TransAM: Transformer-Based Agent Modeling for Multi-Agent Systems via Local Trajectory Encoding

Anonymous authors
Paper under double-blind review

Keywords: Multi-Agent Systems, Agent Modeling, Transformer Networks, Policy Representation, Adaptive Learning.

Summary

Agent modeling is a critical component in developing effective policies within multi-agent systems, as it enables agents to form beliefs about the behaviors, intentions, and competencies of others. Many existing approaches assume access to other agents' episodic trajectories, a condition often unrealistic in real-world applications. Consequently, a practical agent modeling approach must learn a robust representation of the policies of the other agents based only on the local trajectory of the controlled agent. In this paper, we propose `TransAM`, a novel transformer-based agent modeling approach to encode local trajectories into an embedding space that effectively captures the policies of other agents. We evaluate the performance of the proposed method in cooperative, competitive, and mixed multi-agent environments. Extensive experimental results demonstrate that our approach generates strong policy representations, improves agent modeling, and leads to higher episodic returns.

Contribution(s)

1. We eliminate the need for agent information at inference by learning a latent representation that approximates the agent policy based only on local information.
   Context: It is common for methods to assume access to other agent information at execution time (He & Boyd-Graber, 2016; Grover et al., 2018; Jing et al., 2024).
2. By representing the controlled agent's local trajectory as a sequence, we extract more meaningful features over a time horizon. The self-attention mechanism allows the model to pinpoint which parts of the local trajectory are most relevant to the agent's policy.
   Context: Other methods typically construct either an MLP-based agent model (He & Boyd-Graber, 2016), or a recurrent agent model (Papoudakis et al., 2021) which do not take into account the full context of the agent's trajectory throughout the episode.
3. To address the data demands of transformers, we train the agent model and the controlled agent's policy jointly in an online setting, ensuring access to a diverse dataset for enhanced performance.
   Context: Other promising transformer-based agent modeling approaches such as Jing et al. (2024) are based in an offline reinforcement learning setting wherin a pretraining phase is used to learn an initial prior for the task. In contrast, we aim to train the agent model and the policy jointly from scratch.

TransAM: Transformer-Based Agent Modeling for Multi-Agent Systems via Local Trajectory Encoding

Anonymous authors
Paper under double-blind review

Abstract

Agent modeling is a critical component in developing effective policies within multi-agent systems, as it enables agents to form beliefs about the behaviors, intentions, and competencies of others. Many existing approaches assume access to other agents' episodic trajectories, a condition often unrealistic in real-world applications. Consequently, a practical agent modeling approach must learn a robust representation of the policies of the other agents based only on the local trajectory of the controlled agent. In this paper, we propose `TransAM`, a novel transformer-based agent modeling approach to encode local trajectories into an embedding space that effectively captures the policies of other agents. We evaluate the performance of the proposed method in cooperative, competitive, and mixed multi-agent environments. Extensive experimental results demonstrate that our approach generates strong policy representations, improves agent modeling, and leads to higher episodic returns.

1 Introduction

Recent advances in multi-agent systems have led to significant progress in applications such as games (Nowé et al., 2012), traffic control (Wiering et al., 2000), and autonomous driving (Cao et al., 2012). A key challenge in these systems is that the collective actions of all agents influence the overall system's transitions. Therefore, effectively reasoning about the optimal actions requires modeling the behavior of other agents. This process, known as agent modeling, focuses on inferring concealed information about other agents to inform the policy of a controlled agent. In this work, we explore the importance of agent modeling in multi-agent systems and its impact on decision-making strategies.

A primary challenge in agent modeling arises from the need to design agents that can adapt to various agent policies using only the information available during execution. This challenge becomes particularly difficult in scenarios where no direct information about the other agents is accessible, requiring the agent to infer others' behaviors based solely on its own local information. Moreover, since agent policies may appear indistinguishable on the basis of a single transition, it is essential to consider the temporal context for disambiguation. Therefore, an effective agent modeling approach must learn robust representations of agent policies while accounting for their temporal dynamics and long-term effects.

Although recent advances in deep learning have led to various approaches for agent modeling (He & Boyd-Graber, 2016; Grover et al., 2018; Papoudakis et al., 2021; Jing et al., 2024), existing methods often face two key limitations: (1) reliance on access to agent trajectories and (2) inadequate use of the sequence of actions of the controlled agent as a valuable source of information. Inspired by the success of decision transformers (Chen et al., 2021) and their multi-agent variants (Wen et al., 2022), we propose reframing agent modeling as a sequence modeling task using a transformer architecture.

Transformers have recently been applied in reinforcement learning (RL) and demonstrated remarkable success, from feature extraction to end-to-end policy learning (Agarwal et al., 2023). Building on this progress, we propose a transformer-based agent modeling approach that encodes the controlled agent's local trajectory into an embedding space that captures the influence of other agent policies. The model is trained to reconstruct the other agents' trajectories using only the local trajectory embedding, enabling the controlled agent to model others without requiring access to their trajectories at execution. This allows the RL policy to condition its decisions solely on the local trajectory embeddings.

Our contributions are as follows.

1. **Agent Modeling from Local Information:** We eliminate the need for agent information at inference by learning a latent representation that approximates the agent policy based only on local information.
2. **Local Trajectory as a Sequence Modeling Task:** By representing the local trajectory of the controlled agent as a sequence, we extract more meaningful features over a time horizon. The self-attention mechanism allows the model to pinpoint which parts of the local trajectory are most relevant to the agent's policy.
3. **Online Joint Training of Agent Model and Policy:** To address the data demands of transformers, we train the agent model and the controlled agent's policy jointly in an online setting, ensuring access to a diverse dataset for enhanced performance.

We evaluate the proposed approach on cooperative, competitive, and mixed cooperative-competitive multi-agent RL tasks. Our results demonstrate that the proposed method outperforms baseline approaches in agent modeling accuracy, provides robust agent policy representation, and achieves superior episodic returns.

2 Related Work

2.1 Agent Modeling

When operating in a decentralized multi-agent system, it is important to incorporate information about other agents to determine the best response to a given state. In conventional centralized training with decentralized execution (CTDE) approaches, such as MADDPG (Lowe et al., 2017) and MAPPO (Yu et al., 2022), a centralized critic is trained using the joint observations of all agents, and this information is implicitly distilled into the actor policy. Agent modeling is an alternative approach that explicitly learns to model concealed agent information. There is a large body of work on agent modeling in multi-agent settings (Albrecht & Stone, 2018). He & Boyd-Graber (2016) focused on competitive settings and learned to predict opponent Q values and opponent actions given opponent observations. Raileanu et al. (2018) introduced a model that learns to infer the opponent's goal using itself. Grover et al. (2018) implemented a general purpose encoder-decoder architecture using imitation learning and a contrastive triplet loss to both learn to accurately reconstruct agent policies and correctly identify the agent policy within the embedding space. Building on the work of Grover et al. (2018), Papoudakis et al. (2021) also used an encoder-decoder architecture to reconstruct agent policies. However, they model this reconstruction using the controlled agent's local trajectory only. Zhang et al. (2023) introduced an approach that adapts to changing policies, similar to our problem setting. However, agents in this work can change policies within an episode, so the model must learn to quickly adapt. Xing et al. (2023) studied ad hoc teamwork in which an agent must learn to cooperate with other agents who may switch to different goal-oriented policies. In this work, the agent learns both to identify the type of policy of its teammates and to generalize the types of policies to unseen sets of teammates. Finally, Ma et al. (2024) learned an agent policy representation directly from the controlled agent's local observations using contrastive learning.

2.2 Transformers in RL

Transformers were originally intended as replacements for RNNs in machine translation language modeling tasks (Vaswani et al., 2017). However, they have been applied to seemingly every subfield of machine learning, including computer vision Dosovitskiy et al. (2021) and more recently for reinforcement learning (Agarwal et al., 2023). The original transformer model consists of an encoder that maps an input sequence to a latent space and a decoder that generates an output sequence conditioned on the input sequence and the latent embeddings of the input sequence. Reinforcement learning problems have incorporated both parts of the transformer model to pose the problem in different terms. Parisotto et al. (2020) used a modified encoder architecture as a replacement for RNNs in RL policies. Alternatively, Chen et al. (2021) proposed offline RL as a generative sequence modeling task using a GPT-style decoder architecture (Radford et al., 2018). More recently, multi-agent reinforcement learning has been reimagined as a sequence-to-sequence task (Wen et al., 2022) where the model maps input sequences of observations to output sequences of actions. Similarly to our problem setting, Jing et al. (2024) introduced a transformer architecture to learn opponent policy representations from offline datasets. In this paper, we are interested in learning latent representations of the other agents' policies as a function of the controlled agent's local trajectory.

3 Background

3.1 Partially Observable Stochastic Games

Partially observable stochastic games (POSGs) (Hansen et al., 2004) are a common formulation for multi-agent settings. They are described by a set of agents $i \in \{0, \ldots, N\}$ and a finite set of states $s \in \mathcal{S}$. For each agent i, there is a finite action space $\mathcal{A}^i$ where $\mathcal{A} = \mathcal{A}^0 \times \ldots \times \mathcal{A}^N$ represents the joint action space of all agents. Similarly, for each agent i, there is a finite observation space $\mathcal{O}^i$, where $\mathcal{O} = \mathcal{O}^0 \times \ldots \times \mathcal{O}^N$ is the joint observation space of all agents. In addition to the observation space, an agent has an observation function O^i: $\mathcal{A} \times \mathcal{S} \times \mathcal{O}^i \rightarrow [0, 1]$ given by 1

$$\forall a \in \mathcal{A}, \forall s \in \mathcal{S} : \sum_{o^i \in \mathcal{O}^i} O(a, s, o^i) = 1. \tag{1}$$

In addition to the action and observation spaces, each agent has a reward function $\mathcal{R}^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. Finally, similar to the observation function, the game has a state transition probability function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ given by 2

$$\forall a \in \mathcal{A}, \forall s \in \mathcal{S} : \sum_{s' \in \mathcal{S}} P(s, a, s') = 1, \tag{2}$$

where s' is the next state as a result of taking the joint action a in the previous state s.

Agent i selects an action $a^i \in \mathcal{A}^i$ given an observation $o^i \in \mathcal{O}^i$ according to a policy $\pi^i(a^i|o^i)$, which is a probability distribution over the set of actions $\mathcal{A}^i$. The goal of an agent is to learn a policy π such that the expected cumulative reward, or the agent's return, is maximized:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=1}^{L} \gamma^t r_{t+1} \mid \pi\right] \tag{3}$$

where L is the length of the episode and $\gamma \in [0, 1)$ is the discount factor. The action value function $Q^{\pi^i}(s, a^i)$ for agent i defines the expectation of the return given the state s when taking action a^i following policy π^i. Similarly, the value function $V^{\pi^i}(s)$ describes the value of being in state s for agent i following policy π^i. In actor-critic methods, such as A2C (Mnih et al., 2016), the actor π^i and the critic $V^{\pi^i}(s)$ are used to calculate the advantage function $A^{\pi^i}(s, a^i) = Q^{\pi^i}(s, a^i) - V^{\pi^i}(s)$.

3.2 Transformers

Transformers consist of an encoder and a decoder and can use either the encoder, the decoder, or both depending on the applications. Generalizing, encoder-decoder models are used for machine translation tasks (Raffel et al., 2020). Decoder-only models are useful for generative sequence tasks

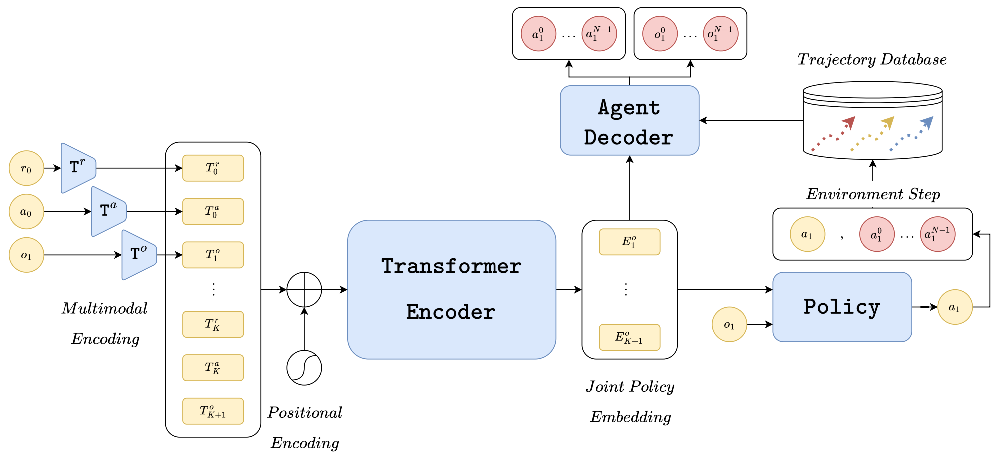


Figure 1: **`TransAM` architecture.** We embed the controlled agent's previous reward, previous action, and current observation into embedding tokens, $T_t^{(r,a,o)}$, and transform them into an output sequence of embedding vectors, $E_t^{(r,a,o)}$. The embedding vectors are used to both condition the controlled agent's policy and reconstruct the other agents' trajectories as a function of the local trajectory only.

(Radford et al., 2018). Encoder-only models are good for sequence understanding tasks (Devlin et al., 2019). We make use of an encoder-only model for our problem, and hence will focus on this portion of the model. The encoder takes as input a sequence of embedding tokens $\{T_t, \dots, T_{t+K}\}$ with context length K and transforms them into representation embedding vectors $\{E_t, \dots, E_{t+K}\}$. The model is composed of several layers of transformer blocks. Each block contains a multi-head self-attention layer and a feed-forward layer, connected by a residual connection with layer normalization at the output of the block. The self-attention function below uses three linear layers to map the input sequence of the i^{th} block into query $\mathcal{Q}_i$, key $\mathcal{K}_i$, and value $\mathcal{V}_i$ matrices which are used to create the output as follows

$$\mathcal{Z}_i = \text{softmax}\left(\frac{\mathcal{Q}_i \mathcal{K}_i^T}{\sqrt{d_k}}\right)\mathcal{V}_i, \tag{4}$$

where d_k is the dimension of the input token vectors. By combining the input tokens into sequence matrices $\mathcal{Q}$, $\mathcal{K}$, and $\mathcal{V}$ the self-attention function attends to the whole sequence, allowing the model to extract relevant information throughout the sequence.

3.3 Problem Formulation

We consider a modified POSG with one learning agent under our control and a set of agents to interact with, which can utilize one of several fixed policies. To be specific, we assume that each individual agent i adopts a policy π^i, whose collection forms the joint agent policy π^{-1}. In this work, we consider the set of M joint policies $\Pi = \{\pi^{-1,m} | m = 1, \dots, M\}$ that can be a combination of heuristic or pretrained RL policies. For simplicity, from now on we refer to the controlled agent without superscript and all other agents with superscript -1. Thus, the agent has an action space $\mathcal{A}$ and an observation space $\mathcal{O}$. Similarly, the other agents have a joint action space $\mathcal{A}^{-1}$ and a joint observation space $\mathcal{O}^{-1}$. Our objective is to learn a policy π_θ parameterized by θ such that the average return is maximized across the set of agent policies Π. The objective in Equation (3) is thus modified as

$$\arg\max_{\theta} \mathbb{E}_{\pi_\theta, \pi^{-1,m} \sim \mathcal{U}(\Pi)} \left[\sum_{t=1}^{L} \gamma^t r_{t+1}\right], \tag{5}$$

where $\pi^{-1,m}$ is uniformly sampled from Π at the beginning of each episode. The agent policy type m is concealed from the controlled agent throughout the episode. This occluded information can either be incorporated into the policy implicitly by simply attempting to maximize the average return for all agent policies, or it can be modeled explicitly and used to condition the policy on which policy m is currently being modeled. In this work, we focus on the latter and introduce a transformer-based approach to modeling such agent policies.

4 Method

4.1 TransAM

We format agent modeling as a sequence modeling task through the lens of episodic trajectories. Consider the tuple (r_{t-1}, a_{t-1}, o_t) where $r_{t-1} \sim \mathcal{R}$ is the previous reward, $a_{t-1} \sim \mathcal{A}$ is the previous action, and $o_t \sim \mathcal{O}$ is the current observation of the controlled agent. The local episodic trajectory of the agent can be viewed as a sequence of these tuples $\mathcal{T} = (r_0, a_0, o_1, \ldots, r_{L-1}, a_{L-1}, o_L)$. Similarly, the other agent trajectories are represented as $\mathcal{T}^{i,m} = (r_0^{i,m}, a_0^{i,m}, o_1^{i,m}, \ldots, r_{L-1}^{i,m}, a_{L-1}^{i,m}, o_L^{i,m})$. Our goal in agent modeling is to learn a representation of the joint agent policy $\pi^{-1,m}$ such that this representation can be used as an inductive bias for the controlled agent policy. Inspired by the recent success of transformers in such problems, we built a transformer encoder model, which we refer to as Transformer-based Agent Modeling (`TransAM`), to encode these sequences into a compact representation. Our proposed architecture can be seen in Figure 1.

We learn a linear mapping from r_t, a_t, o_{t+1} to token embeddings T_t^r, T_t^a, and T_{t+1}^o, respectively. Considering the three modalities, we use a context window of $3K$ tokens as a subset of the agent's local trajectory $\mathcal{T}_{t+K} = (T_{t-1}^r, T_{t-1}^a, T_t^o, \ldots, T_{t+K-1}^r, T_{t+K-1}^a, T_{t+K}^o)$. Using the encoder, we encode this token sequence into a representation embedding sequence $\mathcal{E}_{t+K} = (E_{t-1}^r, E_{t-1}^a, E_t^o, \ldots, E_{t+K-1}^r, E_{t+K-1}^a, E_{t+K}^o)$. Empirically, we find that the reward and action output embeddings do not provide much benefit. Therefore, we only use observation embeddings E_{t+K}^o for downstream tasks. This embedding vector E_{t+K}^o, in addition to observation o_{t+K}, is used to condition the policy $\pi_\theta(a_{t+K} | o_{t+K}, E_{t+K}^o)$. We posit that this incorporation of information is necessary for the agent policy to accurately determine the best response to the current joint agent policy.

Generative Loss To learn an informative representation of the joint agent policy, we introduce an agent trajectory reconstruction head. It decodes the embedding vector E_t^o into the joint observations $o_t^{-1,m} = (o_t^{0,m}, \ldots, o_t^{N-1,m})$ and actions $(a_t^{0,m}, \ldots, a_t^{N-1,m})$ of the other agents. We use the mean squared error loss, $\mathcal{L}_{MSE}$, to learn the observations of the agent and the mean cross-entropy loss $\mathcal{L}_{CE}$ for all actions of the agents $N-1$. In total, the agent modeling loss is given by 6

$$\mathcal{L}_{AM} = \mathcal{L}_{MSE}(\hat{o}_t^{-1,m}, o_t^{-1,m}) + \frac{1}{N-1} \sum_{i=0}^{N-1} \mathcal{L}_{CE}(\hat{a}_t^{i,m}, a_t^{i,m}), \tag{6}$$

where $\hat{o}_t^{-1,m}$ is the predicted joint agent observation and $\hat{a}_t^{i,m}$ is the predicted agent action for agent i. The reconstruction head is only used during training to learn the representation E_t^o. During execution, we only use the encoder, which does not need access to the occluded information of other agents.

4.2 Policy Training

The goal of the controlled agent is to learn a policy that adapts to different joint agent policies $\pi^{-1,m}$. We train `TransAM` such that the embedding vector E_t^o is a good proxy for the true other agent information. By incorporating this vector into the controlled agent policy, it allows the policy to better adapt to varying joint agent policies. From here, any RL algorithm can be used to learn an optimal policy π conditioned on o_t and E_t^o. In this paper, we use the advantage actor-critic (A2C)

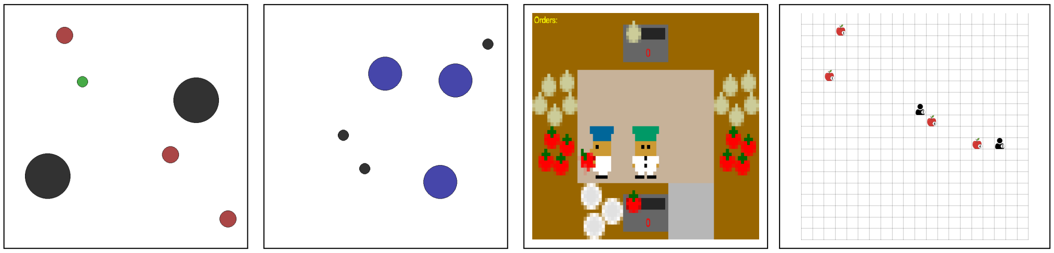


Figure 2: **Experimental environments.** We use four environments (a) Predator-Prey, a competitive pursuit environment (b) Cooperative Navigation, a cooperative navigation environment (c) Overcooked, a cooperative cooking environment (d) Level-Based Foraging a mixed resource allocation environment.

algorithm (Mnih et al., 2016). Thus, the RL objective is given by 7

$$\begin{aligned}\mathcal{L}_{A2C} =& \mathbb{E}_{(o_t,a_t,o_{t+1},r_{t+1})\sim B}[\frac{1}{2}\left(r_{t+1} + V_\phi(o_{t+1}, E^o_{t+1}) - V_\phi(o_t, E^o_t)\right)^2 \\ &- A^\pi(o_t, a_t)\log \pi_\theta(a_t|o_t, E^o_t) - \beta H(\pi_\theta(a_t|o_t, E^o_t))],\end{aligned} \tag{7}$$

where B is a batch of transitions, π_θ is the policy parameterized by θ, V_ϕ is the value function parameterized by ϕ, A^π is the advantage function under policy π, and H is the entropy function weighted by the entropy coefficient β. We optimize (6) and (7) jointly, sampling the set of other agent policies per episode.

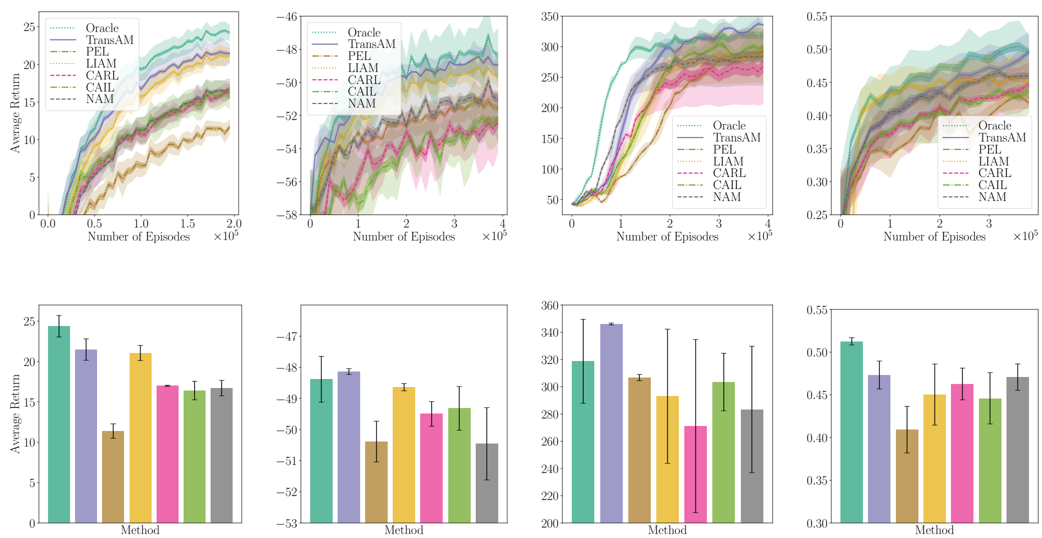
5 Experiments

5.1 Experimental Setup

To validate the effectiveness of our proposed approach, we performed experiments in a variety of settings, including competitive, cooperative, and mixed environments. Specifically, we used Multi-Agent Particle Environments (MPEs) from (Mordatch & Abbeel, 2017) that contain competitive and cooperative scenarios, the cooperative Overcooked environment (Carroll et al., 2019), and the mixed level-based foraging environment (Christianos et al., 2020). Each experiment presents a unique scenario where cooperativeness, competitiveness, or a mixture of both plays a vital role and must be modeled appropriately. Through rigorous analysis, we assessed the performance of our approach in terms of modeling agent behavior and solving the final task. In all of our experiments, we relied on the Advantage Actor-Critic (A2C) algorithm (Mnih et al., 2016) and used one LSTM layer (Hochreiter & Schmidhuber, 1997) and one linear layer, both with a hidden dimension of 128. Furthermore, we used a transformer encoder that is made up of four transformer blocks with four attention heads and a hidden dimension of 128. We trained the controlled agent policy for 10 million time steps and performed evaluations every 100 episodes. To ensure the reproducibility of the results, we performed five different training runs with different random seeds and plotted the average of the results to provide reliable evidence of our approach's performance.

We compare our proposed method with several key baselines that represent a range of solutions in this space. Some baselines employ an explicit agent model, while others are implicit. These baselines can be categorized based on the amount of information available to the controlled agent about the other agents:

- **No Agent Modeling (NAM):** This baseline only has access to the controlled agent's current observation and last action.
- **Contrastive Agent Representation Learning (CARL):** This baseline employs a recurrent encoder to embed the local information of the controlled agent into a vector space representing the



(a) Tag returns. (b) Spread returns. (c) Overcooked returns. (d) LBF returns.

Figure 3: **Average task returns.** (Top) Average episodic returns during training with 95% confidence intervals across four experimental scenarios, evaluated over five random seeds. (Bottom) Average episodic returns over 100 evaluation episodes, also averaged across five random seeds.

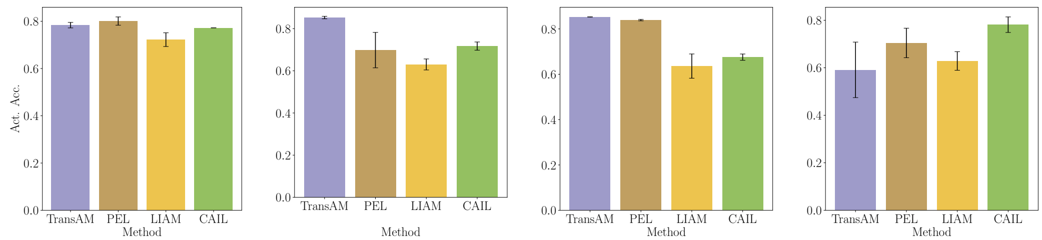
joint policy. The encoder is trained using contrastive loss, specifically InfoNCE (Chen et al., 2020).

- **Conditional Agent Imitation Learning (CAIL):** This baseline uses a recurrent backbone to embed local information into a vector space, which is then used to condition a policy imitation decoder.
- **Local Information Agent Modeling (LIAM):** This baseline from Papoudakis et al. (2021) employs a recurrent encoder-decoder architecture to encode the controlled agent's local information into an embedding space. The decoder reconstructs other agents' observations and actions, but only the encoder is used during inference, restricting access to the controlled agent's information.
- **Policy Embedding Learning (PEL):** Originally proposed in Jing et al. (2024), this approach uses a transformer-based architecture to encode an opponent's trajectory into a policy embedding space. It employs a generative loss for action reconstruction via conditional imitation learning and a contrastive InfoNCE loss to differentiate policies. We adapt this by encoding only the controlled agent's trajectory.
- **Oracle:** This baseline assumes full access to other agents' trajectories, including observations and actions. The controlled agent conditions on a joint vector comprising its local observation, last action, and other agents' observations and actions. With no ambiguity in the intentions or strategies of the agents, this represents an upper performance baseline.

5.2 Experimental Environments

5.2.1 Predator-Prey (Tag)

We use a modified predator-prey environment from Boehmer et al. (2020), featuring two large landmarks, three adversarial predator agents, and one controlled prey agent. The prey is faster, providing a strategic advantage. In this setup, the prey receives a reward of $+1$ if caught by a single adversary, while all adversaries receive -1. If multiple adversaries capture the prey, the prey receives -1 and the adversaries receive $+1$. In addition, the agent incurs a penalty -10 for reaching the boundary of the environment.



(a) Tag. (b) Spread. (c) Overcooked. (d) LBF.

Figure 4: **Agent action reconstruction accuracy.** We compute the agent action reconstruction accuracy for the relevant methods for all four environments averaged across five random seeds.

5.2.2 Cooperative Navigation (Spread)

We use the original cooperative navigation scenario from Mordatch & Abbeel (2017), where three agents and three landmarks start from random positions. Agents must coordinate to cover all landmarks while avoiding collisions. The team's reward is based on the sum of the minimum distances between agents and landmarks, with penalties for collisions.

5.2.3 Overcooked

We utilize the cramped room layout from the simplified Overcooked environment (Carroll et al., 2019), where two chefs collaborate in a confined kitchen to prepare and serve onion soup. The task requires executing a sequence of high-level actions, including placing onions in a pot (cooking for 20 timesteps), transferring soup to bowls, and serving. Each served soup grants both agents a reward of 20, with the objective of maximizing the number of soups served within 400 timesteps. Efficient coordination and multitasking are essential for optimal performance.

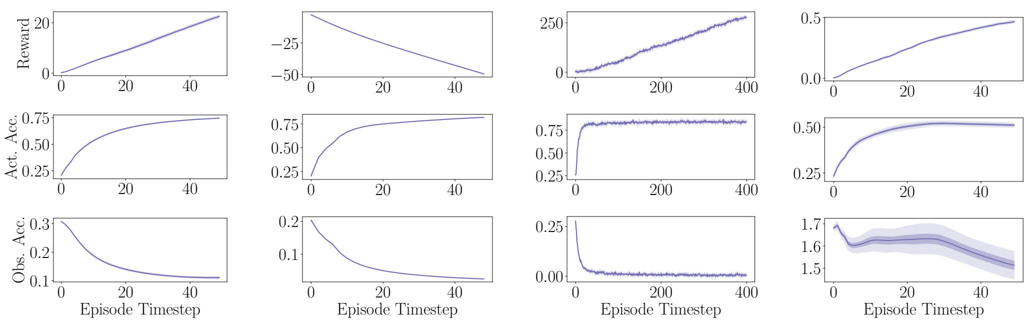
5.2.4 Level-Based Foraging

This scenario features a 20×20 gridworld with two agents and four food locations, each assigned a skill level. An agent can capture food if its skill level exceeds that of the food, and agents can also combine skill levels to capture higher-level food. This creates a mixed cooperative-competitive dynamic, where agents may collaborate for higher rewards or act independently for easier gains. Rewards are distributed based on each agent's contribution to the total captured food. For instance, if one agent captures food of level 1 while the other captures levels 2, 3, and 4, their rewards are proportionally $1/(1+2+3+4)$ and $(2+3+4)/(1+2+3+4)$, respectively.

5.3 Analysis

5.3.1 Task Returns

The average evaluation returns are presented in 3. As expected, Oracle consistently establishes an upper performance baseline. Notably, `TransAM` matches or surpasses Oracle across all environments, while LIAM performs comparably but slightly worse. Both `TransAM` and LIAM achieve higher returns than other baselines, likely due to their ability to encode agent actions and observations, resulting in a more informative policy embedding space. NAM consistently achieves moderate to low returns as it lacks an auxiliary learning objective to enhance performance. CAIL struggles to outperform NAM in predator-prey and level-based foraging but performs well in cooperative navigation and Overcooked, suggesting that reconstructing agent policies is particularly beneficial in cooperative settings. CARL demonstrates moderate performance across all environments, excelling in those with competitive dynamics. PEL yields the lowest returns in three of four environments,



(a) Tag returns. (b) Spread returns. (c) Overcooked returns. (d) LBF returns.

Figure 5: **Evolution of `TransAM` performance across an episode.** We analyze the relationship between cumulative reward (top), agent action reconstruction accuracy (middle), and agent observation reconstruction accuracy (bottom) throughout an episode, averaged over 100 episodes.

indicating that the combination of generative and contrastive losses negatively impacts the final task performance.

5.3.2 Agent Modeling

The agent modeling results for methods with action reconstruction capabilities are shown in 4. `TransAM` consistently excels in reconstructing agent actions, outperforming all baselines in the two cooperative tasks, achieving competitive accuracy in the competitive task, but underperforming in the mixed setting. PEL matches or surpasses `TransAM` in three of four tasks, while CAIL performs comparably but struggles in cooperative environments. Both PEL and CAIL incorporate an imitation learning objective, with PEL additionally using a contrastive loss to better distinguish agent policies. However, this improved agent modeling performance comes at the cost of final task returns, suggesting a trade-off between policy reconstruction and maximizing the controlled agent's reward. This trade-off is evident in LIAM, which lags behind other baselines in agent modeling but achieves significantly higher returns than PEL and CAIL. `TransAM` effectively balances both objectives, demonstrating competitive agent modeling while achieving the highest returns. Notably, `TransAM` is particularly well suited for strictly cooperative settings, where superior agent modeling performance strongly correlates with high average returns, even surpassing the Oracle agent in some cases.

5.3.3 Model Evaluation

To understand the mechanisms behind the success of `TransAM`, we analyze its behavior throughout an episode in each test environment. Figure 5 illustrates the relationship between the accuracy of the agent modeling and the cumulative reward. At the beginning of an episode, the model lacks context about the joint policy with which it is interacting, resulting in a policy embedding E_t^o that provides little additional information on the observation of the agent. However, as the episode progresses, the embeddings become more informative, improving agent modeling accuracy and leading to higher cumulative rewards.

This relationship is further evident when comparing how quickly the model converges on other agents' trajectories to its performance relative to other baselines. For example, in the overcooked environment (Figure 5 (c)), `TransAM` converges the fastest, aligning with its highest reward margin over the baselines (Figure 3(c)). In contrast, in the level-based foraging environment (Figure 5(d)), `TransAM` struggles to model agent behavior, which is correlated with its difficulty in outperforming other baselines (Figure 3(d)). These findings highlight the importance of designing adaptive agents that effectively model policies in environments with complex reward structures.

Table 1: **Model architecture ablation study results.** We test three variations of the model architecture on the cooperative navigation task and report the cumulative episodic return and the agent action reconstruction accuracy. The best results are shown in bold.

Method	Return	Action Accuracy
TransAM	$\mathbf{-48.76}$	$\mathbf{85.72}$
TransAM-*pool*	-49.37	61.67
TransAM-*fuse*	-48.94	78.68
TransAM-*im*	-49.93	72.08

5.4 Model Architecture Ablation Study

We analyze three ablated variants of `TransAM` in the cooperative navigation environment to evaluate the impact of its key architectural components: multimodal embeddings, embedding aggregation, and auxiliary training task. We assess their effects on cumulative episodic reward and agent action reconstruction accuracy.

- `TransAM`-*fuse*: Concatenates the rewards, actions, and observations of the controlled agent into a single fused token embedding, rather than embedding the tokens separately for each modality.
- `TransAM`-*pool*: Uses average pooling to merge all trajectory embeddings instead of relying on the most recent embedding.
- `TransAM`-*im*: Employs conditional imitation learning as the decoder, predicting only agent actions rather than both observations and actions.

The results of this analysis are presented in Table 1.

First, we determine whether our local trajectory representation is beneficial by comparing it against `TransAM`-*fuse*. This design achieves comparable returns; however, it suffers in agent modeling tasks. This suggests that learning token mappings for each modality is beneficial for agent modeling. Next, we consider the approach of pooling trajectory embeddings using `TransAM`-*pool* as opposed to using the most recent embedding vectors to condition the controlled agent's policy. We observe that while this method incorporates information from the entire trajectory, it leads to poor performance for both episodic returns and action reconstruction accuracy. This is because only recent transitions contribute to the identification of specific policies of the joint agent. Finally, we test whether the conditional imitation learning decoder in `TransAM`-*im* provides a benefit over decoding both the observations and actions of the agent. This results in the worst average returns and the second worst agent modeling accuracy. This implies that learning to reconstruct both the other agent observations and actions is beneficial to agent modeling and adapting to various joint agent policies. This is confirmed by the fact that LIAM and `TransAM` consistently achieve top-average episodic returns.

6 Conclusion and Future Work

In this paper, we introduced `TransAM`, a transformer-based agent modeling architecture that operates without access to other agents' information at execution time, ensuring full decentralization of the controlled agent. Using a transformer, `TransAM` effectively extracts and utilizes features from the controlled agent's episodic trajectory. We demonstrated its effectiveness across multiple environments, including Predator-Prey and Cooperative Navigation from the multi-agent particle environments, as well as Overcooked and Level-Based Foraging.

For future work, we aim to investigate the scalability of agent modeling techniques in larger multi-agent systems. Additionally, we seek to explore recursive reasoning domains, where agents must model others while accounting for the fact that their opponents are also performing agent modeling.

References

Pranav Agarwal, Aamer Abdul Rahman, Pierre-Luc St-Charles, Simon J. D. Prince, and Samira Ebrahimi Kahou. Transformers in reinforcement learning: A survey. *CoRR*, abs/2307.05979, 2023. DOI: 10.48550/ARXIV.2307.05979. URL https://doi.org/10.48550/arXiv.2307.05979.

Stefano V. Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artif. Intell.*, 258:66–95, 2018. DOI: 10.1016/J.ARTINT.2018.01.002. URL https://doi.org/10.1016/j.artint.2018.01.002.

Wendelin Boehmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 980–991. PMLR, 2020. URL http://proceedings.mlr.press/v119/boehmer20a.html.

Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1): 427–438, 2012.

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. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 15084–15097, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*, 1:1597–1607, 7 2020. URL http://proceedings.mlr.press/v119/chen20j/chen20j.pdf.

Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. DOI: 10.18653/V1/N19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Aditya Grover, Maruan Al-Shedivat, Jayesh K. Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1797–1806. PMLR, 2018. URL http://proceedings.mlr.press/v80/grover18a.html.

Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In Deborah L. McGuinness and George Ferguson (eds.), *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pp. 709–715. AAAI Press / The MIT Press, 2004. URL http://www.aaai.org/Library/AAAI/2004/aaai04-112.php.

He He and Jordan L. Boyd-Graber. Opponent modeling in deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1804–1813. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/he16.html.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. DOI: 10.1162/NECO.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Yuheng Jing, Kai Li, Bingyun Liu, Yifan Zang, Haobo Fu, QIANG FU, Junliang Xing, and Jian Cheng. Towards offline opponent modeling with in-context learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=2SwHngthig.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6379–6390, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html.

Wenhao Ma, Yu-Cheng Chang, Jie Yang, Yu-Kai Wang, and Chin-Teng Lin. Contrastive learning-based agent modeling for deep reinforcement learning. *CoRR*, abs/2401.00132, 2024. DOI: 10.48550/ARXIV.2401.00132. URL https://doi.org/10.48550/arXiv.2401.00132.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/mniha16.html.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere. Game theory and multi-agent reinforcement learning. *Reinforcement Learning: State-of-the-Art*, pp. 441–470, 2012.

Georgios Papoudakis, Filippos Christianos, and Stefano V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. In Marc'Aurelio Ranzato, Alina

Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 19210–19222, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/a03caec56cd82478bf197475b48c05f9-Abstract.html.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Çaglar Gülçehre, Siddhant M. Jayakumar, Max Jaderberg, Raphaël Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7487–7498. PMLR, 2020. URL http://proceedings.mlr.press/v119/parisotto20a.html.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI*, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4254–4263. PMLR, 2018. URL http://proceedings.mlr.press/v80/raileanu18a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/69413f87e5a34897cd010ca698097d0a-Abstract-Conference.html.

Marco A Wiering et al. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pp. 1151–1158, 2000.

Dong Xing, Pengjie Gu, Qian Zheng, Xinrun Wang, Shanqi Liu, Longtao Zheng, Bo An, and Gang Pan. Controlling type confounding in ad hoc teamwork with instance-wise teammate feedback rectification. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38272–38285. PMLR, 2023. URL https://proceedings.mlr.press/v202/xing23a.html.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre M. Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9c1535a02f0ce079433344e14d910597-Abstract-Datasets_and_Benchmarks.html.

Ziqian Zhang, Lei Yuan, Lihe Li, Ke Xue, Chengxing Jia, Cong Guan, Chao Qian, and Yang Yu. Fast teammate adaptation in the presence of sudden policy change. In Robin J. Evans and Ilya Shpitser (eds.), *Uncertainty in Artificial Intelligence, UAI 2023, July 31 - 4 August 2023, Pittsburgh, PA, USA*, volume 216 of *Proceedings of Machine Learning Research*, pp. 2465–2476. PMLR, 2023. URL https://proceedings.mlr.press/v216/zhang23a.html.