolate the impact of our novel training and adaptation methodology. However, it is still valuable to compare against other leading zero-shot coordination methods, notably ones which also utilize strategy inference or partner modeling modules.

We evaluate against E3T, a self-play method that uses a partner modeling module to anticipate its teammate's future intentions, and PACE, an adaptation framework that injects a peer identification reward term to encourage agents to choose actions which allow them to form more accurate estimates about their teammates' types. We report the results of preliminary experiments in A.4.

We find that TALENTS remains the top performer out of all the assessed methods, while E3T is able to outperform the remaining baselines. PACE underperforms, achieving comparable scores to the population best response baseline. This can likely be attributed to the fact that PACE is an effective *inter*-episodic adaptation framework. However, in our zero-shot coordination problem setting, PACE is unable to form its belief in the short time frame necessary for high performance, and therefore achieves a similar score as the naive population best response baseline. In contrast, TALENTS is able to perform *intra*-episodic adaptation due to its tracking regret minimization module.

Algorithm	Score (± Std. Err.)
E3T	548.66 ± 25.44
PACE	313.34 ± 18.48
GAMMA	374.10 ± 24.99
BR	343.71 ± 24.27
TALENTS	605.88 ± 27.89

Table A.4: Average performance over the evaluated layouts, comparing the additional baselines of E3T and PACE.

C.2 Effect of Cluster Count

We use silhouette analysis in order to partition the VAE latent space into discrete strategy clusters in an unsupervised manner. However, there is no theoretical guarantee that the optimal cluster count as determined by silhouette analysis will result in optimal performance when those clusters are used to train a TALENTS cooperator. To empirically evaluate our approach, we conduct a sensitivity analysis over cluster count, training and evaluating cooperator agents using various numbers of latent clusters. The results of the sensitivity analysis are shown in A.5, while the corresponding silhouette scores are shown in A.7. We find that the silhouette-optimal number of clusters achieves the highest score, suggesting that silhouette scores are an appropriate metric to use for this application.

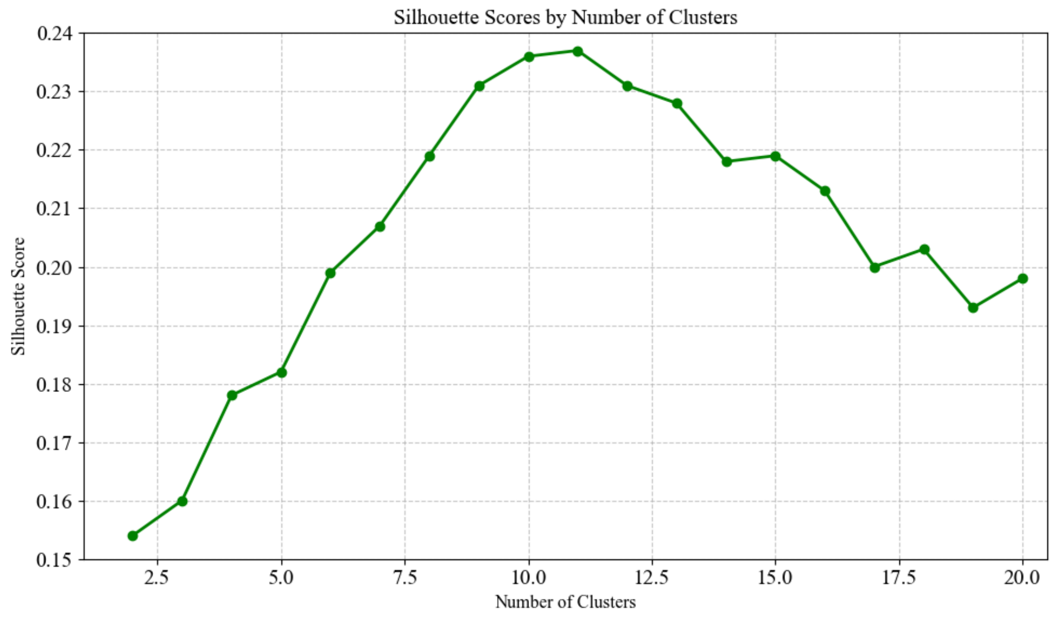


Figure A.7: Silhouette analysis of a VAE latent space in the open layout. Higher silhouette score indicates more optimal clustering, with 11 being the optimal number for this latent space.

Number of Clusters	Score (± Std. Err.)
1	650.83 ± 28.00
2	701.68 ± 11.93
4	620.35 ± 39.01
8	740.93 ± 79.15
11	865.82 ± 15.11
14	724.94 ± 40.60
16	648.57 ± 26.26

Table A.5: Effect of different numbers of clusters on overall performance of a TALENTS cooperator, evaluated in the open layout.

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: We make the following claims:

   - We develop a new method that adapts to agent teammates in Overcooked and achieves SOTA performance. This is shown in Table 1 and holds true for three out of four tested environments.
   - We claim that our agent can successfully adapt to human players, despite being trained only on agent-agent data. This is shown in our human subject study shown in Figure 5, demonstrating that our agent performs the best (score wise) and is preferred by users (according to the four NASA TLX scores)

   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: We discuss our limitations in Section 6.1. Fundamentally, our approach is limited by the generalization capability of the VAE. Furthermore, additional investigations into the coordiation failure in the "Forced Coordination" layout is required.

   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 provide any theoretical results or proofs.

   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: The answer to this question nuanced, but largely [Yes] :

   - We provide the code for our proposed agent, as well as instructions of how to re-train our TALENTS agent
   - We do not release our offline dataset to train our VAE, however, our code outlines how it can be regenerated
   - Due to anonymity and size constraints, we can not provide a pretrained model, however, instructions in the code should be sufficient to reproduce our agents
   - Due to IRB constraints, we are unable to provide the collected human-agent games

   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: This work does not provide a dataset. The dataset used to train our VAE can be re-generated using our provided code. The anonymously released code for review contains instructions on how to re-run our experiments.

   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: Data splits, hyperparameters, and training details are disclosed in the supplemental material and anonymous code release.

   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: We provide error bars and evaluations with respect to the statistical significance of our results in Table 1, Figure 4, and Figure 5.

   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: We provide details of the utilized compute hardware in section 6.1

   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 human-subject study is IRB approved and conforms to the NeurIPS Code of Ethics. We obtained consent and compensated participatns for their time and effort.

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: The research conducted in this work is fundamental research that has no direct societal impact, positive or negative.

    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: This work does not provide a high-risk model, nor is any data released.

    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 are the original owners of our method, TALENTS, and have cited the modified Overcooked [**?**] benchmark utilized in this work.

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: [Yes]

    Justification: We anonymously release our code for review (see abstract for anonymous GitHub code).

    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: [Yes]

    Justification: We have provided details about the human subject study conducted in this work in Section 5.2. The conducted surveys are provided in Section 5.2.1.

    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: [Yes]

    Justification: The human experiments conducted in support of Section **??** are IRB approved and follow the NeurIPS Ethics Code.

    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: LLMs have been utilized for light editing and formatting.

    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.